Towards a Virtual Domain based Authentication Solution for the MapReduce Application

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2018

By
Ibrahim E. Lahmer
School of Computer Science
## Contents

Contents ................................................................................................................................. 3  
Abstract ................................................................................................................................. 11  
Declaration ............................................................................................................................. 13  
Copyright ............................................................................................................................... 13  
Dedication ............................................................................................................................... 14  
Acknowledgement ................................................................................................................ 15  
Definitions ............................................................................................................................. 16  
Glossary of Terms and Acronyms ....................................................................................... 16  
Chapter 1 ................................................................................................................................ 19  
1. Introduction ....................................................................................................................... 19  
   1.1. Distributed Computing .............................................................................................. 19  
   1.2. Research Motivations and Challenges ..................................................................... 20  
   1.3. Research Aim and Objectives ................................................................................ 21  
   1.4. Research Methodology ............................................................................................ 22  
       1.4.1. Literature Review ............................................................................................ 22  
       1.4.2. Architecture and Protocol Designs ................................................................ 22  
       1.4.3. Analysis and Evaluation ................................................................................ 23  
   1.5. Achievement and Novel Contributions ................................................................... 24  
   1.6. Thesis Structure ........................................................................................................ 26  
Chapter 2 ................................................................................................................................ 29  
2. MapReduce: Background and Model Abstraction ........................................................... 29  
   2.1. Chapter Introduction ............................................................................................... 29  
   2.2. MapReduce Overview ............................................................................................. 29  
   2.3. MapReduce Architecture ....................................................................................... 31  
       2.3.1. Physical Components .................................................................................... 32  
       2.3.2. Logical Components ..................................................................................... 33  
   2.4. Executing Jobs using MapReduce: an Example Scenario ....................................... 34  
   2.5. A Generic MapReduce Computational (GMC) Model ........................................... 36  
   2.6. MapReduce Deployment in Cloud .......................................................................... 39  
   2.7. Chapter Summery .................................................................................................... 41  
Chapter 3 ................................................................................................................................ 42  
3. MapReduce: Observations and Security Analysis ........................................................... 42  
   3.1. Chapter Introduction ............................................................................................... 42
5. A Novel Virtual Domain based Authentication Framework (VDAF) ........................................ 81
   5.1. Chapter Introduction ......................................................... 81
   5.2. A High-Level Analysis of an MR-Job Execution on GMC Model .................. 82
   5.3. A Layered Approach to Authentication ................................ 85
   5.4. MR Layered Authentication Model ...................................... 86
   5.5. MR-Job Domain Authentication Layer: VDAF .......................... 88
6. A Novel Protocol for Clients Identification ................................................................. 123
   6.1. Chapter Introduction ......................................................................................... 123
   6.2. Design preliminaries ...................................................................................... 124
       6.2.1. Trust Model .............................................................................................. 124
       6.2.2. Requirements ........................................................................................... 127
       6.2.3. Additional Assumptions ........................................................................... 129
       6.2.4. Notations .................................................................................................. 129
   6.3. CPCrE Protocol Design: an Overview ............................................................. 131
   6.4. MR-Client’s Primary Tokens and Key Structure Design .................................... 135
       6.4.1. Primary Authentication and Verification Tokens ........................................ 135
       6.4.2. Key Structure ............................................................................................ 139
   6.5. CPCrE Protocol in Details ............................................................................... 143
   6.6. Security Analysis and Verification ................................................................... 152

5.6. VDAF Design Preliminary ................................................................................... 89
   5.6.1. Threat model ............................................................................................... 89
   5.6.2. Assumptions ............................................................................................... 90
   5.6.3. Requirements ............................................................................................. 91
   5.6.4. Notations ................................................................................................... 93
   5.6.5. Evaluation Metrics .................................................................................... 94
5.7. VDAF Design: Ideas and Measures .................................................................... 95
5.8. VDAF Architecture Design ................................................................................ 103
   5.8.1. Client identification .................................................................................... 104
   5.8.2. MR-Client Authentication and MR-Job Components Identification .......... 105
   5.8.3. MR-Job Components Authentication ............................................................. 108
   5.8.4. MR-Data Authentication ........................................................................... 111
5.9. Lightweight VDAF Authentication Protocol Suite ............................................... 112
5.10. VDAF Cryptographic Building Blocks ............................................................... 114
    5.10.1. Symmetric Key Cryptosystem .................................................................. 114
    5.10.2. Hashed Message Authentication Code (HMAC) ........................................ 115
    5.10.3. Asymmetric Key Cryptosystem .................................................................. 117
    5.10.4. Digital Signature ..................................................................................... 118
    5.10.5. Public Key Infrastructure (PKI) ................................................................. 119
5.11. Chapter Summery ............................................................................................... 121

Chapter 6 ....................................................................................................................... 123
7. A Novel Set of Protocols for MR-Clients Authentication and MR-Job Components Identification .......................................................... 183

7.1. Chapter Introduction ................................................................................................................. 183

7.2. Design preliminaries ................................................................................................................. 184

7.2.1. Requirements ....................................................................................................................... 184

7.2.2. Notations ............................................................................................................................... 187

7.3. MR-CAuthN Protocols Design for MR-Client Authentication: an Overview ....................... 188

7.3.1. U-AC_AuthN Protocol .......................................................................................................... 190

7.3.2. C-RM_AuthN Protocol ......................................................................................................... 192

7.3.3. C-NN_AuthN Protocol ......................................................................................................... 196

7.4. MR-Client’s Secondary Tokens and Key Structure Design .................................................... 200

7.4.1. Secondary Authentication and Verification Tokens Format ................................................ 200

7.4.2. Key Structure ...................................................................................................................... 204

7.5. MR-CAuthN Protocols in Details ........................................................................................... 206

7.5.1. U-AC_AuthN Protocol .......................................................................................................... 206

7.5.2. C-RM_AuthN Protocol ......................................................................................................... 208

7.5.3. C-NN_AuthN Protocol ......................................................................................................... 216

7.6. JCPCrE Protocol Design for MR-Job Components Identification: an Overview .................... 227

7.7. MR-Job Component’s Primary Tokens and Key Structure Design ........................................ 230

7.7.1. Primary Authentication and Verification Tokens ................................................................. 230

7.7.2. Key Structure ...................................................................................................................... 232

7.8. JCPCrE Protocol in Details ..................................................................................................... 233

7.9. Security Analysis and Verification ........................................................................................ 238

7.9.1. Informal Analysis ............................................................................................................... 239

7.9.2. Protocol Modelling and Formal Verification Results ........................................................ 248

7.10. Performance Evaluation ....................................................................................................... 250
List of Tables

Table 4.1 Related works versus MR authentication requirements ...........................................70
Table 5.1 The secret keys used for MR-Clients authentication .............................................98
Table 5.2 The secret keys used for MR-Job Components authentication ............................. 99
Table 5.3 The MR components and exchanged messages involved in the mutual authentications between MR-Job Components and the DFS cluster .............................................. 109
Table 5.4 LVAP suite ........................................................................................................ 113
Table 6.1 The contents of a client payment-attribute certificate ........................................ 127
Table 6.2 The contents of the primary authentication tokens of an MR-Client ....................... 136
Table 6.3 The contents of the primary verification tokens of an MR-Client .......................... 138
Table 6.4 A comparison of the computational cost of the client identification ..................... 168
Table 7.1 The contents of the secondary authentication tokens of an MR-Client ................. 201
Table 7.2 The contents of the secondary verification tokens of an MR-Client ....................... 203
Table 7.3 The contents of the primary authentication and verification tokens of an MR-Job Component .................................................................................................................. 231
Table 7.4 A comparison of the computational cost of the MR-Client authentication ............ 255

List of Figures

Figure 1.1 Thesis structure ...................................................................................................28
Figure 2.1 An overview of a classic MR implementation .....................................................30
Figure 2.2 MR architecture overview ................................................................................32
Figure 2.3 Tables: examples of input data files ..................................................................35
Figure 2.4 Simplified example of data execution flow in MR application ......................... 36
Figure 2.5 MR-Job execution (i.e. job work-flow) in GMC model .....................................39
Figure 2.6 Hosting MR components using virtualisation in a cloud ................................. 41
Figure 4.1 Malley method: messages exchanges in the MR application .............................. 61
Figure 4.2 Authentication process of the Somu method .................................................... 63
Figure 4.3 Registration and authentication processes of the Rubika method ..................... 65
Figure 4.4 The Wei method: exchanged messages between a Task Tracker and the Job Tracker ... 67
Figure 4.5 OpenID Connect standard: one user with different identities ......................... 75
Figure 4.6 Creating proxy certificate .................................................................................. 76
Figure 4.7 A delegation process using a proxy credential .................................................. 77
Figure 5.1 MR layered authentication model (A layered approach to authentication) ............ 87
Figure 5.2 The key structure of the authentication credential of an MR-Client ................. 98
Figure 5.3 The key structure of the authentication credential of an MR-Job Component ....... 99
Figure 5.4 A key wrap-and-swap operation for mutual authentication ............................. 102
Figure 5.5 An overview of the VDAF architecture ............................................................. 104
Figure 5.6 Client identification ......................................................................................... 105
Figure 5.7 MR-Client authentication ................................................................................ 107
Figure 5.8 MR-Job Components identification .................................................................. 108
Figure 5.9 MR-Job Component authentication: first process ............................................ 110
Figure 5.10 MR-Job Component authentication: second process ....................................... 110
Figure 5.11 The Original-Data authentication ................................................................... 111
Figure 7.16 The five basic roles and their sequences of actions to model C-RM and C-NN_AuthN protocols.................................................................................................................. 249
Figure 7.17 AVISPA validation results ................................................................................................................. 249
Figure 7.18 The computational cost of MR-CAuthN protocols ................................................................. 258
Figure 7.19 The computational cost of the JCPCrE protocol ........................................................................ 259
Figure 7.20 The computational cost introduced by the authentication components of KRB protocol .................................................................................................................................................. 260
Figure 7.21 Communication Cost: the length of the MR-CAuthN protocols messages .................. 261
Figure 7.22 Communication Cost: the length of the JCPCrE protocol messages ............................ 262
Figure 7.23 Total communication overhead of the C-RM, C-NN_AuthN protocols vs. the KRB protocol ................................................................................................................................................. 263
Figure 7.24 Total communication overhead of the JCPCrE protocol vs. the KRB protocol ........... 263
Figure 7.25 The theoretical value of the PET for the C-RM_AuthN protocol .................................. 265
Figure 7.26 The PET of the C-RM_AuthN protocol in detail ........................................................................ 267
Figure 7.27 Modelling C-RM_AuthN protocol ............................................................................................ 268
Figure 7.28 Network Models for MR-CAuthN and KRB protocols ......................................................... 270
Figure 7.29 Comparing the theoretical and network models results of the PET of both the C-RM_AuthN protocol and the KRB protocol .............................................................................................................. 271
Figure 7.30 PET of C-RM, C-NN_AuthN and KRB protocols vs. the number of concurrent requests ............................................................................................................................................ 273
Figure 7.31 The communication rounds and their communication overheads cross the WAN for the C-RM, C-NN, and KRB protocol ........................................................................................................... 273
Figure 7.32 Turning points for the RM, NN, DN and AS ................................................................................. 277
Figure 7.33 Turning points for the AS, TGS and AP .................................................................................. 278
Abstract

Distributed computing frameworks are used to harvest distributed resources to process big data more efficiently. As the popularity of the frameworks increases, it is anticipated that some of the data they are used to process are private or sensitive, therefore require more stringent security protections. As authentication is the first-line of defence in any computing system, providing an effective and efficient authentication service to safeguard data processed or used in a distributed computing framework is paramount to the wide scale application of the framework, and this is still an open issue. To investigate and address this open issue, the author of this thesis has chosen one of the most widely used distributed computing framework, the MapReduce (MR) application framework (or MR application for short), and designed an effective and efficient authentication solution for this framework.

This thesis has examined a use-case scenario of MR computation and presented a generic computational model of MR to capture the characteristics of MR application framework, the main components, process and interactions of a job computation (i.e. job submission and execution). Using this model, the thesis analyses the threats related to the job submission and execution. From the threat analysis a set of requirements for an effective and efficient authentication solution is specified. Based on these requirements, the thesis critically analyses the state-of-the-art authentication solutions used or proposed for the MR application identifying limitations and gaps for improvement. Existing solutions do not adequately capture the characteristics of MR being deployed in a distributed and resource-sharing environment, they largely use centralised approach to authentication and based on a password single factor authentication, and most of them assume that the clients are already identified locally to the MR application and the communication among MR components is through a trusted network.

To address these limitations and gaps, the thesis has made the following contributions. Firstly, a Generic MapReduce Computation (GMC) model is constructed. The main novelty of this model is that it classifies and captures two classes of MR components. One is MR components that are job-independent, and the other is MR components that are job-dependent. As the MR components are distributed and in a resource-sharing environment, their interactions should all be authenticated to safeguard the job resources (i.e. data). Facilitating authentication of these interactions effectively in such an environment is a complex task. To address this complex task, the second novel contribution has been made, that is the proposal of a novel authentication model, called MR Layered Authentication Model (MR-LAM). It uses a layered approach to authentication. The model consists of two layers. The first is the MR-Infrastructure domain authentication layer and it is responsible for a job-independent MR components authentication. The second is the MR-Job domain authentication layer and it is responsible for job-dependent MR components authentication. The third novel contribution of this thesis, our proposed solution for MR-Job domain authentication layer, is the design of a Virtual Domain based Authentication Framework (VDAF) to secure, in term of authentication, the job-resource access in the distributed and resource-sharing environment. At the centre of the VDAF lies our novel authentication
method, called Password and Token-based Multi-factor Multi-point Authentication (PT2M-AuthN) method. In this method, two main ideas are used; the principle of separation of duty-and-credential and the key wrap-and-swap operation to support mutual authentication for both job submission and execution.

To implement the VDAF function and this novel authentication method, four sets of protocols that are collectively referred to as the Lightweight VDAF Authentication Protocol (LVAP) suite, are proposed. The first set consists of MR-Client Primary Credential Establishment (CPCrE) protocol, the second set consists of MR-Job Components Primary Credentials Establishment (JCPCrE) and MR-Clients Authentication (MR-CAuthN) protocols, the third and fourth sets consist of MR-Job Components Authentication (MR-JCAuthN) protocols and MR-Data Authentication (MR-DAuthN) protocols, respectively. The effectiveness (security) and the efficiency of the design have been evaluated. The security evaluation is done by using an informal security analysis and formal security verification using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. The performance evaluation is done by using theoretical and experimental methods, and the latter uses the Riverbed Modeller simulation tool. The evaluation results have been compared with those of related work. The comparison results show that our authentication method provides a stronger level of protection as a result of using two-factor authentication and key wrap-and-swap operation that provide mutual authentication. With regard to performance, the comparison results show that when our authentication method is used, our protocol execution time (PET) is 20% shorter than that of when the most related authentication method is used.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.library.manchester.ac.uk/about/regulations/) and in The University’s policy on Presentation of Theses.
Dedication

In the name of Allah, the most Gracious, the most Merciful

{-Does the human being think that he will be left neglected? Was he not created from semen emitted? Then he became a clot; then shaped and fashioned in due proportion, and made of him a couple (two sexes), male and female. Does the one who did this would not be able to create the human being again after death?-} (36-40) [Al-Qiyaama, the Holly Quran].

*To Mum and Dad who scarify a lot of their life to see me studying and seeking knowledge,*

*and I ask Allah, the creator, have mercy upon them in this life and the life after death.*
Acknowledgement

I thank Allah for all his blessings; without his guidance and success I would not have completed and presented my PhD thesis.

I would thank and appreciate my supervisor Dr. Ning Zhang for her valuable advice throughout my PhD study, including her comments and feedback from the first day of starting this research till the final submission of the thesis.

I would like to express my sincere appreciation and gratitude to my mother who has spent a lot of her time waiting me to come back and seeing the moment of me getting a PhD degree, to my father who passed away while I was doing this research and he was waiting to see me and be proud in me coming this far, and to my wife for her patience and understanding of being busy doing this work until the end of my PhD program. I am also thankful for my sisters, brothers and my relatives for being always supportive. Without their support, this PhD research would have been harder to complete.

Last but not least, many thanks for my friends and colleagues in Tripoli and in Manchester who were always happy to listen and share any personal and/or professional difficulty with me.

This research was sponsored by the Ministry of Higher Education and Scientific Research of Libya and partially supported by National Oil Corporation Libya (NOC-Libya).
## Definitions

### Glossary of Terms and Acronyms

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definition/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR-Client</td>
<td>It is a client that is identified to (i.e. registered with) the MR application and issued with authentication credential to authenticate himself later on to the MR application, so that the client can be part of his job domain.</td>
</tr>
<tr>
<td>MR-Job execution (or a job work-flow)</td>
<td>It refers to the steps of the entire job execution cycle from the moment when an MR-Client sends a new job request to the MR application till the moment when the job is successfully completed. It composes of both job submission and execution.</td>
</tr>
<tr>
<td>Job submission</td>
<td>It refers to a set of procedures and interactions between software components to submit a job to the MR application.</td>
</tr>
<tr>
<td>Job execution</td>
<td>It refers to a set of procedures and interactions between software components to execute a job submitted to the MR application.</td>
</tr>
<tr>
<td>Server node</td>
<td>It refers to a system-software that provides a platform for other software components.</td>
</tr>
<tr>
<td>Cluster</td>
<td>It refers to a set of connected server nodes that communicate and work with each other to make a set of services available to clients.</td>
</tr>
<tr>
<td>Identification information</td>
<td>It refers to a cryptographic information (i.e. piece of data) that is related to define and register a client to MR application, so that the client could be identified to MR application as an MR-Client by issuing him an authentication credential.</td>
</tr>
<tr>
<td>Authentication information</td>
<td>It refers to a cryptographic information (i.e. piece of data) that is used to form part or all of an authentication credential and/or a credential verification token of an entity.</td>
</tr>
<tr>
<td>Authenticator</td>
<td>It refers to a piece of data sent with an authentication message to help the verifier to (i) assure the claimant’s knowledge of the secret key/s of an authentication credential, and (ii) prevent message replay.</td>
</tr>
<tr>
<td>MR-AuthN Component</td>
<td>A portion of a program/software that carries out authentication procedures of an entity of the MR application, so called an MR-Authentication Component (MR-AuthN Component).</td>
</tr>
<tr>
<td>A stego-file</td>
<td>It also referred to as a stego-object, it is an image file that is used to conceal an authentication information.</td>
</tr>
</tbody>
</table>
Approved by A cryptographic information is referred to be approved by an entity or an MR-AuthN Component of the entity when the MR-AuthN Component of an entity either encrypts or hashes the cryptographic information by an encryption or a hash function algorithm using a secret key known to the entity.

Approved to An entity or an MR-AuthN Component is referred to be approved to issue or generate an authentication credential when it receives the cryptographic information that is required to generate such credential.

Generate an authentication token Generating an authentication token by an MR-AuthN Component refers to that the MR-AuthN Component has received the cryptographic information which is issued by another MR-AuthN Component/s, and it issues the other (remaining) cryptographic information that are required to form the entire token.

Authentication factor (or secret) It is a secret authentication information, such as a secret key, linked to specific entity, and it is used to verify the entity’s identity.

Job-resource request It refers to a request that is initiated by an entity in an MR-Job execution to access a resource of an MR-Client’s job, e.g. a request for the input data of an MR-Client’s job.

<table>
<thead>
<tr>
<th>Acronyms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
<td>Authentication Server</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>C</td>
<td>Client</td>
</tr>
<tr>
<td>MR</td>
<td>MapReduce</td>
</tr>
<tr>
<td>DP</td>
<td>Data Processing</td>
</tr>
<tr>
<td>PF</td>
<td>Processing Framework</td>
</tr>
<tr>
<td>DFS</td>
<td>Distributed File System</td>
</tr>
<tr>
<td>RM</td>
<td>Resource Manager</td>
</tr>
<tr>
<td>NN</td>
<td>Name Node</td>
</tr>
<tr>
<td>DN</td>
<td>Data Node</td>
</tr>
<tr>
<td>JT</td>
<td>Job Tracker</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TT</td>
<td>Task Tracker</td>
</tr>
<tr>
<td>MT</td>
<td>Map Task</td>
</tr>
<tr>
<td>RT</td>
<td>Reduce Task</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>VLAN</td>
<td>A Virtual LAN</td>
</tr>
<tr>
<td>WAN</td>
<td>Wired Area Network</td>
</tr>
<tr>
<td>JC</td>
<td>Job Component</td>
</tr>
<tr>
<td>JD</td>
<td>Job Domain</td>
</tr>
<tr>
<td>ID</td>
<td>Infrastructure Domain</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>TGT</td>
<td>Ticket Granting Ticket</td>
</tr>
<tr>
<td>TGS</td>
<td>Ticket Granting Service</td>
</tr>
<tr>
<td>GMC model</td>
<td>Generic MR Computational model</td>
</tr>
<tr>
<td>MR-LAM</td>
<td>MapReduce Layered Authentication Model</td>
</tr>
<tr>
<td>VDAF</td>
<td>Virtual Domain based Authentication Framework</td>
</tr>
<tr>
<td>LVAP suite</td>
<td>Lightweight VDAF Authentication Protocols suite</td>
</tr>
<tr>
<td>AuthN</td>
<td>Authentication</td>
</tr>
<tr>
<td>U-AC_AuthN</td>
<td>User to MR Application Client Authentication</td>
</tr>
<tr>
<td>C-RM_AuthN</td>
<td>MR-Client to Resource Manager Authentication</td>
</tr>
<tr>
<td>C-NN_AuthN</td>
<td>MR-Client to Name Node Authentication</td>
</tr>
<tr>
<td>JT-NN_AuthN</td>
<td>Job Tracker to Name Node Authentication</td>
</tr>
<tr>
<td>TT-NN_AuthN</td>
<td>Task Tracker to Name Node Authentication</td>
</tr>
<tr>
<td>RT-NN_AuthN</td>
<td>Reduce Task to Name Node Authentication</td>
</tr>
<tr>
<td>RT-TT_AuthN</td>
<td>Reduce Task to Task Tracker Authentication</td>
</tr>
<tr>
<td>CPCrE</td>
<td>MR-Client Primary Credential Establishment</td>
</tr>
<tr>
<td>JCPCrE</td>
<td>MR-Job Components Primary Credentials Establishment</td>
</tr>
</tbody>
</table>
Chapter 1

1. Introduction

1.1. Distributed Computing

Distributed computing is a setup in which several computing nodes participate in processing a large volume of data, i.e. Big Data [1] [2]. Typically organisations that have a Big Data may have access to a wealth of information. According to IBM some organisations that process a Big Data have problem to get the information they require out of it within an acceptable timeframe. Processing such data (i.e. job execution) is frequently carried out efficiently and in an acceptable timeframe using a distributed computing framework [3]. A distributed computing framework consists of multiple software and server components distributed on multiple nodes and connected by a communication network. These components interact to accomplish a job execution. The job execution is typically done by assigning multiple tasks to handle the work of the job. These tasks run by the software components that are hosted on a set of server nodes called slave nodes. These server nodes are managed by master server nodes to which a user can submit his job. All of these components appear to their users as a unified system. The distributed computing frameworks have been developed aiming for increasing performance, sharing resources, and providing availability [2].

Over the years, a number of distributed computing frameworks have been proposed. Examples include MapReduce [4], Dryad [5], MPAPI [6] and others [7]. The MapReduce (MR) framework is one of the most widely used as well as the best-documented one. The MR framework is particularly designed to process Big Data, so, different from traditional distributed frameworks which move data closer to the tasks that process the data [1], the MR framework moves tasks closer to data. This is because for data-intensive processing (i.e to process a Big Data), is not easy to move the data around [8] [4]. The MR framework is not only solved this problem of the traditional distributed frameworks, but it also process Big Data on computing clusters efficiently. The MR framework processes it in a way that is tolerant of hardware failures during the job execution. Other key strengths of the MR framework are the high degree of simplicity and parallelism of its programming model (i.e. MR model) along with its applicability to a large variety of application domains. The MR model is regarded as an adequate programming model for large-scale data-intensive
applications. The implementation of the MR model, i.e. the MR framework, is being used increasingly in applications such as those of data mining and scientific computation applications [9] [10], so-called the MR application framework (hereafter referred to as ‘the MR application’ for short). The MR application has been realised to be a highly successful distributed computing framework. Thus, it has created a lot of attention in both the industrial and academia domains. Due to its success, it has been supported by many big organisations in their Big Data platforms, such as Microsoft, Oracle, IBM, Facebook, eBay, and Yahoo [3] [11]. Also, several successful organisations for big data analytics solutions, such as MapR [12] and Cloudera [13], have included the MR application to build their solutions on it [14]. All of these had motivated us to choose the MR application for our research work in order to investigate and address the open issue of the authentication in such distributed computing framework.

1.2. Research Motivations and Challenges

Security in the MR application is a complex and challenging. One of the most important security properties required in this framework is authentication. Providing the authentication for such distributed computing framework is a challenge task as it has a number of special features or characteristics, namely:

1. The execution of a job submitted by a user to the MR application involves the use of multiple software components, and the user has neither control nor interactions with these components.

2. Multiple software components serving single or multiple jobs from single or multiple users may be hosted in a single or multiple server nodes.

3. Different server nodes may be managed in different administrative domains and/or located in different geographic locations.

4. During a job execution, different software components hosted by different server nodes will interact via message passing (i.e. communication) over networks, and some of the interactions involves resource (data) access.

5. Data used (i.e. input data) or produced (i.e. output data) by different jobs are typically stored at different server nodes.
These features indicate that this is a distributed and resource-sharing environment, and there are many entry points where identity theft or impersonation related threats or attacks may be launched. In other words, just identifying and authenticating a user when a job is being submitted is far from enough to protect against impersonation attacks in this environment. Authentication control should be enforced at every point where there is a request for data or other resource access, or an interaction that may lead to such an access. In addition, as the MR resources used to support a job execution in the MR application are assigned dynamically, the authentication service should cover mutual authentication between different software components, between different server components (i.e. inter-components authentication level) and between the client and the MR application (i.e. gate authentication level). Furthermore, it is also important to recognise that the authentication service to be provided should be strong while at the same time the overhead introduced should be as low as possible.

1.3. Research Aim and Objectives

The aim of this research is to design an effective and efficient authentication solution for the MR application that captures the characteristics of the distributed computing frameworks. This aim is supported by the following objectives.

1. Study and investigate the architectural components of the MapReduce, and how it is hosted and operates to understand the MR application.
2. Build a generic MapReduce abstraction model to capture the key features and characteristics of the MR application.
3. Investigate and analyse security threats in the context of the distributed and shared environment, and understand their implications on the provision of the model.
4. Specify a set of MR authentication requirements for designing an authentication service for MR application based on the analysis performed in (3).
5. Investigate and critically analyse the state-of-the-art authentication methods used and proposed for the MR application against the set of requirements specified in (4) to identify limitations in these authentication methods.
6. Design an authentication solution to support the job submission and execution in the MR application. The design is guided by the requirements specified in (4), and based on the outcome of the analysis performed in (3) and (5), advancing the state-of-art by bridging some of the gaps identified in the outcome.
7. Evaluate the security and the performance of the authentication method used to 
satisfy the MR authentication requirements specified in (4) and used in the 
authentication solution designed in (6).

1.4.  Research Methodology

1.4.1.  Literature Review

First task of this project was reading literature searching for a distributed computing 
framework that captures the most common characteristics in the distributed computing 
frameworks. It was discovered that the MR application was the dominant one and used 
mostly as a core of other distributed computing frameworks for Big Data. So the author 
started studying the related literature on the MR application by investigating and 
understanding the structure and the characteristics of the MR application. Based on this 
study, a Generic MR Computational (GMC) model is built. It captures the key features of 
the MR application and a threat analysis was conducted on this model. As a result of this 
analysis, it is understood that the effectiveness in mitigating many of the security threats in 
the MR application is closely dependent on the effectiveness of the authentication service. 
This has lead us to do more research investigating the authentication solutions proposed for 
the MR application. From this analysis, gaps were identified.

1.4.2.  Architecture and Protocol Designs

Following the literature review, a layered approach to authentication is proposed. Through 
the threat analysis and the distributed features of a job execution in the MR application it is 
realised that in the MR shared environment, (i) the issue or the task of authentication is a 
complex task, (ii) there are three levels of authentication provisions which are required; gate 
level, inter-components level, and data level, and (iii) the functionality of the involved 
components and the interactions (i.e. authentication) among them are different based on the 
component role. To recognise these authentication provisions and tackle the complexity of 
such task, there is a need to provide the authentication tasks in the MR application in a 
systematic and module manner. This has led to our novel idea of layered approach to 
authentication for the MR application, so called –MR Layered Authentication Model (MR-
LAM). MR-LAM contains two layers of authentication, called infrastructure domain layer 
and job domain layer. The MR-LAM takes into account all the authentication requirements
of MR application. Based on the MR-LAM, in the job domain layer – the core component of MR-LAM, a Virtual Domain based Authentication Framework (VDAF) is designed. VDAF addresses the following five aspects of authentication needs in the entire job execution cycle from the moment when the job is submitted to the moment when the job is successfully completed: (1) client identification, (2) MR-Client authentication, (3) MR components, assigned to a specific job domain, identification (i.e. MR-Job Components identification), (4) MR-Job Components authentication and (5) MR-Data authentication. Then the authentication method, ideas and measures used in the VDAF design are defined. To facilitate the authentication functions by implementing our ideas in this framework, four sets of protocols called a Lightweight VDAF Authentication Protocols (LVAP) suite is proposed, and they are, respectively, for remote client identification (set-1), for remote MR-Client authentication and MR-Job Component identification (set-2), and for MR-Job Components authentication (set-3) and MR-Data authentication (set-4). Because of that (i) the time limit (ii) the ideas and measures used in set-3 protocols are very similar to those used in set-2 protocols, and (iii) set-4 protocols are outside the scope of this thesis, at the conclusion of this stage the author has only introduced a detailed design and evaluation of the first two sets of LVAP suite, i.e. set-1 and set-2.

1.4.3. Analysis and Evaluation

On completion of the design stage, this stage of our research was to analyse the security of the protocols designed in set-1 and set-2 of the LVAP suite and evaluate their performance. The security analysis is carried out using two methods: informal analysis and formal verification. The former was conducted to analyse the protocols against the security requirements specified for each set of the protocols. The formal verification was then applied to test the correctness and provide systematic verification of the designed protocols. The formal verification was carried out using AVISPA, i.e. Automated Validation of Internet Security Protocols and Applications verification tool. Once the security analysis is completed, the performance evaluation of the protocols is achieved. The protocols are evaluated in terms of their computational and communication overheads along with the protocol execution time. To show the effectiveness of the new authentication method and the ideas used in the VDAF design, the results of the performance evaluation are compared with the most relevant work.
1.5. Achievement and Novel Contributions

The research work presented in this thesis has led to the following achievements and contributions:

1. A Generic MapReduce Computational (GMC) model is devised.
   - The GMC is an abstraction model that is constructed to capture the key features and properties of the most recent MR model implementation (i.e. MR application). This includes the architectural components of the MR application and the interaction among these components when submitting and executing a job, and the sequence and purpose of these interactions.
   - Based on the GMC model, security threats which are closely related to the MR-Job execution are analysed and the implications of such threats on the provision of the GMC model are understood.
   - A set of MR authentication requirements has been specified.
   - A critical analysis of the authentication methods proposed or used for the MR application has been conducted against the specified requirements.

2. An MR Layered Authentication Model (MR-LAM) is proposed.
   - The MR-LAM is based on an idea of layered approach to authentication for the MR application. MR-LAM consists of two authentication layers, MR-Inf. (Infrastructure) domain authentication layer (Layer-1) and MR-Job domain authentication layer (Layer-2). It captures the complex task of authentication for the MR application in a distributed and resource-sharing environment.
   - A set of functional and security authentication requirements for the MR-Job domain authentication layer is specified.
   - A Virtual Domain based Authentication Framework (VDAF) is designed to meet the functional and security requirements of the MR-Job domain authentication layer. In the VDAF design, an authentication method called a Password and Token-based Multi-factor Multi-point Authentication (PT2M-AuthN) method is proposed. In this method, two main ideas are used, a principle of separation of duty-and-credential and a key wrap-and-swap operation to support mutual authentications. They are used to strengthen the authentication function of the VDAF and to carry out this function efficiently. In the implementation of the PT2M-AuthN method, we use (1) a distributed set of MR authentication components, (2) a symmetric key based authentication approach, and (3) multi-
factor authentication tokens. The authentication factors (i.e. secrets) of these tokens are supplied from different trusted entities during an MR-Job execution, and each interaction involving a job-resource request in the MR-Job execution is identified using one of these authentication tokens. These design decisions are based on the considerations that the protection level should be strengthened as much as possible while keeping the overheads introduced as low as possible and there should be no change imposed on the existing MapReduce architecture.

- A Lightweight VDAF Authentication Protocol (LVAP) suite is proposed. LVAP suite consists of four sets of protocols to facilitate the functionality of the architectural components of the VDAF. The first set consists of a single protocol that is used to perform client identification function. The second set consists of four protocols that are used to perform MR-Client authentication and MR-Job Components identification functions. The third set consists of four protocols that are used to perform MR-Job Components authentication function. The fourth set consists of three protocols that are used to perform MR-Data authentication function.

3. An MR-Client Primary Credential Establishment (CPCrE) protocol is designed and evaluated for the first set of LVAP suite. The CPCrE is designed for identifying remote clients to the MR application as MR-Clients without requiring any client to make a physical appearance. Each new remote client is identified securely as an MR-Client by issuing his primary authentication and verification tokens, and the client acquires the local and removable parts of his authentication credential. This is achieved using three cryptographic systems; symmetric key cryptosystem, asymmetric key cryptosystem, and public key infrastructure. This enables a remote MR-Client to identify each interaction of his job submission to the MR application, so that the MR application can verify the job submission is indeed from an authorised MR-Client.

4. MR-Client Authentication (MR-CAuthN) and MR-Job Components Primary Credentials Establishment (JCPCrE) protocols are designed and evaluated for the second set of the LVAP suite. The MR-CAuthN protocols incorporate three protocols named (i) User to the MR Application Client Authentication (U-AC/AuthN) protocol which enables a user login to the MR Application Client to be authenticated locally in the client machine, (ii) MR-Client to Resource Manager Authentication (C-RM_AuthN) protocol which enables a secure mutual
authentication to be achieved between the MR-Client and the Resource Manager, the Resource Manager that manages the computational resources of the MR application, and (iii) MR-Client to the Name Node Authentication (C-NN_AuthN) protocol which enables secure mutual authentications to be achieved between (1) the MR-Client and the Name Node which manages the storage resources of the MR application and (2) the MR-Client and the respective Data Node which stores the MR-Client’s job resources, e.g. the MR-Client data that is written to the storage resources during the job submission. The JCPCrE protocol is designed for identifying the MR components that to be assigned to the job submitted by the MR-Client. It enables (i) an authentication control to be imposed to all requests that require access to the MR-Client’s data during the job execution, and (ii) MR-Client to delegate his permission to access the data on his behalf by issuing and distributing the primary authentication and verification tokens of the MR components assigned to his job.

During the course of this research the following papers have been published.


1.6. Thesis Structure

The remainder of this thesis is organised as follows (the structure of the thesis is further illustrated in Figure 1.1).

Chapter 2 provides a background information about the MapReduce including MapReduce architecture and operation, and it introduces a generic abstraction model that captures the key features and characteristics of the MR application (MR model implementation).
Chapter 3, by considering the GMC Model, a number of observations from the GMC model is identified, and the threat analysis on the GMC Model has been discussed. Based on both the observations and the threat analysis, a number of security countermeasures are discussed, and the service of MR authentication is chosen for further investigation.

Chapter 4 reviews the state-of-art authentication methods proposed or used for the MR application. The chapter also presents what is considered to be missing (gaps) in the existing MR authentication approaches.

Chapter 5 presents a novel layered authentication solution to the MR application, named MR Layered Authentication Model (MR-LAM), discussing the design of the core component of this model; the Virtual Domain based Authentication Framework (VDAF). This includes the authentication method, ideas and measures used in the design, and the VDAF architecture covering its functionality and components. Then the chapter introduces four sets of protocols that collectively called Lightweight VDAF protocol (LVAP) suite facilitate the function of the VDAF. Finally in this chapter, the cryptographic algorithms or building blocks that to be used for LVAP suite are presented.

Chapter 6 presents the design, security analyses and performance evaluation of the CPCrE protocol.

Chapter 7 presents the design, security analyses and performance evaluation of the MR-CAuthN and JCPCrE protocols.

Chapter 8 concludes this thesis and suggests directions for future work/research.
Figure 1.1 Thesis structure.
Chapter 2

2. MapReduce: Background and Model Abstraction

2.1. Chapter Introduction

This chapter provides a background for the MapReduce architecture and its functionality as well as an abstraction model capturing the main features and properties of the MapReduce implementation. In details, Sections 2.2 presents an overview about the MapReduce and discusses the needs of the most recent MR model implementation. Section 2.3 describes its architectural components of such implementation. Section 2.4 gives an example of how MapReduce process a data of a job submitted by a client in a specific data execution flow, followed by a Generic MapReduce Computational (GMC) model in Section 2.5. Then, Section 2.6 describes briefly how the GMC components are hosed using a Virtualisation technique when they are run in a cloud. Finally, Section 2.7 summaries the chapter.

2.2. MapReduce Overview

Over the years, a number of programing models have been proposed. Examples include MapReduce (MR) [4], Dryad [5] and others. MR is one of the most widely used, and it is the current de facto paradigm for writing data-centric applications in both industry and academia. MR is inspired by the use of distributed functions in combination with the divide-and-conquer algorithm [15]. MR is a programing model that supports parallel processing and distributed resource sharing. It uses a set of distributed components that work collaboratively to process large volumes of data by executing jobs submitted by clients. A job execution is carried out in two distinctive phases: Map and Reduce. In the Map phase, a set of components, called Mappers (or Map Tasks), are used to process and convert one set of data (input) into another set (called intermediate result or intermediate data). This set of input data is broken into key-value pairs called tuples and stored and managed in a Distributed File System (DFS). In the Reduce phase, another set of components, called Reducers (or Reduce Tasks), combine and process the intermediate data to produce a smaller set of tuples called job result or output data. As indicated by the name, MapReduce, Map Tasks are always executed before the Reduce Tasks [3].
An earlier MR implementation (hereafter referred to as MR1) is shown in Figure 2.1. The figure shows that the Map and Reduce Tasks are executed on a set of MR cluster nodes, called MR slave nodes which run worker daemons. Each worker daemon has a limited number of predefined task slots, map and reduce slots. The worker daemons are managed by a master daemon run on an MR master node, called a Job Tracker. The Job Tracker is responsible for both managing the MR server nodes (i.e. cluster's resources) and scheduling and coordinating the execution of all MR jobs submitted by clients. It configures, runs and monitors each Map and Reduce Task. If a task fails, it allocates a new slot and rerun the task. After a task finishes, the Job Tracker frees up the MR resource (i.e. CPUs and RAMs of server nodes) by releasing the assigned slot, so it becomes available for other jobs.

![Figure 2.1 An overview of a classic MR implementation](image)

Though MR1 is powerful enough to be used for big data analysis, it has a number of limitations. First, the model is not scalable. As mentioned above, the Job Tracker runs on a single server node and performs several tasks (MR resource management, job and task scheduling and monitoring). It can be a performance bottleneck. The second limitation is its lack of availability. In MR1, the Job Tracker is a single point of failure. If the Job Tracker fails, all the jobs submitted must be restarted. The third limitation is related to MR resource utilization. As mentioned above, for each job submitted, MR1 uses a predefined number of map and reduce slots per MR server node. In cases where the slots assigned to Map Tasks are fully utilized but those assigned to Reduce Tasks are idle (or vice-versa), the resource (i.e. CPUs and RAMs) reserved for the Reduce Tasks cannot be easily released to the Map tasks.
These limitations have been addressed by de-coupling the tasks related to cluster resource management and the tasks related to the MR programming model. This is achieved by adding a new master component, i.e. Resource Manager (RM), into the most recent MR implementation, hereafter referred to as MR2. In MR2, RM takes over the task of cluster resource management from the Job Tracker. Each submitted job is allocated with a Job Tracker to manage it, and the allocated Job Tracker negotiates with the RM for MR resources as needed. As a result, if a Job Tracker fails, only the job managed by this Job Tracker will be affected. The resource utilization is optimized as no MR server nodes’ resources could sit idle. MR2 is more scalable. MR2 can run multiple jobs which belongs to different applications. Due to these reasons, it is regarded as a more adequate programming model for large-scale data-intensive applications. MR2 is being used increasingly in applications such as those of data mining and scientific computation applications.

Owing to the reasons explained above, in this research, we have chosen MR2 as the underlying distributed computing framework. We have built a generic computational model based on MR2, i.e. the Generic MapReduce Computational (GMC) model. The GMC model captures the interactions involved in requesting or accessing the data being processed in a job execution (i.e. job-resource). The following sections describe the architectural components of MR on which the GMC model is constructed, the characteristics and interactions of MR components, and how a job submitted by a client is executed in the GMC model.

2.3. MapReduce Architecture

MR architecture consists of a number of physical and logical components. These components can largely be classified into two main groups. Figure 2.2 summarises the components of these two groups. As shown in the figure, one group is the Distributed File System (DFS) cluster that is responsible for storing data of jobs submitted by clients, and the other group is the Data Processing (DP) cluster (also called Processing Framework (PF)) that is responsible for carrying out distributed computations for the executions of the jobs. The components of these two groups run on master and slave nodes [3] [4] [16]-[18].
2.3.1. Physical Components

MR physical components are the system-software components of MR that are typically used to host MR logical components. Depending on the roles played by these MR components, they may be client, master or slave. The corresponding nodes hosting the MR physical components are therefore referred to as user machines (or client nodes/hosts), master nodes and slave nodes.

1. **User Machines**: A user machine runs an MR-Application Client (also called an MR-Job Client), which largely performs two tasks; submitting a job to the MR master node and monitoring the job execution.

2. **Master nodes**: In Figure 2.2, master nodes are of the following types.

   a. **Resource Manager** (also called Central Cluster Manager Node): This MR component, i.e. MR server node, is responsible for (1) receiving jobs submitted by the clients, and (2) managing the jobs schedules and the cluster.
resources by tracking the server nodes in the cluster and the amount of free resources on these nodes. There is one such MR component per DP cluster.

b. *Name Node* (also called Catalogue Server Node or Metadata Node): This MR server node manages and maintains a set of MR slave nodes called Data Nodes, i.e. the Name Node contains a file system and metadata for all the directories and files stored in Data Nodes. It is usually part of DFS cluster and there is one per cluster.

3. **Slave nodes**: They are of three types, Job Tracker, Task Tracker, and Data Node.

   a. *Job Tracker*: A Job Tracker Node (also called Job/Application Master Node) runs a master daemon to initialize a job scheduled by the Resource Manager and to negotiate with the Resource Manager regarding the use of the DP cluster resources. Usually, there is more than one Job Trackers Node per DP cluster. However, as mentioned in the previous section, in the classic MR application [4], there is one Job Tracker Node, plays the role of the master node of the DP cluster.

   b. *Task Tracker*: A Task Tracker Node may host a single Mapper (i.e. Map Task), a single Reducer (i.e. Reduce Task) or both. Usually, there is more than one Task Tracker Node per cluster. As shown in Figure 2.2, each Task Tracker Node runs a worker daemon to execute multiple tasks.

   c. *Data Node*: This MR component is also part of the DFS and, sometimes, named as a Storage Server Node. It is the MR server node in which the actual data files are stored in units of data blocks and shared. Each data block has its own block ID, and for fault-tolerance consideration, it is replicated in a number of Data Nodes. A Data Node sends periodically a report to the Name Node listing the blocks it stores, as shown in Figure 2.2. There are typically multiple Data Nodes in the DFS cluster.

2.3.2. **Logical Components**

MR logical components present the other software components of MR (i.e. non system-software components of MR) and the MR data. These MR components run and hosted on the above mentioned MR physical components to accomplish the computational tasks of MR. These MR logical components are explained below.
1. **Master daemon** (or Job/Application Master) is executed in the Job Tracker Node. It is responsible for monitoring and tracking the progress of each individual job executions.

2. **Worker daemon** is executed in the Task Tracker Node. Its function is to launch and manage the Mapper and Reducer child daemons (i.e. the mission of a worker daemon is to execute the Map and Reduce Tasks).

3. **Input data** broadly refers to various input data used in a MapReduce computation. It includes the actual input data files (which is referred to as origin data or input data) to be processed and the job configuration data which contains an information needed to run the job (e.g. the number of input splits). The input data or the origin data, is parsed into key-value pairs by Mappers. The Mappers then produce intermediate buffered data.

4. **Intermediate Buffered Data (or Intermediate Data)** is the output of the Map phase. Assuming that the number of reducers used in a job execution is R, then the intermediate data is a set of R regions called partitions (i.e. for each Reducer assigned to a job execution, there is a data partition output from the Map phase to process).

5. **Shuffle Handler** is a service designed to sort and group the intermediate key-values pairs according to the key (an example in the next section explains what this key is and how the key-value is used to present the data). The Shuffle Handler is run in a Task Tracker.

6. **Output Data (or Final Data)** is the final result produced by the Reducers.

### 2.4. Executing Jobs using MapReduce: an Example Scenario

In this section, a practical simplified example is introduced to describe how the MR application processes a data of a job submitted by a client in a specific data execution flow. The example is as follows: Suppose that there is a big number of the UK temperature records. These records contain temperatures of UK cities which are daily bases registered for one year. They are stored in a number of database files as shown in Figure 2.3. These files are stored in a number of data blocks in the DFS. In this example, the job is submitted by the client to the Master. The duty of the job is to find out the maximum temperature of each city (each city has more than one record per file).
Figure 2.3 Tables: examples of input data files

To simplify this example for explanation purpose, let’s consider the number of files $n = 10$, each file stored in one data block, and each input split size is the same as the data block size. Figure 2.4 illustrates, as a layout, how the MR process the input data files for this job. First, the client determines the configuration parameters of his job. In our example, the input files are divided into input splits (10 input splits), and there is an input split per mapper to process. Each Mapper read and parsing its input split into key-value pairs (in our example, the key is the city name and the value is its temperature). The Mapper starts the Map execution Task (in this example, the map phase is to find out the highest temperature for each city in each input block i.e. file). The output result of the Mappers is the intermediate data which is written locally in a number of data partitions. They are equal to the number of theReducers (in our example, two Reducers). After the Mappers completed their tasks, each partition is assigned to a Reducer. The Reducers start their execution task which is finding the maximum temperature per city but before that the Reducers themselves have to perform the shuffle process. In the shuffle process the intermediate data (within the same partition) is sorted and grouped based on the key. (For example, York city (i.e. key) has a couple of temperatures values 18, 11, 23, 9 as an output of different Mappers, as listed in the intermediate result shuffle table in Figure 2.4). The final output result of each reduce task, which is a list of cities and the maximum temperature associated with each city, is written into an output file on the DFS system as shown in Figure 2.4.
2.5. A Generic MapReduce Computational (GMC) Model

As explained in the previous section, the MR application processes the input data files of a job in a specific data execution flow. The data execution flow typically involves a number of MR components and a sequence of interactions between these MR components. This section presents an abstract model of MapReduce computation capturing MR architecture, its components, and the interactions between the client and MR components and among the MR components during a job submission and execution, so-called GMC Model. The following points describe a typical execution flow of a job being submitted to and executed by the MR application that addresses the above limitations discussed in Section 2.2. It describes the execution flow from the moment when the first request of a job submission is sent by the client to the moment when the job is successfully completed. This data execution flow is further illustrated in Figure 2.5.

Figure 2.4 Simplified example of data execution flow in MR application
1. First, the user (on the Client Machine) logs in using his credentials (username and password) and runs the MR Job/Application Client (Figure 2.5 (1) (2)). The used credentials are usually a username and password.

2. The Client requests a Job ID (i.e. an Application ID) and the path to write the Job files (Figure 2.5 (3)) from the Resource Manager.

3. Once the Client acquires the Job ID and the associated path (Figure 2.5 (4)), the Client contacts the Name Node to start writing into the DFS (Figure 2.5 (5)).

4. The Name Node allocates Data Nodes to which the Client is redirected and start writing the job resources (i.e. Input data) to the specified directories (Figure 2.5 (6)).

5. Once the writing is completed the Client informs the Resource Manager that his job is ready to be launched, so up to this point the job is submitted (Figure 2.5 (7)).

6. The Resource Manager allocates a Job Tracker, and the master daemon is launched in this Job Tracker (Figure 2.5 (8) (9)). This daemon creates an object and instant to present the Job and keeps track to the progress of the Tasks of the job.

7. In order for the Job Tracker to create a list of tasks to be run, it retrieves the number of input splits (i.e. a configuration data) from the DFS (Figure 2.5 (10)), whereas one Map Task assigned for each input split. The Job Tracker also determines the Reduce Tasks required for the job.

8. The Job Tracker requests Task Trackers from the Resource Manager for all Map and Reduce Tasks (Figure 2.5 (11)). In this request, as a performance prospective (i.e. consideration), the Job Tracker sends the data location of each Map Task, trying to obtain a Task Tracker close to its input split.

9. Then once the Map and Reduce Tasks have been assigned to Task Trackers, their child daemons are started launching by the worker daemons (Figure 2.5 (12) (14)). However, before the Map Tasks started, the worker daemon retrieves the client input data from the DFS and copy them locally (Figure 2.5 (13)). The Map Task also, parses the input data files into key-value pairs as explained in the previous example.
10. After Map Tasks complete their tasks assigned to them, the Reduce Tasks start to retrieve the output data of the Map Tasks, i.e. the intermediate data (Figure 2.5 (15)). Each Reduce Task is assigned to one data partition (indexed) as explained in the previous example. The intermediate data, at each specified partition, might be located in different nodes as they are an output of different Map Tasks. As a result the Reduce Tasks connect to the respective Task Trackers, in which the data of the indexed partition stored locally, requesting these data (Figure 2.5 (15)).

11. Once Reduce Tasks complete their tasks assigned to them, the output results are written into output files at the DFS (Figure 2.5 (16)).

12. Then the Job Tracker is informed about the completed tasks (Figure 2.5 (17)). Once all assigned job tasks are completed, the MR Job/Application Client is updated with the job status.

This GMC model is derived to capture the key features and characteristics of the most recent implementation of MR model, although what has been captured can also be applied to the classic MR model implementation. As mentioned before in Section 2.2, in the classic implementation of the MR model [3] [4], the Resource Manager is not used, so the client submits his job directly to the Job Tracker (as indicated by the dash-dot lines 3, 4 and 7 in the figure) and the Job Tracker then assigns Map and Reduce Tasks to a set of Task Tracker.
2.6. MapReduce Deployment in Cloud

MR components are typically hosted in a virtual environment. Virtualisation allows multiple independent system software components to be run on a single physical node. It allows the sharing of physical resources by multiple users [19]. The MR model implementation (i.e. MR application) is deployed in such environment due to the advantages provided by such technology. For instance, server consolidation, increase up time (e.g. fault tolerance, live migration, high availability, storage migration) and isolate applications tasks within one physical server [20].
The MR components run in a cloud could be hosted on the top of Virtual Machines (VMs). Figure 2.6 shows how different MR components are hosted. They can be run on the top of the same physical server. They are isolated using virtualisation host machines and virtual switches. These different components share physical resources (e.g. CPUs and RAMs). At the network level, they are isolated by using virtual switches and VLAN mechanisms. Communications among different sets of MR components hosted in different physical servers are carried out through external network fabric. The Hypervisor is the core of the virtualisation technique. As a middleware, the Hypervisor is responsible for creating, managing and executing VM operating system instances and sharing the physical resources. There are number of Cloud service delivery models [21] that could be used for the MR application. These models are different in term of services provided by the MR service providers as following:

1. **Infrastructure-as-a-Service (IaaS):** In this model the customer allocates the virtual resources as needed. The IaaS provider delivers the required storage and networking devices. They also provides a basic security needs, including physical security and system security such as firewalls. The consumer does not manage or control the physical servers or network switches but has control over VMs, operating systems, storage, and applications to be deployed.

2. **Platform-as-a-Service (PaaS):** This service delivery model offers the facilities such as Software Development Kit (SDK) (e.g. Python and Java) which can be used by customers to write their own programs to develop their own applications. With this service deliver model, customers using these facilities for their programs do not have to manage the required software or hardware components such as operating systems, VMs, virtual switches, storage, CPUs and RAMs.

3. **Software as a Service (SaaS):** This delivery model is also defined as an Application as a Service (AaaS) by [22]. In this delivery model. The customer uses the Cloud service provider’s applications running on the cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with exception of limited user-specific configuration settings.
2.7. Chapter Summery

This chapter has presented a background information in relation to MapReduce, in particular on the MR architecture; explaining the MR physical and logical components, their operations and interactions, and the need of such components in the most recent MR model implementation and how they address the limitations of the classic MR model implementation. The chapter has also introduced a use-case scenario describing how MR components process client data by executing a specified job in two distinctive phases; map and reduce (i.e. data execution flow). The chapter has then presented a Generic MapReduce Computational (GMC) model capturing the MR architecture and its components, and specifying a typical MR-Job execution from the moment when a job request is sent from the client to the MR application till the moment when the job is successfully completed by the MR application. Finally, the chapter has shown how the MR components of such model are hosted using virtualisation technique.
Chapter 3

3. MapReduce: Observations and Security Analysis

3.1. Chapter Introduction

This chapter presents some observations made on the GMC model, and it discusses different security issues on the GMC model that related to inadequate security measures and services such as authentication, confidentiality, integrity and accountability. In detail, Section 3.2 presents a number of observations made on the GMC model and identify any features that may have security implications. Section 3.3 analyses security threats that are more closely related to the MR-Job execution and understand the implications of such threats on the provision of the GMC model. Section 3.4 outlines how these threats could be mitigated using various security measures and services. Finally, Section 3.5 summarises the chapter.

3.2. Observations on GMC Model

Based on the workflow of the job submission and execution described on the GMC model in the previous chapter, the following observations which define and belong to the characteristics of the MR application are made.

1) **Observation 1:** Once a client has submitted a job to the Resource Manager, the client no longer interacts with the underlying MR components, i.e. MR logical components. Instead, a master daemon run on a Job Tracker Node interacts with other MR components assigned to the job (e.g. worker daemons run on Task Trackers nodes) on behalf of the client. Also, master and worker daemons on these MR server nodes assume the authority of the client when retrieving the job input splits and data. These services require authority on behalf of the client to access the data. This means that, during a job execution process, the client will need to delegate the authority to the allocated or assigned MR components to carry out further actions required for the job execution. Compromising one of such components could lead to the compromise of more MR components and possibly the entire MR-Job execution, so that during an MR-Job execution, an MR component may interact with one or more MR components, and such interactions should be carried out in an authentic manner.
2) **Observation 2:** The interactions (i.e. transactions) among MR components taking part in an MR-Job execution are largely carried out in a sequential order. For example, the action (i.e. the request) of writing the input data into the DFS does not take place before the client sent a new job request to the Resource Manager to obtain both the job ID and the path for writing the input data. Also, the worker daemons would not be able to retrieve the client data to be processed unless both the job is submitted by the client and the required MR component are allocated. This means that the risk levels caused by any security breach to a transaction in a job work-flow may be different. The earlier the transaction which is compromised, the higher the risk the compromise may bring, as the compromise of an earlier transaction may lead to more opportunities for the attacker to succeed in compromising a later transaction. Compromising one of such interactions could lead to the compromise of one or more MR components and possibly the entire MR-Job execution. This observation indicates that there may be a case to support a solution which applies a stronger level of protection to transactions that are carried out earlier in the MR-Job execution.

3) **Observation 3:** Each client may submit a single job or multiple jobs. The multiple jobs, submitted by a single client and/or by multiple clients, are executed concurrently in the MR application. As mentioned earlier, each job typically consists of a set of tasks, and the tasks may be executed on the same MR server node, or on different MR server nodes. This leads to two parallel processing blocks, one for parallel execution of multiple tasks, which may be generated by a single job or by multiple jobs, and the other for parallel communications among different MR components, again from the same job or from different jobs. There must be measures to ensure that jobs are executed securely and the communications among different MR components are carried out securely too.

4) **Observation 4:** Each job has its own job ID and associated data. The data are input data, intermediate data (i.e. the data generated by the job during the course of its execution) and output data. The input data associated to a job is stored as blocks. These blocks are typically dispatched to different MR server nodes, so-called Data Nodes. A Data Node may be different from the MR server node where the worker daemon is hosted and run. In either cases, i.e. where the worker daemon is hosted and run on the same or in different MR server nodes, measures must be in place to make sure that all the data should be written to, and stored in, secure storage nodes, and only authorised clients and MR component (e.g. the worker daemon to that job ID) could access to the data associated to
the job. An intermediate data generated by any task of the first phase of the MR-Job execution (e.g. a Map Task) is typically stored locally where the task is run by the worker daemon. A task from the second phase of the MR-Job execution (e.g. a Reduce Task), which may be run by another worker daemon and hosted in another MR server node, will need to access the intermediate data produced by the first task. This data access, either locally or remotely, should be granted to authorised MR components, i.e. no component other than the intended tasks or worker daemons, should be allowed to access to the intermediate data. The final result of a job execution may be the output of more than one task, e.g. more than one Reduce Task. As mentioned above, there may be many tasks run by different worker daemons; some of them belong to the job from the same client, but others may belong to jobs from different clients. These tasks need to securely access the data storage nodes to write their results.

5) **Observation 5:** In addition to the main interactions of MR-Job execution discussed before, there are three types of reporting messages which are exchanged periodically between different MR components. First, a master daemon runs on an MR server node, i.e. a Job Tracker Node, receives messages from worker daemons run on other MR server nodes, i.e. Task Tracker nodes. These messages convey the status of the tasks of a submitted job carried out in these MR server nodes. Second, each Data Node sends messages to the Name Node reporting the blocks it stores. Third, the Resource Manager receives heartbeat messages from each MR slave node, e.g. Task Tracker node, to keep track of MR server nodes’ availability. Any unauthorised modifications of these messages will have security implications on the MR master nodes and daemons, and may also influence their follow-up decisions and execution flows. This will affect the MR model implementation as a whole.

6) **Observation 6:** Typically for a fault-tolerance in MR model implementation, there are two considerations for MR server nodes; one for MR master nodes and the second for MR slave nodes. In the first consideration, the MR master nodes, i.e. the Resource Manager and the Name Node, are typically mirrored with passive MR master nodes. In other words, when an MR master node, e.g. Resource Manager, fails, the passive Resource Manager takes over and becomes the active master node. In the second consideration, when an MR slave node (i.e. a Task Tracker node) which hosts and runs a worker daemon is down or disconnected from the DP cluster, the corresponding master daemon allocates another worker daemon, hosted in another MR slave node, to take over
the tasks run by the disconnected worker daemon. Also when an MR slave node in the DFS cluster (i.e. a Data Node) fails, the corresponding MR master node (i.e. Name Node) assigns another Data Node to store and manage the data blocks managed by the failed Data Node. These MR server nodes could be ones of the existing MR server nodes or newly joined MR server nodes. How to ensure that the newly appointed MR server node is trustworthy enough to carry out the work, as a new vulnerable or untrusted MR server node could put existing or new MR-Job executions at risk.

7) **Observation7:** The last observation is that in the most recent implementation of MR model, different tasks of different jobs may run by different worker daemons, and these worker daemons may run in the same or in different MR server node/s, and multiple MR server nodes belongs to the same domain. However, how if part of these MR components, i.e. worker daemons, may be hosted in another MR server nodes that may belong to another domain. In other words, these MR server nodes are provided by the same domain (i.e. single provider). In the future deployment of MR model implementation, this may change. For example, some MR master nodes may belong to one domain, but others may belong to another domain, or the master nodes belong to one domain, but the slave nodes may belong to another domain, possibly provided by a third party provider.

### 3.3. Security Threats on GMC Model

To build on the related work, such as [23] [24], which largely analyses security threats in the cloud context in general, this section focuses on analysing and identifying the threats more closely related to the GMC model and based on the above observations. According to Meetei, M. and Goel, A. [25], more than 60% of the attacks on the Internet is against the web applications. One of the top five security threats on Internet services (e.g. cloud) is password guesses. 29% of data breach cases involved exploitation of this attack. The challenge is that the MR model implementation relies on both a web application (as a client software interface, e.g. web browser) and the password (credential) authentication scheme. The current authentication methods proposed for the MR assumes that clients do not have root access to the MR components, and cannot read or change data packets transmitted over the network of the MR clusters. Even if these assumptions are true, there are still limitations in these authentication methods, which is analysed in the next chapter.

Without having adequate security services and measures in place, such as authentication, MR-Job execution will be vulnerable to a variety of security threats and attacks. To identify
security threats in the MR application, one should consider both MR components and their roles, and the data managed or handled by these components. The threats that are identified and discussed here, can be classified into five main categories: identity spoofing and impersonation threats, non-compliance threat, data disclosure and modification threats, virtualisation related threats, and deny of services (DoS) threat.

3.3.1. Identity Spoofing and Impersonation Threats

This threat occurs when an attacker pretends to be an authorised client or MR component and try to access the MR resources such as Data Nodes. This could happen if the adversary managed to get the credential of an authorised client or MR component. There are a number of means by which the attacker may acquire a legitimate user’s credential (e.g. a user password) [23]. Launching offline brute force attack is one of the used schemes. The attacker for example may penetrate the client machine or may sniff on the network traffic to steal the hash value of the password. If the attacker get hold of such value, then the attacker can start launching offline brute force attack in order to discover the user password and use this credential later. Once the attacker gains the proper credential, he might pretend to be a legitimate client and claims an access to the jobs and/or MR resources.

The spoofing attack is also a form of impersonating attack. It is usually performed by impersonating the address (e.g. IP address) of a legitimate component. This enables an attacker, e.g. the Man-In-The Middle (MITM), to redirect and intercept the request and response messages. These messages content might be altered or fabricated for his purpose [23] [16]. For example, an attacker may spoof both the identity and the address of a genuine Data Node, and if this is succeeded, a client or an MR component (e.g. Task Tracker) may be deceived by accessing the attacker node rather than the designated Data Node. Another example is that an attacker may modify the source address of a retrieving request (e.g. for an input data) with his address. In this case, the attacker may receive the response message which is intended to be sent to a legitimate requester (e.g. a remote Task Tracker), and this response message may contain confidential input data.

Furthermore, one of the idea behind the MR model implementation is to allow a client’s program (i.e. code) to be run as a part of map and reduce tasks. If an attacker could impersonate a trusted and authorised client, he could launch his own code, which could be a
malicious code\textsuperscript{1} [26] [27]. Once the malicious code gets a foothold on an MR component, it can cause significant loss or harm to clients and/or providers. For instance, if an impersonated client (i.e. an attacker) was able to upload (i.e. write) the malicious code as a part of his input files to the Data Nodes, then the malicious code may sniff or gather information about other MR components (e.g. starts learning the other nodes addresses) for further attacks. Then it may propagate through them by waiting for any MR component, e.g. Task Tracker, to request (trigger) this file code and overwhelm the network connection between the MR components. Also, the malicious code might provide the attacker an access to the input files and the intermediate data of other clients, so he might steal their sensitive data, manipulate results or even destroy the data files for disrupting their job execution flow on the MR application.

\subsection*{3.3.2. Data Disclosure and Modifications Threats}

Data disclosure and modification threat is considered to be against the data confidentiality and reliability. A client uploads his input data into remote Data Nodes of the DFS, and processed by Task Trackers (i.e. Map and Reduce Tasks) from the DP cluster. These Data Nodes, in which the client data resides, are shared and accessed by a number of clients and MR components. The threats of data compromising in such resource-sharing environment increases as the number points of access caused by clients and MR components increases [28]. This could cause a security breach in term of accessing the data (of a job submitted by the client) in the DFS [29]. If unauthorised client or a Task Tracker (assigned for different job) or an attacker have access to the Data Nodes, it can start reading and writing or even manipulating the data of other clients in the Data Nodes. The data disclosure threat could also be achieved by MITM attack when there is no secure communication between the DFS and other MR components and clients. The MITM attacker can read and alter the messages which are sent and received by these components.

Using a remote data storage server might cause another data violation threat. It presents in data locality and remanence. Once a client uploads his input data into the DFS, the client no longer possesses the data locally; he may not know where the data has been stored. It might be moved to a location where the client has not given a permission to. The client may prefer a specific location. Also, after the execution of a job submitted by a client is completed, how could the client ensure that the data has been erased successfully, especially, if his data

\footnote{\textsuperscript{1} A malicious code is a program that have been developed with the purpose of performing a misuse. It is also called malware. These include Viruses, Worms, Trojan Horses, and etc.}
is private and confidential, and how he ensures that his data is not used for other purposes (e.g. commercial purposes) either by authorised or unauthorised clients or MR components (i.e. an MR provider) [30].

3.3.3. Non-Compliance Threat

The non-compliance threat is considered to be against the MR data computation (i.e. MR data miscalculation). This threat could take place by a cheating behavior of an MR component. The cheating behavior of an MR component could occur when an MR provider uses MR components from other MR providers to process a job submitted to it. If one child daemon (Map or Reduce Task) produces a false output, that results to incorrect job result. For example, a child daemon (a cheater) could give a wrong or less computational result by: (1) falsify a tiny part of the result or (2) drops the task process at any point before reaches the final point in the task or (3) does not start the process of the task from the beginning. Another example of such behaviors is that a Name Node in the DFS could send a fake response to a requester (e.g. a Task Tracker) and misleads the Task Tracker by redirect the retrieving request to fetch a wrong input data from Data Nodes. If such cheating behavior is successful, the Map and Reduce Tasks would not be able to execute their tasks properly and produce the wrong result. This might disrupt, delay or even stop the executions of legitimate jobs.

