KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering (ESAT)

Lightweight Public Key Cryptography

Jens HERMANS

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering

August 2012

Lightweight Public Key Cryptography

Jens HERMANS

Jury:
Prof. dr. ir. Carlo Vandecasteele, chair
Prof. dr. ir. Bart Preneel, promotor
Dr. ir. Frederik Vercauteren, promotor
Prof. dr. ir. Vincent Rijmen
Prof. dr. Bart Demoen
Prof. dr. Tanja Lange
(Technische Universiteit Eindhoven)
Prof. dr. Serge Vaudenay
(EPFL)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

August 2012

© Katholieke Universiteit Leuven – Faculty of Engineering
Kasteelpark Arenberg 10, bus 2446, 3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2012/7515/90
ISBN 978-94-6018-556-4

Acknowledgements

Vooreerst zou ik mijn promotoren prof. Bart Preneel en dr. Frederik Vercauteren willen bedanken om mij de kans te geven een doctoraat te starten en voor al het advies dat ze mij in de loop van mijn onderzoek gegeven hebben.

I would like to express my gratitude to the members of my jury – prof. Bart Demoen, prof. Tanja Lange, prof. Vincent Rijmen, and prof. Serge Vaudenay – for reviewing this manuscript and for their valuable feedback, and to prof. Carlo Vandecasteele for chairing the jury. Ik zou het Fonds Wetenschappelijk Onderzoek (FWO) – Vlaanderen willen danken voor hun financiële steun die dit onderzoek mogelijk maakte.

Thanks to all my co-authors – among which Michael Schneider, Junfeng Fan, Andreas Pashalidis, Roel Peeters – that I had the privilege of working with. Bedankt Roel, voor de interessante discussies over de vele veilige en onveilige protocollen die we bedachten en voor je (vaak tevergeefse) pogingen om me mee te krijgen naar de Alma. Thank you, Michael Schneider, I really enjoyed working together on lattice enumeration and working week after week on finding the next tweak that would make our code kick ass. During my work on lattice enumeration with Michael Schneider I had the opportunity to visit the TU Darmstadt and the National Taiwan University. I would like to thank CASED and the NTU for funding these visits. Special thanks go out to Chen-Mou Cheng, Bo-Yin Yang and all the people at the Fast Crypto Lab from the National Taiwan University. It was a very enriching experience for me to discover Taipei. Thanks also to the lattice gang and all the others for the nice stay in Darmstadt.

Thanks to all my colleagues and former colleagues at COSIC for all the nice moments at work and for letting me discover life as a researcher. The social events surely helped to make you evolve from colleagues into friends, so a big thanks to all people who helped organizing all these weekends, BBQs and other events.

Nogmaals bedankt Fre, voor de vele uren die we al discussiërend, denkend, maar ook lachend hebben doorgebracht. Bedankt voor je vele geniale ingevingen bij het zoeken naar nieuwe onderzoekspistes en doelwitten. Ik ben blij voor de vele inzichten die je met mij hebt gedeeld en alles wat je mij geleerd hebt. Ook bedankt aan de andere inwoners van de 01.66, het legendarische bewijs dat $\sqrt{1000} \neq 33,33..$ zal me nog lang bijblijven.

Bedankt, Saartje en Svetla, mijn bureaugenoten van het eerste uur, voor de vele aangename momenten in de heksenketel die de 01.65 soms kan zijn. Also thanks to Filipe for offering his chair for inspiring research. Ik zou ook onze supersecretaresse Péla Noë willen bedanken voor de hulp bij de administratie, voor de vele leuke babbels en de gedeelde geheime voorraden in jouw bureau. Wie weet maakt er ook nog iemand een paper over fysieke beveiliging met paperclips. Ik zou ook de helden van de boekhouding – Elvira, Elsy en Wim – willen bedanken voor de hulp, de snelle terugbetaling van alle kosten en het goedkeuren van reparaties. Bedankt Elvira, voor die nieuwe zonnewering. Dat zal ik nooit vergeten...

Dank aan iedereen waarmee ik tijdens en na mijn studententijd heb mogen samenwerken in uiteenlopende organisaties zoals V.T.K., LOKO, ACCO, Student IT, CWlab, de 24-urenloop... Bedankt voor de vele discussies, politiek-filosofisch gemijmer tot in de late uurtjes, hacking van antieke hardware en software... Ook van jullie heb ik veel bijgeleerd over andere zaken dan onderzoek en heb ik veel vrienden overgehouden.

Ik zou mijn ouders willen bedanken voor de kansen die ze me geboden hebben om te studeren. Ook mijn broer Jorn en mijn zus Joni zou ik willen bedanken voor hun enthousiasme en nieuwsgierigheid. Tot slot zou ik mijn aanstaande vrouw Evelien willen bedanken. Het samenleven met een doctoraatsstudent is niet altijd even eenvoudig geweest, maar uiteindelijk zijn we er toch geraakt. Bedankt om me te blijven steunen. Zonder de liefde en steun die jij me gegeven hebt was dit niet gelukt.

Jens Hermans
August 2012

Abstract

The security and privacy risks of lightweight devices have been a growing concern over the past years. Lightweight devices, such as RFID tags, are being used on a large scale in various applications, even if their presence is rarely noticed. Since these devices are so ubiquitous and communication goes unnoticed they can however easily be abused. Information stored on the devices could be compromised, the device could be faked or it could be tracked, compromising the privacy of the product or user.

To solve the above security and privacy issues, cryptographic algorithms and protocols can be used. However, given the constraints on chip area, time, power and energy conventional cryptographic solutions can usually not be applied. An additional problem is that these devices are out in the open, so they can be easily tampered with, revealing the internals of the device. Lightweight cryptography is put forward as a solution to still obtain sufficiently secure cryptography on these devices.

This thesis focuses on several aspects of lightweight public key cryptography. A first question that is put forward is the security of existing lightweight public key primitives. While the computational power for cryptographic attacks is growing, one tries to shrink cryptography to fit on lightweight devices. As a first contribution, we present fast parallel implementations of NTRU encryption and lattice enumeration on GPU. Our implementation of NTRU shows that an extremely high throughput can be achieved even with public key cryptography. This throughput can also be used for the cryptanalysis of NTRU. Our lattice enumeration implementation demonstrates that GPUs can be used for improving the performance of cryptanalysis.

The remainder of the thesis deals with the security and privacy of lightweight protocols for RFID tags. We present new attacks on the security and privacy of several existing protocols. These protocols came without a formal security or privacy proof and were just some of the many protocols that were broken in

the literature.

For the development of our own protocols, we choose an approach using sound protocol design based on provable security. To this end, we analyze several existing RFID privacy models and show poor design choices in several models. Previous proposals also did not allow for strong privacy. We propose a new RFID privacy model that solves these issues and closely models the real world privacy properties a system requires.

Finally we propose new, provably secure and private RFID identification protocols and grouping proofs based on public key cryptography. These protocols achieve the strongest security and privacy properties at a minimal cost compared to other proposals with similar properties.

Beknopte samenvatting

De bezorgheid om de veiligheids- en privacyrisico's van lichtgewicht apparaten is de laatste jaren sterk toegenomen. Lichtgewicht apparaten, zoals RFID tags, worden op grote schaal gebruikt in uiteenlopende toepassingen, ook al merken we hun aanwezigheid zelden op. Aangezien deze apparaten zo alomtegenwoordig zijn en de communicatie ermee onopgemerkt blijft kunnen ze echter eenvoudig misbruikt worden. Informatie opgeslagen op deze apparaten kan gestolen worden, het apparaat kan geïmiteerd worden of het kan getraceerd worden, wat de privacy schendt van het apparaat of de gebruiker ervan.

Cryptografische algoritmes en protocollen kunnen gebruikt worden om de bovenstaande veiligheids- en privacyproblemen op te lossen. Conventionele cryptografische oplossingen kunnen meestal niet gebruikt worden aangezien er beperkingen zijn op de oppervlakte van de chip, rekentijd, vermogen en energie. Een bijkomend probleem is dat deze apparaten vlot toegankelijk zijn, zodat ze gemakkelijk gemanipuleerd kunnen worden om de interne gegevens te extraheren. Lichtgewicht cryptografie wordt daarom voorgesteld als oplossing om toch voldoende veilige cryptografie te realiseren op deze apparaten.

Deze thesis focust op verschillende aspecten van lichtgewicht publieke sleutel cryptografie. Een eerste vraag die naar voren wordt gebracht is de veiligheid van bestaande lichtgewicht publieke sleutel primitieven. Terwijl de rekenkracht voor cryptografische aanvallen toeneemt probeert men de cryptografie in te krimpen om op lichtgewicht apparaten te passen. In een eerste bijdrage stellen we een snelle parallelle implementie voor van NTRU encryptie en roosterenumeratie op een GPU. Onze implementatie van NTRU toont aan dat een extreem hoge doorvoer mogelijk is, zelfs voor publieke sleutel cryptografie. Deze doorvoer kan ook gebruikt worden voor de cryptanalyse van NTRU. Onze implementatie van roosterenumeratie toont aan dat GPU's gebruikt kunnen worden om de performantie van cryptanalyse te verbeteren.

De rest van deze thesis is gewijd aan de veiligheid en privacy van lichtgewicht

protocollen voor RFID tags. We stellen nieuwe aanvallen voor op de veiligheid en privacy van verscheidene bestaande protocollen. Er was geen formeel bewijs van de veiligheid en privacy van deze protocollen. Deze protocollen zijn dan ook slechts enkele van de vele die reeds gebroken zijn in de literatuur.

Voor het ontwikkelen van onze eigen protocollen kozen we voor een benadering gebaseerd op bewijsbare veiligheid. Met dit doel in het achterhoofd analyseerden we verscheidene bestaande RFID privacymodellen en ontdekten we slechte ontwerpkeuzes in verscheidene modellen. Eerdere voorstellen lieten ook geen sterke privacy toe. We stellen zelf een nieuw RFID privacymodel voor dat deze problemen oplost en de werkelijke privacyvereisten voor een systeem dicht benadert.

Tot slot stellen we nieuwe, bewijsbaar veilige en private RFID identificatieprotocollen en groeperingsbewijzen voor gebaseerd op publieke sleutel cryptografie. Deze protocollen hebben de sterkste veiligheids- en privacyeigenschappen tegen een minimale kostprijs vergeleken met eerdere voorstellen met gelijkaardige eigenschappen.

Contents

Acknowledgements i

Abstract iii

Beknopte samenvatting v

Contents vii

List of Figures xv

List of Tables xvii

I General Overview 1

1 Introduction 2

1.1 Motivation 2

1.2 Public Key Cryptography 5

1.2.1 Public Key Encryption 6

1.2.2 Signatures 6

1.2.3 Key Establishment 7

1.3 Lightweight Devices and Applications 7

1.4 Thesis Outline 8

2 Preliminaries **11**

2.1 Lattices 11

2.1.1 Successive Minima 12

2.1.2 Lattice Problems 13

2.1.3 Lattice Basis Reduction 14

2.1.4 Applications 19

2.2 Groups and Discrete Logarithms 20

2.2.1 Groups 20

2.2.2 Discrete Logarithm Problem 21

2.2.3 Diffie-Hellman Problem 23

2.3 Parallel Computing 24

2.3.1 Parallel Programming Models 24

2.3.2 Concurrency Issues 26

2.4 Provable Security and Privacy 26

2.4.1 Security Notions 27

2.4.2 Hardness Assumptions 31

2.4.3 Reductions 31

3 Contributions **35**

3.1 Efficient Implementations 35

3.1.1 NTRU Encryption 35

3.1.2 Lattice Enumeration 36

3.2 RFID Privacy and Protocols 37

3.2.1 RFID Privacy Model 37

3.2.2 Private RFID Protocols 38

3.2.3 Grouping Proofs 39

3.3 Other Publications and Work 39
3.3.1 QR Factorization of Random Circular Matrices 39
3.3.2 Mutual Authentication and Privacy 40

4 Conclusion and Open Problems **41**
4.1 Conclusion 41
4.2 Open Problems and Future Perspectives 42
4.2.1 Security of Lightweight Primitives and Implementations 42
4.2.2 Privacy Models 43
4.2.3 Protocols 44

II Publications **59**

List of Publications **61**

Speed Records for NTRU **63**
1 Introduction 65
2 Related Work 67
3 NTRUEncrypt 68
3.1 Parameter Sets 69
4 GPU Programming 71
4.1 The CUDA Platform 71
5 The Implementation 73
5.1 Operations 73
5.2 Memory Usage - Bit Packing 74
5.3 Encoding 75
5.4 Blocks, Threads and Loop Nesting 75
5.5 Memory Access 76

5.6 Branching 76
6 Results 77
7 Conclusion 79
References 80
A Code Listings 83

Parallel Shortest Lattice Vector Enumeration on Graphics Cards 87

1 Introduction 89
2 Preliminaries 92
2.1 Lattice Basis Reduction 93
2.2 Programming Graphics Cards 95
3 Parallel Enumeration on GPU 97
3.1 Original ENUM Algorithm 97
3.2 Multi-Thread Enumeration 98
3.3 The Iterated Parallel ENUM Algorithm 99
4 Experimental Results 101
5 Further Work 104
References 105

On the Claimed Privacy of EC-RAC III 111

1 Introduction 113
2 The EC-RAC Protocols 114
2.1 EC-RAC I/II and related attacks 114
2.2 EC-RAC III 117
3 Privacy Models 117
4 Attacks on the Protocols 120
4.1 First Attack 120
4.2 Second Attack 121

4.3 Third Attack 122
5 Conclusions 123
References 123

A New RFID Privacy Model **125**
1 Introduction 127
2 Definitions 128
3 Existing Privacy Models 129
3.1 Vaudenay 130
3.2 Canard et al. 134
3.3 Deng, Li, Yung and Zhao 135
3.4 Juels-Weis 137
3.5 Bohli-Pashalidis 138
4 Our Model 139
4.1 Adversarial Model & Privacy 139
4.2 Security, Correctness, Privacy 141
4.3 Motivation and Comparison 142
5 Evaluating Existing Protocols 144
5.1 Vaudenay's Public Key Protocol 144
5.2 RO-based Protocol 146
6 Conclusion 147
References 147
A Extending the Model 150
B Mutual Authentication 152

Wide Strong Private RFID Identification based on Zero-Knowledge **153**
1 Introduction 155
2 Definitions 156

2.1 Privacy Model 157
2.2 Privacy Notions 159
2.3 Private Identification Protocol 159
2.4 Number-theoretical Assumptions 161
3 Previously Proposed Protocols 163
3.1 Zero Knowledge Based Protocols 163
3.2 Public Key Encryption Based Protocols 165
4 A New Protocol 166
4.1 Analysis 169
4.2 Efficiency Optimisation 172
5 Implementation Considerations 173
5.1 Coupons 173
5.2 Comparison 174
6 Conclusions 175
References 175

Private Yoking Proofs: Attacks, Models and New Provable Constructions 181

1 Introduction 183
2 Attacks 184
2.1 First Attack 185
2.2 Second Attack 186
3 Privacy and Security Model 186
3.1 Privacy Model of Hermans et al. 187
3.2 Grouping Proof 188
4 Yoking Proof with Trusted Party 189
4.1 Security and Privacy 191
5 Yoking Proof without Trusted Parties 192

5.1 Security proof 194
6 Conclusion 195
References 195
A Oracles Model Hermans et al. 197
B Privacy Preserving Signatures 198

Curriculum Vitae **201**

List of Figures

I General Overview **1**

II Publications **58**

Speed Records for NTRU **63**

1 NTRU encryption operations per second using ordinary polynomials and the same h ($N = 1171, q = 2048, p = 3$). 79

Parallel Shortest Lattice Vector Enumeration on Graphics Cards **87**

1 Illustration of the algorithm flow. 99

2 Timings for enumeration. 102

On the Claimed Privacy of EC-RAC III **111**

1 ID&Pwd-Transfer protocol from EC-RAC I 115

2 ID-Transfer protocol from EC-RAC II 116

3 ID-transfer protocol (protocol 1) from EC-RAC3 118

4 ID&Pwd-Transfer protocol (protocol 3) from EC-RAC3 119

5 Man-in-the-middle attack on protocols 2 and 3 121

6 Man-in-the-middle attack on protocol 1 122

A New RFID Privacy Model **125**

1 Privacy experiment from Juels-Weis 137

2 Public key RFID protocol from Vaudenay 145

3 RO protocol from Vaudenay 146

Wide Strong Private RFID Identification based on Zero-Knowledge **153**

1 Zero knowledge based protocols. 164

2 Public key encryption based protocols. 167

3 Private RFID identification protocol. 168

4 Optimised private RFID identification protocol. 172

Private Yoking Proofs: Attacks, Models and New Provable Constructions **181**

1 Two-party grouping-proof protocol with colluding tag prevention, proposed by Batina *et al.* 185

2 Two-party grouping-proof protocol with timestamp. 190

3 Two-party grouping-proof protocol without trusted party. . . . 193

List of Tables

I General Overview **1**

1.1 Overview of some lightweight implementations of symmetric and asymmetric algorithms. 9

II Publications **58**

Speed Records for NTRU **63**

1 Performance comparison of NTRU on an Intel Core2 CPU and a Nvidia GTX280 GPU. 85

2 Comparison of several NTRU, RSA and ECC implementations. 86

Parallel Shortest Lattice Vector Enumeration on Graphics Cards **87**

1 Average time needed for enumeration of LLL pre-reduced lattices in each dimension n. 103

2 Average time needed for enumeration of BKZ-20 pre-reduced lattices in each dimension n. 103

On the Claimed Privacy of EC-RAC III **111**

A New RFID Privacy Model **125**

Wide Strong Private RFID Identification based on Zero-Knowledge 153

1 Overview different proposed protocols. 176

Private Yoking Proofs: Attacks, Models and New Provable Constructions 181

Part I

General Overview

Chapter 1

Introduction

1.1 Motivation

Mobile and embedded devices have become ubiquitous in our current society and have created tremendous technical and social challenges. Consider for example the GSM, PDA. eID, Radio Frequency Identification (RFID)... People typically carry around multiple RFID tags (e.g. access cards, passport, product tags) and there are developments towards medical implants with wireless communication, smart clothing, wireless sensor networks... These devices are hardly ever switched off, if it's even possible to do so, and they are very closely connected to a specific person.

