MASTER

A2RiD
An Anonymous RemoteID-compliant Group Signature Scheme for Commercial Drones

Wisse, Eva M.C.

Award date:
2022

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
A²RID: An Anonymous RemoteID-compliant Group Signature Scheme for Commercial Drones

Master’s Thesis

Eva Wisse

Graduation Supervisor:
Dr. S. Sciancalepore

Committee Members:
MSc D. R. George
Prof. dr. ir. N. Meratnia

Final version

Eindhoven, October 2022
A²RID: An Anonymous RemoteID-compliant Group Signature Scheme for Commercial Drones
Abstract

The number of Unmanned Aircraft Systems (UAS) is ever-increasing, making it difficult for regulatory authorities, such as the U.S.-based American Federal Aviation Administration (FAA), to regulate air-traffic and keep the airspace safe. In the U.S., by introducing RemoteID, the FAA integrated UASs into the National Airspace System. RemoteID requires all UASs to send their location, and provide their identity every second, at most. Although this would increase the situational awareness the FAA has in the airspace; it also comes with its consequences, especially in terms of anonymity. Indeed, the RemoteID framework is privacy-invasive and requires UASs to broadcast in plain-text the ID and location of the drone.

With this thesis, we wish to design and implement an anonymous version of RemoteID, still allowing an observer to locate UASs and authenticate the RemoteID messages, but without disclosing the drone’s unique identifier. If necessary, in case of invasion, the FAA can still disclose the identity of the UAS and take action in case of breaches. The anonymous RemoteID version has been implemented, evaluated, and tested (e.g. energy, memory, time) on a constrained platform, i.e., the ESPcopter, based on the ESP8266 board. We demonstrated that it is possible to perform anonymous Remote Identification on such a constrained platform within as quick as 0.16 seconds on the transmitting side, indicating that the proposed scheme fulfils the requirements set by the FAA and increases the anonymity of UASs.
Acknowledgement

I would like to take this opportunity to express my gratitude to some people who were involved in this project. First of all, I would like to express my deepest gratitude to Dr. S. Sciancalepore for being my overall supervisor, who provided his knowledge and expertise throughout this project. Additionally, I would like to thank my committee members MSc D. R. George and Prof. dr. ir. N. Meratnia for agreeing to serve on my defence committee.
# Contents

Contents vi

Acronyms ix

List of Figures xii

List of Tables xii

1 Introduction 1
   1.1 Motivation .................................................. 1
   1.2 Problem Description ......................................... 2
   1.3 Project Description .......................................... 5
   1.4 Report Outline ............................................... 5

2 Background and Preliminaries 7
   2.1 RemoteID .......................................................... 7
      2.1.1 Broadcast .................................................. 7
      2.1.2 Types of RemoteID ........................................ 9
   2.2 Drone Remote ID Protocol (DRIP) Working Group ............ 10
   2.3 Bilinear Groups ............................................... 11
   2.4 Elliptic Curves for Security .................................. 12
   2.5 CPA and CCA2 Anonymity ..................................... 14
   2.6 Public Key Encryption ........................................ 15
   2.7 Digital Signatures ............................................ 15
   2.8 Non-interactive Zero Knowledge Proofs ....................... 16

3 Related work 17
   3.1 Existing RID Protocols ....................................... 17
   3.2 Group Signatures Schemes .................................... 17

4 Dynamic Group Signatures of A^2RID 21
   4.1 Entities in A^2RID .............................................. 21
   4.2 General Description A^2RID ................................... 22
   4.3 Setup ............................................................. 23
   4.4 Join ............................................................... 23
   4.5 Sign .............................................................. 24
      4.5.1 CPA Anonymous Version .................................. 24
      4.5.2 CCA2 Anonymous Version ................................ 25
   4.6 Verify ............................................................ 25
      4.6.1 CPA Anonymous Version .................................. 25
      4.6.2 CCA2 Anonymous Version ................................ 26
   4.7 Open .............................................................. 26
   4.8 Digital Signature ............................................... 26
      4.8.1 Setup ........................................................ 27
CONTENTS

4.8.2 Signing ..................................................... 28
4.8.3 Verify ..................................................... 28
4.9 Non-interactive Zero Knowledge Proof ............................................. 29
4.10 Public Key Encryption ..................................................... 30
  4.10.1 Setup ..................................................... 30
  4.10.2 Encryption ..................................................... 30
  4.10.3 Decrypt ..................................................... 31
4.11 Structure-Preserving Signatures on Equivalence Classes .................. 31
  4.11.1 Setup ..................................................... 31
  4.11.2 Sign ..................................................... 31
  4.11.3 Verify ..................................................... 32
  4.11.4 Changing the Representative Class ....................................... 32
4.12 DRIP Working Group Requirements .............................................. 33
  4.12.1 General ..................................................... 33
  4.12.2 Identifier ..................................................... 34
  4.12.3 Privacy ..................................................... 35
5 Security Analysis .......................................................... 37
  5.1 ProVerif ..................................................... 37
6 Implementation .............................................................. 39
  6.1 ESPcopter ..................................................... 39
  6.2 Cryptographic Libraries ..................................................... 40
  6.3 Selection of the Elliptic Curve ..................................................... 41
  6.4 Configuration Details ..................................................... 43
7 Evaluation and Results .......................................................... 45
  7.1 Time ..................................................... 45
  7.2 Energy Consumption ..................................................... 48
  7.3 Broadcast Bandwidth ..................................................... 51
8 Conclusion ................................................................. 53
Bibliography ................................................................. 55
Appendix ................................................................. 63
  A Mathematical Notations ..................................................... 65
  B BN254 parameters ..................................................... 67
Acronyms

A²RID Anonymous and directly-Authenticatable Remote IDentification. xii, xiii, 1, 3–5, 7, 10, 12, 13, 15, 17, 21–23, 26, 33, 34, 37–41, 43, 45–51, 53

AAA Attestation, Authentication, Authorization, Access Control, Accounting, Attribution, Audit, or any subset thereof. 33

ARID Anonymous Remote IDentification. xiii, 5, 17, 43

BBS Boneh-Boyen-Schacham. 26, 27

BG Bilinear Group. 11

BLS Boneh–Lynn–Shacham. 26–28

BLS-curve Barreto-Lynn-Scott curve. xiii, 13

BN-curve Barreto-Naehrig curve. xiii, 13

CAA Civil Aviation Authority. 10, 21, 22, 26, 33, 35

CAGR Compound Annual Growth Rate. 1

CDHP Computational Diffie-Hellman Problem. 27

CRS Common Reference String. 16, 23, 29

DDHP Decisional Diffie-Hellman Problem. 27

DLP Discrete Logarithm Problem. 13

DP Display Provider. 10, 22, 34

DRIP Drone Remote Identification Protocol. xii, 4, 5, 10, 22, 23, 26, 33–35, 40, 45, 53

DS Digital Signature. xiii, 21, 23, 24, 26–28, 31

DSRC Dedicated Short Range Communication. 17

EASA European Union Aviation Safety Agency. 1

ECDLP Elliptic Curve Discrete Logarithm Problem. 12, 13

ECIES Elliptic Curve Integrated Encryption Scheme. 30

ECP Elliptic Curve Point. 11, 12

ECP2 Elliptic Curve Point over extension field $G_2$. 11–13

EdDSA Edwards-Curve Digital Signature Algorithm. 27
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAA</td>
<td>American Federal Aviation Administration</td>
<td>iii, xii, 1, 2, 4, 5, 7–9, 17, 22, 53</td>
</tr>
<tr>
<td>FPV</td>
<td>First Person View</td>
<td>2</td>
</tr>
<tr>
<td>FRIA</td>
<td>FAA-Recognised Identification Areas</td>
<td>9</td>
</tr>
<tr>
<td>GCS</td>
<td>Ground Control Station</td>
<td>3, 10, 34, 43</td>
</tr>
<tr>
<td>GDH</td>
<td>Gap Diffie-Hellman</td>
<td>27</td>
</tr>
<tr>
<td>HKDF</td>
<td>HMAC-based Extract-and-Expand Key Derivation Function</td>
<td>27, 28</td>
</tr>
<tr>
<td>HMAC</td>
<td>Hashed Message Authentication Code</td>
<td>x, 27</td>
</tr>
<tr>
<td>I2OSP</td>
<td>Integer-to-Octet-String Primitive</td>
<td>27, 28</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
<td>39</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
<td>xiii, 4, 10, 26, 27, 29, 30, 33, 40, 53</td>
</tr>
<tr>
<td>IFP</td>
<td>Integer Factorization Problem</td>
<td>20</td>
</tr>
<tr>
<td>IKM</td>
<td>Input Key Material</td>
<td>27, 28, 30</td>
</tr>
<tr>
<td>ILT</td>
<td>Inspectie Leefomgeving en Transport</td>
<td>22</td>
</tr>
<tr>
<td>IND-CCA2</td>
<td>Indistinguishability under adaptive Chosen Ciphertext Attack</td>
<td>xii, xiii, 7, 14, 15, 21, 24–26, 30, 37, 43, 45, 47, 48, 51, 53</td>
</tr>
<tr>
<td>IND-CPA</td>
<td>Indistinguishability under Chosen Plaintext Attack</td>
<td>xii, xiii, 7, 14, 15, 21, 24–26, 30, 37, 43, 45, 47, 48, 51, 53</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
<td>xiii, 40, 41</td>
</tr>
<tr>
<td>KA</td>
<td>Key Agreement</td>
<td>30, 31</td>
</tr>
<tr>
<td>KCG</td>
<td>Key Generation Center</td>
<td>20</td>
</tr>
<tr>
<td>KDF</td>
<td>Key Derivation Function</td>
<td>27, 30</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
<td>43</td>
</tr>
<tr>
<td>LMS</td>
<td>Leighton-Micali Signature</td>
<td>26, 27</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
<td>30</td>
</tr>
<tr>
<td>MOV-attack</td>
<td>Menezes-Okamoto-Vanstone attack</td>
<td>13</td>
</tr>
<tr>
<td>NET-RID</td>
<td>Network Remote Identification</td>
<td>10, 22, 34</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
<td>xiii, 13, 14, 41</td>
</tr>
<tr>
<td>NIZK</td>
<td>Non-interactive Zero-Knowledge</td>
<td>16, 21, 23, 24, 29, 31</td>
</tr>
<tr>
<td>OKM</td>
<td>Output Keying Material</td>
<td>28</td>
</tr>
<tr>
<td>OS2IP</td>
<td>Octet-String-to-Integer Primitive</td>
<td>28</td>
</tr>
<tr>
<td>PKE</td>
<td>Public Key Encryption</td>
<td>4, 15, 21, 23, 24, 26, 30, 31, 65</td>
</tr>
</tbody>
</table>

A^2RID: An Anonymous RemoteID-compliant Group Signature Scheme for Commercial Drones
List of Figures

2.1 The possible different routing schemes. With the grey area indicating the broadcast range and \( \mathcal{X} \) = the broadcasting UAS, \( \mathcal{O} \) = possible observer but not receiver of the broadcast, \( \mathcal{P} \) = observer and possible receiver of broadcast. .................... 8

2.2 The two different RID schemes proposed by DRIP. .................................. 10

4.1 An UTM architecture based on the architecture presented by the FAA [1] and the DRIP requirements. ................................................................. 22

6.1 Pairing-friendly curves that can perform multiplication of a big number with a \( G_1 \) and a \( G_2 \) elliptic curve point on the ESPcopter. .............................. 42

7.1 Time measurements of each phase of the A\(^2\)RID scheme. The signing phase are executed on the ESPcopter. The join, verify, and open phases have been executed on a HP ZBook Studio G5 laptop equipped with two Intel(R) Core(TM) i7-9750H CPUs running at 2.60GHz. ................................................. 46

7.2 The memory required to store the pre-computed values for A\(^2\)RID on the ESPcopter for IND-CPA and IND-CCA2 anonymous schemes, depending on whether Miracl is used. ...................................................... 47

7.3 Circuit diagram of lab setup and actual lab setup to measure the energy consumption of the ESPcopter while using the A\(^2\)RID protocol. .......................... 49

7.4 Energy measurements of the singing phase of A\(^2\)RID in millijoule. .............. 50

7.5 An example on how the energy consumption is obtained using the RIGOL DS1202Z-E oscilloscope. ................................................................. 50

7.6 Bandwidth requirements using normal and compressed points in the A\(^2\)RID IND-CPA and IND-CCA2 anonymous broadcasts with the MTU of 1500 highlighted. .............................................................. 51

List of Tables

1.1 Cryptographic solutions to enhance the anonymity of RemoteID. \( \checkmark \) = Positive influence, \( \cdot \) = No influence, \( \times \) = Negative influence. .............................. 4

2.1 Content of RemoteID broadcast depending on the type. .............................. 9
<table>
<thead>
<tr>
<th>List of Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2 Popular BLS-curves and BN-curves with the corresponding bit security</td>
<td>13</td>
</tr>
<tr>
<td>levels and representation size in bits. The bit-security level is estimated</td>
<td></td>
</tr>
<tr>
<td>by [2].</td>
<td></td>
</tr>
<tr>
<td>2.3 NIST recommendation of minimal security levels depending on the year [3].</td>
<td>14</td>
</tr>
<tr>
<td>✓=recommended, x=not recommended.</td>
<td></td>
</tr>
<tr>
<td>2.4 Random oracle access for IND-CPA and IND-CCA2 anonymity.</td>
<td>14</td>
</tr>
<tr>
<td>2.5 Trust levels of IND-CPA and IND-CCA2 with respect to the issuer (setup</td>
<td>15</td>
</tr>
<tr>
<td>phase) and the opener (open phase).</td>
<td></td>
</tr>
<tr>
<td>3.1 Evaluating the group signature solutions available in the literature</td>
<td>19</td>
</tr>
<tr>
<td>against the proposed requirements. ✓= present, x= not present, - = unknown.</td>
<td></td>
</tr>
<tr>
<td>3.2 Comparison of [4] and [5] group signature schemes with respect to</td>
<td>20</td>
</tr>
<tr>
<td>characteristics important to the UAS over a 128-bit secure curve.</td>
<td></td>
</tr>
<tr>
<td>4.1 Security assumptions and minimal security level of IETF proposed DS</td>
<td>27</td>
</tr>
<tr>
<td>schemes.</td>
<td></td>
</tr>
<tr>
<td>6.1 Specifications of the ESPcopter.</td>
<td>39</td>
</tr>
<tr>
<td>6.2 Open-source IoT-friendly cryptographic libraries.</td>
<td>41</td>
</tr>
<tr>
<td>6.3 A²RID payload based on content size provided by ARID [6].</td>
<td>43</td>
</tr>
<tr>
<td>7.1 The maximum, mean and minimum time the A²RID signing phase acquired over</td>
<td>47</td>
</tr>
<tr>
<td>50 iterations on the ESPcopter.</td>
<td></td>
</tr>
<tr>
<td>7.2 Percentage of electric energy in the Li-Po battery dedicated to the</td>
<td>48</td>
</tr>
<tr>
<td>generation of one A²RID compliant broadcast message.</td>
<td></td>
</tr>
<tr>
<td>8.1 Research sub-questions and their answers.</td>
<td>53</td>
</tr>
<tr>
<td>A.1 Notation conventions and their description.</td>
<td>65</td>
</tr>
<tr>
<td>B.1 Parameters of pairing-friendly curve BN254.</td>
<td>67</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter is an introduction to the report. This report is the result of my gradation project as part of the master program "Information Security and Technology" at the Eindhoven University of technology. First the motivation for introducing Anonymous and directly-Authenticatable Remote IDentification (A²RID) an anonymous RemoteID-compliant group signature scheme for commercial drones is given. After the motivation is clear, the problem with the current situation is more clearly defined and the research questions are introduced. After the research question are stated a more detailed overview of what has been done to obtain answers to these research questions is given. Lastly we will provide a report outline stating the content of each chapter.