The aim of such cheating behaviors could be motivated for arbitrary (fun) or strategic purposes (i.e. economically motivated) [31]. For example, these cheating behavior may be for using less computation resources for one task and performing more tasks during a short period of time, so that the third party increases their profit or degrades the reputation of the other MR provider, which receives the job submission request, by producing a wrong results to the clients.

3.3.4. Virtualisation Related Threats

The virtual environment becomes a target for attackers, as it is used to host different application services and their components such as the MR application. The virtual machines in which MR components reside might be a target for the adversary. The adversary may launch a sniffing attack on the address of an MR component to find any open ports and exploits any vulnerability within the operating system of the virtual machine. One security
loophole at one virtual machine might affect the other virtual machines running on the same physical shared resources. The attacker may exploit the security loophole to launch a malicious code and establish covert channel [24]. A managed malicious code might allow the attacker to disrupt the execution of the master and worker daemons in an MR-Job execution and/or bringing the system down. This attack might take place in the IaaS cloud delivery model, as in this delivery model, clients would be able to install their own operating systems. These operating systems may be left unpatched by the clients, hence creating vulnerabilities on the virtual machines [16].

In addition to virtual machines’ attacks, the hypervisor or the host operating system which manages the virtual machines within one physical machine is another concern. If an attacker found any loophole in the hypervisor, this means he might control the hypervisor, and this could enable him to launch further attacks and/or may take control of the virtual machines [24] [16]. The attacker might also create a malicious or fake virtual machines’ instance/s. If such attacker is successful, he may use these virtual machines to (1) overwhelm the physical resources or (2) distribute another malicious code to MR components hosted in this virtual environment, all of which could disrupt an MR-Job execution and deprive the MR resources availability.

### 3.3.5. Denial of Services (DoS) Threats

A DoS or Distributed DoS (DDoS) refer to an attack by which one or more client nodes exhausts the MR application to such an extent that the MR application could no longer serve legitimate requests or execute its tasks [23]. The Resource Manager and the Name Node are essential server nodes which might be more vulnerable to such attack. If this attack happens, it does not only affect these MR master nodes but also the other MR components such as Task Trackers and Data Nodes.

When the MR application is delivered to clients using SaaS/AaaS delivery model, this attack could take place in a consecutive form. In this delivery model, the MR application is accessible through a client interface. As mentioned in the previous chapter, the client interface is the MR Application Client running on the client machine, and it is usually a web-based application. Compromised MR Application Clients could exhaust the MR resources by submitting fake jobs to the MR application. In other words, compromised MR Application Clients could overwhelm the MR application by sending new jobs and writing requests to the Resource Manager and the Name Node, respectively, to obtain jobs’ IDs and paths (where
the clients store their input data files). By following the specification of the MR-Job execution sequence, this could proceed to mount further DoS attacks on other MR resources by: (1) requesting to upload invalid data (may contain malicious codes) to the Data Nodes in an attempt to exhaust the Data Nodes by uploading invalid input data, and (2) allocating as much Job Trackers as possible to retrieve invalid inputs splits uploaded into the Data Nodes and these Job Trackers request as much MR resources (i.e. Task Trackers) as possible from the Resource Manager for fake tasks. These consecutive form of DoS attacks can overwhelm the entire MR application. Further, the DoS attack on the MR application could take place in a single form when it is delivered as IaaS. A compromised hypervisor manager, which is responsible for creating and managing the instances of the virtual machines, may create many virtual machines to overwhelm the physical shared resources such as RAMs and CPUs on the physical machines [23].

In addition to the previous forms of DoS attacks on the MR application, the attacker might play with reporting messages, such as those exchanged between the master and the worker daemons (job-status messages), and/or those exchanged between the Resource Manager and the other MR server nodes (heartbeat messages). The job-status messages might be halted, dropped or even altered. In such case, the master daemon will be deceived and any of the following two scenarios might occur. First, a master daemon does not release the worker daemons assigned to it as the master daemon presumes that the worker daemon are still executing the assigned tasks (i.e. Map and Reduce Tasks). Second, the master daemon will not be able to update the clients with the status of the submitted jobs. The Map and Reduce Tasks may be reassigned to other worker daemons when the master daemon believes, by the altered notification messages, that the tasks are not completed successfully, though the tasks have already been completed. Also if the heartbeat messages are dropped and the Resource Manager could not receive the related heartbeat messages from an MR slave node, then the MR slave node is considered to be down. As a result the available MR resources, e.g. Job and Task Tracker nodes, in the PF cluster are considered to be down (i.e. reduced).

These type of attacks will degrade the response of the Resource Manager making it slow in serving authorised clients, or even the Resource Manager becomes unable to assign any new Task Tracker node for new jobs submitted as it runs out of free resources. As mentioned above, the DoS attack may also flood the storage capacity with false data blocks making DFS too stretched to serve the authorised clients. In other words, the DoS attacks could disrupt and deprive the MR components from processing their tasks for authorised clients.
3.4. Security Measures and Services

The security threats on the GMC model described in the previous section could be mitigated by applying different security measures and services.

Identify spoofing and impersonation threats could be mitigated using an authentication service. This is because the authentication service ensures that the identities of a client and an MR component are as claimed to be, and that the data used and the messages exchanged are authentic. The authentication service can be implemented using different authentication methods, such as password-based authentication method and key-based authentication method. However, the authentication service depends on other security services such as confidentiality and integrity. The authentication credentials of the clients or the MR components must be confidentiality and integrity protected. This is because if the confidentiality and the integrity protection of such authentication credentials is compromised, then there is no longer a reasonable expectation that the authentication service is still valid. For example, when a password-based authentication method is used, if an attacker is able to get a client credential, e.g. a password, from the hash value of the password, he can easily impersonate the client using this compromised credential. Furthermore, some authentication service might apply a security measure that locks out a user account if it receives a certain number of unsuccessful password attempts. The attacker might exploit this security measure to launch a DoS attack against the user account. The attacker may disrupt or prevent the user from submitting a job to the MR application, by attempting incorrect or arbitrary passwords for the legitimate user name (i.e. the user account) [32], but, if an MR authentication service does not provide such security measure (user-account lock), that gives the attacker the opportunity to launch an online brute-force attack on the user password. Thus, a strong and secure authentication service which provides the means to verify the identity of a client or an MR component is extremely important.

Data disclosure and modification threats could be mitigated using the confidentiality and integrity protections. The confidentiality is the requirement or property that a data or a message not be disclosed to unauthorised entity. The integrity is the requirement or property that a data or a message has not been altered in unauthorised manner while in storage or transit. Encryption and hashing techniques are used to implement such services. The encryption however could prevent good indexing and searching of data. It has also a high computation cost for encrypting a big data [30]. Very often, cryptographic keys, used for such techniques, are issued and distributed to the designated entity (i.e. clients or MR
components which are involved in encrypting or decrypting process). Several protocols, such as Kerberos, have been developed for, but not limited to, the provision of cryptographic keys (i.e. generating and distributing/exchanging of cryptographic keys). The Kerberos protocol has later been adopted for the MR application. However, the distributed key not only requires a confidentiality and integrity protection but also the authentication of the issuer/sender of the key. In other words, the receiver of a distributed key desires assurance that the key comes from the expected authorised key issuer/sender (e.g. client or MR component).

A non-compliance threat using cheating behavior could be mitigated by enforcing a Service Level Agreement (SLA) between MR providers and clients and among the MR providers. This is because this cheating behavior occurs mostly when an MR provider uses MR components of other MR providers for executing a job submitted to it. SLA requires an accountability technique. A delivered MR application by a public cloud should be made accountable for both the clients and the MR providers. Both parties should be able to check whether the MR application is running as agreed. If a problem appears, they should be able to determine which of them is responsible, and to prove the presence of the problem to a third party, i.e. an arbitrator. In other words, an accountable MR application enables clients and MR providers to determine who causes the fault if an issue occurs. Regulators, auditors or business governance, could monitor any claim made by a client or an MR provider regarding the delivered MR application. They must ensure that the SLA is applied and complied by the MR providers. Examples of issues might escalate are: one worker daemon might be compromised or misconfigured that leads to incorrect results to clients, second, the MR provider might provide inadequate resources as agreed in the SLA which causes degradation in the performance, and third, a client machine might be compromised and a malicious code runs on it, which could cause a disruption to the client job execution. Thus, the accountability technique is a way of trusting the MR provider as it provides both the client and MR provider itself with more transparency about the tasks and the data of their jobs. By such mean, i.e. using accountability, the client could assess the trustworthiness of an MR provider. The accountability might be considered by monitoring the status of a submitted job with more details about the MR components involved in the client job. For instance, a client can see the addresses, locations and the names of the MR components as well as the other MR providers involved, if any, in the execution of his job.
An accountable MR application is reliable if (i) faults can be reliably detected, and (ii) each fault can be undeniably linked to at least one faulty client or MR component. More specifically, it should provide the following features [33].

1) **Identity:** Any function or action takes place in the MR application should be related to a component (a client or MR component) which performs the function.

2) **Secure logs (records):** Any action that has been achieved in the MR application should be recorded according to an identity involved in that action for further investigation.

3) **Auditing and prove of evidence:** The logs of the MR application can be inspected or navigated by regulators (third party auditor) as an evidence to determine who causes the fault based on the identity.

As can be seen from the above three features provided by the accountability, the features provided depend on the identity of an entity. A secure and efficient authentication mechanism could provide reliable identification for any client or MR component. In case if an identity has been compromised or impersonated, that means all accountability features listed above could be fake events rather than trusted ones on which the regulators rely on to investigate the claims.

Furthermore, different security measures and services could be used to mitigate the virtualisation related and DoS threats. Here are some of these security measures and services. First one is to monitor the intercommunication and the behavior activities in among the virtual environment components (virtual switches and machines) and between the clients and the virtual environment components by using monitoring tools such as Intrusion Detection and Prevention Systems (IDPS) [34] [35]. The second is to ensure the last security patches of both the hypervisor and the operating systems of the virtual machines and the clients are installed. The third is to isolate the operating systems of the virtual machines by (i) having separate partitions for resources, and (ii) limiting the communications and the access of the operating systems that each virtual machine has to the other virtual machine, to the hypervisor, and to the host operating system (if any)[36]. The fourth is to consider the use of different security mechanisms such as (i) authentication mechanism in each layer of the virtualisation solution – virtual machine, hypervisor and the host operating system, and (ii) an integrity protection for all virtual machines instances which can be performed using, for examples, integrity checksums or digital signature [37] [38]. The fifth is to provide each
client with a specific MR Application Client interface with a strong authentication mechanism protecting the client’s authentication credential.

3.5. Chapter Summery

In this chapter, a variety of observations on the GMC model have been discussed. These observations describe the status and relationship between different MR components and the job execution flow. Examining these observation raises security concerns on an MR-Job execution. To overcome these security concerns, there are needs of different security measures and services, however, understanding these needs requires conducting a threat analysis. Thus, this chapter has also, categorised and analysed a number of threats in relation to the MR-Job execution and how such threats could be mitigated using different security measures and services such as applying strong authentication mechanisms, confidentiality and integrity services, and an accountable MR application.

According to the threat analysis, conducted in this chapter, it can be seen that the impersonating threat is the most critical threat. This is because, once an impersonation attack is successful, the attacker could use the victim’s identity (may include credential) to access jobs and MR resources which he is not authorised to access, and/or he could use the compromised component (the victim’s component) as a springboard to launch any forms of further attacks. For example DoS might also be launched from an impersonated component. The risk of impersonation attacks is higher if the MR application uses a weak authentication solution, and/or if the authentication solution deployed does not capture the features of the MR application.

Providing an effective and efficient authentication service for an MR job workflow is a challenging task. This is partially due to the complexity of the MR execution environment such as those highlighted in the observation section, and partially due to the fact that the authentication service is closely nested with other security services, including confidentiality, integrity, authorisation and accountability. The security level of the data confidentiality service in MapReduce is dependent on the assurance level (i.e. the security strength) of the authentication service, as the encryption keys are typically derived and distributed during an authentication process. Protecting an authentication credential of a client or an MR component from being compromised is essential to prevent any further attacks. The authentication service also affects the authorisation service as any compromising in an authentication process could lead to unauthorised (impersonation) access to a user data on
the DFS, which means a breach in data confidentiality and integrity. Accountable MR application uses the identity of each client and MR component. So any fake identity or identity theft (i.e. impersonated identity) may lead to incorrect logs being recorded and a mistrust between the MR provider and the clients could be escalated.

To summarise, the effectiveness in mitigating many of the security threats in the MR application is closely dependent on the effectiveness of the authentication service. This motivates us to investigate the authentication methods proposed or used for MR application. Thus, the next chapter presents a critical analysis for these methods and this critical analysis is guided using a set of specified MR authentication requirements.
Chapter 4

4. Authentication Methods: A literature Survey

4.1. Chapter Introduction

This chapter presents a literature survey on the existing authentication methods proposed or used in MR application and in other cloud computing applications. The chapter critically analyses the state-of-the-art in MR authentication methods. This analysis is guided using a set of MR authentication requirements. The purpose of this critical analysis is to examine the suitability and effectiveness of the existing authentication methods in the mist of MR features and characteristics, thus identifying knowledge gaps and areas for improvement. In details, Section 4.2 specifies a set of authentication requirements for the MR application. Section 4.3 critically analyses the existing work on the MR authentication against the specified MR authentication requirements. This analysis covers the authentication methods already adopted by the MR application, and those recently published in the literature for the MR application. In addition to these authentication methods, this chapter briefly discusses the authentication methods designed for cloud computing in Section 4.4. Finally, Section 4.5 summarise the chapter and defines what is missing.

4.2. Requirements for MapReduce Authentication

This section specifies a set of MR authentication requirements. The specification of these requirements has taken into account (1) the characteristics of the GMC model and the outcome of the threat analysis carried out on the model and discussed in the previous chapters, and (3) the existing authentication methods discussed in this chapter.

4.2.1. Entity Identification

To authenticate an MR-Client or an MR component involved in an MR-Job execution the following entities identifiers should be considered.

---

2 Cloud computing refers to the delivery of on-demand computing resources/services—everything from applications to data centres over the internet.
A. MR-Clients IDs: a unique identifier for each MR-Client. This is usually a static ID, and it is typically the username of a user who has been identified to (i.e. registered with) the MR application.

B. MR Components IDs: a unique identifier for each MR component. This is could be static or dynamic ID, and it is typically a name or symbol contains a set of letters and digits. It is used to identify an MR component.

C. MR-Jobs IDs: Each MR-Job should have a unique identifier to distinguish different jobs submitted by different clients.

D. MR-Infrastructure IDs (also called cluster IDs): If there are two or more parties providing hosting nodes, then the hosting nodes provided by a single party may be treated as one infrastructure (or cluster), and each infrastructure should be identified by a unique identifier.

Authentication is typically carried out by demonstrating (by a claimant), and verifying (by a verifier), the knowledge of a secret uniquely associated to an identity. Therefore there is a need for secure issuance and acquisition of an identity secret/s (which is also part of the corresponding authentication credential). This leads to entity identification (or registration). In other words, there should be a secure method for a new client or a new MR component to be identified to the MR application and to establish secret/s associated to the identity.

4.2.2. Entity Authentications

Entity authentication is to make sure that a communicating entity is the one that it claims to be. Multiple entities (i.e. MR clients and components) in the MR application are involved in an MR-Job execution. As mentioned above, some of these components are static components, such as the Resource Manager and the Name Node, while others are dynamic ones, such as Task Trackers. The static components are identified by static identities that, once given, remain the same during the lifetimes of the components, while the dynamic components are identified by dynamic identities. A dynamic identity is assigned to a dynamic component when the component is assigned to a job. If this job is completed in which case the component may be assigned to another job, and if this is the case, this component will be assigned with a new identity. In an authentication solution to be designed for the MR application, all the MR components taking part, or being involved in an MR-Job execution, being static or dynamic, should be securely authenticated. In detail, with
reference to the MR application depicted GMC model, the authentication task should satisfy the following requirements:

(1) Mutual Authentication between an MR-Client and an MR master node

This is to ensure that only an authorised MR-Client can connect to the MR application. More specifically, this should cover the mutual authentication between an MR-Client and the Resource Manager and between the MR-Client and the Name Node.

(2) Mutual Authentication between an MR component and an MR master node

This is to ensure that only an authorised MR component can join the MR application. More specifically, this is to ensure that any new MR server node (e.g. a Task Tracker node) assigned for hosting an MR logical component (e.g. master or worker daemon) is authenticated to an MR master node and vice versa, hence any access to MR resources’ data/information is granted in a secure manner.

(3) Mutual Authentication between any Pair of MR components

This is to ensure that any two MR component involved in the execution of a job submitted by an MR-Client are authenticated to each other when requesting an MR or a job resource and/or exchanging information related to such resource, so as to ensure that any access to the input, intermediate and output data of the job and/or the exchanged information can be granted in a secure manner.

(4) Mutual Authentication between domains (i.e. inter-domain authentication)

This authentication is needed when a third party is involved in an MR-Job execution. It is to ensure that any new MR server node which belongs to another domain (i.e. a domain of a third party) and involved in executing and/or hosting MR logical components is authenticated to the domain to which the job is submitted and managed.

4.2.3. Authenticity of Data and Exchanged Messages

(1) Data Authenticity

This is to ensure the origin authentication and integrity protection of data that are used or produced by the MR application. In other words, the protection should be applied to input data, intermediate data and output data of any job processed by the MR application.
(2) Authenticity of Exchanged Messages

The origin authentication and integrity protection should also be applied to all messages of any protocol facilitating the tasks of authentication in the MR application. The protocol messages are of two types: authentication requests and authentication responses.

4.2.4. Confidentiality of Exchanged Messages

This is a protection of authentication requests and replies from any unauthorised disclosure. To counter eavesdropping attacks, the confidentiality of any such request or response sent between MR-Clients and MR components and among MR components throughout an MR-Job execution should be protected.

4.3. Authentication Methods for MapReduce

This section analyses the authentication methods proposed for the MR application based on the requirements specified above. These authentication methods include those ever adopted by the MR application and also those published in literature.

4.3.1. Authentication Methods ever adopted

Two authentication methods have been adopted by the MR application so far [11] [16] [39]. The first one [11] [16] adopted in the early generation of the MR model implementation, assumed the use of an independent authentication service outside the MR application, e.g. an authentication service comes with the hosting operating system (OS) – so called OS-based authentication method. In other words, the MR application did not have its own authentication service. Rather it relies on the use of an authentication facility provided by the OSes. The user identity is whatever the host OS says it is, as a result the MR component (e.g. Name Node) does not perform users authentication. Any user having access to the cluster domain through a configured client has access to the data stored in the cluster – only limited by file and directory permissions. Further, a user with access to the cluster can create a local user - with the same name as an administrator-user, and may delete or manipulate the data of the other jobs (i.e. clients) [40].

The second method was proposed by O. Malley et al. from the Yahoo MR application team (hereafter referred to as O. Malley method) [39]. This method is symmetric key based
authentication and it is largely built on the Kerberos authentication protocol [41]. At the
time of writing this thesis, the Kerberos authentication protocol is still the default mode of
authentication for the MR application deployed in a private cloud [16] [40]. Figure 4.1
summarises the authentication process using this method. As shown in the figure, a client
or an MR component first authenticates itself to the authentication server. Upon successful
authentication, the MR component will obtain a Ticket Granting Ticket (TGT), which is
then used to acquire a service ticket. The service ticket is then used by the client or the MR
component to access resources located on other MR components. This authentication
process consists of six steps (steps 1 to 6, as shown in the figure), and is identical for all the
MR components in the application. Assuming that a client is to request a new job ID or
writing his job into the MR application as part of a job submission process, to authenticate
himself to the application, the client first makes an authentication request to the
Authentication Service (AS). The AS generates a response containing a TGT and sent to the
client. Then the client uses this TGT to request a service ticket by sending the TGT along
with an authenticator to demonstrate the secret in the TGT to Ticket Granting Service
(TGS). Once the client receives this service ticket, the client uses it to authenticate to an MR
master (e.g. Resource Manager). The same steps are taken by any other MR components,
such as Task Trackers, to get admitted to the cluster and access other (remote) MR
components for retrieving data or other resources used by the client's job. Once an MR
component is authenticated, obtains a service ticket and it is admitted to the master in the
MR application, the master generates a new secret which is used by both the MR component
admitted and the master to authenticate mutually to each other in any further authentication
[16] [39].
This authentication method is a one-factor authentication method. The one factor used by a client to authenticate himself to the AS is the client's password. Knowing the password would allow any entity to acquire a service ticket in the name of the client and to access any resources granted to the client. In other words, for an attacker to impersonate a legitimate component (e.g. a client or an MR component), the attacker needs to obtain a service ticket. To access the ticket, the attacker needs to know the password of the (legitimate) client to whom the ticket has been issued as it is based on the password. If a client's password is compromised, then all the resources assigned to the client will be at risk. Also, the attacker could use this compromised account to launch further attacks in the MR application. In other words, the security level offered by this one-factor authentication method is the same as that offered by the password chosen by a client. If a client chooses a weak password, then the risks imposed on the MR application will increase accordingly. In addition to the use of such authentication secret in this method, i.e. user password, they assume that the client is identified locally to the MR application, and MR components communicate to each other in a trusted network. However, these two assumptions are no longer valid as the client may not have a local user account, i.e. it could be a remote client, and MR components could be communicating over a Wired Area Network (WAN).
4.3.2. Authentication Methods Published in Literature

In addition to the authentication methods discussed above, there are also methods that have been published in the research domain for the MR application. These methods can largely be classified into two groups, symmetric key based and asymmetric key based. The authentication methods proposed by Somu et al. [42] and Rubika et al. [43] are symmetric key based, and their focus is on verifying the identities of clients requesting to access the MR application. On the other hand, the methods proposed by Wei et al. [44], Ruan et al. [45] are asymmetric key based. They focus on verifying the authenticity of an MR component. In addition, the method proposed by Zhao et al. [46] is also asymmetric key based that tried to address both clients' authentication and MR components' authentication.

4.3.2.1. Somu and Rubika Authentication Methods

Somu et al. [42] proposed an authentication method (hereafter referred to as the Somu method) to authenticate clients requesting access to the MR application. This method is symmetric key based. It is similar to the O. Malley method in that both methods use a single authentication factor, relying on the use of a client's username and password, to authenticate the client to the MR application. However, unlike the O. Malley method, the Somu method uses two further ideas to strengthen the security level of the authentication service. The two ideas are: (1) the introduction of a one-time pad key (session valid only), and (2) the use of the principle of the separation of duties. A ciphertext of a client's password, encrypted using the client's one-time pad key, is stored in a Registration Server (one of a two servers used to implement the authentication service) and a ciphertext of the client's one-time pad key, encrypted using the client's password, is stored in different server, called a Backend Server. The two ideas are used in such a manner that no passwords or encrypted passwords are sent over the channel and no cleartext passwords are stored in any of the two servers, thus minimise the exposure of clients' long-term credentials, i.e. the passwords.

Figure 4.2 depicts the authentication process using the Somu method. As shown in the figure, the two servers (i.e. the Registration Server and the Backend Server) are involved in an authentication process (in verifying a client's ID). The verification makes use of three ciphertexts, Ciphertext-1, Ciphertext-2 and Ciphertext-3. Ciphertext-1 is the client's password encrypted using a one-time pad key belonging to the client, and it is stored in the Registration Server. Ciphertext-2 is the one-time pad key encrypted with the user's password and it is pre-stored in the Backend Server. Ciphertext-3 is generated by the Registration
Server each time when an authentication request is received. It is generated by encrypting the one-time pad key using the user's password. Figure 4.2 shows the steps of the Somu authentication method. First, the client sends an authentication request to the Registration Server and this request contains the client's username. The Registration Server forwards this request to the Backend Server. The Backend Server uses the username to fetch and return Ciphertext-2 (it is pre-stored) to the client through the Registration Server. The client decrypts Ciphertext-2 using his password, and sends the pad key back to the Registration Server. These steps are indicated by messages 1, 2, 3, 4 and 5, in the figure. The Registration Server then uses the pad key to decrypt Ciphertext-1 to obtain the password and then uses the password to encrypt the pad key to generate Ciphertext-3. The Registration Server then sends Ciphertext-3 to the Backend Server, as indicated by messages 6, 7, and 8. Finally as indicated by messages 9, 10, 11 and 12, the Backend Server compares Ciphertext-3 with Ciphertext-2 and if the two are equal, the Backend Server will send a positive notification to the Registration Server, which contains the client's Username. The Registration Server compares the Username received from Backend Server with the one received from the user. If they match, then the client is authenticated successfully.

![Figure 4.2 Authentication process of the Somu method](image)
The Somu authentication method provides a confidentiality protection for the clients’ long-term credentials-verifiers. This protection involves the use of a symmetric one-time pad key and two authentication servers. A client's password is encrypted with the one-time pad key, the one-time pad key is encrypted with the password, and the two encrypted items (the two ciphertexts) are, respectively, stored on two different servers. To impersonate a client, an attacker needs to guess or obtain the client's password. To guess or obtain the password by compromising both servers and stealing the two ciphertexts (credential-verifier) is difficult or may not be feasible. For example, if the attacker can steal Ciphertext-1 (the encryption of the password using the one-time pad key) from the Registration Server, the attacker has to guess the pad key to access the password. However, this is computationally difficult, if it is assumed that the used encryption algorithm is secure, besides that the pad key used is valid for one session only. Once the client logs off a session, a new pad key will be generated and used to encrypt the password for further authentication process [42].

Rubika et al. [43] has also proposed an authentication method (hereafter referred to as the Rubika method) to authenticate clients to the MR application. This method uses three servers for authentication, an Authentication Server and two backend servers (Backend Server 1 and Backend Server 2). Figure 4.3 shows the registration and authentication processes of this method. To register, a client submits his username and password to the Authentication Server (or a password is created for the client). The server divides the password, a set of ASCII letter, into three values, p1, p2, and p3, and it also generates three random numbers, r1, r2 and r3. Then the Authentication Server uses the two sets of values, \{p1, p2, p3\} and \{r1, r2, r3\}, to generate a new set of values called angles that are denoted as \{θ1, θ2 and θ3\}. The Username and the random numbers \{r1, r2 and r3\} are stored in Backend Server 1 and the Username and \{θ1, θ2 and θ3\} are stored in Backend Server 2. These two sets of values are used to authenticate the client when the client makes a request to access the MR application.

---

3 A credential-verifier is an authentication information that is used to validate the authentication credential against, e.g. a password-verifier.
As described above, the Rubika method uses three servers for authentication, and one of the three servers, the Authentication Server, is exposed to the public (i.e. accessible to users). The other two servers, i.e. the backend servers, are used to store password-verifiers. In other words, with this approach, there is nothing related to the clients' passwords that are stored in the server accessible by the public. In the Somu method, on the other hand, clients' encrypted passwords are stored in the registration server which is exposed to the public. In addition, with the Rubika method, to compromise a password by stealing the password-verifier (to launch offline password guessing attacks), an attacker would have to compromise the two backend servers, as each password verifier is divided into two portions and each portion is stored on a different server.

However, the Somu and Rubika authentication methods have three limitations. Firstly, it only supports gate-level authentication. In other words, it only supports the client's authentication to the MR application; it does not support the authentication of one MR component to another (e.g. the authentication of a Task Tracker to the Name Node). Secondly, it does not support mutual authentication between the client and the remote servers. Thirdly, similar to the O. Malley’s method, Somu and Rubika authentication methods are a password-based and a single factor authentication method. These
authentication methods are subjected to password guessing and replay attacks as no secure connection is established. Knowing a client password would allow any entity to impersonate the client.

### 4.3.2.2. Wei and Ruan Authentication Methods

Both Somu and Rubika authentication methods are designed to support client authentication only. They do not consider the authentication issue between different MR components. Wei et al and Ruan et al tried to address this issue by proposing a SecureMR Framework [44] and a Trusted MapReduce (TMR) Framework [45], respectively. The SecureMR Framework (hereafter referred to as the Wei method) is aimed to provide integrity and origin authentication of MR data and processing services; namely both the messages that are sent among MR components when Map and Reduce Tasks are assigned, and the data that are processed or generated by these components. However, in Wei method, they assume that a Task Tracker is assigned a single task, so when a Task Tracker is assigned a Map Task, it is called a Mapper, and when it is assigned a Reduce Task, it is called a Reducer.

The Wei’s method protects the authenticity of both the intermediate data and final results of an MR-Job execution. A Task Tracker (Reducer) verifies the authenticity of the intermediate data produced by another Task Tracker (Mapper), and a client should verify the authenticity of the final result generated by a Reduce Task. The method also supports consistency checks of intermediate data and final results from an MR-Job execution. This is done by replicating some Map and Reduce Tasks and assign them to different Task Trackers. At the end of the computation, the job master (i.e. the Job Tracker) compares the results produced by different sets of tasks. If the results are identical, then the consistency of the results (either intermediate results or final results) is assured. The verification process is carried out collaboratively between the Job Tracker and a Task Tracker. For example, Figure 4.4 shows two protocol messages, Assign and Commit, that are used to authenticate and verify the authenticity of both the task assigned and the data produced by the task. As shown in the figure, to assign a Map Task, the Job Tracker sends the Task Tracker an Assign message containing the ID of a Map Task (ID map), and the location of the data (Data Location). The Job Tracker uses its private key to sign the message and then encrypts the message with the public key of the Task Tracker. When the Task Tracker receives the Assign message, the Task Tracker uses (i) its private key to decrypt the message and (2) the
public key of the Job Tracker to verify the signature of the message. Upon a positive verification, the Task Tracker executes the Map Task assigned. After the task execution is completed, the Task Tracker hashes each partition of the intermediate data and signs the hashed values using its private key, and it then constructs and sends a Commit message to the Job Tracker. Upon the receipt of this message, the Job Tracker uses the public key of the Task Tracker to verify the correctness of the Commit message. If the Job Tracker receives more than one Commit message from different Task Trackers (Mappers) but for the same Map Task (replicated task), the Job Tracker will compare the signed values contained in the different Commit messages to see if they are consistent with each other or not [44].

![Diagram](image)

**Figure 4.4 The Wei method: exchanged messages between a Task Tracker and the Job Tracker.**

Similarly, in this authentication method, (i) Task Trackers assigned Reduce Tasks (Reducers) verify the authenticity of the intermediate data, which are produced by another Task Trackers (Mappers), using the public keys of the Mappers, and (ii) the clients have their final result produced verify the authenticity of the result using the public keys of the Reducers. The Wei method, also ensures the confidentiality of the exchanged messages partially, as not all the exchanged messages are confidentiality protected. For example, as explained in Figure 4.4, only the Assign message is encrypted rather than both the Assign and Commit message. This is because they assume that no worker, i.e. a Task Tracker, can forge other workers’ signature.

The major difference between the Wei method and the Somu and Rubika methods is that the Wei method ensures the authenticity of (i) messages sent from one MR component to another and (ii) the intermediate and final results produced by the MR components. These protections are provided by using digital signatures. By using the digital signature, the method also provides the property of non-repudiation of origin; protecting against false denial of having generated or transmitted a message. However, as discussed in [47] and [48], a public key cryptosystem is computationally more costly in comparison with a symmetric key cryptosystem, especially when it is applied to a large-scale computational
environment where a large number of jobs may need to be processed and a large number of distributed components are involved [3].

The TMR Framework is similar to the SecureMR Framework. It uses the notion of trust and a public key cryptosystem based authentication method to facilitate the authentication between MR components. The authentication process is carried out in two phases. The first phase is for initial trust (attestation) establishment, and is carried out when an MR component (e.g. a Task Tracker) sends a connecting request to another MR component (e.g. a Job Tracker). The second phase is for periodical trust updates between the Task Tracker and the Job Tracker, and it is carried out regularly during the lifetime of the MR-Job execution. When a Task Tracker first registers with the Job Tracker, it generates an Attestation Identity Key (AIK) pair, i.e. a pair of public and private keys, and the Task Tracker then sends the public key to the Job Tracker. The TMR Framework also ensures the authenticity of the exchanged messages between the Job Tracker and Task Trackers. However, the TMR Framework does support Task Tracker to Task Tracker authentication, i.e. Reducer to Mapper.

4.3.2.3. Zhao's Authentication Method

J. Zhao et al have proposed an authentication method trying to address the clients and MR components' authentication [46]. In this method, a client (i.e. a user) is assumed to be already identified or registered to the MR application, while the master and the slave components of the MR application, i.e. the Job Tracker and Task Trackers, are identified using a Certificate Authority (CA). To authenticate a user, the user logs into the Job Tracker using his username and password. The Job Tracker node contains the login information of the users to verify each user login. The Job Tracker verifies the username and the password once submitted by the user. If the verification is positive, the user will be allowed to submit his job to the MR application and a user instance is created for the user to indicate that the user has an active job. The subsequent authentication between the MR components associated to the client job is achieved using two types of certificates; proxy certificate and slave certificate.

Each user instance maintained in the Job Tracker is issued with a proxy certificate and each Task Tracker issued with a slave certificate. The proxy certificate contains the public key of the Job Tracker and used to authenticate the Job Tracker (linked to this user instance) to a Task Tracker, while the slave certificate contains the public key of the corresponding
Task Tracker and it used to authenticate the Task Tracker to the Job Tracker. Both certificates contain the identity of CA. When the Job Tracker applies for a proxy certificate for each user instance, a secure connection is set up between the Job Tracker and the CA using the Secure Socket Layer (SSL) protocol. In this way, both the CA and the Job Tracker can be authenticated to each other using this protocol. Then the Job Tracker generates a pair of public and private keys for the user instance. The Job Tracker keeps the private key and sends the public key to the CA through the secure channel just established. The CA adds some information, such as key life time, to form the proxy certificate and signs the certificate with the private key of the CA. The proxy certificate is sent to all Task Trackers that are involved in the user instance, and the slave certificates are sent to the Job Tracker.

This certificate-based authentication method provides mutual authentication between the Job Tracker and a set of Task Trackers involved in an MR-Job execution. It support the authentication between such MR components when they are in different cluster domains, and it can mitigate a number of threats such as Man-In-The-Middle (MITM) attack between the Job Tracker and the Task Trackers. Also, as a secure channel is used between the CA on one side and the Job Tracker on the other, the messages sent in the channels are confidentiality and integrity protected. Furthermore, to evaluate the performance of this method, the authors have implemented the authentication method assuming the following usecases: (1) one Job Tracker with one Task Tracker, (2) one Job Tracker with two Task Trackers and (3) one Job Tracker with three Task Trackers. However, the performance evaluation of this method shows a high cost. The execution time taken by the Job Tracker to authenticate three Task Trackers is about the double of the execution time taken to authenticate two Task Trackers. This means that the execution time could be excessively high if the number of MR components, i.e. the Task Trackers, increases to hundreds. The high cost in this method is due to the use of the asymmetric key cryptosystem, the use of the CA which involves more communication overhead to identify the MR components using the proxy and slave certificates, and the use of SSL to securely distribute the certificates.
Table 4.1: Related works versus MR authentication requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Requirements</th>
<th>O. Malley and et al</th>
<th>N. Somu and et al</th>
<th>S. Rubika and et al</th>
<th>W. Wei and et al</th>
<th>A. Ruan and et al</th>
<th>J. Zhao and et al</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>(R1) Entity Identification and Credential</td>
<td>Number of factors as proof of identity,</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(R1.1) Entity Identification (or registration)</td>
<td>Θ</td>
<td>Θ</td>
<td>Θ</td>
<td>x</td>
<td>Θ</td>
<td>Θ</td>
</tr>
<tr>
<td>2.1</td>
<td>(R2) Mutual Entities Authentication protocol between</td>
<td>(R2.1) An MR-Client and an MR master node</td>
<td>√</td>
<td>Θ</td>
<td>Θ</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2.2</td>
<td>(R2.2) An MR component and an MR master nodes</td>
<td>√</td>
<td>x</td>
<td>x</td>
<td>Θ</td>
<td>Θ</td>
<td>Θ</td>
</tr>
<tr>
<td>2.3</td>
<td>(R2.3) Any pair of MR-components</td>
<td>Θ</td>
<td>x</td>
<td>x</td>
<td>Θ</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2.4</td>
<td>(R2.4) Cross Domains (i.e. providers)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Θ</td>
</tr>
<tr>
<td>3.1</td>
<td>(R3) Authentication</td>
<td>(R3.1) Data</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Θ</td>
<td>Θ</td>
</tr>
<tr>
<td>3.2</td>
<td>(R3.2) Exchanged Messages</td>
<td>Θ</td>
<td>x</td>
<td>x</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>4</td>
<td>(R4) Confidentiality of Exchanged Messages</td>
<td>Θ</td>
<td>x</td>
<td>x</td>
<td>Θ</td>
<td>Θ</td>
<td>Θ</td>
</tr>
</tbody>
</table>

\(\checkmark\) - addressed and achieved; x – not addressed or mentioned; Θ – partially addressed or have rooms for improvement

4.4. Other Authentication Methods

In addition to the above authentication methods proposed for the MR application, this section presents a number of authentication methods proposed and used in cloud computing. This includes generic methods proposed to authenticate a client in such context, a location based authentication methods, cryptographic based authentication methods, security token (assertion) based authentication method, and proxy based authentication method.

4.4.1. Generic Client Authentication Methods

There is a number of generic authentication methods proposed in literature to authenticate a client to a remote cloud service. For example, those are proposed in [49] [50] and [51]. A. Yassin et al [49] have proposed a two-factor (2F) authentication method. In
their method, the first-factor authentication secret is a password associated to the client’s username. It presents something the user knows. The second-factor authentication secret is a set of the client biometric features (i.e. biometric information) that presents something the user is. The client biometric information is extracted from the user’s fingerprint. To extract such biometric information, an extra device and digital image processing algorithms are required. In other words, a fingerprint reader (i.e. scanner) is needed to obtain the fingerprint image, and then to recognise the image a number of fingerprint image processing mechanisms, such as the image pre-processing for reducing noise, the low pass filtering, segmentation and core point detection for feature extraction, are required. The user logs into the service using his/her username and password. Once the remote server of the cloud service verifies the first authentication factor, the server requests the user to submit his/her second authentication factor for verification.

R. Jiang [50] has also proposed a two-factor (2F) authentication method, where a user password is used as a first-factor authentication secret. Unlike A. Yassin et al.’s authentication method, the second-factor authentication secret used in R. Jiang’s method is a random number possessed by the client and stored in his smartcard. However, similar to Yassin’s method in that it requires an extra device (in this method, it is a card reader). The card reader is needed whenever the user logs into the service to read the authentication secret from the smart card.

In addition to these two-factor authentication methods, P. Jain et al [51] have proposed a simpler authentication method in term of using multi-factor authentication. They use a One Time Password (OTP), an International Mobile Station Equipment Identity (IMEI) and a secret arithmetic expression as three-factor (3F) authentication secrets. Although, P. Jain et al method uses a three authentication secrets to authenticate a client to a cloud service, it does not required a complicated authentication process such as those in [49] [50]. Also, P. Jain et al’s method is mainly to support the authentication of the clients who are accessing the cloud services from their mobile devices (i.e. mobile phones) in which the power consumption is a concern. These three examples of the authentication methods discussed in this section could provide a stronger client authentication than those use a single-factor authentication secret, hence if one authentication secret has been compromised by an adversary, the adversary has to compromise the other authentication secret/s to impersonate a client.
4.4.2. Location Based Authentication Method

A location based authentication method verifies the identity of an entity (object or user) by identifying its presence at a specific location [52]. Location based authentication could be used to decide whether an entity trying to access a service from an approved location (e.g. a user’s home or office building) is allowed or not. In this method, information about the location of the entity is quite sensitive especially if such information is used as a key attribute to authenticate the entity to a service [53]. In comparison to other authentication methods, in location based authentication method the entity might not be required to provide explicitly an authentication credential during the authentication exchanges.

A number of location based authentication techniques are proposed [54] [55]. These proposals are based on a well-known location based authentication technique called Active badge. A user wears a badge which transmit an infra-red signal contains a unique identifier, while a number of sensors which are located in fixed positions within the building receive this signal, then a software can define the location and authenticate the user. The limitation of this location based authentication technology is that it provides only local authentication. However, it might be used for a remote location authentication, if the unique identifiers of the users, which are received by the sensor, sent to the remote verifiers.

An IP address based authentication is also a location based authentication method and used to authenticate a client remotely [53] [56]. A cloud server checks the IP address of an entity (e.g. client or service) against a set of allowed IP addresses in the database to verify its identity. IP based authentication method requires that the IP address of the object to be fixed if the verification process is individually based. However, organisations typically use IP address based authentication method to authenticate a set of clients (i.e. subnet of IP addresses) rather than individuals. For instance, some of the universities in the UK provide their students an access to their electronic libraries based on the IP addresses of their PCs which have a specific IP range (e.g. the range of IP addresses specified for a campus). The IP address is typically used as an additional authentication factor with other authentication credentials (e.g. secret keys) to provide stronger authentication method. This is because the IP address based authentication generally suffers from IP spoofing if the IP address is used as a single authentication factor [52] [57].
4.4.3. Cryptographic based Authentication Methods

Cryptographic based authentication method refers to a cryptographic secret key that something the claimant (i.e. the user) has. It is stored in either a physical object or in a file object, so-called Hardware Token or Software Token, respectively. This authentication method is considered to be as a multifactor based authentication method. Usually, the cryptographic key which is stored in the Token presents the first-factor authentication secret, and this secret is encrypted by another secret (e.g. PIN or a password) known only to the user and the latter presents something user knows – as a second-factor authentication secret. The user authenticate to the cloud service by proving that s/he possesses the cryptographic secret key and controls it. Smart cards and USB token are examples of Hardware Tokens, while the keystore of the Windows Cryptographic Service Provider (CSP) and the Mozilla/Firefox Web browser keystore are examples of Software Tokens.

When the Hardware Token is used, a public and private key pair is generated inside the token, and the public key is sent to CA to be signed. Once it is signed it is loaded back to the token. The private key never goes out of the token. In order for an attacker to be able to impersonate the owner of the token, the attacker has to possess the Hardware Token and knows the secret which controls the token. However, the deployment of the Hardware Token is tedious and costly. The physical objects of the Hardware Tokens (i.e. devices) are produced by different companies (manufacturers), but there is a lack of standard. Different tokens use different client software-drivers. In other words, for each specific token device connected to a particular Operating System which runs the client machine, a certain client software (i.e. driver/software for the device of the Hardware Token) has to be installed on the client’s machine so the user can use the token. This deployment hinders the wide-scale use or adoption of the Hardware Token. Also, the Hardware Token might face physical attacks if the token is stolen. Example of physical attack is a Simple Power Analysis (SPA) attack by which an attacker uses power consumption patterns to learn the bits values of the secret key. However this attack, involves statistical techniques to extract small differences in power consumption and extract the values of the bits [52].

In the other hand, when the Software Token is used, the keystore of the Windows or the Internet explorer stores the cryptographic credentials (the private key and the public key certificate) in the registry. They are as secure as underlying the OS. The private key and the certificate of the corresponding public key are protected by both the permission of the file
structure provided by the OS and by the password specified by the user of the credential. So if any user account (or malware infects the user account) has the same file access rights and he is capable to access such encrypted files, he is still have to provide the password to read the keys [52].

4.4.4. **Security Token (Assertion) based Authentication Method**

A Security token or an Assertion is a statement contains information about a claimant (i.e. user). It is for exchanging authentication information between two parties, that is between an Identity Provider (IdP), a producer of the security token, and a Service Provider (SP), a consumer of security tokens. The SP, which is requested by the claimant, uses this information to verify the claimant and make a decision to access the available resources. The security token supports a Single Sign-On (SSO) property. SSO property allows a claimant to be authenticated once to the IdP and gain subsequent services (i.e. resources) from different SPs without the need for further authentication. Based on this principal, SAML (Security Assertion Mark-up Language) and OpenID Connect standards are developed.

SAML is an XML-based language used to exchange the authentication information between trusted entities across different domains [58]. The typical SAML authentication process can be broken down into two interactions. The first interaction is between the claimant and the token provider (i.e. IdP). The second interaction is between the service provider and the token provider. The second interaction can be in push or pull mode. In the push mode, after a claimant is successfully authenticated to the token provider, the token provider generates the token and sends it to the service provider through the claimant (through the user browser), and it is used by the service provider to authenticate the claimant. While in the pull mode, the service provider retrieves the token directly from the token provider using a token reference passed to the service provider through the user browser. More technical description and specifications of SAML are available in [58] [59].

The OpenID Connect [60] is an open standard which allows the claimant to be authenticated using a server named OpenID provider or Authentication Server AS (i.e. IdP Identity Provider). It is considered as a cross domain authentication standard. It authenticates a claimant in one domain and propagates that identity to another domain. Also, a claimant can be presented with a number of identities (whereas the identity is a set of
attributes related to a claimant). So each user can have more than one identity to access different sites (i.e. SP) using these different identities [61], as shown in Figure 4.5.

![Figure 4.5 OpenID Connect standard: one user with different identities](image)

The OpenID Connect is an identity layer works on the top (i.e. based on) of OAuth2 standard. This is because the OAuth2 has no notion of identity and it is an access granting (authorisation) protocol and it is known as a delegation protocol. In OAuth2, an Authorisation Server is delegated by a Resource Owner (i.e. user) to grant a third party called an Application Client an access to the user’s profile (i.e. Resource Server) [62]. OpenID Connect standard works with multiple identity providers using JSON Web Token (JWT) [61]. JSON (JavaScript Object Notation) is a lightweight, text-based, language-independent data interchange format [63]. The benefit of this lightweight token is that it can be passed in HTTP headers and query string over different networks, for example 3G network and bandwidth constrained networks.

OpenID token (JSON token) format is akin to SAML token in terms of having information about the user like user attributes, time of authentication (validity of the token) and how the user is authenticated. However, OpenID Connect token is less expressive and it is more compact than using XML token which is used by SAML. Also, OpenID token is a dynamic token. A dynamic OpenID token means that once the information is obtained, the token is continually checked for any changes or updates (e.g. secret keys), while the SAML one is a static token, once the information of the token is obtained, it will be the same till the next login where the new information is sent [64] [65]. Both such tokens, i.e. both OpenID Connect (OAuth) and SAML’s tokens, require a secure connection between the involved parties. They are vulnerable to a number of MITM attacks such as session hijacking attack unless they use SSL/TLS [66] [67].
4.4.5. Proxy based Authentication Method

In addition to the security token authentication method, there is a proxy credential authentication method which is commonly used in inter-domain environment. The idea of using proxy credentials is described in details by V. Woelch et al [68] [69]. The proxy credential consists of proxy certificate and its private key. It allows a client to be authenticated once without the need to be re-authenticated whenever a process or service on behalf of the client needs to access further resources. This mechanism can be summarised in two stages. The first stage is generating a proxy certificate, and the second stage is delegating the proxy certificate over the network (by creating a new proxy certificate, signed by the private key of the first). The first stage is explained in Figure 4.6 and summarised as following:

- A key pair (a private and a public key) is generated for the use of a new proxy credential.
- The proxy certificate is formed.
- The private key of the client is used to sign the proxy certificate (this might require re-authenticate the owner of the proxy certificate (i.e. the client)).
- Finally the proxy certificate and its corresponding private key are stored in the file system.

![Figure 4.6 Creating proxy certificate](image)

As mentioned above, this proxy credential (i.e. the proxy certificate and its corresponding private key), provides the client with a Single Sign-On property. This property is achieved by delegating a proxy certificate over the network (i.e. the second stage). The proxy credential generated by the client is used to delegate the privilege to other party (e.g. server A) without the need of exchange the private key over the network. The steps of such delegation process is summarised in the following points as shown in Figure 4.7.
- First, a secure connection is established between the client and the server A.
- If the client chooses to delegate its privilege to server A, the later node generates a new key pair, private and public key.
- A request, contains the generated public key by the Server A, is sent to the client via the established secure channel.
- The client generates a new proxy certificate containing the public key generated by server A and signs it using the private key associated with its own proxy certificate (i.e. private key of the client’s proxy certificate).
- Then the signed certificate is sent back to the server A, and it is saved with the newly generated private key.
- Then the new generated proxy certificate can be used by the server A for accessing other services (e.g. server B) on behalf of the client.

![Diagram of delegation process using a proxy credential](image)

**Figure 4.7 A delegation process using a proxy credential**

In fact, the idea of using the proxy credential is based on SPX solution proposed by Joseph and Kannan in 1991 [70]. SPX is a reference for a global authentication method using public and private key mechanism. SPX is an authentication method that provides a delegation feature across two different domains. The SPX uses a data structure named Ticket for delegation rather than proxy certificate. The ticket and its associated private key are used to provide the means of authentication in a similar way as for proxy certificate. Though, the SPX shares many concepts with X509 public key certificate standard, the SPX has its own data structure of the ticket and it operates independently of X509 standard. Also, SPX ticket
format offers impersonation mode, i.e. full delegation to other entities (servers). In contrast, the proxy certificate is based on X.509 public key certificate standard that allows reuse of the existing authentication procedures and software developed for the public key certificates. Proxy certificate also provides format which can offer a limited delegation, i.e. subset of the client’s privileges can be delegated to the other entities by using more than one proxy certificate.

There are challenges of using a proxy based authentication method rather than a security token based authentication method. Although, the former one does not require the involvement of a third party (e.g. AS) as the security token based authentication method does, the proxy based authentication method is less secure [52]. This is because proxy credentials are stored in an ordinary and unencrypted local file format. Proxy credentials are protected just by a basic file permission system provided by the Operating System (OS). So if any user account (or a malicious user account) has the same level of permission could have access to those credentials. Consequently, these credentials are more vulnerable to abuse and theft.

4.5. Chapter summary

This chapter has critically analysed authentication methods used or proposed for MR application. The analysis is guided using a set of MR authentication requirements which are specified based on the GMC model. In additional to this analysis, this chapter has presented an overview of other authentication methods used to support cloud computing, identifying their strengths and limitations.

The critical analysis of the MR authentication methods shows that some methods are designed to support gate-level authentication (i.e. the authentication of users or clients to the MR application), while others only protect the integrity and the origin of protocol messages and data sent among different MR components. Though there are efforts supporting mutual authentication between different MR components (i.e. inter-components), these efforts are largely based on the use of public key credentials, i.e. their authentication methods are asymmetric key based. Asymmetric key based approach require the involvement of a third party (e.g. CA) to provide a secure connection to facilitate inter-component authentication. The costs incurred using such approach are typically high in comparison to symmetric key based approach. These methods do not support the mutual
authentication between the MR components requesting the clients’ job resources (i.e. data) and the DFS (i.e. the Name Node and Data Nodes). Further, there are some methods consider both gate-level and the inter-components authentication, but these methods are based on a weak single-factor authentication secret, i.e. a user password, and the inter-components authentication level is based on the gate-level authentication. If the gate-level authentication is compromised the inter-components will be compromised as well.

By investigating current MR authentication methods, it can be concluded that these MR authentication methods are not effectively applicable for the MR application. This is because of the following observations: First, none of the existing MR authentication methods adequately captures the characteristics of such distributed computing framework that are discussed in Chapter 1, thus none of them meets all the functional MR authentication requirements specified in Section 4.2. Second, the existing MR authentication methods, which support gate-level authentication, are either unilateral authentication and vulnerable to the Man-In-The-Middle (MITM) attack, as they do not provide mutual authentication between the client and the MR application, or assuming that a client is known to the MR application at the gate level where the client is already identified (i.e. registered) to the MR application. Thus, these methods are not efficient for remote implementation of MR application as the client should be identified securely and remotely. Third, most the existing MR authentication methods assume that communication between the client and the MR application and among the MR components is in a trusted network, yet, as it is observed, this is not always the case for the MR application where these entities could communicate over WAN. Though some existing MR authentication methods support MR components authentication over WAN, these methods are inefficient in such context (distributed computing framework) as they use asymmetric key based approach which introduces a high cost. Fourth, some of the existing MR authentication methods try to provide authentication for both the gate and inter-component authentication levels, however, they use a single-factor authentication secret, i.e. a user password, for the gate authentication level while the inter-components authentication level is based on the former. If the authentication of a user or client to the MR application is compromised (i.e. the password is compromised) the inter-components authentication will be compromised, thus the job execution.

The lack of an adequate and effective authentication service design for the MR application that capture the very characteristics of MR application will make the MR application vulnerable to security threats and attacks. The threats and attacks are not just
those in relation to identity thefts, impersonation or replays attacks. A successful compromise of a weak client credential, i.e. a user password, of an MR application will give attackers a better chance to launch further attacks, gaining unauthorised access to data and/or interrupt other job executions.

Thus, the following chapter introduces our vision of designing a novel authentication solution that captures the characteristics of MR application to address such issues.
Chapter 5

5. A Novel Virtual Domain based Authentication Framework (VDAF)

5.1. Chapter Introduction

This chapter discusses a way forward by presenting ideas that help to design an authentication solution satisfying MR authentication requirements. It can be seen from the discussions carried out in chapters 3 and 4, the current state of authentication in MR application is unclear. In other words, MR authentication proposed using different authentication methods do not provide a vision or an architecture design of MR authentication to support a secure MR-Job execution. This could be due to the characteristics and complexity of the MR application. As a result applying inadequate authentication service for MR application will make the MR application vulnerable to security threats and attacks. In order to design an MR authentication architecture or solution that recognises the authentication provisions and tackle the complexity of such task, there is a need to investigate the functionality of MR components and the interactions among them during the MR-Job execution. Our concept in tackling such challenge is that if the MR components could be classified based on their roles and interactions, different layers of authentication domains could be created. Based on both this concept and the analysis conducted on the GMC model, a novel authentication model (called MR Layered Authentication Model (MR-LAM)) to address this challenging task is proposed.

MR-LAM supports the most recent implementation of MR model and captures the characteristics of a shared and distributed computational environment and it realises the whole task of authentication for the MR application. MR-LAM consists of two layers of authentication. The first one is for the authentication of components that serve multiple jobs submitted by different clients and so called MR-Infrastructure domain authentication layer. The second one is for the authentication of the components that are invoked specifically for a particular job and so called MR-Job domain authentication layer. An authentication framework, called a Virtual Domain based Authentication Framework (VDAF), is designed. VDAF imposes authentication control at every request made for a job resource during an MR-Job execution. To achieve this, a new novel authentication method called a Password and Token-based Multi-point Multi-factor Authentication (PT2M-AuthN) method is
proposed, and this method implements two main ideas. The first one is a principle of separation of duty-and-credential, and the second one is a key wrap-and-swap operation to support mutual authentication. By implementing these ideas, (1) the authentication functions and the issuance of the credentials are distributed among a distributed set of MR components, so-called MR-Authentication (MR-AuthN) Components, (2) any interaction point made by an MR-Client or an MR component requesting access to a job resource is identified using two-factor authentication secrets (i.e. keys), and the first authentication key is (wrapped) and swapped with the second authentication key to provide a secure mutual authentication for this interaction. To facilitate the VDAF functionality using these ideas, a set of protocols called Lightweight VDAF Authentication Protocol (LVAP) is proposed.

In detail, Section 5.2 introduces a high-level analysis of the MR application, highlighting the functionalities of and the interactions between the MR-Client and MR components and among MR components in the MR-Job execution. In Sections 5.3 and 5.4, the idea of using the layered approach to authentication for MR application, i.e. MR-LAM is presented. Section 5.5 introduces VDAF. Sections 5.6 and 5.7 present the design preliminary, the authentication method, ideas and measures used in the VDAF. Section 5.8 discusses the design of the VDAF architecture covering its functionality and components. Then LVAP suite is presented in Section 5.9, and the cryptographic building blocks to be used in the LVAP suite are given in Section 5.10. Finally, Section 5.11 summarises this chapter.

5.2. A High-Level Analysis of an MR-Job Execution on GMC Model

From the GMC model discussed in Chapter 2, it can be seen that when executing a job submitted by a client, multiple MR components are involved. Each component executes a well-defined function and the multiple components interact with one another to collaboratively accomplish the job execution [71]-[73]. The MR components, either of one job or multiple jobs submitted by a single or multiple clients, are hosted in two shared clusters, Data Processing (DP) cluster and Distributed File System (DFS) cluster. Each interaction between a pair of MR component is a client-server interaction and should be authenticated. Examples of these interactions are those initiated to the Name Node. Each request is a procedure call. These calls are for reading and writing a job-resource. In other words, these calls for (i) reading the client’s data and typically initiated by a Job Tracker or a Task Tracker, or (ii) writing data submitted by a client or produced from a job execution
(e.g. input and output data of a job) and such calls initiated typically by clients and Reduce Tasks, respectively. Other examples of the client-server interactions are those initiated to and from the Resource Manager. When a client submits a new job, he needs to make a new job-submission request. Upon the receipt of the job-submission request, the Resource Manager needs to make a resource-allocation request to a Job Tracker. In other words, all these calls are for (i) submitting a new job and typically initiated by clients, or (ii) allocating a Job Tracker to master the Map and Reduce Tasks, which are assigned to different Task Trackers related to the job, and these calls (i.e. (i)) are typically initiated by the Resource Manager. Depending on their functionalities, the interactions involved in an MR-Job execution can be classified into three groups: (i) those for submitting a job (Group-1 interactions), (ii) those for allocating MR resources for the execution of the job (Group-2 interactions), and (iii) those for accessing the job-resources (Group-3 interactions).

Group-1 interactions take place when a client submits a job to the MR application. The client submits the job via the Resource Manager, and as part of this submission process, the client also writes the data for the job execution via the Name Node. The Resource Manager is the master node of the DP cluster, and the Name Node is the master node of the DFS cluster. These two components are static MR components.

Group-2 interactions are for allocating MR resources. Three MR components are involved in this group of interactions, the Resource Manager, the Name Node and a Job Tracker. When a job is admitted, the Resource Manager allocates a Job Tracker for managing the job execution. This Job Tracker is assigned multiple Task Trackers. The Map and Reduce tasks of this job are executed on these Task Trackers. The Group-2 interactions also include the interactions carried out by the Name Node to manage and maintain a set of Data Nodes. The Data Nodes host data for all the jobs that are submitted. The functionalities of the Resource Manager and the Name Node are for managing the executions of all the jobs that are submitted to the MR application. The Resource Manager and the Name Node are shared by different jobs and are identified by static identities. These two components are not invoked because of a job submission; they are there to serve every job submitted by any client. It is for this reason, it is referred to these two components as MR Infrastructure components (i.e. MR-Inf. Components for short).

Group-3 interactions are generated when a job is being executed. These interactions are for requesting the job resources and they are performed (i.e. initiated) by four types of MR components, the Job Tracker (JT), Task Trackers (TTs), Map Tasks (MTs) and Reduce Tasks
The JT retrieves the input splits of the data submitted by the client from the DFS. The JT can then start managing the tasks (MTs and RTs). TTs retrieve the input data for the job execution from the DFS. TTs can then start executing the MTs and RTs assigned to them. While executing the tasks, RTs retrieve the intermediate data, which is produced by MTs, from the respective Task Trackers. RTs also write the output results of their computations into the DFS. Group-3 interactions are job dependent as they are invoked for a particular job. This set of components are created or invoked when the job is submitted, and are terminated or reassigned when the job execution is completed. The existence of this set of components is purely for serving a particular job. Therefore these components are regarded as dynamic and identified by dynamic identities. Their identities are short-lived. For these reasons, it is referred them as MR-Job Components.

It should be emphasised that one client may submit multiple jobs, and there will be multiple clients submitting jobs. The interactions taking place for the submission and execution of a single job are regarded as one set, and one such set of interactions consists of the interactions from all the three groups, i.e. \{the set of interactions for the execution of one job\} = \{a set of Group-1 interactions\}+\{a set of Group-2 interactions\}+\{a set of Group-3 interactions\}, where all the sets are related to the submission and execution of this particular job. Further examples can be used to explain the interactions of these three groups. The Group-1 interactions are for submitting a job and these interactions are those of MR-Client to Resource Manager (C-RM), and MR-Client to the Name Node (C-NN). The Group-2 interactions are for allocating MR resources and these interactions are those of Resource Manager to Job Tracker (RM-JT), Job Tracker to Task Tracker (JT-TT), and Name Node to Data Node (NN-DN). These Group-2 interactions are different from Group-1 and Group-3 interactions in that the Group-2 interactions do not involve any access (read, write or retrieve\(^4\)) of a job resource. They are performed for accomplishing cluster functions and for serving the execution of the job. Both Job Trackers and Data Nodes are, respectively, the slave nodes of DP and DFS clusters. Any interaction initiated by a cluster master node (RM or NN) towards a cluster slave node (i.e. RM-JT, or NN-DN), or from a slave node of the DP cluster to another slave node in the same cluster (i.e. JT-TT), does not involve any job resource access. In the other hand, Group-3 interactions are for executing the job and they are identified as follows: Job Tracker to Name Node (JT-NN), Task Tracker to Name Node (TT-NN), Reduce Task to Task Tracker (RT-TT), and Reduce Task to Name Node (RT-NN).

\(^4\) Retrieve involves both read and write; read from remote server (DN) and write locally to another server (TT).
It should also be emphasised that, different from Group-2 interactions, Group-3 interactions involve the access of resources related to the job for which the associated components are created. To ensure authorised access to a resource associated to a particular job, the Group-3 interactions of the job should be effectively authenticated.

5.3. A Layered Approach to Authentication

What can be learned from the high-level analysis of a job execution on the GMC model are two-folds. Firstly, the MR model implementation contains two clusters, and each cluster contains multiple MR server nodes. The nodes collectively host different MR components that are involved in an MR-Job execution. As a set of MR components, hosted on multiple MR server nodes, are involved in an MR-Job execution, both the number of MR components and the membership of the MR server nodes are job-dependent, i.e. different jobs may be assigned with different numbers of MR components and the MR server nodes hosting the assigned MR components may be different. This indicates that there is a need for MR server node-level authentication so that when a new MR server node joins in a cluster and/or when a set of MR components is assigned to a job, the new MR server node/s that host the set of MR components assigned to a job can be authenticated. This MR server node-level authentication is hereafter referred to as MR-Inf. domain authentication.

Secondly, as described above, when a new job is submitted, a set of MR components is assigned to execute the job, i.e. MR-Job Components. Though this set may share MR server nodes with other sets of MR-Job Components of other jobs, the interactions carried out (between the MR components which includes both MR-Job Components and MR-Inf. Components) in relation to the execution of this job, in particular for those leading to access to resources associated to the job, can only be carried out (i.e. initiated) by this set of MR components. Therefore, to protect the data of a job against unauthorised access requires that such interactions to be authenticated. For this reason, the set of MR components that have been assigned to or involved in executing a job are classified into a group, and give this group a name, the MR-Job domain. This MR-Job domain is job dependent, and all the interactions carried out (initiated) by the MR-Job components in this domain to access to the resources associated to the job (e.g. input data), should be authenticated. This authentication is hereafter referred to as MR-Job domain authentication.

The above discussions actually indicate our two novel ideas to the complex task of the authentication of the MR-Job execution, a layered approach to authentication of MR.
application and a virtual job domain based approach to authentication of job specific MR components and their interactions. Putting these two ideas together has led to the design of our novel authentication model, the MR Layered Authentication Model, as shown in Figure 5.1.

5.4. MR Layered Authentication Model

MR Layered Authentication Model (MR-LAM) consists of two authentication layers, MR-Inf. domain authentication layer (Layer-1) and MR-Job domain authentication layer (Layer-2), see Figure 5.1.

The MR-Inf. domain authentication layer serves the authentication of the MR clusters’ server nodes, i.e. MR master and slave nodes. It is responsible for the mutual authentications between (i) any new MR server node (typically an MR slave node joining in an MR cluster) and the MR master node, and (ii) any pair of MR slave nodes in the cluster. In other words, an authentication service for this layer should support the authentication of any new MR slave nodes wanting to join the MR-Inf. domain. An MR slave node should only be admitted to becoming a member of the MR-Inf. domain if the MR slave node has been successfully authenticated. For example, considering the case shown in (Figure 2.5), if many jobs are submitted and the Resource Manager in the DP cluster is running out of resources on the MR slave nodes (e.g. Task Tracker node) and/or if the IT administrator, looking after the cluster, has decided to bring in a new MR slave node into the cluster, then the new MR slave node should be authenticated before it is allowed to become a member of the MR-Inf. domain and connecting other MR server nodes in the domain. This authentication task could be achieved using the Kerberos authentication solution. The Kerberos solution is preferred for the MR-Inf. domain authentication. This is because the Kerberos is widely used as an authentication service for and supported by the clusters’ infrastructure components, e.g. Microsoft windows OS and Red Hat OS, [74]-[76]. Basically, if such authentication method is used as the default mode of the authentication service in a cluster, all the MR slave nodes (i.e. all the members of the MR-Inf. domain) in the cluster should support this mode of authentication service. They will use Kerberos to authenticate themselves to the master node in the cluster and to establish shared secret keys between them. However, the clients who have their jobs admitted into the MR application are not members of the MR-Inf. domain themselves, but they have to be authenticated before their jobs could be admitted. A client will be a part of an MR-Job domain at Layer-2 as he submits the job. The client will be identified to the MR application
and authenticated by the authentication service provided at Layer-2. In other words, the second layer of the authentication model, the MR-Job domain authentication layer, should be able to provide means for clients' authentication.