In general we can describe an RFID tag as a small, wireless device without any user interface. There are many types of RFID tags. Passive RFID tags don't have any power source and are activated by the field of the reader. Passive RFID tags are inexpensive and the most commonly used devices. Active devices have an embedded battery and can initiate communication themselves.

The initial purpose of RFID tags was identification of products. RFID is considered an alternative for barcodes, with the advantage of having unique identifiers for individual items instead of a single identifier for all identical products. An additional advantage is that no line of sight is required for reading the tags, like for barcodes. The use of RFID tags for product identification has been standardized in the Electronic Product Code (EPC) standard [32]. The functionality of EPC tags is limited to providing a unique identifier.

Besides being used as a replacement for barcodes, there are plenty of other areas

where RFID is widely deployed. Consider for example access cards, wireless car keys [60], payment cards [83] or public transport cards [118]. There are plenty of future applications still waiting to be deployed. Tags could be used for anti-counterfeiting protection, automatic checkout for shopping, smart home appliances, medical implants, bank notes ...

One of the major issues with RFID tags is that they can be activated remotely without the user ever knowing it. A problem that this could lead to is the compromise of information stored on the tag or tracking of the user. For example, consider modern electronic passports that have an RFID chip with all personal data embedded in it. It is clear that this personal information should not be publicly available.

At first sight simple product identification tags used in retail, that only contain a unique number, seem harmless when it comes to data compromise. However, based on the product identifiers information can be derived on what kind of items a user carries around or uses, like medication, bank notes or politically sensitive library books. Even if no information is known about the meaning of the identifiers, a profile of a user can be created to track him around.

The cases described above are reason for concern. Producers of RFID tags have done some effort to try and protect the tags. Tags in retail are usually 'killed' when the product goes over the counter: there is an RFID reader at the counter that sends a special command to the tag which disables it permanently. This is however no option for tags with different uses. To solve this, for example, for passports, additional protection was added to access the card: to get access one needs to read out part of the text printed inside the passport and use this to authenticate to the passport and secure the communication [62, 64].

Despite the efforts by some manufacturers and protocol designers, several of the security and privacy issues remain or will reappear when RFID tags will get additional functionality. The RFID world is also plagued by poorly designed security measures, which tend to be broken quite easily [123].

Cryptography. Cryptography can be used to provide data confidentiality, data integrity, authentication and more complex security notions. Implementing cryptography on an RFID tag however comes with a cost: it consumes chip area, power, energy and time, which are very scarcely available on an RFID tag. This implies that standard cryptographic solutions cannot be deployed on lightweight devices like RFID tags.

Thus, the rise of lightweight devices has created new challenges in cryptography. *Lightweight cryptography* has been proposed to answer the demand for cryptography that uses minimal resources. Not only the cryptographic

functions that are implemented on the chip should be minimized in terms of area, power or energy while at the same time preserving an acceptable performance. Also the communication protocols used between parties should be optimized. By choosing different protocol types, some expensive cryptographic functions can be avoided. Since cryptography also comes with a communication overhead, which in case of RFID tags can consume much energy and cause additional delay, this is an additional constraint that needs to be taken into account.

It is clear that lightweight cryptography tries to minimize the impact of security and privacy protection on the performance and cost of devices. On the other hand, there is the risk of breaking the security by using minimalist cryptographic primitives and protocols. Most cryptographic schemes thrive on the fact that it takes enormous amounts of computation time to break a scheme. While attempting to push cryptography on smaller devices, computers are getting stronger every day and so are the attacks on cryptographic schemes. Finding the right trade-off between security and cost is therefore important.

Symmetric versus Public Key Cryptography. A distinction is made between *symmetric* and *public key* cryptography. In symmetric key cryptography a secret value or key is shared by two or more parties. Both parties can perform the same operations, like encrypting/decrypting, signing/verifying. In public key cryptography each party has a public and a private key. The private key is only known to its owner, while the public key can be made available to everyone. This way all operations become asymmetric: instead of both parties being able to compute the encryption/decryption, only the owner of the private key can decrypt or sign.

One of the main advantages attributed to symmetric key cryptography is that it is cheaper to implement than public key cryptography. As discussed further on in Section 1.3, this seems to hold for block ciphers, but for more recent hash functions from the SHA-3 competition, the difference is smaller. Sharing the same key is manageable when a limited number of parties are involved. For growing numbers one either has to share the same key across all parties or resolve to separate keys and key management. The first approach has the advantage of being easy to manage, but on the other hand can be a big security risk when one of the parties is compromised. The second approach implies key management, which becomes involved as the number of parties grows. For public key cryptography the problem is restricted to managing the public keys, the private keys are only stored by the owner. A new issue arises when considering the problem of linking a public key to the correct identity. There exist several solutions, ranging from verification or certification mechanisms to

storing a list of accepted public keys and, optionally, matching identities.

In Sect. 1.2 we introduce several public key operations in more detail.

Tampering. An additional difficulty for mobile devices is that, by definition, they are physically accessible and can be tampered with. This issue needs to be taken into account when designing cryptographic algorithms. An adversary could obtain secret values stored in certain devices under his control and use these to attack the system as a whole.

There are several ways for extracting secret data from tags. A relatively unobtrusive way would be side-channel attacks. Lightweight devices consume power, emit radiation, produce heat, have timing differences for computations... By precisely measuring these values one could obtain information on secret values processed by the chip [41, 69, 70, 104]. One can also introduce errors, by twiddling with the power supply of the chip or the clock frequency in the hope to derive information on the secrets [10, 17]. A more intrusive way could be to open the chip and use probes, lasers... to physically read out or modify data stored in the chip. There exists a wide range of side-channel attacks and countermeasures for these. Since these attacks are mostly on the hardware level we will take a more high-level approach to side-channels and tampering.

Research Goals. This thesis tries to provide answers to the above challenges. We focus on the following research questions:

- How secure are existing lightweight public key cryptographic functions?
- Can lightweight public key cryptographic functions be efficiently implemented in software and how does this impact the security? Can we exploit the growing computation power to break schemes?
- How to develop and analyze lightweight public key protocols with strong security and privacy properties?
- Which theoretical models should be used to evaluate the security and privacy of protocols for lightweight devices, taking into account the risk of tampering?

1.2 Public Key Cryptography

In this section we give a brief overview of the most important functions in public key cryptography. We will discuss encryption, signatures and key establishment.

There exist many other functions (e.g. identity based encryption, homomorphic encryption, oblivious transfer, anonymous authentication and signatures...), but these are not used in this work.

1.2.1 Public Key Encryption

Encryption is used to protect the confidentiality of data. In public key encryption, the sender encrypts the message with the recipient's public key. Public key encryption guarantees that only the recipient who has the matching private key can recover the original message.

A public key encryption scheme consists of three algorithms: `KeyGen`, `Encrypt` (or $\mathcal{E}$), `Decrypt` (or $\mathcal{D}$). The `KeyGen`(k) algorithm generates a public key pk and a secret key sk, based on the security parameter k. All algorithms are probabilistic polynomial time (in the security parameter) Turing machines.

The encryption algorithm $c \leftarrow \mathcal{E}(pk, m)$ takes a message $m \in M$ and outputs the ciphertext $c \in C$. M is the message space and C the ciphertext space. The decryption algorithm $m' \leftarrow \mathcal{D}(sk, c)$ maps the ciphertext c' onto the message m'. The function $\mathcal{E}$ is an injective mapping and $\mathcal{D}$ is surjective. For all key pairs (pk, sk) and all messages $m \in M$ it has to hold that $Dec(sk, Enc(pk, m)) = m$. Note that not all possible ciphertexts $c \in C$ have to be valid (i.e. originate from the encryption of a message). In these cases $\mathcal{D}$ fails and returns $\perp$.

Wellknown encryption schemes include RSA [109], ElGamal [40], Cramer-Shoup [27], NTRUEncrypt [57, 119] and McEliece [86].

1.2.2 Signatures

A signature scheme protects the authenticity of a message. The sender of a message can use its private key to compute a digital signature for the message. Any recipient can use the public key to verify if the message truly originated from the sender, by checking the signature. The symmetric key equivalent of signatures is a Message Authentication Code (MAC). Since the key is shared by both parties, both can sign and validate.

A (public key) signature scheme consists of three algorithms: `KeyGen`, `Sign`, `Verify`. The key generator functions similarly to public key encryption.

The signing algorithm $\sigma \leftarrow$ `Sign`(sk, m) takes a message $m \in M$ and outputs a signature $\sigma \in S$. The verification algorithm $b \leftarrow$ `Verify`(pk, m, σ) takes a message m and a signature σ and outputs a bit b, which is true if the signature

is valid. In some cases the underlying signature scheme supports *message recovery*. In that case the `Verify` algorithm does not need the message m as input, but instead returns m.

Wellknown signature schemes are the Digital Signature Algorithm (DSA) [95] including the elliptic curve variant and RSA signatures [110]. Special signature scheme variants are dedicated verifier signatures, where only a designated party can verify a signature, and blind signatures, where the message that is signed is hidden from the signer.

1.2.3 Key Establishment

When two parties, wishing to communicate with each other have no shared key, key exchange or key establishment allows them to generate a shared, secret key. The Diffie-Hellman key establishment protocol [28] and Merkle's puzzles [88] were the first protocols to achieve this. In Diffie-Hellman key establishment, party A generates some random x and outputs g^x, where g is the generator of some pre-agreed group in which the discrete logarithm problem is believed to be hard. Next, party B generates in the same way g^y. The shared secret key is $g^{xy} = (g^x)^y = (g^y)^x$ and can only be computed by the two parties involved. Key establishment protocols do not give any guarantees on the identities of the parties involved, so the protocols need to be extended with extra protection to check the identities.

1.3 Lightweight Devices and Applications

To get a better understanding of the typical constraints that lightweight devices have, it is important to evaluate the cost of implementing typical cryptographic primitives. Note that we do not aim for completeness in this section, comparing all existing implementations would constitute a separate work and is out of scope.

There are several criteria for evaluating the 'cost' of cryptography: time (clock cycles or latency), power, energy and area. Comparison between several algorithms and even implementations of the same algorithm is difficult because of the several trade-offs one can make. For example, at the cost of additional area, one can obtain faster implementations; increasing the clock allows faster computation, but also increases power consumption. The same holds for the security level and the metrics for implementation cost.

Table 1.1 shows a comparison between several block ciphers (AES-128), hash functions (SHA-256, SHA-3 finalists, SPONGENT-160) and elliptic curve scalar multiplication implementations (over $\mathbb{F}_{2^{163}}$). Obviously the functionality of a block cipher, hash function and an elliptic curve scalar multiplication is completely different. The reason for this comparison is to get a rough idea of whether they can be implemented and at what cost. It is also important to consider the usage of these primitives: elliptic curves allow for public key cryptography, which can be used to provide stronger security and privacy notions than symmetric cryptography such as block ciphers or hash functions.

From Table 1.1 it is clear that block ciphers are by far the cheapest, closely followed by the lightweight hash function SPONGENT (and presumably many other lightweight hash functions with similar performance). The SHA-3 finalists come out quite heavy in the comparison, mainly due to the large state that needs to be stored in memory and also due to the higher security parameters. A full comparison of SHA-3 implementations is available in the ECRYPT SHA-3 Zoo [33]. Note that SPONGENT has a very low throughput compared to these. The ECC implementations are the most expensive in terms of area, speed and power.

When desiging protocols for lightweight devices one has to keep in mind the costs of the underlying algorithms described above. For this reason it might be reasonable to trade security for a smaller implementation cost, by only using symmetric primitives or choosing smaller key sizes. As an alternative one can try to avoid the combination of several (a)symmetric primitives in the same protocol, to avoid having multiple functions on the same chip.

1.4 Thesis Outline

In Part I of this thesis an overview is provided of the publications that are presented in Part II. Chapter 1 puts forward the research goals and questions and situates the research. In Chapter 2 the necessary preliminaries are presented to understand this thesis. Readers familiar with cryptography and concepts like lattices, lattice reduction, provable security and elliptic curves can skip these sections. Chapter 3 summarizes our contributions to the field. Finally, Chapter 4 concludes the thesis and puts forward several new, open research questions based on the contributions we have made to the field.

Table 1.1: Overview of some implementations of symmetric and asymmetric algorithms. Memory and other components (such as a random number generators) are not included in the cost. The hash functions and ECC processors include the memory required for the computations (i.e. registers).

	Area (GE)	Clock (kHz)	Power (μW)	Energy (μJ)	Time (ms)	Throughput (kbps)	Technology
AES-128 [36]	3 400	80 000	4.5	-	-	9900	350 nm
AES-128 [93]	2 400	100	3.7	-	-	56.6	180 nm
SHA-256 [68]	11 300	100 000	1770	-	-	878000	90 nm
SHA-3 [a] [68]	$>$ 22562	100 000	$>$ 2070	-	-	$>$ 1347 $\cdot 10^3$	90 nm
SHA-3 [b] [121]	12 890	80 000	-	-	-	19 800	350 nm
SPONGENT-160 [16]	1 296	100	-	-	-	0.4	45 nm
SPONGENT-160 [16]	2 272	100	-	-	-	17.78	45 nm
ECC [74]	14 566	700	13.8	1.18	85	-	130 nm
ECC [126]	8 958	1000	32.1	9.43	294	-	130 nm
ECC [15]	12 536	847	79	7.5	95	-	220 nm

[a] High throughput implementations of finalists. The numbers are minimal values over all ciphers respectively. Note that minimal values for every metric are not necessarily from the same cipher.

[b] Results are for Skein-256-256, which has the smallest area for a full implementation according to [33].

Chapter 2

Preliminaries

In this chapter we introduce some preliminaries necessary for understanding the publications in Part II and which are not introduced in the publications. The individual papers contain more specific preliminaries, which can be read after the general preliminaries below. We focus mainly on lattices, groups, parallel computing and provable security.

2.1 Lattices

Definition 2.1 (Lattice)**.** Given n linear independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^d$ (or, equivalently, a matrix $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$), the lattice $\mathcal{L}(\mathbf{B})$ is defined as

$$\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i | x_i \in \mathbb{Z}\}\,. \tag{2.1}$$

A lattice is a discrete, additive subgroup of $\mathbb{R}^d$. The dimension n of a lattice is the number of linear independent vectors in the lattice, i.e. the number of basis vectors. When $n = d$ the lattice is called full dimensional. The span of a lattice is the linear space spanned by its basis vectors.

The basis of a lattice is not unique. Every unimodular transformation $\mathbf{M}$, i.e. integer transformation with $\det \mathbf{M} = \pm 1$, turns a basis matrix $\mathbf{B}$ into a second basis $\mathbf{MB}$ of the same lattice.

Definition 2.2 (Fundamental parallelepiped)**.** The fundamental parallelpiped of a lattice basis $\mathbf{B}$ is defined as

$$\mathcal{P}(\mathbf{B}) = \{\mathbf{Bx} | \mathbf{x} \in \mathbb{R}^n, \forall i : 0 \leq x_i < 1\}. \tag{2.2}$$

The determinant of a lattice is the volume of the fundamental parallelepiped, and can be computed as $\det(\mathcal{L}(\mathbf{B})) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$. For full dimensional lattices we have $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$. The determinant of a lattice is invariant under the choice of the lattice basis, which follows from the multiplicative property of the determinant and the fact that basis transformations have determinant ± 1.

The Gram-Schmidt algorithm computes an orthogonalization of a basis. It sequentially computes

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*, \text{with } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \tag{2.3}$$

starting from $i = 1$ up to n.