1.1 Motivation

The number of Unmanned Aircraft Systems (UAS) is ever increasing. The UAS market size is expected to grow from 4.4 billion USD in 2018 to 63.6 billion USD by the end of 2025, at a Compound Annual Growth Rate (CAGR) of 55.9% (the mean annual growth rate of investment) [7]. Due to the increase in the UAS market the aerial landscape is changing as never before, and for this specific reason, the American Federal Aviation Administration (FAA), responsible for regulating all aspects of civil aviation in the U.S., introduced RemoteID. RemoteID is a regulation that mandates the broadcast of UAS identification and location information that can be received by other parties. The requirements of RemoteID are presented in the Final Rule [8]. UAS manufacturers must comply with the Final Rule's requirements from 16 September 2022 and onwards [8]. The problem with this is that the Final Rule requirements do not consider any form of security [9]. While the RemoteID regulations are applicable in the U.S. the European Union Aviation Safety Agency (EASA) is working on a similar proposal [10]. The proposal of EASA is postponed to January 1st 2024 due the scarcity of Remote ID solutions currently available [11, 10]. This highlights the importance of the introduction A²RID scheme.

The publication of the Final Rule by the FAA generated a multitude of anonymity concerns [9]. The current RemoteID scheme uses a static Drone ID, this Drone ID will be sent to the public without any form of encryption or obfuscation. It is obvious that the RemoteID scheme does not adhere any form of anonymity. The lack of anonymity can lead to all sorts of problems:

- An increase in UAS-delivered parcel theft. Amazon introduced a service called Amazon Prime Air, which delivers packages up to five pounds within 30 minutes using UASs [12]. Even back in 2019, DHL launched its first fully automated urban UAS delivery service [13]. When an UAS broadcasts its locations, then it is easy to identify and locate such a UAS and steal its content.
• Only in 2020 in Minnesota alone law enforcement used UAS more than a thousand times [14]. The problem with introduction of RemoteID is that drone surveillance becomes unfeasible. This happens for the obvious reason that when a UAS is used for surveillance, law enforcement does not want the emit their identity since this will corrupt the investigation.

• Some even claim that the use of RemoteID is against the Fourth Amendment. Race Day Quad a online part store selling components for First Person View (FPV) UASs started a lawsuit against the FAA [15]. The Fourth Amendment protects American citizens from unreasonable searches and seizures by the government. RemoteID creates tracking information: the ID in combination with the UAS and controller location. This information can be used without the issuing of a warrant. This possibility of misuse is claimed by many to violate the Fourth Amendment.

The above-mentioned examples are just a fraction of the possible complaints one can have against RemoteID, and indicate a clear need for a change. However, saying that we should disregard or not use the RemoteID is an oversimplification of the problem. The FAA already has measurements in place to limit the disturbance, unease and obstructions an UAS can cause. There exist so-called No-Drone-Zones, namely:

1. The airspace above 400 feet;
2. Critical areas;
3. Public venues.

To not interfere with non-drone aircraft (e.g. aeroplanes) the UAS are not allowed at or above their operating height (400 feet). For the public venues and critical areas, the FAA indicates stadiums, Washington (DC), sporting events, airports, restricted, special use or security-sensitive airspace (e.g. military bases). Even though the FAA already clearly made an effort to make rules allowing for fewer disturbances, the existing guidelines are not enough. When somebody breaks the given rules e.g. flies above a sporting event, there is not much the FAA can do. The only action one can take is either identifying the UAS operator in the crowd or catching/destroying the UAS midair both obvious infeasible options. RemoteID would allow for the identification of the UAS by broadcast, which then on time allows the FAA to take action accordingly. This reasoning in combination with the previously mentioned complaints indicates the need for a more anonymous solution to RemoteID.

1.2 Problem Description

Based on the above motivation we can see that the use of RemoteID is beneficial, but that the security and anonymity requirements of the broadcast need to be adjusted. Instead of sending a broadcast with all the information in plaintext. We should create the possibility of identifying misuse of an UAS, but with additional anonymity enhancements in place. Before we can start to discuss the different options that may or may not increase the anonymity of the existing RemoteID scheme, we first need to have an understanding of what is meant with anonymity. It is important to note that anonymity is not the same as privacy. Where the introduction of anonymous RemoteID would lead to hiding who send the broadcast, introducing privacy would aim at hiding what the content of the broadcast is [16]. Given that hiding the content of the RemoteID broadcast is in contradiction with the goal of RemoteID that is, increasing areal awareness. Anonymity is the next best security notion that still addresses the public concerns.

Currently, the concern is the misuse and uncontrolled leakage of private and sensitive information in combination of direct identification of the UAS [5]. To mitigate the breach
of anonymity, the sensitive information must be handled accordingly and direct identification should not be possible. Based on the new anonymity requirements a combination of different cryptographic solutions are possible. Before we can determine the most feasible/suitable solution for A²RID, the requirements to create more anonymity and RemoteID must first be clarified:

1. **Anonymous.** Nobody, expected a dedicated authority can identify which UAS is sending the broadcast message.

2. **Traceable:** Only the dedicated authority can trace the UAS that signed the broadcast message.

3. **Unlinkable:** Given two broadcast messages, no one besides the dedicated authority can identify whether they are sent by the same UAS.

4. **Authentication:** Broadcast messages must be able to be verified/authenticated directly by everybody that receives the broadcast without the help of a third party.

5. **Independence:** An UAS must be able to create the broadcast independently of external resources (e.g. no dedicated infrastructures, combined computational power, or internet).

6. **Unframable:** No broadcast message can be made on a UAS behalf and be authenticated.

7. **Frequency:** The broadcast must be transmitted within 1 second of the measurement time. In other words, broadcast messages must be sent at least once per second.

8. **Precision:** Drone location and altitude must be accurate within 150 feet ($\pm$ 45.72m) of the true geometric altitude and 100 feet ($\pm$ 30.48m) of the true altitude and latitude, both with a 95 percent probability. Ground Control Station (GCS) location and elevation must be accurate within 15 feet ($\pm$ 4.572m) of the true geometric altitude and 100 feet ($\pm$ 30.48m) of the true altitude and latitude, both with a 95 percent probability.

9. **Representation:** The timestamp must be represented in Coordinated Universal Time (UTC). Locations and altitudes in feet.

10. **Independent Authentication:** Any observer can authenticate broadcast messages and identify forgeries without the help of a dedicated infrastructure or the help of the dedicated authority.

11. **Independent Generation:** The UAS must be able to send a broadcast messages without the help of any other device.

To fulfil the above RemoteID and anonymity requirements, multiple cryptographic solutions may fit. In Table 1.1 different cryptographic primitives are introduced and compared with respect to our requirements. The influences of the cryptographic primitives on the requirement are compared to the original RemoteID scheme.
CHAPTER 1. INTRODUCTION

Table 1.1: Cryptographic solutions to enhance the anonymity of RemoteID. ✓ = Positive influence, = No influence, x= Negative influence.

<table>
<thead>
<tr>
<th>Solution</th>
<th>Anonymity</th>
<th>Traceability</th>
<th>Unlinkability</th>
<th>Authenticity</th>
<th>Independency</th>
<th>Unframeability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>x/✓</td>
<td>✓/✓</td>
<td>✓/x</td>
<td>✓</td>
<td>✓</td>
<td>✓/x</td>
</tr>
<tr>
<td>HASH function</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Digital signature</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>Digital group signature</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

For both Public Key Encryption (PKE) and symmetric key encryption, either anonymity or untraceability and unlinkability are compromised. Symmetric key encryption can allow all UAS to use the same key pair or let every UAS have a unique key pair. Using the same keypair for all UAS allows for an anonymised opening of the broadcast but loses ones ability to link and trace a broadcast to a specific UAS by the dedicated authority. The same principle holds for PKE, an UAS can use its private key to encrypt the broadcast or its public key. In case the former is used an observer of the broadcast must match the public key to the private encryption key, consequently, anonymity is removed. Using a public key to encrypt would require all UAS to use the same key pair (given that a unique keypair is not possible in the case of broadcast) resulting in the same problem as symmetric encryption.

A hash function on itself does not contribute much with respect to increasing anonymity. It does not change the anonymity, traceability, unlinkability, or independence properties concerning the original RemoteID scheme. A hash function only affect the authenticity to a certain degree, it can show that the newly hashed message and the broadcasted message are the same. This however does not prove anything, given that the message and accompanied digest can be altered in transit.

Digital signatures contribute to proving the authenticity and integrity of the broadcast. Given that all UAS must use a unique key otherwise traceability and unlinkability are lost, digital signatures do not fit our requirements. As can be seen from Table 1.1 group signatures are the only cryptographic primitives that on their own fulfil the requirements. Therefore we will in this report, use group signatures to create A²RID. We will implement the group signature scheme and showing that it indeed fulfils the requirements. By doing so this thesis will try the following (sub)questions:

1. Is it possible to create a scheme that can directly but anonymously authenticate RemoteID-compliant broadcast messages?
   
   i) How can this scheme fit the Internet Engineering Task Force (IETF) Drone Remote Identification Protocol (DRIP), as well as the FAA RemoteID requirements?
   
   ii) How can such a scheme be optimised such that the broadcast can be sent within 1 second?
   
   iii) What effect does this scheme have on memory consumption?
   
   iv) What effect does this scheme have on energy consumption?
   
   v) What effect does this scheme have on the bandwidth of the broadcast?

For this one has to keep in mind that the scheme should work independently of the UAS. This means that if the drone is memory or computational restrained it must still be able to send the broadcast with the set requirements.
CHAPTER 1. INTRODUCTION

1.3 Project Description

With this thesis, we introduce a new Remote Identification (RID) method which also adheres to the requirements set by the FAA in addition to our set security requirements. The RID is an extension to the Anonymous Remote IDentification (ARID) protocol introduced by P. Tedeschi et al. [6] and will be referred to as Anonymous and directly-Authenticatable Remote IDentification (A²RID). Where ARID was based on using ephemeral pseudonyms only known by Unmanned Aircraft Systems Traffic Management (UTM), and needed brokered authentication to be verified, the A²RID will be completely independent. A²RID is based on the dynamic group signature protocol introduced by Derler et al. [4]. This group signature scheme is composed of 5 different models:

1. A public key encryption scheme;
2. A non-interactive zero-knowledge proof;
3. A digital signature;
4. A Structure preserving Signature on equivalence classes.

Given that not all these schemes are specified by Derler et al. [4], part of the thesis is selecting a model which fits the given requirements.

The implemented A²RID has been thoroughly tested, and implemented on a low-level device based on free or open-source software. In addition the source code is made publicly available [17].

1.4 Report Outline

This thesis is organised as follows:

• In chapter 2 (Background and Preliminaries), first RemoteID is explained in more detail. Then the work of Drone Remote Identification Protocol (DRIP) working group and their terminology is introduced which will be used throughout this report. In addition, preliminaries required to understand the A²RID protocol are explained.

• In chapter 3 (Related Work), a literature review of existing group signatures is presented. The scope of the related work are studies into existing RemoteID solutions and group signatures schemes.

• In chapter 4 (Dynamic Group signatures of A²RID), first a more detailed description of the chosen group signature scheme is given. Then after the scheme is clarified the dependencies on other algorithms/schemes are explained.

• In chapter 5 (Security Analysis), the introduced group signature scheme is analysed with the help of ProVerif. ProVerif is an automatic cryptographic protocol verifier.

• In chapter 6 (Implementation), the UAS on which A²RID is implemented is introduced as well libraries used. Then after the introduction of the hardware and software requirements, the implementation details are explained.

• In chapter 7 (Evaluation), time, memory, and energy measurements of the implementation in combination with the bandwidth results are introduced and evaluated.

• In chapter 8 (Conclusion), summarises the results of the thesis as well highlights the future research.
Chapter 2

Background and Preliminaries

Before one can even start to explain the dynamic group signature scheme introduced in the paper by Derler et al. [4, 18], one must have a clear understanding of the security notions required. Therefore, this section will give a basic introduction to curve security and bilinear groups. After this the notion of Indistinguishability under Chosen Plaintext Attack (IND-CPA) and Indistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2) anonymity are explained, given they are the basis of the group signature scheme. After the importance and basics of the security requirements are explained this chapter will recall and summarise the cryptographic primitives that will form the basis of our A²RID design. These cryptographic primitives are low-level cryptographic algorithms that are often used within the security area. The primitives we will be discussing are public key encryption, digital signatures, non-interactive zero-knowledge proof, and signatures of knowledge.

2.1 RemoteID

UASs can have different purposes, ranging from photography and videography, geographic mapping, agriculture, border control, building safety inspection, historical and wildlife conservation and search and rescue missions [19, 20, 21, 22, 23, 24]. Nevertheless, independent of the purpose or the type of UAS, the FAA introduced measures that contribute to ensuring public safety and the safety and efficiency of the airspace of the United States which all UASs must uphold. The FAA did so by introducing RemoteID regulation, to increase aerial awareness to themselves, national security agencies, law enforcement entities, and other government officials. In this section, we explain how the FAA designed the RemoteID regulation with the hopes to increase awareness and efficiency. The RemoteID regulation requires a broadcast message with precise specifications.

2.1.1 Broadcast

From a networking perspective, there are multiple methods of sending messages. There are 4 different communication modes, also known as routing schemes. The different routing schemes are displayed in Figure 2.1. From Figure 2.1 it can be seen that:

- **Broadcast** (Figure 2.1a): one-to-all, all observer devices in the proximity receive the message;

- **Unicast** (Figure 2.1b): one-to-one, a specific receiver device in the proximity receives the message;

- **Anycast** (Figure 2.1c): one-to-one-of-many mapping, the nearest observer device of a selected subset receives the message;
CHAPTER 2. BACKGROUND AND PRELIMINARIES

(a) Broadcast.

(b) Unicast.

(c) Anycast.

(d) Multicast.

Figure 2.1: The possible different routing schemes. With the grey area indicating the broadcast range and $\mathbf{X}$ = the broadcasting UAS, $\mathbf{0}$ = possible observer but not receiver of the broadcast, $\mathbf{i}$ = observer and possible receiver of broadcast.

- **Multicast** (Figure 2.1d): a one-to-many-of-many mapping, all observer’s devices of a subset in the proximity receive the message.