Figure 5.1 MR layered authentication model (A layered approach to authentication)

The MR-Job domain authentication layer is responsible for two mutual authentications: (i) between each client submitted a job and the MR application and (ii) among the MR components that are serving the job and accessing its resources. For this layer, an idea of a virtual job domain based approach is used to authentication of job specific MR components and their interactions. The principle behind this approach is to isolate the client along with the MR-Inf. Components and MR-Job Components, which are involved in executing the given job, into one set, and require this set of components to authenticate to each other, so that only the components in this set (including the client) are allowed to access the resource belonging to (or owned by) this particular job and any component outside this domain is not allowed to access to the job resource. To implement this idea, a novel authentication framework, named as the Virtual Domain based Authentication Framework (VDAF), is proposed. This framework is said to be virtual domain based, because (1) MR-Job Components are dynamic as they are created or invoked when the job is submitted and terminated or reassigned when the job is completed, (2) more than one MR-Job domain may co-exist at any one given time in an MR-Inf. domain, and (3) the MR-Job Components of these MR-Job domains work on the top of another group of components –MR server nodes. At this layer, each new client is registered with an Authentication Server (AS) so as to be identified to the MR application. Upon successful registration, the AS issues a long-term authentication credential to the client that identifies him to the MR application, so-called MR-Client, and thus the MR-Client can use this credential to submit his job to the MR application. During the execution of this job, the client would be able to make use of the MR
components as long as these components are assigned to the job. Also, at this layer, the authentication method and protocols used by each MR-Job domain are expected to be the same, but the secrets used in each such domain are different and they should be protected against exposure to other domains. As mentioned earlier, each MR-Job domain has its own MR-Job Components involved in the execution of the job submitted by the client. The client generates and manages the credentials (authentication secrets and other data) used to authenticate the MR-Job Components in this domain (details to follow in Section 5.8).

In other words, this layer is responsible for providing the identification and authentication service by which an MR-Client submits a job and MR-Job Components assigned to the job can be securely identified and authenticated, and do so at every interaction between the MR-Client and MR-Inf. Components, between MR-Job Components and MR-Inf. Components and among MR-Job Components throughout the execution cycle of the job. The Resource Manager, which manages the resources of the MR-Inf. domain, is involved in the authentication of all the MR-Job domains. For the MR-Job domain authentication, the Resource Manager works as a relay to deliver the authentication credentials of each MR-Job domain to the MR-Client and to the MR-Job Components in the MR-Job domain.

Furthermore, each of the MR components (either an MR-Inf. Component or an MR-Job Component) has an authentication module. These authentication modules, depending on the hosting MR components, can be respectively named as Job Tracker Authentication (JT-AuthN) Modules, Task Tracker Authentication (TT-AuthN) Modules, Map Task Authentication (MT-AuthN) Modules, Reduce Task Authentication (RT-AuthN) Modules, Name Node Authentication (NN-AuthN) Module, Data Node Authentication (DN-AuthN) Modules, and Resource Manager Authentication (RM-AuthN) Relay Module. By embedding these authentication modules into their respective MR components, MR-Inf. Components and MR-Job Components can be supported with the authentication among themselves and prevent unauthorised access to resources of a particular job domain.

5.5. MR-Job Domain Authentication Layer: VDAF

MR-LAM uses a layered approach to authentication. The model takes into account of the authentication requirements of all the components of the MR application. Based on this, five aspects of authentication needs for the entire job execution cycle in the MR application are identified. They are client identification, MR-Job Components identification, MR-Client authentication, MR-Job Component authentication, and MR-Data authentication. VDAF is
designed to satisfy these authentication needs. Before describing VDAF, the preliminary used in the VDAF design is introduced.

5.6. VDAF Design Preliminary

The design preliminary covers the threat model, assumptions and requirements, used in the VDAF design.

5.6.1. Threat model

The threat model focuses on authentication attacks mounted against any identification requests made by remote clients or any job-resource requests made during the MR-Job execution, i.e. from the moment when the job is submitted by the client to the moment when it is successfully completed by the MR application. In detail, the following threats and attacks are considered in the threat model.

- **External clients are not trustworthy**: Any client may try to identify himself and/or submit a job to the MR application.

- **Internal clients are curious**: Clients that have been identified to the MR application, i.e. MR-Clients, follow the protocol specified to get their jobs executed. However, they may be curious in that they may try to access a job resource which they are not allowed to access.

- **MR-Client impersonation threats due to weak credentials**: Attackers or clients may try to get hold of another MR-Client’s authentication credential in an attempt to impersonate the MR-Client and to gain access to its job resource. They may use methods such as a brute force or dictionary attacks to guess other MR-Clients’ credentials (e.g. passwords). A weak authentication credential makes such attacks more effective compromising the authentication process and putting MR-Clients’ job resource at a higher level of risks.

- **MR components serving other jobs (i.e. MR-Job components) are considered as untrustworthy**: As mentioned early, when a job is submitted, a set of MR components is assigned to execute the job. Any other components external to this set are considered as untrustworthy. They may try to gain access to credentials or a resource of this job.

- **Threats to data and message integrity**: Modification and forgery attacks may be mounted on clients’ data and/or protocols messages used to identify and authenticate an
entity (e.g. a client or MR component) in an MR-Job domain. If the MR-Client’s data are tampered with, the results of a job execution may be incorrect. Similarly, if a protocol message is tampered with or forged, an entity identification and authentication may be disrupted thus the job execution is disrupted in an MR-Job domain.

- **Eavesdropping attack**: Passively listens to a communication channel between a claimant (e.g. client to be identified, MR-Client, or MR-Job Component) and a server (e.g. AS or MR-Inf. Component) intercepting and obtaining valuable data, such as an identification or authentication information (e.g. crypt key or password), which then may be used by the attacker.

- **Replay attack**: An attacker intercepts and resends a protocol message in an attempt of compromising an authentication process.

- **MITM attack**: An attacker may impersonate a claimant to a server (verifier) or a server to a claimant during an authentication process in an attempt of compromising the process. The attacker may mount such attacks via eavesdropping the channel and intercepting and replaying protocol messages.

- **Identity spoofing and impersonation attack**: An attacker impersonates a claimant to gain access to a job resource. The attacker can successfully impersonate a claimant if it can get hold of an authentication credential of an identified claimant.

### 5.6.2. Assumptions

This section details the assumptions used in the design of VDAF. As VDAF uses a domain-based approach to authentication, some of the assumptions are made with reference to the two domains, the MR-Inf. domain and the MR-Job domain discussed in Section 5.3.

**A1** The MR application is supported by two clusters. One is the DP (Data Processing) cluster and the other is the DFS (Distributing File System) cluster. The DP cluster provides computing capacities for the MR application by hosting multiple computing servers. The DFS cluster provides storage capacities for the MR application by hosting multiple storage servers.

**A2** The DP cluster is managed by the cluster manager (i.e. the Resource Manager). Access to DP is always via the Resource Manager.

**A3** The DFS cluster is managed by another cluster manager (i.e. the Name Node). Access to DFS is always via the Name Node.
The MR server nodes in these clusters, i.e. MR components of the MR-Inf. domain, are assumed to be in one MR-Inf. domain, and they have already authenticated to each other and an infrastructure domain-secret key is established and shared between them.

The Authentication Server, Resource Manager, and Name Node are fully trustworthy.

Authenticated MR-Clients should not deceive or collude with any attacker.

Time should be reliable and trustworthy, and the clocks of both the Authentication Server and all MR server nodes of the MR-Inf. domain are synchronized.

5.6.3. Requirements

This section specifies a set of requirements for the VDAF design. These requirements are divided into functional, security, and performance requirements.

5.6.3.1. Functional Requirements

VDAF is designed to satisfy the following functional requirements.

F1 Achieve identification of a remote client: VDAF should be able to identify each new client that makes request to the MR application and this identification service should support remote clients. For remote clients, it should not impose the requirement of physical appearance by the clients.

F2 Achieve authentication of a remote MR-Client: Each MR-Client should be authenticated securely before the access of the MR application is allowed.

F3 Achieve identification of an MR component involved in a job domain: This includes the identification of a set of MR components that are assigned to a particular job (i.e. MR-Job Components) and the issuance and distribution of authentication credentials to this set of components.

F4 Achieve authentication of MR components involved in a job domain: Each MR component assigned to the client job (MR-Job Component) should be authenticated securely before the access to a client’s job resource is allowed.

F5 Achieve data authenticity during a job execution process: This is to ensure the origin authentication and integrity protection of data provided to or produced by a job
execution. In other words, the protection should be applied to input data, intermediate data and output data of any job.

5.6.3.2. Security Requirements

This section specifies a set of security requirements for the VDAF design. These requirements are based on the (1) specified set of VDAF functional requirements, (2) performed threat analysis, and (3) high level analysis of an MR-Job execution on the GMC model.

S1 Entity Authentications: Entity authentication is to ensure that an entity is the one that it claims to be. VDAF should address the following entity authentication requirements:

S1.a. Mutual Authentication between an MR-Client and the MR application: This is to ensure that only an authorised client can connect to the MR application. This should cover the mutual authentication between an MR-Client and both the Authentication Server and the Resource Manager and between the MR-Client and the DFS cluster.

S1.b. Mutual Authentication between an MR-Job Component and the DFS: This is to ensure that all the MR-Job Components involved in the execution of an MR-Client’s job are authenticated to both the Name Node and the respective Data Node, so as to ensure that any access to the MR-Client’s job resource can be granted in a secure manner. More specifically, the authentication requirement here covers the mutual authentications between the Job Tracker, each Task Tracker and each Reduce Task and both the Name Node and the respective Data Node, in a job domain.

S1.c. Mutual Authentication between any pair of MR-Job Components: This is to ensure that any access to MR-Client's job data (intermediate data) is granted in a secure manner. More specifically, this authentication requirement covers the mutual authentication between a Reduce Task and a Task Tracker in a job domain.

S2 Authenticity of Protocol Messages: The origin authentication and integrity protection should also be applied to all the protocol messages (requests and responses) that facilitate the tasks of identification and authentication in the MR application (i.e. F1 to F5).

S3 Confidentiality of Protocol Messages: The confidentiality of entity identification and authentication data carried in protocol messages should be protected.
5.6.3.3. Performance Requirements

In addition to the functional and security requirements, the design should also satisfy performance requirements to make the design as efficient as possible. In terms of performance, the following requirements are considered.

**P1 The communication overhead should be as low as possible:** In accomplishing the functions defined for VDAF, the communication overhead introduced should be as low as possible. This means that both the number of authentication messages and the length of each message should be as low as possible.

**P2 The computational overhead should be as low as possible:** The computational overhead incurred in accomplishing the functions of the VDAF should be as low as possible. This means that computational costs of any algorithms and/or cryptographic primitive chosen for the implementation of the VDAF functions should be taken into account.

5.6.4. Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i$</td>
<td>user $i$</td>
</tr>
<tr>
<td>$PW_{U_i}$</td>
<td>password of $U_i$</td>
</tr>
<tr>
<td>$USB_{U_i}$</td>
<td>removable storage media (a USB flash drive) owned by $U_i$</td>
</tr>
<tr>
<td>$CH_{U_i}$</td>
<td>client host/machine of $U_i$</td>
</tr>
<tr>
<td>$AC_{U_i}$</td>
<td>MR application client of $U_i$</td>
</tr>
<tr>
<td>$ACR_{U_i}$</td>
<td>authentication credential of $U_i$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>the $i^{th}$ MR-Client, it represents ${U_i + CH_{U_i} + AC_{U_i}}$</td>
</tr>
<tr>
<td>$ACR_{C_i}$</td>
<td>authentication credential of $C_i$</td>
</tr>
<tr>
<td>$L_{ACR_{C_i}}$</td>
<td>local part of $ACR_{C_i}$</td>
</tr>
<tr>
<td>$R_{ACR_{C_i}}$</td>
<td>removable part of $ACR_{C_i}$</td>
</tr>
<tr>
<td>$ID_{C_i}$</td>
<td>identity of $C_i$ (an MR-Client username)</td>
</tr>
</tbody>
</table>
respectively, 1st primary authentication and verification tokens of \( C_i \)

respectively, 2nd primary authentication and verification tokens of \( C_i \)

respectively, 1st secondary authentication and verification tokens of \( C_i \)

respectively, 2nd secondary authentication and verification tokens of \( C_i \)

the \( j \)th job domain (i.e. the \( j \)th MR-Job domain)

an MR-Job Component of the \( j \)th MR-Job domain that belongs to \( i \)th MR-Client

a Job Tracker of the \( j \)th MR-Job domain that belongs to \( i \)th MR-Client

a set of all Task Trackers of the \( j \)th MR-Job domain that belongs to \( i \)th MR-Client

a set of all Reduce Tasks of the \( j \)th MR-Job domain that belongs to \( i \)th MR-Client

respectively, primary authentication and verification tokens of \( JT_j^i \)

respectively, primary authentication and verification tokens of \( TT_j^i \)

respectively, primary authentication and verification tokens of \( RT_j^i \)

keyed-hash message authentication code of \( Msg \) using \( k \)

symmetric encryption of message \( Msg \) using key \( k \)

asymmetric encryption of message \( Msg \) using key \( k \)

symmetric decryption of message \( Msg \) using key \( k \)

asymmetric decryption of message \( Msg \) using key \( k \)

5.6.5. Evaluation Metrics

The following evaluation metrics and methods are used to evaluate the security and the performance of the VDAF design.
5.6.5.1. Performance Evaluation

**Computation overhead cost:** This cost is measured in term of (1) the number of each type of cryptographic primitives or algorithms used, and (2) the processing time introduced by each entity that are needed to perform the VDAF functions F1 to F5.

**Communication overhead cost:** This cost is measured in term of the number and length of each protocol messages that are needed to perform the VDAF functions F1 to F5.

**Protocol Execution Time (PET):** PET is defined as the time elapsed between the time when the execution of a protocol is invoked and the time when the execution finishes. This includes the protocols that are to be designed to accomplish the VDAF functions F1 to F5.

5.6.5.2. Security Evaluation

The security evaluation of VDAF is carried out using the following two methods.

**Informal Security Analysis:** The VDAF protocols are analysed against threats and attacks in relation to the authentication.

**Formal Security Analysis:** In addition to the informal security analysis, the VDAF protocols are also formally verified. The formal verification of a security protocol can help to discover subtle security flaws or errors in the protocol design which may not be discovered by using the informal analysis method [77] [78].

5.7. VDAF Design: Ideas and Measures

VDAF uses a novel authentication method called a Password and Token-based Multi-point Multi-factor Authentication (PT2M-AuthN) method. This method implements two ideas as following:

The first idea is to use the principle of separation of duty-and-credential. This idea means that (i) no single entity is responsible for an authentication function in an MR-Job domain, and (ii) each point of interaction which requires access to a job resource, either from MR client or component, is identified and authenticated using two-factor authentication secrets. These two-factor authentication secrets are supplied by two different and trusted entities and used to form an authentication token. There are four authentication tokens to represent an MR-Client’s authentication credential, and two authentication tokens to represent the
authentication credential of an MR-Job Component. This is achieved using the following measures and additional ideas.

1. A set of distributed components called MR-Authentication (MR-AuthN) Components is used to collectively perform the VDAF authentication service. This set of MR-AuthN Components, one in each MR component, carries out authentication credentials’ distribution and verification. As VDAF enforces authentication on every interaction between components in an MR-Job domain, the authentication credentials used in later interactions are distributed in early interactions, thus reducing the chance of creating a performance bottleneck in the system. In this way no additional interactions are created for credentials distribution or for processing an authentication request.

2. For each user login to the MR Application Client running on the client machine, there is a user authentication credential, $ACR_{U_i}$. $ACR_{U_i}$ contains a unique MR-Client username and password to identify the user login to the MR Application Client locally.

3. For every point of interaction which involves a job-resource request there is an authentication token created, and each such token contains two authentication secrets (i.e. keys) that are supplied by two different entities.

4. Authentication tokens are classified into primary and secondary authentication tokens. The secondary authentication tokens are generated based on the primary authentication tokens. For each of these authentication tokens there is a corresponding verification token which is used to verify the authentication token against.

5. Each verification token is distributed through a channel which is different from the one used to distribute the authentication token.

6. Each MR-Client ($C_i$) is issued with an authentication credential ($ACR_{C_i}$), and each MR-Job Component ($JC$) involved in executing the $j^{th}$ job of the $i^{th}$ MR-Client (i.e. $JC^j_i$) is issued with an authentication credential ($ACR_{JC^j_i}$).

$ACR_{C_i}$ consists of four authentication tokens, two primary and two secondary:

- The two primary authentication tokens ($PT^1_{C_i}$ and $PT^2_{C_i}$) are typically long-term tokens, and they are generated through the client identification to the MR application (details to follow). In the design of $PT^1_{C_i}$ and $PT^2_{C_i}$, three ideas are used: the first one is to generate a one-time dynamic key that is valid for one user login session, and it is involved in the user login, and it is used to protect long-term authentication secrets which constitute $PT^2_{C_i}$, the second one is to construct
local and removable parts of $ACR_{Ci}$ which includes $PT_{Ci}^1$ and $PT_{Ci}^2$, and store the two parts into two separate storage media (local (in $CH_{ui}$) and removable (in $USB_{ui}$) storages), and the third idea is to conceal the local part of $ACR_{Ci}$ using a stego-object (i.e. digital image file). These three ideas are used to enable a strong protection for $PT_{Ci}^1$ and $PT_{Ci}^2$ as they identify the MR-Client to the MR application and to support a strong MR-Client authentication to MR application (details to follow in Chapter 6). $PT_{Ci}^1$ and $PT_{Ci}^2$ are used to authenticate (i) the user to the MR Application Client locally, and (ii) the MR-Client to the Resource Manager and the AS remotely.

- The two secondary authentication tokens ($ST_{Ci}^1$ and $ST_{Ci}^2$) are short-term tokens, valid for one job only. They are generated when a client is submitting a job, and they are used to authenticate (i) the MR-Client to the Name Node and (ii) the MR-Client to the respective Data Node.

$ACR_{JCj}$ consists of two authentication tokens, one primary and one secondary. Both are short-term tokens –valid for the job submitted. The primary token ($PT_{JCj}$) is generated when the client job is being submitted, and it used to authenticate the MR-Job Component ($JC$) to the Name Node. The secondary token ($ST_{JCj}$) is to be generated when the job is being executed, and it is used to authenticate the MR-Job Component to the respective Data Node used in the job domain.

An overview of the key structure of the tokens mentioned above are plotted in Figure 5.2 and Figure 5.3, respectively. From the figures, it can be seen that, eight keys are used to form $ACR_{Ci}$ and four keys to form $ACR_{JCj}$ and their corresponding verification tokens. Table 5.1 and Table 5.2 gives the names and application of the secret keys along with the entities by which they are generated.
Figure 5.2 The key structure of the authentication credential of an MR-Client

Table 5.1 The secret keys used for MR-Clients authentication

<table>
<thead>
<tr>
<th>Keys</th>
<th>Key description</th>
<th>Issued by</th>
<th>Used to construct</th>
</tr>
</thead>
<tbody>
<tr>
<td>key₁</td>
<td>Login password-secret key</td>
<td>C</td>
<td>PT₁₂</td>
</tr>
<tr>
<td></td>
<td>represents a user login by his password</td>
<td></td>
<td></td>
</tr>
<tr>
<td>key₂</td>
<td>Dynamic login-secret key</td>
<td>AS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>represents a new key for each login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>key₃</td>
<td>Client master-secret key</td>
<td>AS</td>
<td>PT₂₂</td>
</tr>
<tr>
<td></td>
<td>represents something a client possesses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>key₄</td>
<td>User password-secret key</td>
<td>C and AS</td>
<td>ST₁₂</td>
</tr>
<tr>
<td></td>
<td>represents something a user knows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>key₅</td>
<td>Writing job-secret key</td>
<td>AS</td>
<td>ST₂₂</td>
</tr>
<tr>
<td></td>
<td>represents a job-resource writing to the DFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>key₆</td>
<td>Writing access-secret key</td>
<td>NN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>represents a writing-access to the DFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>key₇</td>
<td>NN’s public key</td>
<td></td>
<td>Used for transmission</td>
</tr>
<tr>
<td>key₈</td>
<td>Infrastructure domain-secret key</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The second idea is to use a key wrap-and-swap operation to support mutual authentication. This idea means that (i) the first key of each authentication token is encrypted (i.e. wrapped) using the second key of the authentication token to form the corresponding verification token before it is sent to the verifier, and (ii) the two keys used to form the authentication token are swapped to provide a mutual authentication. This idea is built on symmetric key based authentication approach and used to provide a mutual authentication in each interaction which involves a job-resource request in an MR-Job execution. In an authentication process of each interaction, there are three entities involved; an issuer of the token, a claimant who uses the authentication token, and a verifier of the token.

5 In an MR-Job domain, each MR-Job Component has a specific role to play. Any request, which is initiated by an MR-Job Component requesting access to a particular job resource from the DFS during a job execution, is defined based on the role played by the MR-Job Component. For example, when an MR-Job Component takes the role of a Job Tracker, the request type is typically defined as Retrieving Input-Splits Authentication (RIS-AuthN) Request, as the Job Tracker requests access to the Input Splits (one of the job resources) from the Name Node. Then the Name Node redirects such request to the respective Data Node in which this job resource is stored.

6 Once the MR-Job Component is authenticated to the Name Node, a role-access request is initiated by the MR-Job Component and sent to the assigned Data Node. It is defined based on the access given by the Name Node to the MR-Job Component, which plays a particular role during a job execution, to access a particular job resource. For example, as we mentioned above, when the Job Tracker is redirected by the Name Node to a Data Node to access the input splits during the job execution, the Job Tracker sends a Redirected Retrieving-Input-Split Authentication (R-RIS-AuthN) Request to the assigned Data Node.

---

Table 5.2 The secret keys used for MR-Job Components authentication

<table>
<thead>
<tr>
<th>Key</th>
<th>Key description</th>
<th>Issued by</th>
<th>Used to form</th>
</tr>
</thead>
<tbody>
<tr>
<td>key₁</td>
<td>Job domain-secret key</td>
<td>AS</td>
<td>PT⁻¹&lt;sub&gt;JC&lt;/sub&gt;</td>
</tr>
<tr>
<td>key₂</td>
<td>Role based-secret key</td>
<td>C</td>
<td>ST⁻¹&lt;sub&gt;JC&lt;/sub&gt;</td>
</tr>
<tr>
<td>key₃</td>
<td>Role access-secret key</td>
<td>NN</td>
<td></td>
</tr>
<tr>
<td>key₄</td>
<td>Writing job-secret key</td>
<td></td>
<td>Used for transmission</td>
</tr>
</tbody>
</table>
token. In VDAF, the claimant could be an MR-Client or an MR-Job Component and the issuer and the verifier could be the Authentication Server or an MR-Inf. Component. This is achieved by taking the following measures as summarised in Figure 5.4.

1. We use both symmetric key cryptosystem and hashed message authentication code mechanism for encryption, decryption and signing operations, i.e. $E(\cdot), E^{-1}(\cdot)$ and $HMAC(\cdot)$, respectively, that are needed to perform and secure the mutual authentication.

2. To form a corresponding verification token for each interaction, the issuer encrypts the first key of the authentication token ($k_1$) using the second key of the token ($k_2$), i.e. $E(k_2,\{k_1\})$ before it is sent to the verifier as a corresponding verification token.

3. In each interaction there are two authentication exchanges between the claimant and the verifier; an authentication request ($AuthN\_Request$) and an authentication response ($AuthN\_Response$).
   - $AuthN\_Request$ contains two components; the message ($Msg_{Req}$) and the message authentication code ($Mac_{Req}$). $Msg_{Req}$ consists of three parts, the first one contains an authenticator$^7$ ($A$) which is encrypted using the first key of the authentication token, i.e. $E(k_1,\{A\})$, the second one contains the second key of the token which is encrypted using a transmission key ($k_T$) known to the verifier, i.e. $E(k_T,\{k_2\})$, and the third one contains the claimant identity ($ID_c$) along with the timestamp of the request message ($T_{Req}$). These three parts of the message $Msg_{Req}$ are signed using the $HMAC$ operation and the second key of the token, i.e. $Mac_{Req} = HMAC(\{E(k_1,\{A\})\|E(k_T,\{k_2\})\|ID_c\|T_{Req}\}, k_2)$, before they are sent by the claimant to the verifier.
   - Once the verifier receives the $AuthN\_Request$, it uses the encrypted key received in the authentication request from the claimant, i.e. $k_2$, to decrypt the encrypted key in the corresponding verification token, i.e. $E^{-1}(k_2,\{k_1\})$, which is received from the issuer, so that the verifier can verify the correctness of the authenticator received from the claimant in $AuthN\_Request$ as following:
     - The verifier first decrypts the encrypted $k_2$, i.e. $E^{-1}(k_T,\{k_2\})$.
     - It checks $T_{Req}$ and the message authentication code of the received authentication request message, i.e. $Mac_{Req}$, using $k_2$ which is the same as $k_2$ received in the corresponding verification token.

$^7$ It is a piece of data sent with an authentication message to help the verifier to (i) assure the claimant’s knowledge of the secret key/s of an authentication credential, and (ii) prevent message replay.
key used by the claimant to sign the request message, i.e. \( Mac_{Req} = HMAC\left(E(k_1, \{A\})||E(k_T, \{k_2\})||ID_c||T_{Req}\right), k_2\).

- If both \( T_{Req} \) and \( Mac_{Req} \) are valid, the verifier uses \( k_2 \) to decrypt the encrypted \( k_1 \) in the corresponding verification token, received from the issuer, i.e. \( E^{-1}(k_2, \{k_1\})\).

- Finally, it decrypts the encrypted authenticator using \( k_1 \), i.e. \( E^{-1}(k_1, \{A\}) \), and checks if the authenticator is valid, then the claimant is authenticated to the verifier.

This means that when (i) the decryption process of the first part of \( Msg_{Req} \) (i.e. \( E^{-1}(k_1, \{A\}) \)) results a valid authenticator, (ii) the hashing process of \( Msg_{Req} \) results the same received message authentication code (i.e. the HMAC signature is correct), and (iii) the received timestamp is an acceptable timestamp\(^8\), the verifier knows that this claimant is the one claimed to be. This is because the claimant is the only one who knows both \( k_1 \) and \( k_2 \), so the claimant encrypts the authenticator in the \( AuthN\_Request \) and generates the message authentication code of the \( Msg_{Req} \) (i.e. signing the \( Msg_{Req} \)), respectively.

- When the verifier receives a valid authenticator, it swaps the use of the two keys of the authentication token, i.e. \( k_1 \) and \( k_2 \), to form \( AuthN\_Response \).

- \( AuthN\_Response \) also contains two components; a message (\( Msg_{Res} \)) and a message authentication code (\( Mac_{Res} \)). \( Msg_{Res} \) consists of two parts, the first one contains the authenticator which is received in the \( AuthN\_Request \) and, it is encrypted using the second key of the authentication token, i.e. \( E(k_2, \{A\}) \). The second part contains the verifier identity (\( ID_v \)) along with the timestamp of the response message (\( T_{Res} \)). These two parts of \( Msg_{Res} \) are signed using the \( HMAC \) operation and the first key of the token, i.e. \( Mac_{Res} = HMAC\left(E(k_2, \{A\})||ID_v||T_{Res}\right), k_1\), before they are sent.

- Once the claimant receives the \( AuthN\_Response \), it verifies the correctness of the authenticator as following:

---

\(^8\) An acceptable timestamp means that upon the receipt of \( AuthN\_Request \), the verifier compares the timestamp in the request against the local time. If the time skew between these two timestamps is not within the maximum transmission delay, the verifier rejects the request, because an attacker could have replayed the original request, i.e. \( AuthN\_Request \).
The claimant checks $T_{Res}$ and the message authentication code of the response message using $k_1$, i.e. $\text{Mac}_{Res} = \text{HMAC}(E(k_2, \{A\})||ID_v||T_{Res})$. If both $T_{Res}$ and $\text{Mac}_{Res}$ are valid, the verifier uses $k_2$ to decrypt the encrypted authenticator received in the $\text{AuthN\_Response}$, i.e. $E^{-1}(k_2, \{A\})$, that is sent before in the $\text{AuthN\_Request}$. It then checks if the received authenticator is valid, then the verifier is authenticated to the claimant.

This means that when (i) the hashing process of $Msg_{Res}$ results the same received message authentication code (i.e. the HMAC signature is correct) and (ii) the decryption process of the first part of $Msg_{Res}$ (i.e. $E^{-1}(k_2, \{A\})$) results a valid authenticator (the same authenticator sent in the $\text{AuthN\_Request}$), the claimant knows that the verifier is the one claimed to be. This is because the verifier is the only one, besides the claimant, has both keys, i.e. $k_1$ and $k_2$, and he could swap their use for encrypting and signing operations as (1) the verifier has the corresponding verification token which is received from the issuer and contains $k_1$ encrypted by $k_2$, and (2) the verifier receives $k_2$ from the claimant in different communication channel.

**Figure 5.4** A key wrap-and-swap operation for mutual authentication
The following table indicates the computational cost of our authentication method, i.e. the PT2M-AuthN method, discussed above.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Operation types and numbers</th>
<th>Entity involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Issue and send: $E(k_2, {k_1})$</td>
<td>$2T_{\text{Rng}} + 1T_{\text{Enc}}$</td>
</tr>
<tr>
<td>2&amp;3</td>
<td>Construct and send: $\text{AuthN_Request}{\text{Msg_Req}|\text{Mac_Req}}$</td>
<td>$2T_{\text{Enc}} + 1T_{\text{Mac}}$</td>
</tr>
<tr>
<td>4</td>
<td>Verify: $\text{AuthN_Request}$</td>
<td>$3T_{\text{Dec}} + 1T_{\text{Mac}}$</td>
</tr>
<tr>
<td>5&amp;6</td>
<td>Construct and send: $\text{AuthN_Response}{\text{Msg_Res}|\text{Mac_Res}}$</td>
<td>$1T_{\text{Enc}} + 1T_{\text{Mac}}$</td>
</tr>
<tr>
<td>7</td>
<td>Verify: $\text{AuthN_Response}$</td>
<td>$1T_{\text{Dec}} + 1T_{\text{Mac}}$</td>
</tr>
</tbody>
</table>

| Total computational cost | $2T_{\text{Rng}} + 4T_{\text{Enc}} + 4T_{\text{Dec}} + 4T_{\text{Mac}}$ |

$T = \text{Time}; H = \text{Hash}; Enc = \text{Symmetric encryption}; Dec = \text{Symmetric decryption}; Rng = \text{Random-number generator}; Mac = \text{Message Authentication Code}$

### 5.8. VDAF Architecture Design

This section gives an overview of the VDAF architecture and the functions it accomplishes. Basically, it performs identification and authentication functions. It does so by using functional components, called MR-AuthN Components that are integrated into the existing architecture of MR application. As each entity in the MR application performs multiple cryptographic operations to achieve these functions, these operations are encapsulated into one or two MR-AuthN Components that are added onto the entity. Figure 5.5 shows the architecture design of the VDAF. From the figure it can be seen that the VDAF architecture consists of four types of MR-AuthN Components: first, a Gate Authentication (G-AuthN) Service embedded into the Authentication Server (AS), second, two MR-Client authentication agents, a Client Authentication (C-AuthN) Agent and Client Job Authentication (CJ-AuthN) Agent both running on the client’s machine, third, three MR-Master authentication modules, a Resource Manager Authentication (RM-AuthN) Relay Module, a Name Node Authentication (NN-AuthN) Module and a Job Tracker Authentication (JT-AuthN) Module, respectively running on the Resource Manager, the Name Node and the Job Tracker, fourth, a number of the MR-Slave Authentication modules: Data Node Authentication (DN-AuthN) Modules, Task Tracker Authentication (TT-AuthN) Modules, Map Task Authentication (MT-AuthN) Modules, and Reduce Task Authentication (RT-AuthN) Modules, respectively running on Data Nodes and Task Trackers. The figure also shows the interactions for the VDAF functions, i.e. Clients identification (A), MR-Job
Components identification (B), MR-Client authentication (C), MR-Job Components authentication (D), and MR-Data authentication (E).

Figure 5.5 An overview of the VDAF architecture

5.8.1. Client identification

The client identification function is performed by the C-AuthN Agent and G-AuthN Service. It is used to identify a new client to the MR application and once identified, this client will become an MR-Client. In other words, a client uses this function to register itself and
upon a successful registration, the client will be issued with two primary authentication tokens, i.e. $PT_{Ci}^1$ and $PT_{Ci}^2$, and their corresponding verification tokens, i.e. $PVT_{Ci}^1$ and $PVT_{Ci}^2$, which are used to verify the authentication tokens against. As shown in Figure 5.6, the G-AuthN Service receives an identification request, NA-IdenT Request, from a C-AuthN Agent, and it verifies the identification information\(^9\) contained in this request. If the verification process is positive, it generates $PT_{Ci}^2$ and $PVT_{Ci}^2$ and sends $PT_{Ci}^2$ to the C-AuthN Agent in a response message, a NA-IdenT Response. Once the C-AuthN Agent receives $PT_{Ci}^2$, it generates $PT_{Ci}^1$ and $PVT_{Ci}^1$ and constructs the local and removable parts of $ACR_{Ci}$, i.e. $L_{ACR_{Ci}}$ and $R_{ACR_{Ci}}$, which should be stored in a two separate storage disk.

![Figure 5.6 Client identification](image)

5.8.2. MR-Client Authentication and MR-Job Components Identification

The MR-Client authentication consists of three authentication processes. The first one is to authenticate a user to an MR Application Client. The second one is to mutually authenticate the MR-Client and the Resource Manager. The third one is to mutually authenticate the MR-Client and the DFS cluster, i.e. between the MR-Client and the Name Node and between the MR-Client and the respective Data Nodes used. Figure 5.7 shows the MR-AuthN Components involved in performing the MR-Client authentication, and these components are the C-AuthN Agent, CJ-AuthN Agent, RM-AuthN Rely Module, NN-AuthN Module, DN-AuthN Module, and G-AuthN Service.

In the first authentication process, the C-AuthN Agent receives $ACR_{Ui}$ and $USB_{Ui}$ from the user. It validates $PT_{Ci}^1$ against the corresponding $PVT_{Ci}^1$, if all positive, the second authentication process is started.

In the second authentication process, the RM-AuthN Relay Module receives a New-Job Authentication (NJ-AuthN) Request from the CJ-AuthN Agent and it relays this request to

---

\(^9\) The identification information is a cryptographic information (i.e. piece of data) that is related to define and register a client to MR application, so that the client could be identified to MR application as MR-Client by issuing him an authentication credential.
the G-AuthN Service. The G-AuthN Service is responsible for (i) verifying the NJ-AuthN Request, which is identified by the CJ-AuthN Agent using $PT^2_{ci}$, against $PVT^2_{ci}$, (ii) generating $ST^1_{ci}$ and $SVT^1_{ci}$ which are used to provide a mutual authentication between the MR-Client and the Name Node, (iii) issuing a secret key that identifies the job domain and used by the CJ-AuthN Agent to generate the $PT^1_{JCI}$, and (iv) sending $ST^1_{ci}$ and $SVT^1_{ci}$ to the RM-AuthN Relay Module. The RM-AuthN Relay Module is then responsible for forwarding $SVT^1_{ci}$ to the NN-AuthN Module, and sending a NJ-AuthN Response message, which contains $ST^1_{ci}$ to the CJ-AuthN Agent. The NJ-AuthN Response message is identified by both the G-AuthN Service and the RM-AuthN Relay Module respectively. The CJ-AuthN Agent verifies the received response message to complete the mutual authentication process between the MR-Client and both the Authentication Server and the Resource Manager.

In the third authentication process, the NN-AuthN Module receives a Writing-Job Authentication (WJ-AuthN) Request, which is identified by the CJ-AuthN Agent using $ST^1_{ci}$, and verifies this request against $SVT^1_{ci}$. The NN-AuthN Module is also responsible for (i) generating $ST^2_{ci}$ and $SVT^2_{ci}$ which are used to provide mutual authentication between the MR-Client and the respective Data Node, and (ii) sending a notification message that contains $SVT^2_{ci}$ to the DN-AuthN Module before sending a WJ-AuthN Response message containing $ST^2_{ci}$ to the CJ-AuthN Agent. The WJ-AuthN Response message is identified by the NN-AuthN Module to complete the mutual authentication process between the MR-Client and the Name Node. Finally, the CJ-AuthN Agent sends a Redirected WJ-AuthN (R-WJ-AuthN) Request, which is identified by the CJ-AuthN Agent using $ST^2_{ci}$, to the DN-AuthN Module. The DN-AuthN Module is responsible for (i) validating this request against $SVT^2_{ci}$ and (ii) returning an identified response message back to the CJ-AuthN Agent, i.e. R-WJ-AuthN Response message. The CJ-AuthN Agent verifies the received response to complete the mutual authentication process between the MR-Client and the Data Node.
The MR-Job Components identification is to identify a set of MR components assigned
to a particular job domain, i.e. an MR-Job domain. This is done by issuing the primary
authentication and verification tokens of these components. As shown in Figure 5.8, MR-Job
Components identification is performed by the following MR-AuthN Components: the CJ-
AuthN Agent, RM-AuthN Relay Model, JT-AuthN Module, and TT-AuthN Module. The
fundamental function of these components is to (1) generate the primary authentication and
verification tokens for the Job Tracker ($PT_{JT}^j$ and $PVT_{JT}^j$), for the set of Task Trackers
($PT_{TT}^j$ and $PVT_{TT}^j$) and Reduce Tasks ($PT_{RT}^j$ and $PVT_{RT}^j$) of the MR-Job domain and (2)
distribute the respective tokens to the MR-Job Components.
5.8.3. MR-Job Components Authentication

MR-Job Components authentication consists of two authentication processes. The first one is mutual authentication between an MR-Job Component (either the Job Tracker, a Task Tracker, or a Reduce Task) and the DFS cluster, i.e. between an MR-Job Component and the Name Node, and between an MR-Job Component and the respective Data Node assigned to the job. The second one is mutual authentication between a pair of MR-Job Components.

In an MR-Job domain, as there are three types of MR-Job Components interact with the DFS cluster (i.e. the Job Tracker, a Task Tracker, and a Reduce Task), the first authentication process can be classified into three mutual authentication processes: mutual authentication between the Job Tracker and the DFS cluster, between a Task Tracker and the DFS cluster, and between a Reduce Task and the DFS cluster. As these three authentication processes are identical, and without loss of generality, MR-Job Component Authentication (JC-AuthN) Module is used to denote the MR-AuthN Module reside on each MR-Job Component, and the request initiated by any of these components is donated as a Role-Based Authentication (RB-AuthN) Request message\(^\text{10}\). Table 5.3 summarises the names of the MR-AuthN Components.

\(^{10}\) A RB-AuthN Request is defined based on the role played by an MR-Job Component which initiates the request, for example when a role played by the MR-Job Component is a Job Tracker, the request type is typically a Retrieving Input Splits Authentication (RIS-AuthN) Request, see Table 5.3 for more details.
of the MR-Job Components and the type of the messages involved in these authentication processes.

Table 5.3 The MR components and exchanged messages involved in the mutual authentications between MR-Job Components and the DFS cluster.

<table>
<thead>
<tr>
<th>Mutual authentication</th>
<th>Between Job Tracker (JT) and DFS</th>
<th>between Task Tracker (TT) and DFS</th>
<th>between Reduce Task (RT) and DFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR-AuthN Components involved</td>
<td>JT-AuthN Module</td>
<td>TT-AuthN Module</td>
<td>RT-AuthN Module</td>
</tr>
<tr>
<td></td>
<td>NN-AuthN Module</td>
<td>NN-AuthN Module</td>
<td>NN-AuthN Module</td>
</tr>
<tr>
<td></td>
<td>DN-AuthN Module</td>
<td>DN-AuthN Module</td>
<td>DN-AuthN Module</td>
</tr>
</tbody>
</table>

In the first authentication process, i.e. the mutual authentication between an MR-Job Component and the DFS, three MR-AuthN Components are involved. As shown in Figure 5.9, these components are the JC-AuthN Module, NN-AuthN Modules, and DN-AuthN Module. The NN-AuthN Module receives a RB-AuthN Request, which is identified by the JC-AuthN Module using $PT_{JC_i}$, from the JC-AuthN Module and verifies it against $PV_{JT_i}$. Upon a successful verification, the NN-AuthN Module generates $ST_{JC_i}$ and $SV_{JT_i}$ which are used for mutual authentication between the MR-Job Component and the respective Data Node. In addition, it sends a notification message, which contains $SV_{JT_i}$, to the DN-AuthN Module before sending a RB-AuthN Response message containing $ST_{JC_i}$ to the JC-AuthN Module. The response message is identified by the NN-AuthN Module to complete the mutual authentication process between the MR-Job Component and the Name Node. Finally, the JC-AuthN Module sends a Redirected RB-AuthN (R-RB-AuthN) Request message, which is identified by the JC-AuthN Module using $ST_{JC_i}$, to the DN-AuthN Module. The DN-AuthN Module is responsible for (i) validating this request against $SV_{JT_i}$ and (ii) returning an identified response message back to the JC-AuthN Module, i.e. R-RB-AuthN.
Response message. The JC-AthN Module verifies the received response message to complete the mutual authentication process between the MR-Job Component and the Data Node.

Figure 5.9 MR-Job Component authentication: first process

In the second authentication process (i.e. mutual authentication between a pair of MR-Job Components), the RT-AthN Module and TT-AthN Module are involved. The fundamental function of these two components is to perform mutual authentication between a Reduce Task and a Task Tracker. As shown in Figure 5.10, the RT-AthN Module sends a Retrieving-Intermediate-Data Authentication (RID-AthN) Request, which is identified using $PT_{RT_i^j}$, to the TT-AthN Module. The TT-AthN Module is responsible for (i) validating this request, and (ii) sending a response message (RID-AthN Response message) back to the RT-AthN Module. The RT-AthN Module then verifies the received response message to complete the mutual authentication process between the Reduce Task and the Task Tracker.

Figure 5.10 MR-Job Component authentication: second process
5.8.4. MR-Data Authentication

MR-Data authentication consists of three authentication processes through an MR-Job execution. These authentication processes are Original-Data authentication, Intermediate-Data authentication, and Final-Data authentication, and they are used to provide origin authentication and integrity protection of the three sets of the MR data, i.e. input data, intermediate data, and output data. The Original-Data authentication process is performed by the CJ-AuthN Agent and MT-AuthN Module. As shown in Figure 5.11, in this authentication process, the CJ-AuthN Agent generates a signature of MR-Client’s input data and the MT-AuthN Module verifies the signature. The MT-AuthN Module verifies the signature the CJ-AuthN Agent signed on the input data when it is being retrieved from the DFS cluster as long as (i) the assigned Task Tracker has successfully authenticated with the DFS cluster, and (ii) the signature on this data is generated before the data is being written to the DFS cluster.

![Figure 5.11 The Original-Data authentication](image)

The Intermediate-Data authentication process is performed by the MT-AuthN Module and RT-AuthN Module. As shown in Figure 5.12, the MT-AuthN Module signs the intermediate data which is produced by the Map Task in an MR-Job execution, and the RT-AuthN Module verifies the signature. The RT-AuthN Module verifies the signature the MT-AuthN Module signs on the intermediate data when the data is being retrieved from the Task Tracker as long as (i) the assigned Reduce Task has successfully authenticated with the Task Tracker, and (ii) the signature on this data is generated before the data is being written in the Task Tracker.

![Figure 5.12 The Intermediate-Data authentication process](image)
The Final-Data authentication process is performed by the RT-AuthN Module and CJ-AuthN Agent. As shown in Figure 5.13, the RT-AuthN Module signs the final data produced by the Reduce Task, during an MR-Job execution, and the CJ-AuthN Agent verifies the signature. The CJ-AuthN Agent verifies the signature the RT-AuthN Module signed on the final data when the data is being retrieved from the DFS as long as the MR-Client has successfully authenticated with the DFS and the signature on this data is generated before the data is being written to the DFS.

Figure 5.13 The Final-Data authentication process

5.9. Lightweight VDAF Authentication Protocol Suite

To facilitate the authentication functions of the VDAF, four sets of protocols are proposed. They are collectively called the Lightweight VDAF Authentication Protocol (LVAP) suite. The first set contains one protocol called MR-Client Primary Credential Establishment (CPCrE) protocol. The second set contains four protocols, and they are User to MR Application Client Authentication (U-AC_AuthN) protocol, MR-Client to Resource Manager Authentication (C-RM_AuthN) protocol, MR-Client to Name Node Authentication (C-NN_AuthN) protocol, and MR-Job Components Primary Credentials Establishment (JCPCrE) protocol. The third set contains four protocols, and they are Job Tracker to Name Node Authentication (JT-NN_AuthN) protocol, Task Tracker to Name Node Authentication (TT-NN_AuthN) protocol, Reduce Task to Name Node Authentication (RT-NN_AuthN) protocol, and Reduce Task to Task Tracker Authentication (RT-TT_AuthN) protocol. The fourth set contains three protocols called Original-Data Authentication (OD_AuthN) protocol, Intermediate-Data Authentication (ID_AuthN) protocol, and Final-Data Authentication (FD_AuthN) protocol.

Table 5.4 shows these four sets of protocols and the VDAF functions to be facilitated by them. As can be seen from the table, the first two sets of protocols are proposed to perform client identification (F1), MR-Client Authentication (F2) and MR-Job Components Identification (F3), and the second two sets are proposed to perform MR-Job Components
Authentication (F4) and MR-Data Authentication (F5). In this thesis, the remaining two chapters focus on the VDAF functions F1, F2, and F3, which are collectively referred to as a job submission authentication. These two chapters present in details the design and the evaluation of the first two sets of the LVAP suite.

Table 5.4 LVAP suite

<table>
<thead>
<tr>
<th>VDAF functionality</th>
<th>LVAP suite (protocols)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client identification (F1)</td>
<td>MR-Client Primary Credential Establishment protocol (CPCrE) protocol</td>
<td>To identify a new client as an MR-Client to the MR Application by issuing primary authentication and verification tokens of the MR-Client (i.e. $PT_C^1$, $PT_C^2$, $PV_T C_1^1$ and $PV_T C_1^2$).</td>
</tr>
<tr>
<td>User to MR Application Client authentication protocol (U-AC_AuthN protocol)</td>
<td>To authenticate a user to the MR Application Client, i.e. to authenticate the user login to the MR Application Client that is run locally in the client machine and do so by using $PT_C^1$ and $PV_T C_1^1$.</td>
<td></td>
</tr>
<tr>
<td>MR-Client to Resource Manager authentication protocol (C-RM_AuthN protocol)</td>
<td>To provide mutual authentication between the MR-Client and both the Resource Manager and the Authentication Server using $PT_C^1$ and $PV_T C_1^1$, and to generate the 1st secondary authentication and verification tokens of the MR-Client (i.e. $ST_C^1$ and $SV_T C_1^1$).</td>
<td></td>
</tr>
<tr>
<td>MR-Client to Name Node authentication protocol (C-NN_AuthN protocol)</td>
<td>To (i) provide mutual authentication between the MR-Client and the Name Node using $ST_C^1$ and $SV_T C_1^1$, (ii) generate the 2nd secondary authentication and verification tokens of the MR-Client (i.e. $ST_C^2$ and $SV_T C_2^1$), and (iii) provide mutual authentication between the MR-Client and the assigned Data Node using $ST_C^1$ and $SV_T C_1^2$.</td>
<td></td>
</tr>
<tr>
<td>MR-Job Components Primary Credentials Establishment protocol</td>
<td>To identify a set of MR components as MR-Job Components to the MR-Job domain by issuing primary authentication and verification tokens of the MR-Job Components, i.e. $PT_{JT}^1$, $PT_{TT}^1$, $PT_{RR}^1$ and $PV_T JT_{JT}^1$, $PV_T TT_{JT}^1$, $PV_T RR_{JT}^1$.</td>
<td></td>
</tr>
<tr>
<td>Job Tracker to Name Node authentication protocol (JT-NN_AuthN protocol)</td>
<td>To (i) provide mutual authentication between the Job Tracker and the Name Node using $PT_{JT}^1$ and $PV_T JT_{JT}^1$, (ii) generate secondary authentication and verification tokens of the Job Tracker, i.e. $ST_{JT}^1$ and $SV_T JT_{JT}^1$, and (iii) provide a mutual authentication between the Job Tracker and the assigned Data Node using $ST_{JT}^1$ and $SV_T JT_{JT}^1$.</td>
<td></td>
</tr>
<tr>
<td>Task Tracker to Name Node authentication protocol</td>
<td>To (i) provide a mutual authentication between a Task Tracker and the Name Node using $PT_{TT}^1$ and $PV_T TT_{TT}^1$, (ii) generate secondary authentication and verification tokens of the Task Tracker, i.e. $ST_{TT}^1$ and $SV_T TT_{TT}^1$, and (iii) provide a mutual authentication between the Task Tracker and the assigned Data Node using $ST_{TT}^1$ and $SV_T TT_{TT}^1$.</td>
<td></td>
</tr>
</tbody>
</table>
(TT-NN_AuthN protocol) tokens of the Task Tracker, i.e. \( S_{TT_j} \) and \( SV_{TT_j} \), and 
(iii) provide a mutual authentication between the Task Tracker and the assigned Data Node using \( S_{TT_j} \) and \( SV_{TT_j} \).

Reduce Task to Name Node authentication protocol

(RT-NN_AuthN protocol) To (i) provide a mutual authentication between a Reduce Task and the Name Node using \( PT_{RT_i} \) and \( PV_{RT_i} \), (ii) generate secondary authentication and verification tokens of the Reduce Task, i.e. \( ST_{RT_i} \) and \( SV_{RT_i} \), and (iii) provide a mutual authentication between the Reduce Task and the assigned Data Node using \( ST_{RT_i} \) and \( SV_{RT_i} \).

Reduce Task to Task Tracker authentication protocol

(RT-TT_AuthN protocol) To provide a mutual authentication between a Reduce Task and a Task Tracker using \( PT_{RT_i} \).

Original-Data authentication protocol

(OD_AuthN protocol) To ensure the authenticity of the input data of an MR-Client’s job.

Intermediate-Data authentication protocol

(ID_AuthN protocol) To ensure the authenticity of the intermediate data of an MR-Client’s job.

Final-Data authentication protocol

(FD_AuthN protocol) To ensure the authenticity of the output data (i.e. the output result) of an MR-Client’s job.

5.10. VDAF Cryptographic Building Blocks

This section presents the cryptographic algorithms or building blocks that to be used in the design of the LVAP suite. These are symmetric key cryptosystem, hashed message authentication code (HMAC), asymmetric key cryptosystem, digital signature, and public key infrastructure (PKI).

5.10.1. Symmetric Key Cryptosystem

In the symmetric key cryptosystem (also known as a secret-key cryptosystem), the same secret key is used for the encryption and decryption operations. This cryptosystem is commonly used for achieving three security services; message confidentiality, message authenticity, and entity authentication [79]. Most commonly used symmetric key cryptosystems include Triple Data Encryption Algorithm (TDEA) known as Triple DES or 3DES (Data Encryption Standard DES) [80], and the Advanced Encryption Standard (AES)
[81]. Figure 5.14 illustrate how message confidentiality is preserved using the symmetric key cryptosystem. A sender encrypts a message \( (m) \) using symmetric cryptosystem \( (E) \) and a secret key \( (k) \) associated to \( (E) \) to produce a ciphertext \( (C) \) (i.e. encrypted message), i.e. \( C = E(k, m) \). The ciphertext is sent via a network to a receiver. Upon the receipt of the ciphertext, the receiver uses the same key \( (k) \) and the corresponding decryption algorithm \( (E^{-1}) \) to recover message \( (m) \). This cryptographic operation, i.e. the decryption operation, is denoted as \( m = E^{-1}(k, C) \).

In the LVAP suite design, the symmetric key cryptosystem is to be used to (i) protect the confidentiality of the payload (i.e. the identification and authentication information) in each protocol, and the local and removable parts of each MR-Client authentication credential, and (ii) facilitate mutual authentication between (1) MR-Clients and MR application, (2) MR-Job Components and the DFS cluster, and (3) any pair of MR-Job Components during an MR-Job execution.

5.10.2. Hashed Message Authentication Code (HMAC)

A Hashed Message Authentication Code (HMAC) is a function which is used to generate message authentication code. It is built on cryptographic hash functions. A cryptographic hash function, \( H() \), is a function that takes an arbitrary length of data as an input, such as a message \( m \), and generates a fixed size of an output, i.e. the hash value of \( m \), denoted as \( h = H(m) \). \( h \) is also called message digest. Generally, a hash function should have the following properties [79]:

- **One-way property**: for a message \( m \), it is easy to compute the hash value of \( m \) (i.e. \( h = H(m) \)), but for a given hash value \( h \), it is hard to compute \( m \).
- **Collision-resistance:** for a given message $m$, it is hard to find another message $m'$, such that $H(m) = H(m')$.

HMAC is a key-dependent hash function, so it is called a keyed hash function. The same symmetric key is used to generate and verify the HMAC message digest. A message digest generated by the HMAC function is referred to as a message authentication code ($Mac$) or an HMAC signature [79]. Figure 5.15 shows the process of $Mac$ generation and verification using HMAC. A sender applies the HMAC function, $HMAC()$ to a message $m$ using a secret key $k$, producing an HMAC signature ($Mac$ value), i.e. $Mac = HMAC(m, k)$. This $Mac$ value along with the message $m$ are sent to a receiver. The receiver computes a fresh HMAC signature $Mac'$ using the received message $m$ and the same symmetric key $k$, which is shared between the sender and the receiver, i.e. $Mac' = HMAC(m', k)$. The receiver then compares $Mac'$ with the received one, to see if ($Mac' = Mac$). If the two values are equal, the receiver is assured that the received message is authentic.

![Figure 5.15 HMAC cryptographic operations (HMAC signature generation and verification)](image)

HMAC provides message authenticity (i.e. origin authentication and message integrity). This is because an HMAC signature can only be verified using the same symmetric key which is used to generate it. Origin authentication assures that the source of a received message is as claimed, while the message integrity guarantees that a received message has not been modified during its transmission. Examples of well-known standard HMAC functions are HMAC-MD5, HMAC-SHA1, HMAC-SHA256, and HMAC-SHA512 [82] [83].

HMAC functions are to be used in the LVAP suite design to provide message authenticity for MR-CAuthN, JCPCrE and MR-JCAuthN protocols.
5.10.3. Asymmetric Key Cryptosystem

It is also known as public key cryptosystem. Different from the symmetric key cryptosystem, in the asymmetric key cryptosystem, encryption and decryption keys are different. One is public key made known to the entity that wishes to communicate with the key owner. The other key is a private key only known to the key owner. For confidential communication with the key owner, the public key is used for encryption, i.e. to encrypt the message, and the private key (in the pair) is used for decryption, i.e. to decrypt the message. These two keys are mathematically linked in such a way that if a message or data is encrypted with one key it can only be decrypted by the other key. An asymmetric cryptosystem is commonly used to provide the following security services: (1) message confidentiality, (2) message authenticity, and (3) non-repudiation [79]. Figure 5.16 illustrates an example of how the confidentiality of a message is preserved using an asymmetric cryptosystem. A sender encrypts a message \((m)\) using an asymmetric encryption algorithm \((Enc)\) along with the receiver’s public key \((PU_R)\) to produce a ciphertext \((C)\). This cryptographic operation is denoted as \(C = Enc(PU_R, m)\). The ciphertext is then sent to the receiver. The receiver uses the private key \((PR_R)\) and decryption algorithm \(Enc^{-1}\) to decrypt the ciphertext and recover the message \(m\).

Asymmetric key cryptosystem is computationally much more expensive than its symmetric key counterpart, it is typically used for confidential transportation of symmetric keys and symmetric keys are used for bulk data encryption [84]. Rivest Shamir Adleman (RSA) [85] and ElGamal [86] are well known asymmetric key cryptosystems.

In the LVAP suite design, asymmetric key cryptosystem are to be used to provide confidentiality protection to symmetric keys transportations between a pair of MR-AuthN Components, e.g. in the design of the CPCrE and C-NN authentication protocols.
5.10.4. Digital Signature

A digital signature scheme typically consists of two cryptographic operations: signature generation and verification. Figure 5.17 illustrates the process of signature generation ($SignGen()$) and signature verification ($SignVer()$). To sign a message $m$, (i.e. to generate a digital signature), a sender first generates a hash value $h$ of $m$, using a hash function $H()$, i.e. $h = H(m)$, and encrypts this hash value using his private key, $PR_S$, to generate the digital signature, $Sig$. The sender then sends both the message $m$ and the signature $Sig$ to the receiver. To verify the signature the receiver first uses the public key of the sender, $PU_S$, to decrypt the signature $Sig$ to obtain the hash value $h$. It then uses the same hash function $H()$ to generate fresh hash value from the received message $m'$, i.e. $h' = H(m')$, and compares the two hash values. If the two hash values are equal, the signature is valid. Otherwise it is considered invalid and should be rejected.

![Diagram of digital signature cryptographic operations](image)

Figure 5.17 Digital signature cryptographic operations

Digital signatures are typically used to provide two security services; message authenticity and non-repudiation [79]. Message authenticity is assured when a receiver has successfully verified the digital signature signed by claimed sender on the received message. If the digital signature is valid, it means that the signature and its message have not been altered while in transit and it comes from the claimed signer (i.e. sender). A valid signature along with a valid certified signature verification key also indicates that non-repudiation is achieved; the signer cannot later deny that he/she has signed the message. This is because the signature has been verified with a public key that has been certified to belong to the
signer, and it is generated with a private key corresponding to the public key and only the signer has the knowledge of this private key.

Well-known digital signature schemes are DSA and RSA [79] [85] [87]. The RSA scheme is used in the CPCrE protocol to sign and verify digital certificate, such as the client payment-attribute certificate (details to follow in next chapter), and to sign and verify the protocol messages. The RSA scheme is chosen rather than DSA scheme as the former is much faster than the latter in term of signature verification, although the DSA scheme performs faster than the RSA scheme in generating the digital signature [88]. In our protocol context, a digital certificate is signed once by an issuer and verified many times by verifiers. The RSA scheme is also much more widely deployed for public key certificates.

5.10.5. Public Key Infrastructure (PKI)

A Public Key Infrastructure (PKI) is a system for managing digital certificates. Large-scale applications of asymmetric cryptosystems rely on the use of PKI. A digital certificate, i.e. a public key certificate, is used to assure that a public key indeed belongs to a claimed owner and it is valid (i.e. demonstrate the authenticity of the public key), and to establish secure channels with other entities. In a typical PKI, a trusted authority called a Certificate Authority (CA) is used. CA is used to issue and sign a public key certificate for the claimant (i.e. the entity). This public key certificate contains the public key of the entity and other information such as the entity’s identity and CA’s signature [79].

5.10.5.1. Public key certificate

A public key certificate is a data structure which is generated and signed by a CA [79]. It contains a number of attributes (also called data fields) that are defined by the X.509 Certificate Specification [79] [89] as follows:

- **Version**: This field defines the version of the public key certificate.
- **Serial Number**: This field presents a unique number that is assigned by the CA for each public key certificate.
- **Signature Algorithm Identifier**: It defines the algorithm that used by the CA to generate the digital signature of the certificate.
- **Issuer**: It identifies the entity that has generated and signed the certificate.
- **Validity:** This field specifies the time interval by which the certificate is considered to be valid. It contains two dates: the date from which the certificate validity period begins (notBefore) and the date to which the certificate validity period ends (notAfter).

- **Subject:** This field identifies the owner of the public key, in other words, to whom the public key certificate is issued.

- **Subject Public Key Info:** This field contains the public key, and it identifies the algorithm with which the key is used for (e.g., DSA or RSA).

- **Extensions:** This field is a sequence of one or more certificate extensions that provide methods for associating additional attributes with users or public keys, such as alternative subject names or public keys.

- **CA’s Digital Signature:** This field contains the issuer's signature on the above data fields, i.e. it contains the CA’s signature on the issued public key certificate. Figure 5.18 shows the public key certificate structure.

![Figure 5.18 Public key certificate structure](image)

### 5.10.5.2. Certificate Authority (CA)

A Certificate Authority (CA) is a trusted authority. It is responsible for issuing and signing a public key certificate for an entity requesting a certificate. Upon the verification of the entity’s identity, the CA generates a public key certificate that contains the entity's public key [79].

### 5.10.5.3. Certificate Lifecycles and Key Management

A typical lifecycle of a public key certificate includes the following stages:
Key Generation: In this stage a public/private key pair associated to a public key certificate is generated.

Identity Submission: During this stage credentials of an entity, such as the identifier of the entity and email address, are submitted to a CA.

Registration: This is the stage where the identity of the requesting entity is verified and registered.

Certification: This is the stage where the identity of the entity is validated, and a public key certificate is generated and digitally signed by the CA.

Distribution: At this stage the issued public key certificate is distributed to the owner.

Usage: At this stage the certificate is used.

Expiration: In this stage the certificate becomes expired unless it is revoked or renewed.

5.11. Chapter Summery

This chapter has presented a high-level analysis of how an authentication service should be provided for MR application and given the idea of using a layered approach to authentication in this context. To realise the complex task of authentication for MR application in the distributed and resource-sharing environment, the MR Layered Authentication Model (MR-LAM) is proposed. With this model, authentication service is provided at multiple levels, and it takes into account of the authentication requirements of all the components of the MR application. It consists of two layers; (1) MR-Inf. domain authentication in which the MR server nodes authentication should be addressed, and (2) MR-Job domain authentication in which the authentication of MR-Client and MR components of his job should be addressed. This chapter has also discussed different authentication threats and attacks that may be mounted against the MR-Client’s job, i.e. an MR-Job domain. This analysis led to specify a set of requirements, and these requirements are used to guide the design of a secure, efficient and practical Virtual Domain based Authentication Framework (VDAF). VDAF imposes authentication control to all the job-resource requests made during the entire cycle of a job execution. In the design of the VDAF a novel method of authentication called a Password and Token-based Multi-point Multi-factor Authentication (PT2M-AuthN) is proposed.
PTSM-AuthN method implements two ideas; first, we apply a principle of separation of duty-and-credential to the implementation of the authentication function of the VDAF, and second, we use a key wrap-and-swap operation to support a mutual authentication in each interaction involves a job-resource request in an MR-Job domain. These two ideas are used to strengthen the security level of the authentication function of VDAF and to carry out this function efficiently. To facilitate the authentication functions of VDAF, four sets of protocols called the LVAP suite are proposed. In the LVAP suite a number of cryptographic building blocks, including symmetric and asymmetric key cryptosystems and message authentication code mechanisms, is to be used to achieve the security properties defined for the protocols.

The following chapter presents the design and evaluation of the first set of the LVAP suite, i.e. the CPCrE protocol, which allows the VDAF to perform the client identification function.
Chapter 6

6. A Novel Protocol for Clients Identification

6.1. Chapter Introduction

This chapter presents the design and evaluation of MR-Client Primary Credential Establishment (CPCrE) protocol. The CPCrE protocol is designed to realise the Client identification function of VDAF. The CPCrE protocol makes use of three building blocks; a symmetric key cryptosystem, an asymmetric key cryptosystem, and public key infrastructure that have been explained in Section 5.10. By using these three building blocks, the protocol (i) allows a remote client to be identified securely as an MR-Client to the MR application by issuing the primary authentication and verification tokens to the client, (ii) supports a secure MR-Clients’ identification task by providing a confidentiality and integrity protection for the tokens and a confidentiality and authenticity protection for the exchanged messages, and (iii) allows MR application to authenticate MR-Clients using multi-factor authentication. The evaluation of the CPCrE protocol is performed in two stages. First, both an informal security analysis and a formal security verification of the protocol are performed. This stage is performed to demonstrate the withstanding of the protocol against known attacks, such as those defined in the threat model (Section 5.6.1), and to validate the security properties of the protocol. Second, the performance evaluation of the protocol is performed to measure both the computational and communication overheads and the protocol execution time. The design and evaluation of the CPCrE protocol is our third contribution in this thesis.

In detail, Section 6.2 presents the design preliminaries for the CPCrE protocol, including a trust model for a client payment and identification, requirements and additional assumptions. Section 6.3 gives an overview of the protocol design and Section 6.4 presents the design of the primary authentication and verification tokens of an MR-Client and their key structure. Section 6.5 gives a detailed description of the protocol design which is followed by a security analysis and modelling of the protocol in Section 6.6. The performance evaluation of the protocol is reported in Section 6.7. Finally, Section 6.8 summarises this chapter.
6.2. Design preliminaries

This section details a trust model, requirements, and additional assumptions used in the design of the CPCrE protocol.

6.2.1. Trust Model

A trust model for a client payment and identification is used in the design of the CPCrE protocol. The trust model, as shown in Figure 6.1, consists of the following entities: the MR application, the MR Authentication Server (AS), the MR Payment Server (PS), i.e. a payment certificate issuer, the Payment Processors and Networks, the Public Key Infrastructure (PKI) and clients. The first three entities, i.e. MR-Application, AS, and PS, represent the entities of an MR application provider¹¹. The MR application and the AS have been already introduced in the previous chapters. The PS is responsible for issuing a payment-attribute certificate for each new client. The payment-attribute certificate is used to proof the client has paid his registration fees to use MR application before he can be identified as an MR-Client. The registration fees cover a client subscription and allowed-period to use the MR application. The client’s subscription represents the account of the client to be identified as MR-Client, and in practice it is typically a monthly or yearly subscription. The client’s allowed-period represents the expected period (i.e. time interval) for an MR-Client to use the MR application. In our CPCrE protocol, it is assumed that the client has paid his registration fees in advance. The Payment Processors and Networks are a set of services that process the payment transaction (from T.1 to T.7, see Figure 6.1) on behalf of the MR application provider. More details about how this transaction cycle works and what entities are involved in Payment Processors and Networks can be found in [90] [91] [92]. The PKI, as explained in [89] [93] establishes and maintains a trustworthy networking environment by providing keys and certificates management services, the services that enable cryptographic operations such as encryption/decryption and digital signature capabilities across these entities. The use of such capabilities in our CPCrE protocol is explained in the following sections.