It produces an orthogonal matrix $\mathbf{B}^*$, which is not necessarily a basis of the lattice. The Gram-Schmidt orthogonalization is used frequently in lattice algorithms and can also be used to compute $\det(\mathcal{L}(\mathbf{B})) = \prod_i \|\mathbf{b}_i^*\|$.

The Gram-Schmidt orthogonalization is related to the QR-decomposition $\mathbf{B} = \mathbf{Q}\cdot\mathbf{R}$ with $\mathbf{Q}$ an orthonormal matrix and $\mathbf{R}$ upper-triangular. The matrix $\mathbf{Q}$ can be obtained trivially from the Gram-Schmidt orthogonalization by normalizing the columns of $\mathbf{B}^*$.

2.1.1 Successive Minima

In cryptographic applications of lattices one is often interested in short lattice vectors. In this thesis, lengths of vectors are always considered using the Euclidean norm of the vector: $\|\mathbf{b}\| = \sqrt{\sum_i b_i^2}$. The length of a shortest vector of a lattice $\mathcal{L}(\mathbf{B})$ is denoted $\lambda_1(\mathcal{L}(\mathbf{B}))$. The successive minima (i.e. second and following shortest vectors, linearly independent from the previous minima) are denoted as $\lambda_i(\mathcal{L}(\mathbf{B}))$ for $i = 1, \ldots$.

There exist several bounds and estimations for the length of the shortest lattice vector.

Theorem 2.1 (Minkowski's Convex Body Theorem)**.** *Let $\mathcal{L}(\mathbf{B})$ be a full dimensional lattice ($n = d$). For any centrally-symmetric convex body S, if $vol(S) > 2^n \det \mathcal{L}\mathbf{B}$, then S contains a nonzero lattice point.*

By applying the theorem to an n-dimensional ball, Minkowski's first theorem is obtained:

Theorem 2.2 (Minkowski's First Theorem)**.** *Let $\mathcal{L}(\mathbf{B})$ be a full dimensional lattice ($n = d$), then $\lambda_1(\mathcal{L}(\mathbf{B})) \leq \sqrt{n} \det \mathcal{L}(\mathbf{B})^{1/n}$.*

For a full dimensional random lattice, the Gaussian heuristic [58] states that

$$\lambda_1(\mathcal{L}(\mathbf{B})) \approx \sqrt{\frac{n}{2\pi e}} \det(\mathcal{L}(\mathbf{B}))^{1/n}. \tag{2.4}$$

Note that the Gaussian heuristic only gives an estimation of the length of the shortest vector of a lattice.

2.1.2 Lattice Problems

Several (hard) problems can be defined over lattices, given a specific basis $\mathbf{B}$. One of the most common problems is the Shortest Vector Problem (SVP):

Definition 2.3 (Search SVP)**.** Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{d \times n}$, find $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L}(\mathbf{B}))$.

There also exists a Decisional SVP, where one has to determine if $\|\lambda_1(\mathcal{L}(\mathbf{B}))\| < r$, for a given $r \in \mathbb{R}$; and an Optimization SVP, where one only has to determine $\lambda_1(\mathcal{L}(\mathbf{B}))$. It can be shown that these three variants are equivalent.

The Search SVP is $\mathcal{NP}$-hard (at least under randomized reductions) [29,67,108], so often the approximate version γ-SVP is considered:

Definition 2.4 (Search γ-SVP)**.** Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{d \times n}$ and $\gamma \geq 1$, find $\mathbf{v} \in \mathcal{L}\mathbf{B}$ such that $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$.

Another related problem is the Closest Vector Problem (CVP) and its approximation variant γ-CVP, where one searches for a lattice point closest to a given target vector $\mathbf{t} \in \mathbb{Z}^d$:

Definition 2.5 (Search γ-CVP)**.** Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{d \times n}$, a target vector $\mathbf{t} \in \mathbb{Z}^d$ and $\gamma \geq 1$, find $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\forall \mathbf{x} \in \mathcal{L}(\mathbf{B}), \|\mathbf{t} - \mathbf{v}\| \leq \gamma \cdot \|\mathbf{t} - \mathbf{x}\|$.

There exist several other lattice problems, like Bounded Distance Decoding and the Shortest Independent Vector Problem. For a more detailed description we refer to Micciancio and Goldwasser [90].

2.1.3 Lattice Basis Reduction

Some lattice bases are more useful when trying to solve the problems from Sect. 2.1.2. The goal of lattice basis reduction is to find a basis consisting of short and almost orthogonal lattice vectors.

LLL Reduction. Gauss presented the first lattice reduction algorithm in 1801 for lattices of dimension $n = 2$. In 1982 Lenstra, Lenstra, and Lovász [77] introduced the LLL algorithm, which was the first polynomial time algorithm to solve the approximate shortest vector problem in higher dimensions. The algorithm approximates the shortest vector to within a bound exponential in the lattice dimension. The LLL algorithm transforms a given basis $\mathbf{B}$ in an LLL-reduced basis.

Definition 2.6 (LLL reduced basis)**.** Take a basis $\mathbf{B}$ and its Gram-Schmidt orthogonalization $\mathbf{B}^*$. The basis $\mathbf{B}$ is δ-LLL reduced if

1. $\|\mu_{i,j}\| \leq \frac{1}{2}$, $\forall 1 \leq j < i \leq n$, and
2. $\delta\|\mathbf{b}_i^*\|^2 \leq \|\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\|^2$.

Algorithm 1 presents the basic version of the LLL algorithm. In the first part of the algorithm, the basis is size reduced, i.e. the length of several basis vectors is reduced, while preserving the lattice. This ensures the first property for an LLL reduced basis, i.e. $\|\mu_{i,j}\| \leq \frac{1}{2}$. The second part of the algorithm swaps basis vectors and ensures the second LLL property. The whole algorithm iterates until the basis satisfies the requirements for an LLL basis. It can be shown that the algorithm terminates in time polynomial in the dimension n and the length of the bit representation of $\mathbf{B}$. The algorithm presented here uses rational numbers. For performance floating point numbers are recommended, but the algorithm needs to be modified to avoid numerical unstability [113].

Several LLL variants were presented by Schnorr [112], Nguyen and Stehlé [96], and Gama and Nguyen [38]. Primal-dual reduction [71] uses the dual of a lattice for reducing. Random sampling [18,112] combines LLL-like algorithms with an exhaustive point search in a set of lattice vectors that is likely to contain short vectors.

BKZ Reduction and Enumeration. Another algorithm is the BKZ block algorithm of Schnorr and Euchner [113]. Let $\pi_i : \mathbb{R}^n \to span(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp$ denote the orthogonal projection such that $\mathbf{b} - \pi_i(\mathbf{b}) \in span(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$. Let L_i denote the lattice $\pi_i(\mathcal{L}(\mathbf{B}))$.

Algorithm 1: The LLL Algorithm

Input: Lattice basis $\mathbf{B}$
Output: Reduced basis $\mathbf{B}$

```
Compute B* and [μ_{i,j}]
for i = 2 to n do
    for j = i − 1 to 1 do
        b_i ← b_i − ⌈⟨b_i, b*_j⟩/⟨b*_j, b*_j⟩⌋ · b_j
    end for
end for
if δ‖b*_i‖² > ‖μ_{i+1,i} b*_i + b*_{i+1}‖² then
    b_i ↔ b_j
    Go back to 1.
end if
```

Definition 2.7 (Korkine-Zolotarev Reduced)**.** Take a basis $\mathbf{B}$ and its Gram-Schmidt orthogonalization $\mathbf{B}^*$. The basis $\mathbf{B}$ is Korkine-Zolotarev reduced if it is size-reduced and $\|\mathbf{b}_i^*\|^2 = \lambda_1(L_i)$.

It can be shown [66] that every Korkine-Zolotarev basis satisfies

$$\frac{4}{i+3} \leq \frac{\|\mathbf{b}_i\|^2}{\lambda_i^2} \leq \frac{i+3}{4}\,. \tag{2.5}$$

For $i = 1$ this guarantees that $\mathbf{b}_1$ will be the shortest vector. In practice however, the complexity of the best algorithm is rather high; $(\sqrt{n})^{n+o(n)} + O(n^4 \log B)$ with $B = \max(\|\mathbf{b}_1\|^2, \ldots, \|\mathbf{b}_n\|^2)$.

To relax the Korkine-Zolotarev conditions, one applies the definition only to certain blocks of the basis vectors:

Definition 2.8 (Block Korkine-Zolotarev reduced)**.** Given a basis $\mathbf{B}$, its Gram-Schmidt orthogonalization $\mathbf{B}^*$ and an integer $\beta \in [2, m-1]$. The basis $\mathbf{B}$ is β-block Korkine-Zolotarev reduced if it is size-reduced and

$$\delta\|\mathbf{b}_i^*\| \leq \lambda_1(L_i(\mathbf{b}_1, \ldots, \mathbf{b}_{\min(i+\beta-1,n)}))\,. \tag{2.6}$$

After LLL reducing the full basis $\mathbf{B}$, the BKZ algorithm reduces each block $\mathbf{b}_i, \ldots, \mathbf{b}_{\min(i+\beta-1,n)}$ of size β for $i = 1 \ldots n$ to ensure that the first vector of each block satisfies the BKZ requirements and is the shortest vector in the projected lattice. In Algorithm 2, ENUM computes the minimum of the function

$$f(x_j, \ldots, x_k) = \sum_{s=j}^{k} \left(\sum_{i=s}^{k} x_i \mu_{i,s} \|\mathbf{b}_s^*\| \right)^2 . \tag{2.7}$$

Algorithm 2: The BKZ Algorithm (Sketch)

Input: Lattice basis $\mathbf{B}$, $0 < \delta < 1$
Output: BKZ-Reduced basis $\mathbf{B}$

LLL-reduce $\mathbf{B}, \delta$
$z, j \leftarrow 0$
while $z < m - 1$ **do**
  $j \leftarrow j + 1, k \leftarrow \min(j + \beta - 1, n)$
  if $j = n$ **then**
    $j \leftarrow 1, k \leftarrow n$
  end if
  $(x_j, \ldots, x_k) \leftarrow \text{ENUM}(\mathbf{B}, j, k)$
  $h \leftarrow \min(k + 1, n)$
  if $(x_j, \ldots, x_k) \neq (1, 0, \ldots, 0)$ **then**
    LLL-reduce $[\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}, \mathbf{b}_{j,new}, \mathbf{b}_j, \ldots, \mathbf{b}_h], \delta$
    $z \leftarrow 0$
  else
    $z \leftarrow z + 1$
    LLL-reduce $[\mathbf{b}_1, \ldots, \mathbf{b}_h], \delta$
  end if
end while

The ENUM algorithm essentialy applies the projection π_j to the sublattice $(\mathbf{b}_j, \ldots, \mathbf{b}_k)$ and finds the vector $\mathbf{b}_{j,new} = \sum_{i=j}^{k} \mathbf{b}_i x_i$ such that $\|\pi_j(\mathbf{b}_{j,new})\| = \lambda_1(\pi_j(\mathbf{b}_j, \ldots, \mathbf{b}_k))$.

In practice, this is the algorithm that gives the best solution to lattice reduction so far although it cannot be shown that the algorithm finishes in polynomial time. The enumeration algorithm (ENUM) is a variant of the Fincke-Pohst [37] and Kannan [65] algorithms. The ENUM algorithm is the fastest algorithm in practice to solve the exact shortest vector problem using complete enumeration of all lattice vectors in a suitable search space. The enumeration algorithm considers a search tree, with each level representing a coordinate x_i.

The ENUM algorithm performs an exhaustive search for all coordinate vectors $\mathbf{x} \in \mathbb{Z}^{k-j+1}$ to search for the shortest (projection) vector $\mathbf{b}_j^{new} = \sum_{i=j}^{k} \mathbf{b}_i x_i$. For simplicity we ignore the projection π_j, so we can assume we are enumerating a full lattice, i.e. $j = 1$ and $k = n$. The enumeration algorithm organizes these linear combinations of the basis vectors in a search tree. Let $i = 1$ be the bottom of the tree and $i = n$ the top. The value i in the algorithm indicates the current level in the tree and A indicates the squared length of the shortest vector found so far.

Algorithm 3: The ENUM Algorithm

Input: Lattice basis $\mathbf{B}$, $[\mu_{i,j}]$, boundaries $1 \leq j < k \leq n$
Output: Coordinates $\hat{\mathbf{x}} = (\hat{x}_j, \ldots, \hat{x}_k)$ of the shortest vector

$r_i \leftarrow \|\mathbf{b}_i^*\|^2$, $A \leftarrow r_j$
$x_j, \delta_j, \Delta_j \leftarrow 1$; $c_j \leftarrow 0$
$x_i, \Delta_i, c_i, y_i \leftarrow 0; \delta_i \leftarrow -1 \quad \forall i \in [j+1, k+1]$
$i \leftarrow j$
while $i \leq k$ **do**
    $l_i = l_{i+1} + (x_i - c_i)^2 r_i$
    if $l_i < A$ **then**
        if $i > j$ **then**
            $i \leftarrow i - 1$
            $c_i \leftarrow -\sum_{t=i+1}^{k} x_t \mu_{t,i}$
            $x_i \leftarrow \lceil c_i \rfloor$
            $\Delta_i \leftarrow 0$
            if $x_i > c_i$ **then**
                $\delta_i \leftarrow -1$
            else
                $\delta_i \leftarrow 1$
            end if
        else
            $A \leftarrow l_i$, $\hat{x}_t \leftarrow x_t \quad \forall t \in [j, k]$
        end if
    else
        $i \leftarrow i + 1$
        $\delta_i \leftarrow -\delta_i$, $\Delta_i \leftarrow -\Delta_i + \delta_i$
        $x_i \leftarrow x_i + \Delta_i$
    end if
end while

The ENUM algorithm can be considered a specialized branch-and-bound algorithm [73]. In a branch-and-bound algorithm the minimization (maximization) problem is divided in subsets (the branching phase) for which a lower (upper) bound is computed (the bounding phase) to select branches with feasible solutions. Unlike many branch-and-bound algorithms, the branching is implicit in the ENUM algorithm due to the way the search tree is traversed. The branching is lazy, i.e. the algorithm does not compute a list of branches which would be hard since the range of the coordinates is infinite. Only one position in the tree is active at any given point in time, which reduces storage requirements. The selection of the next position is done depth-first and the enumeration starts at a leaf of the tree. The bounding phase involves the computation of a lower

bound for the norm of the shortest vector of a subtree and comparing this to the best-known shortest vector.

We define $l_i = l_{i+1} + (x_i - c_i)^2 r_i$, which is an 'intermediate' norm at level i for the vector with coordinates $x_i, \ldots, x_n$. It holds that $l_t \geq l_i$ for all coordinate vectors $\mathbf{x}$ ending in the same $x_i, \ldots, x_k$ and $t < i$. Note that $l_1 = \| \sum_{i=1}^{k} \mathbf{b}_i x_i \|^2$. The algorithm starts at the bottom of the tree, with the coordinate vector $\mathbf{x} = (1, 0, \ldots, 0)$. At every level it checks if $l_i < A$. If so, it goes one level lower into the tree. Otherwise, it goes one level up and the current branch is abandoned.

The value of x_i always belongs to an interval of length $\sqrt{\frac{A - l_{i+1}}{r_i}}$ centered at c_i. When moving down the tree, the first value for x_i is $\lceil c_i \rfloor$, with $c_i = -\sum_{t=i+1}^{k} x_t \mu_{t,i}$, which is the center of the interval. When the algorithm reaches level i again when moving upwards, it will move to the next value of x_i using a zig-zag pattern around the center c_i. The variables $\delta_i \in \{-1, 1\}$ and Δ_i are used to generate this zig-zag pattern.

If an estimate of the length of the shortest vector is known a priori, A can be lowered at the start of the algorithm. This will cut off many branches and thus improve the runtime of the algorithm. The papers [113, 114] present a probabilistic improvement of ENUM, called tree pruning. The idea is to prune subtrees that are unlikely to contain shorter vectors. In extreme pruning [39] the lattice basis that is input to the enumeration algorithm is randomized. Throughout the enumeration very tight pruning is applied. This obviously increases the risk of missing the shorest vector. However, it turns out that by applying the enumeration algorithm to multiple randomized bases of the same lattice the overal performance increases, despite the increase in failure probability for individual enumerations.

In [103] Pujol and Stehlé analyze the stability of the enumeration when using floating point arithmetic. In [47], improved complexity bounds for Kannan's algorithm are presented. This paper also suggests some better preprocessing of lattice bases, i.e., the authors suggest to BKZ reduce a basis before running enumeration. This approach lowers the runtime of enumeration.

In [38] Gama and Nguyen compare the `NTL` implementation [116] of floating point LLL, the deep insertion variant of LLL and the BKZ algorithm. It is the first comprehensive comparison of lattice basis reduction algorithms and helps understanding their practical behavior. Later work [23] provides estimates of the running time of lattice reduction in higher dimension, which allows to assess the impact of lattice reduction on the security of cryptographic schemes.

Sieving. The papers [97] and [91] present improved sieving variants, where the Gauss-sieving algorithm of [91] is shown to be really competitive to enumeration algorithms in practically interesting dimensions.