The RemoteID regulation requires broadcasts. This means that everybody within the proximity of an UAS can receive these messages. The reception range of the messages depends on the specific communication technology integrated on the UAS. This means that the message has no destination device (endpoint), it just sends its message and does not care which devices (or how many) receive the message. Other examples of broadcasts are radio stations or regular TV networks. Radio stations and regular TV networks create their data flow and they have no control over who receives the message as long as you have the right receiver (a TV or radio). The same principle holds for RemoteID: if you have a UAS compatible with RemoteID, the UAS will broadcast the message and any observer device which can receive a broadcast will be able to access the content of the message. By the set specification of the FAA, simple personal devices such as smartphones, tablets and laptops should be able to access the content of the RemoteID broadcast.
2.1.2 Types of RemoteID

As stated in section 2.1, there are different types of UASs. We can classify based on motor/wings or purpose, but also on the UAS’s ability to have RemoteID, which we use as the manner of classification in this section. This gives us three types:

1. UASs which have RemoteID built-in;
2. UASs which have a RemoteID module attached;
3. UASs which have no RemoteID.

By law, UAS manufacturers must comply with the RemoteID requirements from 16 September 2022 and onwards [8]. This means that the UAS manufacturers must either produce UASs that have a RemoteID built-in, or produce a module that follows the FAA’s requirements, or specifically state that the UAS has no RemoteID. However not having a RemoteID introduces additional restrictions which we will discuss later. RemoteID-compatible broadcast messages include the following information:

1. **UAS ID**: an unique UAS identifier;
2. **UAS longitude, altitude and elevation**: sent within 1 second of retrieving the information, and accurate within 100 feet of the true position, with 95 percent probability;
3. **UAS velocity**: has not prescribed specific requirements;
4. **Control station longitude, altitude and elevation**: no specific type of position source was used to determine this information,
5. **Time mark**: following the Coordinated Universal Time (UTC) time standard;
6. **Emergency status**: indicates unexpected behaviour. Caused by e.g. low battery levels and lost link.

Based on the type of UAS, the above information must be broadcasted every second (see Table 2.1). In addition to the removal or addition of certain message elements, UASs of type-2 are only allowed to fly within the line-of-sight. This means that the UAS operator can always see the UAS without the help of additional technology (e.g. binoculars) and without any obstruction (e.g. trees and buildings) [25]. Type 3 with no RemoteID will not broadcast any RemoteID-related information. As a consequence, type 3 UASs may only fly in dedicated areas. The FAA calls these locations a FAA-Recognised Identification Areas (FRIA) [26], which allows all types of UASs (type 1 through 3) but while types 1 and 2 are allowed outside this zone type 3 is not.

Table 2.1: Content of RemoteID broadcast depending on the type.

<table>
<thead>
<tr>
<th></th>
<th>Type-1 Buildin RemoteID</th>
<th>Type-2 Module RemoteID</th>
<th>Type-3 No RemoteID</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAS ID</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>UAS location</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>UAS velocity</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Control station location</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Time mark</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Emergency status</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
2.2 Drone Remote ID Protocol (DRIP) Working Group

The Internet Engineering Task Force (IETF) is a standards organisation aiming to create, implement, and deploy (new) standards related to the Internet protocol suite (TCP/IP) [27, 28]. A part of the efforts of one of their working groups has resulted in Drone Remote Identification Protocol (DRIP). DRIP aims to create guidelines on how different Civil Aviation Authority (CAA) throughout the whole world can implement Remote Identification (RID) in such a way that it can be employed in a reliable and trustworthy manner on an UAS [29]. Given that the IETF is a well-known and respected organisation, the conventions and names used in the DRIP architecture will be used in this thesis. By doing so, we intend to show that A²RID fulfils the DRIP requirements of being directly and easily integratable with the current efforts of the scientific community. For instance, a drone will be referred to as a UAS.

The IETF introduced two different DRIP architectures: Network Remote Identification (NET-RID) (see Figure 2.2b) and broadcast DRIP (see Figure 2.2a). The broadcast RID requires only a Radio Frequency (RF) data link, and no Internet or Wide Area Network (WAN). The RF data link limits the reach, the observer needs to be within RF LOS, limiting the range to roughly 1km. For the network RID, there are three additional elements:

1. NET-RID Service Provider (SP), collects RID messages and responds to NET-RID Display Provider (DP) queries;
2. NET-RID Display Provider (DP), receives data from NET-RID SP to respond to user queries;
3. Ground Control Station (GCS), the UAS subsystem used to remotely operate the aircraft.

Within this thesis, we aim to show that our approach is so generic it works in the environment limitations of both RID schemes.

![Diagram of the two different RID schemes proposed by DRIP.](image)

Figure 2.2: The two different RID schemes proposed by DRIP.
2.3 Bilinear Groups

**Definition 2.3.1: Bilinear Group**

Let \( BG = (p, G_1, G_2, e, P, \hat{P}) \) be a Bilinear Group (BG) with:

- \( p \), the prime order of groups \( G_1 \) and \( G_2 \);
- \( G_1 \) and \( G_2 \), cyclic groups of order \( p \);
- \( e \), a bilinear map \( e : G_1 \times G_2 \rightarrow G_T \);
- \( P \), generator of \( G_1 \);
- \( \hat{P} \), generator of \( G_2 \).

Within a BG, the cyclic groups \( G_1 \) and \( G_2 \) are subgroups of elliptic curves over the finite field. This means that operations within these groups will always be in the range of \( \{1, 2, \ldots, p - 1\} \). The difference between a normal group and a BG is that they are bilinear and non-degenerate:

**Definition 2.3.2: Bilinearity**

\[ e(g^a, h^b) = e(g, h)^{ab} \]

with \( g \in G_1 \), \( h \in G_2 \) and \( a, b \in \mathbb{Z}_p \).

**Definition 2.3.3: Non-degenerate**

\[ e(P, \hat{P}) \neq 1 \]

with \( P \) and \( \hat{P} \) the generators of group \( G_1 \) and \( G_2 \) respectively.

The non-degenerate property follows from the fact that the target group \( G_T \) has also prime order \( p \). This indicates that the pairing of \( e(P, \hat{P}) \) is simply the generator of \( G_T \). While all types of Bilinear Group (BG) satisfy Definition 2.3.2 and Definition 2.3.3, there are still three different types BGs [30]:

1. The symmetric setting where \( G_1 = G_2 \);
2. The asymmetric setting where \( G_1 \neq G_2 \) with an efficiently computable homomorphism \( \phi : G_2 \rightarrow G_1 \). This means that an Elliptic Curve Point over extension field \( G_2 \) (ECP2) in \( G_2 \) can be mapped/converted/cast to an Elliptic Curve Point (ECP) in \( G_1 \);
3. The asymmetric setting where \( G_1 \neq G_2 \) with no efficiently computable homomorphism in either direction.

So a Bilinear Group (BG) is of Type 1 if no homomorphic mapping like \( \phi : G_2 \rightarrow G_1 \) exits between the groups and \( G_1 = G_2 \). From this follows that \( G_1 = G_2 = G_T \). A BG is of Type 2, if there is no efficiently computable homomorphism from \( G_2 \) to \( G_2 \). Type 3, occurs when neither (\( G_1 \) to \( G_2 \) or \( G_2 \) to \( G_2 \)) exists.
2.4 Elliptic Curves for Security

Elliptic curves are the basis of the A²RID protocol. They will be used for encryption, signature generation, signature verification and many more. An elliptic curve over the finite field is defined as [31]:

**Definition 2.4.1: Elliptic Curve over Finite Prime Fields**

An elliptic curve $E$ over a finite prime field $\mathbb{F}_p$ with $p > 3$ is defined by the short Weierstrass equation:

$$y^2 \mod p = x^3 + Ax + B \mod p$$

while satisfying:

$$\Delta = 4A^3 + 27B^2 \mod p \neq 0 \mod p$$

with $A, B \in \mathbb{F}_p$, therefore the points on $E$ are defined by the short Weierstrass equation and a special point at $\infty$, to ensure defined point addition.

By definition 2.4.1, it becomes that depending on the values of $A$ and $B$ the curve changes. All elements of $E$ are integers between 0 and $p - 1$. Prime number $p$ must be chosen to secure the cryptographic operations in the finite field. The Standards for Efficient Cryptography Group (SECG) specify in Standards for Efficient Cryptography (SEC) the values of $p$ to be $[\log_2 p] \in \{192, 224, 384, 521\}$ to be efficient computationally and memory-wise (word alignment) [32].

If $E$ is defined over $\mathbb{F}_p$, then by definition $E$ is also defined over any extension $\mathbb{F}_{p^k}$ [31], which are the points generated by $G_2$ (as defined in section 2.3).

It is important to select an elliptic curve that is efficient and secure for all the operations we have to perform. Point addition (ECP and ECP2), negation and doubling do influence the choice of our curve. These point operations are not the limiting factor. Because all elliptic curves can perform point operations while pairing operations can be unfeasible on certain curves. This requires us to choose a pairing-friendly curve. A curve is pairing-friendly if it has a [33]:

1. Small embedding degree $k$;
2. Large prime order $E(\mathbb{F}_p)$.

Pairing-friendly curves can be classified into different groups. The taxonomy introduced by D. Freeman et al. [33] is most commonly used. This classification creates 6 different distinct groups. However, in the last couple of years, some advances were made concerning finding efficient pairing-friendly curves. This leads to different curves which can not be classified in the 6 categories introduced by Freeman et al. [34, 35, 36].

Not all of these different categories will be useful for IoT purposes, or in our case A²RID. Given that elements in $G_1$ are defined over $\mathbb{F}_p$ and extension field points $G_2$ over $\mathbb{F}_{p^k}$, where $k$ is the embedding the degree of the elliptic curve. This indicates that when $k$ is large elements in $G_2$ will be infeasible to transmit or perform point operations on. To highlight the importance of the embedding degree it is important to note that the embedding degree is also known as the security multiplier. This indicates that there is a trade that we need to consider between security and efficiency. The security of an elliptic curve relates to the Elliptic Curve Discrete Logarithm Problem (ECDLP) and therefore to $k$ which is defined as [37, 38]:

**Definition 2.4.2: Elliptic Curve Discrete Logarithm Problem (ECDLP)**

Given elliptic curve $E$ and points $P, Q \in E(\mathbb{F}_p)$ find an integer $a \in \mathbb{F}_p$ such that $Q = aP$ or...
return ⊥ if no such integer exists.

Hence based on Definition 2.4.2 we can see that the security of elliptic curve cryptography depends on the ability to compute a point multiplication (aP) and the inability to compute the multiplicand (a) given the original and product point (Q). The ECDLP, is defined over $E(\mathbb{F}_p)$ which can be converted in an instance of the Discrete Logarithm Problem (DLP) [39]. This is better known as the Menezes-Okamoto-Vanstone attack (MOV-attack). The MOV-attack shows that $P$ and $Q$ on $E(\mathbb{F}_p)$ can be mapped to an element in the finite field $\mathbb{F}_{p^k}$ [40], resulting in the DLP. The DLP is defined as [41]:

**Definition 2.4.3: Discrete Logarithm Problem (DLP)**

Given $a, b \in \mathbb{F}_p$, find an integer $x \in \mathbb{F}_p$ such that $b = a^x$.

From this, it can be easily seen that this attack is not useful in case $k$ is big.

Luckily for the use of IoT devices, we can use some useful mathematical principles. One which is especially useful in our case is the twist of an elliptic curve $d$. When an elliptic curve has a twist curve this means that [42, 43]:

1. Degree $d$ of twist curve $E'$ is a factor of $k$
2. Twist curve $E'$ is curve isomorphic to $E$ over $\mathbb{F}_p$.

This allows us to do operations in $\mathbb{F}_{p^d}$ instead of the bigger extension field $\mathbb{F}_{p^k}$ while having the same security properties [44]. For this reason, there are two types of pairing-friendly curves useful A²RID:

1. Barreto-Lynn-Scott curve (BLS-curve)
2. Barreto-Naehrig curve (BN-curve)

Some of the more popular BLS-curves and BN-curves are shown in Table 2.2.

Table 2.2: Popular BLS-curves and BN-curves with the corresponding bit security levels and representation size in bits. The bit-security level is estimated by [2].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BN254</td>
<td>100</td>
<td>508</td>
<td>1024</td>
<td>3072</td>
</tr>
<tr>
<td>BN462</td>
<td>128</td>
<td>924</td>
<td>1848</td>
<td>5544</td>
</tr>
<tr>
<td>BLS12-381</td>
<td>128</td>
<td>762</td>
<td>1524</td>
<td>4572</td>
</tr>
<tr>
<td>BLS48-581</td>
<td>256</td>
<td>1162</td>
<td>2324</td>
<td>6972</td>
</tr>
</tbody>
</table>

Note that the $n$-bit security level in Table 2.2 is with respect to the most recent attacks and size requirements [2]. This security level is estimated based on the computational cost to solve the ECDLP and DLP, with the most efficient algorithms [42]. When the BN254 curve has introduced the assumption was made that the curve had a 128-bit security level, based on the algorithms and computational power available at the time [42]. This however does not diminish the advantage of using a BLS-curve or BN-curve given that the ECP2 points in $G_2$ would have been of sizes 6144, 1108, 9144 and 55776 bits respectively while using a non-twisted curve. National Institute of Standards and Technology (NIST), part of the U.S. Department of Commerce has introduced recommendations regarding the minimum security levels. In Table 2.3, these levels are shown based which year. However, NIST is not the only organisation making recommendations. Based on recent surveys other authors and institutions claim to e.g. abolish 128-bit level security as soon as 2026 [45], making the choice of a curve even more difficult.


Table 2.3: NIST recommendation of minimal security levels depending on the year [3]. ✓=recommended, x=not recommended.

<table>
<thead>
<tr>
<th>Security level [b]</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;2030</td>
</tr>
<tr>
<td>&lt;112</td>
<td>x</td>
</tr>
<tr>
<td>112</td>
<td>✓</td>
</tr>
<tr>
<td>128</td>
<td>✓</td>
</tr>
<tr>
<td>192</td>
<td>✓</td>
</tr>
<tr>
<td>256</td>
<td>✓</td>
</tr>
</tbody>
</table>

2.5 CPA and CCA2 Anonymity

Indistinguishability under Chosen Plaintext Attack (IND-CPA) with respect to group signatures is a different notion then IND-CPA regarding an encryption scheme. The notion of IND-CPA-anonymity we will use is defined concerning the model provided by D. Boneh et al. [46]. The model used for Indistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2)-anonymity by M. Bellare et al. [47]. The difference between adversary assumptions of the models is highlighted in Table 2.4. A random cryptographic oracle is a black box method, that allows one to send a cryptic query and get an immediate result. Given that an oracle is a theoretical concept used in proving cryptographic concepts, the possibility of answering the query immediately does not have to exist (e.g. computational impossible). The random oracles are defined as follows:

- \( \text{Signature}(m,i) \), given message \( m \), the signature is returned by a key at index \( i \);
- \( \text{Private key}(i) \), the private key at index \( i \) is returned;
- \( \text{hash}(x) \), deterministic hash returned based on input \( x \);
- \( \text{Open}(\sigma) \), based on input signature \( \sigma \) the signer is revealed.

Table 2.4: Random oracle access for IND-CPA and IND-CCA2 anonymity.