As mentioned above, for a client to be identified as an MR-Client to the MR application using the CPCrE protocol, the client has to obtain a valid payment-attribute certificate. To obtain (or i.e. purchase) a valid payment-attribute certificate, as shown in Figure 6.1, the

¹¹ The MR application provider is a service provider that offers the MR application to MR-Clients.
client first submits his payment information, i.e. his credit card information (Figure 6.1 step 1). Then this information is sent to the Payment Processors and Networks (Figure 6.1 step 2). The Payment Processors and Networks authorise the payment either by accepting or declining the payment process (Figure 6.1 step 3). Once the payment process has been completed successfully (i.e. accepted), the client receives a payment-attribute certificate (Figure 6.1 step 4) which is issued by the PS. Then, the client sends his payment-attribute certificate with other identification information to the AS (Figure 6.1 step 5) using our CPCrE protocol, thus the client can be identified as an MR-Client to the MR application (details to follow in Sections 6.3-6.4).

This trust model for the client payment and identification is used for the following reasons: (i) it facilitates a flexible and practical remote client identification for using MR application by (1) a secure online (i.e. remote) payment, so the client does not have to pay his fees in person, (2) using a payment-attribute certificate that holds the client account and approved by the PS to provide an evidence of the client payment, (ii) it supports the common
online payment process (the online transaction cycle, i.e. T.1 to T.7) used by different online payment service providers listed in [94] which are complied with the Payment Card Industry Data Security Standard PCI DSS [95]. and (iii) it allows the AS to verify the client identification request while (1) the AS does not have to be in a trust with each individual entity involved in the payment process or (2) the AS does not have to store the payment information of the clients, as the AS has only the PS’s public key, which is certified using the Certificate Authority of the PKI, to verify each client identification request.

6.2.1.1. A Payment-Attribute Certificate

A payment-attribute certificate is a digital document containing attributes that are associated to a client by the issuer (PS). As shown in Figure 6.2, a payment-attribute certificate for a client consists of the following: a payment-attribute certificate ID, an ID of PS, an ID of a client to be identified to the MR application, the crypto type, the expiry date, the allowed-period, and the signature of the PS. The advantage of using the payment attribute-certificate includes: (i) it provides the digital signature of the PS using the PS’s private key that cannot be repudiated, and (ii) it is a scalable solution to proof the client payment. This is because the AS does not have to keep all the clients’ payment information and their secret keys rather the AS uses the certified PS’s public key to verify the payments for any applied, remote client. The use of each content of a client payment-attribute certificate is explained in Table 6.1

![Figure 6.2 A client payment-attribute certificate structure](image-url)
Table 6.1 The contents of a client payment-attribute certificate

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payment-attribute certificate ID</td>
<td>Used to uniquely identify the payment-attribute certificate.</td>
</tr>
<tr>
<td>MR Payment Server ID</td>
<td>Used to identify the PS, belongs to MR application provider, which issues this payment-attribute certificate, as it is assumed that each MR service provider has one PS.</td>
</tr>
<tr>
<td>Client ID</td>
<td>Used to identify the client which has paid to use the MR application.</td>
</tr>
<tr>
<td>Crypto type</td>
<td>Used to specify the cryptographic algorithm by which this certificate is signed.</td>
</tr>
<tr>
<td>Expiry date ($Exp_{C_i}$)</td>
<td>Used to specify the expiry date of a client subscription. It represents the account expiry date of a client to be identified as an MR-Client.</td>
</tr>
<tr>
<td>Allowed period ($AP_{C_i}$)</td>
<td>Used to specify the allowed period (i.e. time interval) for an MR-Client to use the MR application as long as his account has not been expired.</td>
</tr>
<tr>
<td>PS’s signature</td>
<td>The signature of the issuer (i.e. PS) over the whole data above.</td>
</tr>
</tbody>
</table>

6.2.2. Requirements

This section presents the design requirements for the CPCrE protocol. These requirements are divided into functional requirements, security requirements, and performance requirements.

6.2.2.1. Functional Requirements

As mentioned in Section 5.8, the client identification function of VDAF allows new clients to be remotely identified as MR-Clients. In order to achieve this task, the CPCrE protocol should satisfy the following functional requirements.

**F1.1** The AS should be able to verify the correctness of each client identification request, i.e. an NA-IdenT Request, in term of both its authenticity and the used identification information.
F1.2 The primary authentication and verification tokens that are used to identify an MR-Client to the MR application should be issued.

F1.3 The primary authentication tokens of an MR-Client should be unique for each client so that no two MR-Clients could ever use the same primary authentication tokens.

F1.4 The secret keys of $PT_1^{C_i}$ and $PVT_1^{C_i}$ should be structured in such a way that enables (i) the user login credential to be verified locally in the client machine, and (ii) the NJ-AuthN Request to be identified locally using $PT_2^{C_i}$ and verified remotely against $PVT_2^{C_i}$.

F1.5 $PT_1^{C_i}$ and $PT_2^{C_i}$ should be used to form the local and removable parts of the MR-Client authentication credential.

6.2.2.2. Security Requirements

S1 Mutual Authentication: both the client and the AS should have assurance that they communicate to each other. In other words, the AS should have assurance that it communicates to the client that it claims to be, and the client should have assurance that it communicates to the AS that it claims to be.

S2 Message Authenticity: The recipient of a message (i.e. AS or client) should be assured that the message has not been altered during transit, is fresh and indeed from the claimed source (client or AS, respectively).

S3 Message Confidentiality: This is a protection of NA-IdenT Request and Response messages from any unauthorised disclosure. To counter eavesdropping attacks, the confidentiality of any such message sent between a client and the AS should be protected.

S4 Support strong MR-Client authentication using multi-factor authentication secrets and provide a confidentiality and integrity protection for $PT_2^{C_i}$ and $PT_2^{C_i}$: the CPCrE protocol should be able to identify the client to the MR application using multi-factor authentication secrets, and when any secret key of these tokens are stored in a local and/or removable storage of the MR-Client, it should be confidentiality and integrity protected.

S5 Provide protection against a client-payment impersonation: the AS should be assured that the identity and the payment of a client belong to the client who claims to be (i.e. the payment has not been used by another client or attacker).
6.2.2.3. Performance Requirements

P1 Computational cost imposed on the AS and each client as a result of (i) verifying the identification request and response, (ii) issuing the primary authentication and verification tokens of an MR-Client, and (iii) defining the client attributes and protecting the primary authentication tokens of the MR-Client should be as low as possible.

P2 The communication cost introduced by the CPCrE protocol as a result of exchanging messages between the AS and each client should be as low as possible.

6.2.3. Additional Assumptions

In addition to the assumptions presented in Section 5.6 the following assumptions are used in the CPCrE protocol design.

A1 Each MR application provider has one PS.
A2 The AS and PS are trustworthy.
A3 Each client has a public/private key pair, i.e. $PU_{Ci}$/$PR_{Ci}$. The public key, $PU_{Ci}$, has been certified by a trusted CA and the private key, $PR_{Ci}$, is kept secret by the client.
A4 Respectively, the AS, the PS, has a public/private key pair, i.e. $PU_{AS}$/$PR_{AS}$, $PU_{PS}$/$PR_{PS}$. The public key has been certified by a trusted CA and the private key is kept secret by the respective holder.
A5 Each client has paid his registration fee and received his payment-attribute certificate, $PCert_{Ci}$, which is signed by the PS’s private key, $PR_{PS}$.

6.2.4. Notations

In addition to those notation listed in Sections 5.6, this section defines some of new notations used throughout this chapter.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$IP_{Cu_{i}}$</td>
<td>IP address of $CH_{u_{i}}$</td>
</tr>
<tr>
<td>$CVP_{Ci}$</td>
<td>$C_i$’s long-term credential validity period {it consists of $t_s$ : not before and $t_n$ : not after}. It presents the validity period of MR-Client’s primary tokens</td>
</tr>
<tr>
<td>$Exp_{Ci}$</td>
<td>$C_i$’s account expiry date</td>
</tr>
</tbody>
</table>
PCert<sub>ci</sub>  
C<sub>i</sub>’s payment-attribute certificate

AP<sub>ci</sub>  
allowed period (i.e. time interval), for C<sub>i</sub> to use the MR application

ID<sub>AS</sub>  
AS’s identity

K<sub>ci</sub>  
user password-secret key, it is Known to C<sub>i</sub>

P<sub>ci</sub>  
client matter-secret key, it is Possessed by C<sub>i</sub>

L<sub>ci</sub>  
client location-secret code, it indicates where C<sub>i</sub> is Located

r<sub>x</sub>  
a secure random number generated by an entity x

t<sub>rx</sub>  
current timestamp when r<sub>x</sub> is generated

{PU<sub>x</sub>, PR<sub>x</sub>}  
a pair of public and private keys for an entity x

SK<sub>x</sub><sup>LP</sup>  
login password-secret key of C<sub>i</sub>

SK<sub>x</sub><sup>SP</sup>  
session password-secret key of C<sub>i</sub>

SK<sub>x</sub><sup>DL</sup>  
dynamic login-secret key of C<sub>i</sub>

t<sub>ci</sub>  
current timestamp when C<sub>i</sub> receives a message

t<sub>ci,m</sub>  
current timestamp when C<sub>i</sub> initiates a message m

t<sub>AS</sub> , t<sub>x</sub>  
respectively, the current timestamp of the AS and an MR component x when a message is received.

t<sub>AS,m</sub> , t<sub>x,m</sub>  
respectively, the current timestamp of the AS and an MR component x when a message m is initiated.

ΔT  
a maximum transmission delay between two entity

Msg<sub>m,x</sub>  
a message m generated by an entity x

LAT<sub>U_i</sub> and LVT<sub>U_i</sub>  
respectively, login authentication token and local verification token of U<sub>i</sub>

SignGen(PR<sub>x</sub> ,m)  
signature generation function, it signs a message m with entity x’s private key (i.e. signer’s private key)

Sig<sub>m,x</sub>  
a message signature of message m generated by an entity x

SignVer(PU<sub>x</sub> ,m, Sig<sub>m,x</sub>)  
signature verification function, it verifies the signature Sig<sub>m,x</sub> of message m using x’s public key

(·)  
one-way cryptographic hash function.

(·)  
concatenation operator

=  
assign operator, example: X = Y means Y is assigned to X

==  
equal operator e.g. (X == Y) means X is equal to Y

AuthN<sub>ci</sub>  
AuthN Agent of C<sub>i</sub>
6.3. CPCrE Protocol Design: an Overview

This section presents an overview of our CPCrE protocol for identifying a new client to the MR application as an MR-Client.

To authenticate an MR-Client to an MR application, the MR-Client has to be identified first. The Client identification, as defined in Section 5.8, is a VDAF function by which a new client is registered and identified as an MR-Client to the MR application. By performing this function, the authentication credential of the MR-Client is issued and its format is defined. The client identification could be done using a registration process in which a client registers him/herself, i.e. making him/herself known, to an MR application. The registration process could be achieved using a registration service that is either the authentication service itself or a third party trusted by the authentication service. In this process, a new client has also to identify him/herself to the registration service before issue him/her an authentication credential which is used later to identify him/herself to the MR application.

First before describing how the CPCrE protocol identifies the client, we motivate the need for the CPCrE protocol by highlighting the limitations of the current methods in identifying a client to the MR application. In most existing authentication methods, the client identification is not considered, or they assume that clients register themselves in person with the registration service. These methods which use this assumption, identify the new clients with a single-factor authentication secret (i.e. password). In other words, they issue an authentication credential that relies on just a one secret to authenticate the MR-Client (i.e. identified client) to the MR application. Making a physical appearance for new clients, is not always the case for the MR application, where clients should be identified remotely. As the author writing this thesis, none of the current authentication methods considers a secure client identification for remote clients. Currently, secure client identification for remote clients is required as there have been efforts to provide the MR application remotely in public clouds. However, how to issue and securely distribute an authentication credential to a client without requiring the client to make a physical appearance to the credential issuer, and how we could prevent a remote client gets hold of another client attributes and impersonates him. Therefore,
to address these limitations and challenges, we propose the CPCrE protocol that identify a remote client as an MR-Client efficiently and securely.

As mentioned above, when a client (i.e. a user) requests to access an MR application, the client should successfully pass through an authentication process, for which the client will need to make use of an authentication credential. To acquire the credential, the client needs to identify him/herself to a registration service. Here, in our design of the CPCrE protocol, it is assumed that the registration service is provided by the same entity that also provides the authentication service.

The CPCrE protocol is the first protocol of the LVAP suite by which the VDAF design requirement (F1) is achieved. By executing the CPCrE protocol, a new remote client will (i) be identified as an MR-Client by issuing his primary authentication and verification tokens, i.e. $PT_{C_i}^1 / PT_{C_i}^2$ and $PVT_{C_i}^1 / PVT_{C_i}^2$, and (ii) acquire the local and removable parts of his authentication credential which is constructed using $PT_{C_i}^1$ and $PT_{C_i}^2$ and stored into two separate storage.

An overview of the CPCrE protocol is depicted in Figure 6.3. As it can be seen from the figure, the CPCrE protocol is executed between a C-AuthN Agent and the G-AuthN Service. The C-AuthN Agent and the G-AuthN Service use a NA-IdenT Request and Response to exchange the identification and authentication information, respectively. The protocol consists of seven steps, each of them performs a specific function as following:

**Step1 - Define a Client Identification Information:** In this step, an identification information of a new client is defined. The C-AuthN Agent defines the client’s identification information which can be classified into three types of attributes and one secret key. The first type of attribute is a unique-attribute which has a unique value for each client, such as the identity of the client to be identified, $ID_{C_i}$, and the user’s email address $email_{U_i}$. The second type of attribute is a common-attribute which may have similar value between the clients such as $CVP_{C_i}$’s long-term credential validity period. The third type of attribute is a certificate attribute. It represents the client payment-attribute certificate, $PCert_{C_i}$. The client payment-attribute certificate comprises of both types of attributes as explained before in Section 6.2. The secret is a session password-secret key, $SK_{C_i}^{SP}$, which is also uniquely identified.
Step 2 - Send a NA-IdenT Request: The C-AuthN Agent sends a NA-IdenT Request message along with its signature (i.e. a NA-IdenT Request object) to the G-AuthN Service. The symmetric and asymmetric along with the digital signature algorithms, discussed in Section 5.10, are used to secure the NA-IdenT Request. The client’s identification information, defined in the previous step, is encrypted using $SK_{C_i}^{SP}$, and this symmetric key ($SK_{C_i}^{SP}$) is encrypted using the public key of the Authentication Server $PU_{AS}$. The two ciphertexts along with $ID_{C_i}$ and the timestamp of $C_i$ are signed with the client’s private key, $PR_{C_i}$.

Step 3 - NA-Request Verification: The G-AuthN Service verifies correctness of the NA-IdenT Request in term of two aspects. First, it verifies both the freshness and the authenticity of the received message. Second, it verifies the identification information which includes $PCert_{C_i}$.

Step 4 - Generate the Client Authentication Information: In this step, the G-AuthN Service generates the authentication information of the client to be identified. This authentication information is a set of three independent secret keys and a client location-secret code. The set of three secret keys contains the user password-secret key, $K_{C_i}$, the client master-secret key $P_{C_i}$, and the dynamic login-secret key, $SK_{C_i}^{DL}$. $K_{C_i}$ and $P_{C_i}$ are used as two-factor authentication secrets that represent something the MR-Client knows and possesses, respectively (details to follow). $SK_{C_i}^{DL}$ represents the current login of the user and it is renewed for every new login. The client location-secret code, i.e. $L_{C_i}$, indicates that somewhere the client is located when it is registered. $L_{C_i}$ is used as an additional authentication secret. These authentication information are used along with the identification information defined by the C-AuthN Agent in Step 1 - to constitute (i.e form) $PT_{C_i}^2$ and $PVT_{C_i}^2$ (details to follow in the next Section).

Step 5 - Send NA-IdenT Response: The G-AuthN Service sends a NA-AuthN Response message along with its signature (i.e. a NA-IdenT Response object) to the C-AuthN Agent. The symmetric key and digital signature algorithms, discussed in Section 5.10, are used to secure this response object. In this message, $PT_{C_i}^2$ and $SK_{C_i}^{DL}$ are encrypted using $SK_{C_i}^{SP}$. The ciphertext of this cryptographic operation along with the identity and the timestamp of the AS are signed using AS’s private key, $PR_{AS}$. 

133
Step 6 - NA-Response Verification: The C-AuthN Agent verifies the NA-IdenT Response message in term of the message freshness, integrity and origin, so the C-AuthN Agent can rely on the received authentication information to construct the local and removable parts of the MR-Client authentication credential (L_{ACR_{Ci}} and R_{ACR_{Ci}}) in the following step.

Step 7 - Prepare to MR-Client Authentication: In this step, L_{ACR_{Ci}} and R_{ACR_{Ci}} are constructed, so that the MR-Client can use this credential later to authenticate to the MR application. The C-AuthN Agent first generates a login password-secret key (SK_{C_i}^{LP}), a login authentication token LAT_{U_i}, and a local verification token LVT_{U_i}. All of them are used along with the SK_{C_i}^{DL} to constitute (i.e form) PT_{Ci}^1 and PV_{Ci}^1, PT_{Ci}^1 and PV_{Ci}^1 along with PT_{Ci}^2 which is received in the NA-IdenT Response are used to construct L_{ACR_{Ci}} and R_{ACR_{Ci}} as following: LAT_{U_i} is concatenated with the PT_{Ci}^2 to form the removable part of the MR-Client authentication credential, R_{ACR_{Ci}}. The R_{ACR_{Ci}} is encrypted using SK_{C_i}^{DL}, and stored in a removable storage disk, i.e. USB_{U_i} LVT_{U_i}, is concatenated with SK_{C_i}^{DL} to form the local part of the MR-Client authentication credential, L_{ACR_{Ci}}. L_{ACR_{Ci}} is encrypted using the SK_{C_i}^{LP} and embedded in a stego-object. The stego-object is saved locally in the local storage disk. Finally the user can install the CJ-AuthN Agent and uses it to (i) send authentication requests to the MR application and (ii) generate the primary authentication and verification tokens of his MR-Job Components.
6.4. MR-Client’s Primary Tokens and Key Structure Design

This section describes the design of the MR-Client’s primary authentication and verification tokens and their key structure in details, and it explains how the local and removable parts of the MR-Client credential is constructed using such tokens and key structure.

6.4.1. Primary Authentication and Verification Tokens

As discussed before in Section 5.5, each MR-Client issued with an MR-Client authentication credential, i.e. $ACR_{C_i}$, which consists of four tokens, two are primary tokens, i.e. $PT_{C_i}^1$ and $PT_{C_i}^2$, for long-term use, and two are secondary tokens, i.e. $ST_{C_i}^1$ and $ST_{C_i}^2$, for short-term use. For each of these tokens there is a corresponding verification token issued. This section describes the format of the primary authentication and verification tokens which
are issued using the CPCrE protocol and used to identify the MR-Client to the MR
application.

- **MR-Client’s Primary Authentication Tokens ($PT^1_{Ci}$ and $PT^2_{Ci}$):** Figure 6.4 shows the
design of both $PT^1_{Ci}$ and $PT^2_{Ci}$. As shown in the figure, $PT^1_{Ci}$ consists of three attributes:
an MR-Client ID attribute, a user-password attribute, and a login-token attribute, while
$PT^2_{Ci}$ consists of six attributes, named as a password-key attribute, a master-key
attribute, a location-code attribute, an issue-date attribute, a valid-from and a valid-to
attributes. The values of these attributes are chosen pragmatically and incorporated
technically to form (i) multi-factor authentication secretes, and (ii) the local and
removable parts of the MR-Client authentication credential, i.e. $L_{ACR_{Ci}}$ and $R_{ACR_{Ci}}$,
(details to follow in the key structure), that are used for MR-Client authentication to MR
application. Table 6.2 describes the value and the use of each one of these attributes.

![The primary authentication tokens of ACR_{Ci}](image)

**Figure 6.4 The data structure of the MR-Client’s primary authentication tokens**

**Table 6.2 The contents of the primary authentication tokens of an MR-Client**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR-Client ID attribute</td>
<td>MR-Client ID attribute is used to capture the MR-Client username which is used to distinguish each MR-Client user. Each MR-Client has a unique MR-Client identifier ($ID_{Ci}$). It is chosen by the user. A valid $ID_{Ci}$ is required for MR-Client authentication protocols, i.e. $U$-$AC$, $C$-$RM$ and $C$-$NN$-$AuthN$ protocols.</td>
</tr>
<tr>
<td>User-password attribute</td>
<td>User-password attribute is a secret password, $PW_i$, (word or string of characters) that is chosen, memorised and kept secret by the user. $PW_i$ along with $ID_{Ci}$ present the user authentication credential, i.e. $ACR_{Ci}$. It works as a first-factor authentication secret to authenticate the user to the MR Application Client locally in the client machine using $U$-$AC$_$AuthN$ protocol.</td>
</tr>
<tr>
<td>Login-token attribute</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td>Login-token attribute it is a secret attribute that contains a login authentication token of $U_i (LAT_U_i)$. $LAT_U_i$ is a dynamic login-secret key of $C_i (SKC_i)$, which is issued by the G-AuthN Service, and possessed and stored by the user in his removable storage media, i.e. $USB_U_i$. It works as a second-factor authentication secret to authenticate the user to the MR Application Client locally using U-AC_AuthN protocol.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Password-key attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password-key attribute is a secret attribute that represents something $C_i$ knows. It contains the user password-secret key, i.e. $KC_i$, which is approved by AS using $PRA_S$ and it used as a first-factor authentication secret to authenticate $C_i$ to the RM and AS using C-RM_AuthN protocol.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Master-key attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master-key attribute is also a secret attribute that represents something $C_i$ possesses. It contains the client master-secret key, i.e. $PC_i$, which is issued by the G-AuthN Service and used by $C_i$ as a second-factor authentication secret to authenticate $C_i$ to the RM and AS using C-RM_AuthN protocol.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location-code attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location-code attribute contains the client location-secret code, i.e. $LC_i$. It indicates somewhere the MR-Client is located. It is computed by the G-AuthN Service when $C_i$ is first identified to the MR application, and it is used as an additional-factor authentication secret to authenticate $C_i$ to the RM and AS using C-RM_AuthN protocol.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>An attribute value specifies when $PT^{2}_{C_i}$ is generated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Valid-from</th>
</tr>
</thead>
<tbody>
<tr>
<td>An attribute value specifies when $PT^{2}_{C_i}$ can be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Valid-to</th>
</tr>
</thead>
<tbody>
<tr>
<td>An attribute value specifies whether $PT^{2}_{C_i}$ is expired or not. It is used along with the value of the valid-from attribute to inform the client about the period of when his long-term authentication credential can be used.</td>
</tr>
</tbody>
</table>
- **MR-Client’s Primary Corresponding Verification Tokens** ($PVT_{Ci}^1$ and $PVT_{Ci}^2$): Figure 6.5 shows the design of both $PVT_{Ci}^1$ and $PVT_{Ci}^2$. As shown in the figure, $PVT_{Ci}^1$ consists of a login-key attribute and a local-token attribute. These attributes are approved using the login password-secret key and used to validate the user login which is identified using $PT_{Ci}^1$. $PVT_{Ci}^2$, in the other hand, consists of a set of seven attributes, named MR-Client ID attribute, a session-key attribute, a private-key attribute, a master-key attribute, a location-code attribute, a valid-from attribute, and a valid-to attribute. Five attributes of this set, as show in the figure, are approved by the AS using the user password-secret key, and they are used to validate the NJ-AuthN Request which is identified using $PT_{Ci}^2$ and received by the Resource Manager from the MR-Client. Table 6.3 describes the value and the use of each one of these attributes.

![Data structure of an MR-Client’s primary corresponding verification tokens](image)

**Figure 6.5** The data structure of an MR-Client’s primary corresponding verification tokens

**Table 6.3** The contents of the primary verification tokens of an MR-Client

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login-key attribute</td>
<td>Login-key attribute is a secret attribute that is defined by the G-AuthN Service and applied for $PT_{Ci}^1$. It contains the dynamic login-secret key ($SK_{Ci}^{DS}$) that is involved in validating a user’s login credential when the user logs in using his authentication credential and the login authentication token, i.e. $ACR_{U_i}$ and $LAT_{U_i}$.</td>
</tr>
<tr>
<td>Local-token attribute</td>
<td>Local-token attribute contains a local verification token of $U_i$ ($LVT_{U_i}$) which is issued and stored locally in the client machine. It is used along with $SK_{Ci}^{DL}$ to validate the user’s login credential. $LVT_{U_i}$ also validates the correctness of $PT_{Ci}^2$, which is stored in the $USB_{U_i}$. Thus, an MR-Client can use $PT_{Ci}^2$ to identify his request to the MR application.</td>
</tr>
<tr>
<td>MR-Client ID attribute</td>
<td>MR-Client ID attribute specifies MR-Client username. It is the same attribute applied for MR-Client’s primary authentication tokens.</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Session-key attribute</td>
<td>Session-key attribute contains a session password-secret key of $C_i (SK_{C_i}^2)$. It is required to generate the user password-secret key, which is applied for both $PT_{C_i}^2$ and $ST_{C_i}^1$, and used to validate the NJ-AuthN and WJ-AuthN Requests received by the Resource Manager and the Name Node, respectively.</td>
</tr>
<tr>
<td>private-key attribute</td>
<td>Private-key attribute is a secret attribute that represents the private key of the AS which is only known to the AS, and used to approve the user password-secret key.</td>
</tr>
<tr>
<td>Master-key attribute</td>
<td>Master-key attribute is a secret attribute which is applied for $PT_{C_i}^1$. It is used to validate the client master-secret key which is involved in identifying the NJ-AuthN Request received by the Resource Manager.</td>
</tr>
<tr>
<td>Location-code attribute</td>
<td>Location-code attribute is a secret attribute, which is also applied for $PT_{C_i}^2$, to validate the location of an MR-Client when the Resource Manager receives the NJ-AuthN Request from the MR-Client.</td>
</tr>
<tr>
<td>Valid-from, Valid-to</td>
<td>These attributes are the same attributes applied for $PT_{C_i}^2$. When $PT_{C_i}^2$ is verified, the valid-from and valid-to attributes can be revealed to confirm the credential validity,</td>
</tr>
</tbody>
</table>

### 6.4.2. Key Structure

This section explains the key generation and structure of the values of the attributes discussed above. As mentioned before in Section 6.3, the CPCrE protocol issues a variety of cryptographic secret keys and values of items to form the 1st and 2nd primary authentication tokens of an MR-Client, i.e. $PT_{C_i}^1$ and $PT_{C_i}^2$, and their corresponding verification tokens, i.e. $PTV_{C_i}^1$ and $PTV_{C_i}^2$. Figure 6.6 shows the key generation and structure of these tokens and how they are used to construct the $L_{ACR_{C_i}}$ and $R_{ACR_{C_i}}$. As can be seen from the figure, the issuance of these secret keys and values of items is a random basis. The issuance is subjected to the existence of a three entities (as indicated by the red dash-dot lines in Figure 6.6); a user, a C-AuthN Agent and the G-AuthN Service. The user enters his
authentication credential $ACR_{U_i}$, i.e. $ID_{C_i}$ and $PW_{U_i}$. The C-AuthN Agent and the G-AuthN Service generate a set of random numbers and timestamps. The username and the password along with this set of the random numbers and timestamps are used to generate the secret keys and the values of items. These secret keys and values of items are used to form $PT^1_{C_i}$ and $PT^2_{C_i}$ along with $PVT^1_{C_i}$ and $PVT^2_{C_i}$, some of which are used to construct the $L_{ACR_{C_i}}$ and $R_{ACR_{C_i}}$.

The key structure of $L_{ACR_{C_i}}$ consists of a local verification token ($LVT_{U_i}$), and a dynamic login-secret key ($SK^{DL}_{C_i}$), all of which are encrypted using a login password-secret key ($SK^{LP}_{C_i}$) to form $PVT^1_{C_i}$. $LVT_{U_i}$ is generated from a random number generated by the C-AuthN Agent ($r_{c_i}$), a timestamp when $r_{c_i}$ is generated ($t_{r_{c_i}}$), the $ACR_{U_i}$, and $SK^{DL}_{C_i}$. The $SK^{DL}_{C_i}$ is generated from a random number ($r_{1AS}$) which is generated by the G-AuthN Service, and a timestamp when $r_{1AS}$ is generated ($t_{r_{1AS}}$). $SK^{LP}_{C_i}$ is generated by the C-AuthN Agent, from $ID_{C_i}$ and $PW_{U_i}$, and used to encrypt the $L_{ACR_{C_i}}$. The encrypted $L_{ACR_{C_i}}$ is embedded in a stego-object and stored in the local disk of the client machine. $L_{ACR_{C_i}}$ (i.e. $LVT_{U_i}$ and $SK^{DL}_{C_i}$) is used (i) along with $ACR_{U_i}$ and the login authentication token, i.e. $LAT_{U_i}$ (which is a part of $R_{ACR_{C_i}}$) to authenticate the user login to the MR Application Client locally, and (ii) to protect the confidentiality and integrity of the 2nd primary authentication token of the MR-Client, i.e. $PT^2_{C_i}$. 
The key structure of $R_{ACR_C_i}$ consists mainly of two secret keys and two secret values of items. The two secret keys are the client master-secret key ($P_{C_i}$) and the user password-secret key ($K_{C_i}$). $P_{C_i}$ is generated from $ID_{C_i}$, a random number $r_{2AS}$ generated by G-AuthN Service, and the timestamp when $r_{2AS}$ is generated $t_{r_{2AS}}$. $P_{C_i}$ represents something the MR-Client possesses. This is because the $P_{C_i}$ is generated uniquely by the G-AuthN Service, and it is possessed only by this particular client ID and stored in his removable storage media, i.e. an external disk $USB_{U_i}$. $K_{C_i}$ represents something the client knows. This is because the
user password-secret key is mainly generated from the session password-secret key ($SK_{SP}^C$) which contains the password known just by the user. The user password-secret key is approved\(^{12}\) by the G-AuthN Service using the AS’s private key ($PR_{AS}$). $SK_{SP}^C$ is a symmetric key generated by the C-AuthN Agent using $PW_{U_i}$, $r_{ci}$, and $t_{AS}$. It is used to protect the confidentiality of the protocol messages and it is sent to the G-AuthN Service using asymmetric key cryptosystem as explained in the Step2 of the CPCrE protocol. The two secret values of items are the client location-secret code ($LC_i$) and the login authentication token ($LAT_{U_i}$). $LC_i$ is generated from $ID_{ci}$, the timestamp when the client is identified ($t_{AS}$), and the IP address of the client host ($IP_{CH_Ui}$). As mentioned above, $LC_i$ is used to indicate from where the client is identified. $LAT_{U_i}$ is generated from $r_{ci}$, $t_{ci}$ and $SK_{DL}^C$. $R\_ACR_{ci}$ is encrypted using $SK_{DL}^C$ and stored in USB $U_i$. $SK_{DL}^C$ is encrypted and located separately on the local disk. Part of the contents of $R\_ACR_{ci}$, i.e. $LAT_{U_i}$, along with $ACR_{U_i}$ represent (i.e. used to form) $PT_{ci}^1$, $P_{ci}$, $K_{ci}$ and $LC_i$ are used to form $PT_{ci}^2$. Further, $P_{ci}$ along with $SK_{SP}^C$ and $PR_{AS}$ are used to form $PVT_{ci}^2$, where $P_{ci}$ is encrypted using $K_{ci}$. $K_{ci}$ is generated whenever it is needed (to decrypt the encrypted $P_{ci}$ using $SK_{SP}^C$ and $PR_{AS}$) to validate a NJ-AuthN Request identified using $PT_{ci}^2$.

The security of the secret keys of $PVT_{ci}^2$ is also essential. Anyone who can obtain the secret keys of $PVT_{ci}^2$ can use it to impersonate the AS during the MR-Client authentication exchanges. Therefore, measures should be taken to assure that these secret keys are protected from unauthorised use. An attacker with access to the AS might use low-level disk utilities to locate the contents of the $PVT_{ci}^2$ on the hard disk. In other words, the attacker may steal the encrypted $P_{ci}$ and both $SK_{SP}^C$ and $PR_{AS}$ from the hard disk. To mitigate the risks associated with the AS compromise, whether via network or device theft, it is important to (i) apply a tamper resistance technology to store the AS’s private key, i.e. $PR_{AS}$, rather than to store it on the AS’s hard disk drive (i.e. a local Database), and (ii) store the encrypted $P_{ci}$ not only in a separate location (i.e. separate server) from where the $SK_{SP}^C$ and $PR_{AS}$ are stored, but also in a different network segment behind additional layers of security [96]-[100].

\(^{12}\) Details about how an MR-AuthN Component approves authentication information are given in the Definitions Section.
The design of these tokens and their key structure has the following three-folds. First, this key structure provides a confidentiality and integrity protection to the primary authentication tokens of the MR-Client, i.e. protects the MR-Client’s primary authentication tokens form unauthorised access. Second, the keys are generated automatically without any user’s involvement except the user has to submit his user name and password to the C-AuthN Agent. Third, this key structure supports a local verification process in the client machine before any remote verification process takes place through the MR-Client authentication. The local verification process could (i) mitigate a number of known attacks that may be mounted against the user’s account and credential, and (ii) ensures the correctness of the MR-Client’s primary authentication tokens as the three entities; $U_i$ who knows the $PW_{U_i}$, $CH_{U_i}$ which holds $L_{ACR_{C_i}}$, and $USB_{U_i}$ which holds $R_{ACR_{C_i}}$, have to be exist in the same time (details to follow in Sections 6.6.1 and 7.9.1).

6.5. CPCrE Protocol in Details

The CPCrE protocol is described in details below and shown in Figure 6.7.

Step1 - Define a Client Identification Information: This function is executed by $U_i$ and C-AuthN Agent ($AuthN_{C_i}$). The details are given below.

- $U_i$ defines the following: $C_i$’s username, $ID_{C_i}$, $U_i$’s password, $PW_{U_i}$, $U_i$’s email address, $email_{U_i}$, the $C_i$’s payment-attribute certificate, $PCert_{C_i}$, and $C_i$’s long-term credential validity period, $CVP_{C_i}$. Here $ID_{C_i}$ must be the same identifier which is selected in the payment process and used to issue the client payment-attribute certificate. $PW_{U_i}$ should be memorized by $U_i$. The email address will be used as an out-of-band channel\textsuperscript{13}. $CVP_{C_i}$ defines the duration during which $C_i$ would like to use the MR application. The value of this attribute, i.e. $CVP_{C_i}$, is defined by the client and it is often specified by the amount of payment that the client has paid to the MR application provider. $PCert_{C_i}$ proves that the client has paid the registration fees and used to verify the duration requested by the client to use the MR application (i.e. $CVP_{C_i}$).

\textsuperscript{13} Out of band channel means a way of communication used to exchange data between the G-AuthN Service and the C-AuthN Agent rather than using the communication session established between them. Example of this is using an email system rather than using the CPCrE protocol communication session itself to send the CJ-AuthN Agent (details to follow).
Once the $U_i$ has selects the above attributes and secrets, the C-AuthN Agent does the following operations:

- It generates a random secret number, $r_{c_i}$, and it specifies the timestamp when this random number is generated, $t_{r_{c_i}}$. This random number and the respective timestamp are used (i) along with the user’s password to generate the session password-secret key ($SK_{C_i}^{SP}$) and the local verification tokens ($LVT_{U_i}$), and (ii) to generate both the login authentication token $LAT_{U_i}$. The issuance and the use of $SK_{C_i}^{SP}$, $LAT_{U_i}$, and $LVT_{U_i}$ have been explained before in Section 6.4.

- It reads the IP address of the user machine, $IP_{CH_{U_i}}$, from the network connection of the client host. $IP_{CH_{U_i}}$ is used by the G-AuthN Service to compute the client location-secret code, $L_{C_i}$. The $L_{C_i}$ indicates that from where the client has been identified. This information is used for MR-Client authentication (details to follow).

- It issues the session password-secret key, $SK_{C_i}^{SP} = \mathbb{H} (PW_{U_i} || r_{c_i} || t_{r_{c_i}})$.

- It constructs $Idl_{C_i} = \{IP_{CH_{U_i}} || email_{U_i} || CVP_{c_i} || PCert_{U_i}\}$, where, $Idl_{C_i}$ represents the identification information which is sent in the NA-IdenT Request.

- It then encrypts $Idl_{C_i}$ using $SK_{C_i}^{SP}$, i.e. $E (SK_{C_i}^{SP}, Idl_{C_i})$. Once the $Idl_{C_i}$ is encrypted, the C-AuthN Agent encrypts $SK_{C_i}^{SP}$ using the public key of the AS, i.e. $Enc\ (PU_{AS}, SK_{C_i}^{SP})$, where $E$ and $Enc$ represent the symmetric and asymmetric key algorithms, which are discussed in Section 5.10. These two encryptions operations, i.e. $E (SK_{C_i}^{SP}, Idl_{C_i})$ and $Enc\ (PU_{AS}, SK_{C_i}^{SP})$, are used to protect the confidentiality of $Idl_{C_i}$ and $SK_{C_i}^{SP}$ against eavesdropping attacks and unauthorised entities.

- It constructs the NA-IdenT AuthN Request message, i.e. $Msg_{NA,C_i} = \{ E (SK_{C_i}^{SP}, Idl_{C_i}) || Enc\ (PU_{AS}, SK_{C_i}^{SP}) || ID_{C_i} || t_{C_i,NA} \}$, where $ID_{C_i}$ is the identity of $C_i$, $t_{C_i,NA}$ is $C_i$’s current timestamp when NA-IdenT Request is constructed and it used to resist replay attacks.
It computes the digital signature of $Msg_{NA,C_i}$, i.e. $\text{Sig}_{NA,C_i} = \text{SignGen}(PR_{C_i},Msg_{NA,C_i})$, where $\text{SignGen}$ is the digital signature function (generator) which is discussed in Section 5.10, $PR_{C_i}$ is $C_i$’s private key, and $Msg_{NA,C_i}$ is the NA-IdenT Request message of $C_i$. This operation is used to resist any forgery or unauthorised modification of any of the message contents (i.e. active attacks).

Finally, C-AuthN Agent constructs a NA-IdenT Request object, i.e. $\{Msg_{NA,C_i}||\text{Sig}_{NA,C_i}\}$.

**Step2 - Send a NA-IdenT Request:** In this step, the C-AuthN Agent ($AuthN_{C_i}$) sends the NA-IdenT Request to the G-AuthN Service ($AuthN_{AS}$).

$$AuthN_{C_i} \rightarrow AuthN_{AS} : \text{NA-IdenT Request} \{Msg_{NA,C_i}||\text{Sig}_{NA,C_i}\}$$

Where $Msg_{NA,C_i} = \{ E(SK_{C_i}^{SP},IdC_i)|| Enc( PU_{AS},SK_{C_i}^{SP}) || ID_{C_i} || t_{C_i,NA}\}$, and $\text{Sig}_{NA,C_i} = \text{SignGen}(PR_{C_i},Msg_{NA,C_i})$.

**Step3 - NA-Request Verification:** In this step, upon the receipt of the NA-IdenT Request, the G-AuthN Service verifies the correctness of (i) the received NA-IdenT Request (i.e. $Msg_{NA,C_i}||\text{Sig}_{NA,C_i}$), and (ii) the identification information that identifies the client to be as an MR-Client. To do so (i.e. to satisfy (F1.1)), it does the following operations.

- G-AuthN Service reads $C_i$’s identity and current timestamp, i.e. $ID_{C_i}$ and $t_{C_i,NA}$, along with the signature of the NA-IdenT Request message $\text{Sig}_{NA,C_i}$.
- The G-AuthN Service then verifies the NA-IdenT Request in term of:
  - Freshness: It checks if the difference between its local timestamp ($t_{AS}$) and the timestamp contained in the message ($t_{C_i,NA}$) is less than a predefined value ($\Delta T$), i.e. $(t_{AS} - t_{C_i,NA} < \Delta T)$.
  - Trustworthiness of the signature verification key: when the message freshness was valid, the G-AuthN Service checks if $C_i$’s public key is trustworthy. In other words, it checks if the public key of the claimed client, $ID_{C_i}$, is certified by a trustworthy CA and is not expired. If this key is valid, this means that the sign process of the message using the private key (the sender secret key) should be authentic and still valid, and the G-AuthN Service then verifies the message signature. Otherwise it rejects the request.
message, considers the message as invalid message, and terminate the session.

- **Signature:** It checks the correctness of the digital signature of the C-AuthN Agent on $Msg_{NA,C_i}$ which is computed in Step1. This is done by using $SignVer()$ function which is described in Section 5.10, i.e. $SignVer(PU_{C_i},Msg_{NA,C_i},Sig_{NA,C_i})$. This function receives $C_i$’s public key, $PU_{C_i}$, and both $Msg_{NA,C_i}$ and its digital signature $Sig_{NA,C_i}$, that are received in the NA-IdenT Request. If the output of $SignVer()$ function is valid (i.e. a positive), then the message is indeed generated by the C-AuthN Agent, as the signature operation is signed with $C_i$’s private key. Otherwise, if the verification fails, the G-AuthN rejects the request message, considers the message as invalid message, and terminate the session.

- **G-AuthN Service decrypts** the encrypted $SK_{C_i}^{SP}$ using $PR_{AS}$, i.e. $Enc^{-1}(PR_{AS},SK_{C_i}^{SP})$.

- It then uses $SK_{C_i}^{SP}$ to decrypt the encrypted $IdI_{C_i} = \{IP_{CH_i} || email_{U_i} || CVP_{C_i} || PCert_{U_i}\}$, defined in Step1, i.e. $E^{-1}(SK_{C_i}^{SP},IdI_{C_i})$.

- Once the G-AuthN Service decrypts $IdI_{C_i}$, it then reads the identification information which includes the client payment-attribute certificate, i.e. $PCert_{C_i}$, and verifies the identification information in term of:

  - **Payment:** It verifies the correctness of his payment-attribute certificate, $PCert_{C_i}$, described in Section 6.2. The G-AuthN Service first checks if the PS’s public key, $PU_{PS}$, which is used to sign the contents of $C_i$’s payment-attribute certificate, is trustworthy. In other words, it checks if the public key of the PS is certified by a trustworthy CA and is not expired. If the $PU_{PS}$ is valid, the G-AuthN Service use $PU_{PS}$ to verify the signature of the $PCert_{C_i}$. If it is valid, the G-AuthN Service then compares $C_i$’s identity, which is in the payment-attribute certificate, $ID_{C_i}'$, with the one in the NA-IdenT Request received in Step2, i.e. $ID_{C_i}$. This is to ensure that the one (i.e. the client) who sent the message is the same one who owns the payment-attribute certificate. If they are equal, the G-AuthN Service
proceeds to verify the allowed period requested by the client. Otherwise, it rejects the request, considers it as invalid request, and terminate the session.

- Expiry date and allowed period: the G-AuthN Service checks if (i) the \( C_i \)'s payment-attribute certificate is not expired, and (ii) the period requested by \( C_i \) (using \( CVP_{C_i} \) attribute which is defined in Step1 -) is valid. In other words, the G-AuthN Service verifies the requested period for the long-term credential (i.e. \( PT_{C_i}^1 \) and \( PT_{C_i}^2 \)), where \( CVP_{C_i} = \{ t_s : \text{not before}, t_n : \text{not after} \} \). It does these two verification processes by checking if (i) the aggregate value of the current time when the request is received, \( t_{AS} \), and the allowed period in \( C_i \)'s payment-attribute certificate, \( AP_{C_i} \), is less than or equal to the \( C_i \)'s account expiry date, \( Exp_{C_i} \), i.e. \( t_{AS} + AP_{C_i} \leq Exp_{C_i} \), and (ii) the difference between \( t_n \) (not after) and \( t_s \) (not before), i.e. the duration for using the MR application is less than or equal the allowed period in the \( C_i \)'s payment-attribute certificate, i.e. \( t_n - t_s \leq AP_{C_i} \).

**Step4 - Generate a Client Authentication Information:** This function is executed by G-AuthN Service (\( AuthN_{AS} \)). If all the verifications in Step3 - are positive, the G AuthN Service generates the dynamic login-secret key of \( C_i \), i.e. \( SK_{C_i}^{DL} \), and the second primary authentication and verification tokens of the MR-Client, i.e. \( PT_{C_i}^2 \) and \( PVT_{C_i}^2 \). In details, the G-AuthN Service does the following operations:

- It generates two random secret numbers, \( r_{1AS} \) and \( r_{2AS} \) and specifies the timestamps when these numbers are generated, \( t_{r1AS} \) and \( t_{r2AS} \), respectively. As explained in Figure 6.6, these two random numbers and the respective timestamps are used along with other \( C_i \)'s attributes, defined in Step1 - , to generate \( SK_{C_i}^{DL} \), \( PT_{C_i}^2 \), and \( PVT_{C_i}^2 \) (satisfy (F1.2) and (F1.3)) as following:

  $$
  \{ \begin{array}{l}
  SK_{C_i}^{DL} = \mathbb{H}(r_{1AS} \parallel t_{r1AS}) \\
  PT_{C_i}^2 = \mathbb{H}(ID_{C_i} \parallel r_{2AS} \parallel t_{r2AS}) \\
  P_{C_i} = \mathbb{H}(ID_{C_i} \parallel r_{2AS} \parallel t_{r2AS}) \\
  L_{C_i} = \mathbb{H}(ID_{C_i} \parallel IP_{CH_i} \parallel t_{AS}) \\
  \quad \text{IssueDate} = t_{AS} \\
  \quad \text{ValidFrom} = t_s \\
  \quad \text{ValidTo} = t_n
  \end{array} \}
  $$

147
Where, $\mathbb{H}$ is a one-way cryptographic hash function and $E$ is a symmetric key encryption algorithm.

- It encrypts $SK_{C_{1}^{DL}}$ and $PT_{C_{1}^{2}}$ using $SK_{C_{1}^{SP}}$ which is received in the NA-IdenT Request and shared with the C-AuthN Agent, i.e. the $E (SK_{C_{1}^{SP}} , \{ SK_{C_{1}^{DL}} || PT_{C_{1}^{2}} \})$ to protect the confidentiality of both $SK_{C_{1}^{DL}}$ and $PT_{C_{1}^{2}}$ against eavesdropping by attackers or unauthorised entities.
- It constructs the NA-IdenT Response message, i.e. $Msg_{NA,AS} = E (SK_{C_{1}^{SP}} , \{ SK_{C_{1}^{DL}} || PT_{C_{1}^{2}} \}) \ || t_{AS,NA} \ || ID_{AS}$, where $ID_{AS}$ is AS’s identity, and $t_{AS,NA}$ is AS’s current timestamp and it is used to resist replay attacks.
- It computes the digital signature of $Msg_{NA,AS}$ using $PR_{AS}$ i.e. $Sig_{NA,AS} = SignGen( PR_{AS} , Msg_{NA,AS} )$. This operation is used to resist any active attacks.
- Finally, G-AuthN Service constructs NA-IdenT Response object, i.e. $\{ Msg_{NA,AS} \| Sig_{NA,AS} \}$.

**Step5 - Send a NA-IdenT Response:** In this step, the G-AuthN Service ($AuthN_{AS}$) sends a NA-IdenT Response object to the C-AuthN Agent ($AuthN_{C_{1}}$) using the same communication session (in-band). It also sends the CJ-AuthN Agent ($AuthN_{C_{1}^{I}}$) using $U_{i}$’s email address (i.e. $email_{U_{i}}$) as an out-of-band communication.

$$AuthN_{AS} \rightarrow AuthN_{C_{1}} : \text{NA-IdenT Response} \{ Msg_{NA,AS} \| Sig_{NA,AS} \}$$

$$AuthN_{AS} \quad \text{via} \quad email_{U_{i}} \quad U_{i} : \{ AuthN_{C_{1}^{I}} \}$$

Where $Msg_{NA,AS} = E (SK_{C_{1}^{SP}} , \{ SK_{C_{1}^{DL}} || PT_{C_{1}^{2}} \}) \ || t_{AS,NA} \ || ID_{AS}, Sig_{NA,AS} = SignGen( PR_{AS} , Msg_{NA,AS} )$, and $AuthN_{C_{1}^{I}}$ is the CJ-AuthN Agent that is designed using a set of cryptographic algorithms and methods and it is used for MR-Client authentication and MR-Job Components identification functions.
Step6 - NA-Response Verification: In this step, upon the receipt of the NA-IdenT Response, the C-AuthN Agent verifies the correctness of the received NA-IdenT Response. To do so, it does the following operations.

- C-AuthN Agent reads $ID_{AS}$ and $t_{AS,NA}$ along with $Sig_{NA,AS}$.
- C-AuthN Agent verifies the received NA-IdenT Response in term of:
  - Freshness: it checks if the difference between the local timestamp of the client ($t_{Ci}$) and the timestamp contained in the message ($t_{AS,NA}$) is less than a predefined value ($\Delta T$), i.e. $t_{Ci} - t_{AS,NA} < \Delta T$.
  - Trustworthiness of the signature verification key: when the message freshness is valid, the C-AuthN Agent checks if the AS’s public key is trustworthy. In other words, it checks if the public key of the claimed AS, $ID_{AS}$, is certified by a trustworthy CA and is not expired. If it is valid, this means that the sign process of the message using the private key (the sender secret key) should be authentic and still valid, and the C-AuthN Agent then verifies the message signature. Otherwise it rejects the response message, considers the message as invalid message, and terminate the session.
  - Signature: It checks the correctness of the digital signature of the G-AuthN Service on $Msg_{NA,AS}$ which is computed in Step1-. This is done using $SignVer()$ function, i.e. $SignVer(PU_{AS}, Msg_{NA,AS}, Sig_{NA,AS})$. This function receives AS’s public key and both the message and its signature that are received in the NA-IdenT Response. If the output of the function is valid, then the message is indeed generated by the G-AuthN Service, as the signature operation is signed with the AS’s private key. In this case, the C-AuthN Agent will proceed to the next step. Otherwise, if the verification fails, the C-AuthN Agent rejects the NA-IdenT Response message, considers the message as invalid message, and terminate the session.

Step7 - Prepare for MR-Client Authentication: This function is executed by the C-AuthN Agent. If all verifications’ processes in the previous step are positive, the C-AuthN Agent constructs the local and removable parts of the MR-Client authentication credential, i.e. $L_{ACR_{Ci}}$ and $R_{ACR_{Ci}}$ (satisfy (F1.5)). In details, the C-AuthN Agent does the following operations:
- It decrypts the encrypted $SK_{CL_i}^{DL}$ and $PT_{CL_i}^2$, using $SK_{CL_i}^{SP}$, which is generated in Step1 - i.e. $E^{-1}(SK_{CL_i}^{SP}, [SK_{CL_i}^{DL} || PT_{CL_i}^2])$.

- It then generates a login authentication token, $LAT_{U_i}$, by computing $LAT_{U_i} = \tau (r_{c_i} || t_{r_{c_i}} || SK_{CL_i}^{DL})$, where $r_{c_i}$ is the random number generated in Step1 -, and $t_{r_{c_i}}$ is the timestamp when this number is generated.

- It concatenates $LAT_{U_i}$ with $PT_{CL_i}^2$ to form the $R_{ACR_{CL_i}}$, i.e. $R_{ACR_{CL_i}} = LAT_{U_i} || PT_{CL_i}^2$.

- It encrypts $R_{ACR_{CL_i}}$ using $SK_{CL_i}^{DL}$ which is received from the G-AuthN Service, i.e. $E(SK_{CL_i}^{DL}, R_{ACR_{CL_i}})$.

- It then stores the ciphertext, i.e. $E(SK_{CL_i}^{DL}, R_{ACR_{CL_i}})$, to a removable storage media possessed by $U_i$, i.e. $USB_{U_i}$.

  $$AuthN_{CL_i} \rightarrow USB_{U_i} : \{E(SK_{CL_i}^{DL}, R_{ACR_{CL_i}})\}$$

- It requests the user to re-login using his $ACR_{U_i}$ to ensure that the authorised user is still the one who runs this step of the protocol.

  $$U_i \rightarrow AuthN_{CL_i} : Login \{ID_{CL_i}, PW_{U_i}\}$$

- Once $U_i$ enters $ID_{CL_i}$ and $PW_{U_i}$, the C-AuthN Agent compares them with $ID_{CL_i}$ and $PW_{U_i}$ which are selected by $U_i$ in Step1 -. If they match, the C-AuthN Agent generates the login password-secret key, i.e. $SK_{LP}^{ID} = \tau (ID_{CL_i} || PW_{U_i})$, otherwise if the user fails to enter $ID_{CL_i}$ and $PW_{U_i}$, the C-AuthN Agent terminates the process.

- It generates a local verification token of $U_i$, i.e. $LVT_{U_i}$, by computing $LVT_{U_i} = \tau (\tau (ID_{CL_i} || PW_{U_i}) || \tau (r_{c_i} || t_{r_{c_i}} || SK_{CL_i}^{DL}))$.

- It concatenates $LVT_{U_i}$ with $SK_{CL_i}^{DL}$ to form the $L_{ACR_{CL_i}}$, i.e. $L_{ACR_{CL_i}} = LVT_{U_i} || SK_{CL_i}^{DL}$.

- It encrypts the $L_{ACR_{CL_i}}$ using the $SK_{LP}^{ID}$, i.e. $E(SK_{LP}^{ID}, L_{ACR_{CL_i}})$. It then embeds the encrypted value ($E(SK_{LP}^{ID}, L_{ACR_{CL_i}})$) to a stego-object, i.e. Stego-File{$E(SK_{LP}^{ID}, L_{ACR_{CL_i}})$}. Both the encryption and the steganography mechanisms are used to protect both the $L_{ACR_{CL_i}}$ and $R_{ACR_{CL_i}}$ against unauthorised use (i.e. access) in particular when $CH_{U_i}$ is compromised.

- Finally, $U_i$ installs the CJ-AuthN Agent, $AuthN_{CL_i}^J$, so $C_i$ can (1) send his NJ-AuthN and WJ-AuthN Requests to the Resource Manager and the Name Node using C-RM and C-NN-AuthN protocols, respectively, and (2) generate the primary authentication and
verification tokens of his MR-Job Components using JCPCrE protocol as explained in the following chapter.

Figure 6.7 CPCrE protocol in detail
6.6. Security Analysis and Verification

This section gives an informal analysis of the CPCrE protocol against the security requirements specified in Section 6.2.2.2. To provide further proof of the security of the protocol, we have modelled and performed formal verification of the protocol using the Automated Validation of Internet Security Protocols and Applications tool (AVISPA).

6.6.1. Informal Analysis

The propositions below capture the security requirements specified in Section 6.2.2.2. This section explains how the CPCrE protocol can achieve mutual authentication, message confidentiality and authenticity, and a confidentiality and integrity protection for MR-Client’s primary authentication tokens. It also discusses how the protocol withstands threats and attacks, such as those discussed in Section 5.6.1 that may be mounted against its functionality, and how it can support strong MR-Client authentication to MR application.

Propositions 1: The CPCrE protocol can support mutual authentication and message authenticity (satisfy (S1) and (S2))

These two security features mean that (i) an adversary could not impersonate any new client sends a NA-IdenT Request to the AS or impersonate the AS sends a NA-IdenT Response to the client and (ii) any modification on the protocol messages can be detected. Each message in the CPCrE protocol contains a digital signature. The C-AuthN Agent generates the digital signature of the NA-IdenT Request message, $\text{Sig}_{NA,C_1}$, and the G-AuthN Service generates the digital signature of the NA-IdenT Response message, $\text{Sig}_{NA,AS}$, using their private keys, $PR_{C_1}, PR_{AS}$, respectively. Both digital signature operations, i.e. signature generation and verification, are explained in Section 5.10. $\text{SignGen}()$ is considered to provide a secure digital signature. Secure digital signature means that it is computationally infeasible to generate a signature without knowing the private key, and computationally infeasible to find a message with a given signature or two messages with the same signature [101] [102]. This could be achieved when three aspects are valid; the first one is that the hash function used for digital signatures requires collision resistance, the second one is that the respective entity’s signature signing key is stored on a tamper-proof environment, and the third one is that the corresponding signature verification keys (i.e. $PU_{C_1}$ and $PU_{AS}$ ) are certified by a trusted CA. Taking this consideration, the
CPCrE protocol then ensures the following: source authentication (of both the client and the AS), non-repudiation of origin and integrity of the protocol messages (i.e. NA-IdenT Request and NA-IdenT Response messages) (satisfy (S1) and (S2)). Any active attacks on the contents of any of these messages in transit can be detected and the modified message discarded.

**Proposition 2: The CPCrE protocol can support message confidentiality (satisfy (S3))**

In the CPCrE protocol, the protocol messages, i.e. the NA-IdenT Request and NA-IdenT Response messages, are confidentiality protected. This is because (i) the new client’s identification information, i.e. \(IdI_{CI}\), is encrypted using the session password-secret key (i.e. \(SK^{SP}_{CI}\)) which is encrypted at the source (C-AuthN Agent) using the AS’s public key (i.e. \(PU_{AS}\)) before being sent to the AS, and (ii) both the dynamic login-secret key (i.e. \(SK^{DL}_{CI}\)) and the second primary authentication token of the MR-Client (i.e. \(PT^{2}_{CI}\)) are encrypted at the source (G-AuthN Service) using \(SK^{SP}_{CI}\) which is received from the C-AuthN Agent. The CPCrE protocol ensures that the G-AuthN service is the MR-AuthN Component can decrypt the encrypted \(SK^{SP}_{CI}\), as it is the only component has \(PR_{AS}\). The protocol also ensures that the G-AuthN Service and the C-AuthN Agent are the MR-AuthN Components that can decrypt \(IdI_{CI}\) along with \(SK^{DL}_{CI}\) and \(PT^{2}_{CI}\), respectively. This is because (i) they are the only MR-AuthN Components knowing (i.e. sharing) \(SK^{SP}_{CI}\), and (ii) taking into account that (1) the used symmetric key algorithm, i.e. 3DES/AES, is secure against chosen plaintext attacks (assuming the use of initialisation vector or nonce in combination with a cryptographic mode such as Cipher Block Chaining CBC [103][104]), and (2) the used asymmetric key algorithm, i.e. RSA, is also secure against chosen plaintext attacks (assuming the use of OAEP: Optimal Asymmetric Encryption Padding – away of format the message before encryption in order to reach a higher security level [26] [105]). Therefore, our method satisfy (S3). In other words, only the C-AuthN Agent and the G-AuthN Service can access the messages’ contents (i.e. \(IdI_{CI}\) and \(SK^{DL}_{CI}\) along with \(PT^{2}_{CI}\)).

By including the message signature and the timestamp in each protocol message, and by taking into account the assumption A7 (in Section 5.6.2) the protocol also ensures that the received message is fresh. Therefore, based on all achievement of these security features, i.e. message authenticity (origin and integrity), confidentiality and freshness, the other registered or unregistered (i.e. curious) clients and attackers would not be able to perform a
MITM attack to impersonate a legitimate or masquerade as a new authorised client using CPCrE protocol.

**Proposition 3: The CPCrE protocol can support strong MR-Client authentication to MR application (satisfy (S4))**

This security property means that it is hard for an attacker to impersonate an MR-Client to abuse the MR application. To achieve a strong MR-Client authentication to MR application, the CPCrE protocol makes use of separation of tokens and secret storage locations, where the tokens are protected using static and dynamic secret keys along with a steganography mechanism. The protocol issues two primary authentication tokens that identify the MR-Client to the MR application, i.e $PT^1_{Ci}$ and $PT^2_{Ci}$, and they are confidentiality and integrity protected. As discussed before in Section 6.4, $PT^2_{Ci}$ is concatenated with $LAT_{Ui}$ (part of $PT^1_{Ci}$) to form the removable part of the MR-Client authentication credential, $R_{ACR_{Ci}}$. $R_{ACR_{Ci}}$ is encrypted using $SK^DL_{Ci}$, i.e. $E(SK^DL_{Ci}, R_{ACR_{Ci}})$, and stored in $USB_{Ui}$. $SK^DL_{Ci}$ is a dynamic login-secret key and concatenated with $LVT_{Ui}$ to form the local part of the MR-Client authentication credential, $L_{ACR_{Ci}}$. $L_{ACR_{Ci}}$ is encrypted using $SK^{LP}_{Ci}$, i.e. $E(SK^{LP}_{Ci}, L_{ACR_{Ci}})$, which is generated using the user password $PW_{Ui}$. $SK^{LP}_{Ci}$ is a static key and it could be updated only when $PW_{Ui}$ is renewed. The encrypted $L_{ACR_{Ci}}$ is embedded in a local stego-file (i.e. stego-object) in which the encrypted $L_{ACR_{Ci}}$ could be PIN protected.

The stego-object is stored locally in the user machine, $CH_{Ui}$, which is in a different location from where $E(SK^DL_{Ci}, R_{ACR_{Ci}})$ is stored. Beside the use of $LAT_{Ui}$ and $LVT_{Ui}$ in the authentication of the user login locally in $CH_{Ui}$, they ensure the integrity of both $L_{ACR_{Ci}}$ and $R_{ACR_{Ci}}$ before using $PT^2_{Ci}$ to authenticate the MR-Client to the RM and the AS, remotely. To authenticate MR-Client to the RM and the AS, three items have to exist at the same time to facilitate such authentication; (i) the user, $Ui$, who memorise the password, (ii) the removable storage media, $USB_{Ui}$, in which the encrypted $R_{ACR_{Ci}}$ is stored, and (iii) the user machine, $CH_{Ui'}$ where the stego-object is stored and used to hide the encrypted $L_{ACR_{Ci}}$. Thus, compromising the user password merely would not help an attacker to impersonate an MR-Client.

Further, to impersonate a remote MR-Client by identifying his NJ-AuthN Request, an adversary needs to obtain the primary authentication token of the MR-Client $PT^2_{Ci}$ which
contained in $R_{ACR_C_i}$ and stored in the removable storage media of the user $USB_{U_i}$. If $USB_{U_i}$ is stolen or lost, it is hard for an adversary to access any contents of $PT_{C_i}^2$. This is because $R_{ACR_C_i}$ is encrypted using $SK_{C_i}^{DL}$ and the adversary has to know the encryption key to recover the $PT_{C_i}^2$. Also, as mentioned above, $SK_{C_i}^{DL}$ is a dynamic secret key. This key is renewed for each new (next) login using MR-CAuthN protocols (details to follow in the next chapter), so that if an attacker was able to compromise the $SK_{C_i}^{DL}$ from the previous authentication session, the attacker would don’t be able to use it to abuse the primary authentication token of the MR-Client and impersonate an MR-Client. To know the $SK_{C_i}^{DL}$ for the next login authentication, the adversary has not only to extract (detect) the hidden encrypted value of the $SK_{C_i}^{DL}$ which is embedded in the stego-object, but also it has to know the $SK_{C_i}^{DP}$, which is used to encrypt the $SK_{C_i}^{DL}$.

Therefore, the CPCrE protocol provides strong MR-Client authentication to MR application, as the CPCrE protocol (i) minimises the risk of impersonating an MR-Client when (1) his username and password pair is compromised, or (2) when the $USB_{U_i}$ is lost or stolen, and (ii) provides assurance of the confidentiality and integrity protection for the primary authentication tokens of an MR-Client, (so satisfy (S4)). All other registered or unregistered (i.e. curious) clients and attackers would not be able to access the primary authentication tokens of the MR-Client as they are encrypted and stored in two separate storages (i.e. locations). Thus it is difficult to impersonate a legitimate client.

**Proposition 4: The CPCrE protocol can support a secure remote client payment (satisfy (S5))**

In this security feature, secure means that the CPCrE protocol is designed with resilience to a client-payment impersonation attack in mind. The client-payment impersonation attack is an attack in which an adversary successfully assumes the payment of an identity of a legitimate client to be identified using the CPCrE protocol. In other words, the AS should assure that the client who made the payment (i.e. who own a payment-attribute certificate) is the same client who claims the identification request (i.e. NA-IdenT Request). By considering the trust model for the client payment and identification, discussed in Section 6.2.1, the AS can be assured about the payment of the new client. The trust model includes the Payment Processors and Networks by which the payment process is achieved. Once the payment process completed successfully, the PS issues a client payment-attribute
certificate ($PCert_{Ci}$). $PCert_{Ci}$ is signed using PS’s private key (i.e. $PR_{PS}$). The issuance of $PCert_{Ci}$ is based on an authorised bank account. We consider that clients have authorised bank accounts when they open their bank account by making a physical appearance along with their photoID documentations, e.g. passports, and a set of other documents, such as client’s certificates or a license profile, and no client is interested to share the details of his bank account with others, unless someone who he/she trust. When the client sends a NA-IdenT Request, the AS verifies the signature of $PCert_{Ci}$ using the PS’s public key (i.e. $PU_{PS}$) which is certified using a trusted CA. The AS also validates the client identity (i.e.$ID_{Ci}$) contained in the $PCert_{Ci}$ against the one received in the NA-IdenT Request and signed by the client private key (i.e. $PR_{Ci}$).

Therefore, it is hard for an attacker or any curious client to impersonate a client-payment. This is because the attacker has not only to compromise $PCert_{Ci}$ which is signed using the PS’s private key, but also has to compromise the client’ private key, so satisfy (S5).

6.6.2. Formal Verification

In addition to the informal security analysis reported in the previous section, the CPCrE protocol is also formally verified.

6.6.2.1. Protocol Modelling using AVISPA

Prior to the formal verification of the CPCrE protocol by modelling the protocol using AVISPA Tool [106], this section gives (i) a brief discussion of the reasons of (i.e. the need of) using the formal verification and (ii) an overview of the used formal verification tool.