2.1.4 Applications

Lattices have many applications in cryptography. The foundation of some cryptographic primitives is based on the hardness of lattice problems.

The Learning With Errors (LWE) problem was introduced by Regev [107]. Solving LWE implies solving SVP and SIVP using a quantum reduction.

Definition 2.9 (Search LWE)**.** Given samples $\mathbf{a_i}, b_i$ with

$$b_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i \mod p\,, \tag{2.8}$$

where $\mathbf{s}, \mathbf{a}_i$ are chosen independently and uniformly from $\mathbb{Z}_p^n$, and errors $e_i \in \mathbb{Z}_p$ chosen from a distribution χ; recover $\mathbf{s}$.

The Small Integer Solution (SIS) is similar to LWE, but requires finding a *small* $\mathbf{s}$ such that $\mathbf{As} = 0 \mod p$. Note that SIS does not involve any error vector.

Lattice reduction helps determining the practical hardness of those problems and can thus be used to determine the security of real world applications of those hash functions, signatures, and encryption schemes. There exist several so-called 'worst-case to average-case' reductions [5, 19, 89, 106] that show that breaking certain cryptographic primitives (in the average case) is hard assuming that solving certain lattice problems is hard (in the worst case). This type of reduction is especially interesting since only information on the worst case hardness of problems is usually known, i.e. there exist very easy instances of a hard problem.

Well known lattice based cryptographic primitives are the SWIFFT hash functions [81], the signature schemes of [22,43,79,80], or the encryption schemes of [6, 101, 120]. The NTRU [1, 57] and GGH [45] schemes do not provide a security proof, but the best attacks are also lattice based. To improve the efficiency ideal lattices can be used. These lattices have an additional structure overlayed on top of their group structure. When interpreting a lattice vector as a polynomial vector, all these polynomials are elements of an ideal. Many lattice schemes can be easily transformed to support ideal lattices. Recently a provably secure variant of NTRU based on ideal lattices was proposed by Stehlé and Steinfeld [119].

In cryptanalysis, there are further applications of lattice basis reduction. Not only lattice-based systems can be broken using this technique. There are also

attacks on RSA and similar systems, using lattice reduction to find small roots of polynomials [25, 26, 31, 85]. Low density knapsack cryptosystems were successfully attacked with lattice reduction [72]. Other applications of lattice basis reduction are factoring integers and computing discrete logarithms using diophantine approximations [111]. In discrete optimization, lattice reduction can be used to solve linear integer programs [78].

2.2 Groups and Discrete Logarithms

2.2.1 Groups

Definition 2.10 (Group)**.** A (multiplicative) group G is a set such that:

- $\forall a, b \in G$, $ab \in G$ (closed)
- $\forall a, b, c \in G$, $a(bc) = (ab)c$ (associative)
- $\exists 1 \in G$ such that $\forall a \in G$, $a\,1 = 1\,a = a$ (identity element)
- $\forall a \in G$, $\exists a^{-1} \in G$ such that $aa^{-1} = a^{-1}a = 1$ (invertible).

We use $|G|$ to denote the number of elements in G, which is also called the *order* of the group. A group is called cyclic if there exists a $g \in G$ (called generator) such that for every $a \in G$ there is an $x \in \mathbb{Z}$ such that $g^x = a$.

For finite groups, by Lagrange's theorem, the order of a subgroup H of G always divides $|G|$. For a finite cyclic group the existence of exactly one subgroup of order d can be shown for every divisor d of $|G|$.

Finite Fields. The simplest example of a finite group can be obtained by considering the integers with addition modulo n, denoted $\langle \mathbb{Z}_n, + \rangle$. This group is however not interesting for direct usage in cryptographic primitives, because of its simple structure. As an alternative, one can consider the multiplicative group of a finite field $\mathbb{F}_p$, i.e. the integers with multiplication modulo a prime p. This group, denoted $\mathbb{F}_p^*$ has order $p - 1$ and is usually represented by the integers $1 \ldots p - 1$.

One can also consider the field $\mathbb{F}_{p^n}$. In this case multiplication is defined modulo an irreducible polynomial $f(X)$ of degree n over $\mathbb{F}_p$. The most common choice of this type are binary fields, i.e. $p = 2$, since these can be efficiently implemented.

Elliptic Curves. Elliptic curves over a finite field are an alternative to directly using finite fields in cryptography. An elliptic curve $\mathbb{E}$ over a field of characteristic ≥ 3 can be specified using the short Weierstrass equation

$$y^2 = x^3 + ax + b\,, \tag{2.9}$$

where all variables and constants are elements of the finite field. All solutions (x, y) to the above equation are (rational) points on the elliptic curve, i.e. $(x, y) \in \mathbb{E}$. Besides these points, the point at infinity O is added to the set of points. Curves are typically defined over $\mathbb{F}_p$ (prime curves) or over $\mathbb{F}_{2^n}$ (binary curves).

The group law (point addition) is defined by taking two elliptic curve points P, Q and constructing a straight line through these points. There is always a third intersection point $R = (x_R, y_R)$ on the curve. The result of the addition is $-R = (x_R, -y_R)$. Note that $P + Q + R = O$. In case $P = Q$ (point doubling) the tangent line to the curve at P is used to determine R.

Computing point additions using an affine coordinate system involves inversions over the finite field. There exist several alternative coordinate systems, such as projective or Jacobian coordinates, that compute the group law in a different way. The choice of coordinate system is highly dependent on the underlying hardware or software platform.

We now consider scalar multiplication, i.e. given an $a \in \mathbb{Z}$ and a point $P \in \mathbb{E}$, compute $aP = \sum_{i=1}^{a} P$. Scalar multiplication can be computed efficiently by the double-and-add algorithm or the Montgomery ladder [92], which require at most $\log_2 a$ point additions and doublings.

Let G be a generator of (a subgroup of) the elliptic curve $\mathbb{E}$ with order n, i.e. $n \cdot G = O$. We call $h = \frac{|E|}{n}$ the cofactor of the curve.

For a more complete overview of elliptic curve cryptography we refer the reader to Cohen and Frey [24].

2.2.2 Discrete Logarithm Problem

Let G be a finite cyclic group with generator g. Given an element $h \in G$, the discrete logarithm problem (DLP) of h w.r.t. g, consists of finding $x \in \mathbb{N}, 0 \leq x < |G|$ such that $h = g^x$. The discrete logarithm problem or one of its variants lies at the basis of numerous cryptographic systems. The discrete logarithm problem is considered computationaly infeasible to solve for large cyclic groups of prime order.

The Elliptic Curve Discrete Logarithm Problem (ECDLP) can be defined in the same way as for any group: given $P, Q \in \mathbb{E}$, find $a \in \mathbb{Z}$ such that $Q = aP$.

There are several variants of the discrete logarithm problem. The One More Discrete Logarithm Problem (OMDL), introduced by Bellare *et al.* [12], deals with multiple simultaneous discrete logarithm problems and gives the adversary an oracle $\mathcal{O}_1$ that returns random elements $A_i = a_i P$ of G. Through a second oracle $\mathcal{O}_2(\cdot)$ the adversary can compute the discrete logarithm of an arbitrary group element. At the end of the game the adversary has to output all the discrete logarithms $\{a_i\}_i$ of the elements received from the m calls to $\mathcal{O}_1$ while making strictly less than m queries to $\mathcal{O}_2$ (with $m > 0$).

Attacks on the Discrete Logarithm Problem. Independent of the specific group, there are generic attacks which break the discrete logarithm problem. It can be shown that solving the discrete logarithm in a group is equivalent to solving the discrete log in its subgroups. To avoid these attacks using the Chinese remainder theorem, the group should have a large enough subgroup of prime order. When working with the multiplicative group of a finite field $\mathbb{F}_p$, one often chooses a prime $p = rq + 1$, with q also prime. For $r = 2$ this is called a *safe prime*. It follows that $|\mathbb{F}_p^*| = 2q$, which implies the existence of a subgroup, called the *Schnorr group*, of large prime order q. A generator g can easily be found by taking $g = h^r \mod p$ with $h^r \neq 1 \mod p$.

The best known generic method to solve the discrete logarithm problem is the Pollard rho method [102], which runs in time $O(\sqrt{p})$ for a group of prime order p. Black box groups are a concept where an (arbitrary) representation is used for group elements. The group operations are not performed directly but are handled by some 'black box', which takes two element representations as input and returns a representation for the result. This way, the internals of a specific group are hidden and only the generic properties of a group can be used. Using black box groups with a random representation, it was shown by Shoup [117] that the lower bound for all possible generic attacks is $\Omega(\sqrt{p})$, which implies that the Pollard rho method is optimal up to a constant factor.

Depending on the concrete group that is used, there are more specific, faster attacks. For several straightforward choices of the group G, like $\mathbb{F}_{2^n}^*, \mathbb{F}_p^*$, there are index calculus based methods [4, 48] that run in subexponential time. For elliptic curves the cofactor should be chosen small. Specific curves are vulnerable to attacks. For example, curves over $\mathbb{F}_{2^n}$ with n composite or a Fermat prime are subject to Weil-descent attacks [42]. Additive reduction attacks, which map the curve to an additive group, can be applied to prime curves with $|\mathbb{E}(\mathbb{F}_p)| = p$ [115]. Multiplicative reductions can be applied by mapping $\mathbb{E}(\mathbb{F}_p)$ to $\mathbb{F}_{p^k}$. The extension degree k should be sufficiently high,

where k is the order of p in $\mathbb{F}_q$ (with q the largest prime factor of $|\mathbb{E}(\mathbb{F}_p)|$) [87]. Since choosing a curve is generally not part of key generation, but is considered a system parameter, these attacks can easily be taken into account when selecting curves.

2.2.3 Diffie-Hellman Problem

The Diffie-Hellman Problem [28] is closely related to the Discrete Logarithm Problem (DLP). The most used variants are the Computational Diffie Hellman Problem (CDH) and the Decisional Diffie-Hellman Problem (DDH):

Definition 2.11 (Computational Diffie-Hellman Problem (CDH))**.** Given a group G and aP, bP with a, b chosen at random from $\mathbb{Z}$, determine abP.

Definition 2.12 (Decisional Diffie-Hellman Problem (DDH))**.** Given a group G and $A = aP, B = bP$ with a, b chosen at random from $\mathbb{Z}$ and a value C. Determine whether $C = xyP$ or whether $C = rP$ with r random.

Obviously, when given an algorithm that solves the DLP, we can trivially solve both the computational and decisional Diffie-Hellman Problem. Note however that, contrary to some of the lattice problems discussed in Sect. 2.1.2, the CDH and DDH are not equivalent. The best strategy for solving CDH seems to be the DLP [84]. On the other hand the DDH assumption does not hold in the multiplicative group $\mathbb{F}_p^*$ (with p prime and g a generator of the group) and for elliptic curves with small embedding degrees although the DLP and CDH assumptions do seem to hold. Therefore the DDH assumption is considered to be a stronger assumption than the DLP.

Like the DLP, the CDH and DDH assumptions also come in several variants. In the gap-CDH the adversary has to solve a CDH but is also given access to an oracle that solves the DDH. The Oracle Diffie Hellman (ODH) problem is to solve the DDH with access to a restricted CDH oracle, i.e. the adversary is given $A = aP, B = bP, H(C)$ and an oracle $H(bZ)$ (for $Z \neq A$) and has to determine whether $C = abP$ or $C = rP$. The function $H(\cdot)$ is a one-way function. The problem was introduced by Abdalla *et al.* [2] for usage in the DHIES encryption scheme. Lower bounds for the security of the problem have been shown in [3].

2.3 Parallel Computing

Computer programs, inspired by the concept of Turing machines [122], are traditionally executed sequentially with only one active instruction at the same time. The underlying hardware that executes the computer programs is however highly parallel by definition: (electrical) signals can flow simultaneously through all parts of the chip to perform a computation, provided the necessary measures for a correct timing are taken.

In this section the basic concepts of parallel computing are introduced. The goal of parallel processing is to obtain a speedup (in latency or total execution time) compared to sequential execution.

Note that we only consider *real* parallelism and not *perceived* parallelism. The latter is often provided by operating systems, that allow multiple processes and threads to run *simultaneously*. In reality however, the processes and threads are scheduled sequentially by the operating system. We can only speak of true parallelism if the underlying processor has multiple cores or other forms of hardware parallelism. We will however not consider parallelism *hidden* from the programmer, like for example optimizations in modern processors that automatically schedule multiple instructions at the same time.

The speedup that can be obtained by using multiple parallel processors is limited by the data dependencies in the algorithm. The critical path is the longest chain of data dependencies. Instead of explicitly studying the data dependencies one often simplifies the situation and assumes a fraction $1 - \alpha$ of the program that is inherently unparallelizable. Amdahl's law [7] relates the speedup achieved for the parallelizable part of the program to the total speedup for the algorithm under the assumption that the total size of the problem remains constant. Assume a fraction α of the program can be parallelized on N processors, then the total speedup would be $\frac{1}{1-\alpha+\alpha/N}$. As the number of processors N grows, the speedup becomes limited by the sequential part. In this work we will always compare the speed of the parallel implementation to existing sequential implementations for a similar problem size, thereby following the idea of Amdahl's law.

2.3.1 Parallel Programming Models

Threads. Threads are one of the most basic parallel programming models. A thread is an independent flow through the same program code. The programmer is given explicit control of the creation and destruction of threads and can determine which thread executes which instructions. The threads

usually share the same memory space, which requires additional memory management between threads.

When a threaded program is executed on a multi-core CPU or a SMP (Symmetric Multiprocessing) system actual parallelism is achieved. This approach is often called Multiple Instruction Multiple Data (MIMD).

SIMD and Bit-level Parallelism. The most basic parallelism that can be found in any computer is at the bit-level. Most processors support arithmetic and bit operations on 8 to 128 bit registers. With increasing register length, more data can be processed with every operation. Algorithms can only exploit long registers for their arithmetic operations if the register length does not exceed the required precision. To resolve this, Single Instruction Multiple Data (SIMD) instructions can be used. The MMX and SSE instruction sets from Intel [61] are the best known examples of SIMD instructions. SIMD instructions allow packing several short data elements in one register and executing operations on those elements with a single instruction.

A more recent example of a SIMD approach is a Graphical Processing Unit (GPU). GPUs used to be restricted to computing graphical operations, such as shaders, 3D rendering... Nowadays manufacturers have created GPUs and programming tools that give programmers full control of the device. A single program, called 'kernel' is uploaded to the device. The kernel is executed many times in parallel (multiple threads). The threads can use the identifier to differentiate the memory areas that they operate on to achieve a SIMD approach. Contrary to a rigid SIMD approach, GPUs allow more flexibility and the operations in each thread can be differentiated using branches.

Distributed Computing. In distributed computing several computers are combined in a network. The computers operate with their own local memory and typically communicate over a network. The communication network can be a local network or even the internet. Due to the network latency communication between computers has to be limited. Since distributed computing is based on independent computers, there is a very high level of flexibility for the programmer. There exist many libraries and tools to facilitate distributed computing [94, 98].

Specific Purpose Hardware. Besides the generic devices and programming models described above, it is also possible to use dedicated hardware. An Application-Specific Integrated Circuit (ASIC) is a full custom chip, that can be designed for a single specific application. Because of the custom design, an

ASIC will always outperform a general computer, but this comes at a high cost for the production of the chip.

A Field-Programmable Gate Array (FPGA) is a processor that contains several generic building blocks that can be rewired for a specific application. This way one can implement a custom chip at a minimal cost.

2.3.2 Concurrency Issues

Parallel programs create several concurrency issues, since many programming models do not guarantee the exact way a parallel program is executed.

Shared Memory. To avoid race conditions, memory operations on shared memory should be mutually exclusive. A mutex (mutual exclusion) can be used to limit access to a memory area to a single thread. A mutex allows exactly one thread to claim access. Other threads that attempt to do so will be blocked. There are more complicated types of mutexes, such as semaphores, that allow more than one thread to claim access. Although mutexes solve issues that could occur by simultaneous operations on memory, they also introduce new problems, such as deadlocks or starvation, in which a program is blocked from executing further since mutexes are held by different threads.

When shared memory is known to remain constant, reading the same memory area with multiple threads poses no problem and usually offers a performance advantage. When the memory area is loaded and the processor has a cache, the cost is only one memory load. GPUs automatically detect simultaneous memory access and will only perform a single memory load per warp. While the memory load is executed, the GPU schedules another task. Note however that simultaneous memory access to random memory areas can be expensive on a GPU: the whole group of threads is blocked until all memory is loaded.

2.4 Provable Security and Privacy

One is usually interested in giving strong evidence of the security of a certain cryptographic system. A key technique are security proofs. To achieve such a proof one has to consider every possible attacker, often subject to certain restrictions like computation time and space, and check if none of these is successful.