<table>
<thead>
<tr>
<th>Oracle</th>
<th>IND-CPA</th>
<th>IND-CCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature(m,i)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Private key(i)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hash(x)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Open(\sigma)</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

For the IND-CPA model this shows that as the private key of the group does not leak IND-CPA anonymity holds. In case the group private key leaks then all signatures can be traced back to the broadcasting UAS. For the IND-CCA2 model to be anonymous, it must be the case that even when arbitrary signatures are open anonymity is still reserved. This indicates that the trust level one can have in the involved parties differs for IND-CPA and IND-CCA2 anonymity. In Table 2.5 shows the difference in trust. This table shows the lowest level of trust one can have in the open and issuer of the credentials while the anonymity still holds.
### Chapter 2. Background and Preliminaries

Table 2.5: Trust levels of IND-CPA and IND-CCA2 with respect to the issuer (setup phase) and the opener (open phase).

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Opener</th>
<th>Issuer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IND-CPA</td>
<td>IND-CCA2</td>
</tr>
<tr>
<td></td>
<td>IND-CPA</td>
<td>IND-CCA2</td>
</tr>
<tr>
<td>Anonymity</td>
<td>uncorrupt</td>
<td>fully corrupt</td>
</tr>
<tr>
<td>Traceability</td>
<td>uncorrupt</td>
<td>partially corrupt</td>
</tr>
<tr>
<td>Non-frameability</td>
<td>fully corrupt</td>
<td>fully corrupt</td>
</tr>
</tbody>
</table>

#### 2.6 Public Key Encryption

Public Key Encryption (PKE) also referred to as asymmetric cryptography uses a one-way function in combination with public and private key pairs to achieve reversible randomisation of the input. For A²RID we require encryption to be done using the public key and the decryption using the private key. While the main principles of all PKE schemes are the same there are still quite some differences related to the keys (e.g. size, generation and distribution methods), the computationally difficult problems PKE schemes rely on (e.g. discrete logarithm problem, integer factorisation, decisional composite residue assumption), but independent of the key size and the mathematical assumptions all PKE can be labelled and proven to be a specific security level. For the purpose of A²RID we require the PKE to be Indistinguishability under Chosen Plaintext Attack (IND-CPA) and Indistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA2) secure. IND-CPA secure concerning encryption means that even when an attacker can send his input (plaintext) to the PKE scheme and receive the corresponding output (ciphertext) the attacker still can not gain any information from the PKE system which influences the security. As the name of IND-CCA2 indicates instead of an attacker being able to choose the plaintext as is done with IND-CPA, the attacker is now able to obtain the decryption of any ciphertext of his choosing, except the challenge. The challenge is the actual ciphertext he wants to decrypt. Again when this process does not provide any information that can influence the security of the PKE scheme, the system is IND-CCA2 secure. From the above it follows that when a PKE system is IND-CCA2 secure it is automatically IND-CPA secure [48]. Based on all the above we can conclude that we need an PKE scheme that consists of the following 3 phases:

1. pke . setup (1^k) : the key generation phase, generates a linked public and private key pair (pk_0, sk_0), and the distribute the public key pk_0 to the public.
2. pke . enc (pk_0, m) : the public party encrypts their data m with the public key pk_0 and send the encrypted ciphertext c to the owner of the private key sk_0.
3. pke . dec (sk_0, c) : the owner of the private key sk_0 decrypts the ciphertext c successfully to obtain the plaintext m iff the plaintext has been encrypted with the corresponding public key pk_0.

#### 2.7 Digital Signatures

Digital signatures enable one to verify the authenticity and integrity of data. Data authenticity confirms the source of the data, a signature is source specific and can be traced. Data integrity entails that data is not altered since it has been signed by the authenticated party. As for the PKE scheme, a digital signature scheme can be proven to have a certain security level. In our case, we require a digital signature scheme which can ensure Existential Unforgeability under Chosen Message Attack (UEF-CMA) security. For a signature to be UEF-CMA, the attacker must not be able to forge any signature [49]. A digital signature scheme can be divided into three different phases:
CHAPTER 2. BACKGROUND AND PRELIMINARIES

1. \texttt{ds.setu}(k^k)\texttt{p}: the generation of the key pair \((usk, upk)\), where the secret key \(usk\) is used for the signing and the public key \(upk\) for verification.

2. \texttt{ds.sign}(sk, m): secret key \(usk\) is used to sign the data \(m\) to generate a signature \(\sigma\).

3. \texttt{ds.vrf}(pk, m, \sigma): given the signature \(\sigma\) and the corresponding data \(m\), the public key \(upk\) is used to verify the authenticity and integrity of the received message.

2.8 Non-interactive Zero Knowledge Proofs

A Non-interactive Zero-Knowledge (NIZK) proof is a cryptographic scheme between two parties a verifier and a prover. A prover wants to show knowledge of a certain value \(x\) with the use of a witness \(w\) and a pre-shared Common Reference String (CRS). The value of the witness \((w)\) and the trusted setup in which the witness is created are published such the verifying party has access. The building blocks of a NIZK proof are as follows:

1. \texttt{nizk.setu}(k^k): the generation of Common Reference String (CRS), which is made public or directly send to the verifier.

2. \texttt{nizk.proof}(CRS, x, w): given the CRS and secret \(x\), and a witness called \(w\) the proof \(\pi\) is generated.

3. \texttt{nizk.vrf}(CRS, x, \pi): given the CRS in combination of the witness \(w\) proof \(\pi\) can be verified.

It is important to note the distinction between interactive and non-interactive zero. Where an interactive proof requires active interaction with the verifier in the \texttt{nizk.proof} stage, zero-interaction proofs are standalone. All interactive proofs can be transformed to non-interactive by using the Fiat-Shamir transformation [50, 51]. The Fiat-Shamir transformation eliminates the requirement of interactive zero-knowledge proofs to publish the random responses of the verifier. Instead of using the challenge produced by the verifier a hash of the previous protocol messages is generated and used. The hash now provides the unpredictability of the proof otherwise produces by the random challenge of the verifier.
Chapter 3

Related work

The related work section first introduces, compares and explains the existing published work regarding improving the security of RID protocols while also satisfy the FAA requirements. After these different works are discussed, different group signature schemes will be discussed. The different group signature schemes are discussed to show why the scheme by Derler et al. [4] will be the basis of A²RID.

3.1 Existing RID Protocols

In this section, existing solutions to the RemoteID problem are discussed. While there are plenty of law suites and complaints against RemoteID there are not that many research papers published with technical solutions to the anonymity issues RemoteID presents. There is only one contribution directly related to the anonymity of RemoteID, called ARID, proposed by P. Tedeschi et al. [6]. ARID introduces RemoteID anonymity by the use of ephemeral pseudonyms that only designated people/institutions can link to a long-term identifier. While the use of ARID has shown to be secure, energy efficient and fast enough, it does not allow direct authentication. A generic receiver can not directly verify the authenticity of the received broadcast without the help of a Trusted Third Party (TTP). While there are no other direct contributions to introduce anonymity to a RemoteID compliant broadcast, there is ample of research into anonymization of general broadcast messages. Most of the technical advances regarding broadcast anonymization over the last couple of years are related to Vehicle-To-Vehicle (V2V) communication [52, 53, 54, 55, 56, 57, 58, 59]. However, to solutions proposed for V2V require either an internet connection [58] or even dedicated networks such as Dedicated Short Range Communication (DSRC) [52, 56] or Vehicular ad-hoc Networks (VANET) [53, 54, 55, 57, 59]. Given that we aim to introduce a protocol that allows for anonymous RemoteID-compliant authentication for commercial drones, we can not depend a network infrastructure as V2V does.

3.2 Group Signatures Schemes

A anonymous group signature scheme allows a member of a group to sign messages anonymously on behalf of his group. Membership to a group is determined in the setup phase by the dedicated authority. This signing method allows members to stay anonymous and only provide the information that they are a member. This signed message can only be opened by the dedicated authority to identify the signer. Basic properties nearly all group signatures possess are:
1. **Coalition resistant**: even when members work together, all their information and computing power combined should not produce a verifiable signature;

2. **Framing resistant**: a group member cannot be held accountable for a message that they did not sign;

3. **Unforgeable**: only signers that are a member of the group managed by the dedicated authority can sign verifiable messages;

4. **Sound and complete**: every message signed by a member following the correct signing protocol presented by the group signature scheme must be accepted (read authenticated) by the corresponding verifying algorithm of the group signature scheme;

5. **Anonymous**: a random bystander or another member that did not sign the message is unable to pinpoint the source member;

6. **Unlinkable**: given two different signed messages nobody without opening the signature can tell whether the messages originate from the same member;

7. **Traceable**: the dedicated authority can trace back a signature produced by a member. The processes of tracing can only be successfully performed by the dedicated authority, and not by any other observer of the message (e.g. a random bystander).

We aim, as explained in chapter 1, to achieve the basic group signature properties in combination with the independence of generation, verification and authentication while not using pairing operations while signing. Most group signatures by definition are comprised out of the 5 phases proposed by [60]:

1. **Setup**: generating and storing the globally used parameters;

2. **Join**: a protocol that allows the group to be expanded with new members. This is a protocol between the dedicated authority and the new member which results in the generation of the signer certificates;

3. **Sign**: an algorithm that is based on parameters and keys generated in the setup and join phase. This allows a signer to create a signature, making the message authenticatable but still anonymous;

4. **Verify**: an algorithm which can be performed by anyone in possession of the public parameters. It allows the broadcasts by a signer to be authenticated. If authenticated the message is signed by a group member, otherwise not;

5. **Open**: in case of a dispute the dedicated authority can open the authenticated broadcast message and trace it back to its signer.

In Table 3.1, the different properties of group signatures are compared. To reduce implementation problems, we will only consider group signatures that have been proven to work on IoT devices.

As can be seen in Table 3.1 the group signatures presented in [70, 69, 68] all fail to be independent. There are two types of group signatures:

1. **Sign-Encrypt-Prove (SEP)**, provides a **Signature of Knowledge (SOK)** to show knowledge of the cipher-text in addition to an encrypted membership certificate to illustrate membership [65];

2. **Sign-Randomize-Proof (SRP)**, provides a **SOK** to show knowledge of the cipher-text in addition to a pre-generated signature is randomised to show group membership [71].
Table 3.1: Evaluating the group signature solutions available in the literature against the proposed requirements.

✓ = present, x = not present, - = unknown.

<table>
<thead>
<tr>
<th>Anonymity</th>
<th>Traceability</th>
<th>Unlinkability</th>
<th>Authenticity</th>
<th>non-frameability/Exculpability</th>
<th>Unforgeability</th>
<th>Independent verification/generation</th>
<th>No pairing</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>[5]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>[61]</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>✓</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[62]</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>✓</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[63]</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>x</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[64]</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>✓</td>
<td>x</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[65]</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>✓</td>
<td>x</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[66]</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[67]</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[68]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[69]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>[70]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
The group signature scheme of S. Ricci et al. [69] is of the former type. As per RemoteID requirements, anybody with a personal device should be able to verify the broadcast sent by an UAS. The group signature scheme by S. Ricci et al. however requires a receiver group. This receiver group requires a manager to enrol new receivers and create new receiver keys. This that all personal devices that wish to verify a broadcast need a unique key pair. This is not in line with the RemoteID requirement of public verification. The scheme introduced by X. Chen. et al [68] is also a SEP scheme but does not require additional key generation. This scheme however requires interaction between the group member (UAS) and the receiver, clearly not satisfying the set requirements.

The group signature introduced by V. Gayoso Marínez et al. [70] is an efficient group signature scheme based on the SDLP and IFP designed to use on restricted devices. This scheme would have been a great fit, would it not depend on other UAS group members. To generate a signature an UAS member is chosen at random by a Key Generation Center (KCG). This means the UAS requires persistent communication with the KCG, a external party. This communication requirement removes this group signature proposal given the signature generation is not independent.

The group signature schemes by X. Boyen et al. [67] and R. Xie et al. [66] both do not discuss Unlinkability. While J. Cammenisch [65] and H. Ge et al [64] don’t mention traceability as a part of their scheme. Given that this thesis aims to implement an autonomous directly authenticatable broadcast scheme, proving/verifying whether these properties are present is out of scope.

From Table 3.1 we can see that the only contenders are [4, 5, 61, 62]. The reason Implemented is a criterion is since even though a solution is provided, and mathematically proven this does not implicitly indicate that the implementation works.

Since the signature generation is the only part performed on the UAS, this is the part we need to pay attention to. A comparison of both proposals and their respective sizes and number of operations are shown in Table 3.2 both concerning 100-bit RSA security level.

Table 3.2: Comparison of [4] and [5] group signature schemes with respect to characteristics important to the UAS over a 128-bit secure curve.

<table>
<thead>
<tr>
<th>signature size [b]</th>
<th>Operating time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4] 3309</td>
<td>0.771</td>
</tr>
<tr>
<td>[5] 648000</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Based on Table 3.2, it can be seen that the size of [4] is significantly smaller, in addition, the generation of the signature is faster as well. Note that the signature generation is done on an ARM-Cortex-M0+ with drop-in hardware accelerator at 48MHz for [4] and a 64-bit Intel(R) Core(TM) i7-8700 CPU at 3.20 GHz 3.19 GHz, with the former being a more constrained platform.
Chapter 4

Dynamic Group Signatures of A²RID

As illustrated in the related work (see chapter 3), the dynamic signature scheme introduced by Derler et al. [4, 18] fits the requirements of an Anonymous and directly-Authenticatable Remote IDentification (A²RID) scheme the best. The paper by Derler et al. introduces three different schemes:

1. A general dynamic group signature scheme;
2. A dynamic group signature scheme proven to be IND-CPA anonymous;
3. A dynamic group signature scheme proven to be IND-CCA2 anonymous.

For the purpose of A²RID, the IND-CPA and IND-CCA2 anonymous schemes both satisfy our requirements. For this reason, we disregard the general dynamic group signature scheme. The remaining IND-CPA and IND-CCA2 anonymous schemes require:

1. A Digital Signature (DS) scheme;
2. A Public Key Encryption (PKE) scheme;
3. A Non-interactive Zero-Knowledge (NIZK) proof;
4. The Structure-Preserving Signatures on Equivalence Classes (SPS-EQ) introduced by G. Fuchsbauer et al. [72].

Given that the DS scheme, PKE scheme and NIZK proof are not given, this chapter will be dedicated also to choosing suitable schemes. These schemes will be introduced after we explain the group signature scheme by Derler et al., which consists of 4 phases Setup, Join, Sign, Open and Verify. The reason for introducing the DS scheme, PKE scheme and NIZK after the group signatures is because they are open for choice and hence not necessary to understand the group signature scheme. But before first the entities involved are introduced.

4.1 Entities in A²RID

Figure 4.1 shows the architecture of the Unmanned Aircraft Systems Traffic Management (UTM), where different entities are involved in the RID of A²RID. The Unmanned Aircraft Systems Traffic Management (UTM), is an environment created by different government entities trying to ensure compliance with physical and software infrastructures and flight information management [1]. The Civil Aviation Authority (CAA) is responsible for many services, including:
1. Penalising non-compliance with guidelines;
2. Creating and specifying guidelines for messages to be transmitted;
3. Observing and monitoring USS;
4. Investigating or delegating the investigations of air traffic incidents.