A formal verification of a security protocol can help to discover security flaws in a protocol design. The design of a security protocol is meant to provide a secure service, but the design of any security protocol, such as the CPCrE protocol, is particularly an error prone process. If the CPCrE protocol is not deigned properly, some of the security flaws could be subtle and difficult to find [77], and the protocol may fail to provide a secure service. Thus, it is important to formally verify any new security protocol designed in order to detect any subtle flaws that may not be discovered (i.e. realised) by the informal security analysis. A well-known public key protocol [107], designed by Needham-Schroeder and
called Needham-Schroeder’s protocol, is a good example (along with other protocols cited in [78]) presenting the importance of the formal verification of a security protocol. Needham-Schroeder’s protocol was believed to be secure for 17 years before it was formally verified [78] by a number of formal verification tools such as AVISPA. Automatic verification tools (e.g. AVISPA) can therefore be very helpful to obtain actual guarantees that a new security protocol designed is correct and safe [108].

Four reasons why the AVISPA tool is used; (i) modelling security protocols using AVISPA has reached a fairly mature state [108], (ii) AVISPA provides a modular, expressive and versatile formal language, a role-based High Level Protocol Specifications Language (HLPSL), for specifying protocols and their security properties, (iii) AVISPA has been designed to be easily usable by IT engineers, professionals, and protocol designers working in research, industry or standardization organisations [109], and (iv) AVISPA is widely used to formally verify a large collection of practically relevant, authentication protocols, being drafted, proposed, and even standardized by organisations like the IETF and ITU [109]-[116].

The AVISPA Tool is structured in two levels of languages, high and low level of languages. The high level of language is HLPSL and the low level of language is the Intermediate Format (IF) [117] [118]. As shown in Figure 6.8, a HLPSL-Specification is translated into the IF, using ‘hlpsl2if’ translator. The IF is read directly by the back-ends to the AVISPA Tool. The IF specification of a modelled protocol is then input to the back-ends of the AVISPA Tool to analyse if the specified goals (i.e. security properties) are satisfied or violated. AVISPA Tool uses two common verification tools called On-the-Fly Model-Checker (OFMC) [119] and Constraint-Logic-based Attack Searcher (CL-ATSE) [120] that implement variety of automatic analysis techniques.

![AVISPA Tool architecture](image)

**Figure 6.8 AVISPA Tool architecture**
Using AVISPA, the CPCrE protocol is modelled as following.

HLPSL is a role-based language, so first, the sequences of actions of our protocol entities, i.e. the Authentication Server and the Client, are specified in a module. The module is called a basic role. This specification can later be instantiated by one or more agents playing the given role. Then, how the entities interact with each other is specified by conjoining multiple basic roles together into one role called composed role. The composed role describes sessions of the protocol, i.e. defines all the channels used by the basic roles. Further, before the security properties of the CPCrE protocol are declared using a goal statement, a top-level role called an environment role is defined in the HLPSL-Specification. The environment role has global constants along with a composition of one or more sessions (i.e. composed roles) where an intruder may play a role as a legitimate agent and his initial knowledge is described. When modelling the CPCrE protocol, it can be a good idea to start with a diagram of the flow of exchanged messages between the protocol entities in A&B notation as shown in Figure 6.9. Then we continue with the specification of each basic role.

![Figure 6.9 A&B notation for the CPCrE protocol](image)

In the CPCrE protocol, as shown in the figure above, there are two basic roles, called gateAuthNService and clientAuthNAgent. Note that the role name begins with lowercase letters as it is required by the HLPSL’s syntax. These names, i.e. gateAuthNService and clientAuthNAgent, are used to denote the roles themselves, while the names of these agents playing the roles are called AS and C, respectively. Each basic role defines what parameters (i.e. information) the protocol entity can use initially in its initial state, and ways in which the state can change (transitions). Figure 6.9 already shows the HLPSL-Specification of the CPCrE protocol, the sequences of actions part. The following are the initial parts of the two roles, clientAuthNAgent and gateAuthNService:
role clientAuthNAgent { C, AS : agent, 
PUC, PUas : public_key, 
Hash : function, 
SND, RCV : channel(dy)}

played_by C def=

local State : nat, 
IPch, Emailu, CVpc, PCertc : text, 
IdIc : text.text.text.text.text, 
SKc_sp : symmetric_key, 
Tc, Tas : text, 
R1as, T1asg, R2as, T2asg : text, 
IssueDate : text, 
ValidTo, ValidFrom : text, 
SigNAreq, SigNAres : message

init State := 0 
transition ....

role gateAuthNService { C, AS : agent, 
PUC, PUas : public_key, 
Hash : function, 
SND, RCV : channel(dy)}

played_by AS def=

local State : nat, 
IPch, Emailu, CVpc, PCertc : text, 
Tc, Tas : text, 
R1as, T1asg, R2as, T2asg : text, 
SKc_sp : symmetric_key, 
IssueDate : text, 
ValidTo, ValidFrom : text, 
IdIc : text.text.text.text, 
SKc_dl, Kc, Pc, Lc : message, 
PT2c :

message.message.message.message.message.text.text.text.text.text, 
SigNAreq, SigNAres : message

init State := 1 
transition ....

All parameters, i.e. the variables and constants, are typed in HLPSL. All variables begin 
with capital letters, and all constants begin with small letters. For instance, the role 
clientAuthNAgent in the CPCrE protocol, has a number of parameters. C and AS are of type 
agent, PUC and PUas are of type public key, and Hash is of type function (i.e. hash function). 
The SND and RCV parameters are of type channel, indicating that these are channels 
through which clientAuthNAgent playing the role client will communicate to the other 
agent. The attribute dy to the channel type, for Dolev-Yao, represents the intruder model 
assumed for this channel. Note that the notations which are used to represent the typed 
parameters and variables in HLPSL to model the CPCrE protocol, are slightly different from 
those notations, listed in Section 5.6.4 and in Section 6.2.4. For example, $PT_{Ci}^2$, is 
represented as PT2c in the HLPSL to fulfil the syntax and semantic conditions of the HLPSL 
language.
The parameter \( C \), appears in the section played_by, means that \( C \) denotes the name of the agent which plays the client role. The local section defines local variables of clientAuthNAgent, such as State, which is a natural number and initialized to zero in the init section.

The transition section of an HLPSL-Specification, in a basic role such as clientAuthNAgent and gateAuthNService, contains a set of transitions. Generally, each transition represents the sending of a request and the receipt of a response (an example of a transition in the role of clientAuthNAgent), or the receipt of a request and the sending of a response (an example of a transition in the role of gateAuthNService). A transition of any basic role consists of a trigger (i.e., precondition) and an action. When a trigger event occurs, an action or actions is/are performed. For instance, the role of gateAuthNService in our CPCrE protocol contains the following transition (where some details have been omitted for simplicity):

```
transition
Step 1.
State = 1 /
\[ RCV\{IPch'.Emailu'.CVPc'.PCertc'\}_SKc_sp'.PUas.C.Tc'._{Hash\{IdIc'_SKc_sp'.PUas.C.Tc'}}_inv(PUC) \Rightarrow \]
State':= 3 ...... /
\[ R1as':= new() / T1asg':= new() / R2as':= new() / T2asg' :=
new() / Tas':= new() / SKc_dl':= Hash(Rlas'.Tlasg') ...... / \]
\[ SigNAres' :=
\{Hash\{Hash(Rlas'.Tlasg').Hash(SKc_sp'.inv(PUas)).Hash(C.R2as'.T2asg').Hash(C.IPch'.Tas').IssueDate'.ValidFrom'.ValidTo'\}_SKc_sp'.AS.Tas')_inv(PUas)
\}
\] /
\[ SND\{Hash(Rlas'.Tlasg').Hash(SKc_sp'.inv(PUas)).Hash(C.R2as'.T2asg').Hash(C.IPch'.Tas').IssueDate'.ValidFrom'.ValidTo'\}_SKc_sp'.AS.Tas'.SigNAres'
\] ...
```

The above transition is called step 1. It specifies that if the State value is equal to 1 and the received message on channel RCV has the expected structure defined above (between the two brackets of RCV (...)), this transition then fires (all actions comes after ‘=|>’ notation). It sets the State variable to a new value (equal to 3) and generates both (i) a set of secret keys and values of items (e.g., dynamic login-secret key, SKc_dl') and (ii) the response message signature SigNAres, as well as it sends the items on channel SND. Here we see an example of priming; State'. It means the new value of the variable State. The value of variable will be changed after the current transition is completed. So, the right-hand side of ‘:=’ notation tells that the value of the State variable, after transition step1 fires, will be 3.

In the RCV channel, there is another example of using the primed variable. In this case, the primed variable is bound to whatever is received. Also, the structure of the message
which is expected to be received is specified within RCV. In the transition above, for example it is expected that the first part of the message, which is between the first two curly brackets, is encrypted with the key SKc_sp (the session password-secret key). The fact is that the variables of this part of the message and this key itself can be arbitrary. Whatever are in there, they will be bound to variables in the next step. This is because they are prime. However, the variable of the encryption key of the second part of the message, i.e. the AS’s public key, PUas, is not primed. It indicates that the received message must have the same value as the current value of the variable. The other transitions of the clientAuthNAgent are specified similarly. This is how one may model the way in which the information available to a role may change.

Both basic roles are then composed together in a composed role. The composed role has no transition section, but rather it instantiates the two basic roles. The composed role also glues the two basic roles together so they execute together, usually in parallel by means of the operator ‘/\’. The following composed role which instantiates one instance of each basic role (i.e. for clientAuthNAgent and gateAuthNService) and thus describes one whole protocol session. So for convention, a composed role is called a session role.

```
role session ( C, AS : agent,
             PUc, PUas : public_key,
             Hash : function)
def=
    local  SNDC, RCVc, SNDas, RCVas : channel (dy)

composition
clientAuthNAgent(C,AS,PUc,PUas,Hash,SNDC,RCVc)/
gateAuthNService(C,AS,PUc,PUas,Hash,SNDas,RCVAS)
end role
```

In the session role, usually all variables that are used to instantiate the clientAuthNAgent and gateAuthNService roles are declared. This includes all the channels used by these roles. As mentioned before in the basic role, the channel type takes an additional attribute called Dolev-Yao intruder model, i.e. dy. In this model, the intruder has full control over the network, so any message sent by C or AS (i.e. agents) goes also to the intruder. The intruder may capture, analyse, and/or alter messages. The intruder may also send any message he comprises to whoever he wants, posing as an agent as explained in the following final top-level role, i.e. the environment role, in modelling the CPCrE protocol.

The environment role is the last role to be defined in the HLPSL-Specification. It contains global constants and composition of two sessions. Also, in this role, the intruder
plays a role as a legitimate agent, and a statement describes the initial knowledge of the
intruder is instantiated. Typically, this includes the names of all agents, i.e. the Client (c)
and the AS (as), all public keys, i.e. C’s public key (puc), AS’s public key (puas), and
Intruder’s public key (pui), the intruder’s own private key (inv(pui)), and the hash function h.

role environment ()
def=
    const c, as, i : agent,
puc, puas, pui : public_key,
h : function,
sec_SKc_sp : protocol_id,
sec_IdIc : protocol_id,
sec_SKc_dI, sec_PT2c : protocol_id,
signareg_c2as, signares_as2c : protocol_id
intruder_knowledge = {c, as, puc, puas, pui, inv(pui)}
composition
    session(c, as, puc, puas, h) / \session(c, i, puc, pui, h)
end role

6.6.2.2. Formal Verification Results and Discussion

As mentioned in the previous section, parameters of each basic role define what
information that the role begins with, and are passed in as arguments from the session role.
In the CPCrE protocol modelling, the session role is used to describe the execution of the
protocol. The session role composes two roles together and defines what information each
role begins with by passing the information in each role as arguments as following:

clientAuthNAgent(C, AS, PUc, PUas, Hash, SNDc, RCVC) / \gateAuthNService(C, AS, PUc, PUas, Hash, SNDAS, RCVAS)

while environment role describes two concurrent sessions as following:

    session(c, as, puc, puas, h) / \session(c, i, puc, pui, h)

The first session in the environment role is a typical session with the legitimate agents,
C and AS. Note that all of the arguments are in lower-case within the environment role. This
is because they are constants rather than variables. The second session is a typical session
by which the intruder has full control over the network. From the arguments it can be seen
that the intruder is playing the role of a legitimate agent in order to attack the CPCrE
protocol.
As shown in Figure 6.10, when the CPCrE protocol is modelled using the AVISPA Tool, the validation results of the tool show that the protocol is correct and safe, no attacks are found.

![AVISPA validation results](image)

The above validation results of the CPCrE protocol modelling ensure two security properties; the secrecy of the identification and authentication information and the mutual authentication between the Client (C) and the Authentication Server (AS). These security properties are modelled by augmenting the transitions of the basic roles with what are called goal facts. Goal facts, are special events, express a combination of facts and conditions that specify what are desirable and what are considered as an attack (details to follow). This could be achieved with the respect to the protocol goals which are declared in the goal section as following:

```plaintext
goal
  authentication_on_signareg_c2as
  authentication_on_signares_as2c
  secrecy_of_sec_SKc_sp
  secrecy_of_sec_IdIc
  secrecy_of_sec_SKc_dl
  secrecy_of_sec_PT2c
end goal
```

The first security property is to ensure the confidentiality of the contents of the protocol messages. In other words, both (i) the contents of the NA-IdenT Request message which represents the client identification information, i.e. IdIc, and the session password-secret key, i.e. SKc_sp, and (ii) the contents of the NA-IdenT Response message, which represents the dynamic login-secret key, i.e. SKc_dl, and the 2nd primary authentication token of the MR-Client, i.e. PT2c, are kept secret among both entities, C and AS. As mentioned before, this is modelled by adding goal facts, called secret events, in the transition of the basic roles. Secret events assert which values should be kept secret, and which agents are allowed to know such secrets. As a modelling rule, it is recommended to place these goal facts (i.e.
secret events) in the role that creates the value which is meant to be secret. In the CPCrE
protocol, clientAuthNAgent creates both the IdIc and SKc_sp and the gateAuthNService
creates both SKc_dl and PT2c. These events (i.e. goal facts of IdIc, SKc_sp, SKc_dl2 and
PT2c) are augmented in the last transition where they have been created in each respective
role as the following:

For clientAuthNAgent role:

:\ secret (SKc_sp',sec_SKc_sp,{C,AS})
:\ secret (IdIc',sec_IdIc,{C,AS})

For gateAuthNService role:

:\ secret (SKc_dl',sec_SKc_dl,{C,AS})
:\ secret (PT2c',sec_PT2c,{C,AS})

These goal facts state that (i.e. read as:) any values appearing in the first arguments
of these secret events (i.e. goal facts), i.e. IdIc, SKc_sp, SKc_dl2 and PT2c, are considered to
be secret. If the intruder learns any of these values and the intruder is not in the set given in
the third argument of the associated goal fact, then this represents an attack. In other words,
if the intruder is not set in between the two curly brackets {}, i.e. in \{C,AS\} in any of the
formulas above, then this represents an attack. This is achieved with the respect to the
second arguments of the goal facts, i.e. sec_SKc_sp, sec_IdIc, sec_SKc_dl and sec_PT2c
that are specified in the goal section and declared as protocol ids in the environment role.

The second security property is the mutual authentication. The mutual authentication is
to ensure that the communicating entity is the one that it claims to be. This is modelled by
adding another two goal facts, related to authentication, to the HLPSL-Specification. They
are called witness and wrequest events. Witness and wrequest events are used to verify that
an agent is right in believing that its communicated peer is exist in the current session and
agrees on a certain value, which typically is fresh and not tampered. Both events are used,
in the CPCrE protocol modelling, to ensure that an agent (either C or AS) agrees on a
particular value, which is typically fresh and not tampered, with its intended peer presented
in the communication session. Witness and wrequest always appear in pairs with identical
third parameter as explained below.

In the CPCrE protocol modelling, the two agents, C and AS, should certainly agree on
the values of the exchanged messages’ signatures, i.e. SigNAreq and SigNAres. In
particular, when AS wishes to be sure that the signature of the NA-IdenT Request message
(i.e. SigNAreq) was indeed created by the client (i.e. C) that it was created for the purpose of being originated by C and that it was not tampered or replayed from a previous session. To achieve this, the following goal fact is added in the last transition of the gateAuthNService

\[ \text{wrequest}(\text{AS}, C, \text{signareq}_c2as, \text{SigNAreq}') \]

This goal fact is read as follows: agent AS accepts the value \( \text{SigNAreq}' \) and now relies on the guarantee that agent C exists and agrees with her on this value. Moreover, the third argument \( \text{signareq}_c2as \) is used for distinguishing different authentication pairs, as there is another authentication request in the CPCrE modelling which authenticates the AS to the C (as can be seen in the goal section of the HLPSL-Specification). It is for declaring with what purpose the value is being interpreted. As the CPCrE protocol modelling convention, the third argument is typically the names of the authenticating agent (AS), the agent to be authenticated (C), and the name of the variable being checked (signareq ), all in lower case, concatenated together. This argument is declared as a constant of type protocol_id in the environment role.

The other matching goal fact, i.e. witness, is added in role clientAuthNAgent as part of the transition in which the value signareq is sent to the AS.

\[ \text{witness}(C, \text{AS}, \text{signareq}_c2as, \text{SigNAreq}') \]

The above goal fact is read as: agent C asserts that it wants to be the peer of agent AS, agreeing on \( \text{SigNAreq}' \) as an authentication value identified by the protocol_id signareq_c2as. To indicate that the wrequest and witness events containing such protocol_ids, signareq_c2as and signares_as2c, for the authentication, they are modelled in the goal section of the protocol as following:

\[ \text{authentication_on signares_as2c} \\
\text{authentication_on signareq}_c2as \]

The validation result of modelling CPCrE protocol using these goal facts; secret, wrequest, and witness events, indicates the following.

- The confidentiality of the four secret values, i.e. IdIc, SKc_sp, SKc_dl2 and PT2c, is protected. In other words, these secret values are known only for C and AS. This is because the AS is the only entity can decrypt the encrypted session password-secret key,
i.e. SKc_sp which is generated by the C and used to encrypt the other secret values (IdIc, SKc_dl2 and PT2c), so only the C and AS can use SKc_sp to decrypt these secret values.

- The AS authenticates the C on the fresh value of signareg which is received from clientAuthNAgent. This is because the clientAuthNAgent is the only entity can generate this value using the corresponding private key (i.e. C’s private key, inv(PUc)).
- The C authenticates the AS on the fresh value of signares which is received from gateAuthNService. This is because the gateAuthNService is the only entity can generate this value using the corresponding private key (i.e. AS’s private key, inv(PUas)).

### 6.7. Performance Evaluation

This section evaluates the performance of the CPCrE protocol. The evaluation is carried out in terms of computational and communication costs and the protocol execution time. Two experiments are carried out, one for measuring the computational and communication costs, and the other is for measuring the protocol execution time. The CPCrE protocol is a building block of the VDAF authentication method. It is more complex than the existing ones. This is due to that in our method, to identify a client to the MR application using multi-factor authentication secrets, two primary authentication tokens with two secret keys each are used. The primary authentication tokens are used to form the local and removable parts of the MR-Client authentication credential that are stored in two separate storages. As explained in Section 6.4, the issuance and the protection of the authentication secrets, using CPCrE protocol, involve a complex key structure. This will undoubtedly introduce additional costs. The performance evaluation presented in this chapter is to evaluate the cost by using such approach.

#### 6.7.1. Computational Cost

This section evaluates the computational cost introduced by the MR-AuthN Components of the CPCrE protocol. The evaluation is carried out using two methods, theoretical analysis and experimental evaluation. In the theoretical analysis, the types and the number of the cryptographic operations performed by the C-AuthN Agent and the G-AuthN Service are classified and compared with those of the related protocols used in the existing MR client identification methods. In the experimental evaluation, the processing times experienced by the C-AuthN Agent and G-AuthN Service, are measured.
6.7.1.1. Theoretical Evaluation

Cryptographic algorithms used in the design of the CPCrE protocol can be classified based on their operations into two categories, computationally inexpensive ones, i.e. symmetric encryption/decryption, hash and random-number-generator algorithms, all of which are denoted as ‘Inexp’, and computationally expensive ones, asymmetric-encryption/decryption and digital signature generation and verification algorithms, denoted as ‘Exp’. A summary of the costs by the CPCrE protocol and the related client identification methods is given in Table 6.4. As described in Section 6.5, when executing the CPCrE protocol, the C-AuthN Agent and the G-AuthN Service perform the following operations:

- **In Step1 - Define a Client Identification Information:** The C-AuthN Agent performs three Inexp operations (one random-number generator operation and one hash operation to generate the session password-secret key, and one symmetric encryption operation to protect the confidentiality of the client identification information), and two Exp operations (one asymmetric encryption operation to protect the confidentiality of the freshly generated session password-secret key, and one signature generation operation to protect the authenticity of the NA-IdenT Request message).

- **In Step2 - Send NA-IdenT Request:** The C-AuthN Agent does not perform any Exp and Inexp operations as this step is to send the NA-IdenT Request once it is constructed.

- **In Step3 - NA-Request Verification:** The G-AuthN Service performs three Exp operations (one asymmetric decryption operation to obtain the session password-secret key, two signature verification operations to verify the authenticity of the received NA-IdenT Request and the authenticity of the client payment-attribute certificate), and one Inexp operation (a symmetric decryption operation to obtain the client identification information).

- **In Step4 - Generate a Client Authentication Information:** The G-AuthN Service performs seven Inexp operations (two random-number generator operations and four hash operations to generate the following items: a dynamic login-secret key, a user password-secret key, a client master-secret key and a client location-secret code, which are used to form the primary tokens of the MR-Client, and one symmetric encryption operation to protect the confidentiality of the NA-IdenT Response message), and one Exp operation (a signature generation operation to protect the authenticity of the NA-IdenT Response message).
- In **Step 5 - Send a NA-IdenT Response**: The G-AuthN Service does not perform any Exp or Inexp operation as this step is to send the NA-IdenT Response once it is constructed.

- In **Step 6 - NA-Response Verification**: The C-AuthN Agent performs one Exp operation (a signature verification operation to verify the authenticity of the received NA-IdenT Response message).

- In **Step 7 - Prepare to MR-Client Authentication**: The C-AuthN Agent performs six Inexp operations to form the local and removable parts of the MR-Client authentication credential (one symmetric decryption to obtain the dynamic login-secret key and the second primary authentication token of the MR-Client, and three hash operations along with two symmetric encryption operations to construct and protect the confidentiality and integrity of the local and removable parts of the MR-Client authentication credential).

### Table 6.4 A comparison of the computational cost of the client identification

<table>
<thead>
<tr>
<th>Protocol/methods</th>
<th>Computational costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>The CPCrE protocol</td>
<td>(4T_H + 3T_{Enc} + 1T_{Dec} + 1T_{rng})</td>
</tr>
<tr>
<td>C-AuthN Agent</td>
<td>Inexp: (1T_{Enc} + 1T_{Dec} + 1T_{rng}), Exp: (1T_{AEnc} + 1T_{Sig} + 1T_{Ver})</td>
</tr>
<tr>
<td>G-AuthN Service</td>
<td>Inexp: (4T_H + 4T_{Enc} + 1T_{Dec} + 2T_{rng}), Exp: (1T_{Adec} + 1T_{Sig} + 2T_{Ver})</td>
</tr>
<tr>
<td>Malley method</td>
<td>Inexp: (1T_H)</td>
</tr>
<tr>
<td>Somu method</td>
<td>Inexp: (3T_{Enc})</td>
</tr>
<tr>
<td>Rubika method</td>
<td>Inexp: (3T_{rng})</td>
</tr>
<tr>
<td>Wei method</td>
<td>-</td>
</tr>
<tr>
<td>Ruan method</td>
<td>-</td>
</tr>
<tr>
<td>Zhao method</td>
<td>-</td>
</tr>
</tbody>
</table>

\(T = \text{Time}; \ H = \text{Hash}; \ Enc = \text{Symmetric encryption}; \ Dec = \text{Symmetric decryption}; \ Rng = \text{Random-number generator}; \ AEnc = \text{Asymmetric encryption}; \ ADec = \text{Asymmetric decryption}; \ Sig = \text{Signature generation}, \text{and Ver} = \text{Signature verification.}

### 6.7.1.2. Experimental Evaluation

This section reports the experimental evaluation of the computational cost introduced by a C-AuthN Agent and the G-AuthN service when executing our CPCrE protocol. In other
words, it reports the processing times experienced by the C-AuthN Agent and G-AuthN Service respectively when executing the protocol. As the issuance and protection of the MR-Client’s primary authentication and verification tokens are based on a complex key structure, and the computational cost introduced by each cryptographic operation could vary from one algorithm to another, the first experiment is conducted to measure and compare the processing times using two scenarios: in scenario-1, the AES algorithm with a 128-bit key is used, and in scenario-2, the 3DES algorithm with a 192-bit key is used. The experiment uses the following benchmark:

- Asymmetric key cipher used is RSA algorithm with a key of 2048 bits long.
- Hash functions used are SHA-256/512.
- Random-numbers generator used is SHA1-PRNG with output of 1024 bytes long.
- An identifier used for each entity in all the protocol is 15 bytes long.
- The Microsoft Active Directory Certificate Service (AD-CS) is used to issue and manage the standard private keys and the corresponding public key certificate (X.509 certificates).

The Microsoft AD-CS (using Windows server 2012) facilitates the deployment of the Public-Key Cryptography Standard (PKCS) of the private keys and their public key certificates. It is used for installing and configuring a trusted root Certificate Authority (CA). The root CA is used to issue and manage each private key and the respective public key certificate of the CA itself, AS, PS and an applied client.

- The Java Standard Edition (Java SE) in the NetBeans Integrated Development Environment (NetBeans IDE v.8) platform is used for developing the prototype.

Java offers software portability (i.e. write once and run anywhere), network awareness and security. Java is also used to develop the MR application itself. The NetBeans IDE v.8 platform uses Java Standard Edition (Java SE v. 8.91). Java SE version 8.91 is a stable version that includes Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE). JCA and JCE provide the implementations of several cryptographic primitives and key management services required for our CPCrE protocol implementation, these include:

  - A secure random number generator: It implements the function for generating non-deterministic numbers required to generate different secret keys, such as client master-secret key and session password secret-key, and different secret...
values of items that are involved in the issuance of the MR-Client’s primary authentication and verification tokens.

- **A message digest class:** It implements hash function using (1) SHA-512 function to generate a login authentication token and a local verification token (in the issuance process of the local and removable parts of the MR-Client authentication credential), and (2) SHA-256 function to generate a key material, in bytes, (in the generating process of a secret key).

- **Signature class:** It implements the digital signature scheme for signing and verifying the protocol messages using SHA-256 along with RSA algorithms.

- **Cipher class:** It implements both symmetric (3DES/AES) and asymmetric (RSA) cryptographic algorithms. For examples, RSA along with 3DES/AES are required in **Step1 -Define a Client Identification Information** before sending the NA-IdenT Request to encrypt the session password-secret key and the identification information respectively.

- **Key management facilities:** It includes (i) KeyStore for storing cryptographic keys and certificates, (ii) CertificateFactory for managing certificates in the KeyStore, and (iii) Secret Key Specs for (1) constructing secret keys from the key material generated by the message digest class and (2) specifying the secret key to be as a 3DES key or a AES key.

- A desktop computer running Microsoft Windows 8.1 version 64-bit based Operating System 2013 with Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz and 8GB of RAM, and an IBM server machine running Microsoft Windows Server 2012 Version 64-bit based Operating System with Intel(R) Xeon(R) E5440 CPU @2.8GHz and 16GB of RAM are used to implement the developed prototype.

To ensure statistical significance of the experimental data, a number of iterations (n) over which a processing time is measured is determined. This is to reduce the effect of arbitrariness on the processing time. The experiment was conducted by setting n into different values (starting from n = 1) while measuring the average of the processing times introduced by both the C-AuthN Component and the G-AuthN Service. Figure 6.11 shows the results of this experiment for the two scenarios. As can be seen from the figure, the bigger the value of n, the less effect the other initialization factors have on the computational cost. Also the trend of the graph shows (i) an n value higher than 3k (i.e. 3000 iterations) is adequate, and (ii) there is no noteworthy difference in the processing time of both scenarios, i.e. the difference in the processing time between the two scenarios, introduced by the same
MR-AthN Component to perform a particular step (e.g. Define a Client Identification Information), is negligible. As shown in the figure, under n=3500, the C-AthN Agent takes about 17 milliseconds to define the identification information and form the NA-IdenT Request before sending the request. The C-AthN Agent takes 7.3 milliseconds to verify the received NA-IdenT Response and form the local and removable parts of the MR-Clien authentication credential. In addition to this 7.3 milliseconds, the C-AthN Agent takes about 3.5 seconds to embed and save the encrypted local part of the MR-Clien authentication to a stego-object. In the other side, the G-AthN Service takes about 48.6 milliseconds to verify the NA-IdenT Request message and generate the client authentication information. Note that this cost (i.e. 48.6 milliseconds) includes the time needed by the G-AthN Service to form the NA-IdenT Response object before start sending it to the C-AthN Agent.

Figure 6.11 The computational cost for CPCrE protocol using AES and 3DES

6.7.2. Communication Cost

The cost of the communication overheads of the CPCrE protocol is evaluated in terms of the number and length of the exchanged messages. As can be seen from Figure 6.7, in each client identification performed by our CPCrE protocol, the client identification and authentication information are exchanged between a C-AthN Agent and the G-AthN Service. Once the client defines his identification information, the client sends a NA-IdenT Request to the AS. The AS verifies these identification information received in the request
and sends a NA-IdenT Response contains $PT_{C_i}^{2}$ to the client. Thus in this section we measure the communication overheads (in bytes) experienced by the protocol. The cost of the communication overhead is measured using the same benchmark of the first experiment discussed above. Figure 6.12 shows the CPCrE protocol messages exchanged and their respective lengths in the two scenarios. The figure shows that the lengths of the NA-IdenT Request are mostly the same, whereas there is a slight difference in the lengths of the NA-IdenT Response. This is because the size of the key used in the 3DES algorithm (192bit) is a slightly longer than the key used in the AES algorithm (128bit). These keys, including the user password-secret key and the client master-secret, are the main contents of the second primary authentication token ($PT_{C_i}^{2}$) which is carried in the NA-IdenT Response message.

![Figure 6.12 Comparison between the CPCrE protocol messages using AES and 3DES](image)

6.7.3. Protocol Execution Time (PET)

A protocol execution time, PET, is defined as the time elapsed between the time when the execution of a protocol is invoked and the time when the execution finishes. We investigate the PET experienced by the CPCrE protocol. For doing so, this section first discusses a theoretical model that is constructed and used to calculate the theoretical values of the PET. Then it reports the experimental evaluation of the PET.

6.7.3.1. Theoretical Model

The PET of a protocol is the total delay experienced by the protocol. It is the sum of the delays of all the rounds of communications taking place during the protocol execution. It can be expressed by the following equation:
\[ \text{PET} = \sum_{r=1}^{r=k} (\text{Delay}_{\text{COMR}, r}) \quad \text{(EQ 6.1)} \]

where, \( \text{Delay}_{\text{COMR}, r} \) is the delay experienced by communication round \( r \) (i.e. \( \text{COMR}_r \)), and \( \sum_{r=1}^{r=k} (\cdot) \) is the summation of the delays for \( k \) communication rounds.

In detail, \( \text{Delay}_{\text{COMR}, r} \) consists of two components. The first is the delays caused by the two MR-AuthN Components (\( \text{MRC} \)) of a communication round \( r \), i.e. experienced by the protocol messages in the two MR-AuthN Components. The second one is the delay caused by the network (\( \text{NET} \)) that connects these two MR-AuthN Components, i.e. experienced by the protocol messages in transit in the network.

\[ \text{Delay}_{\text{COMR}, r} = \text{Delay}_{\text{MRC}, r} + \text{Delay}_{\text{NET}, r} \quad \text{(EQ 6.2)} \]

Figure 6.13 illustrates these two delays which could be experienced in the communication rounds of the CPCrE protocol. As can be seen from the figure, the CPCrE protocol only consists of one communication round, i.e. \( k = 1 \), so the PET of the CPCrE protocol is:

\[ \text{PET} = \sum_{r=1}^{r=1} (\text{Delay}_{\text{COMR}, r}) = \text{Delay}_{\text{COMR}1} \]

![Diagram of Communication Rounds](image)

**Figure 6.13 The theoretical value of PET for the CPCrE protocol**

\( \text{Delay}_{\text{MRC}, r} \) has further two components, one for each protocol message. One is for a NA-IdenT Request (\( \text{Req} \)) and the second one is for a NA-IdenT Response (\( \text{Res} \)). Each component is a function of three delays that a \( \text{Req} \) or a \( \text{Res} \) experienced in \( \text{MRC}_r \). The three delays are a queuing delay (\( \text{QueuD} \)), processing delay (\( \text{ProceD} \)), and transmission delay (\( \text{TranD} \)). \( \text{Delay}_{\text{MRC}, r} \) cab be calculated using the following equation.
Delay\(_{MRC_r}\) is similar to Delay\(_{NET_r}\) in that Delay\(_{NET_r}\) has two components one for each protocol message, i.e. Req or Res. However, Delay\(_{NET_r}\) is a function of four delays that a Req or a Res experienced in NET\(_r\), i.e. experienced at each intermediate node and the network media connecting the nodes en route from the source to the destination node. The four delays are queuing delay (\(QueuD\)), processing delay (\(ProcD\)), transmission delay (\(TranD\)), and propagation delay (\(PropD\)). Assuming there are \(n\) intermediate nodes connecting the two MR-AuthN Components. Delay\(_{NET_r}\) is calculated as follows:

\[
\text{Delay}_{NET_r} = \left\{ \sum_{i=1}^{n} \left( QueuD_{i}^{Req} + ProcD_{i}^{Req} + TranD_{i}^{Req} + PropD_{1}^{Req} \right) \right\} + \left\{ \sum_{i=1}^{n} \left( QueuD_{i}^{Res} + ProcD_{i}^{Res} + TranD_{i}^{Res} + PropD_{2}^{Res} \right) \right\} \\
\text{-------- (EQ 6.4)}
\]

where \(PropD_{1}^{Req}\) is the propagation delay experienced in a network media (NM) connecting the first MR-AuthN Component (e.g. C-AuthN Agent in our CPCrE protocol) that initiates the Req and the first intermediate node that receives the Req, while \(PropD_{2}^{Res}\) is the propagation delay experienced in a NM connecting the second MR-AuthN Component (e.g. G-AuthN Service) that initiates the Res and the first intermediate node that receives the Res.

Figure 6.14 illustrates these delays and shows how the total delay in the one round of the CPCrE protocol message exchange is calculated. In the figure the Req and Res of the CPCrE protocol, i.e. the NA-IdenT Request and Response are respectively denoted as NAResponse and NAResponse.
The following definitions explain how the values of the components delays are calculated.

**Transmission Delay** \( (\text{TranD}_{\text{Req/Res}}) \)

\( \text{TranD}_{\text{Req/Res}} \) is the time taken to transmit a \( \text{Req} \) or a \( \text{Res} \). The equation for calculating \( \text{TranDelay}_{\text{Req/Res}} \) is as follows:

\[
\text{TranD}_{\text{Req/Res}} = \frac{CO_{\text{Req/Res}}}{DTR} \quad \text{(EQ 6.5)}
\]

where, \( CO \) (communication overhead) refers to the length of a \( \text{Req} \) or a \( \text{Res} \) in byte. \( DTR \) is the data transfer rate in bytes per second.

**Propagation Delay** \( (\text{PropD}_{\text{Req/Res}}) \)

\( \text{PropD}_{\text{Req/Res}} \) is the time taken for a \( \text{Req} \) or a \( \text{Res} \) to propagate from one node to another. The following equation calculates \( \text{PropD}_{\text{Req/Res}} \).

\[
\text{PropD}_{\text{Req/Res}} = \frac{D}{\text{Propagation speed}} \quad \text{(EQ 6.6)}
\]

where, \( D \) is the distance in meters between two directly connected nodes. \( \text{Propagation speed} \) usually is taken as the speed of light which equals to 300,000,000 meters per second.

**Queuing Delay** \( (\text{QueuD}_{\text{Req/Res}}) \)

A message, i.e. a \( \text{Req} \) or \( \text{Res} \), may experience a queuing delay at each node. Each queuing delay consists of a message arrival-queuing delay \( (A\text{QueuD}_{\text{Req/Res}}) / \)
AQueduD_{i}^{Req/Res}\) and a message transmission-queuing delay \((TQueduD_{c}^{Req/Res}/TQueduD_{i}^{Req/Res})\). The message arrival-queuing delay is the time period in which a received message spends in the arrival-queue before being processed. The message transmission-queuing delay is the time period in which a processed message waits in the transmission-queue for transmission.

**Processing Delay** \((Proc_{c}^{Req/Res})\)

\(Proc_{c}^{Req/Res}\) is the time taken by (i) an MR-AuthN Component to construct an outgoing Req or Res or to verify an in-coming Req or Res, or by (ii) an intermediate node to process a received Req or Res to forward or direct it to the next node in an communication round \(r\).

### 6.7.3.2. Experimental Evaluation

This section reports the experimental results of the PET for the CPCrE protocol. This experiment was carried out using the Riverbed Modeler simulation tool [121].

**Protocol and Network modelling**

The Riverbed Modeler simulation tool version 17.5 is chosen because of its advanced simulation capabilities [122]. It has a large library of utility objects, used for modelling the CPCrE protocol, including (i) configuration-utility objects, such as Application Config-Utility Object, Task Config-Utility Object, Profile Config-Utility Object, and (ii) network-utility objects, such as Network Nodes and Network Media Objects. The protocol is defined using an Application Config-Utility Object, through a single task object. Each task is further divided into multiple phases. These phases represent the protocol steps which are defined using Task Config-Utility Object. A client profile is defined using a Profile Config-Utility Object which specifies how the protocol is executed by the client. Figure 6.15, shows a summary of how the task of the CPCrE protocol and its phases are defined and configured in the Application Config-Utility Object (indicated in the figure as an Application Definition) and the Task Config-Utility Object (indicated in the figure as a Task Definition), respectively, and how it is linked to the Client profile using Profile Config-Utility Object (indicated in the figure as a Profile Definition). The process of simulating a protocol is fairly complex. So, prior to configuring the configuration-utility objects, it is prudent to carefully
examine and specify all the network-utility objects for the protocol. Modelling the protocol using different network nodes or network media may provide different PET.

![Diagram of network model](image)

**Figure 6.15 Modelling the CPCrE protocol using Riverbed Modeler simulation tool**

As explained in the previous section, a network model describing the number of the intermediate nodes and the type of the network media linking the nodes between the C-AuthN Agent and the G-AuthN Service is a major factor affecting the PET value of the CPCrE protocol. Therefore, a network model is constructed and used to simulate the CPCrE protocol and measure its PET. As depicted in Figure 6.16, the network model consists of two Local Area Networks (LANs) that are connected by the Internet. The first one represents the Client’s site. It hosts a Client which runs the C-AuthN Agent, and a Layer 3 Switch which runs as a network switch and router. The Layer 3 Switch is connected to the Client using a Gigabit Ethernet cable and to the Internet using a T1 cable. The second one
represents the network of the MR application site. It hosts an Authentication Server which runs the G-AuthN Service and a Layer 3 Switch which runs as a network switch and router. The Authentication Server is connected to the Layer 3 Switch using a Gigabit Ethernet Cable and the Layer 3 Switch connects to the Internet using a T1 cable.

![Network model for CPCrE protocol](image)

**Simulation Model Validation**

The network model designed above is validated so that the collected results are considered reliable. The validation is done by comparing the PET result from the network model against the PET result obtained from the theoretical model under the same set of conditions. The network model, depicted in Figure 6.16, is run under the following conditions:

- The network does not have any background traffic, i.e. the only traffic in the network is the messages of the CPCrE protocol. By assuming a `zero' level of background traffic, queuing delays can be omitted, so the queuing delay is set to zero when calculating the theoretical results.
- All the entities are in fixed locations. The Client and the Authentication Server are each 1m away from their respective Layer 3 Switch using a Gigabit Ethernet cable, and the Layer 3 Switch is 10Km away from the Internet using a T1 cable.
- The propagation speed of the Gigabit Ethernet and T1 cable is set to 300,000,000 meters per second.
- The bandwidth of each Gigabit Ethernet cable is set to 1Gbps, and that of T1 cable is set to 1.544Mbps.
- The amount of processing time taken by the C-AuthN Agent and the G-AuthN Service is based on the processing time calculated in the first experiment. As there is a very little difference in the processing time of the two scenarios of the first experiment, in this experiment, the processing times are set to 17 and 7.3 milliseconds for C-AuthN Agent and 48.6 milliseconds for G-AuthN Service.
- The length of each protocol message is set to the length which is measured in the first experiment, i.e. 1351 bytes for NA-IdenT Request and 599 bytes for NA-IdenT Response.

The PET value from the theoretical model is compared with the result obtained from the network model. The comparison is shown in Figure 6.17. As can be seen from the figure the result from the theoretical model successfully validates the one from the network model, as the difference (i.e. the gab) between the theoretical and experimental results is only about 3%. This gap could be due to the processing delay taken by the C-AuthN Agent and the G-AuthN Service to encapsulate the request and response messages before a message be transmitted into the network media. These results provide confidence in the network model; hence any further results collected from the network model are considered reliable.

In the network model designed before it is assumed that the only traffic in the network is the messages of our CPCrE protocol. In a real network, there are background traffics. The background traffics may be from other services run on the network and/or other clients using the same service. In practice, the G-AuthN Service may receive identification requests simultaneously from other clients. So the PET for the CPCrE protocol is evaluated with a background traffic applied in the network model and examined versus a number of concurrent requests served by the G-AuthN Service. The number of concurrent requests
varied between one and one hundred. Figure 6.18 shows the results. It can be seen from the figure that the PET increases steadily as the number of requests received simultaneously increases. The more concurrent requests the G-AuthN Service receives, the longer it takes to process these requests. For example, when the G-AuthN Service receives only 10 concurrent requests, the PET is about a half second. When the requests received increases to 100 concurrent requests, the response time increases to about 3 seconds. The increase in the PET of 2.5 seconds which is six times is not as large as the increase in the number of concurrent requests which is about ten times, i.e. 900%, from $n = 10$ to $n = 100$. This is due to that these concurrent requests are processed by the G-AuthN Service in parallel.

Obviously, if the concurrent request messages are processed sequentially by the G-AuthN Service, then the PET will be high. In practice, this may take place when the arrival rate of the request messages is higher than the rate the G-AuthN Service takes to process received requests. In other words, the arrival-queue begins to build up when the messages arriving rate is faster than the messages processing rate. As a result, clients may experience a higher level of delay. The higher the request messages arrival rate, the longer a Client would have to wait before his/her identification request is processed, i.e. the longer the PET. This raise a question that when the PET value become longer? In other words, what is the turning point (i.e. threshold) beyond which the arrival-queue starts to build up? The threshold is the point beyond which the CPCrE protocol is expected to perform slowly (i.e. have a longer PET). In such case, there may be a need to add more Authentication Servers to process the requests received.

![CPCrE Protocol](image)

**Figure 6.18 PET for different concurrent requests**
To study and find out the turning point, it is assumed that the G-AuthN Service uses a first-come-first-served queue and it process one request message at a time. In other words, the G-AuthN Service processes all request messages sequentially. The result of such study is plotted in Figure 6.19. The figure shows time delays experienced by the G-AuthN Service against different messages arrival rates. As can be seen from the figure, the turning point beyond which the arrival-queue starts to build up is around the arrival rate of 21 NA-IdenT Request message per second. The queuing time increases steadily when the arrival rate goes beyond this point. This is because, from this point onwards, the average interval between the request arrivals is shorter than the average request processing time. In other words, when the arrival rate is 21 message/second, the average interval between two request messages is about 47.62 milliseconds, which is shorter than the average processing time (48.6 milliseconds to process each NA-IdenT Request). To reduce the request arrival-queuing delay, more Authentication Servers should be used.

![Figure 6.19 Turning point for the Authentication Server](image)

**Figure 6.19 Turning point for the Authentication Server**

### 6.8. Chapter Summary

This chapter has presented a novel protocol, the CPCrE protocol, for MR-Client Primary Credential Establishment. The protocol makes use of symmetric and asymmetric key cryptosystems and it allows a client to be remotely identified as an MR-Client to the MR application securely. Upon a successful execution of the protocol the client becomes an MR-Client and can obtain two primary authentication tokens. These tokens are then used to form the local and removable parts of the authentication credential for the MR-Client. In these parts, a key structure is designed enabling a strong MR-Client authentication. With this key structure, the primary authentication tokens are confidentiality and integrity protected and
these tokens enable multi-factor authentication. Also the three items required to identify an MR-Client authentication request must be available at the time the request is being made. These are (1) the user’s password (something the user knows), (2) the user’s machine on which one part of the MR-Client authentication credential is encrypted and embedded in a stego-object, and (3) the user’s removable storage media where the other part of the MR-Client’s authentication credential is encrypted and stored (something the user possess).

The chapter has presented the evaluation of the protocol both in terms of security and performance. The security evaluation is carried out using informal analysis and formal verification. The security analysis of the protocol demonstrates that the CPCrE protocol satisfies the security requirements specified. The formal verification, using the AVISPA tool, shows that the protocol is correct and safe. The performance of the protocol is evaluated in terms of the computational and communication overhead costs and the protocol execution time (PET). Both theoretical and experimental evaluations were carried out. Our protocol introduces more computation overhead in comparison to the existing work. This is because our protocol uses a complex key structure to support two-factor mutual authentication and involves more cryptographic operations to identify securely a remote client to the MR application. These two properties (two-factor mutual authentication and secure remote client identification) are not supported in the existing solutions, and they come at an additional computational cost.

The next chapter presents a set of protocols designed to allow an MR-Client, which has been identified using the CPCrE protocol, to (1) authenticate to the MR application, and (2) identify a set of MR components assigned to the MR-Client’s job domain, securely and efficiently.
Chapter 7

7. A Novel Set of Protocols for MR-Clients Authentication and MR-Job Components Identification

7.1. Chapter Introduction

This chapter presents the design and evaluation of the MR-Client Authentication (MR-CAuthN) and MR-Job Components Primary Credenital Establishment (JCPCrE) protocols. The MR-CAuthN and JCPCrE protocols are designed to perform the MR-Client authentication and the MR-Job Components identification, respectively, in an MR-Job domain. These protocols can support the following four folds: First, our method to authenticate an MR-Client to the MR application mitigates the DoS attack against the user account, as it involves a local verification process using the local and removable parts of the MR-Client authentication credential rather than using only the username and password pair to send an authentication request to the MR application. Second, a verifier (e.g. Resource Manager and Name Node) does not have to store the verification tokens that are used to validate the MR-Client’s authentication requests. This is because to validate a received request, the verifier either relays the authentication request to a backend server when the request is identified using a long-term authentication token, or uses the verification token received in a preceding interaction (in an MR-Job execution) when the request is identified using a short-term authentication token. When the verifier no longer needs this token, it can discard it.

Third, separate authentication tokens with multi-factor authentication secrets are used for MR-Client and MR-Job Components authentications. Every interaction of an MR-Client’s job submission has an authentication token to identify it. The Job Tracker, the set of all Task Trackers, and the set of all Reduce Tasks have issued with different authentication tokens to be used to identify their role-based requests in an MR-Job domain, so the MR-Client does not have to delegate his tokens. Fourth, our authentication method does not require a secure connection between the MR-Client and the verifier when the later receives an authentication request from the the MR-Client, as the authentication secrets of the authentication token are used to identify the request itself.

The evaluation of the MR-CAuthN and JCPCrE protocols is performed in two stages. First, an informal security analysis and a formal security verification of the protocols are
performed to demonstrate their resilience against known attacks and to validate their security properties. Second, the performance evaluation of the protocols is performed in terms of the computational and communication costs along with the protocol execution time. The evaluation results have been compared with those of the most related work. The design and evaluation of the MR-CAuthN and JCPCrE protocols is the fourth contribution in this thesis.

In detail, Section 7.2 presents the design preliminaries for MR-CAuthN and the JCPCrE protocols. Section 7.3 presents an overview of our MR-CAuthN protocols. Section 7.4 presents the design of the secondary authentication and verification tokens of an MR-Client and their key structure, followed by a detailed description of the MR-CAuthN protocol in Section 7.5. Section 7.6 and 7.7 present an overview of the JCPCrE protocol and the design of the primary authentication and verification tokens of the MR-Job Components and their key structure, respectively. Section 7.8 gives a detailed description of the JCPCrE protocol. Section 7.9 presents the security analysis and verification of the protocols before the performance evaluation of these protocols are reported in Section 7.10. Finally, Section 7.11 summarises this chapter.

7.2. Design preliminaries

7.2.1. Requirements

This section presents the design requirements for the MR-CAuthN and JCPCrE protocols. These requirements are divided into functional requirements, security requirements, and performance requirements.

7.2.1.1. Functional Requirements

The MR-CAuthN protocols are designed to provide a function for authentication of a remote MR-Client. This includes the following:

F2.1 The user’s login to the MR Application Client should be verified locally in the MR-Client machine before the NJ-AuthN Request of the MR-Client is sent to the Resource Manager.

F2.2 The Authentication Server should be able to verify the NJ-AuthN Request on behalf of the Resource Manager as long as the user’s login is successful.
F2.3 The MR-Client should be able to verify the NJ-AuthN Response received from the Resource Manager.

F2.4 The Name Node should be able to verify the WJ-AuthN Request of the MR-Client as long as the mutual authentication between the MR-Client and the Resource Manager is successful.

F2.5 The MR-Client should be able to verify the WJ-AuthN Response received from the Name Node.

F2.6 The Data Node should be able to verify the R-WJ-AuthN Request of the MR-Client as long as the mutual authentication between the MR-Client and the Name Node is successful.

F2.7 The MR-Client should be able to verify the R-WJ-AuthN Response received from the Data Node.

F2.8 The Authentication Server should be able to issue the secondary authentication and verification tokens that are used to provide a mutual authentication between the MR-Client and the Name Node.

F2.9 The Name Node should be able to issue the secondary authentication and verification tokens that are used to provide a mutual authentication between the MR-Client and the Data Node.

F2.10 The secondary authentication tokens should be unique for each MR-Client so no two MR-Clients could ever use the same secondary authentication tokens.

The JCPCrE protocol is designed to provide a function for identification of MR-Job Components. This includes the following:

F3.1 The MR-Client should be able to issue the primary authentication and verification tokens that provide a mutual authentication between the MR-Job Components assigned to his job and the Name Node.

F3.2 The primary authentication tokens should be unique for each Job Tracker and each set of all the MR-Job Components assigned to perform the same task, so no Job Tracker and any two MR-Job Components from different sets of the same or different MR-Job domains could ever use the same primary authentication tokens.
7.2.1.2. Security Requirements

S1 A valid user login should be assured locally. In other words, the MR Application Client should have assured locally that the user who identifies himself is the one who claims to be before sending any remote NJ-AuthN Request to the MR application.

S2 A mutual Authentication between an MR-Client and the Authentication Server and between the MR-Client and the Resources Manager should be assured.

S3 A mutual Authentication between the MR-Client and the Name Node and between the MR-Client and the Data Node should be assured.

S4 Message Authenticity: The recipient of a message (e.g. an MR-Inf. Component and MR-Client) should be assured that the message has not been altered during transit, is fresh and indeed from the claimed source.

S5 Message Confidentiality: This is a protection of the authentication requests and responses messages from any unauthorised disclosure. To counter eavesdropping attacks on the authentication information (e.g. secret keys) exchanged between the MR-AuthN Components, the protocols messages should be confidentiality protected.

S6 The primary authentication and verification tokens of MR-Job Components should be confidentiality and integrity protected while they are distributed to the respective MR components.

7.2.1.3. Performance Requirements

P1 Computational cost imposed on an MR-Client, the Authentication Server and each MR-Inf. Component as a result of (i) verifying the authentication requests and responses, and (ii) issuing the secondary authentication and verification tokens of the MR-Client, should be as low as possible.

P2 Computational cost imposed on the MR-Client as a result of issuing the primary authentication and verification tokens of his MR-Job Components should be as low as possible.

P3 The communication cost introduced by the MR-CAuthN and JCPCrE protocols as a result of exchanging messages among MR-AuthN Components should be as low as possible.
7.2.2. Notations

In addition to those notation listed in Sections 5.6 and 6.2, this section defines some of new notations used throughout this chapter.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReqFlag(c_i)</td>
<td>a request flag indicates that the type of the request, made by the (c_i) to MR application, is either to submit a new job (i.e. NJ-Submission ) or to monitor a submitted job (i.e. SJ-Monitoring )</td>
</tr>
<tr>
<td>AuthFlag(c_i)</td>
<td>an authentication-flag decision indicates that the authentication decision made on a NJ-AuthN Request of (c_i) is either allow or deny.</td>
</tr>
<tr>
<td>(CVP_{c_i})</td>
<td>(c_i)'s short-term credential validity period {it consists of (t'_s) : not before and (t'_a) : not after}. It represents the validity period of MR-Client’s secondary tokens</td>
</tr>
<tr>
<td>(CL_{c_i})</td>
<td>current location code of (c_i)</td>
</tr>
<tr>
<td>(V_{c_i,AS})</td>
<td>verification code shared between (c_i) and (AS)</td>
</tr>
<tr>
<td>(SK_{c_i}^{JD})</td>
<td>job domain-secret key of (c_i)</td>
</tr>
<tr>
<td>(SK_{c_i}^{JS})</td>
<td>job submission-secret key of (c_i)</td>
</tr>
<tr>
<td>(SK_{c_i}^{WJ})</td>
<td>writing job-secret key of (c_i),</td>
</tr>
<tr>
<td>(SK_{c_i}^{WA})</td>
<td>writing access-secret key of (c_i)</td>
</tr>
<tr>
<td>(SK_{{c_i}^{ID}})</td>
<td>infrastructure domain-secret key of all members of MR-Inf. domain</td>
</tr>
<tr>
<td>(SK_{RTj}^{RIj})</td>
<td>retrieving input-split-secret key of (RTj)</td>
</tr>
<tr>
<td>(SK_{RTj}^{ROj})</td>
<td>retrieving origin-data-secret key of (RTj)</td>
</tr>
<tr>
<td>(SK_{RTj}^{WF})</td>
<td>writing final-data-secret key of (RTj)</td>
</tr>
<tr>
<td>(Mac_{m,x})</td>
<td>a message authentication code (i.e. HMAC signature) of message (m) generated by an entity (x)</td>
</tr>
<tr>
<td>(AuthN_{RM})</td>
<td>AuthN Relay Module of the Resource Manager</td>
</tr>
<tr>
<td>(AuthN_{NN})</td>
<td>AuthN Module of the Name Node</td>
</tr>
<tr>
<td>(AuthN_{DN})</td>
<td>AuthN Module of the Data Node</td>
</tr>
</tbody>
</table>
7.3. MR-CAuthN Protocols Design for MR-Client Authentication: an Overview

This section presents an overview of our three MR-CAuthN protocols for authenticating a remote MR-Client to the MR application, i.e. U-AC, C-RM, and C-NM_AuthN protocols. Prior to describing how these protocols authenticate the MR-Client, we motivate the need for our MR-CAuthN protocols by highlighting the limitations of the current methods in authenticating an MR-Client to the MR application.

The first default and early authentication method assumed the use of an independent authentication service outside the MR application, e.g. an authentication service comes with the host operating system (OS). This is so called OS-based authentication method. In other words, the MR application then did not have its own authentication service to enforce the verification process on each request made to the MR application, rather it relies on the use of an authentication facility provided by the OSes. The user identity is whatever the host OS says it is, as a result the MR-Inf. Component (e.g. Name Node) does not perform users authentication. Any user having access to the MR-Inf. domain (cluster domain) through a configured client has access to the data stored in the cluster – only limited by file and directory permissions. Further, a user with access to the MR-Inf. domain can create a local user - with the same name as an administrator-user, and may delete or manipulate the data of the other jobs (i.e. clients).

The current authentication methods are single factor and based on password only. For example, in the Kerberos based authentication solution currently deployed for MR, each client uses a password to authenticate him/herself when submitting a job to the MR application. Then, a secret is derived and used to facilitate authentication among different MR components in the MR-Job execution. In other words, in this solution, a client’s password is used for gate level authentication, and a secret based on the password is derived and used to facilitate mutual authentications among MR components, i.e. inter-components authentication. Due to the conflict between ‘hard to guess’ and ‘hard to memorize’, clients often choose passwords that are easy to memorize. This is a single point of weakness in such authentication methods due to the fact that such passwords are usually easy to guess. If a
password is successfully guessed, then the secret will be compromised and the whole authentication chain will be put at risk. In other words, when using a weak single password, the risk of compromising the MR-Job execution is high. These authentication methods use a password based authentication approach in the gate authentication where the inter-components authentication is based on it. This means if the client to MR application authentication is compromised the inter-components authentication will be compromised as well, thus the job execution. This weakness makes the authentication methods vulnerable to traditional password attacks, namely guessing attacks which include dictionary and online and offline brute force attacks. Therefore, there is a need to strengthen the whole process of the authentication chain, making it harder to compromise both the gate-level and the inter-components authentication even if a password is successfully guessed.

As we writing this thesis, none of the current authentication methods considers a remote MR-Client authentication with multi-factor authentication. Currently, a remote MR-Client authentication with multi-factor authentication is required. This is because there have been efforts to provide the MR application in public clouds and a multi-factor authentication method could provide a stronger authentication. The current MR-Client authentication methods are considered weak techniques in such context as they are a password-based authentication method with a single-factor authentication. A more secure method could be via using two-factor authentication method, the method that does not only verify the remote MR-Client’s username and password pair which the user knows, but also involves another factor of authentication to be verified. Another factor of authentication is a secret authentication information that is supplied by another source than the user’s knowledge, such as a security token that the user have. Here, the challenge is how we could (1) establish multi-factor authentication secrets that could be used to securely authenticate every interaction of the job submission of an MR-Client to the MR application and (2) reduce the risk of an attacker or a curious MR-Client that gets hold of one of the authentication secrets, from impersonating a legitimate MR-Client. As shown in GMC model (Section 2.5), for each job submission, a client needs to make multiple interactions with multiple components of the MR application. The authentication solution should ensure that all such interactions are authenticated. Therefore, to overcome these limitations and challenges, we propose the new set of MR-CAuthN protocols which are based on the CPCrE protocol discussed in Chapter 6. This set of protocols is designed to authenticate a remote MR-Client to the MR application securely and efficiently.
With MR-CAuthN protocols, i.e. U-AC, C-RM, and C-NN_AuthN protocols, the VDAF design requirement F2 is achieved. By executing this set of protocols, (i) a user login to the MR Application Client is authenticated locally in the client machine, (ii) mutual authentications between a remote MR-Client and the Authentication Server and between the MR-Client and the Resource Manager are achieved, (iii) the secondary authentication and verification tokens, i.e. \( ST_{C_i}^1 / ST_{C_i}^2 \) and \( SVT_{C_i}^1 / SVT_{C_i}^2 \), are issued, and (iv) mutual authentications between the MR-Client and the Name Node and between the MR-Client and the respective Data Node are achieved using these secondary authentication tokens.

### 7.3.1. U-AC_AuthN Protocol

The U-AC_AuthN protocol is designed to authenticate a user login to the MR Application Client. In the design of the U-AC_AuthN protocol, (i) the user authentication credential, i.e. \( ACR_{U_i} \), is locally verified by validating \( PT_{C_i}^1 \) against \( PVT_{C_i}^1 \) in the user machine, and (ii) the local verification is based on using both the hashing mechanism and the symmetric key cryptosystem. A one-way cryptographic hash function \((H)\) is used to generate the login password-secret key, i.e. \( SK_{C_i}^{LP} \) and the local verification token, i.e. \( LVT_{U_i}^1 \). A decryption function \((E^{-1})\) is used to decrypt the local and removable parts of the MR-Client authentication credential, i.e. \( LACR_{C_i} \) and \( RACR_{C_i} \) which are issued and encrypted using the CPCrE protocol. The basic formulas behind these cryptographic operations are the following:

\[
SK_{C_i}^{LP} = H(MR\text{-Client username } ID_{C_i}, \text{ and user password } PW_{U_i} \text{ which are provided by } U_i) \quad \quad (1)
\]

\[
E^{-1}(SK_{C_i}^{LP}, \{LACR_{C_i} \text{ contains } SK_{C_i}^{LP} \text{ and } LVT_{U_i}^1\} \text{ stored locally in } CH_{U_i} ) \Rightarrow SK_{C_i}^{LP} \quad \quad (2)
\]

\[
E^{-1}(SK_{C_i}^{LP}, \{RACR_{C_i} \text{ contains } PT_{C_i}^2 \text{ and part of } PT_{C_i}^2, \text{ i.e. } LAT_{U_i}\} \text{ stored in a removable storage media } USB_{U_i} ) \Rightarrow PT_{C_i}^2 \quad \quad (3)
\]

\[
LVT_{U_i}^1 = H(SK_{C_i}^{LP} \text{ and } LAT_{U_i}) \quad \quad (4)
\]

If \( LVT_{U_i}^1 = LVT_{U_i} \) then invoke CJ-AuthN Agent and use \( PT_{C_i}^2 \) for sending an authentication request.

These formulas mean that besides what a user has memorized as part of his authentication credential (MR-Client username and the user password, i.e. \( ID_{C_i} \) and \( PW_{U_i} \)), s/he must have both the \( CH_{U_i} \) and \( USB_{U_i} \) where his encrypted \( LACR_{C_i} \) and \( RACR_{C_i} \) are located, respectively, before the client can use \( PT_{C_i}^2 \) to identify and send any authentication request to the MR application. Also, the consecutive use of the hash \((H)\) and decryption \((E^{-1})\)
functions, as mentioned above in formulas 1, 4 and 2, 3, respectively, results in the local verification token $LVT_{U_i}$ which is computed using the CPCrE protocol. If $LVT_{U_i}$ is valid, this means that (i) the user is successfully logged in to the MR Application Client, and (ii) the MR-Client assures the use of the origin and secure $L_{ACR_{C_i}}$ and $R_{ACR_{C_i}}$ to send his NJ-AuthN and WJ-AuthN requests to the MR application.

An overview of the U-AC_AuthN protocol is depicted in Figure 6.3. As it can be seen from the figure, the protocol is executed between the user, $U_i$, and the C-AuthN Agent. The protocol consists of two steps as following:

**Step1 - A User Login:** In this step, the user inserts $USB_{U_i}$ which contains the encrypted removable part of the MR-Client authentication credential ($R_{ACR_{C_i}}$) to his machine where the encrypted local part of the MR-Client authentication credential ($L_{ACR_{C_i}}$) is resided. The MR-Client username ($ID_{C_i}$) and the user password ($PW_{U_i}$) are then requested by the C-AuthN Agent and entered by the user to start the user login verification process.

**Step2 - A User Login Verification:** During this step, upon the recipient of $ID_{C_i}$ and $PW_{U_i}$ from the user, the C-AuthN Agent generates $SK_{C_i}$ and decrypts the encrypted $L_{ACR_{C_i}}$ and $R_{ACR_{C_i}}$. The C-AuthN Agent can then verify the user authentication credential using one-way cryptographic hash function and the symmetric key cryptosystem, as it has the values of $L_{ACR_{C_i}}$, $R_{ACR_{C_i}}$, $ID_{C_i}$ and $PW_{U_i}$ which are required for the local verification process. Upon a valid login, the C-AuthN Agent invokes the CJ-AuthN Agent so that the MR-Client can send the NJ-AuthN and WJ-AuthN requests to the Resource Manager and the Name Node using the C-RM and C-NN_AuthN protocols, respectively.
7.3.2. C-RM_AuthN Protocol

The C-RM_AuthN protocol is designed to achieve a mutual authentication between the MR-Client and the Authentication Server and between the MR-Client and the Resource Manager. In this design, (i) a symmetric key-based authentication method with two-factor authentication secrets (i.e. keys) are used, and (ii) four secret keys are involved; $key_1$, $key_2$, $key_3$ and $key_4$. Figure 7.2 shows an overview of the structure of these keys. As shown in the figure, $key_1$ and $key_2$ are the two-factor authentication secrets which are specified using the CPCrE protocol. $key_1$ is a user password-secret key, i.e. $K_{Ci}$, which is generated by the MR-Client using a secret information which is only known to the user (i.e. the user password $PW_{U_i}$). It is approved by the Authentication Server and presents something the MR-Client knows. $key_2$ is a client master-secret key, i.e. $P_{Ci}$, which is generated by the Authentication Server and encrypted and stored in $USB_{U_i}$. It presents something the MR-Client possesses. $key_1$ is encrypted using $key_2$. $key_2$ is derived from $key_3$ and $key_4$. $key_3$ is a session password-secret key, i.e. $SK_{Ci}^{SP}$, and $key_4$ is the Authentication Server’s private key, i.e. $PR_{AS}$. 
An overview of the C-RM_AuthN protocol is depicted in Figure 7.3. As it can be seen from the figure, the protocol is executed for every new job request to be send to the MR application, where the CJ-AuthN Agent, RM-AuthN Relay Module, and G-AuthN Service are involved. The protocol consists of eleven steps, each of which performs a specific function as following:

**Step1 - NJ-Request Initialization:** During this step, the CJ-AuthN Agent forms a NJ-AuthN Request object. The NJ-AuthN Request object consists of the message of the NJ-AuthN Request and its signature. The CJ-AuthN Agent specifies different parameters and values needed to constitute the request. It defines the current location of the user’s machine and generates the MR-Client’s authenticator. The MR-Client’s authenticator is encrypted using $P_{CI}$. The CJ-AuthN Agent then signs the encrypted authenticator along with the MR-Client’s ID and timestamp using $K_{CI}$.

**Step2 - Send NJ-AuthN Request:** In this step, the CJ-AuthN Agent sends the NJ-AuthN Request to the RM-AuthN Relay Module.