A system consists of a set of algorithms (parties) that can interact with each other through some communication channel. In formal terms one can model these algorithms as a set of interactive (probabilistic) Turing machines. For the scope of this thesis, this formalism is however not necessary. For simplicity all algorithms can be considered as some form of computer program with the ability to use randomness. We refer the reader to Goldreich [44] for a more formal treatment of the subject.

Before trying to construct a security proof it is crucial to specify exactly which security features the system should have. This is done in the form of a *security notion* (or privacy notion), which typically specifies what an adversary should not be able to achieve. If there exists one adversary that 'breaks' the security notion for a specific system, then the system is considered broken.

A security notion is usually specified in the form of a security game with a challenger, where an (unspecified) adversary is given a set of oracles to interact with the system. At the end of the security game the adversary has to produce an output, which will be verified and will determine whether or not the adversary succeeds.

After specifying the security notion, a proof can be constructed. To show that there exists no adversary that breaks the security notion, one usually relates the security of the system to one or more presumably hard problems. We now assume the existence of an algorithm $\mathcal{A}$ that breaks the security of the system $\mathcal{S}$ in time T with success probability ϵ. Note that it is not required to actually construct or specify the adversary $\mathcal{A}$. Using $\mathcal{A}$ as a subroutine, an algorithm $\mathcal{B}$ is constructed which breaks one of the presumed hard problems in time T' with success probability ϵ'. The values of T' and ϵ' can be related to the assumed hardness of the problem(s). Based on this assumed hardness one has to conclude that no such $\mathcal{A}$ exists. This process is called a 'reduction' from the security of the system to the hard problem.

We will now discuss more in depth the three main components of a security proof for a system: a security notion, one or more assumptions on the hardness of a problem and the reduction.

2.4.1 Security Notions

When modeling a security notion one needs to make an abstraction of reality, since an almighty adversary can always break the security. The capabilities of the adversary are limited by providing only a number of clearly defined oracles to interact with the actual system. This way, the adversary can be refrained from learning, for example, the internal state of one or more of the

parties, since this will often reveal secrets that are crucial to achieving a security notion. The setup phase, where parties are initialized, is often omitted from the security notion, i.e. when the adversary starts the system is considered to be honestly initialized.

The random oracle (RO) model is used to make abstraction of cryptographic hash functions. In this model, one assumes the existence of an oracle that maps every input value to a random value from the output domain. By using a random oracle as an idealization of hash functions it becomes possible to construct proofs for certain protocols. One of the problems of the RO model is that there are no efficient implementations for a random oracle. There are also some protocols, that are insecure in practice, but can be proven secure in the RO model [21].

Another model, similar to the RO model, is the generic group model. The generic group model provides an idealization of group operations. Instead of directly using an efficient encoding of a group (e.g. coordinates for elliptic curve points), a random encoding is used. An oracle is provided to perform group operations, i.e. the oracle takes two random encodings of group elements and outputs the random encoding of the result. The strength of the generic group model is that it allows verification of protocols without considering the actual group used in the implementation or any of the particular issues that might arise due to the use of a specific encoding. However, the model suffers from similar issues as the random oracle model.

Throughout this thesis we will not rely on idealizations like a random oracle or the generic group model, but instead we make specific assumptions on the building blocks used. All our proofs are in the standard model.

For encryption schemes popular security notions are indistinguishability under chosen plaintext attack (IND-CPA) and indistinguishability under chosen ciphertext attack (IND-CCA). In an *indistinguishability* game a challenger (which is just a predefined algorithm) sets up an encryption scheme which the adversary can interact with through the challenger. Based on a random bit, selected by the challenger, it will behave differently. The adversary has to guess this bit by interacting with the challenger.

Algorithm 4 is the basic IND-CPA game. The adversary consists of two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. After setting up the encryption scheme, the challenger calls $\mathcal{A}_1(pk)$. The adversary can perform any number of encryptions since it has the public key. Finally it outputs two messages (m_0, m_1) and a state S. The challenger computes the encryption c of one of these messages, selected by the random bit b. Then it calls the second part of the adversary $\mathcal{A}_2$ with the ciphertext and the state S from $\mathcal{A}_1$. At the end the adversary outputs a guess

bit g, which is compared with b.

An encryption scheme is said to be IND-CPA secure if no probabilistic polynomial time (p.p.t.) adversary $(\mathcal{A}_1, \mathcal{A}_2)$ has non-negligible advantage over guessing. An adversary has non-negligible advantage if it wins the IND-CPA game (i.e. $b = g$) with probability $\frac{1}{2} + \epsilon(k)$ where $\epsilon(k)$ is non-negligible in the security parameter k.

The intuition behind IND-CPA security is that nobody can distinguish the ciphertexts for different messages without having the private key.

Algorithm 4: The IND-CPA game

1: $b \xleftarrow{\$} 0, 1$
2: $pk, sk \leftarrow KeyGen(1^k)$
3: $(m_0, m_1, S) \leftarrow \mathcal{A}_1(pk)$
4: $c \leftarrow \mathcal{E}_{pk}(m_b)$
5: $g \leftarrow \mathcal{A}_2(c, S)$
6: Check if $b = g$

A *side channel attack* is an attack that is not based on theoretical weaknesses of the algorithm, which are usually captured by the security model, but on the implementation of the cryptosystem. Bleichenbacher [14] demonstrated that IND-CPA is not sufficient for real world scenarios by using an easily available side channel. He showed that an adversary can also use a decryption oracle to assist in attacking the scheme. In the case of RSA PKCS #1 [110] (version 1.5) the adversary can send a ciphertext to the receiver, which will decrypt it and then verify if the plaintext was correctly formatted. By only using this pass/fail information from the receiver, arbitrary messages can be decrypted. Later Manger [82] also showed an attack on RSA-OAEP, which was introduced in RSA PKCS #1 version 2.0 after the Bleichenbacher attack.

IND-CCA2 security provides an answer to this problem [105]. Algorithm 5 shows the IND-CCA2 game. In this case the adversary also gets access to a decryption oracle $\mathcal{O}$. In the first phase, $\mathcal{A}_1$ can query the decryption oracle on any ciphertext. In the second phase, $\mathcal{A}_2$ cannot ask for a decryption of c, since

this would allow the adversary to trivially win the game.

Algorithm 5: The IND-CCA2 game

1: $b \xleftarrow{\$} 0, 1$
2: $pk, sk \leftarrow KeyGen(1^k)$
3: Set up an oracle $\mathcal{O}(d) := \mathcal{D}_{sk}(d)$
4: $(m_0, m1, S) \leftarrow \mathcal{A}_1(pk, \mathcal{O}(\cdot))$
5: $c \leftarrow \mathcal{E}_{pk}(m_b)$
6: Set up an oracle $\mathcal{O}'(d) := \mathcal{D}_{sk}(d)$ if $d \neq c$ or $\perp$ if $d = c$
7: $g \leftarrow \mathcal{A}_2(c, S, \mathcal{O}'(\cdot))$
8: Check if $b = g$

The notion of 'semantic security', as introduced by Goldwasser and Micali [46], implies that an adversary cannot extract any significant information on the plaintext message from a given ciphertext. A scheme is semantically secure if a simulator (that does not get any ciphertext) exists that produces the same output as the real adversary (which does get the ciphertext). Semantic security comes in CPA and CCA2 variants. It was shown that both variants of semantic security are equivalent to the respective indistinguishability based security notions from above [125].

Besides these notions there is also *Non-Malleability* (NME) as introduced by Dolev *et al.* [30]. Intuitively non-malleability implies that an adversary should not be able to produce new, valid ciphertexts that are related to given ciphertexts. There exist both indistinguishability (IND-NME) and simulation-based definitions of non-malleability (SIM-NME) [99].

In a simulation-based definition there has to exist a *simulator*, a p.p.t. algorithm, that manages to produce a similar output as the adversary would do, without getting access to the actual system. The output of the simulator and the real adversary should be indistinguishable for every p.p.t. algorithm. Intuitively, if such a simulator exists for every possible adversary, we have to conclude that an adversary cannot do anything more than an algorithm that does nothing at all with the encryption system. Obviously this is a very strong security notion.

It can be shown that in some cases SIM-NME is stronger than IND-NME and in some cases equivalent [99], depending on the exact definitions used and additional properties of the encryption scheme. One of the main open questions is whether simulation-based security is really required to have a system that is secure in practice. There exist protocols that achieve IND-NME but not SIM-NME, but these examples are artificial and do not seem to have a relation with practical cryptoschemes.

For signature schemes the most common notion is *existential unforgeability*, which implies that it is hard to forge any message-signature pair, even when the adversary can determine the message itself.

2.4.2 Hardness Assumptions

At the basis of every security proof there are one or more hardness assumptions. In the case of public key cryptography, these are usually assumptions on the hardness of certain number theoretical problems. Common number theoretical problems are the discrete logarithm problem (see Sect. 2.2.2), the Diffie-Hellman problem (see Sect. 2.2.3), the RSA problem [109]...

Instead of directly using number theoretical hardness assumptions, it is common to use higher level building blocks and make assumptions about these. Common building blocks are, for example, IND-CPA and IND-CCA2 encryption schemes or an existentially unforgeable signature scheme.

Most schemes used in hardness assumptions can be tuned by a security parameter, denoted k. When increasing k it becomes harder to break the underlying assumption. Ideally the hardness should grow exponentially with k. The security parameter affects the efficiency of the algorithms: as the hardness increases one has to use more or longer data values in the scheme, which slows down the operations.

When working with asymptotic security we often use terms such as *polynomial* and *(non-)negligible* functions. An explicit reference to the security parameter k is often omitted.

A function $f : \mathbb{N} \to \mathbb{R}$ is called 'polynomial' in the security parameter $k \in \mathbb{Z}$ if $f(k) = O(k^n)$, with $n \in \mathbb{N}$. It is called 'negligible' if, for every $c \in \mathbb{N}$ there exists an integer k_c such that $f(k) \leq k^{-c}$ for all $k > k_c$. We denote a negligible function by $\epsilon(k)$ or simply ϵ.

2.4.3 Reductions

Instead of considering the exact time and probability for breaking the system, the asymptotic complexity is considered. In most cases only polynomial time adversaries are considered and a system is considered asymptotically secure if there exist no p.p.t. adversaries. Both the hardness assumption and the security notion will be formulated in the asymptotic sense. For the reduction it suffices to find a p.p.t. algorithm that breaks the hardness assumption, given an algorithm that breaks the security notion for a specific system.

A major issue when trying to map asymptotic security to real life security is the tightness of the reduction. The tightness of a reduction is the loss factor between the time/probability (T'/ϵ') required for breaking the hardness assumption and the time/probability (T/ϵ) for breaking the specific system. A reduction can make multiple calls to the algorithm that breaks the security notion and can also do its own computations, all of which will increase the time/probability gap. To allow for actual conclusions on the real security of a practical system, tight reductions are required, which relate attacks against the system to an attack against the hardness assumption in the same time $T = T'$ and success probability $\epsilon = \epsilon'$ (e.g. [13]).

Hybrid Argument. A common problem in reductions is that a hardness assumption only holds for a single instance of a problem, but the system requires multiple instances of the same problem (e.g. multiple protocol runs or multiple bits). A hybrid argument is a proof technique that is used to relate the hardness of a single instance of a problem to the hardness of multiple instances.

Consider two distributions (samplable in polynomial time) α and β. We say that α is indistinguishable from β (denoted $\alpha \approx \beta$) if for all p.p.t. adversaries $\mathcal{A}$ it holds that

$$\Pr\left[x_0 \xleftarrow{\$} \alpha, x_1 \xleftarrow{\$} \beta, b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{A}(x_b), b = b'\right] = \frac{1}{2} + \epsilon(k) \tag{2.10}$$

with $\epsilon(k)$ negligible.

Theorem 2.3 (Hybrid argument)**.** *Assume a set of distributions* $\{\alpha_i\}$ *for* $0 \leq i \leq n$ *and assume that* $\alpha_{i-1} \approx \alpha_i$ *with advantage at most* ϵ_i*. It follows that* $\alpha_0 \approx \alpha_n$ *with advantage at most* $\sum_{i=1}^{n} \epsilon_i$*.*

Theorem 2.3 is the core of the hybrid argument. It assumes a sequence of distributions (*hybrid distributions*, hence the name). By assuming the indistinguishability of two consecutive distributions, the indistinguishability of the two extreme distributions α_0 and α_n follows. The two consecutive distributions usually differ only by the application of a cryptographic primitive or other hard problem. One of the extremes corresponds to the real system, while the other is usually some idealized or randomized version of the scheme.

Note that when a hybrid argument is used, there is always a loss factor n, the length of the hybrid sequence, so the reduction is non-tight. In some cases it is possible to avoid this loss factor and construct tight reductions [35], but it is in no way a general result.

A simple example is multi-message IND-CPA security. In the IND-CPA game from above the adversary can only ask the oracle for the encryption of a single message. In a multi-message IND-CPA game the adversary can ask the encryption of n messages, i.e. it can give two vectors of messages $\mathbf{m}^0 = (m_1^0, m_2^0, \ldots, m_n^0), \mathbf{m}^1 = (m_1^1, m_2^1, \ldots, m_k^1)$ to the encryption oracle. The adversary then has to distinguish between $(c_1^0, c_2^0, \ldots, c_n^0)$ and $(c_1^1, c_2^1, \ldots, c_n^1)$ where $c_j^i = \mathcal{E}_{pk}(m_j^i)$.

We will denote the output distribution $(c_1^1, c_2^1, \ldots, c_n^1)$ by α_0 and $(c_1^0, c_2^0, \ldots, c_n^0)$ by α_n. The hybrid distributions are defined as

$$\alpha_i = (c_1^1, \ldots, c_i^1, c_{i+1}^0, \ldots, c_n^0)\,. \tag{2.11}$$

For an encryption scheme that is single message IND-CPA secure it obviously holds that $\alpha_i \approx \alpha_{i+1}$. Using the hybrid argument it follows that $\alpha_0 \approx \alpha_n$, i.e. the encryption scheme is also multi-message IND-CPA secure.

Chapter 3

Contributions

In this chapter we summarize our research contributions. The first section of this chapter describes the efficient software implementations and cryptanalysis contributions. The second section discusses the contributions concerning RFID privacy and protocols.

3.1 Efficient Implementations

With the onset of GPU computing we explored the possibilities of using GPUs for efficient implementations of lightweight public key algorithms and cryptanalysis of these algorithms.

3.1.1 NTRU Encryption

In [54] we present an efficient implementation of NTRU encryption on GPUs. GPUs are very fast when it comes to massive parallel computation. Before the introduction of general purpose GPU programming, GPUs were restricted to running shaders. Shaders typically perform identical operations (i.e. 3D transformations or pixel operations) on large amounts of data. By abusing these shaders it was already possible to perform fast cryptographic operations. General purpose GPU programming greatly simplified this.

Due to evolutions in the cryptanalysis of NTRU encryption [56, 59], the security parameters were increased quite severely, which had an impact on

the performance of the algorithm. We decided to use the highest available security level to test the efficiency of the NTRU algorithm on GPU, which uses polynomials in $\mathbb{Z}_{2^{11}}[X]/(X^{1171}-1)$. Previous implementations were based on much lower security parameters (e.g. [8, 9]).

NTRU encryption mainly consists of vector convolutions, a very simple operation without any branching. We show that the algorithm lends itself excellently to parallel computation on GPU. Due to the limited amount of computation involved the main bottleneck was going to be memory access latency, given the length of the vector. When computing a large number of NTRU encryptions in parallel however, this memory latency is hidden since the GPU automatically reschedules threads on the hardware level. Moreover, since all memory access is coalesced (threads in parallel access consecutive or identical memory areas), the GPU only needs to reschedule once for every block of threads. Due to these specific mechanisms for simultaneous memory access the performance penalty is minimized and we achieve performance rates of up to 220 000 encryptions per second. We compare our implementation both with existing NTRU implementations and with other encryption schemes, such as RSA and ECC. At a high security level NTRU is much faster than RSA (around five orders of magnitude) and ECC (around three orders of magnitude). Even when only performing a single operation NTRU is still faster by approximately a factor of 35 for 2048 bit RSA and 3 for ECC NIST-244.

3.1.2 Lattice Enumeration

Motivated by the high performance rate for NTRU encryption we attempted to perform lattice enumeration on a GPU [52, 53]. Lattice enumeration is a common technique for finding short vectors in lattices, which is used frequently in cryptanalysis. The most important enumeration algorithm is due to Schnorr and Euchner [113]. The algorithm performs an exhaustive tree search, using a specific pruning technique to remove infeasible branches. Compared to the NTRU encryption algorithm, lattice enumeration is much more computationally intensive. Moreover, it requires more local memory (registers and shared memory), which will limit the dimension of the lattice that can be enumerated.

One of the key challenges was how to parallelize an inheritly serial algorithm, such as a branch-and-bound tree search. An initial strategy was to start enumeration from the root of the search tree and stop at a certain depth. These vectors would then be uploaded to the GPU for parallel enumeration of the subtrees. This strategy however turned out to be inefficient, since the

execution times for the subtrees were unbalanced, leaving many GPU threads idle.