Note that we only discuss the UAS related functions of the CAA. Examples of CAAs are the FAA in the U.S. or the Inspectie Leefomgeving en Transport (ILT) (translated: Human Environment and Transport Inspectorate) in The Netherlands [73, 74]. The CAA regulates the control of the air traffic. This means that it makes sure other entities will carry it out. This can be done by UAS Service Suppliers (USS), and DRIP requires NET-RID Service Provider (SP) and NET-RID Display Provider (DP) as part of their model, they are shown as an USSs in Figure 4.1.

For the sake of explaining A²RID, and to show the division of responsibilities, two additional USS are introduced:

- USS$_g$, responsible for the generation of the group data;
- USS$_u$, responsible for the generation of the user (UAS) join data.

These USS are just examples and are not required for actual deployment. The goal of using USS$_g$ and USS$_u$ is to show how the operational heavy operations are not required to be performed on the UAS and how identities are separated to provide anonymity. The DRIP architecture requires a distinction between public observers. There is the general public indicated by General Public Observer and there is the Public Safety Observer. The Public Safety observer forwards to the USSs the received broadcasts that disobey the No-Drone-Zones, as explained in chapter 1.

### 4.2 General Description A²RID

The A²RID scheme consists just as most group signatures out of a:
CHAPTER 4. DYNAMIC GROUP SIGNATURES OF A²RID

• **Setup phase:** the A²RID groups are created and the public parameters are published. The DRIP working group and RemoteID do not introduce groups. Therefore the group division is not set in stone. Either one can create one big group per e.g. country or smaller regions. If one opts to create multiple groups then a group identifier needs to be added to the group.

• **Join phase:** an UAS joins a group such that it can send anonymous broadcast messages on behalf of the group it joined.

• **Sign phase:** an UAS broadcasts messages on behalf of the group it is member of.

• **Verify phase:** any observer can authenticate whether a broadcast is legit.

• **Open phase:** if an UAS is in violation with the No-Drone-Zones an observer can report this. The UTM can open the broadcast. By opening the broadcast the identity of UAS is recovered. The UTM can take the appropriate measures against the UAS.

### 4.3 Setup

In this phase, all the group-specific data are generated and stored. This phase is performed offline by the USS_g and not on the UAS, to make sure there are no computational restrictions.

1. UTM verifies and approves the USS_g, USS_u;
2. USS_g generates bilinear group $BG = (p, G_1, G_2, G_T, e, P, \hat{P})$, as defined in section 2.3;
3. USS_g generates as part of SPS-EQ key pair $(sk_R, pk_R) = ((sk_{R0}, pk_{R0}), (sk_{R1}, pk_{R1}))$ by invoking $\text{spseq.setup}(BG, 2)$;
4. USS_g generates as part of PKE a key pair $(sk_O, pk_O)$ by invoking $\text{pke.setup}(1^k)$;
5. USS_g generates as part of NIZK a CRS by invoking $\text{nizk.setup}(1^k)$;
6. USS_g sends the group public key $gpk = (pk_R, pk_O, CRS)$ to SDSP;
7. SDSP stores this new $gpk$ in its Public Registry;
8. USS_g stores the private components of the generated keys.

### 4.4 Join

A UAS can request to join a group, whose parameters have been generated in the setup phase (see section 4.3). If a UAS requests to join a group, then USS_u performs the join operations to ensure no computational restrictions. This means that the joining of a group is done offline, so not while operating the UAS. The parameters computed by USS_u are communicated to the UAS. Note that in case a verification step fails the protocol is stopped.

1. USS_u generates as of part of DS a UAS specific key pair $(usk, upk)$ by invoking $\text{ds.setup}(1^k)$;
2. USS_u requests $gpk = (pk_R, pk_O, CRS)$ from SDSP;
3. USS_u sends $upk$ to the SDSP;
4. SDSP stores upk in the public registry;
5. USS_u generates nonces $q, r \overset{\$}{\leftarrow} \mathbb{Z}_p$;
6. USS_u calculates \((U, Q) = (r \cdot qP, qP)\);
7. USS_u generates as part of PKE cipher text \(C\) by invoking \(pke.enc(pk_o, r \hat{P}, \omega)\);
8. USS_u generates as part of DS a digital signature \(\sigma_{ds}\) by invoking \(ds.gen(usk, C)\);
9. USS_u generates as part of NIZK a proof of knowledge \(\pi\) by invoking \(nizk.proof(crs, ((U, Q), C, pk_o), (r, \omega))\);
10. USS_u sends \(M = ((U, Q), C, \sigma_{ds}, \pi)\) to USS_g;
11. USS_g receives \(M\);
12. USS_g parses \(M\) such that \(M = ((U, Q), C, \sigma_{ds}, \pi)\);
13. USS_g verifies as part of NIZK the proof of knowledge \(\pi\) by invoking \(nizk.vrf(crs, ((U, Q), C, pk), \pi)\);
14. USS_g verifies as part of DS digital signature \(\sigma_{ds}\) by invoking \(ds.vrf(upk, C, \sigma_{ds})\);
15. USS_g generates as part of SPS-EQ signature \(\sigma\) by invoking \(\sigma = spseq.sign((U, Q), sk_R)\);
16. USS_g sends \(\sigma\) to USS_u;
17. USS_u receives \(\sigma\);
18. USS_u verifies as part of SPS-EQ signature \(\sigma\) by invoking \(spseq.vrf((U, Q), \sigma, pk_R)\);
19. USS_u generates UAS key \(((r, P), \sigma)\) by invoking \(spseq.ChgReg((U, Q), \sigma, q^{-1}, pk_R)\);
20. USS_u sends private parameters to the UAS.

4.5 Sign

An UAS signs its broadcast message on behalf of its group. The signing phase is performed online, so while the UA is flying. The broadcast consists of the outcome of group signature scheme [4] and message \(m\). The content of \(m\) depends on the type of RemoteID is used (see subsection 2.1.2). By the requirements, a broadcast needs to be emitted every second, with the parameters of \(m\) being obtained within one second of sending. There are two ways of signing this message IND-CPA and IND-CCA2 anonymous.

4.5.1 CPA Anonymous Version

Given group public key \(gpk\), their private key and its message \(m\) an UAS can create a IND-CPA-anonymous signature as follows:

1. The UAS generates a message \(m\) the content depending on whether the UAS has type-1 or type-2 RemoteID (see Table 2.1).
2. The UAS generates nonce \(\rho, v \leftarrow Z_p\);
3. The UAS generates as part of SPS-EQ randomisation \(\sigma_1\) by invoking \(spseq.ChgReg(gsk, \rho, pk_R)\);
4. The UAS calculates \(N = vP\);
5. The UAS generates hash \(c = H(N || \sigma_1 || m)\);
6. The UAS calculates \(z = v + c \cdot \rho \mod p\);
7. The UAS sets \(\sigma_2 = (c, z)\);
8. The UAS broadcasts \((\sigma_1, \sigma_2, m)\).
4.5.2 CCA2 Anonymous Version

Given group public key \( g pk \), their private key and its message \( m \) a UAS can create a IND-CCA2-anonymous signature as follows:

1. The UAS generates a message \( m \) the content depending on whether the UAS has type-1 or type-2 RemoteID (see Table 2.1).

2. The UAS generates nonce \( \rho, \mu, u, v, \eta \sim \mathbb{Z}_p \);

3. The UAS generates as part of SPS-EQ randomisation \( \sigma_1 \) by invoking \( \text{spsseq.ChgReg}(gsk, \rho, pk_R) \);

4. The UAS calculates \( N = vP \);

5. The UAS calculates \( \hat{M}_1 = \eta \hat{Y} \);

6. The UAS calculates \( \hat{M}_2 = (v + \eta)\hat{P} \);

7. The UAS calculates \( \hat{C}_1 = \mu \hat{Y} \);

8. The UAS calculates \( \hat{C}_2 = (\rho + u)\hat{Y} \);

9. The UAS generates hash \( c = H(N||\hat{M}_1||\hat{M}_2||\sigma_1||m) \);

10. The UAS calculates \( z_1 = v + c \cdot \rho \mod p \);

11. The UAS calculates \( z_2 = \eta + c \cdot m \mod p \);

12. The UAS sets \( \sigma_2 = (\hat{C}_1, \hat{C}_2, c, z_1, z_2) \);

13. The UAS broadcasts \( (\sigma_1, \sigma_2, m) \).

4.6 Verify

The verification can be performed by both types of observers as shown in Figure 4.1. The verification of the broadcast message does depend on the type of signature that is used. The RemoteID scheme assumes that the device of the observer is computationally equipped to receive and verify a broadcast. Note that the broadcast are of different sizes depending on whether the IND-CPA or IND-CCA2 signing scheme is been used. The observers’ device, therefore, knows which verification methods to use based on the size.

4.6.1 CPA Anonymous Version

In case an IND-CPA-anonymous signature is broadcasted the following steps will lead to verification:

1. The observer parses \( (\sigma_1, \sigma_2) \) such that \( (\sigma_1, \sigma_2) = ((R', P'), (c, z)) \);

2. The observer verifies as part of SPS-EQ signature \( \sigma_1 \) by invoking \( \text{spspeq.vrf}(\sigma_1, pk_R) \);

3. The observer calculates \( N = zP - cP' \);

4. The observer calculates \( c' = H(N||\sigma_1||m) \);

5. The observer verifies \( c = c' \).
CHAPTER 4. DYNAMIC GROUP SIGNATURES OF A²RID

4.6.2 CCA2 Anonymous Version

In case an IND-CCA2-an anonymous signature is broadcasted the following steps will lead to verification:

1. The observer parses \((σ_1, σ_2)\) such that \((σ_1, σ_2) = ((R', P'), (c, z_1, z_2))\);
2. The observer verifies as part of SPS-EQ signature \(σ_1\) by invoking \(\text{spspeq.vrf}(σ_1, pk_R)\);
3. The observer calculates \(N = z_1 P - c P'\)
4. The observer calculates \(\hat{M}_1 = z_2 \hat{Y} - c \hat{C}_1\);
5. The observer calculates \(\hat{M}_2 = (z_1 + z_2) \hat{P} - c \hat{C}_2\);
6. The observer calculates \(c' = H(N || \hat{M}_1 || \hat{M}_2 || σ_1 || m)\);
7. The observer verifies \(c = c'\).

4.7 Open

The open phase can only be performed by the CAA. Opening a signature allows the CAA to identify which specific group member is in violation. The opening of the signature is independent of the signing method used (IND-CPA or IND-CCA2).

1. UTM decrypts \(C\) as part of PKE by invoking \(\text{pke.dec}(sk_o, C) = \hat{R}\);
2. UTM verifies:
   \[e(rP, \hat{P}) = e(P, \hat{R})\]
   \[e(rP, \hat{P}') = e(P, r \hat{P})\]  By def. of \(\hat{R}\)
   \[e(P, \hat{P})' = e(P, r \hat{P}')\]  By def. bilinearity
   \[e(P, \hat{P})'' = e(P, \hat{P})'\]  By def. bilinearity

and if this holds take action on basis of the violation.

4.8 Digital Signature

As stated in the chapter 2 Derler et al. specifies that the DS scheme must be correct and be UEF-CMA secure [4]. The DS is only used during setup and join. This means the signature is not generated on the UAS, allowing the use of computational heavier schemes. There are enough options to choose from: a recent survey by J. Chia et al. [75] concluded that there are 16 different proven to be UEF-CMA secure DS schemes. As we aim to show that A²RID is compliant with DRIP, IETF reviewed DS schemes will only be considered. This produces the following possible DS schemes:

1. Leighton-Micali Signature (LMS) [76, 77];
2. Boneh-Lynn-Shacham (BLS) Signature [78, 79];
3. eXtended Merkle Signature Scheme (XMSS) [77, 80];
4. Boneh-Boyen-Schacham (BBS) Signature [81, 82];
5. Rivest-Shamir-Adleman (RSA) Blind Signatures [83, 84];
6. Edwards-Curve Digital Signature Algorithm (EdDSA) [85].

Of these 6 DS schemes, BBS does not possess the required setup. As explained in chapter 2 a DS scheme requires to have a ds.setup, ds.sign, and ds.verify phase. The BBS consists of 5 stages namely: setup, Sign, Verify, ProofGen and ProofVerify [81], resulting in incapability with our design requirements. For the remaining 5 possible schemes, their security assumptions and whether they are at least UEF-CMA secure are shown in Table 4.1.

Table 4.1: Security assumptions and minimal security level of IETF proposed DS schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Security assumption</th>
<th>UEF-CMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS</td>
<td>Preimage resistance and Second preimage resistance [86]</td>
<td>✓ [76]</td>
</tr>
<tr>
<td>BLS signature</td>
<td>Gap Diffie-Hellman computation [87, 88]</td>
<td>✓ [89]</td>
</tr>
<tr>
<td>XMSS</td>
<td>Preimage resistance and Second preimage resistance [86]</td>
<td>✓ [80]</td>
</tr>
<tr>
<td>RSA blind signature</td>
<td>Integer factorisation + Discrete Logarithm Problem [87]</td>
<td>x [90]</td>
</tr>
<tr>
<td>EdDSA</td>
<td>ECDLP [75]</td>
<td>✓ [75]</td>
</tr>
</tbody>
</table>

Based on the information presented in Table 4.1, we can see that LMS, BLS, XMSS and EdDSA are suitable options for our DS scheme. Given that we are not on a restricting platform and all 4 of the scheme meet the minimum requirements, there is no direct preference for one system above the other. For this project, we have chosen to use BLS.

The BLS digital signature scheme allows one to determine the authenticity of the signer. The security of BLS digital signatures relays on the Computational Diffie-Hellman Problem (CDHP) within a Gap Diffie-Hellman (GDH) group. GDH groups are cyclic groups with certain properties. Let us call groups possessing these properties G. G is a group generated by P of prime order p. Within these groups there exist an efficiently computable bilinear map (see section 2.3). Group G in addition possesses the following properties [91]:

1. The Computational Diffie-Hellman Problem (CDHP) within group G is solvable in polynomial time. The CDHP indicates the difficulty of computing \( abP \) with \( a, b \in G \) given \((P, aP, bP)\).

2. The Decisional Diffie-Hellman Problem (DDHP) within group G is not solvable in polynomial time. The DDHP indicates the difficulty of deciding whether \( c = ab \) with \( a, b, c \in G \) given \((P, aP, bP, cP)\).

So if the former can be solved in polynomial time while the latter cannot group G is called a GDH group. The BLS digital signature consists as required of three stages setup, sign and verify which will be discussed respectively.