**Step3 - NJ-Request Relaying:** In this step, the RM-AuthN Relay Module verifies the freshness of the received NJ-AuthN Request to relay the request to the G-AuthN Service as a JA-AsseT Request.

**Step4 - Send JA-AsseT Request:** In this step, the RM-AuthN Relay Module sends the JA-AsseT Request which is the same object of the NJ-AuthN Request. In other words, the RM-AuthN Relay relays the authentication information received in the request sent in Step2 - by the CJ-AuthN Agent. So that the AS verifies the MR-Client request on behalf of the Resource Manager.

**Step5 - JA-Request Verification:** The G-AuthN Service verifies the JA-AsseT Request against $PVT_{CI}^2$ as following: the G-AuthN Service computes $K_{CI}$ using $PR_A$ and

![Figure 7.2 The four keys used for the C-RM_AuthN protocol](image)
which is received from the MR-Client using the CPCrE protocol, the G-AuthN Service then uses $K_{Ci}$ to (i) validate the signature of the received Request, and (ii) decrypt the encrypted part of the $PVT_{Ci}^2$ which contains $P_{Ci}$. Finally, the G-AuthN Service uses $P_{Ci}$ to validate the encrypted authenticator which is received in the request. If this verification process is valid, the MR-Client is authenticated to the AS.

**Step6 - Generate a Writing-Job Authentication Information:** Upon a positive authentication of the MR-Client to the AS, the G-AuthN Service generates (i) a new dynamic login-secret key $SK_{Ci}^{DL}$, (ii) a job submission-secret key, i.e. $SK_{Ci}^{JS}$, and (iii) the writing-job authentication information of the MR-Client. This authentication information includes the writing job-secret key, i.e. $SK_{Ci}^{WJ}$, and the job domain-secret key, i.e. $SK_{Ci}^{ID}$, that are used to constitute: (1) the MR-Client’s secondary authentication and verification tokens which are used to provide a mutual authentication between the MR-Client and the Name Node, i.e. $ST_{Ci}^1$ and $SVT_{Ci}^1$, and (2) the primary authentication and verification tokens of the MR-Job Components of the MR-Client’s job, i.e. $PT_{i,j} = \{PT_{JT}^i, PT_{TT}^i, PT_{RT}^i\}$ and $PVT_{i,j} = \{PVT_{JT}^{ij}, PVT_{TT}^{ij}, PVT_{RT}^{ij}\}$. Note that $ST_{Ci}^1$ and $SVT_{Ci}^1$ are generated by the G-AuthN Service in this step, while $PT_{i,j}^i$ and $PVT_{i,j}^i$ are generated by the CJ-AuthN Agent using JCPCrE once the mutual authentication between the MR-Client and the Resource Manager is successful (details to follow).

**Step7 - Send JA-AsserT Response:** In this step, the G-AuthN Service sends a JA-AsserT Response message along with its signature (i.e. JA-AsserT Response object) to the RM-AuthN Relay Module. The object consists of two main parts. The first part contains the authenticator, which is received from the MR-Client, and the $ST_{Ci}^1$ along with the authentication information needed to form $PT_{i,j}^i$ and $PVT_{i,j}^i$. This part is encrypted using $K_{Ci}$ and signed using $P_{Ci}$. The second part contains the $SVT_{Ci}^1$ and the authentication-decision\(^{14}\) (valid or invalid), and it is encrypted using the infrastructure domain-secret key $SK_{i}^{ID}$. The G-AuthN Service then signs these two parts along with the AS’s ID and the timestamp using $SK_{i}^{ID}$.

\(^{14}\) The authentication-decision is a digital information that specifies the result of the verification process on an authentication request. It is either Allow or Deny the request.
Step8 - JA-Response Relaying: In this step, the RM-AuthN Relay Module verifies the correctness of the received JA-AsserT Response. It decrypts the second part of the JA-AsserT Response using $SK^D_{ID}$ to read both the authentication-decision and the $SVT^3_{CI}$.

When the received JA-AsserT Response is fresh and not modified, and the authentication-decision is Allow (i.e. valid request), MR-Client is authenticated to the Resource Manager. The RM-AuthN Relay Module then generates unique identifier for the $j^{th}$ MR-Job domain created, i.e. $ID_{JD}$, and a role-identifier\(^{15}\) for each MR-Job Component or a set of MR-Job Components of this $j^{th}$ MR-Job domain and belongs to the $i^{th}$ MR-Client. Finally, the RM-AuthN Relay Module signs the first part of the JA-AsserT Response (received from the G-AuthN Service) along with the MR-Job ID and the roles-identifiers using $SK^S_{CI}$.

Step9 - Forward: In this step, the RM-AuthN Relay Module forwards the $SVT^1_{CI}$ generated in Step6 - to the NN-AuthN Module.

Step10 - Send NJ-AuthN Response: In this step, the RM-AuthN Relay Module sends the first part of the JA-AsserT Response along with the MR-Job ID and the roles-identifiers to the CJ-AuthN Agent.

Step11 - NJ-Response Verification: In this step, the CJ-AuthN Agent checks the authenticity of the NJ-AuthN Response. The CJ-AuthN Agent verifies the signature and decrypts the first part of the JA-AsserT Response using $PCi$ and $KCi$, respectively. It then verifies (i) the received authenticator and (ii) the signature of the NJ-AuthN Response message which is generated by the RM using $SK^JS_{CI}$. If they are valid, both the AS and the RM are authenticated to the MR-Client.

\(^{15}\) A role-identifier of an MR-Job Component is an identity of the MR-Job Component that presents the task that the MR-Job Component is allocated or assigned to do/play in an MR-Job domain.
7.3.3. C-NM_AuthN Protocol

The C-NM_AuthN protocol is designed to achieve a mutual authentication between the MR-Client and the DFS cluster, i.e. between the MR-Client and the Name Node and between the MR-Client and a Data Node. In this design, (i) a symmetric key-based authentication method with three-factor authentication secrets (i.e. keys) are used, and (ii) six secret keys are involved, i.e. key$_3$ to key$_6$. Figure 7.4 shows an overview of the structure of these keys. As shown in the figure key$_1$, key$_2$ and key$_3$ are the three-factor authentication secrets. key$_1$ is $K_{Ci}$, $K_{Ci}$, as mentioned in the previous section, represents something the MR-Client knows. key$_2$ and key$_3$ are supplied by two different entities and they represent something the MR-Client possesses. key$_2$ is the writing job-secret key, i.e. $SK_{Ci}^{WJ}$. It is generated by the Authentication Server and delivered to the MR-Client using the C-RM_AuthN protocol. key$_3$ is the writing access-secret key, i.e. $SK_{Ci}^{WA}$, which is generated by the Name Node. key$_3$ is sent to the MR-Client in an encrypted format through a positive authentication of the MR-
Client with the Name Node. $key_3$ is encrypted using $key_4$. $key_4$ is the infrastructure domain-secret key, i.e. $SK^{ID}$. $key_5$ and $key_6$ are the key-pair of the Name Node, i.e. the private and the public key of the Name Node ($PR_{NN}$ and $PU_{NN}$), and they are used to send $key_2$ to the Name Node (details to follow).

**Figure 7.4 The six keys used for C-NN_AuthN protocol**

An overview of the C-NN_AuthN protocol is depicted in Figure 7.5. As it can be seen from the figure, the protocol is executed for every writing job request of the MR-Client, where the CJ-AuthN Agent, NN-AuthN Module and DN-AuthN Module are involved. The protocol consists of twelve steps, each of which performs a specific function as following:

**Step1 - WJ-Request Initialization:** During this step, the CJ-AuthN Agent forms a WJ-AuthN Request object. The WJ-AuthN Request object consists of a message of the WJ-AuthN Request and its signature. To form the object, the CJ-AuthN Agent (i) identifies different parameters and values needed, this includes the user password-secret key, i.e.$K_{Ci}$, the writing job-secret key, i.e.$SK^{WJ}_{Ci}$, and the primary corresponding verification tokens of the MR-Job Components, i.e. $PVT^{*}_{j}$ (note that $PVT^{*}_{j}$ is generated using JCPCrE protocol (details to follow)), (ii) generates the MR-Client’s authenticator, (iii) encrypts the authenticator and $PVT^{*}_{j}$ using $K_{Ci}$ and encrypts $SK^{WJ}_{Ci}$ using $PU_{NN}$, and (iv) signs these encrypted parts along with the MR-Client’s ID and timestamp using $SK^{WJ}_{Ci}$.

**Step2 - Send WJ-AuthN Request:** In this step, the CJ-AuthN Agent sends the WJ-AuthN Request to the NN-AuthN Module.

**Step3 - WJ-Request Verification:** The NN-AuthN Module verifies the WJ-AuthN Request against $SVT^{3}_{Ci}$. The NN-AuthN Module decrypts the encrypted $SK^{WJ}_{Ci}$ using $PR_{NN}$. $SK^{WJ}_{Ci}$ is used to validate the signature of the received WJ-AuthN Request
message. If the signature is valid, the NN-AuthN Module decrypts the encrypted part of $SVT_{Ci}^1$, which is generated by the AS and forwarded to the Name Node via the C-RM_AuthN protocol. It is decrypted using $SK_{Ci}^{WJ}$ to read $K_{Ci}$. $K_{Ci}$ is then used to validate the encrypted authenticator. If this verification process is valid, the MR-Client is authenticated to the Name Node.

**Step4 - Generate a Writing-Access Authentication Information:** Upon a positive authentication of the MR-Client with the Name Node, the NN-AuthN Module generates a writing-access authentication information of the MR-Client which includes a writing access-secret key (i.e. $SK_{Ci}^{WA}$). This authentication information is used to constitute the secondary authentication and verification tokens that are used to provide a mutual authentication between the MR-Client and the Data Node, i.e. $ST_{Ci}^2$ and $SVT_{Ci}^2$.

**Step5 - Notify:** In this step, once $ST_{Ci}^2$ and $SVT_{Ci}^2$ are constructed, the NN-AuthN Module notifies the respective Data Node about the writing job request by sending $SVT_{Ci}^2$ to the respective DN-AuthN Module.

**Step6 - Send WJ-AuthN Response:** In this step, the NN-AuthN Module sends a WJ-AuthN Response message along with its signature (i.e. WJ-AuthN Response object) to the CJ-AuthN Agent. The message contains both $ST_{Ci}^2$ and the MR-Client’s authenticator received in the WJ-AuthN Request. These contents of the message is encrypted using $K_{Ci}$ and signed along with the Name Node’s ID and timestamp using $SK_{Ci}^{WJ}$.

**Step7 - WJ-Response Verification:** In this step, the CJ-AuthN Agent checks the WJ-AuthN Response signature using $K_{Ci}$. If the signature is valid, the CJ-AuthN Agent decrypts the contents of the response message using $SK_{Ci}^{WJ}$ to verify the received authenticator. If the authenticator is valid, the Name Node is authenticated to the MR-Client.

**Step8 - R-WJ-Request Initialization:** During this step, the CJ-AuthN Agent forms a R-WJ-AuthN Request object. The R-WJ-AuthN Request object consists of a message of the R-WJ-AuthN Request and its signature. The CJ-AuthN Agent (i) identifies different parameters and values needed to constitute the object, this includes the writing access-secret key, i.e. $SK_{Ci}^{WA}$, the writing job-secret key $SK_{Ci}^{WJ}$, and the encrypted $SK_{Ci}^{WA}$ which is encrypted using the infrastructure domain-secret key, i.e. $E(SK_{iD}, SK_{Ci}^{WA})$, (see
Figure 7.4) and defined as part of $ST_{Ci}^2$ in Step4 - (details to follow), (ii) generates the MR-Client’s authenticator, (iii) encrypts the authenticator using $SK_{Ci}^{WA}$, and (iv) signs the encrypted $SK_{Ci}^{WA}$ and the encrypted authenticator along with the MR-Client’s ID and the timestamp using $SK_{Ci}^{WA}$.

**Step9 - Send R-WJ-AuthN Request:** In this step, the CJ-AuthN Agent sends the R-WJ-AuthN Request to the DN-AuthN Module.

**Step10 - R-WJ-Request Verification:** The DN-AuthN Module verifies the R-WJ-AuthN Request against $SVT_{Ci}^2$. The DN-AuthN Module decrypts the encrypted $SK_{Ci}^{WA}$ using the infrastructure domain-secret key, i.e. $SK^{ID}$. $SK_{Ci}^{WA}$ is used to validate the signature of the received R-WJ-AuthN Request message. If the signature is valid, the DN-AuthN Module decrypts the encrypted part of $SVT_{Ci}^2$, which is received in Step5 - , using $SK_{Ci}^{WA}$. It is decrypted to read $SK_{Ci}^{WJ}$, which is generated by the AS using the C-RM_AuthN protocol.

The DN-AuthN Module uses $SK_{Ci}^{WJ}$ to validate the encrypted authenticator. If this verification process is valid, the MR-Client is authenticated to the Data Node.

**Step11 - Send R-WJ-AuthN Response:** Upon a positive authentication of the MR-Client with the Data Node, the DN-AuthN Module sends a R-WJ-AuthN Response message along with its signature (i.e. R-WJ-AuthN Response object) to the CJ-AuthN Agent. The message contains the MR-Client’s authenticator received in the R-WJ-AuthN Request. The authenticator is encrypted using $SK_{Ci}^{WA}$ and signed along with the Data Node’s ID and timestamp using $SK_{Ci}^{WJ}$.

**Step12 - R-WJ-Response Verification and MR-Client Authentication Credential Protection:** In this step, the CJ-AuthN Agent checks the R-WJ-AuthN Response signature using $SK_{Ci}^{WJ}$. If the signature is valid, the CJ-AuthN Agent decrypts the encrypted authenticator received in the R-WJ-Response message using $SK_{Ci}^{WA}$. If the authenticator is valid, the Data Node is authenticated to the MR-Client. Then the CJ-AuthN Agent protects the local and removable parts of the MR-Client authentication credential, i.e. $L_{ACR_{Ci}}$ and $R_{ACR_{Ci}}$, as the CJ-AuthN receives a new dynamic login-secret key from the AS.
7.4. MR-Client’s Secondary Tokens and Key Structure Design

This section describes the design of the secondary authentication and verification tokens of an MR-Client and their key structure in details.

7.4.1. Secondary Authentication and Verification Tokens Format

As discussed in the previous section, each MR-Client is issued with two secondary authentication tokens, i.e. $ST_{Ci}^1$ and $ST_{Ci}^2$, and each of these tokens has a corresponding verification token, i.e. $SVT_{Ci}^1$ and $SVT_{Ci}^2$, that are used to provide mutual authentications between the MR-Client and the Name Node and between the MR-Client and the Data Node, respectively. The following describes the format of these tokens.

- **MR-Client’s Secondary Authentication Tokens ($ST_{Ci}^1$ and $ST_{Ci}^2$):** Figure 7.6 shows the design of both $ST_{Ci}^1$ and $ST_{Ci}^2$. As shown in the figure, $ST_{Ci}^1$ consists of a password-
key attribute and a writing job-key attribute. The password-key attribute is defined at the issuance of the $PT^2_{Cl}$ when the MR-Client is identified to the MR application using the CPCrE protocol, while the writing job-key attribute is defined when the MR-Client is authenticated to the AS. In other words, the writing job-key attribute is defined when the NJ-AuthN Request, which is identified using $PT^2_{Cl}$, is verified against $PVT^2_{Cl}$ using the C-RM_AuthN protocol. $ST^2_{Cl}$ also has two key attributes. The first is the writing job-key attribute and the second one is a writing access-key attribute. The writing access-key attribute is defined when the MR-Client is authenticated to the Name Node, i.e. when the WJ-AuthN Request, which is identified using $ST^1_{Cl}$, is verified against $SVT^1_{Cl}$ using the C-NN_AuthN protocol. As shown in the figure, these attributes, i.e. the password-key attribute, the writing-job key attribute, and the writing-access key attribute are, respectively, approved using the AS’s private key, the client master-secret key, and the infrastructure domain-secret key. The values of these attributes are chosen pragmatically and incorporated technically to form three-factor authentication secrets and provide a mutual authentications between the MR-Client and the Name Node and between the MR-Client and the Data Node. Table 7.1 describes the values of these attributes.

![The secondary authentication tokens of $ACR_{Cl}$](image)

**Figure 7.6 The data structure of the secondary authentication tokens of an MR-Client**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password-key attribute</td>
<td>Password-key attribute is a secret attribute, which is also applied for $PT^2_{Cl}$. It represents something $U_i$ knows (user password, $PW_{U_i}$). It contains the user password-secret key which is approved by the AS using AS’s private key and used as a first-factor authentication secret to authenticate $C_i$ to the DFS via C-NN_AuthN protocol.</td>
</tr>
</tbody>
</table>

Table 7.1 The contents of the secondary authentication tokens of an MR-Client
Writing job-key attribute

Writing job-key attribute is also a secret attribute that is applied for both $ST_{Ci}^{1}$ and $ST_{Ci}^{2}$. It represents something $Ci$ possesses, named writing job-secret key. It is issued by the G-AuthN Service and approved using the client master-secret key through a positive authentication of the MR-Client with the AS. It is used as a second-factor authentication secret to authenticate $Ci$ to the DFS via C-NN_AuthN protocol.

Writing access-key attribute

Writing access-key attribute contains a writing access-secret key of $Ci$, i.e. $SK_{WJ}^{1}$. This attribute is applied for $ST_{Ci}^{2}$. It is issued by the Name Node and approved using the infrastructure domain-secret key through a positive authentication of the MR-Client with the Name Node. The writing access-secret key is used as a third-factor authentication secret to authenticate the MR-Client to the DFS via C-NN_AuthN protocol.

- **MR-Client’s Secondary Corresponding Verification Tokens ($SVT_{Ci}^{1}$ and $SVT_{Ci}^{2}$):**
  Figure 7.7 shows the design of both $SVT_{Ci}^{1}$ and $SVT_{Ci}^{2}$. As shown in the figure, $SVT_{Ci}^{1}$ consists of three basic attribute and one key attribute. The three basic attributes are the MR-Client ID attribute, and valid-from and to attributes. The one key attribute is the password-key attribute which is generated at the issuance of the $PVT_{Ci}^{2}$ using the CPCrE protocol. When the Name Node uses this key attribute, i.e. the password-key, to validate the WJ-AuthN Request received from the MR-Client, the valid-from and valid-to attributes can be revealed to confirm the token validity period. These attributes are approved using the writing job-secret key which is generated at the issuance of the $ST_{Ci}^{1}$. This set of approved attributes is used to verify the WJ-AuthN Request that is identified using $ST_{Ci}^{1}$ and sent to the Name Node by the MR-Client. $SVT_{Ci}^{2}$ consists of a set of three attributes; a MR-Client ID attribute, a MR-Job ID attribute and a writing job-key attribute. As shown in the figure, these attributes are approved by the Name Node using the writing access-secret key which is defined at the issuance of $ST_{Ci}^{2}$. This set of attributes is used to validate the R-WJ-AuthN Request that is identified using $ST_{Ci}^{2}$ and sent to the respective Data Node. Table 7.2 describes the values of these attributes.
**Figure 7.7 The data structure of the secondary verification tokens of an MR-Client**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR-Client ID attribute</td>
<td>MR-Client ID attribute specifies MR-Client username which is used to distinguish each MR-Client user, i.e. $ID_{C_i}$. It is the same attribute applied for $PT_{C_i}$.</td>
</tr>
<tr>
<td>MR-Job ID attribute</td>
<td>MR-Job ID attribute specifies the MR-Job domain’s identifier of an MR-Client. It is defined by the Resource Manager when the MR-Client is authenticated to the Resource Manager. This attribute is applied for both $SVT_{C_i}^1$ and $SVT_{C_i}^2$.</td>
</tr>
<tr>
<td>Password-key attribute</td>
<td>Password-key attribute is a secret attribute which is applied for $SVT_{C_i}^2$. It is used to validate the user password-secret key that is used to identify the WJ-AuthN Request received by the Name Node.</td>
</tr>
<tr>
<td>Writing job-key attribute</td>
<td>Writing job-key attribute is a secret attribute which is applied for both $SVT_{C_i}^1$ and $SVT_{C_i}^2$. It is used to validate the writing job-secret key that is used to identify both the WJ-AuthN Request and the R-WJ-AuthN Request received by the Name Node and the Data Node, respectively.</td>
</tr>
<tr>
<td>Writing access-key attribute</td>
<td>Writing access-key attribute is also a secret attribute which is applied for $SVT_{C_i}^2$. It is used to validate the writing access-secret key that is used to identify the R-WJ-AuthN Request received by the respective Data Node.</td>
</tr>
<tr>
<td>Valid-from</td>
<td>An attribute value specifies when $SVT_{C_i}^2$ can be used, thus $ST_{C_i}^2$.</td>
</tr>
<tr>
<td>Valid-to</td>
<td>An attribute value specifies whether $SVT_{C_i}^1$ is expired or not, thus $ST_{C_i}^1$.</td>
</tr>
</tbody>
</table>
7.4.2. Key Structure

This section explains the key generation and structure of the values of the attributes discussed above. As mentioned in Section 7.3, once the MR-Client is authenticated to AS, variety of cryptographic secret keys are issued to form his 1st and 2nd secondary authentication tokens, i.e. $ST_{Ci}^1$ and $ST_{Ci}^2$, and their corresponding verification tokens, i.e.$SVT_{Ci}^1$ and $SVT_{Ci}^2$. Figure 7.8 shows the key structure of these tokens. As can be seen from the figure, three independent secret keys, i.e. the user password-secret key ($K_{Ci}$), the writing job-secret key ($SK_{Ci}^{WJ}$), and the writing access-secret key ($SK_{Ci}^{WA}$), are used to form the tokens. These keys are generated randomly using a set of random numbers and timestamps and they are supplied by different MR-AuthN Components.

The key structure of $ST_{Ci}^1$ consists of $K_{Ci}$ and $SK_{Ci}^{WJ}$. As discussed before in Section 6.4, $K_{Ci}$ is generated by the C-AuthN Agent using $SK_{Ci}^{SP}$ and approved by the G-AuthN Service using the AS’s private key, i.e. $PR_{AS}$. $K_{Ci}$ represents something the MR-Client knows. This is because $SK_{Ci}^{SP}$ is generated from the user password, i.e. $PW_{Ui}$, which is selected and known only by the user, the random number which is generated by the C-AuthN Agent, i.e. $r_{Ci}$, and the timestamp when $r_{Ci}$ is generated. $SK_{Ci}^{WJ}$ is generated (see Figure 7.8) from $P_{Ci}$, a random number ($r_{AS}$), and the timestamp when $r_{AS}$ is generated ($t_{r_{AS}}$), all of which are generated by the G-AuthN Service. $SK_{Ci}^{WJ}$ represents something the MR-Client possesses. This is because, $SK_{Ci}^{WJ}$ is mainly generated from the client master-secret key, i.e. $P_{Ci}$, which is generated randomly and uniquely by the G-AuthN Service and possessed only by the particular MR-Client ID in the removable storage media $USB_{Ui}$. $SK_{Ci}^{WJ}$ is encrypted using the public key of the Name Node ($PU_{NN}$) when the MR-Client sends his WJ-AuthN Request to the Name Node.
The key structure of \( ST_{Ci}^2 \) consists of \( SK_{Ci}^{WJ} \) and \( SK_{Ci}^{WA} \). \( SK_{Ci}^{WA} \) is generated from a random number \( (r_{NN}) \) and the timestamp when \( r_{NN} \) is generated \( (\tau_{FNN}) \) along with the MR-Client and job domain IDs, i.e. \( ID_{Ci} \) and \( ID_{Dj} \). \( SK_{Ci}^{WA} \) also represents something the MR-Client possesses. This is because the key is generated randomly and uniquely by the NN-AuthN Module. The MR-Client obtains this key, when the MR-Client is authenticated to the Name Node, in order to authenticate his access request sent to the respective Data Node in the DFS. \( SK_{Ci}^{WA} \) is encrypted using the infrastructure domain-secret key when MR-Client sends his access request (i.e. redirected WJ-AuthN Request) to the respective Data Node. Further, as shown in the figure, the corresponding verification tokens, i.e. \( SVT_{Ci}^1 \) and \( SVT_{Ci}^2 \), consist of the same secret keys that are used to form of \( ST_{Ci}^1 \) and \( ST_{Ci}^2 \), and they are structured in an encrypted keychain to form these verification tokens. \( K_{Ci} \) is encrypted using \( SK_{Ci}^{WJ} \) which is encrypted using \( SK_{Ci}^{WA} \).

[Figure 7.8 The key generation and structure of the secondary authentication and verification tokens of an MR-Client]

The design of these tokens and their key structure has the following four-folds: First, the MR-Client’s secondary authentication tokens, i.e. \( ST_{Ci}^1 \) and \( ST_{Ci}^2 \), and their corresponding verification tokens, i.e. \( SVT_{Ci}^1 \) and \( SVT_{Ci}^2 \), are generated and used for a short-term through a job submission and not stored for a long-term use. Second, \( SVT_{Ci}^1 \) and \( SVT_{Ci}^2 \) are designed in
an encrypted keychain format which gives (1) the Name Node and the respective Data Node (i.e. verifiers) the ability to verify both the WJ-AuthN Request and the R-WJ-AuthN Request of MR-Client, (2) an assurance that the secrets of these verification tokens are confidentiality protected. Third, these keys are generated automatically and does not require any user’s involvement when the MR-Client’s authentication requests are being validated. Fourth, three independent secret keys are generated in a secure random-basis and supplied by three separate resources (MR-Client, AS and NN), so that these keys are used as three-factor authentication secrets to authenticate the MR-Client to the DFS.

7.5. MR-CAuthN Protocols in Details

7.5.1. U-AC_AuthN Protocol

The U-AC_AuthN protocol is described in details below and shown in Figure 7.9.

Step1 - A User Login: In this step,
- The user $U_i$ inserts his removable storage media, i.e. $USB_{U_i}$, which contains $E(SK_{CL}^{DL}, R_{ACR}_{C_i})$, to the $CH_{U_i}$, and runs the C-AuthN Agent. Then $U_i$ chooses the request type by setting the $ReqFlag_{C_i}$ to a NJ-Submission$^{16}$.
- The CJ-AuthN Agent $(AuthN_{C_i})$ then prompts $U_i$ to enter his MR-Client username and password, i.e. $ID_{C_i}$ and $PW_{U_i}$.
- $U_i$ enters $ID_{C_i}$ and $PW_{U_i}$.

Step2 - A User Login Verification: This function is executed by the C-AuthN Agent. Upon the receipt of $ID_{C_i}$ and $PW_{U_i}$, the C-AuthN Agent verifies the user login credential using the following operations:
- The C-AuthN Agent generates the login password-secret key, i.e. $SK_{CL}^{LP}$ by computing $SK_{CL}^{LP} = \mathbb{H}(ID_{C_i} \parallel PW_{U_i})$.
- Once $SK_{CL}^{LP}$ is generated, the C-AuthN Agent reads the encrypted local part of the MR-Client’s authentication credential, i.e. $L_{ACR}_{C_i}$, from the local stego-file, and decrypts it using $SK_{CL}^{LP}$, i.e. $E^{-1}(SK_{CL}^{LP}, L_{ACR}_{C_i})$.

---

$^{16}$ There are two types of requests that may sent to the Resource Manager. The first one is to submit a new job (i.e. NJ-Submission), and the second one is to monitor a submitted job (i.e. SJ-Monitoring). In this chapter we include the authentication procedures to submit a new job, i.e. when the $ReqFlag_{C_i}$ is set to NJ-Submission.
• Then, the C-AuthN Agent reads both the dynamic login-secret key, i.e. $SK_{DL_{ci}}$, and the local verification token of the user, i.e. $LVT_{UL_i}$, from $L_{ACR_{ci}}$ as $L_{ACR_{ci}} = LVT_{UL_i}|SK_{DL_{ci}}$, and the $LVT_{UL_i} = \mathbb{H}(ID_{ci} \| PW_{UL_i} \| \mathbb{H}(r_{ci} \| t_{rc_{ci}} \| SK_{DL_{ci}}))$.

• The C-AuthN Agent reads the encrypted removable part of the MR-Client’s authentication credential, $R_{ACR_{ci}}$, which is located in $USB_{UL_i}$, and decrypts it using $SK_{DL_{ci}}$, i.e. $E^{-1}(SK_{DL_{ci}}, R_{ACR_{ci}})$.

• Once it is decrypted, the C-AuthN Agent reads the login authentication token of the MR-Client, i.e. $LAT_{UL_i} = \mathbb{H}(r_{ci} \| t_{rc_{ci}} \| SK_{DL_{ci}})$, from $R_{ACR_{ci}}$ as $R_{ACR_{ci}} = LAT_{UL_i}||PT_{2C_{ci}}$.

• Finally, the C-AuthN Agent computes $LVT_{UL_i}^* = \mathbb{H}(SK_{UL_i}^{LP} \| LAT_{UL_i})$, where $SK_{UL_i}^{LP} = \mathbb{H}(ID_{ci} \| PW_{UL_i})$ and $LAT_{UL_i} = \mathbb{H}(r_{ci} \| t_{rc_{ci}} \| SK_{DL_{ci}})$, and checks if the computed $LVT_{UL_i}^*$ equals to the one read from $L_{ACR_{ci}}$. If they match (i.e. $LVT_{UL_i}^* = LVT_{UL_i}$), the user is authenticated to the MR Application Client and the CJ-AuthN Agent ($AuthN^1_{C_{ci}}$) is invoked. So that the MR-Client is ready to use $PT_{2C_{ci}}$ to identify and send his NJ-AuthN Request to the Resource Manager (satisfy (F2.1)).

---

**Figure 7.9 U-AC authentication protocol**

---

207
7.5.2. C-RM_AuthN Protocol

The C-RM_AuthN protocol is described in details below and shown in Figure 7.10.

Step 1 - NJ-Request Initialization: Upon a successful user login using U-AC_AuthN protocol, the CJ-AuthN Agent initiates the NJ-AuthN Request as following:

- The CJ-AuthN Agent reads the user password-secret key, i.e. $K_{Ci}$, the client master-secret key, i.e. $P_{Ci}$, and the client location-secret code, i.e. $L_{Ci}$, from R_ACR$_{Ci}$ as $R_{ACR}_{Ci} = LAT_{U_i} || PTC_{Ci}^2; L_{Ci}$ to be used to form the MR-Client authenticator, $P_{Ci}$ to be used to encrypt the authenticator, and $K_{Ci}$ to be used to sign the contents of the NJ-AuthN Request message which includes the encrypted authenticator (details to follow).

- It constructs an MR-Client authenticator for the NJ-AuthN Request ($Authenticator^{NJ}_{Ci}$) using $ID_{Ci}$, a current location-secret code ($CL_{Ci}$), and the current timestamp of the MR-Client ($t_{Ci,NJ}$), i.e. $Authenticator^{NJ}_{Ci} = \{ID_{Ci}||CL_{Ci}|| t_{Ci,NJ}\}$, where $CL_{Ci} = \mathbb{H}(IP_{CH_{U_i}}||L_{Ci})$, $IP_{CH_{U_i}}$ is the IP address of the current location of the MR-Client, and $L_{Ci}$ is the client location-secret code, which represents where the MR-Client is identified by the AS using the CPCrE protocol.

- The CJ-AuthN Agent then encrypts $Authenticator^{NJ}_{Ci}$ using $P_{Ci}$, i.e. $E(P_{Ci}, Authenticator^{NJ}_{Ci})$. Besides the confidentiality protection provided for the authenticator against eavesdropping attacks and unauthorised entities, this encryption, i.e. $E$, provides a proof of having the authentication secret $P_{Ci}$.

- Once $Authenticator^{NJ}_{Ci}$ is encrypted, the CJ-AuthN Agent constructs the NJ-AuthN Request message, i.e. $Msg_{NJ,Ci} = \{ E(P_{Ci}, Authenticator^{NJ}_{Ci}) || ReqFlag_{Ci} || ID_{Ci} || t_{Ci,NJ} \}$, where $ReqFlag_{Ci}$ is identified using U-AC_AuthN protocol and $t_{Ci,NJ}$ is used to resist replay attacks.

- It computes the HMAC signature of the message. In other words, it signs the message by computing the message authentication code using HMAC function, discussed in Section 5.10, i.e. $Mac_{NJ,Ci} = HMAC(Msg_{NJ,Ci}, K_{Ci})$ where, $Mac_{NJ,Ci}$ is the message authentication code of $Msg_{NJ,Ci}$, and HMAC is the keyed-hash function which uses $K_{Ci}$ to sign the message. In addition to the message origin
authentication provided by \( HMAC \) signature using the authentication secret \( K_{C_i} \), it is used to resist any forgery or unauthorised modification of the message contents.

- Finally, the CJ-AuthN Agent constructs a NJ-AuthN Request object which contains both the NJ-AuthN Request message and its HMAC signature, i.e. \( \{ Msg_{NJC_i} || Mac_{NJC_i} \} \).

**Step 2 - Send NJ-AuthN Request:** In this step, the CJ-AuthN Agent \(( AuthN^l_{C_i} )\) sends the NJ-AuthN Request to the RM-AuthN Relay Module \(( AuthN_{RM} )\).

\[
AuthN^l_{C_i} \rightarrow AuthN_{RM} : NJ-AuthN Request \{ Msg_{NJC_i} || Mac_{NJC_i} \}
\]

where \( Msg_{NJC_i} = \{ E( P_{C_i}, Authenticator^{NJ}_{C_i})|| ReqFlag_{C_i}|| ID_{C_i}|| t_{C_i,NJ} \} \), and \( Mac_{NJC_i} = HMAC(Msg_{NJC_i}, K_{C_i}) \).

**Step 3 - NJ-Request Relaying:** The RM-AuthN Relay Module verifies the NJ-AuthN Request in terms of its freshness and type. First, it checks if the difference between its local timestamp \( (t_{RM}) \) and the timestamp contained in the message, i.e. \( t_{C_i,NJ} \), is less than a predefined value \( (\Delta T) \), i.e. \( (t_{RM} - t_{C_i,NJ} < \Delta T) \). Second, it checks if the request type is a NJ-Submission (i.e. \( ReqFlag_{C_i} = NJ-Submission \)). If these two verification processes are valid, the RM-AuthN Relay Module relays the request to the G-AuthN Service as a JA-Assert Request to the G-AuthN Service.

**Step 4 - Send JA-AssertT Request:** In this step, the RM-AuthN Relay Module sends the JA-AssertT Request, i.e. the message \( Msg_{JAR,RM} \) along with its signature \( Mac_{JAR,RM} \), to the G-AuthN Service \(( AuthN_{AS} )\).

\[
AuthN_{RM} \rightarrow AuthN_{AS} : JA-AssertT Request \{ Msg_{JAR,RM} || Mac_{JAR,RM} \}
\]

where \( Msg_{JAR,RM} = Msg_{NJC_i} \) and \( Mac_{JAR,RM} = Mac_{NJC_i} \).

**Step 5 - JA-Request Verification:** In this step, upon the receipt of the JA-Assert Request, the G-AuthN Service validates the authenticity of the request to confirm the identity of the MR-Client. To do so (i.e. satisfy F2.2), it does the following operations.

- The G-AuthN Service reads \( C_i \)'s identity and timestamp, i.e. \( ID_{C_i} \) and \( t_{C_i,NJ} \), and the HMAC signature of the received message i.e. \( Mac_{NJC_i} \).
• The G-AuthN Service verifies the freshness of the received message by checking if the difference between its local timestamp ($t_{AS}$) and the timestamp contained in the message ($t_{Ci,NJ}$) is less than a predefined value ($\Delta T$), i.e. $(t_{AS} - t_{Ci,NJ} < \Delta T)$.

• Once the freshness is valid, the G-AuthN Service generates the user password-secret key, i.e. $K_{Ci} = \text{HMAC}(SK_{Ci}^{SP} \| PR_{AS})$, to verify the correctness of the HMAC signature of the received message, where $SK_{Ci}^{SP}$ is the session password-secret key which is generated using the CPCrE protocol, and $PR_{AS}$ is the AS’s private key which is only known to the AS.

• The G-AuthN Service computes the HMAC signature of the received message using the generated $K_{Ci}$, i.e. $Mac_{NJ,Ci}^{*} = \text{HMAC}(Msg_{NJ,Ci}, K_{Ci})$, to check whether the computed HMAC signature, $Mac_{NJ,Ci}^{*}$, equals the received one, i.e. if $(Mac_{NJ,Ci}^{*} = Mac_{NJ,Ci})$. If the signature is valid, the G-AuthN Service accepts the authentication secret $K_{Ci}$. Otherwise, it considers the request as invalid request and terminate the session.

• The G-AuthN Service decrypts the encrypted part of the $PVT_{Ci}^{2}$ using $K_{Ci}$, i.e. $E^{-1}(K_{Ci}, \{ID_{Ci} || P_{Ci} || L_{Ci} || ts || tn\})$, and verifies the $C_{i}$’s long-term credential validity period. In other words it verifies that if the timestamp of the AS when the request is received is not before $t_s$ and not after $t_n$, i.e. if $(t_s \leq t_{AS} \leq t_n)$. If the result is valid, the G-AuthN Service reads the client master-secret key $P_{Ci}$. Otherwise, it considers the request as invalid and terminate the session.

• The G-AuthN Service decrypts the encrypted MR-Client’s authenticator in the received message using $P_{Ci}$, i.e. $E^{-1}(P_{Ci}, Authenticator_{Ci}^{N})$.

• The G-AuthN Service computes the current location-secret code, i.e. $CL_{Ci}^{*} = \text{HMAC}(IP_{CH,ij} || L_{Ci})$ and constructs a new MR-Client’s authenticator, i.e. $Authenticator_{Ci}^{N} = \{ID_{Ci}^{*} || CL_{Ci}^{*} || t_{Ci,NJ}\}$, where $ID_{Ci}^{*}$ is the MR-Client’s identifier which is read from $PVT_{Ci}^{2}$, and $t_{Ci,NJ}$ is the MR-Client’s timestamp which is read from the received NJ-AuthN Request. The G-AuthN Service then verifies if the $Authenticator_{Ci}^{N}$ equals to the one received in the request, i.e. if $(Authenticator_{Ci}^{N} = Authenticator_{Ci}^{N})$, if the authenticator is valid, the G-AuthN Service accepts both the authentication secret $P_{Ci}$ and the additional
authentication secret, i.e. the location-secret code. Otherwise an MR-Client verification code is generated by the Authentication Server, i.e. $V_{C_i AS}$, and sent to the MR-Client via his email. This is to ensure that the MR-Client who moved to a new location is the one who claims to be. If the G-AuthN Service receives a valid verification code from the CJ-AuthN Agent, then it accepts the request and continues to the next step, up to this point the MR-Client is authenticated to the Authentication Server. Otherwise, it considers the request as invalid and terminate the session.

**Step 6 - Generate a Writing-Job Authentication Information:** This function is executed by the G-AuthN Service. Upon a positive authentication of the MR-Client with the AS, the G-AuthN Service generates (i) a new dynamic login-secret key ($SK_{C_i}^{DL'}$), (ii) a job submission-secret key ($SK_{C_i}^{JS}$), and (iii) the writing-job authentication information of the MR-Client which includes the writing job-secret key, i.e. $SK_{C_i}^{WJ}$, and the job domain-secret key, i.e. $SK_{C_i}^{JD}$. This authentication information are used to constitute: (1) his secondary authentication and verification tokens that are used to provide a mutual authentication between the MR-Client and the Name Node, i.e. $ST_{C_i}^1$ and $SVT_{C_i}^1$, and (2) the primary authentication and verification tokens that are used to identify the MR-Job Components to be assigned to his job domain using the JCPCrE protocol, i.e. $PT_{*j} = \{PT_{JT}^j, PT_{TT}^j, PT_{RT}^j\}$ and $PVT_{*j} = \{PVT_{JT}^j, PVT_{TT}^j, PVT_{RT}^j\}$. (details to follow). In details, the G-AuthN Service does the following operations (i.e. satisfy (F2.8)):

- It generates four random secret numbers $r_{1'AS}$, $r_{3AS}$, $r_{4AS}$ and $r_{5AS}$. It also specifies the timestamps when these random numbers are generated, $t_{r_{1'AS}}$, $t_{r_{3AS}}$, $t_{r_{4AS}}$ and $t_{r_{5AS}}$, respectively. As explained in Section 7.4, these random numbers and timestamps along with $ID_{C_i}$ and $P_{C_i}$ are used to generate $SK_{C_i}^{DL'}$, $SK_{C_i}^{JS}$, $SK_{C_i}^{WJ}$, and $SK_{C_i}^{JD}$ as following:

\[
SK_{C_i}^{DL'} = \mathbb{H}(r_{1'AS} \parallel t_{r_{1'AS}})
\]
\[
SK_{C_i}^{JS} = \mathbb{H}(r_{3AS} \parallel t_{r_{5AS}})
\]
\[
SK_{C_i}^{WJ} = \mathbb{H}(P_{C_i} \parallel r_{4AS} \parallel t_{r_{3AS}})
\]
\[
SK_{C_i}^{JD} = \mathbb{H}(P_{C_i} \parallel r_{5AS} \parallel t_{r_{4AS}})
\]
The G-AuthN Service encrypts $ID_{C_i}$ and $K_{C_i}$ along with $CVP'_{C_i}$ (i.e. $t'_i$ and $t''_i$) using $SK_{C_i}^{WJ}$, i.e. $E(SK_{C_i}^{WJ}, \{ID_{C_i}||K_{C_i}||t'_i||t''_i\})$.

It then constructs $ST_{C_i}^3$ and $SVT_{C_i}^3$, where $ST_{C_i}^3 = \{K_{C_i}||SK_{C_i}^{WJ}\}$, and $SVT_{C_i}^3 = \{ID_{C_i}\|E(SK_{C_i}^{WJ},\{ID_{C_i}||K_{C_i}||t'_i||t''_i\})\}$.

The G-AuthN Service then encrypts $ST_{C_i}^3$ with other authentication information using $K_{C_i}$ as follows:

$E(K_{C_i}, \{ST_{C_i}^3||SK_{C_i}^{JH}||SK_{C_i}^{JS}||SK_{C_i}^{DL}||ID_{NN}||IP_{NN}||PU_{NN}||Authenticator_{C_i}^{NJ^*}\})$, where $PU_{NN}$ is the NN’s public key which is used by the MR-Client to encrypt $SK_{C_i}^{WJ}$, $ID_{NN}$ and $IP_{NN}$ are, respectively, the identity and the IP address of the Name Node to which the WJ-AuthN Request to be sent, and the $Authenticator_{C_i}^{NJ^*}$ is the MR-Client’s authenticator which is received in the NJ-AuthN Request.

The G-AuthN Service constructs the first part of the JA-Assert Response message $Msg_{JA,AS}^3$, and computes its HMAC signature of $Msg_{JA,AS}^3$ using $P_{C_i}$ as following:

$$Msg_{JA,AS}^3 = E(K_{C_i}, \{ST_{C_i}^3||SK_{C_i}^{JH}||SK_{C_i}^{JS}||SK_{C_i}^{DL}||ID_{NN}||IP_{NN}||PU_{NN}||Authenticator_{C_i}^{NJ^*}\})||t_{AS,JA}||ID_{AS}$$

$$Mac_{JA,AS}^1 = HMAC (Msg_{JA,AS}^3, P_{C_i})$$

where $t_{AS,JA}$ is AS’s timestamp when the G-AuthN Service constructs the $Msg_{JA,AS}^3$. $t_{AS,JA}$ is used to resist replay attacks, and the HMAC signature is used to resist any forgery or unauthorised modification of the contents of $Msg_{JA,AS}^3$ and provides its origin authentication. Both, i.e. $Msg_{JA,AS}^1$ and $Mac_{JA,AS}^1$, to be relayed to the CJ-AuthN Agent so that the AS can be authenticated to the MR-Client and the CJ-AuthN Agent is approved to generate the MR-Job authentication information of his MR-Job domain (details to follow).

Once $Msg_{JA,AS}^1$ and $Mac_{JA,AS}^1$ are constructed, the G-AuthN Service encrypts $SVT_{C_i}^3$ along with the $SK_{C_i}^{JS}, ID_{C_i}$ and $AuthFlag_{C_i}$ using the infrastructure domain-secret key ($SK_{C_i}^{ID}$) i.e. $E(SK_{C_i}^{ID}, \{SVT_{C_i}^3||SK_{C_i}^{JS}||ID_{C_i}||AuthFlag_{C_i}\})$, where the authentication-flag decision is set to a allow status, i.e. $AuthFlag_{C_i} = Allow$. This

212
encryption is used to protect the confidentiality of these contents against eavesdropping by attackers or unauthorised entities.

- The G-AuthN Service constructs a JA-AssertT Response message, i.e. \( \text{Msg}_{\text{JA,AS}} = \{\{\text{Msg}_{\text{JA,AS}} \| \text{Mac}_{\text{JA,AS}}\| E(\text{SK}^ID, \{SVT^I_{C{i}}\|SK^J_{C{i}}\|ID_{C{i}}\|\text{AuthFlag}_{C{i}})\}) \| ID_{AS} \| t_{AS,JA}\} \), and computes its HMAC signature using \( \text{SK}^ID \) to provide message origin authentication and resist any active attack against the message, i.e. \( \text{Mac}_{\text{JA,AS}} = \text{HMAC}(\text{Msg}_{\text{JA,AS}}, \text{SK}^ID) \).

- Finally, the G-AuthN Service constructs an JA-AssertT Response object, i.e. \( \{\text{Msg}_{\text{JA,AS}} \| \text{Mac}_{\text{JA,AS}}\} \).

**Step7 - Send JA-AssertT Response:** In this step, the G-AuthN Service sends the JA-AssertT Response to the RM-AuthN Relay Module.

\[
\text{AuthN}_{\text{AS}} \rightarrow \text{AuthN}_{\text{RM}} : \text{JA-AssertT Response} \{\text{Msg}_{\text{JA,AS}} \| \text{Mac}_{\text{JA,AS}}\}
\]

where \( \text{Msg}_{\text{JA,AS}} = \{\{\text{Msg}_{\text{JA,AS}} \| \text{Mac}_{\text{JA,AS}}\| E(\text{SK}^ID, \{SVT^I_{C{i}}\|SK^J_{C{i}}\|ID_{C{i}}\|\text{AuthFlag}_{C{i}})\}) \| ID_{AS} \| t_{AS,JA}\} \), and \( \text{Mac}_{\text{JA,AS}} = \text{HMAC}(\text{Msg}_{\text{JA,AS}}, \text{SK}^ID) \).

**Step8 - JA-Response Relaying:** When the JA-AssertT Response is received, the RM-AuthN Relay Module performs the following:

- The RM-AuthN Relay Module reads \( ID_{AS} \), \( t_{AS,JA} \), and \( \text{Mac}_{\text{JA,AS}} \).

- It checks if the difference between its local timestamp (\( t_{RM} \)) and the timestamp contained in the message (\( t_{AS,JA} \)) is less than a predefined value (\( \Delta T \)), i.e. \( (t_{RM} - t_{AS,JA} < \Delta T) \).

- If the message is fresh, the RM-AuthN Relay Module verifies the HMAC signature of the received response message using \( \text{SK}^ID \). It computes \( \text{Mac}_{j,AS} = \text{HMAC}(\text{Msg}_{j,AS,}, \text{SK}^ID) \), to check whether the computed HMAC signature, \( \text{Mac}_{j,AS} \), equals the received one, i.e. if \( \text{Mac}_{j,AS} = \text{Mac}_{j,AS} \).

- The RM-AuthN Relay Module then decrypts \( E(\text{SK}^ID, \{SVT^I_{C{i}}\|SK^J_{C{i}}\|ID_{C{i}}\|\text{AuthFlag}_{C{i}})\}) \) using \( \text{SK}^ID \), i.e. \( E^{-1}(\text{SK}^ID, \{SVT^I_{C{i}}\|SK^J_{C{i}}\|ID_{C{i}}\|\text{AuthFlag}_{C{i}})\}) \).

- As the \( \text{AuthFlag}_{C{i}} \) status is Allow, the MR-Client is authenticated to the Resource Manager. The RM-AuthN Relay Module generates a unique identifier for the \( j^{th} \) MR-
Job domain created, i.e. $ID_{JD,i}$, and a role-identifier for each MR-Job Component or set of MR-Job Components of this $i^{th}$ MR-Job domain and belongs to the $i^{th}$ MR-Client. In other words, it generates the Job Tracker role ID, $ID_{JT,i}$, the Task Tracker role ID, $ID_{TT,i}$, and the Reduce Task role ID, $ID_{RT,i}$. These identities are sent to the CJ-AuthN Agent as they are used along with the other authentication information defined in $Msg_{J,AS}^1$ to generate $PT_{i,j}$ and $PVT_{i,j}$.

- The RM-AuthN Relay Module constructs the NJ-AuthN Response message, i.e. $Msg_{NJ,RM} = \{Msg_{J,AS}^1 || Mac_{J,AS}^1 || ID_{JD} || ID_{JT} || ID_{TT} || ID_{RT} || ID_{RM} || t_{RM,NJ}\}$ and computes its HMAC signature using $SK_{c_i}^{JS}$, i.e. $Mac_{NJ,RM} = HMAC (Msg_{NJ,RM}, SK_{c_i}^{JS})$, where $ID_{RM}$ is the identity of the Resource Manager, $t_{RM,NJ}$ is the timestamp of the Resource Manager when $Msg_{NJ,RM}$ is generated, and it is used to resist replay attacks.

- It also constructs a Forward message, i.e. $Msg_{F,RM} = \{E(SK_{c_i}^{ID},\{SVT_{c_i}^1 || ID_{JD}\}) || ID_{RM} || t_{RM,F}\}$, and computes its HMAC signature using $SK_{c_i}^{ID}$, i.e. $Mac_{F,RM} = HMAC (Msg_{F,RM}, SK_{c_i}^{ID})$, where $t_{RM,F}$ is RM’s timestamp when the Forward message is generated, and it is used to resist replay attacks.

- Finally, the RM-AuthN Relay Module constructs both NJ-AuthN Response object, i.e. $\{Msg_{NJ,RM} || Mac_{NJ,RM}\}$, and the Forward object, i.e. $\{Msg_{F,RM} || Mac_{F,RM}\}$.

Step9 - Forward: In this step, the RM-AuthN Relay Module sends the Forward object to the NN-AuthN Module ($AuthN_{NN}$).

$$AuthN_{RM} \rightarrow AuthN_{NN} : \textbf{Forward} \{Msg_{F,RM} || Mac_{F,RM}\}$$

Where $Msg_{F,RM} = \{E(SK_{c_i}^{ID},\{SVT_{c_i}^1 || ID_{JD}\}) || ID_{RM} || t_{RM,F}\}$, and $Mac_{F,RM} = HMAC (Msg_{F,RM}, SK_{c_i}^{ID})$.

Step10 - Send NJ-AuthN Response: In this step, the RM-AuthN Relay Module sends the NJ-AuthN Response to the CJ-AuthN Agent.

$$AuthN_{RM} \rightarrow AuthN_{c_i}^J : \textbf{NJ-AuthN Response} \{Msg_{NJ,RM} || Mac_{NJ,RM}\}$$

where $Msg_{NJ,RM} = \{Msg_{J,AS}^1 || Mac_{J,AS}^1 || ID_{JD} || ID_{JT} || ID_{TT} || ID_{RT} || ID_{RM} || t_{RM,NJ}\}$, and $Mac_{NJ,RM} = HMAC (Msg_{NJ,RM}, SK_{c_i}^{JS})$.

214
Step11 - NJ-Response Verification: In this step, upon the receipt of the NJ-AuthN Response, the CJ-AuthN Agent validates the authenticity of both $Msg_{JAA,S}$ and $Msg_{NJ,RM}$ to confirm the identity of both the Authentication Server and the Resource Manager, respectively. To do so (i.e. satisfy (F2.3)), it does the following operations.

- The CJ-AuthN Agent reads $Msg_{JAA,S}$ and $Mac_{JAA,S}$, and it checks if the difference between the MR-Client local timestamp ($t_{ci}$) and the timestamp contained in $Msg_{JAA,S}$ ($t_{AS,JA}$) is less than a predefined value ($\Delta T$), i.e. ($t_{ci} - t_{AS,JA} < \Delta T$).
- If $t_{AS,JA}$ is valid, the CJ-AuthN Agent computes the HMAC signature of the received $Msg_{JAA,S}$ using $P_{ci}$, i.e. $Mac_{JAA,S} = HMAC(Msg_{JAA,S}, P_{ci})$, to check whether the computed HMAC signature $Mac_{JAA,S}$ equals the one received $Mac_{JAA,S}$, i.e. ($Mac_{JAA,S} = Mac_{JAA,S}$).
- If the HMAC signature is valid, the CJ-AuthN Agent decrypts the encrypted part of $Msg_{JAA,S}$ using $K_{ci}$, i.e. $E^{-1}(K_{ci}, ST_{ci}^I||SK_{ci}^I||SK_{ci}^{DL}||ID_{NN}||IP_{NN}||PU_{NN}||Authenticator_{ci}^N)$. The CJ-AuthN Agent verifies the correctness of the received authenticator by checking whether the decrypted authenticator equals the authenticator sent in the NJ-AuthN Request in Step2, i.e. if ($Authenticator_{ci}^N = Authenticator_{ci}^N$). If they match, the Authentication Server is authenticated to the MR-Client.
- The CJ-AuthN Agent then verifies the freshness of the received NJ-AuthN Response message by checking if the difference between the local timestamp of the MR-Client ($t_{ci}$) and the timestamp contained in the message ($t_{RM,NJ}$) is less than a predefined value ($\Delta T$), i.e. ($t_{ci} - t_{RM,NJ} < \Delta T$). If the message is fresh, the CJ-AuthN Agent computes the HMAC signature of the received $Msg_{NJ,RM}$ using $SK_{ci}^S$, i.e. $Mac_{NJ,RM} = HMAC(Msg_{NJ,RM}, SK_{ci}^S)$, to check whether the computed HMAC signature, i.e. $Mac_{NJ,RM}$, equals the received one, i.e. if ($Mac_{NJ,RM} = Mac_{NJ,RM}$). If the HMAC signature is valid, the Resource Manager is authenticated to the MR-Client.
7.5.3. C-NN_AuthN Protocol

The C-NN_AuthN protocol is described in details below and shown in Figure 7.11.

**Step 1 - WJ-Request Initialization:** Once a mutual authentication between the MR-Client and the Resource Manager is performed using the C-RM_AuthN protocol and the MR-
Job Components’ credentials are generated using JCPCrE protocol (details to follow), the CJ-AuthN Agent initiates the WJ-AuthN Request as following:

- The CJ-AuthN Agent identifies (i) the MR-Job ID assigned to the MR-Client, i.e. $ID_{IDJ}$, to be used along with the identity and the timestamp of the MR-Client to form his/her authenticator, (ii) the primary corresponding verification tokens of the MR-Job Components, i.e. $PVT_{s_j}$, to be used to validate the MR-Job Components’ requests identified by $PT_{s_j}$, (iii) the user password-secret key, i.e. $K_{c_i}$, to be used to encrypt both the authenticator and the $PVT_{s_j}$, and (iv) the writing job-secret key, i.e. $SK_{c_i}^{WJ}$, to be used to sign the contents of the WJ-AuthN Request message.

- It constructs the MR-Client authenticator for the WJ-AuthN Request, i.e. $Authenticator_{C_i}^{WJ} = \{ID_{C_i}||ID_{IDJ}||t_{C_i,WJ}\}$, which is verified later by the NN-AuthN Module.

- The CJ-AuthN Agent encrypts both $Authenticator_{C_i}^{WJ}$ and $PVT_{s_j}$ using $K_{c_i}$, i.e. $E(K_{c_i}, \{Authenticator_{C_i}^{WJ}||PVT_{s_j}\})$, and encrypts $SK_{c_i}^{WJ}$ using the public key of the Name Node, i.e. $PU_{NN}$, i.e. $Enc(\ PU_{NN}, SK_{c_i}^{WJ})$. Besides the confidentiality provided to the $Authenticator_{C_i}^{WJ}, PVT_{s_j}$, and $SK_{c_i}^{WJ}$, the first encryption, i.e. $E$, provides a proof of having the authentication secret $K_{c_i}$.

- The CJ-AuthN Agent constructs the WJ-AuthN Request message, i.e. $Msg_{WJ,C_i} = \{E( K_{c_i}, \{Authenticator_{C_i}^{WJ}||PVT_{s_j}\})||Enc(\ PU_{NN}, SK_{c_i}^{WJ})||ID_{C_i}||t_{C_i,WJ} \}$, where $Enc$ is asymmetric encryption function which encrypts the $SK_{c_i}^{WJ}$ using $PU_{NN}$, and $t_{C_i,WJ}$ is the current timestamp of $C_i$ which is used to resist replay attacks.

- It computes the HMAC signature of the message. In other words, it signs the message by computing the message authentication code using HMAC function, discussed in Section 5.10, i.e. $Mac_{WJ,C_i} = HMAC(Msg_{WJ,C_i},SK_{c_i}^{WJ})$ where, $Mac_{WJ,C_i}$ is the message authentication code of $Msg_{WJ,C_i}$, and HMAC is the keyed-hash function which uses $SK_{c_i}^{WJ}$ to sign the message. Besides the protection against active attacks to the message contents, the HMAC signature provides the message origin authentication using the authentication secret $SK_{c_i}^{WJ}$.
• Finally, the CJ-AuthN Agent constructs a WJ-AuthN Request object, i.e. \( \{ \text{Msg}_{WJ,C_i} || \text{Mac}_{WJ,C_i} \} \).

**Step 2 - Send WJ-AuthN Request**: In this step, the CJ-AuthN Agent sends the WJ-AuthN Request to the NN-AuthN Module.

\[
\text{AuthN}^j_{C_i} \rightarrow \text{AuthN}^j_{NN} : \text{WJ-AuthN Request} \{ \text{Msg}_{WJ,C_i} || \text{Mac}_{WJ,C_i} \}
\]

where \( \text{Msg}_{WJ,C_i} = \{ E( K_{C_i}, \{ \text{Authenticator}^W_{C_i} || \text{PVT}_{j} \}) || \text{Enc}( PU_{NN}, SK^W_{C_i} ) || ID_{C_i} || t_{C_i,WJ} \} \), and

\( \text{Mac}_{WJ,C_i} = \text{HMAC}( \text{Msg}_{WJ,C_i}, SK^W_{C_i} ) \).

**Step 3 - WJ-Request Verification**: In this step, upon the receipt of the WJ-AuthN Request, the NN-AuthN Module validates the authenticity of the request message to confirm the MR-Client identity. To do so (i.e. satisfy (F2.4)), it does the following operations.

- The NN-AuthN Module reads \( ID_{C_i}, t_{C_i,WJ} \), and the HMAC signature of the \( \text{Msg}_{WJ,C_i} \), i.e. \( \text{Mac}_{WJ,C_i} \).

- The NN-AuthN Module verifies the freshness of the received request by checking if the difference between the local timestamp of the Name Node, i.e. \( t_{NN} \), and the timestamp contained in the message, i.e. \( t_{C_i,WJ} \), is less than a predefined value \( (\Delta T) \), i.e. \( t_{NN} - t_{C_i,WJ} < \Delta T \).

- Once the freshness is valid, the NN-AuthN Module decrypts the writing job-secret key, \( SK^W_{C_i} \), using the private key of the Name Node, i.e. \( \text{Enc}^{-1}( PR_{NN}, SK^W_{C_i} ) \).

- It then uses \( SK^W_{C_i} \) to compute the HMAC signature of the received message i.e. \( \text{Mac}^{*}_{WJ,C_i} = \text{HMAC}( \text{Msg}_{WJ,C_i}, SK^W_{C_i} ) \), to check whether the computed HMAC signature, \( \text{Mac}^{*}_{WJ,C_i} \), equals the received one, i.e. if \( \text{Mac}^{*}_{WJ,C_i} = \text{Mac}_{WJ,C_i} \). If the signature is valid, the NN-AuthN Module accepts the authentication secret \( SK^W_{C_i} \). Otherwise, it rejects the request, considers it as invalid, and terminate the session.

- The NN-AuthN Module decrypts the encrypted part of \( SVT^j_{C_i} \) using \( SK^W_{C_i} \), i.e. \( E^{-1} ( SK^W_{C_i}, \{ ID_{C_i} || K_{C_i} || t_{s} || t_{n} \} ) \), and verifies the \( C_i \)’s short-term credential validity period. In other words, it verifies if the timestamp of the Name Node, when it receives the request (\( t_{NN} \)), is not before \( t_{s} \) and not after \( t_{n} \), i.e. if \( t_{s} \leq t_{NN} \leq t_{n} \).
If the result is valid, the NN-AuthN Module reads the user password-secret key, i.e. $K_{C_i}$, Otherwise, it rejects the request, considers it as invalid request, and terminate the session.

- The NN-AuthN Module decrypts the encrypted MR-Client’s authenticator in the received request using $K_{C_i}$, i.e. $E^{-1}(K_{C_i}, \{\text{Authenticator}^W_{C_i}|pvt_i\})$.

- Finally, the NN-AuthN Module constructs a new MR-Client’s authenticator, i.e. $\text{Authenticator}^W_{C_i} = \{ID_{C_i}||ID_{D_j}||t_{C_i,WJ}\}$, where $ID_{C_i}$ is read from $SVT^{1}_{C_i}$, $ID_{D_j}$ from the received Forward message in C-RM_AuthN protocol, and $t_{C_i,WJ}$ is read from the received WJ-AuthN Request. It then checks if $\text{Authenticator}^W_{C_i}$ equals the one received in the WJ-AuthN Request message, i.e. if $(\text{Authenticator}^W_{C_i} = = \text{Authenticator}^W_{C_i})$. If they match the NN-AuthN Module accepts the authentication secret $K_{C_i}$ and performs the next step. Otherwise, it rejects the request, considers it as invalid request, and terminate the session.

**Step4 - Generate a Writing-Access Authentication Information:** This function is executed by the NN-AuthN Module. Upon a positive authentication of the MR-Client with the Name Node, the NN-AuthN Module generates the writing-access authentication information of the MR-Client. It is used to constitute the secondary authentication and verification tokens that are used to provide a mutual authentication between the MR-Client and the Data Node, i.e. $ST^2_{C_i}$ and $SVT^2_{C_i}$ (satisfy (F2.9)). In details, NN-AuthN Module does the following operations:

- It generates a random secret numbers, $r_{1_{NN}}$, and specifies the timestamp when this random number is generated, i.e. $t_{r_{1_{NN}}}$. As explained in Section 7.4, $r_{1_{NN}}$ and $t_{r_{1_{NN}}}$ along with $ID_{C_i}$ and $ID_{D_j}$ are used to generate the writing access-secret key, i.e. $SK^{WA}_{C_i} = \mathcal{H}(ID_{C_i}||ID_{D_j}||r_{1_{NN}}||t_{r_{1_{NN}}})$. Both $SK^{WA}_{C_i}$ and $SK^{WJ}_{C_i}$, which are generated randomly and securely, are used to issue the $ST^2_{C_i}$ and $SVT^2_{C_i}$ (satisfy (F2.10)).

- It encrypts $SK^{WA}_{C_i}$ using $SK^{ID}_{C_i}$, i.e. $E(SK^{ID}_{C_i}, SK^{WA}_{C_i})$, and constructs $ST^2_{C_i}$, i.e. $ST^2_{C_i} = \{SK^{WA}_{C_i}||SK^{WJ}_{C_i}||E(SK^{ID}_{C_i}, SK^{WA}_{C_i})\}$
• The NN-AuthN Module encrypts \( ID_{c_i} \) and \( ID_{d_j} \) along with \( SK_{c_i}^{WJ} \) using \( SK_{c_i}^{WA} \), i.e. \( E(SK_{c_i}^{WA}, \{ID_{c_i} \parallel ID_{d_j} \parallel SK_{c_i}^{WJ}\}) \). Then NN-AuthN Module constructs \( SVT_{c_i}^2 \), i.e. \( SVT_{c_i}^2 = 1ID_{c_i} \parallel E(SK_{c_i}^{WA}, \{ID_{c_i} \parallel ID_{d_j} \parallel SK_{c_i}^{WJ}\}) \).

• After that, the NN-AuthN Module encrypts \( ST_{c_i}^2 \) along with \( ID_{DN}, IP_{DN} \) and the \( Authenticator_{c_i}^{WJ^*} \) using, \( SK_{c_i}^{WJ} \), i.e. \( E(SK_{c_i}^{WJ}, \{ST_{c_i}^2 || ID_{DN} || IP_{DN} || Authenticator_{c_i}^{WJ^*}\}) \), where \( ID_{DN} \) and \( IP_{DN} \) are the identity and the IP address of the Date Node to which the WJ-AuthN Request is redirected, and the \( Authenticator_{c_i}^{WJ^*} \) is the MR-Client’s authenticator which is received in the WJ-AuthN Request.

• The NN-AuthN Module constructs the WJ-AuthN Response message, i.e. \( Msg_{WJ,NN} = \{E(SK_{c_i}^{WJ}, \{ST_{c_i}^2 || ID_{DN} || IP_{DN} || Authenticator_{c_i}^{WJ^*}\}) \parallel ID_{NN} \parallel t_{NN,WJ}\) and computes its HMAC signature using \( K_{c_i} \), i.e. \( Mac_{WJ,NN} = HMAC(Msg_{WJ,NN}, K_{c_i}) \), where the timestamp of the Name Node, i.e. \( t_{NN,WJ} \) is used to resist replay attacks, and the HMAC signature of \( Msg_{WJ,NN} \) is used to resist any active attacks.