To solve the issues with unbalanced subtrees, our program uploads more starting vectors than threads to the GPU. The GPU kernel has an automatic reloading routing, which allows a thread to select a new subtree for enumeration after finishing the previous one. The downside of this approach is that there is a penalty for the rescheduling routine, which slows down all threads running in parallel even when just a single thread requires reloading. However, this strategy performs better than the plain approach with subtree enumeration. For sufficiently high lattice dimensions we obtain timings up to five times faster than a CPU implementation.

3.2 RFID Privacy and Protocols

3.2.1 RFID Privacy Model

The main motivation for the research related to RFID privacy models was the problem of achieving *strong privacy*, put forward by Vaudenay [124]. Vaudenay shows that, in his model, it is *theoretically impossible to achieve strong privacy.* The proof heavily relies on the required simulation of the reader output (i.e. accepting/rejecting). A protocol is considered private if there exists a blinder that perfectly simulates the RFID system to the adversary without having access to the real system. By assuming a protocol with a low privacy level, one can show the existence of a blinder that simulates the reader output without access to the secrets of the reader. The proof is sound, but on the other hand uses games involving only a single tag and no communication with that tag at all. Intuitively, breaking the privacy of a tag would require communication with at least one tag.

This raises the question of whether the impossibility result was due to the model or due to more fundamental issues underlying the general concept of privacy for RFID tags. We support the former view and in [50] we presented a new model based on indistinguishability. Using this model we show, with a concrete protocol, that strong privacy can indeed be achieved without introducing unnecessary restrictions to the adversary in the model.

One of the design goals for the new privacy model was that it should be easy to apply in reductions and should match the real world notion of RFID privacy as closely as possible. In our model the adversary is given the ability to interactively select which tags to use in protocol runs in a ‘left’ and ‘right’ world indistinguishability game. The challenger executes one of these worlds

based on a challenge bit and the goal for the adversary is to guess this bit. This construction not only guarantees that individual protocol runs and tags are indistinguishable, but also more complicated interactions involving multiple tags.

By using an indistinguishability based framework we avoid the issues surrounding the use of simulators. The issues for RFID privacy strongly resemble those encountered with simulatability based encryption models. As with encryption, it remains an open problem whether there is truly a need for using simulatability based frameworks or whether indistinguishability is required.

In our work we also evaluate several existing RFID privacy models and try to provide some guidance on the strengths and weaknesses of these models. We demonstrate that some of these models do not appropriately model the actual threats or rely on poorly motivated assumptions.

3.2.2 Private RFID Protocols

The ECRAC protocol (in all of its versions [74–76]) only requires elliptic curve operations and was claimed to be an efficient wide strong RFID identification protocol. Since only a security argument and no security proof was presented in the paper, there was reason to either try to find a valid security proof or an attack on the protocol. In [34] we show that the protocol fails to provide any of the claimed privacy levels and establish an upper bound for the remaining privacy (which still remains unproven).

The authors of ECRAC do however provide an efficient hardware implementation for elliptic curve operations, which motivated our goal to achieve strong privacy using elliptic curve operations only, without requiring any additional symmetric components. An additional goal put forward was to have proofs in the standard model for security and privacy.

In [100] we address these open questions and present several protocols satisfying these requirements. Before, the only known way of achieving strong privacy was using a protocol based on IND-CCA2 encryption or equivalent notions [20,124], which requires more operations or involves the evaluation of a hash function.

We present two variants of the same protocol, with different soundness (authentication) properties and efficiency. All variants achieve strong privacy. Our most efficient protocol requires only two elliptic curve scalar multiplications. One of the new properties we introduce is *extended soundness*, which implies that a reader cannot impersonate a tag. This property was missing in previous proposals based on IND-CCA2 encryption since these relied on a shared secret

between reader and tag. Using extended soundness multiple independent readers can be supported.

Compared to the initial publication the model in [100] has some minor technical changes, mainly to support tag-initiated protocols and to allow simultaneous interactions with all tags instead of only half of the tags. These changes do not effect the previous privacy and security proofs at all.

3.2.3 Grouping Proofs

Grouping proofs (or yoking protocols) were first introduced by Juels [63] as a system to allow offline verification of the fact that several tags interacted at a specific time. Many protocols for constructing grouping proofs can be found in literature, but, again, hardly any protocol has a security proof. Some protocols even require interactions with the verifier during construction of the grouping proof, thereby violating the whole concept of offline verification of the proofs. Despite the abundance of protocols for grouping proofs, there exists no formal security definition for a grouping proof, which raises even higher doubts about the existing protocols: what do these protocols actualy offer?

In [51] we give the first formal definition for a grouping proof. The definition comes in two flavors: a timed version which ensures that the tags participated in the grouping proof at a specific time and a non-timed definition that only ensures that the tags participated at least once in the past in constructing the grouping proof. The timed version obviously requires a trusted timestamping authority.

Using this definition we present two efficient, narrow strong private grouping proofs (one timed and one non-timed). We also construct an attack on a previously proposed private grouping proof [11], which enables us to create arbitrary forgeries of grouping proofs.

3.3 Other Publications and Work

3.3.1 QR Factorization of Random Circular Matrices

In [49] the QR factorization of random circulant matrices is investigated. We show that the orthogonal basis $\mathbf{Q}$ has an almost circulant structure, with only small deviations from a true circulant matrix. The upper triangular matrix $\mathbf{R}$ has an almost Toeplitz structure. These structures could be exploited to

speed up the peformance of computations on circulant matrices, especially when orthogonalization is required.

3.3.2 Mutual Authentication and Privacy

In recent work we further extend the privacy model and the authentication protocols to support mutual authentication. Unlike previous proposals, the reader authenticates to the tag first, after which the tag authenticates to the reader. By doing authentication in this order it is easier to achieve private protocols: the tag is certain of the identity of the reader before revealing its own identity. Moreover it allows for narrow protocols, where the reader does not give any output on success or failure of the tag authentication, whereas this would be impossible if tag authentication preceeds reader authentication.

Chapter 4

Conclusion and Open Problems

4.1 Conclusion

The uprise of lightweight devices, such as RFID tags, has created new security and privacy challenges. Since these devices are so ubiquitous and communication goes unnoticed they can easily be abused. Cryptographic algorithms and protocols are used to provide security and privacy protection. Given the constraints on chip area, time, power and energy conventional cryptographic solutions can usually not be applied. Lightweight cryptography is put forward as a solution to still obtain sufficiently secure cryptography on these devices.

Developing lightweight cryptography is challenging, not only because of the constraints of the devices, but also because of the growing computational power that can be used for attacking cryptography. In addition these devices are susceptible to tampering, so when designing systems it has to be taken into account that secrets will be extracted from some devices.

The main question is how secure lightweight cryptography still is, with continuous improvements being made in cryptanalysis and computing power. We have tried to partially answer this question by implementing NTRU encryption and lattice reduction for the first time an a GPU. At the time of writing GPUs were relatively new in the world of cryptography and it was unclear what the impact could be on the performance of cryptographic

algorithms and speeding up cryptanalysis. Our implementation of NTRU shows that extremely high throughput (up to 200 000 encryptions per second) can be achieved even with public key cryptography. Our lattice enumeration implementation demonstrates that GPUs can be used for improving the performance of cryptanalysis.

In previous work a myriad of RFID identification and authentication protocols can be found, with varying quality and claimed security and privacy properties. Most times proofs for security and privacy were omitted by the authors, resulting in an avalanche of new proposals, followed by swift attacks and subsequent fixes. Instead of aiming for trial-and-error protocol design, we have chosen the direction of sound protocol design based on provable security. Key questions were how to model privacy and security for RFID tags and if it was possible to develop *efficient* protocols that still achieve very strong privacy and security notions.

In our work we compared the pros and cons of several of the existing models and showed poor design choices in several models that overly restrict adversaries. Previous proposals also did not allow for strong privacy. We have proposed a new model that does support this notion without introducing unnecessary restrictions or compromising on the 'real world' privacy properties.

Finally we proposed new private RFID identification protocols and grouping proofs, using the new model to prove privacy properties. This way, we demonstrated that IND-CCA2 encryption is not strictly required to achieve strong privacy but that more efficient solutions exist.

4.2 Open Problems and Future Perspectives

Our work has answered several questions in the area of lightweight public-key cryptography by providing theoretical models, protocols and cryptanalysis results. Many areas were however left uncovered and our work has also created new research questions. We give an overview of some open research problems that already sparked our interest or came to our mind as being important directions for future research by the wider community.

4.2.1 Security of Lightweight Primitives and Implementations

- New lattice reduction techniques have emerged, among which extreme pruning for lattice enumeration [39] and sieving techniques. Whether or not these algorithms can be massively parallelized, on a GPU or other

platform, is still an open question. The extreme pruning technique should be easy to implement in our existing implementation, but might have side-effects on the depth of subtrees and performance of the scheduler. Sieving on the other hand is very memory intensive, which could be a problem for parallel computing.

- An interesting option is also the combination of lattice reduction techniques and ordinary brute forcing, such as the hybrid attacks on NTRU [59], or brute force attacks in general. By definition these should be very easy to speed up by parallelization.

- There exist many other platforms for parallel computing. One could set up CPU or GPU clusters to perform massive parallel computing and break cryptographic schemes with high security parameters instead of toy examples.

- Recently a variant of NTRU encryption was proposed by Stehlé *et al.* [119] that can be shown provable secure under assumptions on ideal lattices. The main difference is that the variant operates over a different ring $(\mathbb{Z}[X]_q/(X^N + 1)$, with $N = 2^n)$, uses a different distribution for the randomness and adds a small error term in the encryption operation. It is unclear if this variant can be implemented as efficiently on a GPU as the original NTRU encryption. The main questions are how to sample from the random distribution efficiently on a GPU and if the additional error term causes additional memory overhead.

4.2.2 Privacy Models

- With so many privacy models around an obvious question is which one to use? One could argue that the strength of the model is an important factor. This implies however establishing a hierarchy between all the models, using reductions or by giving separation results. To date, few reductions are available. Establishing these results is not easy and may even require making some changes to the models to make them more comparable.

- Strength is however not the sole criterion for selecting security/privacy models. Another element that is equally important and might be easier to evaluate is the way the model captures the real world threats. When looking only at the strength there is a risk of over-aiming and imposing restrictions on algorithms that do not prevent any realistic attack. More work should be done on comparing the features of privacy models and trying to determine if models capture actual threats instead of providing

an overly artificial abstraction of real threats. In the case of such artificial abstractions one should wonder what the protocols that are private under such a model actualy provide in a more realistic setting.

- Tag corruption is approached very differently by most models. A first difference is that the output of the corruption differs: some models only release the long term state, others also consider temporary values or intermediate computation. A second difference is that the time of corruption or the tags that can be corrupted is restricted sometimes. Models that use some 'challenge' tag(s) often do not allow the corruption of these tags. This type of restriction seems to stem more from modelling issues rather than capturing an actual constraint of a real adversary. What is the actual impact on privacy of this type of model-inspired restrictions and can these restrictions be removed? Or are the restrictions necessary when using certain models? This last question can also be reversed: should the model be changed because it overly restricts corruption?
- One of the main theoretical issues is the difference between simulatability and indistinguishability based privacy models. There is no doubt that most simulatability based models are stronger. The question whether there exist any real threats that are not captured by an indistinguishability model is still left open. Are there really convincing arguments why to prefer simulatability or is indistinguishability sufficient? This is exactly the same open problem as in encryption, where only artificial counterexamples are known.
- There is a link between leakage resilient cryptography and RFID privacy. RFID privacy considers device corruption, while leakage resilient cryptography considers leakage of information from the internals of the algorithm. How much do these concepts overlap and can concepts from leakage resilient cryptography be introduced for RFID privacy and security?

4.2.3 Protocols

- Which other use cases are there for RFID tags (and other lightweight devices) besides identification? The work on RFID privacy seems to be very general and applicable in many areas. Can lightweight, private equivalents of existing conventional cryptographic protocols be developed?
- In our proposals variants of the discrete logarithm problem (for security) and the Diffie-Hellman problem (for privacy) are used. Even if it is

quite common in literature to rely on DLP and DDH variants, there are few results about the relationship among all these problems. Consider the optimized RFID identification protocol from [100], where we use a combination of DDH and DLP to prove privacy to limit the number of variables required. By removing the strict separation between DDH and DLP the cost is reduced from four to two point multiplications, but on the other hand this combined primitive is less studied in the literature. These problems need further investigation, since the efficiency of algorithms can be improved by using these variants.

- In recent work new lightweight authentication protocols based on lattice assumptions or similar problems (e.g. ring LPN) have been proposed [55]. The protocols are however not provably secure against man-in-the-middle attacks and do not have privacy protection. Is it possible to use these assumptions to build even more efficient protocols that can provide strong security and privacy features?

Bibliography

[1] 1363 Working Group of the C/MSC Committee. IEEE P1363.1 Standard Specification for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, 2009. Available at `http://grouper.ieee.org/groups/1363/`.

[2] M. Abdalla, M. Bellare, and P. Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman Problem. *Micro*, 2020:1–30, 2001.

[3] M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In D. Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[4] L. M. Adleman. A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography (Abstract). In *FOCS*, pages 55–60. IEEE Computer Society, 1979.

[5] M. Ajtai. Generating Hard Instances of Lattice Problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(7), 1996.

[6] M. Ajtai and C. Dwork. A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence. In F. T. Leighton and P. W. Shor, editors, *STOC*, pages 284–293. ACM, 1997.

[7] G. Amdahl. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. *AFIPS Conference Proceedings*, 30:483–485, 1976.

[8] A. C. Atici, L. Batina, J. Fan, I. Verbauwhede, and S. B. Örs. Low-cost implementations of NTRU for Pervasive Security. In *ASAP*, pages 79–84. IEEE Computer Society, 2008.

[9] D. V. Bailey, D. Coffin, A. J. Elbirt, J. H. Silverman, and A. D. Woodbury. NTRU in Constrained Devices. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 262–272. Springer, 2001.

[10] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The Sorcerers Apprentice Guide to Fault Attacks. *IACR Cryptology ePrint Archive*, 2004:100, 2004.

[11] L. Batina, Y. K. Lee, S. Seys, D. Singelée, and I. Verbauwhede. Privacy-Preserving ECC-Based Grouping Proofs for RFID. In M. Burmester, G. Tsudik, S. S. Magliveras, and I. Ilic, editors, *ISC*, volume 6531 of *Lecture Notes in Computer Science*, pages 159–165. Springer, 2010.

[12] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum?s Blind Signature Scheme. *Journal of Cryptology*, 16:185–215, 2003.

[13] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures - HOw to Sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[14] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

[15] H. Bock, M. Braun, M. Dichtl, E. Hess, J. Heyszl, W. Kargl, H. Koroschetz, B. Meyer, and H. Seuschek. A Milestone Towards RFID Products Offering Asymmetric Authentication Based on Elliptic Curve Cryptography. In *RFIDSec 2008*, 2008.

[16] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede. SPONGENT: A Lightweight Hash Function. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–327, Nara,JP, 2011. Springer.

[17] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Eliminating Errors in Cryptographic Computations. *J. Cryptology*, 14(2):101–119, 2001.

[18] J. Buchmann and C. Ludwig. Practical Lattice Basis Sampling Reduction. In *ANTS*, volume 4076 of *Lecture Notes in Computer Science*, pages 222–237. Springer, 2006.

[19] J. Cai and A. Nerurkar. An Improved Worst-Case to Average-Case Connection for Lattice Problems. In *FOCS*, pages 468–477. IEEE Computer Society, 1997.

[20] S. Canard, I. Coisel, J. Etrog, and M. Girault. Privacy-Preserving RFID Systems: Model and Constructions. *IACR Cryptology ePrint Archive*, 2010:405, 2010.

[21] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In J. S. Vitter, editor, *STOC*, pages 209–218. ACM, 1998.

[22] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai Trees, or How to Delegate a Lattice Basis. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, 2010.

[23] Y. Chen and P. Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In D. H. Lee and X. Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

[24] H. Cohen and G. Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, volume 34 of *Discrete Mathematics and Its Applications*. Chapman & Hall/CRC, 2005.

[25] D. Coppersmith. Finding a Small Root of a Univariate Modular Equation. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 1996.

[26] C. Coupé, P. Q. Nguyen, and J. Stern. The Effectiveness of Lattice Attacks Against Low-Exponent RSA. In H. Imai and Y. Zheng, editors, *PKC*, volume 1560 of *Lecture Notes in Computer Science*, pages 204–218. Springer, 1999.

[27] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[28] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[29] I. Dinur. Approximating $\mathrm{SVP}_{\mathrm{infinity}}$ to within Almost-Polynomial Factors is NP-hard. *Theor. Comput. Sci.*, 285(1):55–71, 2002.

[30] D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[31] G. Durfee and P. Q. Nguyen. Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt '99. In T. Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2000.

[32] EPCglobal Inc. EPC Tag Data Standard . Available at `http://www.epcglobalinc.org/`.