4.8.1 Setup

In the setup phase, the key pair \((usk, upk)\) is generated and the public key upk is distributed. The private key generation is based on a HMAC-based Extract-and-Expand Key Derivation Function (HKDF). A normal Key Derivation Function (KDF) takes an Input Key Material (IKM) as input and derives a cryptographically strong secret key from it [92]. The difference with HKDF is already in the name, where a KDF makes no use of the properties of a hash function, the HKDF does. The HKDF is a two-step function: HKDF-Extract and HKDF-Expand. HKDF-Extract takes as input a salt (a non-secret value) and the IKM, the result is a Pseudorandom Key (PRK). The length of this key depends on the hash function used. The length in our case is 32 bytes [78]. The PRK serves as input for the HKDF-Expand step, as well as the required output length L, and additional optional key info. Both HKDF-Extract and HKDF-Expand take a input called Integer-to-Octet-String Primitive (I2OSP). As the name indicates I2OSP(x, 1en) takes a non-negative integer x and converts it to an octet (string...
CHAPTER 4. DYNAMIC GROUP SIGNATURES OF A2RID

of ordered eight-bit bytes) of the specified length [93]. Given that we require the BLS signature's secret key to be non-negative integer Octet-String-to-Integer Primitive (OS2IP) is used to convert the output of HKDF-Expand: Output Keying Material (OKM) to the BLS's Secret key [94]. Leading to the following steps of DS.set up (1^k):

1. The USS_u selects a curve of at least k--bit security;
2. The USS_u generates random Input Key Material (IKM) such that IKM \( \in \{0, 1\}^{64} \);
3. The USS_u requires a 32 byte key and specific such by setting key_info=32;
4. The USS_u sets the salt (a non secret value) to BLS-SIG-KEYGEN-SALT to increase the hash space;
5. The USS_u sets the length of PRK to \( L = \lceil (3 \cdot \lceil \log_2(r) \rceil) / 16 \rceil \);
6. The USS_u invokes the HKDF to create Output Keying Material (OKM);
7. The USS_u converts OKM to a non-negative integer by invoking OS2IP;
8. The USS_u computes public key point \( pk \) by multiplying usk with the curve generator \( P \) such that \( pk = usk \cdot P \);
9. The USS_u computes public key upk by mapping point \( pk \) to an octet by invoking I2OSP;
10. The USS_u stores the DS keypair \((upk, usk)\);
11. The USS_u shares upk with USS_g.

4.8.2 Signing

The signing phase of the BLS scheme is more straightforward. A signature on message \( m \) using secrete key \( usk \) is defined as:

1. The USS_u takes input message \( \hat{C} \) and maps it to a point on the curve called \( Q \);
2. The USS_u calculates a new point \( R \) by multiplying the DS secret key \( usk \) by point \( Q \) such that \( R = usk \cdot Q \);
3. The USS_u takes point \( R \) and maps it to a non-negative value resulting in signature \( \sigma \).

4.8.3 Verify

The verification of the signature \( \sigma \) requires pairing, and the use of DS's public key upk and message \( m \). Which gives the following verification method:

1. The USS_g converts signature \( \sigma \) to a point on the curve called \( R \);
2. The USS_g check whether \( R \) is a point on the curve;
3. The USS_g converts public key \( upk \) to a point on the curve called \( xP \);
4. The USS_g checks the pairing

\[
\begin{align*}
    e(R, P) &= e(Q, xP) & \text{By def. of } C_1 \text{and } C_2 \\
    e(R, P) &= e(Q, sk \cdot P) & \text{By def. of } xP \\
    e(sk \cdot Q, P) &= e(Q, sk \cdot P) & \text{By def. of } R \\
    e(Q, P)^k &= e(Q, sk \cdot P) & \text{By def. of bilinearity} \\
    e(Q, P)^k &= e(Q, P)^k & \text{By def. of bilinearity} \\
    e(Q, P) &= e(Q, P) & \text{By def. of bilinearity}
\end{align*}
\]

if this holds the signature is verified.
4.9 Non-interactive Zero Knowledge Proof

As for the other schemes compliance with IETF is preferred. The protocol IETF only mentions one NIZK proof namely: Schnorr’s protocol. is also the most common scheme used in literature overall [95] Schnorr’s protocol is a widely used protocol based on the concepts of discrete logarithms [96, 95]. This protocol allows a prover to show knowledge of a given discrete logarithm without giving anything away about its value. The protocol originally is an interactive protocol consisting of three phases. The witness helps one to efficiently verify if the given statement is true. The original approach has three points of interaction with the verifier. In the next section, the non-interactive version will be introduced. The NIZK is in direct relation with the original interactive version. To transform Schnorr’s to the non-interactive version we use the Fiat–Shamir heuristic [97]. The interactive steps in the original proof are replaced by non-interactive random oracles or in our case a hash function [98].

Setup

The only setup required by an NIZK is an understanding of the curve over which the proof is calculated. The curve is defined by its order $p$, and generators $P$ and $\hat{P}$ of groups $G_1$ and $G_2$ respectively. This leads to the following two steps:

1. The USS$_g$ selects a curve as CRS;
2. The USS$_g$ publishes the CRS.

Proof

The proof $\pi$ is generated over curve CRS given witness $A$ and statement $a$ such that:

1. The USS$_u$ selects the curve defined by CRS;
2. The USS$_u$ generates nonce $v \leftarrow \mathbb{Z}_p$;
3. The USS$_u$ computes $V = v \cdot P$;
4. The USS$_u$ computes hash $c = H(P|V|A)$ where $A$ is our witness;
5. The USS$_u$ calculates $r = v - a \cdot c \mod p$;
6. The USS$_u$ creates proof $\pi = (V, c, r)$.

Verify

Given proof $\pi$ over curve CRS, the proof is either verified or declined.

1. The USS$_g$ retrieves CRS;
2. The USS$_g$ parses proof $\pi$ such that $\pi = (V, c, r)$;
3. The USS$_g$ calculates $V$ based on proof $\pi$ such that $V = r \cdot P + c \cdot A$;
4. The USS$_g$ calculates hash $c'$ such that $c' = H(P|V|A)$;
5. The USS$_g$ verifies $c == c'$. 
4.10 Public Key Encryption

The IETF does not propose any PKE scheme. The PKE scheme must consist of a non-interactive key generation phase, independent encryption and independent decryption phases. In addition the PKE scheme must be IND-CPA and IND-CCA2 secure. This provides us with the following options:

1. Cramer–Shoup cryptosystem [99];
2. Elliptic Curve Integrated Encryption Scheme (ECIES) [100];
3. Sponge Based Asymmetric Encryption Padding (SpAEP) [101].

Given that we are again not in a restrictive environment we freely choose any of the above. In this project we choose Elliptic Curve Integrated Encryption Scheme (ECIES) as our PKE scheme. The Elliptic Curve Integrated Encryption Scheme (ECIES) is a PKE scheme comprised out of a five cryptographic primitives [102, 103]:

1. A hash function: a deterministic mapping of data of an arbitrary to data of a fixed size;
2. An encryption algorithm, a symmetric encryption scheme;
3. A Key Derivation Function (KDF), on the input of IKM cryptographic secure keys are outputted;
4. Medium Access Control (MAC), a short tag of information used to authenticate the message;
5. Key Agreement (KA), a protocol allowing the sender and the receiver to agree on their shared key.

These 5 different cryptographic primitives are used throughout the ECIES to establish an IND-CPA and IND-CCA2 secure encryption scheme. Next, the 3 phases of the ECIES scheme are discussed.

4.10.1 Setup

In the setup phase, an agreement is reached about which elliptic curve is used throughout the ECIES algorithm. The receiver/recipient requires to have a key pair, where \( pk_O \) is used in the encryption of the message.

1. The USS\(_g\) selects a curve of at least \( k \)-bit security;
2. The USS\(_g\) generates \( sk_O \) by selecting \( sk \leftarrow Z_p \);
3. The USS\(_g\) calculates \( pk = sk \cdot P \);
4. The USS\(_g\) sets the PKE key pair to \((sk_O, pk_O)\).

4.10.2 Encryption

Based on the recipients’ public and a newly generated empirical key pair (generated in steps 1 and 2) the KA creates the corresponding secret value (step 3). The secret value generated for each message to be sent acts as input for the KDF. Based on the secret value the KDF outputs the secret symmetric encryption key \( k_{\text{ENC}} \) and MAC key \( k_{\text{MAC}} \). Where \( k_{\text{ENC}} \) is used to encrypt the message, to provide confidentiality. The \( k_{\text{MAC}} \) key is used to produce \( \text{tag}_{\text{ENC}} \) which will be used for authentication purposes.
1. The USS$_u$ generates nonce $u \leftarrow \mathbb{Z}_p$;
2. The USS$_u$ calculates $U = u \cdot P$;
3. The USS$_u$ generates key pair $\text{MAC}, k_{\text{ENC}}$ by invoking $KDF$ such that $KDF(V) = (k_{\text{MAC}}, k_{\text{ENC}})$;
4. The USS$_u$ generates cipher $C$ by invoking $\text{ENC}(k_{\text{ENC}}, m)$;
5. The USS$_u$ generates a tag by invoking $\text{MAC}(k_{\text{MAC}}, c)$;
6. The USS$_u$ sets the cipher to $c = (U, C, \text{tag})$.

### 4.10.3 Decrypt

Because of the nature of the KA, and the mathematical relation between the sender ephemeral public key the secret value can be recomputed.

1. The USS$_g$ parses $c$ such that $c = (U, \text{cipher, tag}_{\text{ENC}})$;
2. The USS$_g$ computes $V = sk_o \cdot U$;
3. The USS$_g$ computes $(k_{\text{MAC}}, k_{\text{DEC}}) = KDF(V)$;
4. The USS$_g$ decrypts $c$ such that the result is plaintext $m = \text{DEC}(k_{\text{DEC}}, c)$;
5. The USS$_g$ creates $\text{tag}_{\text{DEC}}$ by invoking $\text{tag}_{\text{DEC}} = \text{MAC}(k_{\text{MAC}}, c)$;
6. The USS$_g$ verifies that $\text{tag}_{\text{DEC}} = \text{tag}_{\text{ENC}}$.

### 4.11 Structure-Preserving Signatures on Equivalence Classes

While the choice of a PKE, the DS and the NIZK was only limited by a set of requirements, the Structure-Preserving Signatures on Equivalence Classes (SPS-EQ) is provided. The SPS-EQ scheme is introduced by G. Fuchsbauer et al. in [72].

#### 4.11.1 Setup

The signature scheme uses a type-3 pairing-friendly curve to perform its operations. Over this pairing-friendly curve, a bilinear group $BG$ is defined. This bilinear group $BG$ is used to define our key pair $(sk_R, pk_R)$ which is generated in the setup phase.

1. The USS$_g$ selects at random secret key $sk_R$ such that $sk_R = (sk_{R1}, sk_{R2}) \leftarrow (\mathbb{Z}_p)^2$;
2. The USS$_g$ computes public key $pk_R$ such that $pk_R = (pk_{R1}, pk_{R2}) = (sk_{R1} \cdot P, sk_{R2} \cdot P)$;
3. The USS$_g$ set the SPS-EQ key pair to $(pk_R, sk_R)$.

#### 4.11.2 Sign

The signature phase is performed on a UAS. The sign phase does not use any pairing operations and is therefore fit to be performed on a computational limited device. It is assumed that the UAS is aware of which type-3 curve is used to perform our point multiplication. Message $M$ is signed using secret key $sk_R$.

1. The USS$_u$ generates nonce $y \leftarrow \mathbb{Z}_p$;
2. The USS computes \( Y = \frac{1}{y} P; \)
3. The USS computes \( \hat{Y} = \frac{1}{\hat{y}} \hat{P}; \)
4. The USS computes \( Z = y \cdot (sk_{R_1} M_1 + sk_{R_2} M_2); \)
5. The USS creates signature \( \sigma = (Z, Y, \hat{Y}). \)

### 4.11.3 Verify

The verification can be performed by any observer given that the \( pk_R \) is part of the Public Registry (as shown in Figure 4.1). Given that the verification of signature \( \sigma \) is performed on an observer device we can assume this device can perform pairing.

1. The observer parses \( M \) into \( M_1 \) and \( M_2; \)
2. The observer checks whether the following holds:

\[
\begin{align*}
    e(M_1, pk_1) e(M_2, pk_2) &= e(Z, \hat{Y}) & \text{By def. of \( pk \)} \\
    e(M_1, sk_1 \cdot \hat{P}) e(M_2, sk_2 \cdot \hat{P}) &= e(Z, \hat{Y}) & \text{By def. of bilinearity} \\
    e(M_1, \hat{P})^{sk_1} e(M_2, \hat{P})^{sk_2} &= e(Z, \hat{Y}) & \text{By def. of bilinearity} \\
    e(y(M_1 sk_1 + M_2 sk_2), \hat{Y}) &= e(y(M_1 sk_1 + M_2 sk_2), \frac{1}{\hat{y}}) & \text{By def. of} \ Z \\
    e(y(M_1 sk_1 + M_2 sk_2), \hat{P}) &= e(y(M_1 sk_1 + M_2 sk_2), \hat{P})^{\frac{1}{\hat{y}}} & \text{By def. of} \ \hat{Y} \\
    e(M_1, \hat{P})^{sk_1} e(M_2, \hat{P})^{sk_2} &= e(M_1, \hat{P})^{sk_1} e(M_2, \hat{P})^{sk_2} & \text{By def. of bilinearity} \\
    e(M_1 sk_1 + M_2 sk_2, \hat{P})^y &= e(M_1 sk_1 + M_2 sk_2, \hat{P})^\hat{y} & \text{By def. of linearity [104]} \\
    e(M_1, \hat{P})^{sk_1} e(M_2, \hat{P})^{sk_2} &= e(M_1, \hat{P})^{sk_1} e(M_2, \hat{P})^{sk_2} & \text{By def. of bilinearity}
\end{align*}
\]

### 4.11.4 Changing the Representative Class

The randomisation of the UAS secret key. Since the ratio between the original value and the new one is kept recovery is possible. Based on input message \( M \) and random nonce \( \rho \) the new signature is calculated.

1. The UAS parses \( M \) into \( M_1 \) and \( M_2; \)
2. The UAS computes the new key such that \( (M_1, M_2) = (\rho M_1, \rho M_2) \)
3. The UAS generates nonce \( \psi \in \mathbb{Z}_p; \)
4. The UAS parses signature \( \sigma \) such that \( \sigma = (Z, Y, \hat{Y}); \)
5. The UAS computes \( Z \) such that \( Z = \mu \psi Z; \)
6. The UAS computes \( Y \) such that \( Y = \frac{1}{\psi} Y; \)
7. The UAS computes \( \hat{Y} \) such that \( \hat{Y} = \frac{1}{\psi} \hat{Y}; \)
8. The UAS sets signature \( \sigma \) to \( \sigma = (Z, Y, \hat{Y}). \)

---

A²RID: An Anonymous RemoteID-compliant Group Signature Scheme for Commercial Drones

---

\( \frac{1}{y} \), \( \frac{1}{\hat{y}} \), \( \mu \psi \), \( \psi \), \( \frac{1}{\psi} \), \( \frac{1}{\psi} \hat{Y} \).
4.12 DRIP Working Group Requirements

To show that A²RID satisfies the DRIP requirements set by the IETF working group, we will list them in the coming sections. The DRIP requirements are divided into three different categories: general, identifier and privacy, of which the MUST requirements will be discussed respectively.

4.12.1 General

The general requirements relate to the over operation of the DRIP protocols. Normative general requirements set by DRIP working group are:

1. **Requirement**  "Provable Ownership: DRIP enable verification that the asserted entity (typically UAS) ID is that of the actual current sender (i.e., the Entity ID in the DRIP authenticated message set is not a replay attack or other spoof), even on an Observer device lacking Internet connectivity at the time of observation." [105]
   
   **Explanation**  A²RID is coalition-resistant, exculpable, unforgeable and can be authenticated by observer devices that lack Internet connection.