• It also constructs the a Notify message, i.e. \( Msg_{N,NN} = \{E(SK_{c_i}^{ID}, \{SVT_{c_i}^2\}) \parallel ID_{NN} \parallel t_{NN,N}\} \) and computes its HMAC signature using \( SK_{c_i}^{ID} \), i.e. \( Mac_{N,NN} = HMAC(Msg_{N,NN}, SK_{c_i}^{ID}) \), where \( t_{NN,N} \) is the timestamp of the Name Node when the Notify message is generated and it is used to resist replay attacks, and \( Mac_{N,NN} \) is generated to resist any active attacks.

• Finally, the NN-AuthN Module constructs both the WJ-AuthN Response object, i.e. \( \{Msg_{WJ,NN} \parallel Mac_{WJ,NN}\} \) and the Notify object, i.e. \( \{Msg_{N,NN} \parallel Mac_{N,NN}\} \).

**Step 5 - Notify:** In this step, the NN-AuthN Module sends the Notify object to the DN-AuthN Module (\( AuthN_{DN} \)).

\[
AuthN_{NN} \rightarrow AuthN_{DN} : Notify \{Msg_{N,NN} \parallel Mac_{N,NN}\}
\]

where \( Msg_{N,NN} = \{E(SK_{c_i}^{ID}, \{SVT_{c_i}^2\}) \parallel ID_{NN} \parallel t_{NN,N}\} \), and \( Mac_{N,NN} = HMAC(Msg_{N,NN}, SK_{c_i}^{ID}) \)

**Step 6 - Send WJ-AuthN Response:** The NN-AuthN Module then sends the WJ-AuthN Response to the NN-AuthN Module.

\[
AuthN_{NN} \rightarrow AuthN_{ci}^{\prime} : WJ-AuthN Response \{Msg_{WJ,NN} \parallel Sig_{WJ,NN}\}
\]
where $Msg_{WJ,NN} = \{ E(SK_{C_i}^W), \{ST_{C_i}^2||ID_{DN}||IP_{DN}||Authenticator_{C_i}^W \} ||ID_{NN}||t_{NN,WJ},$
and $Mac_{WJ,NN} = HMAC(Msg_{WJ,NN}, K_{C_i})$

**Step7 - WJ-Response Verification:** In this step, upon the receipt of the WJ-AuthN Response, the CJ-AuthN Agent validates the authenticity of $Msg_{WJ,NN}$ to confirm the identity of the Name Node. To do so (i.e. satisfy (F2.5)), it does the following operations.

- The CJ-AuthN Agent reads $ID_{NN}, t_{NN,WJ}$ and $Mac_{WJ,NN}$.
- The CJ-AuthN Agent verifies the freshness of $Msg_{WJ,NN}$ by checking if the difference between the local timestamp of the MR-Client, i.e. $t_{C_i}$, and the timestamp contained in the message, i.e. $t_{NN,WJ}$, is less than a predefined value ($\Delta T$), i.e. $(t_{C_i} - t_{NN,WJ} < \Delta T)$.
- If the $t_{NN,WJ}$ is valid, the CJ-AuthN Agent computes the HMAC signature of the received $Msg_{WJ,NN}$ using $K_{C_i}$, i.e. $Mac_{WJ,NN}^* = HMAC(Msg_{WJ,NN}^*), K_{C_i}$, to check whether $Mac_{WJ,NN}^*$ equals the received one, i.e. if $(Mac_{WJ,NN}^* == Mac_{WJ,NN})$.
- If the HMAC signature is valid, the CJ-AuthN Agent decrypts the encrypted part of $Msg_{WJ,NN}$ using $SK_{C_i}^W$, i.e. $E^{-1}(SK_{C_i}^W, \{ST_{C_i}^2||ID_{DN}||IP_{DN}||Authenticator_{C_i}^W \}^*)$.
- The CJ-AuthN Agent then verifies the correctness of the MR-Client’s authenticator by checking whether the decrypted MR-Client’s authenticator equals the one sent in the WJ-AuthN Request in Step2 - , i.e. if $(Authenticator_{C_i}^W == Authenticator_{C_i}^W)$. If the authenticator is valid, the Name Node is authenticated to the MR-Client.

**Step8 - R-WJ-Request Initialization:** Once a mutual authentication between the MR-Client and the Name Node is performed and the respective Data Node is notified about the writing request of the MR-Client, the CJ-AuthN Agent initiates the R-WJ-AuthN Request as following:

- It constructs the MR-Client authenticator for the R-WJ-AuthN Request using $ID_{C_i}, ID_{JD_i}$, and $t_{C_i},RWJ$, i.e. $Authenticator_{C_i}^{RW} = \{ ID_{C_i}||ID_{JD_i} ||t_{C_i},RWJ \}$.
- The CJ-AuthN Agent then encrypts $Authenticator_{C_i}^{RW}$ using $SK_{C_i}^W$, i.e. $E(SK_{C_i}^W, Authenticator_{C_i}^{RW})$. This encryption operation is done to protect the confidentiality of the authenticator and provides a proof of having the authentication secret $SK_{C_i}^W$.
- Once the Authenticator\textsuperscript{RWJ}\textsubscript{ci} is encrypted and the encrypted writing access-secret key is identified, i.e. $E(SK\textsuperscript{ID}_{ci}, SK\textsuperscript{WA}_{ci})$, the CJ-AuthN Agent constructs the R-WJ-AuthN Request message, i.e. $Msg_{RWJ,ci} = \{ E( SK\textsuperscript{WJ}_{ci}, Authenticator\textsuperscript{RWJ}_{ci}) || E(SK\textsuperscript{ID}_{ci}, SK\textsuperscript{WA}_{ci}) || ID_{ci} || t_{ci,RWJ} \}$, where $t_{ci,RWJ}$ is the timestamp of the MR-Client when the message is generated.

- It computes the HMAC signature of the message, i.e. $Mac_{RWJ,ci} = HMAC(Msg_{RWJ,ci}, SK\textsuperscript{WA}_{ci})$. Besides the message origin authentication provided by the HMAC signature using the authentication secret $SK\textsuperscript{WA}_{ci}$, it is used to resist any unauthorised modification of the message contents.

- Finally, the CJ-AuthN Agent constructs a R-WJ-AuthN Request object, i.e. $\{Msg_{RWJ,ci} || Mac_{RWJ,ci}\}$.

**Step 9 - Send R-WJ-AuthN Request:** In this step, the CJ-AuthN Agent sends the R-WJ-AuthN Request to the NN-AuthN Module.

$$\text{AuthN}_{ci}^J \rightarrow \text{AuthN}_{NN}^{R-WJ-AuthN Request} \{Msg_{RWJ,ci} || Mac_{RWJ,ci}\}$$

where $Msg_{RWJ,ci} = \{ E( SK\textsuperscript{WJ}_{ci}, Authenticator\textsuperscript{RWJ}_{ci}) || E(SK\textsuperscript{ID}_{ci}, SK\textsuperscript{WA}_{ci}) || ID_{ci} || t_{ci,RWJ} \}$, and $Mac_{RWJ,ci} = HMAC(Msg_{RWJ,ci}, SK\textsuperscript{WA}_{ci})$.

**Step 10 - R-WJ-Request Verification:** In this step, upon the receipt of the R-WJ-AuthN Request, the DN-AuthN Module validates the authenticity of the request message to confirm the MR-Client identity. To do so (i.e. satisfy (F2.6)), it does the following operations.

- The NN-AuthN Module reads $ID_{ci}$ and $t_{ci,RWJ}$, along with $Mac_{RWJ,ci}$ from the received request.

- The DN-AuthN Module verifies the freshness of the received request by checking if the difference between the local timestamp of the Data Node, i.e. $t_{DN}$, and the timestamp contained in the message, i.e. $t_{ci,RWJ}$, is less than a predefined value ($\Delta T$), i.e. $(t_{DN} - t_{ci,RWJ} < \Delta T)$.

- Once $t_{ci,RWJ}$ is valid, the DN-AuthN Module decrypts the encrypted writing access-secret key received in the request using $SK\textsuperscript{ID}_{ci}$, i.e. $E^{-1}(SK\textsuperscript{ID}_{ci}, SK\textsuperscript{WA}_{ci})$. 

222
• It then uses $SK_{CI}^{WA}$ to compute the HMAC signature of the received request i.e. 
$Mac_{CI}^{*} = HMAC(Msg_{RWJ,C_i}^{}, SK_{CI}^{WA})$, to check whether the computed signature equals the received one, i.e. if $(Mac_{CI}^{*} = Mac_{RWJ,C_i}^{})$. If the signature is valid, the DN-AuthN Module accepts the authentication secret $SK_{CI}^{WA}$. Otherwise, it rejects the request, considers it as invalid request, and terminate the session.

• The DN-AuthN Module decrypts the encrypted part of the $SVT_{CI}^{2}$ using $SK_{CI}^{WA}$, i.e. $E^{-1}(SK_{CI}^{WA}, [ID_{CI} || ID_{Dj} || SK_{CI}^{WJ}])$.

• The DN-AuthN Module then decrypts the encrypted MR-Client’s authenticator in the received request using $SK_{CI}^{WJ}$, i.e. $E^{-1}(SK_{CI}^{WJ}, Authenticator_{CI}^{RWJ})$.

• The DN-AuthN Module then constructs a new MR-Client’s authenticator, i.e. $Authenticator_{CI}^{RWJ^{*}} = [ID_{CI} || ID_{Dj} || t_{CI,RWJ}]$, where $ID_{CI}$ is read from $SVT_{CI}^{2}$, $ID_{Dj}$ from the received Notify message in Step5 - , and $t_{CI,RWJ}$ from the received R-WJ-AuthN Request. It then verifies if $Authenticator_{CI}^{RWJ^{*}}$ equals the one received in the R-WJ-AuthN Request message, i.e. if $(Authenticator_{CI}^{RWJ^{*}} = Authenticator_{CI}^{RWJ})$. If they match, the DN-AuthN Module accepts the authentication secret $SK_{CI}^{WJ}$. Otherwise, it rejects the request message, considers the message as invalid message, and terminate the session.

• The DN-AuthN Module encrypts $Authenticator_{CI}^{RWJ^{*}}$ using, $SK_{CI}^{WA}$ i.e. $E(SK_{CI}^{WA}, Authenticator_{CI}^{RWJ^{*}})$.

• The DN-AuthN Module constructs the R-WJ-AuthN Response message, i.e. 
$Msg_{RWJ,DN} = E(SK_{CI}^{WA}, Authenticator_{CI}^{RWJ^{*}}) || t_{DN,RWJ} || ID_{DN}$, and computes the HMAC signature of the message using $SK_{CI}^{WJ}$, i.e. $Mac_{RWJ,DN} = HMAC(Msg_{RWJ,DN}, SK_{CI}^{WJ})$.

Finally, the DN-AuthN Module constructs the R-WJ-AuthN Response object, i.e. $\{Msg_{RWJ,DN} || Mac_{RWJ,DN}\}$.

**Step11 - Send R-WJ-AuthN Response:** In this step, the DN-AuthN Module sends the R-WJ-AuthN Response to the CJ-AuthN Agent.

$AuthN_{DN} \rightarrow AuthN_{CI}^{I} : R-WJ-AuthN Response \{Msg_{RWJ,DN} || Mac_{RWJ,DN}\}$
Where \( \text{Msg}_{\text{RWJ, DN}} = \{E(SK_{C_i}^W, \{\text{Authenticator}_{C_i}^{R W J}^*\}) \| t_{DN,RWJ} \| ID_{DN} \} \), and
\( \text{Mac}_{\text{RWJ, DN}} = \text{HMAC}(\text{Msg}_{\text{RWJ, DN}}, SK_{C_i}^W) \)

**Step 12 - R-WJ-Response Verification and MR-Client Authentication Credential Protection:** In this step, upon the receipt of the R-WJ-AuthN Response, the CJ-AuthN Agent validates the authenticity of \( \text{Msg}_{\text{RWJ, DN}} \) to confirm the identity of the Data Node (i.e. satisfy (F2.7)), and it protects the local and removable parts of the MR-Client authentication credential. To do so, it does the following operations.

- The CJ-AuthN Agent reads \( ID_{DN} \) and \( t_{DN,RWJ} \) along with \( \text{Mac}_{\text{RWJ, DN}} \) from the received response.
- The CJ-AuthN Agent verifies the freshness of \( \text{Msg}_{\text{RWJ, DN}} \) by checking if the difference between the MR-Client’s local timestamp, i.e. \( t_{C_i} \), and the timestamp contained in the message, i.e. \( t_{DN,RWJ} \), is less than a predefined value (\( \Delta T \)), i.e. \((t_{C_i} - t_{DN,RWJ} < \Delta T)\).
- If the \( t_{DN,RWJ} \) is valid, the CJ-AuthN Agent computes the HMAC signature of \( \text{Msg}_{\text{RWJ, DN}} \) using \( SK_{C_i}^W \), i.e. \( \text{Mac}_{\text{RWJ, DN}}^* = \text{HMAC}(\text{Msg}_{\text{RWJ, DN}}, SK_{C_i}^W) \), to check whether the computed HMAC signature, \( \text{Mac}_{\text{RWJ, DN}}^* \), equals the received one, i.e. if \((\text{Mac}_{\text{RWJ, DN}}^* == \text{Mac}_{\text{RWJ, DN}})\).
- If the HMAC signature is valid, the CJ-AuthN Agent decrypts the encrypted \( \text{Authenticator}_{C_i}^{R W J}^* \) using \( SK_{C_i}^W \), i.e. \( E^{-1}(SK_{C_i}^W, \{\text{Authenticator}_{C_i}^{R W J}^*\}) \).
- The CJ-AuthN Agent then verifies the correctness of the authenticator by checking whether the decrypted one equals the authenticator sent in the R-WJ-AuthN Request in Step 9, i.e. if \((\text{Authenticator}_{C_i}^{R W J}^* == \text{Authenticator}_{C_i}^{R W J})\) If the authenticator is valid, the Data Node is authenticated to the MR-Client.
- The CJ-AuthN Agent then protects the local and removable parts of the MR-Client authentication credential as following:
  - It generates a new login authentication token for the next login, \( \text{LAT}_{U_i} \), by computing \( \text{LAT}_{U_i} = \text{H}(r_{C_i}' \| t_{r_{C_i}'} \| SK_{C_i}^{DL'}) \), where \( r_{C_i}' \) is a new random number generated, \( t_{r_{C_i}'} \) is the timestamp when \( r_{C_i}' \) is generated, and \( SK_{C_i}^{DL'} \) is the new dynamic login-secret key received from the G-AuthN Service via the C-RM_AuthN protocol.
The CJ-AuthN Agent encrypts the removable part of the MR-Client credential using the new dynamic login-secret key, $SK_{DL_i}^{i'}$, i.e. $E(SK_{DL_i}^{i'}, R_{ACR_{Ci}}^{i'})$, where $R_{ACR_{Ci}}^{i'} = \{LAT_{U_i}^{i'} \parallel PT_2^{i'}\}$, and $PT_2^{i'} = PT_1^{i'} || SK_{Ci}^{JS}$. It then stores the ciphertext, i.e. $E(SK_{DL_i}^{i'}, R_{ACR_{Ci}}^{i'})$, into the removable storage media, i.e. $USB_{U_i}$, which is possessed by the user.

$AuthN_{Ci}^{i} \rightarrow USB_{U_i} : \{E(SK_{DL_i}^{i'}, R_{ACR_{Ci}}^{i'})\}$

Finally, the CJ-AuthN Agent encrypts the local part of the MR-Client credential using the login password-secret key $SK_{LP_i}^{i'}$, i.e. $E(SK_{LP_i}^{i'}, L_{ACR_{Ci}}^{i'})$, and it embeds the ciphertext, i.e. $E(SK_{LP_i}^{i'}, L_{ACR_{Ci}}^{i'})$ to a local stego-file, i.e. Stego-File{$E(SK_{LP_i}^{i'}, L_{ACR_{Ci}}^{i'})$}, where $L_{ACR_{Ci}}^{i'} = LVT_{U_i}^{i'} || SK_{DL_i}^{i'}$, and $LVT_{U_i}^{i'}$ is a new local verification token which is used along with the new login authentication token, i.e. $LAT_{U_i}^{i'}$, generated before, for the next login. The encryption and steganography operations are used to protect $L_{ACR_{Ci}}^{i'}$ and $R_{ACR_{Ci}}^{i'}$ against unauthorised use especially if $CH_{U_i}$ is compromised.
Figure 7.11 C-NN_AuthN protocol in detail
7.6. JCPCrE Protocol Design for MR-Job Components Identification: an Overview

This section presents an overview of the JCPCrE protocol for identifying a new set of MR-Job Components to an MR-Job domain. Firstly before describing how the protocol identifies the MR-Job Components, we motivate the need for the JCPCrE protocol by highlighting the limitations of the current methods in identifying new MR components to be assigned to a client job, i.e. MR-Job Components.

Most of the current methods do not consider MR-Job Components identification. This means that these methods are unable to identify an impersonator of an MR-Job Component. In other words, implementing these methods allows access to different job resources without any MR-Job Components authentication, so, it is very easy to access all the data of any job hosted on the shared MR infrastructure. When MR components (MR-Inf. or MR-Job Components) communicate with each other, they don’t verify that the other MR component is really what it claims to be. As a result, it is feasible to start a rogue (or impersonating an) MR-Job Component such as a Task Tracker to get access to the data of different jobs or to disrupt other jobs. Though there are efforts on supporting MR-Job Components identification, these efforts are largely based on the use of public key credentials, where the public key (i.e. asymmetric key) based solution requires the involvement of a third party (e.g. CA) for credential issuance and distribution. The costs incurred in such solution are typically high in comparison to the symmetric key based solution.

Further, as the author writing this thesis, none of the current authentication methods identifies the MR-Job Components with multi-factor authentication secrets. In other words, in those methods which support the MR-Job Components identification, the authentication credential of an MR-Job Component is issued and based on single-factor authentication secret to authenticate the MR-Job Component to the MR-Inf. Component. Multi-factor authentication is preferred as it provides a higher level of security.

Taking into account the characteristics of the MR application which include (i) there are multiple interactions sprout out from a single job execution by different MR components, (ii) the MR-Job Components that are invoked to process the job submitted by an MR-Client are hosted on shared MR server nodes, (iii) these shared MR server nodes are different from those which the MR-Client has logged into and submits his job to, and (iv) once the job is successfully submitted, these MR-Job Components will execute the job and access the MR-
Client’s data on his behalf through the MR-Job execution, the challenges are, (1) how we could identify a set of MR-Job Components securely with multi-factor authentication using a symmetric key based authentication method, and (2) how we could issue and securely distribute the multi-factor authentication secrets to this set of MR-Job Components in such environment. Therefore, to overcome these limitations and challenges we propose the new JCPCrE protocol to identify this set of MR-Job Components to the MR-Client’s job domain, i.e. MR-Job domain, securely and efficiently.

With JCPCrE protocol, the VDAF design requirement F3 is achieved. By executing this protocol, (i) MR-Job Components are identified (by the job owner) by issuing the primary authentication and verification tokens of the Job Tracker ($PT_{JT}^j$ and $PV'T_{JT}^j$), the set of all Task Trackers ($PT_{TT}^j$ and $PV'T_{TT}^j$), and the set of all Reduce Tasks ($PT_{RT}^j$ and $PV'T_{RT}^j$) of the MR-Job domain, and (ii) these tokens to be submitted to the MR application and distributed to the respective MR components upon a positive authentication of the MR-Client with the Name Node. Note that the JCPCrE protocol works in conjunction with the C-NN_AuthN protocol, where the $PV'T_{JT}^j$, $PV'T_{TT}^j$ and $PV'T_{RT}^j$ are delivered to the Name Node (i.e. verifier) using the WJ-AuthN Request message of the C-NN_AuthN protocol, upon the first step of the JCPCrE protocol is completed (details to follow).

An overview of the JCPCrE protocol is depicted in Figure 7.12. As it can be seen from the figure, the protocol is executed for every MR-Job domain, where the CJ-AUTHN Agent, RM-AUTHN Relay Module, JT-AUTHN Module, and TT-AUTHN Module are involved. The protocol consists of four steps, each of them performs a specific function as following:

**Step1 - Generate an MR-Job Authentication Information:** once (i) a successful mutual authentication between the MR-Client and the Resource Manager is achieved and (ii) the MR-Client received the writing job-authentication information during the C-RM/AuthN protocol, the MR-Client is authorised to issue $PT_{^j}$ and $PV'T_{^j}$. The CJ-AUTHN Agent generates the MR-Job authentication information. The MR-Job authentication information includes three role based-secret keys; a retrieving input-split-secret key ($SK^R_{JT}^j$) for a Job Tracker, a retrieving origin-data-secret key ($SK^R_{TT}^j$) for a set of all Task Trackers, and a writing final-data-secret key ($SK^WF_{RT}^j$) for a set of all Reduce Tasks of the MR-Job domain. The CJ-AUTHN Agent then uses these authentication information (i.e. writing-job and MR-Job authentication information) to constitute the primary
authentication and verification tokens of the Job Tracker \((PT_{JT}^j\) and \(PVT_{JT}^j\)), the set of all Task Trackers \((PT_{TT}^j\) and \(PVT_{TT}^j\)), and the set of all Reduce Tasks \((PT_{RT}^j\) and \(PVT_{RT}^j\)) of the MR-Job domain, i.e. it issues \(PT_{*}^j = \{ PT_{JT}^j, PT_{TT}^j, PT_{RT}^j \} \) and \(PVT_{*}^j = \{ PVT_{JT}^j, PVT_{TT}^j, PVT_{RT}^j \} \).

**Step2 - Submit:** During this step, the CJ-AuthN Agent submits \(PT_{*}^j\) to the RM-AuthN Relay Module using a Submit message. \(PT_{*}^j\) is encrypted and signed along with the identity and the timestamp of the MR-Client using the job submission-secret key secret key, \(SK_{CS}^C\), which is received in the NJ-AuthN Response message (see C-RM_AuthN protocol).

**Step3 - Allocate:** In this step, the RM-AuthN Relay Module verifies the correctness of the received Submit message. The RM-AuthN Relay then decrypts the encrypted \(PT_{*}^j\) and relays it to the allocated Job Tracker using an Allocate message. In the Allocate message, \(PT_{*}^j\) is encrypted and signed along with the identity and the timestamp of the Resource Manager using the infrastructure domain-secret key \(SK_{ID}^I\).

**Step4 - Assign:** In this step, the JT-AuthN Module verifies the correctness of the received Allocate message. The JT-AuthN Module then decrypts the encrypted \(PT_{*}^j\) and sends \(PT_{TT}^j\) and \(PT_{RT}^j\) to the assigned Task Tracker using the Assign message. In the Assign message, \(PT_{TT}^j\) and \(PT_{RT}^j\) are encrypted and signed along with the identity and the timestamp of the Job Tracker using \(SK_{ID}^J\).

![Figure 7.12 An Overview of JCPCrE protocol](image-url)
7.7. MR-Job Component’s Primary Tokens and Key Structure Design

This section describes the design of the primary authentication and verification tokens of an MR-Job Component and their key structure in details.

7.7.1. Primary Authentication and Verification Tokens

As discussed in Section 5.5, each MR-Job Component is issued with an MR-Job Component authentication credential, i.e. $ACR_{JC_j}$, which consists of two tokens, primary and secondary tokens, i.e. $PT_{JC_j}$ and $ST_{JC_j}$. For each of these tokens there is a corresponding verification token, i.e. $PVT_{JC_j}$ and $SVT_{JC_j}$. This section describes the primary authentication and verification tokens which are issued using the JCPCrE protocol and used to identify the MR-Job Component to an MR-Job domain.

- MR-Job Component’s Primary Authentication and Verification Tokens ($PT_{JC_j}$ and $PVT_{JC_j}$): Figure 7.13 shows the design of both $PT_{JC_j}$ and $PVT_{JC_j}$. As shown in the figure, these tokens consist of two basic attributes and two key attributes. The basic attributes are called role ID attribute and MR-Job ID attribute, which are defined by the Resource Manager and used to identify the MR-Job Component and the MR-Job domain, respectively. The two key attributes are a role based-key attribute and a job domain-key attribute. The value of the role based-key attribute is defined when the MR-Client is approved to issue $PT_{JC_j}$ and $PVT_{JC_j}$, and the job domain-key attribute is defined when the MR-Client is authenticated to the Authentication Server. In the design of $PT_{JC_j}$, the role based-key attribute is approved using the writing job-secret key, while in $PVT_{JC_j}$, the job domain-key attribute is approved using the role based-secret key. Table 7.3 describes the values of these attributes. The values of these attributes are chosen pragmatically and incorporated technically in the authentication and verification tokens to form two-factor authentication secrets and provide a mutual authentication between the respective MR-Job Component and the Name Node which is discussed in Section 5.5.
Figure 7.13 The data structure of the primary authentication and verification tokens of an MR-Job Component

Table 7.3 The contents of the primary authentication and verification tokens of an MR-Job Component

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role ID attribute</td>
<td>Role ID attribute specifies the role of an MR-Job Component assigned to an MR-Job domain, and this role is defined based on the type of the request initiated by the MR-Job Component. In other words, this attribute specifies the purpose of the authentication request of the MR-Job Component. As discussed in Section 5.8, there are three authentication requests; the RIS-AuthN Request for the Job Tracker (IDJT), ROD-AuthN Request for the Task Tracker (IDTT), and WFD-AuthN Request for the Reduce Task (IDRT). It is the same attribute applied for both PTJCj and PVTJCj.</td>
</tr>
<tr>
<td>MR-Job ID attribute</td>
<td>MR-Job ID attribute specifies the MR-Job domain identifier of an MR-Client. It is applied for both PTJCj and PVTJCj.</td>
</tr>
<tr>
<td>Job domain-key attribute</td>
<td>Job domain-key attribute is a secret attribute. It contains a job domain-secret key. This attribute is applied for both PTJCj and PVTJCj. It is issued by the G-AuthN Service through a positive authentication of the MR-Client with the AS, and approved by the MR-Client using the role based-secret key. It is issued and approved to be used as a first-factor authentication secret to authenticate the MR-Job Component to the Name Node.</td>
</tr>
<tr>
<td>Role based-key attribute</td>
<td>Role based-key attribute is also a secret attribute which contains a role based-secret key. This attribute is applied for PTJCj. It is issued by the CJ-AuthN Agent, and approved by the AS using the writing job-secret key which is issued using C-RM_AuthN protocol. It is issued and approved to be used by the respective MR-Job Component as a second-factor authentication secret to authenticate the MR-Job Component to the Name Node.</td>
</tr>
</tbody>
</table>
7.7.2. Key Structure

This section explains the key generation and structure of the values of the attributes discussed above. As mentioned before in Section 7.6, upon a successful mutual authentication between the MR-Client and the Resource Manager, a variety of cryptographic secret keys are issued to form the primary authentication and verification tokens of the MR-Job Components, i.e. \( PT_{JC_j} \) and \( PVT_{JC_j} \). Figure 7.14 shows the key structure used for the authentication and verification tokens of an MR-Job Component. As can be seen from the figure, two independent secret keys, the job domain-secret key \( SK_{JCD} \) and the role based-secret key \(^{17} SK_{RBCJ} \) are used. These keys are generated randomly using a set of random numbers and timestamps, and they are supplied by different MR-AuthN Components.

The key structure of \( PT_{JC_j} \) consists of \( SK_{RBCJ} \) and \( SK_{JCD} \). \( SK_{RBCJ} \) is generated from the user password-secret key, \( K_{Ci} \), a random number generated by the CJ-AuthN Agent, \( r_{ci} \), and the timestamp when \( r_{ci} \) is generated. \( SK_{RBCJ} \) is encrypted using the writing job-secret key, i.e. \( SK_{WCJ} \), to protect the confidentiality of the key and provide an evidence to the verifier (i.e. the Name Node) that the key is approved by the AS when a RB-AuthN Request is sent to the Name Node by the MR-Job Component. \( SK_{WCJ} \) is used for such protection and proof. This is because it is generated by the AS and only known to the job owner and the verifier. \( SK_{JCD} \) is generated from the client master-secret key, \( P_{Ci} \), a random number \( r_{AS} \), and the timestamp when \( r_{AS} \) is generated, all of which are generated by the G-AuthN Service. As mentioned before, \( SK_{JCD} \) is issued by the Authentication Server and sent to the MR-Client so that the MR-Client can use it to issue these tokens, i.e. \( PT_{JC_j} \) and \( PVT_{JC_j} \). Similar to \( PT_{JC_j} \), the key structure of \( PVT_{JC_j} \) consists of both \( SK_{RBCJ} \) and \( SK_{JCD} \), whereas \( SK_{JCD} \) is encrypted along with other authentication information using \( SK_{RBCJ} \) so that any role-based authentication request initiated by an MR-Job Component and identified using \( PT_{JC_j} \) can be verified against \( PVT_{JC_j} \).

---

\(^{17}\) The role based-secret key, i.e. \( SK_{RBCJ} \), could be a retrieving input-split-secret key for a Job Tracker \( (SK_{RT_j}^{RI}) \), a retrieving origin-data-secret key for a set of all Task Trackers \( (SK_{TT_j}^{RO}) \), and a writing final-data-secret key for a set of all Reduce Tasks \( (SK_{RT_j}^{WF}) \) of an MR-Job Domain.
Figure 7.14 Key generation and structure of the primary authentication and verification tokens of an MR-Job Component.

The design of these tokens and their key structure has the following three-folds: First, \( PT_{JC_i} \) and \( PVT_{JC_i} \) are generated for a short-term use through a job execution. Second, the \( PT_{JC_i} \) and \( PVT_{JC_i} \) are designed in such a way that gives the Name Node (i.e. the verifier) (1) the ability to verify the authentication request of an MR-Job Component of an MR-Job domain, (2) an assurance that the secrets of the verification token is confidentiality protected. Third, the keys are generated automatically without any user’s involvement once the MR-Client is authenticated to the Resource Manager. Third, two independent secret keys, i.e. \( SK_{JC_i}^{RB} \) and \( SK_{JC_i}^{ID} \), are (i) generated, in a secure random-basis, from two secret keys which represents what the MR-Client knows and possess, and (ii) supplied by two separate resources (MR-Client and AS), so that these keys are used as two-factor authentication secrets to authenticate an MR-Job Component of an MR-Job domain to the Name Node.

7.8. JCPCrE Protocol in Details

The JCPCrE protocol is described in details below and shown in Figure 7.15

Step1 - Generate an MR-Job Authentication Information: This function is executed by the CJ-AuthN Agent to generate an MR-Job authentication information of the MR-
Client’s domain. It constitutes the primary authentication and verification tokens that are used to authenticate the MR-Job Components (to be assigned to the MR-Client’s job domain) to the Name Node, $PT_{*i}$ and $PVT_{*i}$, (i.e. satisfy F3.1). In details, the CJ-AuthN Agent does the following operations:

- It generates three random secret numbers, i.e. $r_{2c_i}, r_{3c_i}$, and $r_{4c_i}$, and specifies the timestamps when these random numbers are generated, $t_{r_{2c_i}}, t_{r_{3c_i}}$ and $t_{r_{4c_i}}$, respectively. As explained in Section 7.7, these items along with $K_{C_i}$ are used to generate the three role-based-secret keys, i.e. $SK_{JT}^{RI}, SK_{TT}^{RO}$, and $SK_{RT}^{WF}$.

$$SK_{JT}^{RI} = \mathbb{H}(K_{C_i} || r_{2c_i} \ || t_{r_{2c_i}})$$

$$SK_{TT}^{RO} = \mathbb{H}(K_{C_i} || r_{3c_i} \ || t_{r_{3c_i}})$$

$$SK_{RT}^{WF} = \mathbb{H}(K_{C_i} || r_{4c_i} \ || t_{r_{4c_i}})$$

These three keys along with $SK_{C_i}^{WJ}$ and $SK_{C_i}^{ID}$, which are generated randomly and securely, are used to issue $PT_{*i}$ and $PVT_{*i}$ (satisfy (F3.2)).

- It encrypts each of the role based-secret key using the writing job-secret key, i.e. $E(SK_{C_i}^{WJ}, SK_{JT}^{RI}), E(SK_{C_i}^{WJ}, SK_{TT}^{RO})$ and $E(SK_{C_i}^{WJ}, SK_{RT}^{WF})$.

- It constructs the primary authentication token of the Job Tracker, the set of all Task Trackers, the set of all Reduce Tasks of an MR-Job domain that belongs to an MR-Client as following:

$$PT_{JT}^j = \{ID_{JT}||ID_{JD}||SK_{JT}^{RI}||SK_{C_i}^{JD}||E(SK_{C_i}^{WJ}, SK_{JT}^{RI})\}$$

$$PT_{TT}^j = \{ID_{TT}||ID_{JD}||SK_{TT}^{RO}||SK_{C_i}^{JD}||E(SK_{C_i}^{WJ}, SK_{TT}^{RO})\}$$

$$PT_{RT}^j = \{ID_{RT}||ID_{JD}||SK_{RT}^{WF}||SK_{C_i}^{JD}||E(SK_{C_i}^{WJ}, SK_{RT}^{WF})\}$$

- Once $PT_{JT}^j$, $PT_{TT}^j$, and $PT_{RT}^j$ are issued, the CJ-AuthN Agent encrypts the role ID, the MR-Job ID and the $SK_{C_i}^{JD}$ using the respective role-based-secret key, to construct the corresponding verification tokens of the Job Tracker, the set of all Task Tracker,
and the set of all Reduce Tasks of the MR-Job domain, i.e. $PVT_{JT_j}$, $PVT_{TT_j}$, and $PVT_{RT_j}$, as following:

$$PVT_{JT_j} = \{ ID_{JT_j} || ID_{JD_j} || E(SK_{JT_j}^{RI}, \{ ID_{JT_j} || ID_{JD_j} || SK_{C_i}^{ID} \}) \}$$

$$PVT_{TT_j} = \{ ID_{TT_j} || ID_{JD_j} || E(SK_{TT_j}^{RO}, \{ ID_{TT_j} || ID_{JD_j} || SK_{C_i}^{ID} \}) \}$$

$$PVT_{RT_j} = \{ ID_{RT_j} || ID_{JD_j} || E(SK_{RT_j}^{WF}, \{ ID_{RT_j} || ID_{JD_j} || SK_{C_i}^{ID} \}) \}$$

- Finally, it constructs $PT_{*j} = \{ PT_{JT_j} || PT_{TT_j} || PT_{RT_j} \}$ and $PVTS_{*j} = \{ PVT_{JT_j} || PVT_{TT_j} || PVT_{RT_j} \}$.

**Step 2 - Submit:** During this step, the CJ-AuthN Agent submits $PT_{*j}$ to the RM-AuthN Relay Module as following:

- Once $PT_{*j}$ is generated, the CJ-AuthN Agent constructs the Submit message, i.e. $Msg_{S,C_i} = \{ E(SK_{C_i}^{JS}, PT_{*j}) || ID_{C_i} || t_{c_i} \}$ and computes the HMAC signature of the message using $SK_{C_i}^{JS}$, i.e. $Mac_{S,C_i} = HMAC (Msg_{S,C_i}, SK_{C_i}^{JS})$, where the encryption, i.e. $E$, and the HMAC operations are used to protect the confidentiality of $PT_{*j}$ and resist any forgery or unauthorised modification of the message content (i.e. active attacks), respectively.

- The CJ-AuthN Agent then constructs the Submit object, i.e. $\{ Msg_{S,C_i}, Mac_{S,C_i} \}$.

- Finally the CJ-AuthN Agent ($AuthN_{C_i}^L$) sends the Submit object to the RM-AuthN Relay Module ($AuthN_{RM}$).

$$AuthN_{C_i}^L \rightarrow AuthN_{RM} \; : \; Submit \{ Msg_{S,C_i} || Mac_{S,C_i} \}$$

**Step 3 - Allocate:** During this step, when the Submit is received, the RM-AuthN Relay Module verifies its correctness and allocate $PT_{*j}$ to the respective Job Tracker as following:

- The RM-AuthN Relay Module reads the identity and the timestamp of the MR-Client, i.e. $ID_{C_i}$ and $t_{C_i,S}$, along with the HMAC signature of $Msg_{S,C_i}$, i.e. $Mac_{S,C_i}$.  

235
It checks if the difference between the local timestamp of the Resource Manager, i.e. $t_{RM}$, and the timestamp contained in the message, i.e. $t_{C_i,S}$, is less than a predefined value ($\Delta T$), i.e. $(t_{RM} - t_{C_i,S} < \Delta T)$.

If the message is fresh, the RM-AuthN Relay Module computes the HMAC signature of the received message using $SK_{C_i}^{IS}$, i.e. $Mac_{C_i}^{*} = HMAC (Msg_{S,C_i}, SK_{C_i}^{IS})$, to check whether the computed HMAC signature, $Mac_{C_i}^{*}$, equals the received one, i.e. if $(Mac_{C_i}^{*} == Mac_{S,C_i})$.

Once the verification is positive, the RM-AuthN Relay Module then decrypts $PT_{i_j}$ using $SK_{C_i}^{IS}$.

After that the RM-AuthN Relay Module constructs the Allocate message, i.e. $Msg_{A,RM} = \{E((SK_{ID}^{ID}, PT_{i_j})||ID_{RM}||t_{RM,A})$ and computes the HMAC signature of the message using $SK_{ID}^{ID}$, i.e. $Mac_{A,RM} = HMAC (Msg_{A,RM}, SK_{ID}^{ID})$.

The RM-AuthN Relay Module then constructs the Allocate object, i.e. $\{Msg_{A,RM} || Mac_{A,RM}\}$.

Finally the RM-AuthN Relay Module sends the Allocate object to the JT-AuthN Module ($AuthN_{JT_i}$).

$$AuthN_{RM} \rightarrow AuthN_{JT_i} : Allocate \{Msg_{A,RM} || Mac_{A,RM}\}$$

**Step4 - Assign:** During this step, when the Allocate is received, the JT-AuthN Module verifies its correctness and assigns $PT_{TT_i}$ and $PT_{RT_i}$ to the respective Task Tracker as following:

- The JT-AuthN Module reads $ID_{RM}, t_{RM},$ and the HMAC Signature of $Msg_{A,RM}$, i.e. $Mac_{A,RM}$.
- It checks if the difference between the local timestamp of the Job Tracker, $t_{JT_i}$, and the timestamp contained in the message, i.e. $t_{RM,A}$, is less than a predefined value ($\Delta T$), i.e. $(t_{JT_i} - t_{RM,A} < \Delta T)$.
- If the message is fresh, the JT-AuthN Module computes the HMAC signature of the received message using $SK_{ID}^{ID}$, i.e. $Mac_{A,RM}^{*} = HMAC (Msg_{A,RM}, SK_{ID}^{ID})$, to check
whether the computed HMAC signature, $Mac_{A, RM}^*$ equals the received one, i.e. if
$(Mac_{A, RM}^* = = Mac_{A, RM})$.

- Once the verification is positive, the JT-AuthN Module decrypts $PT_{ij}$ using $SK_{ID}^*$
  and reads $PT_{JTj}^*$.

- After that the JT-AuthN Module constructs the Assign message, i.e. $Msg_{A,JTj}^*$ =
  $\{E((SK_{ID}^*, \{PT_{TTj}^||PT_{RTj}^\})||ID_{JTj}||t_{JTj,A})\}$ and computes the HMAC signature of the
  message using $SK_{ID}^*$ , i.e. $Mac_{A,JTj}^* = HMAC (Msg_{A,JTj}^*, SK_{ID}^*)$.

- The JT-AuthN Module then constructs the Assign object, i.e. $\{Msg_{A,JTj}^* || Mac_{A,JTj}^*\}$.

- The JT-AuthN Module sends the Assign object to the TT-AuthN Module ($AuthN_{TTj}$).

  $(AuthN_{JTj}^* \to AuthN_{TTj}^* : Assign \{Msg_{A,JTj}^* || Mac_{A,JTj}^*\})$

- Finally, when the Assign is received, the TT-AuthN Module verifies its correctness
  and reads $PT_{TTj}^*$ and $PT_{RTj}^*$ as following:
  
  o The TT-AuthN Module reads $ID_{JTj}$ and $t_{JTj,A}$ , along with $Mac_{A,JTj}^*$ from
    the received message.
  o It checks if the difference between the local timestamp of the Task Tracker,
    $t_{TTj}^*$, and the timestamp contained in the message, i.e. $t_{JTj,A}$, is less than a
    predefined value ($\Delta T$), i.e. $(t_{TTj}^* - t_{JTj,A} < \Delta T)$.
  o If the message is fresh, the TT-AuthN Module computes the HMAC
    signature of the received message using $SK_{ID}^*$, i.e. $Mac_{A,JTj}^* = HMAC (Msg_{A,JTj}^*, SK_{ID}^*)$, to check whether the computed HMAC
    signature, $Mac_{A,JTj}^*$, equals the received one, i.e. if $(Mac_{A,JTj}^* = = Mac_{A,JTj}^*)$. 

237
The TT-AuthN Module then decrypts and reads \( PT_{TT_j} \) and \( PT_{RT_j} \) from the received message.

Figure 7.15 JCPCrE protocol in detail

### 7.9 Security Analysis and Verification

This section gives an informal analysis of the MR-CAuthN and JCPCrE protocols against the security requirements specified in Section 7.2.1.2. To provide further proof of the security
of the protocols, we have modelled and performed a formal verification of the protocols using the Automated Validation of Internet Security Protocols and Applications tool (AVISPA).

7.9.1. Informal Analysis

The propositions below capture the security requirements of the MR-CAuthN and JCPCrE protocols specified in Section 7.2.1.2. This section explains why these protocols can achieve a secure user login, mutual authentication, message confidentiality and authenticity, and withstand known attack, such as those discussed in Section 5.6.1, that may be mounted against their functionality.

**Proposition 1:** The U-AC_AuthN protocol can support more secure user login with two-factor authentication secrets (satisfy (S1))

The U-AC_AuthN protocol provides a more secure user to MR Application Client login using two-factor authentication secrets. In other words, an adversary could not impersonate a legitimate user to the MR Application Client when his/her login credential (i.e. the username and password pair) is compromised. In our U-AC_AuthN protocol, two independent authentication factors are used. The first one is the user authentication credential \(ACR_{U_i}\). \(ACR_{U_i}\) consists of the MR-Client username \((ID_{C_i})\) and the user password \((PW_{U_i})\) which represents something the user knows. The second one is the login authentication token, i.e. \(LAT_{U_i}\), which is generated from \(SK_{C_i}^{DL}\), and represents the second authentication factor – something the user possesses. The user identifies himself/herself with his/her username and password to the C-AuthN Agent. The C-AuthN Agent validates the credential when both the local and removable parts of the MR-Client authentication credential are supplied, \(L_{ACR_{C_i}}\) and \(R_{ACR_{C_i}}\).

According to our scheme mentioned before in Sections 7.3.1 and 7.5.1, only the genuine user who knows and possesses the authentication secrets can successfully bring the factors – \(PW_{U_i}\) and \(LAT_{U_i}\). The user inserts the encrypted \(R_{ACR_{C_i}}\) (i.e. \(E(SK_{C_i}^{DL}, R_{ACR_{C_i}})\)) to the \(CH_{U_i}\) in which the encrypted \(L_{ACR_{C_i}}\) (i.e. \(E(SK_{C_i}^{LP}, L_{ACR_{C_i}})\)) is located and the C-AuthN Agent runs. The C-AuthN Agent validates the login password-secret key \((SK_{C_i}^{LP})\) and the login authentication token \(LAT_{U_i}\) against \(PVT_{C_i}^{1}\) which contains the local verification token of the user, i.e. \(LVT_{U_i}\). As \(L_{ACR_{C_i}}\) contains \(LVT_{U_i}|| SK_{C_i}^{DL}\) and \(R_{ACR_{C_i}}\) contains \(LAT_{U_i}|| PT_{C_i}^{2}\), the C-AuthN Agent reads (i) \(LVT_{U_i}\) from \(L_{ACR_{C_i}}\) which must be
decrypted using $SK_{C_i}^{LP}$, and $SK_{C_i}^{LP}$ is generated using $PW_{U_i}$ that is only known to the user, i.e. $SK_{C_i}^{LP} = \mathbb{H}(ID_{C_i} || PW_{U_i})$, and (ii) $LAT_{U_i}$ from the $R_{ACR_{C_i}}$ which is possessed by the user in his removable storage media, i.e. $USB_{U_i}$, and it must be decrypted using the dynamic login-secret key $SK_{C_i}^{DL}$. $SK_{C_i}^{DL}$ is generated by the Authentication Server and possessed by the user using the CPCrE protocol. If $LVT_{U_i}$ is valid, the user is logged in to the MR Application Client successfully and the CJ-AuthN Agent is invoked to proceed with the C-RM_AuthN protocol. This scheme supports MR-Client authentication with a local verification of user’s login credential to access MR Application Client before sending any authentication request to the MR application (satisfy (S1)). The local verification ensures (i) the existence of (i.e. the availability of) the three objects, $U_i$, $CH_{U_i}$, and $USB_{U_i}$ and (ii) both the correctness and origin of $L_{ACR_{C_i}}$ and $R_{ACR_{C_i}}$. This security property protect the MR-Client from being deceived to send multiple NJ-AuthN Request identified using invalid authentication token (i.e. $PT_{C_i}^2$), sending such requests indelibly by a genuine MR-Client could cause a DoS attack on his own account. Similarly, the local verification process could mitigate the denial of service attack, launched by an attacker on the user account when it uses online password guessing attacks, i.e. dictionary or brute-force attack (details to follow in Propositions 2).

To impersonate a user, the attacker has not only to know the user password, but the attacker must (i) possess the encrypted $R_{ACR_{C_i}}$, which is stored in $USB_{U_i}$, and (ii) gain access to the $CH_{U_i}$ and retrieve the encrypted $L_{ACR_{C_i}}$. Therefore, our U-AC_AuthN protocol achieves more secure user login service than the one-factor authentication password-based solution.

**Proposition 2:** The CPCrE and U-AC_AuthN protocols can mitigate password guessing and DoS attacks

Two mode of password guessing attacks can be used, online and offline. Broadly, an online password guessing attacks defined as when an intruder (could be a curious user or an attacker) is allowed to generate fake messages and to supply them to an honest authentication component, for checking whether a certain guess is correct. Online password guessing attack can be launched against the authentication of MR application by anyone (e.g. a curious client) using an MR Application Client, once he has access to the client machine at any time (or from other user machine in case of external client). Or an attacker could not capture the network traffic between the client and the MR application, but
somehow he was able to compromise the client machine and he would like to follow the authentication protocol specification to launch an online password guessing attacks. The motivation of the online password guessing attacks is mainly to discover the user password so the intruder can impersonate the client later on.

The online password guessing attacks (using dictionary or brute force attacks) could lead to DoS attack against the user account. The DoS attack against the user account is not solved with MR authentication methods. An attacker can prevent a client from participating in the proper authentication steps. This is because in practice, to mitigate online password guessing attacks, MR authentication methods, such as Malley’s method, apply a separate security measure that locks out a user account if it receives a certain number of unsuccessful password attempts from the client. In other words, if such security measure is applied, the intruder can launch a DoS attack against the user account and prevent the user from performing a job submission authentication with MR application. To mitigate such attacks, we use in CPCrE and CU-AC_AuthN protocols (i) two secrets; a local verification token and a login authentication token of the user, $LAT_U_i$ and $LVT_U_i$, and (2) a local verification process (in the client machine) for the user’s login credential. The local verification process checks whether a certain password guess is correct, before any remote message is sent to the remote server. By using these two secrets in the user login, it is difficult for an attacker to launch his online guessing attack using the local verification process in the client machine. This is because it requires that the two secrets to be available when the intruder launch such attack. In other words, the attacker has not only to recover the encrypted $L_{ACR_C_i}$ which contains $LVT_U_i$, from the stego-object, but also the encrypted $R_{ACR_C_i}$ which contains $LAT_U_i$ must be available to start the local verification process. Thus our protocol mitigates the risk of the online password guessing attacks which could lead to the DoS attack against the user account.

The offline password guessing attacks can also be launched using dictionary or brute force attack. For example, in Malley’s method, the offline guessing attacks against a client long-term credential, i.e. password, are not solved. If a user chooses a poor password, it is possible for an attacker to successfully mount a guessing attacks by repeatedly attempting to decrypt, by testing trial passwords, messages obtained which are encrypted under a key generated from the user’s password. In practice, to obtain such messages, the attacker has only to eavesdrop on the network connection (i.e. has access to such network) between the
client and the remote server, e.g. AS, thus the attacker can launch his offline dictionary or brute force attack to discover the user password, and if such password is compromised the attacker can later impersonate the user, thus his job submission and execution could be compromised. To mitigate the risk of the offline guessing attacks, CPCrE and CU-AC_AuthN protocols use a steganography mechanism to conceal the dynamic login-secret key which is encrypted by the login password-secret key, where the latter is derived from the user password. For an attacker to be able to launch his offline guessing attacks against the user password, he has not only to eavesdrop on the network connection but also he has to compromise the user machine and recover the encrypted key from the stego-object, so he can launch his offline password guessing attacks. Optionally, the encrypted key which is embedded in the stego-object can also be pin-code protected, thus further minimising the risk of unauthorised use of a stego-object. Therefore, the CPCrE and U-AC_AuthN protocols can reduce the risk of the online and offline guessing attacks on user password and a DoS attack on the user account.

**Proposition 3: The C-RM_AuthN protocol can support mutual authentication and message authenticity (satisfy (S2) and (S4))**

Mutual authentication means that the Authentication Server and the Resource Manager should each be assured that they communicate with the MR-Client that it claims to be, and the MR-Client should be assured that it communicates with the claimed Authentication Server and Resource Manager, respectively. Message authenticity means that any modification to a C-RM_AuthN protocol message can be detected and its origin is assured. According to our method discussed in Sections 7.3.2 and 7.5.2, two cryptographic operations, i.e. $E/E^{-1}$ and $HMAC$, and two-factor authentication secrets, i.e. $K_{Ci}$ and $P_{Ci}$ along with $SK_{Ci}^{JS}$ are used to achieve these security properties. $K_{Ci}$ is the first authentication secret that represents something the MR-Client knows. It is derived from the session password-secret key which is generated by the MR-Client using $PW_{ui}$ and approved by the AS using AS’s private key (in CPCrE protocol). $P_{Ci}$ is the second authentication secret that represents something the MR-Client possesses, and it is generated by the Authentication Server and delivered to the MR-Client. $SK_{Ci}^{JS}$ is the job submission-secret key that is generated by the authentication Server and delivered to the Resource Manager and the MR-Client. It is used to authenticate the Resource Manager to the MR-Client once the latter is authenticated to the Authentication Server.
In our method, $K_{Ci}$ and $P_{Ci}$ are used as two-factor authentication secrets to authenticate the MR-Client to both the Authentication Server and the Resource Manager. When the Authentication Server can (i) decrypt the encrypted $Authenticator_{Ci}^{NJ}$ of the received NJ-AuthN Request, i.e. $E^{-1}(P_{Ci}', Authenticator_{Ci}^{NJ})$, to obtain a valid MR-Client authenticator, i.e. $Authenticator_{Ci}^{NJ}$, and (ii) hash the received NJ-AuthN Request message $Msg_{NJ,Ci}$, i.e. $Mac_{NJ,Ci}^* = HMAC(Msg_{NJ,Ci}, K_{Ci})$ to obtain a matching message authentication code, i.e. $Mac_{NJ,Ci}^* = Mac_{NJ,Ci}$, then the Authentication Server is assured of the MR-Client’s identity, as the MR-Client has both the client master-secret key, $P_{Ci}$, and the user password-secret key, $K_{Ci}$. This is because this MR-Client is the only one who (i) has $P_{Ci}$ for encrypting the $Authenticator_{Ci}^{NJ}$, and (ii) knows $K_{Ci}$ for signing the NJ-AuthN Request message, i.e. computing the HMAC signature of $Msg_{NJ,Ci}$. If the Authentication Server is assured that the NJ-AuthN Request is indeed initiated by an authorised MR-Client, the Authentication Server sends a JA-AsseT Response to the Resource Manager notifying that the MR-Client is authenticated, so that the Resource Manager is also assured of the authenticity of the MR-Client.

Furthermore, our method also provides the authentication of both the Authentication Server and the Resources Manager to the MR-Client. With regard to the authentication of the Authentication Server to the MR-Client, when (i) the decryption of the encrypted part of $Msg_{JA,AS}^1$, i.e. $E^{-1}(K_{Ci}, \ldots ||Authenticator_{Ci}^{NJ} || \ldots )$ produces the same MR-Client’s authenticator of the NJ-AuthN Request, i.e. $Authenticator_{Ci}^{NJ} = Authenticator_{Ci}^{NJ}$, and (ii) the hashing of the first part of the JA-AuthN Response message which is received in the NJ-AuthN Response $Msg_{JA,AS}^1$, i.e. $Mac_{JA,AS}^1 = HMAC(Msg_{JA,AS}^1, P_{Ci})$, is the same as the received message authentication code $Mac_{JA,AS}^1$, i.e. $Mac_{JA,AS}^1 = Mac_{JA,AS}^1$, the MR-Client is assured of the authentication of Authentication Server as the Authentication Server has both $K_{Ci}$ and $P_{Ci}$. This is because the Authentication Server is the only one could (i) generate $K_{Ci}$ using $PR_{AS}$, and (ii) swap the use of both authentication secret keys $K_{Ci}$ and $P_{Ci}$ for encrypting the same authenticator of the NJ-AuthN Request message and signing (i.e. computing the HMAC signature of) $Msg_{JA,AS}^1$, respectively. With regard to the authentication of the Resource Manager to the MR-Client, when the keyed hash value of the received NJ-AuthN Response message, i.e. $Mac_{NJ,RM}^1 =$
\(HMAC(Msg_{NJ, RM}, SK^I_{C_i})\), is the same as the received message authentication code \(Mac_{NJ, RM}\), i.e. \(Mac^*_{NJ, RM} = Mac_{NJ, RM}\), the MR-Client is assured that the Resource Manager is authentic as the Resource Manager has the job submission-secret key \(SK^I_{C_i}\) that is to sign the NJ-AuthN Response message. As mentioned early, \(SK^I_{C_i}\) is a secret key which is generated by the Authentication Server and distributed to the Resource Manager and the MR-Client using \(Msg_{JA, AS}\) and \(Msg_{JA, AS}^1\), respectively, where \(Msg_{JA, AS} = \{., ||E(SK^I_{ID}, \{SK^I_{C_i}|| ... \})||ID_{AS}||t_{AS, JA}\}\) and \(Msg_{JA, AS}^1 = \{E(K_{C_i}, \{..||SK^I_{C_i}\}Authenticator^N_{C_i}|| ... \}) ||t_{AS, JA}||ID_{AS}\}.

We use AES algorithm for \(E/E^{-1}\) operations and HMAC-SHA algorithm for \(HMAC\) operation, as these are secure algorithms for encryption and MAC (message authentication code), respectively. Secure AES means that it is computationally infeasible to find the message from its given ciphertext without knowing the secret key. This aspect is achievable when the AES algorithms is secure against chosen plaintext attacks (assuming the use of initialisation vector or nonce in combination with a cryptographic mode such as Cipher Block Chaining CBC [103] [104]). Secure HMAC-SHA means that it is computationally infeasible to generate a MAC value without knowing the secret key, and computationally infeasible to find a message with a given MAC or two messages with the same MAC [82] [26]. This could be achieved when the hash function used for MAC algorithm is a collision resistance and assuming the use of an inner and outer hash functions, a way of hashing the message using the secret key, in order to reach a higher security level [123] [124][102].

Therefore, the C-RM_AuthN protocol provides assurance of (i) the authentication of the MR-Client, the Authentication Server, and the Resource Manager, and (ii) the authenticity of the protocols messages, the NJ-AuthN Request and Response and the JA-Assert Request and Response (satisfy (S2) and (S4)). Any active attacks on the contents of the messages in transit can be detected and modified messages discarded.

Proposition 4: The C-NN_AuthN protocol can support mutual authentication and message authenticity (satisfy (S3) and (S4))

Mutual authentication means that the Name Node and the Data Node should each be assured that they communicate with the MR-Client that it claims to be, and the MR-Client should be assured that it communicates with the claimed Name Node and Data Node,
respectively. Message authenticity means that any modification to a C-NN_AuthN protocol message can be detected and its origin is assured. According to our method discussed in Sections 7.3.3 and 7.5.3, two cryptographic operations, i.e. $E/E^{-1}$ and HMAC, and three-factor authentication secrets, i.e. $K_{C_{i_1}}$, $SK_{C_{i_1}}^{WJ}$ and $SK_{C_{i_1}}^{WA}$, are used to achieve these security properties.

In our method, $K_{C_{i_1}}$ and $SK_{C_{i_1}}^{WJ}$ are used to mutually authenticate MR-Client and the Name Node, and $SK_{C_{i_1}}^{WJ}$ and $SK_{C_{i_1}}^{WA}$ to mutually authenticate MR-Client and the Data Node. As the idea used in these two authentication processes is identical, and the difference is the involved entities and the authentication secrets, we only explain why our C-NN_AuthN protocol can support mutual authentication and message authenticity between the MR-Client and the Name Node. With regard to the authentication of the MR-Client to the Name Node, if (i) the decryption of the encrypted Authenticator$_{C_{i_1}}^{WJ}$ of the received WJ-AuthN Request, i.e. $E^{-1}(K_{C_{i_1}}, \{\text{Authenticator}_{C_{i_1}}^{WJ}|\ldots\})$, results a valid MR-Client authenticator, i.e. Authenticator$_{C_{i_1}}^{WJ}$, and (ii) the hashing of the received WJ-AuthN Request $Msg_{WJ,C_{i_1}}$, i.e. $Mac_{WJ,C_{i_1}} = HMAC(Msg_{WJ,C_{i_1}}, SK_{C_{i_1}}^{WJ})$, produces the same received message authentication code $Mac_{WJ,C_{i_1}}$, i.e. $Mac_{WJ,C_{i_1}}^{*} = Mac_{WJ,C_{i_1}}$, then the Name Node is assured of the MR-Client identity. This is because this MR-Client the only one who has both the user password-secret key $K_{C_{i_1}}$ for encrypting Authenticator$_{C_{i_1}}^{WJ}$ and the writing job-secret key $SK_{C_{i_1}}^{WJ}$ for signing $Msg_{WJ,C_{i_1}}$.

With regard of the authentication of the Name Node to the MR-Client, when the MR-Client can (i) decrypt the encrypted Authenticator$_{C_{i_1}}^{WJ^*}$, i.e. $E^{-1}(SK_{C_{i_1}}^{WJ},\{\text{Authenticator}_{C_{i_1}}^{WJ^*}|\ldots\})$, and (ii) hash the received WJ-AuthN Response message $Msg_{WJ,NN}$, i.e. $Mac_{WJ,NN}^{*} = HMAC(Msg_{WJ,NN}, K_{C_{i_1}})$, to obtain a matching authenticator, i.e. Authenticator$_{C_{i_1}}^{WJ^*} = $ Authenticator$_{C_{i_1}}^{WJ}$ and a matching message authentication code, i.e. $Mac_{WJ,NN}^{*} = Mac_{WJ,NN}$, the MR-Client is assured of the identity of the Name Node, as the Name Node has both $SK_{C_{i_1}}^{WJ}$ and $K_{C_{i_1}}$. This is because only the Name Node could (i) recover $K_{C_{i_1}}$, which is wrapped using $SK_{C_{i_1}}^{WJ}$, and (ii) swap the use of both authentication secrets $SK_{C_{i_1}}^{WJ}$ and $K_{C_{i_1}}$, for encrypting the same authenticator of the WJ-
AuthN Request message and signing (i.e. computing the HMAC signature) of the WJ-AuthN Response, i.e. $Msg_{WJ,NN}$.

Therefore, similar to the C-RM_AuthN protocol, if a secure AES and HMAC-SHA algorithms are used, the C-NN_AuthN protocol provides assurance of (i) the authentication of the MR-Client, the Name Node, and the Data Node using three independent authentication secrets, and (ii) the authenticity of the protocols messages, the WJ-AuthN Request and Response, and the R-WJ-AuthN Request and Response (satisfy (S3) and (S4)). Thus, any active attacks on the contents of the protocol messages in transit can be detected and the modified message discarded.

**Proposition 5: The MR-CAuthN protocols can support message confidentiality (satisfy (S5))**

Our MR-CAuthN protocols support the confidentiality of the protocol messages exchanged between MR-AuthN Components. In the C-RM_AuthN protocol, the MR-Client’s authenticator of the NJ-AuthN Request, $Authenticator_{C_i}^{NJ}$, is encrypted at his source (CJ-AuthN Agent) using $P_{C_i}$, before being sent to the RM-AuthN Relay Module. Then the NJ-AuthN Request is relayed to the G-AuthN Service. Once received, it is decrypted by the G-AuthN Service to recover $Authenticator_{C_i}^{NJ}$ using $P_{C_i}$. Further, this authenticator along with other authentication information, which includes $ST_{C_i}^{1}$, $SK_{C_i}^{JD}$ and $SK_{C_i}^{JS}$, are encrypted at the source (G-AuthN Service) using $K_{C_i}$. $SVT_{C_i}^{1}$ is also encrypted at the same source using $SK_{C_i}^{JD}$ before being sent to the RM-AuthN Relay Module. Then the RM-AuthN Relay Module forwards the encrypted $SVT_{C_i}^{1}$ to the NN-AuthN Module before the encrypted $ST_{C_i}^{1}$, $SK_{C_i}^{JD}$ and $SK_{C_i}^{JS}$ are delivered to the CJ-AuthN Agent using the NJ-AuthN Response.

In the C-NN_AuthN protocol, the MR-Client’s authenticator of the WJ-AuthN Request, $Authenticator_{C_i}^{WJ}$, and the writing job-secret key, $SK_{C_i}^{WJ}$, are encrypted at the source (CJ-AuthN Agent) using $K_{C_i}$ and $PU_{NN}$, respectively, before being sent to the NN-AuthN Module. Once received, the NN-AuthN Module decrypts the encrypted $SK_{C_i}^{WJ}$ using $PR_{NN}$ to recover $K_{C_i}$ from $SVT_{C_i}^{1}$, and decrypts the encrypted $Authenticator_{C_i}^{WJ}$ using $K_{C_i}$. This authenticator, i.e. $Authenticator_{C_i}^{WJ}$, and $ST_{C_i}^{2}$ are encrypted at the source (NN-AuthN
Module) using $SK_{Ci}^{WJ}$ before being sent to the CJ-AuthN Agent, and $SVT_{Ci}^{2}$ is encrypted at the same source using $SK_{Li}^{ID}$ before being sent to the DN-AuthN Module. Further, the MR-Client’s authenticator of the R-WJ-AuthN Request, $Authenticator_{Ci}^{RWJ}$, is encrypted at his source (CJ-AuthN Agent) using the $SK_{Ci}^{WJ}$, and the writing access-secret key, $SK_{Ci}^{WA}$, is encrypted using $SK_{Li}^{ID}$, before being sent to the DN-AuthN Module. Once received, the DN-AuthN Module decrypts the encrypted $SK_{Ci}^{WA}$ using $SK_{Li}^{ID}$ to recover $SK_{Ci}^{WJ}$ from $SVT_{Ci}^{2}$, and decrypts the encrypted $Authenticator_{Ci}^{RWJ}$ using $SK_{Ci}^{WJ}$. Also, this authenticator, i.e. $Authenticator_{Ci}^{RWJ}$, is encrypted at the source (DN-AuthN Module) using the $SK_{Ci}^{WA}$ before being sent to the CJ-AuthN Agent.

Taking into account that (1) the used symmetric and asymmetric key algorithms, i.e. AES and RSA, are secure, and (2) the exchanged messages of the MR-CAuthN protocols are authentic (i.e. established using HMAC algorithm), only authorised entities can access the authentication information of the MR-Client (satisfy (S5)). Further, by including a timestamp in each protocol message and the assumption A7 (in Section 5.6.2), the protocol ensures that the received message is fresh.

Therefore, based on all achievement of these security features, i.e. message authenticity, confidentiality and freshness, all other unauthorised entities would not be able to perform MITM attack to impersonate or masquerade as a legitimate MR-Client or an MR-Inf. Component using any of the MR-CAuthN protocols.

**Proposition 6:** The JCPCrE and C-NN_AuthN protocols can distribute the primary authentication and verification tokens securely (satisfy (S6))

This security property means that when the primary authentication and verification tokens of the MR-Job Components, i.e. $PT_{*i}^{j} = \{PT_{JT}^{j}||PT_{TT}^{j}||PT_{RT}^{j}\}$ and $PVT_{*i}^{j} = \{PVT_{JT}^{j}||PVT_{TT}^{j}||PVT_{RT}^{j}\}$, are distributed to the respective MR Components, they should be confidentiality and integrity protected.

In the JCPCrE protocol, $PT_{*i}^{j}$ is encrypted and signed at the source (CJ-AuthN Agent) using $SK_{Ci}^{IS}$ before being submitted to the RM-AuthN Relay Module. Once received, the RM-AuthN Relay Module verifies the correctness of $PT_{*i}^{j}$ before being allocated to the respective JT-AuthN Module. To allocate $PT_{*i}^{j}$, $PT_{*i}^{j}$ is encrypted and signed in the source
(G-AuthN Relay Module) using $SK_i^{ID}$. Once received, the JT-AuthN Module verifies the correctness of $PT_{ij}$ and reads $PT_{TTj}$ and $PT_{RTj}$ from $PT_{ij}$. $PT_{TTj}$ and $PT_{RTj}$ are encrypted and signed at the source (JT-AuthN Module) using $SK_i^{ID}$ before being assigned to the respective Task Trackers.

Further, as mentioned in Section 7.6, the JCPCrE protocol works in conjunction with the C-NN_AuthN protocol. $PVT_{ij}$ is encrypted and signed at the source (CJ-AuthN Agent) using $K_{Ci}$ and $SK_{WJ}^{Ci}$, respectively, before being sent to the NN-AuthN Module. Once received, the NN-AuthN Module verifies the integrity and origin of $PVT_{ij}$ before being used by the NN-AuthN Module.