[33] European Network of Excellence in Cryptology ECRYPT. The SHA-3 Zoo: SHA-3 Hardware Implementations, July 2012. Available at `http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations`.

[34] J. Fan, J. Hermans, and F. Vercauteren. On the Claimed Privacy of EC-RAC III. In S. B. O. Yalcin, editor, *RFIDSec*, volume 6370 of *Lecture Notes in Computer Science*, pages 66–74. Springer, 2010.

[35] B. Fefferman, R. Shaltiel, C. Umans, and E. Viola. On Beating the Hybrid Argument. In S. Goldwasser, editor, *ITCS*, pages 468–483. ACM, 2012.

[36] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES implementation on a grain of sand. *IEEE Proceedings Information Security*, 152(1):13–20, 2005.

[37] U. Fincke and M. Pohst. A Procedure for Determining Algebraic Integers of Given Norm. In *EUROCAL 1983*, volume 162 of *Lecture Notes in Computer Science*, pages 194–202. Springer, 1983.

[38] N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC*, pages 207–216. ACM, 2008.

[39] N. Gama, P. Q. Nguyen, and O. Regev. Lattice Enumeration Using Extreme Pruning. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer, 2010.

[40] T. E. Gamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[41] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[42] P. Gaudry, F. Hess, and N. P. Smart. Constructive and Destructive Facets of Weil Descent on Elliptic Curves. *J. Cryptology*, 15(1):19–46, 2002.

[43] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and new Cryptographic Constructions. In C. Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[44] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[45] O. Goldreich, S. Goldwasser, and S. Halevi. Public-Key Cryptosystems from Lattice Reduction Problems. In B. S. K. Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 1997.

[46] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[47] G. Hanrot and D. Stehlé. Improved Analysis of Kannan's Shortest Lattice Vector Algorithm. In A. Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 170–186. Springer, 2007.

[48] M. E. Hellman and J. M. Reyneri. Fast Computation of Discrete Logarithms in GF(q). In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *CRYPTO*, pages 3–13. Plenum Press, New York, 1982.

[49] J. Hermans. QR Factorization of Circulant Matrices. Cosic internal report, 2009.

[50] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A New RFID Privacy Model. In V. Atluri and C. Diaz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 568–587. Springer, 2011.

[51] J. Hermans and R. Peeters. Private Yoking Proofs: Attacks, Models and new Provable Constructions. In I. Verbauwhede, editor, *RFIDSec*, Lecture Notes in Computer Science. Springer, 2012. To appear.

[52] J. Hermans, M. Schneider, J. Buchmann, B. Preneel, and F. Vercauteren. Parallel Shortest Lattice Vector Enumeration on Graphics Cards. In D. J. Bernstein and T. Lange, editors, *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 52–68. Springer, 2010.

[53] J. Hermans, M. Schneider, F. Vercauteren, J. Buchmann, and B. Preneel. Shortest Lattice Vector Enumeration on Graphics Cards. In *SHARCS*, Lausanne, CH, 2009.

[54] J. Hermans, F. Vercauteren, and B. Preneel. Speed Records for NTRU. In J. Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2010.

[55] S. Heyse, E. Kiltz, V. Lyubashevsky, C. Paar, and K. Pietrzak. Lapin: An efficient authentication protocol based on Ring-LPN. In *FSE*, Lecture Notes in Computer Science. Springer, 2012.

[56] P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte. Choosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *ACNS*, volume 5536 of *Lecture Notes in Computer Science*, pages 437–455, 2009.

[57] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In J. Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[58] J. Hoffstein, J. H. Silverman, and W. Whyte. Estimated Breaking Times for NTRU Lattices. Technical report, NTRU Cryptosystems, 2003.

[59] N. Howgrave-Graham. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In A. Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007.

[60] S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel. A Practical Attack on KeeLoq. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[61] Intel. Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference Manual.

[62] International Civil Aviation Organization. Machine Readable Travel Documents (Doc 9309), 2006.

[63] A. Juels. "Yoking-Proofs" for RFID Tags. In *PerCom Workshops*, pages 138–143. IEEE Computer Society, 2004.

[64] A. Juels, D. Molnar, and D. Wagner. Security and Privacy Issues in E-passports. In *SecureComm*, pages 74–88. IEEE, 2005.

[65] R. Kannan. Improved algorithms for integer programming and related lattice problems. In D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, editors, *STOC*, pages 193–206. ACM, 1983.

[66] R. Kannan. Minkowski's Convex Body Theorem and Integer Programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

[67] S. Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005.

[68] M. Knezevic, K. Kobayashi, J. Ikegami, S. Matsuo, A. Satoh, U. Kocabas, J. Fan, T. Katashita, T. Sugawara, K. Sakiyama, I. Verbauwhede, K. Ohta, N. Homma, and T. Aoki. Fair and Consistent Hardware Evaluation of Fourteen Round Two SHA-3 Candidates. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–13, 2011.

[69] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[70] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[71] H. Koy. Primale-Duale Segment-Reduktion. Available at `http://www.mi.informatik.uni-frankfurt.de/research/papers.html`, 2004.

[72] J. C. Lagarias and A. M. Odlyzko. Solving Low-Density Subset Sum Problems. *J. ACM*, 32(1):229–246, 1985.

[73] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14:699–719, 1966.

[74] Y. K. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-Cost Untraceable Authentication Protocols for RFID. In *WISEC*, Hoboken, NJ, USA, 2010. ACM. Preprint.

[75] Y. K. Lee, L. Batina, and I. Verbauwhede. EC-RAC (ECDLP Based Randomized Access Control): Provably Secure RFID authentication protocol. In *IEEE International Conference on RFID*, pages 97–104, Las Vegas, NA, USA, 2008. IEEE.

[76] Y. K. Lee, L. Batina, and I. Verbauwhede. Untraceable RFID Authentication Protocols: Revision of EC-RAC. In *IEEE International Conference on RFID*, pages 178–185, Orlando, FL, USA, 2009. IEEE.

[77] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[78] H. W. Lenstra. Integer Programming with a Fixed Number of Variables. *Math. Oper. Res.*, 8:538–548, 1983.

[79] V. Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

[80] V. Lyubashevsky and D. Micciancio. Asymptotically Efficient Lattice-Based Digital Signatures. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2008.

[81] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: A Modest Proposal for FFT Hashing. In K. Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer, 2008.

[82] J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0. In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238. Springer, 2001.

[83] Mastercard Inc. Mastercard Paypass. Available at `http://www.paypass.com/`.

[84] U. M. Maurer. Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Algorithms. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 271–281. Springer, 1994.

[85] A. May. Using LLL-Reduction for Solving RSA and Factorization Problems. In P. Q. Nguyen and B. Vallée, editors, *The LLL algorithm*, pages 315–348. Springer, 2010.

[86] R. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. Technical report, 1978. Available at `http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF`.

[87] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[88] R. C. Merkle. Secure Communications Over Insecure Channels. *Commun. ACM*, 21(4):294–299, 1978.

[89] D. Micciancio. Almost Perfect Lattices, the Covering Radius Problem, and Applications to Ajtai's Connection Factor. *SIAM J. Comput.*, 34(1):118–169, 2004.

[90] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002.

[91] D. Micciancio and P. Voulgaris. Faster Exponential Time Algorithms for the Shortest Vector Problem. In M. Charikar, editor, *SODA*, pages 1468–1480. SIAM, 2010.

[92] P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177), 1987.

[93] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.

[94] MPI Forum. Message Passing Interface Forum. See `http://www.mpi-forum.org/`.

[95] National Institute for Standards and Technology. Digital signature standard (DSS) federal information processing standard 186-2, 2000. Available at `http://csrc.nist.gov/publications/fips/`.

[96] P. Q. Nguyen and D. Stehlé. Floating-Point LLL Revisited. In *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer, 2005.

[97] P. Q. Nguyen and T. Vidick. Sieve Algorithms for the Shortest Vector Problem are Practical. *J. of Mathematical Cryptology*, 2(2), 2008.

[98] Nvidia. NVIDIA Cuda C Programming Guide, 2012. Available at `http://developer.nvidia.com`.

[99] R. Pass, abhi shelat, and V. Vaikuntanathan. Relations Among Notions of Non-malleability for Encryption. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 519–535. Springer, 2007.

[100] R. Peeters and J. Hermans. Wide Strong Private RFID Identification based on Zero-Knowledge, 2012. In submission.

[101] C. Peikert. Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem: extended abstract. In M. Mitzenmacher, editor, *STOC*, pages 333–342. ACM, 2009.

[102] J. M. Pollard. A Monte Carlo Method for Factorization. *BIT Numerical Mathematics*, 15(3):331–334, 1975.

[103] X. Pujol and D. Stehlé. Rigorous and Efficient Short Lattice Vectors Enumeration. In J. Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 390–405. Springer, 2008.

[104] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In I. Attali and T. P. Jensen, editors, *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[105] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In J. Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[106] O. Regev. New Lattice-Based Cryptographic Constructions. *J. ACM*, 51(6):899–942, 2004.

[107] O. Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 84–93. ACM, 2005.

[108] O. Regev and R. Rosen. Lattice Problems and Norm Embeddings. In J. M. Kleinberg, editor, *STOC*, pages 447–456. ACM, 2006.

[109] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[110] RSA Laboratories. PKCS#1 version 2.1, 2002. Available at `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`.

[111] C.-P. Schnorr. Factoring Integers and Computing Discrete Logarithms via Diophantine Approximations. In D. W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 281–293. Springer, 1991.

[112] C.-P. Schnorr. Lattice Reduction by Random Sampling and Birthday Methods. In *STACS*, volume 2607 of *Lecture Notes in Computer Science*, pages 146–156. Springer, 2003.

[113] C.-P. Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. In L. Budach, editor, *FCT*, volume 529 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 1991.

[114] C.-P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction. In *EUROCRYPT*, volume 921 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1995.

[115] I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Math. Comput.*, 67(221):353–356, 1998.

[116] V. Shoup. Number Theory Library (NTL) for C++. Available at `http://www.shoup.net/ntl/`.

[117] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In W. Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[118] F.-X. Standaert and F. Koeune. Qu'attend-on d'un ticket de métro ? *Le Soir*, 27/08/2009.

[119] D. Stehlé and R. Steinfeld. Making NTRU as Secure as Worst-Case Problems over Ideal Lattices. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 27–47. Springer, 2011.

[120] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient Public Key Encryption Based on Ideal Lattices. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer, 2009.

[121] S. Tillich, M. Feldhofer, W. Issovits, T. Kern, H. Kureck, M. Mühlberghuber, G. Neubauer, A. Reiter, A. Köfler, and M. Mayrhofer. Compact Hardware Implementations of the SHA-3 Candidates ARIRANG, BLAKE, Grøstl, and Skein. Cryptology ePrint Archive, Report 2009/349, 2009. `http://eprint.iacr.org/`.

[122] A. Turing. Intelligent machinery, 1948.

[123] T. van Deursen and S. Radomirovic. Attacks on RFID Protocols. Cryptology ePrint Archive, Report 2008/310, 2008. `http://eprint.iacr.org/`.

[124] S. Vaudenay. On Privacy Models for RFID. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2007.

[125] Y. Watanabe, J. Shikata, and H. Imai. Equivalence between Semantic Security and Indistinguishability against Chosen Ciphertext Attacks. In

Y. Desmedt, editor, *PKC*, volume 2567 of *Lecture Notes in Computer Science*, pages 71–84. Springer, 2003.

[126] E. Wenger and M. Hutter. A Hardware Processor Supporting Elliptic Curve Cryptography for Less than 9 kGEs. In E. Prouff, editor, *CARDIS*, volume 7079 of *Lecture Notes in Computer Science*, pages 182–198. Springer, 2011.

Part II

Publications

List of Publications

International Conferences and Workshops

1. Jens Hermans and Roel Peeters. Private Yoking Proofs: Attacks, Models and new Provable Constructions. In Ingrid Verbauwhede, editor, *RFIDSec*, Lecture Notes in Computer Science. Springer, 2012. To appear.

   – See p. 181.

2. Roel Peeters and Jens Hermans. Wide Strong Private RFID Identification based on Zero-Knowledge, 2012. In submission.

   – See p. 153

3. Jens Hermans, Andreas Pashalidis, Frederik Vercauteren, and Bart Preneel. A New RFID Privacy Model. In Vijay Atluri and Claudia Diaz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 568–587. Springer, 2011.

   – See p. 125.

4. Junfeng Fan, Jens Hermans, and Frederik Vercauteren. On the Claimed Privacy of EC-RAC III. In Siddika Berna Ors Yalcin, editor, *RFIDSec*, volume 6370 of *Lecture Notes in Computer Science*, pages 66–74. Springer, 2010.

   – See p. 111.

5. Jens Hermans, Michael Schneider, Johannes Buchmann, Bart Preneel, and Frederik Vercauteren. Parallel Shortest Lattice Vector Enumeration on Graphics Cards. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 52–68. Springer, 2010.

   – See p. 87.

6. Jens Hermans, Frederik Vercauteren, and Bart Preneel. Speed Records for NTRU. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2010.

   – See p. 63.

7. Jens Hermans, Michael Schneider, Frederik Vercauteren, Johannes Buchmann, and Bart Preneel. Shortest Lattice Vector Enumeration on Graphics Cards. In *SHARCS*, Lausanne, CH, 2009.

Technical Reports

1. Jens Hermans. QR Factorization of Circulant Matrices. Cosic internal report, 2009.

Publication

Speed Records for NTRU

Publication Data

Jens Hermans, Frederik Vercauteren, and Bart Preneel. Speed Records for NTRU. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2010.

Contributions

- Principal author.

Speed Records for NTRU *

Jens Hermans †, Frederik Vercauteren ‡, and Bart Preneel

Katholieke Universiteit Leuven, ESAT/SCD-COSIC and IBBT
Kasteelpark Arenberg 10
B-3001 Leuven-Heverlee, Belgium
{jens.hermans,frederik.vercauteren,bart.preneel}@esat.kuleuven.be

Abstract. In this paper NTRUEncrypt is implemented for the first time on a GPU using the CUDA platform. As is shown, this operation lends itself perfectly for parallelization and performs extremely well compared to similar security levels for ECC and RSA giving speedups of around three to five orders of magnitude. The focus is on achieving a high throughput, in this case performing a large number of encryptions/decryptions in parallel. Using a modern GTX280 GPU a throughput of up to 200 000 encryptions per second can be reached at a security level of 256 bits. This gives a theoretical data throughput of 47.8 MB/s. Comparing this to a symmetric cipher (not a very common comparison), this is only around 20 times slower than a recent AES implementation on a GPU.

Keywords: NTRU encryption, Graphical Processing Unit, Parallelization, CUDA.

1 Introduction

Graphical Processing Units (GPUs) have long been used only for the rendering of games and other graphical applications. More recent GPUs are also used for general purpose parallel programming, using new programming models. A General Purpose GPU (GPGPU) contains a large number of processor cores (240 for the GTX280 [24]) that run at frequencies that are mostly lower than CPUs (1.2 GHz for the GTX280 GPU compared to 3.8 GHz for a recent Intel Pentium 4 [19]). Compared to a CPU a GPU provides a much larger computing

*This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

†Research assistant, sponsored by the Fund for Scientific Research - Flanders (FWO).

‡Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

power (several GFlops, or even TFlops for multiple GPUs) for specific parallel applications, because of the large number of cores. The recent change towards general scalar processor cores, that support 32- or 64-bit integer and bitwise operations, offers a new opportunity to implement cryptographic applications on GPUs.

There are several cryptographic ciphers that have a high level of parallelism, making them suitable for implementation on GPU. For performing a single encryption/decryption GPUs might not be very well suited: there is a latency compared to a CPU, because of the transfer of the data between main memory and GPU memory. In many applications the focus is not on the latency of a single cryptographic application, but on the throughput: one wants to perform a large number of encryptions/decryptions as fast as possible. In the case of a symmetric block cipher this will be the case when operating on a large block of data (using a suitable block cipher mode). Asymmetric ciphers are not often used in such a mode of operation, but more likely on servers that need to process many different secured connections where a large number of asymmetric cryptographic operations need to be performed. Currently cryptographic co-processors are used to speed up these operations, but a GPU might provide an alternative for these co-processors. An advantage is the fact that GPUs are almost by default present in modern computers and are also much underused. Another advantage is the flexibility: GPUs are easy to reprogram, making it an interesting co-processor to add to large computing farms. The large power consumption of the fastest GPUs is however a disadvantage, especially with a growing focus on the energy performance of data centers. One of the most likely uses of GPUs will be performing attacks on ciphers. GPUs have a very good computing power / price ratio, making them very economic for bulk computations. One can have around 200 GFlops (around 1 TFlops theoretically) for less than €500. In many attacks multiple cryptographic operations need to be performed, or at least part of these operations, so implementing and optimizing the original cryptographic operation on GPU is also of great use for attacks.