2. "**Requirement**  Provable Binding: DRIP enable the cryptographic binding of all other messages from the same actual current sender to the UAS ID asserted in the Basic ID Message. " [105]
   
   **Explanation**  Traceability allows the CAA to track all message to a group membership and the UTM can trace the UAS ID of the broadcast.

3. **Requirement**  "Provable Registration: DRIP enable cryptographically secure verification that the UAS ID is in a registry and identification of that registry, even on an Observer device lacking Internet connectivity at the time of observation; the same sender may have multiple IDs, potentially in different registries, but each ID must clearly indicate in which registry it can be found." [105]
   
   **Explanation**  Each UAS is a member of a group. All UAS a required to send broadcast messages every second. Each broadcast message contains a dedicated section for the group ID.

4. **Requirement**  "Readability: DRIP enable information (regulation required elements, whether sent via UAS RID or looked up in registries) to be read and utilized by both humans and software." [105]
   
   **Explanation**  No encryption on the broadcast message is used hence the message with accompanied signature is readable by humans and software.

5. **Requirement**  "Gateway: DRIP enable application-layer gateways from Broadcast RID to Network RID to stamp messages with precise date/time received and receiver location, then relay them to a network service (e.g., SDSP or distributed ledger) whenever the gateway has Internet connectivity." [105]
   
   **Explanation**  All communication besides the UASs broadcasts is Out-of-scope.

6. **Requirement**  "Contact: DRIP enable dynamically establishing, with AAA, per policy, strongly mutually authenticated, end-to-end strongly encrypted communications with the UAS RID sender and entities looked up from the UAS ID, including at least the (1) pilot (Remote Pilot or Pilot In Command), (2) the USS (if any) under which the operation is being conducted, and (3) registries in which data on the UA and pilot are held. This requirement applies whenever each party to such desired communications has a currently usable means of resolving the other party's DRIP Entity Identifier to a locator (IP address) and currently usable bidirectional IP (not necessarily Internet) connectivity with the other party." [105]
   
   **Explanation**  All communication besides the UASs broadcasts is out-of-scope.
CHAPTER 4. DYNAMIC GROUP SIGNATURES OF A²RID

   **Explanation** All communication besides the UASs broadcasts is out-of-scope.

8. **Requirement** "Mobility: DRIP support physical and logical mobility of UA, GCS, and Observers. DRIP support mobility of essentially all participating nodes (UA, GCS, Observers, NET-RID SP, NET-RID DP, Private Registries, SDSP, and potentially others as RID and UTM evolve)." [105]  
   **Explanation** All communication besides the UASs broadcasts is out-of-scope.

9. **Requirement** Multihoming: DRIP support multihoming of UA and GCS, for make-before-break smooth handoff and resiliency against path or link failure. DRIP support multihoming of essentially all participating nodes." [105]  
   **Explanation** All communication besides the UASs broadcasts is out-of-scope.

4.12.2 **Identifier**

Identifier requirements relate to the ID subfield in the broadcast message. Normative Identifier requirements set DRIP working group are:

1. **Requirement** "Length: The DRIP Entity Identifier be longer than 19 bytes, to fit in the Specific Session ID subfield of the UAS ID field of the Basic ID Message of the proposed revision of (at the time of writing)." [105]  
   **Explanation** The ID present on the UAS is a group ID. Multiple UAS are members of group, making the ID number smaller. In A²RID 4 bytes are dedicated to the ID (see chapter 6).

2. **Requirement** "Registry ID: The DRIP identifier be sufficient to identify a registry in which the entity identified therewith is listed." [105]  
   **Explanation** In A²RID the group ID present in the broadcast identifies the group the UAS is a member of. This information can be used by the UTM which tries to open the message by searching in their registry for the \((\hat{C}, \sigma)\) combination that decrypts \(R = \text{pke.dec}(sk_O, \hat{C})\) such that \(e(r_P, \hat{P}) = e(p, R)\) holds.

3. **Requirement** "Entity ID: The DRIP identifier be sufficient to enable lookups of other data associated with the entity identified therewith in that registry." [105]  
   **Explanation** Same as above.

4. **Requirement** "Uniqueness: The DRIP identifier be unique within the applicable global identifier space from when it is first registered therein until it is explicitly deregistered therefrom (due to, e.g., expiration after a specified lifetime, revocation by the registry, or surrender by the operator)" [105]  
   **Explanation** The A²RID can dynamically add new UAS to a group.

5. **Requirement** "Non-spoofability: The DRIP identifier be spoofable within the context of a minimal Remote ID broadcast message set (to be specified within DRIP to be sufficient collectively to prove sender ownership of the claimed identifier)." [105]  
   **Explanation** A²RID is unforgable as proven in [4, 18].

6. **Requirement** "Unlinkability: The DRIP identifier facilitate adversarial correlation over multiple operations. If this is accomplished by limiting each identifier to a single use or brief period of usage, the DRIP identifier support well-defined, scalable, timely registration methods." [105]  
   **Explanation** A²RID is unlinkable as proven in [4, 18].
CHAPTER 4. DYNAMIC GROUP SIGNATURES OF A²RID

4.12.3 Privacy

Privacy requirements relate to handling of privacy sensitive data, required by the DRIP protocol. Normative privacy requirements set by DRIP working group are:

1. **Requirement** "Confidential Handling: DRIP enable confidential handling of private information (i.e., any and all information that neither the cognizant authority nor the information owner has designated as public, e.g., personal data)." [105]
   **Explanation** The processing, storing, transfer of data is out-of-scope.

2. **Requirement** "Encrypted Transport: DRIP enable selective strong encryption of private data in motion in such a manner that only authorized actors can recover it. If transport is via IP, then encryption be end-to-end, at or above the IP layer. DRIP encrypt safety critical data to be transmitted over Broadcast RID in any situation where it is unlikely that local Observers authorized to access the plaintext will be able to decrypt it or obtain it from a service able to decrypt it. DRIP encrypt data when/where doing so would conflict with applicable regulations or CAA policies/procedures, i.e., DRIP support configurable disabling of encryption." [105]
   **Explanation** The processing, storing, transfer of data is out-of-scope.
Chapter 5

Security Analysis

In chapter 4 the A²RID protocol has been introduced. To show that we satisfy our security requirements we will dedicate this chapter to showing a formal cryptographic verification.

5.1 ProVerif

ProVerif is a tool that can be used for the verification of cryptographic protocols [106]. In our case, the protocol of interest is A²RID. A²RID is based on different cryptographic primitives, which enable us to create:

- Cryptographic keying material;
- Cryptographic hashes;
- Digital signatures.

These cryptographic primitives and materials are however not the objective of ProVerif. ProVerif solely focuses on the communication aspect of the protocol. Since A²RID uses a broadcast channel to transmit a message, the transport medium can not be trusted. The goal of ProVerif is to show, even when an untrusted channel is used, that:

1. The long-term identity of an UAS is not revealed and therefore stays secret overtime;
2. Broadcast messages sent by an UAS that is a legit group member can always be authenticated.

Ideally, we would like to show that the verification summary of IND-CPA-anonymous and the IND-CCA2-anonymous scheme are the same. Given that the difference between IND-CPA and IND-CCA2 anonymity is not related to the messaging part of the protocol, but the capabilities of the attacker.

The ProVerif tool assumes the Dolev-Yao attacker model [107]. In this model, the attacker/adversary has control over all communication channels. However it can not attack the cryptographic primitives mentioned earlier and even a brute force attack is not considered. The attacker can however [108]:

- Read a message and block further transmission;
- Decompose the broadcast messages and remember the decomposed parts;
- Generate fresh new (non cryptographic) data;
- Compose a new broadcast message from known data and broadcast the message.
CHAPTER 5. SECURITY ANALYSIS

The attacker’s abilities are all communication related. To prove anonymity and authenticity, we check three different properties:

1. Sender authenticity;
2. Impersonation resistance;
3. Non-interference.

The authenticity of the A²RID broadcast is checked by using ProVerif correspondence assertions. Correspondence assertions are used to capture a relationship between different events. In our case, we want to show that a broadcast \( x \) has been sent by UAS as a member of the group \( G \) that only then \( x \) can be verified by the observer using the public parameters of the group \( G \). Resulting in a ProVerif query of the form:

\[
event(last\_event()) \Rightarrow event(previous\_event()) \text{ is true} \quad (5.1)
\]

with \( last\_event \) being the verification of the signature, and \( previous\_event() \) the broadcasting of the message. The query only evaluates to true when \( previous\_event() \) is really executed before \( last\_event \). In addition, we want to check that only if broadcast \( x \) is been sent by an UAS as a member of the group \( G \) that only then the UTM can open broadcast \( x \).

Impersonation resistance is shown by using ProVerif properties: reachability and secrecy. Reachability and secrecy are used to show whether on not secret information is in reach of the attacker. In our case, we check whether \( gsk \) is in reach of the attacker. Resulting in a ProVerif query of the form

\[
\neg attacker(gsk) \text{ is true.} \quad (5.2)
\]

Strong secrecy/non-interference is used to check whether the attacker can deduce any information about the value of \( gsk \) based on the broadcast messages.

Implementing these queries in ProVerif gives us the verification summary as shown in Listing 5.1.1. Showing that anonymity has been verified by ProVerif.

**Listing 5.1.1: ProVerif verification summary of A²RID**

- Query \( \text{event(sig\_verified(gpk))} \Rightarrow \text{event(send\_message(gpk)) is true.} \)
- Query \( \text{event(sig\_open(gpk))} \Rightarrow \text{event(send\_message(gpk)) is true.} \)
- Query \( \neg \text{attacker(gsk) is true.} \)
- Non-interference \( \text{secret\_gsk is true.} \)

The ProVerif tool confirms that our three security properties we aimed to proof hold. First, only broadcasts sent by a group member can be authenticated by an observer. Second, no secret information is leaked while a broadcast is sent. Lastly, the attacker can not distinguish broadcast on the basis of which key is used to generate the signature.
Chapter 6

Implementation

In this chapter, the details regarding the implementation of $A^2\text{RID}$ are given. All sources used for the implementation are freely available, allowing anyone to reproduce the results. In addition, one can test the application autonomously as well, since the source code is published under GNU General Public License [17]. This section will introduce and discuss the platform and dependencies on which $A^2\text{RID}$ relies.

6.1 ESPcopter

The ESPcopter is a programmable mini-drone [109]. Besides being programmable, the ESPcopter is wirelessly networkable and interactive and can be expended by development modules. The modules include, among other things, an ESPcopter Altitude Hold Module and an ESPcopter Temperature, Pressure and Humidity Module [110]. All of these add-ons and the ESPcopter itself are powered by the ESP8266 microcontroller, produced by EspressIf [111]. This leads to a low-level mini-drone with the specifications shown in Table 6.1.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>250mAh Li-Po</td>
</tr>
<tr>
<td>Processor</td>
<td>L106 32-bit RISC microprocessor</td>
</tr>
<tr>
<td>Flash</td>
<td>4MB</td>
</tr>
<tr>
<td>CPU frequency</td>
<td>80/160 MHz</td>
</tr>
<tr>
<td>Wi-fi module</td>
<td>WiFi 802.11 b/g/n</td>
</tr>
<tr>
<td>Flash</td>
<td>16 MBytes</td>
</tr>
<tr>
<td>RAM</td>
<td>160 KBytes</td>
</tr>
</tbody>
</table>

The specifications as shown in Table 6.1 indicate that the ESPcopter is one of the most restricted research/educational platforms on which one can implement, run and test protocols. For this specific reason, the ESPcopter is a suitable choice to show whether or not it is possible to create an anonymous and variable broadcast scheme, given the computational restrictions on the device. To program $A^2\text{RID}$ on the ESPcopter, one needs the following 4 freely available programs and libraries [112]:

1. An Arduino Integrated Development Environment (IDE) [113];
2. The ESP8266 library [114];
3. The ESPcopter library [115];
4. A suitable set of drivers to recognise the ESPcpter connection [116].

After one has installed these on the ESPcpter, it can be programmed to enable \( A^2\text{RID} \) broadcast messages.

### 6.2 Cryptographic Libraries

The group signature scheme on which \( A^2\text{RID} \) is based is presented in detail in section 2.7. The signing phase of the signature scheme is performed on the UAS. This means that we need to be able to perform the following operations on the constrained device:

- Modular arithmetic of big numbers;
- Elliptic curve point arithmetic on \( G_1 \) points;
- Elliptic curve point arithmetic on \( G_2 \) points;
- Hashing of elliptic curve points and big numbers.

This requires us to choose a cryptographic library designed for constrained devices. In Table 6.2 different Open-source IoT friendly cryptographic libraries are shown and compared to our requirements.

Note that even though the UAS does not perform any pairing, the observer does. To increase the chances of successful employability, the same library is used to perform operations on the UAS side and on the observer side. Of the libraries shown in Table 6.2, only the MIRACL library and the WolfCrypt library satisfy all our requirements. The IETF evaluated the security of the implementation of the pairing-friendly curves available in Miracl [42], while they did not for WolfCrypt. In addition the Miracl documentation is actively showing they are following the IETF standard [123]. For this thesis, we want to show that \( A^2\text{RID} \) is compatible with DRIP, and therefore we have opted to use the Miracl library.
Table 6.2: Open-source IoT-friendly cryptographic libraries.

<table>
<thead>
<tr>
<th>Library</th>
<th>Language</th>
<th>Supported on ESPcopter</th>
<th>Bilinear Pairing</th>
<th>Big Number Modular Arithmetic</th>
<th>Hash</th>
<th>Elliptic curve Point Operations</th>
<th>Curve selection</th>
<th>No dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBC [117]</td>
<td>C C++ C C C</td>
<td>✓ ✓ ✓ ✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TinyECC [118]</td>
<td>C</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RelicToolKit [119]</td>
<td>C</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mbedtls [120]</td>
<td>C</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>wolfCrypt [121]</td>
<td>C</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Miracl [122]</td>
<td>C/C++/Go/JavaScript/Python/Rust/Swift</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

6.3 Selection of the Elliptic Curve

The MIRACL library provides a variety of curves. Given that we require the curve to be pairing-friendly the number of suitable solutions is shortened. Also the device, i.e. the ESPcopter equipped with the ESP8266 chip, restricts the choice of a suitable curve even further. To select a suitable pairing-friendly curve we have to look at the security level and computational requirements the curve posses. The operation the ESPcopter has to perform online are as stated in section 6.2, of these 4 different types operations the multiplication of points $G_1$ and $G_2$ processing-intensive operations to execute. Elliptic curve multiplication is achieved by point addition and point doubling.