Taking into account that (1) the used symmetric key and HMAC algorithms, i.e. AES and SHA256, are secure, and (2) both the exchanged messages of the JCPCrE protocol and the WJ-AuthN Request message of the C-NN_AuthN protocol are authentic, the confidentiality and integrity of the primary authentication and verification tokens of the MR-Job Components are assured and only the authorised entities can access them (satisfy (S6)).

### 7.9.2. Protocol Modelling and Formal Verification Results

In addition to the security analysis conducted in the previous section, the C-RM and C-NN_AuthN protocols are also formally verified. Similar to the CPCrE protocol, the C-RM and C-NN_AuthN protocols are modelled using the AVISPA tool. Figure 7.16 shows A&B notation that illustrates the five basic roles (modules) and their sequences of actions to model the protocols, and Figure 7.17 shows the validation results of modelling such protocols. As can be seen in the figure, the validation results of the tool shows that the protocols are correct and safe; no attacks are found.
Figure 7.16 The five basic roles and their sequences of actions to model C-RM and C-NN_AuthN protocols

C-RM_AuthN protocol

**SUMMARY**
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS PROTOCOL
/home/span/span/testsuite/results/C_RMAuthenticationProtocol.if
GOAL
as specified
BACKEND OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 4.46s
visitedNodes: 548 nodes
depth: 7 plies

C-NN_AuthN protocol

**SUMMARY**
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS PROTOCOL
/home/span/span/testsuite/results/C_NNAuthenticationProtocol.if
GOAL
as specified
BACKEND OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 1.11s
visitedNodes: 266 nodes
depth: 6 plies

(A) OFMC

(B) ATSE

Figure 7.17 AVISPA validation results
7.10. Performance Evaluation

This section discusses the performance evaluation of the MR-CAuthN and JCPCrE protocols. The performance is evaluated in terms of the computational and communication costs and the protocol execution time. Two experiments are carried out. The first one is carried out to measure the computational and communication costs, and the second one is carried out to measure the protocol execution time. The result is compared with the Malley’s method. The Malley’s method is chosen for the comparison because (i) the Malley’s method is the only method adopted in the MR model implementation, (ii) like our method, the Malley’s method is a symmetric key based authentication method, and (ii) the Malley’s method is more clearly explained in literature. As the Malley’s method is a Kerberos based so it is referred to as KRB, and to its authentication components used as KRB_C for client, KRB_AS for the Authentication Server, KRB_TGS for the Ticket Granting Ticket Server, and KRB_AP for the server that the client sends his new job or writing job request to, e.g. the Resource Manager or Name Node, respectively (i.e. Application Server).

7.10.1. Computational Cost

This section evaluates and compares the computational cost introduced by the MR-CAuthN and JCPCrE protocols and KRB protocol both theoretically and experimentally. With regard to the theoretical evaluation, the number and the types of the cryptographic algorithms used in each protocol are identified. With regard to the experimental evaluation, the computational cost is measured in term of the processing time that an MR-AuthN Component takes in each step to process a sent or received message of the MR-CAuthN and JCPCrE protocols and compared with those of the KRB protocol.

7.10.1.1. Theoretical Evaluation

The computational cost imposed on MR-AuthN Components by each step of the MR-CAuthN and the JCPCrE protocol is evaluated. The evaluation is explained below and a summary of the evaluation along with the corresponding by the KRB protocol is given in Table 7.4.

**Computational cost of the U-AC_AuthN protocol**

As described in Section 7.5.1, the MR-AuthN Components of the protocol each performs a number of cryptographic operations, as detailed below:
In Step1- a User Login, the C-AuthN Agent does not perform any cryptographic operation. Rather it is to enter the removable part of the MR-Client authentication credential and the user authentication credential (i.e. the MR-Client’s username and password pair) into the user machine and the C-AuthN Agent, respectively.

In Step2- a User Login Verification, the C-AuthN Agent performs two hash operations to generate the login password-secret key and the local verification token, and two symmetric decryption operations to decrypt the local and removable parts of the MR-Client authentication credential.

Computational cost of the C-RM_AuthN protocol
In the C-RM_AuthN protocol, as described in Section 7.5.2, the MR-AuthN Components of the protocol each performs a number of cryptographic operations, detailed as following:

- In Step1- NJ-Request Initialization, the CJ-AuthN Agent performs one hash operation to generate the current location-secret code of the MR-Client, one symmetric encryption operation to protect the confidentiality of the freshly constructed MR-Client’s authenticator of the NJ-AuthN Request message, and one message authentication code operation to protect the authenticity of the message.

- In Step2- Send NJ-AuthN Request, the CJ-AuthN Agent does not perform any cryptographic operation as this step is to send the NJ-AuthN Request once its object constructed.

- In Step3- NJ-Request Relaying and Step4- Send JA-Assertion Request, the RM-AuthN Relay Module does not perform any cryptographic operation as these steps are to relay the NJ-AuthN Request to the G-AuthN Service as a JA-AssertT Request.

- In Step5- JA-AssertT Request Verification, the G-AuthN Service performs two hash operations to generate the user password-secret key and the location secret-code of the MR-Client, one message authentication code operation to verify the HMAC signature of the NJ-AuthN Request message, and two symmetric decryption operations to obtain the client master-secret key and decrypt the received MR-Client’s authenticator.

- In Step6- Generate a Writing-Job Authentication Information, the G-AuthN Service performs four random-number generator operations and four hash operations to generate the following keys: a fresh dynamic login-secret key, a writing job-secret key, a job domain-secret key and a job submission-secret key, and three symmetric encryption and two message authentication code operations to
protect the confidentiality of the contents and the authenticity of the JA-AsserT Response message, respectively.

- **In Step7- Send a JA-AsserT Response**, the G-AuthN Service does not perform any cryptographic operation as this step is to send the JA-AsserT Response once its object constructed.

- **In Step8- JA-Response Relaying**, the RM-AuthN Relay Module performs *three* message authentication code to verify the HMAC signature of the JA-AsserT Response and to protect the authenticity of both the NJ-AuthN Response and the Forward, respectively, *one* symmetric decryption operation to recover the job submission-secret key, and *one* symmetric encryption operation to protect the confidentiality of the first secondary authentication token of the MR-Client.

- **In Step9- Forward and Step10- NJ-AuthN Response**, the RM-AuthN Relay Module does not perform any cryptographic operation as these steps are to send the Forward and the NJ-AuthN Response once their objects constructed.

- **In Step11- NJ-AuthN Response Verification**, the CJ-AuthN Agent performs *two* message authentication code operations to verify two HMAC signatures in the NJ-AuthN Response, which are generated by the G-AuthN Service and the RM-AuthN Relay Module respectively, *one* symmetric decryption to obtain both the MR-Client’s authenticator and the secondary MR-Client authentication token which is used to authenticate the MR-Client to the Name Node.

**Computational cost of the C-NN_AuthN protocol**

In the C-NN_AuthN protocol, as described in Section 7.5.3, the MR-AuthN Components of the protocol perform a number of cryptographic operations, as detailed below:

- **In Step1- WJ-Request Initialization**, the CJ-AuthN Agent performs *one* symmetric encryption operation to protect the confidentiality of both the freshly constructed MR-Client’s authenticator of the WJ-AuthN Request message and the primary verification tokens of the MR-Job Components, *one* message authentication code operation to protect the authenticity of the message, and *one* asymmetric encryption operation to protect the confidentiality of the writing job-secret key when it is delivered to the NN-AuthN Module in the WJ-AuthN Request message.

- **In Step2- Send WJ-AuthN Request**, the CJ-AuthN Agent does not perform any cryptographic operation as this step is to send the WJ-AuthN Request once its object constructed.
In **Step3- WJ-Request Verification**, the NN-AuthN Module performs *one* asymmetric decryption operation to obtain the writing job-secret key, *one* message authentication code operation to verify the HMAC signature of the WJ-AuthN Request message, and *two* symmetric decryption operations to obtain the user password-secret key and decrypt the received MR-Client’s authenticator to verify its correctness.

In **Step4- Generate a Writing Access Authentication Information**, the NN-AuthN Module performs *one* random-number generator operation and *one* hash operation to generate the writing access-secret key, and *four* symmetric encryption and *two* message authentication code operations to protect the confidentiality of the contents and the authenticity of both the WJ-AuthN Response and the Notify messages.

In **Step5- Notify and Step6- Send a WJ-AuthN Response**, the NN-AuthN Module does not perform any cryptographic operation as these steps are to send the Notify and the WJ-AuthN Response once their objects constructed.

In **Step7- WJ-AuthN Response Verification**, the CJ-AuthN Agent performs *one* message authentication code operation to verify the HMAC signature of the WJ-AuthN Response which is generated by the NN-AuthN Module, and *one* symmetric decryption operation to obtain both the MR-Client’s authenticator and the MR-Client’s secondary authentication token which is used to authenticate the MR-Client to the Data Node.

In **Step8- R-WJ-Request Initialization**, the CJ-AuthN Agent performs *one* symmetric encryption operation to protect the confidentiality of a freshly constructed MR-Client’s authenticator of the R-WJ-AuthN Request, and *one* message authentication code operation to protect the authenticity of the R-WJ-AuthN Request message.

In **Step9- Send R-WJ-AuthN Request**, the CJ-AuthN Agent does not perform any cryptographic operation as this step is to send the R-WJ-AuthN Request once its object is constructed.

In **Step10- R-WJ-Request Verification**, the DN-AuthN Module performs *two* message authentication code operations (*one* is to verify the HMAC signature of the R-WJ-AuthN Request message and *one* is to protect the authenticity of the R-WJ-AuthN Response message), *three* symmetric decryption operations to obtain the writing access-secret key, the writing job-secret key and the received MR-Client’s
authenticator to verify its correctness, and one symmetric encryption operation to protect the confidentiality of the contents of the R-WJ-AuthN Response message.

- **In Step11- Send a R-WJ-AuthN Response**, the DN-AuthN Module does not perform any cryptographic operation as this step is to send the R-WJ-AuthN Response once its object constructed.

- **In Step12- WJ-AuthN Response Verification and MR-Client Authentication Credential Protection**, the CJ-AuthN Agent performs one message authentication code operation to verify the HMAC signature of the R-WJ-AuthN Response message which is generated by the DN-AuthN Module, one symmetric decryption operation to obtain the MR-Client’s authenticator, one random-number generator and two hash operations to issue a local authentication token and local verification token, and two symmetric encryption operations to protect confidentiality of the local and removable parts of the MR-Client authentication credential.

**Computational cost of the JCPCrE protocol**

In the JCPCrE protocol, as described in Section 7.8, the MR-AuthN Components of the protocol perform a number of cryptographic operations, as detailed below:

- **In Step1- Generate an MR-Job Authentication Information**, the CJ-AuthN Agent performs three random-number generator operations and three hash operations to generate three role based-secret keys (i.e. retrieving input-split-secret key, retrieving origin-data-secret key, and writing final-data-secret key), and six symmetric encryption operations, these operations are used to form the primary authentication and verification tokens of the MR-Job Components.

- **In Step2- Submit**, the CJ-AuthN Agent performs one symmetric encryption operation to protect the confidentiality of the primary authentication tokens of the MR-Job Components, and one message authentication code operation to protect the authenticity of the Submit message.

- **In Step3- Allocate**, the RM-AuthN Relay Module performs one message authentication code operation to verify the authenticity of the Submit message, one symmetric decryption operation to obtain the primary authentication tokens of the MR-Job Components, and one symmetric encryption operation and one message authentication code operation to protect the confidentiality and the authenticity of the Allocate message respectively.

- **In Step4- Assign**, the JT-AuthN Module performs one message authentication code operation to verify the authenticity of the Allocate message, one symmetric
decryption operation to obtain the primary authentication token of the allocated Job Tracker, and one symmetric encryption operation and one message authentication code operation to protect the confidentiality and the authenticity of the Assign message, respectively. Also, the TT-AuthN Module performs one message authentication code operation to verify the authenticity of the Assign message, one symmetric decryption operation to obtain the primary authentication tokens of the assigned Task Tracker and Reduce Tasks.

Table 7.4 A comparison of the computational cost of the MR-Client authentication

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Our method</th>
<th>Malley’s method</th>
<th>KRB protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U-AC_AuthN protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C-AuthN Agent</td>
<td>2TH+2TDec</td>
<td>KRB_C, KRB_AS, KRB_TGS, and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KRB_RM/NN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C-RM_AuthN protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CJ-AuthN Agent</td>
<td>1TH + 1TEnc + 3TMac + THDec</td>
<td>14TH + 8TEnc + 8TDec + 2TDec</td>
</tr>
<tr>
<td></td>
<td>RM-AuthN Relay Module</td>
<td>3TMac + 1TEnc + 1TDec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>G-AuthN Service</td>
<td>6TH + 3TEnc + 2TDec + 3TMac + 4TRng</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C-NN_AuthN protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CJ-AuthN Agent</td>
<td>5TEnc + 4TMac + 2TDec + 1TAEnc + 2TH + 1TRng</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NN-AuthN Module</td>
<td>3TMac + 4TEnc + 2TDec + 1TAdc + 1TH + 1TRng</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DN-AuthN Module</td>
<td>1TEnc + 2TMac + 3TDec</td>
<td>14TH + 8TEnc + 8TDec + 2TDec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JCPCrE protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CJ-AuthN Agent</td>
<td>7TEnc + 1TMac + 3TH + 3TRng</td>
<td>15TH + 8TEnc + 8TDec + 3TRng</td>
</tr>
<tr>
<td></td>
<td>RM-AuthN Relay Module</td>
<td>1TEnc + 1TDec + 1TMac + 2TMac</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JT-AuthN Module</td>
<td>1TEnc + 1TDec + 1TMac</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TT-AuthN Module</td>
<td>1TDec + 1TMac</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MR-Job Components Identification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T = Time; H = Hash; Enc = Symmetric encryption; Dec = Symmetric decryption; Rng = Random-number generator; AEnc = Asymmetric encryption; ADec = Asymmetric decryption; Mac = Message Authentication Code.
7.10.1.2. Experimental Evaluation

This section reports the experimental evaluation of the computational cost (i.e. the processing time in second) introduced by each MR-AuthN Component for both MR-CAuthN and JCPCrE protocols and compares the costs with the processing time introduced by the authentication components of the KRB protocol. As the processing time is mainly based on the cost introduced by the cryptographic operations (e.g. encryption), and this cost may vary from one algorithm to another (e.g. AES or DES) and from one benchmark to another, this experiment is conducted using the following algorithms and benchmark.

- AES algorithm with a key of 128 bits long is used for the symmetric key operations (encryption/decryption).
- RSA algorithm with a key of 2048 bits long is used for the asymmetric key operations (encryption/decryption).
- HMAC-SHA256 is used for message authentication code operations.
- SHA-256/512 is used for hashing operations.
- SHA1-PRNG with 1024 bytes long is used for random-numbers generating operations.
- An identifier used for each entity of all the protocols is 15 bytes long.
- The Microsoft Active Directory Certificate Service (AD CS) is used to issue and manage the standard private keys and the corresponding public key certificates.
- The Java Standard Edition (Java SE) in the NetBeans Integrated Development Environment (NetBeans IDE v.8) platform is used to develop the prototype.
- In addition, the following hardware and software components are used: (1) a desktop computer running Microsoft Windows 8.1 version 64-bit based Operating System 2013 with Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz and 8GB of RAM, and (2) an IBM Server machine running Microsoft Windows Server 2012 Version 64-bit based Operating System with Intel(R) Xeon(R) E5440 CPU @2.8GHz and 16GB of RAM.

To ensure statistical significance of the experimental data, a number of iterations (n) over which a processing time is measured is determined. The experiment has been conducted by setting n into different values (starting from n = 1) and the average of processing times taken by each MR-AuthN Component are measured. The bigger the value of the iteration n, the less effect the other initialization factors have on the computational cost. An n value higher than 3K (i.e. 3000 iterations) is adequate, so n = 3.5 is chosed for
the data collection. Figure 7.18 to 7.20 show the results of this experiment. Figure 7.18 shows the average processing time taken by each MR-AuthN Component of the MR-CAuthN protocols. From the figure, it can be seen that the C-AuthN Agent takes less than a half second, i.e. 310 ms\textsuperscript{18}, to recover the encrypted local part of the MR-Client authentication credential from the stego-object. It also takes about 0.583 ms to complete both a user login and verification. Here, it is assumed that the time introduced by the user to enter his removable storage media and his authentication credential (i.e. MR-Client username and password) is zero. The CJ-AuthN Agent takes about 0.157 ms to initialize the NJ-AuthN Request, 1 ms to verify the NJ-AuthN Response and initialize the WJ-AuthN Request, 0.111 ms to verify the WJ-AuthN Response and initialize the R-WJ-AuthN Request, and about 0.052 ms to verify the R-WJ-AuthN Response. The RM-AuthN Relay Module takes only about 0.003 ms to relay the NJ-AuthN Request to the G-AuthN Service and about 3.5 ms to complete the JA-Response Relaying step. The G-AuthN Service takes about 9.6 ms to verify the JA-AsserT Request and generate the job authentication information of an MR-Client. The NN-AuthN Module takes about 15.5 ms to verify the WJ-AuthN Request and generate a writing access authentication information. Finally, the DN-AuthN Module takes about 0.149 ms to verify the R-WJ-AuthN Request.

\textsuperscript{18} ms: millisecond/s
Further, Figure 7.19 shows the average processing time introduced by the CJ-AuthN Agent, RM-AuthN Relay Module, and the JT-AuthN Module for the JCPCrE protocol. As shown in the figure the CJ-AuthN Agent takes about 0.413 ms to generate the MR-Job.
authentication information and to initialize the Submit message, while the RM-AuthN Relay Module takes about 0.065 ms to verify the Submit message and initialize the Allocate message and JT-AuthN Module takes about 0.09 ms to verify the Allocate message and initialize the Assign message.

![The computational cost introduced by](image)

**Figure 7.19 The computational cost of the JCPCrE protocol**

On the other hand, Figure 7.20 shows the average processing time introduced by the authentication components of the KRB protocol. As shown in the figure the KRB_C takes about 0.123 ms to initialize KRB_AS_REQ, 0.095 ms to verify the KRB_AS_RES and initialize the KRB_TGS_REQ, 0.094 ms to verify the KRB_TGS_RES and initialize the KRB_AP_REQ, and about 0.053 ms to verify the KRB_AP_RES. Also, the KRB_AS takes about 0.306 ms to verify the KRB_AS_REQ and initialize the KRB_AS_RES, and KRB_TGS takes about 0.279 ms to verify the KRB_TGS_REQ and initialize the KRB_TGS_RES. Further, the KRB_AP takes about 0.216 ms to verify the KRB_AP_REQ and initialize the KRB_AP_RES.
The computational cost introduced by the authentication components of KRB protocol (~1.2 ms) is lower than the one introduced by the MR-AuthN Components of our MR-CAuthN protocols (total of about ~30.6 ms, where ~0.583 ms for U-AC_AuthN protocol, ~14.3 ms for C-RM_AuthN protocol and ~15.8 ms for C-NN_AuthN protocol). The extra cost is due to that the MR-CAuthN protocols involve asymmetric key cryptographic operations to deliver the writing job-secret key ($Enc^{-1}(Enc(PU_{NN},SK_{CI}^{WJ}))$), and hashing message authentication code operations to protect the authenticity of each authentication message. In addition, the MR-CAuthN protocols messages are longer than KRB protocol messages as the former contains the authentication data, $ST_{CI}^{1}, SVT_{CI}^{1}, ST_{CI}^{2},$ and $SVT_{CI}^{2}$, and these data are encrypted before being transferred and decrypted before being used. This introduce additional computational cost as well.

### 7.10.2. Communication Cost

The communication cost introduced by the protocols is evaluated in terms of the number and the length of the protocol messages in each protocol execution. In each MR-Client authentication instance performed by using the MR-CAuthN protocols, once the MR-Client is authenticated to both the Authentication Server and the Resource Manager, the CJ-AuthN Agent receives his secondary authentication token to identify his WJ-AuthN Request. The WJ-AuthN Request contains the primary verification tokens of the MR-Job Components of his job domain which are sent to the Name Node. Also, in each MR-Job Components
identification instance performed by using the JCPCrE protocol, the CJ-AuthN Agent submits the primary authentication tokens of the MR-Job Components to the Resource Manage which allocates them to the respective MR-Job Components. Thus, in this section, the communication cost of both the MR-CAuthN protocols and the JCPCrE protocol are measured and compared with KRB protocol.

The length of each exchanged message of MR-CAuthN protocols and JCPCrE protocol are measured in bytes. Figure 7.21 illustrates the messages, and their lengths, that are exchanged in both C-RM and C-NN_AuthN protocols to perform an MR-Client authentication instance. As can be seen from the figure, 10 messages are exchanged. The three messages, NJ-AuthN Response, JA-AsserT Response and WJ-AuthN Request have larger sizes than other messages. They are each more than 1K byte long. This is because the JA-AsserT Response message contains both the secondary authentication and verification tokens of the MR-Client, i.e. $|S^1_{TC_i}|$ and $|S^1_{VT_i}|$, while the NJ-AuthN Response and the WJ-AuthN Request messages contain the 2nd secondary authentication token of the MR-Client, i.e. $|S^2_{TC_i}|$ and the primary verification tokens of the MR-Job Components $|PV_{TC_i}|$, respectively.

![Figure 7.21 Communication Cost: the length of the MR-CAuthN protocols messages](image)

Further, Figure 7.22 illustrates the lengths of protocol messages exchanged during an execution of the JCPCrE protocol. As can be seen from the figure, the Submit and Allocate messages have the same message length, which is just over 600 bytes long. This is about 25% larger than the Assign message size. This is because the former messages contain the primary authentication tokens of both the Job Tracker and the sets of all the Task Trackers and Reduce Tasks assigned to an MR-Job domain ($|PT_{TC_i}| = |PT_{JT_i}| |PT_{TT_i}| |PT_{RT_i}|$). On
the other hand, the Assign contains only the primary authentication tokens of the sets of the Task Trackers and Reduce Tasks \(|PT_{TT_j}|\) and \(|PT_{RT_j}|\).

![Figure 7.22 Communication Cost: the length of the JCPCrE protocol messages](image)

The total communication overhead of our MR-CAuthN and JCPCrE protocols are measured and compared with the KRB protocol. The total communication overhead (TCO) is the total length (in bytes) of all the messages exchanged in each protocol run. As shown in Figure 7.23, the C-RM and C-NN_AuthN protocols each introduces a larger TCO (with 27% and 4%, respectively) than KRB protocol. The C-RM_AuthN protocol introduces the highest level of TCO in comparison with the other protocols. The C-RM_AuthN protocol has about 3.6K bytes whereas the C-NN_AuthN protocol and the KRB protocol have about 2.7K and 2.6K bytes, respectively. This is because in our authentication method, a push mode of authentication for the C-NN_AuthN and the JCPCrE protocols is used. The C-NN_AuthN protocol requires the transmission of the secondary verification token of the MR-Client \(|SVT^2_{Ci}|\) to the Name Node, to allow the Name Node to validate the WJ-AuthN Request of the MR-Client. The JCPCrE protocol requires the transmission of the authentication information to the MR-Client, and the primary verification tokens of the MR-Job Components \(|PVT_{j}|\) to the Name Node, to allow the Name Node to validate the RB-AuthN Requests of MR-Job Components in an MR-Job domain. The KRB protocol, on the other hand, uses a pre shared secret key between the ticket provider (i.e. the TGS) and the verifier (e.g. the Resource Manager or the Name Node) to validate the service ticket used to identify a NJ-AuthN Request or a WJ-AuthN Request, respectively. Also, the exchanged messages, in KRB protocol, do not contain any authentication information related to the MR-Client’s job domain.
7.10.3. Protocol Execution Time

In this section, PETs experienced by the MR-CAuthN protocols are examined and compared with that experienced by the KRB protocol. We first present a theoretical model
that is constructed and used to calculate the theoretical values of the PETs. We then report
the experimental measurements of the PET for the protocols.

7.10.3.1. Theoretical Model

As mentioned in Section 6.7.3.1, the PET of a protocol is the sum of delays of all the
rounds of communications taking place when the protocol is executed and it can be
calculated using the following equation:

\[
PET = \sum_{r=1}^{r=k} (\text{Delay}_{\text{COMR}_r}) \quad \text{(EQ 7.1)}
\]

The theoretical model is constructed to be identical for all protocols of the LVAP suite
and the KRB protocol. In the previous chapter, Section 6.7.3.1 discussed how such model
is constructed and used to calculate the theoretical values of the PET of the CPCrE protocol.
As can be seen in Figure 6.14, the CPCrE protocol consists of one communication
round \( k = 1 \). However, C-RM, C-NN, and KRB protocols each consists of three
communication rounds \( k = 3 \), that is more complex to construct the model and use it to
compute the theoretical values of the PET. So here in this section, without loss of generality,
the C-RM_AuthN protocol is used as an example to explain how such model could be
constructed and used to calculate the PET of these protocols when \( k = 3 \).

Based on the equation above, i.e. EQ 7.1, the PET of the C-RM_AuthN protocol is as
following:

\[
PET = \sum_{r=1}^{r=3} (\text{Delay}_{\text{COMR}_r}) = \text{Delay}_{\text{COMR}_1} + \text{Delay}_{\text{COMR}_2} + \text{Delay}_{\text{COMR}_3}
\]

Each \( \text{Delay}_{\text{COMR}_r} \) as explained below in EQ 7.2, consists of two components. The first
is the delays caused by the two MR-Authority Components (MRC) of a communication
round \( r \), i.e. experienced by the protocol messages in the two MR-Authority Components of
each \( r \). The second one is the delay caused by the network (NET) that connects these two
MR-Authority Components, i.e. experienced by the protocol messages in transit in the network
in the communication round \( r \).

\[
\text{Delay}_{\text{COMR}_r} = \text{Delay}_{\text{MRC}_r} + \text{Delay}_{\text{NET}_r} \quad \text{(EQ 7.2)}
\]
Figure 7.25 illustrates these two delays which could be experienced in each round of the C-RM_AuthN protocol, where $k = 3$.

Figure 7.25 The theoretical value of the PET for the C-RM_AuthN protocol

$Delay_{MRC_r}$ of each communication round shown in the figure above, i.e. $Delay_{MRC_1}$, $Delay_{MRC_2}$ and $Delay_{MRC_3}$, has further two components, one for each protocol message. For example in the first communication round $k = 1$, One is for a NJ-AuthN Request ($Req$) and the second one is for a NJ-AuthN Response ($Res$). Each component is a function of three delays that a $Req$ or a $Res$ experienced in $MRC_1$. The three delays are a queuing delay ($QueuD$), processing delay ($ProcD$), and transmission delay ($TranD$). $Delay_{MRC_r}$ can be calculated using the following equation.

$$
Delay_{MRC_r} = \left\{ \begin{array}{l}
\sum_{e=1}^{2} (ProcD_{e}^{Req} + QueuD_{e}^{Req}) + TranD_{e}^{Req} \\
\sum_{e=1}^{2} (ProcD_{e}^{Res} + QueuD_{e}^{Res}) + TranD_{e}^{Res}
\end{array} \right\} \quad \text{------- (EQ 7.3)}
$$

$Delay_{NET_r}$ is similar to $Delay_{MRC_r}$ in that $Delay_{NET_r}$ has two components one for each protocol message, i.e. $Req$ or $Res$. However, $Delay_{NET_r}$ is a function of four delays that a $Req$ or a $Res$ experienced in $NET_r$, i.e. experienced at each intermediate node and the network media connecting the nodes en route from the source to the destination node. The
four delays are queuing delay ($\text{QueuD}$), processing delay ($\text{ProcD}$), transmission delay ($\text{TranD}$), and propagation delay ($\text{PropD}$). Assuming there are $n$ intermediate nodes connecting the two MR-AuthN Components. $\text{Delay}_{\text{NET}_r}$ is calculated as follows:

\[
\text{Delay}_{\text{NET}_r} = \left\{ \sum_{i=1}^{n} (\text{QueuD}_{i}^{\text{Req}} + \text{ProcD}_{i}^{\text{Req}} + \text{TranD}_{i}^{\text{Req}} + \text{PropD}_{i}^{\text{Req}}) + \text{PropD}_{1}^{\text{Req}} \right\} + \left\{ \sum_{i=1}^{n} (\text{QueuD}_{i}^{\text{Res}} + \text{ProcD}_{i}^{\text{Res}} + \text{TranD}_{i}^{\text{Res}} + \text{PropD}_{i}^{\text{Res}}) + \text{PropD}_{2}^{\text{Res}} \right\} 
\]

where $\text{PropD}_{1}^{\text{Req}}$ is the propagation delay that experienced in a network media (NM) connecting the first MR-AuthN Component (e.g. CJ-AuthN Agent) that initiates the Req (e.g. NJ-AuthN Request) and the first intermediate node that receives the Req, while $\text{PropD}_{2}^{\text{Res}}$ is the propagation delay that experienced in a NM connecting the second MR-AuthN Component (e.g. RM-AuthN Relay Module) that initiates the Res (e.g. NJ-AuthN Response) and the first intermediate node that receives the Res.

Figure 7.26 illustrates these delays and shows how the total delay, i.e. $\text{Delay}_{\text{COMR}_r}$, in the three rounds of the C-RM_AuthN protocol messages exchange are calculated. In the figure the $\text{Req}$ and $\text{Res}$ messages of the C-RM_AuthN protocol, i.e. the NJ-AuthN Request and Response, JA-AsserT Request and Response, and Forward are denote as NJRequest/Response, JARequest/Response, and Forward, respectively. The definition of each of these components delays, i.e. transmission delay ($\text{TranD}_{*}^{\text{Req/Res}}$), propagation delay ($\text{PropD}_{*}^{\text{Req/Res}}$), queuing delay ($\text{QueuD}_{*}^{\text{Req/Res}}$), and processing delay ($\text{ProcD}_{*}^{\text{Req/Res}}$), and how the value of each one could be calculated are defined in Section 6.7.3.1.
7.10.3.2. Experimental Evaluation

This section reports the experimental evaluation of the PET of MR-CAuthN protocols and compared with the KRB. In particular, we carried out this experiment to measure and compare the PET of these protocols using the Riverbed Modeler simulation tool [121].

Protocol and Network modelling

The Riverbed Modeler simulation tool version 17.5 is used for modelling the MR-CAuthN and KRB protocols. This tool is used because of its advanced simulation capabilities which are discussed in Section 6.7.3.2. Figure 7.27 shows the C-RM_AuthN protocol...
protocol modelling as an example of how MR-AuthN protocols are modelled using the Riverbed Modeler simulation tool. The figure shows a summary of how the task of the C-RM_AuthN protocol and its phases are defined and configured in both the Application Config-Utility Object (indicated in the figure as an Application Definition) and the Task Config-Utility Object (indicated in the figure as a Task Definition), respectively, and how the task is linked to the Client profile using the Profile Config-Utility Object (indicated in the figure as a Profile Definition). However, the process of simulating a protocol is fairly complex. For this reason, prior to starting the configuration of these configuration-utility objects, it is prudent to carefully examine and specify all network-utility objects for the protocol. Modelling the protocol using different network nodes or network media may provide different PET.
To examine the PET of our MR-CAuthN protocols and the KRB protocol, we design two identical network models in terms of the number of the intermediate nodes used and the types of the channels connecting the MR-AuthN Components. This is because as explained in the previous section, the PET consists of a number of delays that are experienced in each communication round and the values of these delays are largely based on the network model which describes such nodes and channels.

The first network model is used to measure the PET of the MR-CAuthN protocols. As depicted in Figure 7.28, it is composed of two Local Area Networks (LANs). The first one represents the network of the MR-Client site. It hosts an MR-Client node which runs the CJ-AuthN Agent, and a Layer 3 Switch which runs as a network switch and router. The Layer 3 Switch is connected to the MR-Client using a Gigabit Ethernet Cable and to the Internet using T1 cable. The second LAN represents the MR application site. It hosts an Authentication Server which runs the G-AuthN Service, a Resource Manager which runs the RM-AuthN Relay Module, a Name Node which runs the NN-AuthN Module, a Data Node which runs the DN-AuthN Module, and a Layer 3 Switch which runs as a network switch and router. The four MR-AuthN Components, i.e. the G-AuthN Service, RM-AuthN Module, NN-AuthN Module, and DN-AuthN Module, are connected to each other using Gigabit Ethernet Cables via the Layer 3 Switch which is connected to the Internet using T1 cable.

The second network model is used to measure the PET of the KRB protocol. As depicted in Figure 7.28, it consists of two LANs of two remote sites. The first one is of the MR-Client site and it hosts the same network nodes and media of the first network model. The only difference from the first model is that the MR-Client runs the KRB-Client rather than the CJ-AuthN Agent. The second one, i.e. LAN, represents the MR-Application site. It consists of an Authentication Server which runs the KRB_AS, a Ticket Granting Server which runs a KRB_TGS, an Application Server which runs the KRB_AP (i.e. KRB_RM or KRB_NN), and a Layer 3 Switch which runs as a network switch and router. Similar to the first network model, the KRB_AS, KRB_TGS and KRB_AP, are connected to each other using Gigabit Ethernet Cables via the Layer 3 Switch which is connected to the Internet using T1 cable.

In each network model, the Layer 3 Switch in each site works as an edge to connect the LAN of the site to the Wired Area Network (WAN) device.
Network Model Validation and Evaluation Results

The two network models designed above are validated so that any collected results are considered reliable. The validation is done by comparing the PET values calculated using the theoretical models described in the previous section with the respective results obtained from the execution of the C-RM_AuthN protocol and the KRB protocol on the first and second network models under the same conditions. The two network models, depicted in Figure 7.28, are run on the following conditions:
- The network does not have any background traffic, i.e. the only traffic in the network is the messages of the executed protocol. By assuming a 'zero' level of background traffic, queuing delays can be omitted, so the queuing delay is set to zero when calculating the theoretical results.

- In each site, every MR-Authorization Component node is placed at a fixed distance of 1m of a Gigabit Ethernet cable from the Layer 3 Switch, and the Layer 3 Switch is placed at a fixed distance of 10 Km of T1 cable from the Internet.

- The propagation speed in each cable (i.e. Gigabit Ethernet and T1 cable) is set to 300,000,000 meters per second.

- The bandwidth of each Gigabit Ethernet cable is set to 1Gbps, and each T1 cable is set to 1.544Mbps.

- The amount of processing time taken by each MR-Authorization Component of each protocol is based on the processing time calculated in the first experiment.

- The amount of the time introduced by the user to enter his credentials is a zero second.

- The length of each message of the protocols is set to the length which is measured in the first experiment for every respective message.

The comparison is shown in Figure 7.29. From the figure it can be seen that the results from the theoretical model are close to those from the simulation run. The gap between the theoretical and experimental result is only about 6%. This gap could be due to the processing delay taken by MR-Authorization Components to encapsulate the request and response messages before a message is transmitted into the network media.

![Figure 7.29 Comparing the theoretical and network models results of the PET of both the C-RM_AuthN protocol and the KRB protocol](image-url)
In the two theoretical network models, it is assumed that there is no background traffic except for the traffic of the protocol being executed on the network model. However, in any network implementation, i.e. real/practical networks, the network usually has some existing background traffic comes from other services, and the MR-AuthN Components may receive simultaneously authentication requests. Thus, the PET for the MR-CAuthN and KRB protocols is evaluated when a background traffic is applied in the network models. The same background traffic is applied for both network models. In addition, we investigate the impact of the number of concurrent requests, served by the MR-AuthN Components in parallel, on the PET of each protocol. The number of concurrent requests varied between one and one hundred requests. Figure 7.30 shows the experiment results of the PET for the C-RM_AuthN protocol, C-NN_AuthN protocol and KRB protocol versus the number of concurrent requests. It can be seen from the figure that the PET for each protocol is proportional to the number of concurrent requests, and surprisingly the KRB protocol has the largest PET, although it has a lowest overall processing time (as discussed in Section 7.10.1.2) and the smallest size of the total communication overhead TCO (see Figure 7.23). Also, it can be seen from Figure 7.30 that the PET values of both the C-RM_AuthN protocol and the C-NN_AuthN protocol are respectively about 55% and 21% less than those of the KRB protocol. This is due to that the KRB protocol has a larger number of communication rounds (CR) with a largest communication overhead (CO) exchanged between the MR-AuthN Components over the Wired Area Network (WAN) than the MR-CAuthN protocols. As mentioned in the theoretical model (Section 7.10.3.1), the C-RM, C-NN_AuthN protocols and KRB protocol each contains three rounds. As shown in Figure 7.31, the KRB protocol has the three communication rounds over the WAN with more than 2600 bytes of communication overhead, whereas the C-RM_AuthN protocol has only one communication round with about 1390, and the C-NN_AuthN protocol has two communication rounds with 2260 of communication overhead, respectively.
Figure 7.30 PET of C-RM, C-NN_AuthN and KRB protocols vs. the number of concurrent requests

Figure 7.31 The communication rounds and their communication overheads cross the WAN for the C-RM, C-NN, and KRB protocol

The Figures show that the PET is significantly influenced by the communication overhead of each protocol exchanged over the WAN. As discussed in Section 7.10.3.1, the end-to-end delay for each communication round is a function of two delays: the delay caused by the two MR-AuthN Components of each round, i.e. $Delay_{MRC_r}$, and the delay caused by the network connects the two MR-AuthN Components, i.e. $Delay_{NET_r}$, where

$$Delay_{MRC_r} = \sum_{c=1}^{2} \left( QueueD_{c}^{Req/Res} + ProcD_{c}^{Req/Res} \right) + TranD_{c=1}^{Req} + TranD_{c=2}^{Res},$$

and $Delay_{NET_r} =$
\[
\sum_{i=1}^{n} \left( Q_{\text{ueu}D_{i}}^{\text{Req}/\text{Res}} + P_{\text{roc}D_{i}}^{\text{Req}/\text{Res}} + T_{\text{ran}D_{i}}^{\text{Req}/\text{Res}} + P_{\text{rop}D_{i}}^{\text{Req}/\text{Res}} \right) + P_{\text{rop}D_{i}}^{\text{Req}} + P_{\text{rop}D_{i}}^{\text{Res}}.
\]

The observation that can be made from these figures and equations is that the delay that could have largely affected the PET is the transmission delay rather than other delays components in each communication round. This is can be explained as follows:

First, the propagation delay of each communication round is proportional to the distance (i.e. the length of the network cables) between any two nodes in each network model and it is not influenced by the bandwidth of the network media or the size of the communication overhead. This is because; (i) the length of the LAN and the WAN’s cables are fixed to 1m and 10km each, respectively, (ii) the number of intermediate nodes is identical for the two network models, and (iii) the propagation speed is constant and equivalent to the speed of light. Second, the transmission delay of each communication round is largely proportional to the size of the communication overhead (CO) as it is a function of and proportional to the size of the communication overhead and the data transmission rate of the network media \( (T_{\text{ran}D_{i}}^{\text{Req}/\text{Res}} = \frac{\text{CO}}{\text{DTR}}) \). When the bandwidth is fixed for the two network models (which are used to compare the PETs for all the protocols), and as the communication overhead in each round increases the transmission delay experienced by an MR-AuthN Component and by an intermediate node increases, so the PET increases.

Third, the time that communication overhead (a request or a response message) spends in the transmission-queue to be transmitted to the network media depends on the transmission delay and the bandwidth utilisation of each network media. If the background traffic which utilises the bandwidth of the network media is fixed, the time is taken to transmit each request or response message may increase the transmission-queuing delay. For example, if, at a given time instance, an MR-AuthN Component has two messages of 10K byte each in the transmission-queue and these two messages will be transmitted one at a time. There is a background traffic that utilises a 50% of the bandwidth of the network media, and this network media has a data transfer rate of 1K byte per second. When the MR-AuthN Component starts at time = 0 to transmit the first message, this means that the second message transmission-queuing delay will be doubled. This is because the first message takes about 20 seconds rather than 10 seconds before the transmission of the first bit of the second message is started. As a result the transmission delay increases the
transmission-queuing delay in each round, and so that the PET value is influenced and it increases accordingly. Fourth, the time is taken to process each received request or response message may increase the arrival-queuing delay. As the longer the message, usually the longer the processing time is, and this can add into the queuing-delay as well. However, this addition should be negligible as the received request and/or response messages are processed in parallel. So that the influence of message arrival queuing and processing delays on the PET is too small or negligible.

The above analysis shows that when the MR-AuthN Components of each communication round process the received request or response messages in parallel, the PET is largely influenced by the communication overheads exchanged over the WAN. However, when the received request or response messages are processed sequentially by an MR-AuthN Component, then the PET will be high. In practice, this may take place when the arrival rate of the request or response messages is higher than the rate the MR-AuthN Component takes to process the received request or response messages. This raises a question that at what value of the message arrival rate would the arrival-queue starts to build up. In other word, what is the turning point (i.e. threshold) beyond which the arrival-queue starts to build up? The threshold or the turning point is the point beyond which the MR-AuthN and KRB protocols are expected to perform slowly (i.e. have a longer PETs).

To study and find out the turning points of MR-AuthN Components of all the protocols, i.e. C-RM, C-NN_AuthN and KRB protocols, it is assumed that each MR-AuthN Component uses a first-come-first-served queue and it process one request or response message at a time. The result of such study is plotted in Figure 7.32 and Figure 7.33. Figure 7.32 shows five sets of time delays (A, B, C, D, and E) experienced by the MR-AuthN Components of MR-AuthN protocols against different messages arrival rates. The first two sets, i.e. A and B, are experienced by the RM-AuthN Relay Module when it receives the NJ-AuthN Request messages and JA-AsserT Response messages, respectively. C is experienced by the NN-AuthN Module when it receives WJ-AuthN Request messages, D is experienced by the DN-AuthN Module when it receives R-WJ-AuthN Request messages, and E is experienced by the G-AuthN Service when it receives JA-AsserT Response messages. As can be seen from the figure, the turning points beyond which the messages arrival-queues start to build up are respectively around the arrival rates of 346000, 290, 65, 6750 and 105 messages per second for NJ-AuthN Request, JA-AsserT Response, WJ-AuthN Request, R-WJ-AuthN Request, and JA-AsserT Request. The messages arrival-
queuing delays for the five sets increase steadily when the arrival rates go beyond these points. This is because, from these points onwards, the average interval between the message arrivals for each set is shorter than the average processing time for the received message. In other words, when the arrival rate for (i) the NJ-AuthN Request is about 346000 requests/second, the average interval between two NJ-AuthN Requests is about 0.00289 ms, (ii) the JA-AsserT Response is about 290 responses/second, the average interval between two JA-AsserT Responses is about 3.448 ms, (iii) the WJ-AuthN Request is about 65 requests/second, the average interval between two WJ-AuthN Request is about 15.38 ms, (iv) the R-WJ-AuthN Request is 6750 requests/second, the average interval between two R-WJ-AuthN Request is about 0.148 ms, and (v) the JA-AsserT Request is 105 requests/second, the average interval between two JA-AsserT Requests is about 9.52 ms. These average intervals are respectively shorter than the average processing times of 0.003 ms, 3.5 ms, 15.5 ms, 0.149 ms and 9.6 ms for the respective received messages.
Figure 7.32 Turning points for the RM, NN, DN and AS

On the other hand, Figure 7.33 shows three sets of time delays experienced by the KRB authentication components against different messages arrival rates. The first set is experienced by the KRB_AS when it receives the KRB_AS_REQ messages. The second set is experienced by KRB_TGS when it receives the KRB_TGS_REQ messages, and third set is experienced by the KRB_AP when it receives the KRB_AP_REQ messages. The trends shown in this figure are similar to those shown in Figure 7.32. The difference is that the threshold beyond which the messages arrival-queues start to build up are around the arrival rates of 3290 KRB_AS_REQ message per second, 3595 KRB_TGS_REQ message...
per second, and 4635 KRB_AP_REQ message per second. The messages arrival-queuing delay for the three sets increase steadily when the arrival rate goes beyond these points. This is because, from these points onwards, the average interval between the message arrivals for each set is shorter than the average processing time for the received message. In other words, when the arrival rate for (i) KRB_AS_REQ is about 3290 requests/second, the average interval between two KRB_AS_REQ messages is about 0.304 ms, (ii) the KRB_TGS_REQ is about 3595 requests/second, the average interval between two KRB_TGS_REQ messages is about 0.2781 ms, (iii) the KRB_AS_REQ is about 4635 requests/second, the average interval between two KRB_AS_REQ messages is about 0.2157 ms. These average intervals are respectively shorter than the average processing times of 0.306 ms, 0.279 ms, and 0.216 ms for the respective received messages.

The observation that can be made from Figure 7.30 to 7.33 is that though the KRB protocol has larger PET value than the MR-AuthN protocols, the KRB authentication components resulted in lower values of turning points. In other word, the KRB authentication components can process a larger number of concurrent authentication requests before being reached to the threshold. To minimise the message arrival-queuing delay, the system administrator may consider the use of multiple servers run the MR-AuthN Components to process the messages received.

![Figure 7.33 Turning points for the AS, TGS and AP](image-url)
7.11. Chapter Summary

This chapter has presented a set of new protocols to support MR-Client authentication and MR-Job Components identification in the MR application. These protocols are the MR-CAuthN and JCPCrE protocols. MR-CAuthN protocols allows the MR application to authenticate securely and efficiently a job submission of a remote MR-Client. MR-CAuthN protocols use multi-factor authentication tokens, and each interaction of the job submission is identified by using an authentication token. The JCPCrE protocol allows an MR-Client to securely and efficiently identify the MR-Job Components assigned to his/her MR-Job domain without any involvement by the user, so that the Name Node can validate each request to access the job resource of the MR-Client on his/her behalf through the job execution. Through the protocols executions, an MR-Client can obtain his/her secondary authentication tokens and his/her MR-Job Components are issued with the primary authentication tokens. The key structure design and distribution of these tokens supports a reliable MR-Client and MR-Job Components authentication, ensures the confidentiality and integrity protection of the tokens, and supports multi-factor authentication where different factors are generated securely and randomly and supplied by different entities.

The chapter has also presented the security analysis and performance evaluation for this set of protocols. The analysis demonstrates that the protocols satisfy the security requirements specified. The MR-CAuthN protocols are formally modelled using the AVISPA tool to further verify the security properties of the protocols. The validation result shows that security properties of the MR-CAuthN protocols is preserved and the protocols are secure; no attacks are found. The performance evaluation is carried out in term of the computational and communication overhead costs and the protocol execution time (PET) of this set of protocols, and compared with the KRB protocol. The performance evaluation shows that though the MR-CAuthN protocols have about 27% (for C-RM_AuthN protocol) and 4% (for C-NN_AuthN protocol) additional communication overheads than the KRB protocol, the former performs better in term of PET.

The following chapter concludes this thesis and outlines the future work.
Chapter 8

8. Conclusion and Future work

Authentication is a major security service in modern distributed computing frameworks such as MR application framework (i.e. MR application for short). Using ineffective or inadequate authentication service for MR application makes the job submission and execution (i.e. MR-Job execution) vulnerable to different security threats and attacks. The focus of this thesis is to address this open issue, by designing a secure, efficient and practical authentication solution for MR application. The need to design such solution comes from the fact that the current MR authentication proposals are not applicable for MR application deployed in a distributed and resource-sharing environment, as they do not adequately capture the characteristics of MR application and its authentication requirements when it is deployed in such environment.

Designing an effective MR authentication solution that captures the very characteristics of MR application and meets its functional and security requirements is a complex and challenging task. This is due to a number of reasons. Firstly, MR clients typically access the MR application remotely via the Internet, so there is an issue of how to establish authentication credentials that can facilitate a trust relationship between MR clients and the MR application without requiring the clients to make a physical presence to a registration authority of MR provider. Secondly, submitting a remote MR client’ job to the MR application involves multiple interactions with different MR components. Thirdly, when submitting an MR job, the data of the job are partitioned, and the partitioned data may be stored across multiple distributed MR components, called MR server nodes. These MR server nodes could be managed by single or multiple administrative domain and/or located in different geographic locations. In other words, the writing and reading of these data may be carried out over local or wide-area networks, so there is a need to securely identify the MR clients and components that write and read the data. Fourthly, executing a job submitted by a remote MR client involves the use of multiple software components assigned to the job. Multiple software components serving a single or multiple jobs from a single or multiple MR clients may be also run/hosted in a single or multiple MR server nodes. Fifthly, MR server nodes are shared among other MR clients or jobs submitted by other MR clients. This shared environment brings more threats and risks. The design of the authentication solution should take these issues into account, in terms of how the distributed set of
components, some are shared by different jobs but others are dedicated in serving a particular job, should be identified and authenticated, when the authentication of the components should be carried out and how authentication credentials may be established or distributed.

8.1. Thesis Contributions

To address this complex and challenging task, i.e. designing an effective MR authentication solution for MR application, this thesis has presented the following contributions and discoveries:

- A Generic MR Computational (GMC) model is presented to capture the key features and properties of the MR model implementation, i.e. the MR application framework. The GMC model was the first step toward (1) identifying the limitation and the gaps in the existing MR being deployed in a complex, distributed and resource-sharing environment, and (2) realizing the classification of MR components and their interactions based on the nature and the role of the MR components being involved in MR-Job execution in such environment. Also, based on this model, a security analysis has been conducted to identify the threats and the attacks that are more closely related to the MR-Job execution and understand the implications of such threats on the provision of the GMC model. This analysis has led to specify a set of MR authentication requirements, and these requirements are used to guide the design of a secure, efficient and practical authentication solution for MR application.

- A novel layered approach, called MR Layered Authentication Model (MR-LAM), has been proposed to address MR authentication in such complex environment. More specifically, this approach embeds two novel ideas: a layered approach to tackling the complex task of authentication in MR and a virtual job domain based approach to accommodating the need for authentication of a set of distributed components (a user, the user’s client, and the MR components serving a job submitted by the client). In other words, with this layered approach, MR components are classified into two groups and tackle the authentication of these two groups of components at two different layers. The first layer is called the MR-Infrastructure domain authentication, and it is for the authentication of MR components that serve multiple jobs submitted by different clients. These MR components, the so-called MR-Inf. Components, are not job specific and the examples of these components are MR master nodes (i.e. the Resource Manager and the
Name Node). The other layer is called MR-Job domain authentication, and it is for the authentication of the MR components that are created or invoked specifically for a particular job, so-called MR-Job Components, and examples of these components are the MR components that run on the MR slave nodes (i.e. Job Tracker, Task Tracker, and Map and Reduce Tasks). This set of components, i.e. MR-Job Components, is a job specific and their invocations and existence are purely for serving this particular job.

MR-LAM meets the authentication requirements and facilitates the complex authentication task of MR in a modular (or systematic) and flexible manner. A modular means that any change or upgrade in an authentication method used at one layer does not affect the authentication of the other layer. Flexible means that MR-LAM could run in (1) a small scale trusted network in which the authentication secrets (i.e. keys) of the MR-Inf. Components could be preconfigured, i.e. an IT administrator could manually install and maintain the authentication keys of the MR components of the MR-Inf. domain, and (2) untrusted network (or when dynamic MR-Inf. Components involved) in which more complex authentication approach could be used without influencing the job domain authentication layer.

- A novel framework, named as the Virtual Domain based Authentication Framework (VDAF) is designed to implement the idea of using a virtual job domain based approach. The benefits or the implications of implementing VDAF design are four folds. First, each job submitted by a client has its own set of authentication and verification tokens that are used to authenticate the distributed set of MR components assigned and involved in executing the client’s job. This set of MR components is isolated from other sets (i.e. other jobs’ components) by issuing them job-specific authentication tokens that are used to identify and validate their job-resource request and response messages during the MR-Job execution. It is the interest of the client to protect the access to their job-resources. The client is involved in managing his job authentication tokens by keeping the secrets of the authentication tokens protected while they are used in the MR application. In other words, the real threat during the MR-Job execution in MR application is two types; external threat and internal threat. External threat is the threat originating from outside the MR resources, i.e. it is the threat originated by external clients that do not have the right to use the MR resources. Internal threat is the threat originated from within the MR entities, i.e. from MR-Clients and Components that have the right to use the MR resources but do not have the right to use the other jobs-resources. One client may want to steal the other client data (a job-resource), this
problem is solved by separating the secrets used for different jobs. Such attack could be from malware running in the MR application, this layer of protection (i.e. job domain authentication layer) is added, where the job authentication secrets are encrypted while they are distributed and used during MR-Job execution in the MR application. This is achieved using VDAF.

Second, a novel authentication method, called Password and Token-based Multi-point Multi-factor Authentication (PT2M-AuthN) method, is proposed in the VDAF design. This method uses two main ideas; the principle of separation of duty-and-credential and key wrap-and-swap operation to support mutual authentication. The novelty in the PT2M-AuthN method is that the other authentication solutions use multi-factor authentication methods for a user to system authentication, but in the VDAF, our multi-factor authentication method, i.e. PTSM-AuthN method, is used for system authentication (i.e. inter-software components authentication) besides the user to system authentication, and more importantly this is achieved without introducing any additional control messages and without any additional secure connection, such as SSL. In other words, by implementing this method, VDAF performs its authentication function, which includes the delivery of the authentication and verification tokens, using the original MR architecture and messages and no modification to the existing MR architecture is required. Also, by using this method, when MR-Job domain’s components (i.e. MR-Job Components) request job-resources that are located in different MR-Inf. domain’s components (i.e. MR-Inf. Components), there is no need to establish a trust between the MR-Job domain and the MR-Inf. domain to authenticate the MR-Job Components of the first domain to the MR-Inf. Components of the second domain which could be administrated separately in different location.

Third, based on the ideas used in the PT2M-AuthN method, VDAF imposes authentication control at every interaction where there is a request for a job resource made by a client or an MR components during the entire cycle of an MR-Job execution. For each client identified to the MR application, i.e. an MR-Client, four authentication tokens are issued; two primary and two secondary, and for each MR-Job Component there are two authentication tokens; one primary and one secondary. For each of these authentication tokens there is a corresponding verification token that is used to verify the authentication token against. These tokens are used to authenticate the job submission and execution. Fourth, the VDAF uses a delegation feature that gives the MR-Job Components the access to the MR-Client data, so the MR-Client does not have
to propagate his authentication credential to the MR-Job Components, and/or to authenticate each time an MR-Job Component accesses his job resources on his behalf.

Fifth, VDAF uses a decentralized approach to building trust, and do so by using a distributed set of components to facilitate the authentication functions and by letting different trusted entities to issue multiple authentication tokens, so that (1) no single entity is responsible for the authentication functions, and (2) authentication-exchanges between the MR entities and a centralised Authentication Server (AS) (as used in centralised authentication approaches) can be prevented. In other words, this distributed approach can prevent overloading a single AS with authentication protocol messages and the associated processing overheads, thus reducing the risk of creating a performance bottleneck that could be caused by the AS. In this sense, VDAF could scale similarly as the MR application. In addition to these benefits, the way that the MR-LAM and VDAF are designed allows the MR application to be provided as an Application as a Service (AaaS). This is because the design takes a layered approach to authentication, where the management of credentials are separated based on the nature of the MR components involved. An MR-Client does not have to install or configure the MR-AuthN Components of the MR-Inf. and MR-Job domains, nor be involved in the task of managing the authentication credentials or secrets of the MR-Inf. Components. Rather, an MR-Client, the owner of a job, needs only to be responsible for the task of managing the authentication credentials for the MR-Job Components assigned to executing his job (an MR-Job domain).

To facilitate the VDAF functions, four sets of protocols called LVAP suite are proposed and the authentication method and ideas used in the design are evaluated. The first set contains one protocol, the CPCrE protocol, to perform client identification function. In other words, the protocol is designed to identify a new remote client as an MR-Client by issuing the primary authentication and verification tokens and forming the local and removable parts of his authentication credential, so that the client becomes a member of his job domain. The second set consists of MR-CAuthN protocols and JCPCrE protocol to perform MR-Client authentication and MR components identification functions, respectively. The MR-CAuthN protocols are three protocols, named as User to MR Application Client Authentication (U-AC_AuthN) protocol, MR-Client to Resource Manager Authentication (C-RM_AuthN) protocol, MR-Client to Name Node authentication (C-NN_AuthN) protocol. These three protocols are designed to allow, respectively, the VDAF to (1) authenticate the user to the MR Application Client locally
in the client machine, (2) to mutually authenticate the MR-Client and the Data Processing (DP) cluster of MR and generate the first part of the secondary authentication and verification tokens of the MR-Client, and (3) to mutually authenticate the MR-Client and the Distributed File System (DFS) cluster and generate the second part of the secondary authentication and verification tokens of the MR-Client. The JCPCrE protocol is designed to identify a set of MR components as MR-Job Components to the MR-Client’s job domain by issuing them primary authentication and verification tokens, and distributing the tokens to the respective MR components. The third set of the LVAP suite consists of MR-JCAuthN protocols, and they are Job Tracker to Name Node authentication (JT-NN_AuthN) protocol, Task Tracker to Name Node authentication (TT-NN_AuthN) protocol, Reduce Task to Name Node authentication (RT-NN_AuthN) protocol, and Reduce Task to Task Tracker authentication (RT-TT_AuthN) protocol. They are designed to enable the VDAF to perform a mutual authentication between (1) each MR-Job Component assigned to the MR-Client job and the DFS and (2) a pair of MR-Job Components, and issue the MR-Job Components’ secondary authentication and verification tokens. The fourth set of LVAP suite consists of three protocols and they are Original-Data authentication (OD_AuthN) protocol, Intermediate-Data authentication (ID_AuthN) protocol, and Final-Data authentication (FD_AuthN) protocol. These protocols enable the VDAF to ensure the authenticity of the three data sets of the MR-Client’s job, i.e. input data, intermediate data and output data, through an MR-Job execution.

The protocols have been evaluated in terms of security and performance and they are compared with the most related existing work. The security analysis shows that the CPCrE protocol achieves a secure remote client identification and it supports a stronger MR-Client authentication to the MR application than the existing work by designing a novel key structure. The key structure enables two-factor mutual authentication between the MR-Client and the MR application using MR-CAuthN protocols. The MR-CAuthN protocols prevent MITM attacker from launching a DoS attack against the user account. The MR-CAuthN protocols allow MR application to mutually authenticate each interaction of MR-Client’s job submission using a multi-factor authentication based approach rather than a password single-factor authentication based approach, which is used in the existing work. The security analysis also shows that the VDAF functionality is secured. This is achieved by providing confidentiality and integrity protections for the tokens and confidentiality and authenticity protections for the exchanged messages. This
enables the protocols to withstand threats and attacks, such as replay and impersonation attacks, that may be mounted against the VDAF functionality.

In term of performance, the evaluation results show that our authentication method used in the design introduces more computational and communication overheads than the most existing authentication method (i.e. the one used and adopted in Malley’s protocol). The additional cost is the result of providing a more secure authentication service. However, it is worth mentioning that our method provides a better performance in term of the Protocol Execution Time (PET); our PET is 20% less than that of the Malley’s method. This is due to the fact that our method achieves mutual authentication without introducing or using any additional control messages and without requiring any additional secure connection. In other words, our authentication method does not require the transmission of authentication secrets. Therefore it does not require the establishment or use of a secure connection between the claimant and the verifier when the later receives an authentication request from the claimant. This is because the authentication secrets are used to identify and secure the authentication request and response messages themselves. This is an attractive feature particularly for a distributed computing environment that typically uses Wide Area Networks (WANs), such as the Internet, to connect a distributed set of hosts and that communication overheads and delays are often non negligible.

To the author’s best knowledge, the MR-LAM solution is the first one that has taken layered approach to the complex task of authentication for a distributed computing framework. It should be emphasised that, although the MR-LAM is designed for MR application, it is applicable to other distributed computing frameworks that are based on master-slave architecture and are characterised by the following: There are a set of software components that are managed and run by another set of software components (i.e. system-software components), and among the system-software components there are two types of components, master and slave. A client submits a job to the master components. For every job submitted by the client, (1) a set of software components are created or invoked (i.e. assigned) when the job is submitted, and they are terminated or reassigned when the job execution is completed, (2) the client does not have control nor interactions with these software components, and (3) the software components assigned to his job and its data are

\[\text{Secure in term of message origin, integrity and confidentiality.}\]
hosted on and distributed to a set of slave system-software components that could be managed by different domains and/or located in different locations.

8.2. Future Work

The author has the following recommendations for future work.

- **Performance evaluation of MR-Job Components authentication:** As discussed in Chapters 1 and 5, the ideas and measures which are used for MR-JCAuthN protocols are identical for those used for C-NN_AuthN protocol. However, it would be interesting to simulate the MR-JCAuthN protocols and assess the effects of different number of MR-Job Components in the protocol execution time and compared with the related work.

- **Authentication of MR-Data:** While this thesis identified the need of MR-Data authentication and how such authentication may be carried out, further research is necessary to investigate how to ensure the authenticity protection for all the data types of the MR-Client's job (MR data). Typically, there are three types of data; structured, semi structured and unstructured data that could be processed by the MR application, and performing authenticity protection for MR data may vary from data type to another.

- **Authentication of monitoring the progress of MR-Job:** The MR-CAuthN protocols, discussed in Chapter 7, are designed for authenticating an MR-Client sending a request to MR application to submit a new job. However, the client may send a request merely to monitor the progress of the execution of the job has been submitted. The MR-CAuthN protocols can be extended to accommodate the authentication requirements and measures of this request. Further study is needed to determine such requirements and measures. This is because monitoring the progress of the execution of a job which has already been submitted involves different type of interactions between MR components from those for submitting the job, accessing the job-resources, and allocating the MR resources.

- **Authentication for inter MR-Inf. domains:** The design of the VDAF (Chapter 5) assumes that MR-Inf. Components are in one MR-Inf. domain, and the MR-Job domain obtains the infrastructure domain-secret key which is shared between these components once they have been authenticated successfully to each other. What is now needed is more study about MR-Inf. domain authentication involving a cross-domain
authentication. Establishing such secret cross infrastructure domains is a challenging task, as different MR-Inf. Components could belong to different domains and each domain may use different authentication solutions or methods.

- **Finally**, it would be interesting to evaluate the VDAF in a real system setup. This may include single or multiple job domains, where authentication assurance level of the MR infrastructure domain is already in place, to assess VDAF against the adopted MR authentication method (Malley’s method) and see the real cost of this solution.
Reference


295


Appendix A

This section presents activity diagrams that summarise the activities performed by each MR-AuthN Component while executing the CPCrE protocol and both the MR-CAuthN and JCPCrE protocols.

A.1. CPCrE protocol
A.2. MR-CAuthN and JCPCrE protocols
CJ-AuthN Agent

CJ3
Receive the WJ-AuthN Response object

Verify the response freshness

Valid

No

Yes

User password-secret key

Compute the HMAC signature of the response message and verify it

Valid

No

Yes

Writing job-secret key

Decrypt the encrypted contents of the response message and verify the received authenticator in the message

Valid

No

Yes

MR-Client username

MR-Client timestamp

Generate an MR-Client authenticator for R-WJ-AuthN Request

MR-Job identifier

Writing job-secret key

- Encrypt the authenticator
- Construct a R-WJ-AuthN Request message and compute its HMAC signature.

Send the R-WJ-AuthN Request object

DN2

CJ4
Receive the R-WJ-AuthN Response object

Verify the response freshness

Valid

No

Yes

Writing job-secret key

Compute the HMAC signature of the response message and verify it

Valid

No

Yes

Writing access-secret key

Decrypt the encrypted contents of the response message and verify the received authenticator in the message

Valid

No

Yes

Dynamic login-secret key

- Generate new login authentication and local verification tokens
- Encrypt and store the local and removable parts of the MR-Client authentication credential

Job submission-secret key

- Encrypt the primary authentication tokens of the MR-Job Components (PT_
- Construct a Submit message and compute its HMAC signature

Send the Submit object

RM3
G-AuthN Service

AS1
Receive the JA-AssertT Request object

Verify the request freshness

Valid
Session password-secret key
No
AS's private key

Compute the HMAC signature of the request message and verify it

Valid
User password-secret key
No

Decrypt the encrypted part of PT_2^T and verify the client credential validity period

Valid
Client master-secret key

End

Encrypt the authenticator received in the JA-AuthN Request along with SK_1^T and ST_2

Construct the first part of the JA-AssertT Response message and compute its HMAC signature

Infrastructure domain-secret key

Client master-secret key

User password-secret key

Generate a writing-job authentication information (SK_1^T, ST_1, and SYT_2)

MR-Client authenticator for JA-AuthN Request

Decrypt and verify the MR-Client authenticator for JA-AuthN Request

Valid
MR-Client IP address

MR-Client timestamp

Send verification code to user via his email

User

Verify the verification code

Valid
Yes

End

No
Verification code
**DN-AutN Module**

**DN1**
- Receive the Notify object
- Verify the Notify message freshness and integrity
  - Valid: Yes → **NN3**
  - No → End

**DN2**
- Receive the R-Wj-AutN Request object
- Verify the request freshness
  - Valid: Yes → Decrypts the wiring access-secret key received in the request
    - Compute the HMAC signature of the request message and verify it
      - Valid: Yes → Send the WJ-AutN Response object
        - Encrypt the authenticator received in the R-WJ-AutN Request
          - Construct the WJ-AutN Response message and compute its HMAC signature
        - Writing access-secret key
      - No → End
    - No → Writing access-secret key
  - No → Writing access-secret key
- MR-Client username
- MR-Job identifier
- MR-Client timestamp
  - Valid: Yes → End
  - No
- Writing job-secret key
- Encrypt the encrypted part of $\text{SVT}_J^C$
- Decrypt and verify the MR-Client authenticator for R-WJ-AutN Request
- Writing job-secret key