In Figure 6.1, a selection of the pairing-friendly curves available by Miracl are shown. While Miracl gives us the option to use one of their 12 curves, the ESPcopter is only able to perform elliptic curve point multiplication in both $G_1$ or $G_2$, for the 5 curves (crashes otherwise) in Figure 6.1. As Figure 6.1 shows the most time-efficient curve is BN254. Given that RemoteID requires a message to be sent every second, the A²RID implementation will be using the BN254 curve. Based on Table 2.2 and Table 2.3 given in section 3.2, it seems like the BN254 does not fulfil the requirements set by NIST. NIST however made specific recommendations for elliptic curve cryptography stating that if $p = 224 - 255$ its classified as 112-bit security [3]. Based on these requirements the BN254 curve can be used for testing, when one wants to use a more secure curve, the settings of the Miracl Library can be easily adjusted.
Figure 6.1: Pairing-friendly curves that can perform multiplication of a big number with a \( G_1 \) and a \( G_2 \) elliptic curve point on the ESPcopter.
6.4 Configuration Details

As stated in section 6.3, the curve of choice is the BN254, which is of the form $y^2 \equiv x^3 + Ax + B$. The parameters of BN254 can be found in the appendix (see Appendix B).

As becomes clear from Table B.1, the size of the broadcast depends on the curve used, given that the curve determines the modulus of the operations. The transport layer protocol User Datagram Protocol (UDP) is used to RemoteID broadcast the message. To send our broadcast using UDP, we require a link layer connection:

1. Connect the ESPcopter to a Wireless Access Point (WAP) (e.g. a hot-spot);
2. Use the ESPcopter as a station.

By doing so, the ESPcopter is part of the Local Area Network (LAN). To send our message as a broadcast we need to define the broadcast IP and port number. The packet header will be sent following the UDP header format [124]. In Table 6.3 the sizes of each section of the broadcast message are given. As can be seen, the broadcast size is either 945 or 1524 bytes depending on whether IND-CPA or IND-CCA2 is used respectively.

Table 6.3: A²RID payload based on content size provided by ARID [6].

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Size [B]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>906 or 1485</td>
<td>IND-CPA-anonymous signature OR IND-CCA2-anonymous signature</td>
</tr>
<tr>
<td>UAS\textsubscript{id}</td>
<td>4</td>
<td>UAS group ID.</td>
</tr>
<tr>
<td>UAS\textsubscript{lat}</td>
<td>4</td>
<td>UAS latitudinal position</td>
</tr>
<tr>
<td>UAS\textsubscript{lon}</td>
<td>4</td>
<td>UAS longitudinal position</td>
</tr>
<tr>
<td>UAS\textsubscript{alt}</td>
<td>4</td>
<td>UAS altitudinal position</td>
</tr>
<tr>
<td>UAS\textsubscript{vx}</td>
<td>2</td>
<td>UAS speed w.r.t x axis</td>
</tr>
<tr>
<td>UAS\textsubscript{vy}</td>
<td>2</td>
<td>UAS speed w.r.t y axis</td>
</tr>
<tr>
<td>UAS\textsubscript{vz}</td>
<td>2</td>
<td>UAS speed w.r.t z axis</td>
</tr>
<tr>
<td>GCS\textsubscript{lat}</td>
<td>4</td>
<td>GCS latitudinal position</td>
</tr>
<tr>
<td>GCS\textsubscript{lon}</td>
<td>4</td>
<td>GCS longitudinal position</td>
</tr>
<tr>
<td>GCS\textsubscript{alt}</td>
<td>4</td>
<td>GCS altitudinal position</td>
</tr>
<tr>
<td>TS</td>
<td>4</td>
<td>Message timestamp</td>
</tr>
<tr>
<td>ES</td>
<td>1</td>
<td>UAS Emergency status</td>
</tr>
</tbody>
</table>
CHAPTER 6. IMPLEMENTATION

A²RID: An Anonymous RemoteID-compliant Group Signature Scheme for Commercial Drones
Chapter 7

Evaluation and Results

This chapter will report the results of the $A^2$RID protocol implemented on the ESPcopter on the pairing-friendly curve BN254. This section will first explain the implementation challenges that resulted in two different $A^2$RID schemes. After the two different implementations are explained, their execution time is evaluated. Next, the energy consumption of the different schemes is estimated. Lastly, the bandwidth of this scheme and potential other curves are evaluated.

7.1 Time

The implementation of the IND-CPA and IND-CCA2-anonymous $A^2$RID schemes must fulfil the requirements set RemoteID and should adhere the DRIP requirements and recommendations. The implementation results showed that both versions (IND-CPA and IND-CCA2) take more than one second to send a broadcast. This is in contradiction with the RemoteID requirements. Therefore, another version of the $A^2$RID scheme has been created. This version has the same online and offline phases as the original scheme with the difference that values are pre-computed. This can be achieved in several ways:

- The UAS has enough memory and computational power to pre-compute values in a reasonable time. The UAS needs to be turned on for a given time before takeoff allowing time for pre-computations.

- The UAS has some memory and not enough computational power to pre-compute values in a reasonable time. The UAS needs to be connected to a desktop before take-off to load the pre-computed values.

- The UAS has enough memory and not enough computational power to pre-compute values in a reasonable time. All possible options are pre-computed and loaded on the UAS during the join phase.

By allowing pre-computed values, we get the time measurements as shown in Figure 7.1. The execution time of all $A^2$RID phases is shown. Note that the signing phase is executed on the ESPcopter. The join, verify, and open phases have been executed on an HP ZBook Studio G5 laptop equipped with two Intel(R) Core(TM) i7-9750H CPUs running at 2.60GHz. As can be seen in Figure 7.1, the plain IND-CPA and IND-CCA2 require more than 1 second, while the pre-computation version are well within the 1 second limit. The exact values of the signing phases can be found in Table 7.1. The ESPcopter has a maximum flight time of 7 minutes, requiring 437185 and 6.60096 MBytes of storage for IND-CCA2 and IND-CPA respectively as can be seen in Figure 7.2a. The big difference in memory requirements between IND-CPA and IND-CCA2 comes from the fact that in IND-CCA2 values depend on other
Figure 7.1: Time measurements of each phase of the A^2RID scheme. The signing phase are executed on the ESPcopter. The join, verify, and open phases have been executed on a HP ZBook Studio G5 laptop equipped with two Intel(R) Core(TM) i7-9750H CPUs running at 2.60GHz.

values requiring multiple combinations to be computed. One must note that these memory requirements are Miracl specific. If not Miracl was used but another library that allows efficient reading of elliptic curve points from memory we require the memory usage as shown in Figure 7.2b.
Table 7.1: The maximum, mean and minimum time the $A^2$RID signing phase acquired over 50 iterations on the ESPcopter.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time min. [ms]</th>
<th>Time mean [ms]</th>
<th>Time max. [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CPA-$A^2$RID</td>
<td>1784.019</td>
<td>1784.08342</td>
<td>1784.291</td>
</tr>
<tr>
<td>Pre-comp IND-CPA-$A^2$RID</td>
<td>137.858</td>
<td>161.93528</td>
<td>164.168</td>
</tr>
<tr>
<td>IND-CCA2-$A^2$RID</td>
<td>4616.32</td>
<td>4616.36962</td>
<td>4616.668</td>
</tr>
<tr>
<td>Pre-comp IND-CCA2-$A^2$RID</td>
<td>195.789</td>
<td>219.42584</td>
<td>222.23</td>
</tr>
</tbody>
</table>

(a) The memory required to store the pre-computed values for $A^2$RID on the ESPcopter for IND-CPA and IND-CCA2 anonymous schemes while using the Miracl library.

(b) The memory required to store the pre-computed values for $A^2$RID on the ESPcopter for IND-CPA and IND-CCA2 anonymous schemes while not using the Miracl library.

Figure 7.2: The memory required to store the pre-computed values for $A^2$RID on the ESPcopter for IND-CPA and IND-CCA2 anonymous schemes, depending on whether Miracl is used.
7.2 Energy Consumption

To measure the exact current consumption we will use a RIGOL DS1202Z-E oscilloscope in combination with a 0.1Ω resistor, as can be seen in Figure 7.3. By using an 0.1Ω resistor we can create a circuit as shown in Figure 7.3a, where the resistor acts as a shunt resistor. This way we can get the current required by the ESPcopter by measuring the voltage over the resistor.

The electrical energy is computed using:

$$E = \left( \frac{V^2}{r} \right) \cdot t$$  \hspace{1cm} (7.1)

where $E$ is the electrical energy in Joule, $V$ is the voltage in volt, $r$ is the shunt resistor in ohm and $t$ is the time in seconds. For the voltage, the operating voltage of the Li-Po battery is used which is 3.7V. Using the setup as shown in Figure 7.3a this gives us the results as shown in Figure 7.4. The results are based on the values shown on the graphical interface of the oscilloscope. The setup is prone to errors (see Figure 7.5) by dialling down the intensity a peak can be seen. This peak is the combination of the energy required by $A^2$RID and the energy required to turn on a led. After the value of led is subtracted we are left with the energy consumption of $A^2$RID. The results as expected show that the more time consuming schemes requires more electric energy. The Li-Po battery used to power the ESPcopter is rated at a capacity of 0.260 Ampere-Hours at 3.7V [110]. The amount of electric energy stored in one full charge of the Li-Po battery can be defined by:

$$E = q \cdot V \cdot 3600$$  \hspace{1cm} (7.2)

with $E$ the electric energy in Joules, $q$ the electric charge in ampere-hours, $V$ the voltage in volt. From Equation 7.2 we get a total of 3463.2 Joules stored in the attached Li-Po battery. This shows that the use $A^2$RID only takes a fraction of the total available electric energy. The percentage of electric energy in the Li-Po battery dedicated to the generation of one $A^2$RID compliant broadcast message is given in Table 7.2.

<table>
<thead>
<tr>
<th>Percentage of battery</th>
<th>IND-CPA</th>
<th>Pre-comp. IND-CPA</th>
<th>IND-CCA2</th>
<th>Pre-comp. IND-CCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-comp. IND-CPA</td>
<td>2.14768-05%</td>
<td>2.67347e-07%</td>
<td>1.21412e-4%</td>
<td>3.67949e-07%</td>
</tr>
</tbody>
</table>

Table 7.2: Percentage of electric energy in the Li-Po battery dedicated to the generation of one $A^2$RID compliant broadcast message.
(a) Circuit diagram.

(b) Lab setup.

Figure 7.3: Circuit diagram of lab setup and actual lab setup to measure the energy consumption of the ESPcopter while using the A²RID protocol.
Figure 7.4: Energy measurements of the singing phase of $A^2$RID in millijoule.

Figure 7.5: An example on how the energy consumption is obtained using the RIGOL DS1202Z-E oscilloscope.
CHAPTER 7. EVALUATION AND RESULTS

7.3 Broadcast Bandwidth

Now that we know the exact content of the broadcast messages, we can also have an idea of the broadcast sizes with respect to different curves. In Figure 7.6, we can see the size of each broadcast with respect to different BLS and BN curves which are the most efficient curves for IoT devices as explained in section 3.2.

![Figure 7.6: Bandwidth requirements using normal and compressed points in the A²RID IND-CPA and IND-CCA2 anonymous broadcasts with the MTU of 1500 highlighted.](image_url)

---

A²RID: An Anonymous RemoteID-compliant Group Signature Scheme for Commercial Drones
Chapter 8

Conclusion

In this report, we have introduced an anonymous RemoteID-compliant group signature scheme for commercial drones. We explained how an UAS can become a member of a group and can send a broadcast message that are either IND-CPA or IND-CCA2-anonymous. A version of A²RID scheme has been implemented on one of the most restricted research/educational platforms, i.e., the ESPcopter. Evaluation has shown that a pre-computed IND-CPA and IND-CCA2 are 6.17 and 4.55 times faster then the requirements of RemoteID. In addition, we have answered the research questions throughout this paper as can been seen in Table 8.1.

Table 8.1: Research sub-questions and their answers.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answered</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>How can this scheme fit the IETF DRIP, as well as the FAA RemoteID requirements?</td>
<td>✓</td>
<td>section 4.1</td>
</tr>
<tr>
<td>How can a scheme be optimised such that the broadcast can be sent within 1 second?</td>
<td>✓</td>
<td>section 7.1</td>
</tr>
<tr>
<td>What effect does this scheme have on memory consumption?</td>
<td>✓</td>
<td>section 7.1</td>
</tr>
<tr>
<td>What effect does this scheme have on energy consumption?</td>
<td>✓</td>
<td>section 7.2</td>
</tr>
<tr>
<td>What effect does this scheme have on the bandwidth of the broadcast?</td>
<td>✓</td>
<td>section 7.3</td>
</tr>
</tbody>
</table>

Further research into A²RID can include lab based energy measurements. This will lead to more insight into the exact energy consumption of A²RID. Also the scope of A²RID can be broadened, at the moment the DRIP privacy requirements and the communication between different DRIP entities are out of scope. With respect to implementation, different cryptographic libraries can be used for A²RID to see the effect on the execution time and memory consumption.
Bibliography


[48] D. Dachman-Soled, “A black-box construction of a CCA2 encryption scheme from a plaintext aware (sPA1) encryption scheme,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8388 LNCS, 2014, pp. 37–55. 15


10.1007/10.11007/10.4419-5906-5.475 27


## Appendix A

### Mathematical Notations

Table A.1: Notation conventions and their description.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>((sk_D, pk_D))</td>
<td>Secret and public keys of PKE</td>
</tr>
<tr>
<td>((sk_R, pk_R))</td>
<td>Secret and public key of SPS-EQ</td>
</tr>
<tr>
<td>(A \parallel B)</td>
<td>(C) of (A) and (B)</td>
</tr>
<tr>
<td>(H())</td>
<td>SHA256 hash function</td>
</tr>
<tr>
<td>(\mathbb{Z}_p)</td>
<td>Integer field modulus (p)</td>
</tr>
<tr>
<td>(\leftarrow)</td>
<td>Uniform random assign</td>
</tr>
<tr>
<td>(e)</td>
<td>paring operation</td>
</tr>
<tr>
<td>([x])</td>
<td>Least integer greater or equal to (x)</td>
</tr>
</tbody>
</table>
Appendix B

BN254 parameters

Table B.1: Parameters of pairing-friendly curve BN254.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>0x0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>$B$</td>
<td>0x0000000000000000000000000000000000000000000000000000000000000002</td>
</tr>
<tr>
<td>$p$</td>
<td>0x2523648240000001ba344d800000007ff9f80000000010a10000000000000</td>
</tr>
<tr>
<td>$\hat{p}$</td>
<td>(0x2523648240000001ba344d8000000086121000000000013a70000000000012,</td>
</tr>
<tr>
<td></td>
<td>0x0000000000000000000000000000000000000000000000000000000000000000)</td>
</tr>
<tr>
<td></td>
<td>([0x061a10bb519eb62feb8d8c7e8c61edeb6a4648bbb4898bf0d91ee4224c803fb2b,</td>
</tr>
<tr>
<td></td>
<td>0x0516aaf9ba737833310aa78c5982aa5b1f4d746bae3784b70d8c34c1e7d54cf3],</td>
</tr>
<tr>
<td></td>
<td>[0x021897a06ba93439a90e096698c822329bd0ae6bdbe09bd19f0e07891cd2b9a,</td>
</tr>
<tr>
<td></td>
<td>0x0ebb2b0e7c8b15268f6d4456f5f38d37b09006fffd739c9578a2d1ae6b3ace9b])</td>
</tr>
</tbody>
</table>