The choice for NTRUEncrypt (in short: NTRU) as the cryptographic cipher is less obvious: RSA [25] and ECC [7] are currently the respectively dominant and rising ciphers. NTRU has a large potential as a future cipher, given the very simple nature of its core operation: the convolution (compared to a modular exponentiation for RSA and repeated squaring/doubling for ECC). This simple operation makes it very suitable for embedded devices with limited computing power, but also for parallelization, since a convolution can be split up over several processors. NTRU also has a good (asymptotic) performance of $O(N^2)$ (or even $O(N \log N)$ using FFT), compared to, for example, $O(N^3)$ for RSA. So, NTRU is expected to outperform RSA and ECC at similar security levels,

and NTRU will also provide a good scalability for the future.

Because of these properties of NTRU, it was chosen as the cipher to be implemented on a GPU in this paper. For this paper the *ees1171ep1* parameter set is used, a high security ($k = 256$, the symmetric key size in bits) parameter set as claimed in [1]. Besides this parameter set a special version using product-form polynomials is also implemented. Product-form polynomials improve performance even further. Taking this high security level into account, NTRU performs very well when comparing with RSA (which would require a 15360 bit modulus) and ECC. For high througput applications a speedup of three to five orders of magnitude is reached compared to RSA and ECC. The GPU implementation reaches a throughput of up to 200 000 encryptions per second which is equivalent to a theoretical data throughput of 47.8 MB/s.

Organization

In Section 2 previous work on cryptography on GPUs and NTRU implementations is discussed. Next, a brief introduction is given to the NTRU cryptosystem in Section 3, especially on the parameter sets that have been proposed in the literature. In Section 4 the basics of GPGPU programming are explained, with a focus on the CUDA platform. This knowledge of NTRU and CUDA is combined to make an optimized GPU implementation of NTRU in Section 5. Finally the performance of the implementation is evaluated and compared to other implementations and other ciphers in Section 6.

2 Related Work

There is already much software available for GPUs, ranging from simple linear algebra packages to complex physical simulations. There has not been much development of cryptographic applications for the GPU, until recently when GPUs started supporting integer and bitwise operations. For example, AES was implemented on GPU [21] [15] [9], offering a maximum throughput of 831 MBytes/s (128 bit key, [21]).

RSA [25] has been implemented before the introduction of recent GPGPU platforms, using the OpenGL API [22] and more recently using modern platforms [27] [13], reaching up to 813 modular exponentiations (1024-bit integers) per second [27]. GPUs are also used to launch attacks, for example elliptic curve integer factoring [5] and brute force attacks, like for wireless networks [26].

There are no GPU implementations for NTRU. NTRU has however been implemented on a variety of platforms, like embedded devices, FPGAs [3] and custom hardware [2]. NTRU turns out to perform very well on devices with limited computing capabilities, given the simple nature of the convolution that is the central encryption/decryption operation. Compared to other modern cryptosystems like ECC, NTRU turns out to be very fast [20].

3 NTRUEncrypt

In this section the basics of NTRU are briefly introduced, based upon [11], to which we refer for further, more complete, information.

Let $\mathbb{Z}$ denote the ring of integers and $\mathbb{Z}_q$ the integers modulo q. NTRUEncrypt is a public-key cryptosystem that works with the polynomial ring $P(N) = \mathbb{Z}[X]/(X^N - 1)$ (and $P_q(N) = \mathbb{Z}_q[X]/(X^N - 1)$), where N is a positive prime. A vector from $\mathbb{Z}^N$ (resp. $\mathbb{Z}_q^N$) can be associated with a polynomial by $f = (f_0, f_1, \ldots, f_{N-1}) = \sum_{i=0}^{N-1} f_i X^i$

The multiplication of two polynomials $h = f \star g$ is defined as the cyclic convolution of their coefficients:

$$h_k = (f \star g)_k = \sum_{i+j \equiv k \mod N} f_i \cdot g_j \ (0 \leq k < N) \tag{1}$$

which is the ordinary polynomial multiplication modulo $X^N - 1$.

The polynomials used in NTRU are selected from several polynomial sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r$ and $\mathcal{L}_m$. First the basic operations (key creation, encryption and decryption) of NTRU are introduced and afterwards, in Section 3.1, the structure of the polynomials and the parameter sets are discussed.

Key Creation

The private key is a polynomial f, chosen at random from the set $\mathcal{L}_f$. Another polynomial $g \in \mathcal{L}_g$ is also chosen at random, but is not needed anymore after key generation. From these polynomials the public key h can be computed as

$$h = p \star f_q^{-1} \star g \mod q \tag{2}$$

where f_q^{-1} is the inverse of f in $P_q(N)$ and p is a polynomial (usually 3 or $X + 2$).

The polynomials f and g generally have small coefficients, while h has large coefficients.

Encryption

The message $m \in \mathcal{L}_m$ can be encrypted by choosing a random polynomial $r \in \mathcal{L}_r$ as a blinding factor and computing the ciphertext as

$$e = r \star h + m \mod q. \tag{3}$$

In practical schemes the message is padded with random bits and masked. For this paper, these steps are ignored, and only the computation of $r \star h + m \mod q$ is considered.

Decryption

Decryption can be done by convolving the ciphertext e with the private key f

$$a \equiv e \star f \equiv p \star r \star g + m \star f \mod q \tag{4}$$

and next convolving by $f_p^{-1} \mod p$. By a careful choice of f it can be assured that $f_p^{-1} = 1$, so only a reduction mod p is needed.

One of the problems NTRU faces are decryption failures: the first step of the decryption only computes $a \mod q$ and not a. The problem is that knowing $a \mod q$ is not enough to know $a \mod p$. The problem of decryption failures has been studied extensively in [18]. In this paper it suffices to pick the coefficients of a from $(-q/2, q/2]$ and assume the probability of decryption failures is negligibly low.

3.1 Parameter Sets

The parameter N must always be chosen to be prime, since composites allows the problem to be decomposed [14]. The parameter q is mostly chosen as a power of 2, to ease the computations modulo q. The parameter p must be relatively prime to q, but it is not necessary that p is an integer, it can be a polynomial. Popular choices for p are 3 and $X + 2$.

Besides the parameters N, p, q there are the sets of polynomials $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_m, \mathcal{L}_r$ that have to be defined. The message space $\mathcal{L}_m$ is defined as $P_p(N)$, since the message is obtained during the decryption after reducing modulo p.

The other sets of polynomials are chosen as ternary (for $p = 3$) or binary (for $p = X + 2$) polynomials.

Ternary Polynomials

Define $\mathcal{L}(d_x, d_y)$ as the set of all ternary polynomials that have d_x coefficients set to 1 and d_y coefficients set to -1 (all other coefficients are 0).

One of the most natural choices for the polynomial sets is

$$\mathcal{L}_f = \{1 + p \star F : F \in \mathcal{L}(d_f, d_f)\} \ , \ \mathcal{L}_r = \mathcal{L}(d_r, d_r) \ , \ \mathcal{L}_g = \mathcal{L}(d_g, d_g)$$

which is also used in the most recent standards draft [1]. The choice of $\mathcal{L}_f$ as $1 + p \star F$ guarantees that $f_p^{-1} = 1$.

For ternary polynomials p is set to 3.

Binary Polynomials

Binary polynomials offer an alternative for ternary polynomials and are much easier to implement in hardware and software. A disadvantage is that binary polynomials are by definition unbalanced, so $f(1) \neq 0$. As a consequence information on m, namely $m(1)$, leaks.

In [8] the following parameters are used:

$$\mathcal{L}_f = \{1 + p \star F : F \in \mathcal{L}(d_f, d_f)\} \ , \ \mathcal{L}_r = \mathcal{L}(d_r, 0) \ , \ \mathcal{L}_g = \mathcal{L}(d_g, 0)$$

Product-form Polynomials

The central operation when encrypting is a convolution with a binary/ternary polynomial. The number of non-zero elements in $r \in \mathcal{L}_r$ is crucial for the performance of the encryption operation. A smaller number of non-zero elements will make the convolution faster (and lower memory usage, depending on the storage strategy) but will also degrade the security. By taking

$$\mathcal{L}_r \quad = \quad \{r_1 \star r_2 + r_3 : r_1 \in \mathcal{L}_{r_1}, r_2 \in \mathcal{L}_{r_2}, r_3 \in \mathcal{L}_{r_3}\}$$

with $d_{r_1}, d_{r_2}, d_{r_3} \ll d_r$ the convolution is still secure, since $r_1 \star r_2 + r_3$ still contains roughly the same amount of randomness as a single random r [16]. For our implementation $d_{r_1} = d_{r_2} = d_{r_3} = 5$, so each polynomial r_i has 10 non-zero coordinates. The performance is however increased drastically. The convolution $t = r \star h \mod q$ can be computed in several steps as in [3]:

$$t_1 \leftarrow r_2 \star h \quad ; \quad t_2 \leftarrow r_1 \star t_1 \quad ; \quad t_3 \leftarrow r_3 \star h \quad ; \quad t \leftarrow t_2 + t_3 \mod q \qquad (5)$$

Since each of r_1, r_2, r_3 have a low number of non-zero elements, the convolutions in (5) are much faster, requiring less additions than $r \star h$. Another advantage is the lower storage requirement.

4 GPU Programming

4.1 The CUDA Platform

The CUDA programming guide [23] explains in detail all aspects of the platform and programming model and was used as a basis for the following sections. The GTX280 that was used for this paper is a GPU that belongs to the range of Tesla Architecture GPUs from Nvidia. The Tesla architecture is based upon an array of multiprocessors (30 for the GTX280) that each contain 8 scalar processors. A multiprocessor is operated as a SIMT-processor (Single-Instruction, Multiple-Thread): a single instruction uploaded to the GPU causes multiple threads to perform the same scalar operation (on different data). The CUDA programming model from Nvidia, that is used to program their GPUs, provides a C-like language to program the GPU.

Programming Model

As stated above, all programming is done using scalar operations: one needs to program a single thread which will then be executed in multitude on the GPU. Threads are grouped into blocks. All blocks together form a 'grid' of blocks. Threads within the same block can use shared memory. Both threads and blocks can be addressed in a multi-dimensional way. All scheduling of instructions (threads) on the multiprocessors is hidden from the programmer and is done on-chip. Threads are scheduled in *warps* of 32 threads. For optimal performance divergent branching inside the same half-warp (16 threads) must be avoided: each thread in a half-warp must execute the same instruction, otherwise the execution will be serialized. If divergent branching occurs, one possible strategy is to ensure that the thread ID for which divergence occurs coincides with a change of half-warp.

Memory

A multiprocessor contains fast on-chip memory in the form of registers, shared memory and caches. Off-chip memory is also available in the form of global

memory and specialized texture and constant memory. The global memory is not cached. The GTX280 provides 1GB of off-chip memory.

Each of the memory types has specific features and caveats [1]:

- Global memory: as the global memory is off-chip there is a large performance hit (hundreds of cycles). Another issue is that multiple threads might access different global memory addresses at the same time, which creates a bottle-neck and forces the scheduler to stop the execution of the block until all memory is loaded. It is recommended to run a large number of blocks, to ensure the scheduler can keep the multiprocessors busy, while memory loading takes place. One way to avoid such large performance penalties are coalesced memory reads, in which all threads from a half-warp access either the same address or a block of consecutive addresses. In the case of loading a single address the total cost is only one memory load.
- Registers: care has to be taken to limit the number of registers per thread as the registers are shared among all threads and blocks running on the same multiprocessor.
- Shared memory: shared memory is stored in banks, such that consecutive 32 bits are stored in consecutive banks. When accessing shared memory one needs to ensure that threads within the same warp access different banks, to avoid 'bank conflicts'. Bank conflicts result in serialization of the execution.
- Constant memory: the advantage of using constant memory is the presence of a special read-only cache, which allows for fast access times.

Instructions

Almost all operations that are available in C can be used in CUDA. CUDA only uses 32-bit (`int`, `float`) and 64-bit variables (`long`, `double`) for arithmetic, other types are converted first. In this paper, we will refer to 32-bit integers as '`int`' (or just 'integer') and to 64-bit integers as a '`long`'. Integer addition and bitwise operations take 4 clock cycles. 32-bit integer multiplication takes 16 cycles. Integer division and modulo are expensive and should be avoided.

[1] Texture memory is not used in this paper, so details have been omitted.

5 The Implementation

For the implementation the *ees1171ep1* parameter set from [1] is used. This parameter set (with ternary polynomials and $N = 1171, p = 3, q = 2048 = 2^{11}, d_r = 106$) is one of the three strongest from the draft standard. Considering the relatively young age of NTRU and recent attacks (e.g. [17]), it is better to be rather conservative in the parameter choices and take one of the strong parameter sets.

Two implementations were made: one using the default ternary polynomials, the other using product-form ternary polynomials. In the last case $d_{r_1} = d_{r_2} = d_{r_3} = 5$.

The generation of random data (needed for encryption) is performed by the CPU, although parallel implementations exist for CUDA. There are several reasons for this choice: first of all it is the goal of this paper to compare the central NTRU operation, the convolution, and not to compare choices of random number generators. By computing the random numbers beforehand on CPU, any influence of the choice of the random generator is excluded. Second, one might consider an attack strategy in which the opponent would explicitly choose r, instead of using random numbers. Another advantage of performing the generation of r on CPU is exploiting the parallel computation by using both CPU and GPU.

5.1 Operations

Both parallel encryption (two variants) and parallel decryption are implemented on CUDA. The superscript i in m^i denotes the i-th message that is used in the parallel computation. The operations are defined as follows:

- **Encryption:** given $r^i \in \mathcal{L}_r$, h^i and $m^i \in \mathcal{L}_m$ (for $i \in [0, P)$, with P the number of parallel encryption operations) the kernel computes $e^i = r^i \star h^i + m^i \mod q$. Two strategies for the public key are considered: one which uses the same public key for all encryptions ($\forall i : h_i = h$) and one with different public keys for every operation.
- **Decryption:** given e^i and f, compute m^i. The private key is the same for all decryptions.

Key generation was not implemented, although situations exist where one would like to generate multiple keys in parallel.

For encryption both ordinary and product-form ternary polynomials are used as r.

The decryption operation can be written as

$$e \star f \equiv e \star (1 + p \star F) \equiv e + (e \star F) + (e \star F) \ll 1 \mod q \tag{6}$$

where "$\ll$" is a left bit shift. Besides some extra scalar operations for each coefficient, one can reuse the encryption algorithm. In the next sections only encryption is discussed. The results section only includes results for the case that F is an ordinary ternary polynomial. Because there was no performance difference compared to encryption, decryption was not implemented for product-form polynomials.

5.2 Memory Usage - Bit Packing

Since all data must be transferred from the main computer memory to the GPU (device) memory, it is in the best interest to limit the amount of memory used.

One standard technique is bit packing. The ternary coefficients of r can be encoded as 2 bits, of which 32 can be packed into a 64-bit `long`. The coefficients of h are each 11 bit long, allowing for up to 5 coefficients to be stored in a `long`. We however pack only 4 elements of h in a long. The extra unused bits come in handy when performing an addition on the entire long, so that the overflow does not corrupt one of the packed values stored higher in the bit array. Note that although the polynomial m also has ternary coefficients we choose to store it using 11 bits per element. This way, the result of the encryption e (which is mod q) can be written in the same space as m, which results in a smaller memory usage. In total 623 `long`'s are required to store h, m and r.

For the implementation with product-form polynomials the values of r_1, r_2 and r_3 can be stored in a different way. Instead of encoding each ternary coefficient as two bits, the indices of the non-zero coefficients are stored, as in [3]. Since each index is in $[0, N-1]$, $\lceil \log_2(N) \rceil = 11$ bits are needed to store each index. These indices are again packed, but not aligned to 16 bit multiples, since the access is sequential (see further). The memory consumption is only lowered moderately to 592 `long`s, but the new structure of the convolution has a large impact on the construction of the loops and thus the performance.

Since multiple encryption/decryption operations are performed, multiple messages m and blinds r need to be uploaded to the device. All variables are stored in one large array of `long`'s, e.g. a single m^b is packed to 293 `long`s, with the total array being $293 \times P$ `long`'s. Note that the time for bit packing

the data on CPU is not included in the timing results and that all host-memory is page-locked.

In the next sections and the algorithms in Appendix A, we use the notation $x_{\text{packed},i}$ to refer to the `long` containing the i-th element of the x polynomial (which is denoted as x_i). $P(i)$ is used to denote the index of the `long` that contains x_i. When there is a reference to x_i in the pseudo-code, the index calculation and decoding are implicit.

5.3 Encoding

The coefficients of h are encoded as 11 bit integers, in the range $[0, q-1]$. The blind r, consisting of ternary coefficients, is encoded by mapping $\{0, 1, -1\}$ to 2-bit values (which can be chosen arbitrarily). The message m also consists of ternary coefficients, but for efficient computation, these are loaded in the memory space that will contain the result e. Because of this, the ternary coefficients are stored as 11-bit values in two's complement (e.g. $(-1)_3 = 2^{11} - 1$).

5.4 Blocks, Threads and Loop Nesting

Parallelism is present at two levels: at the level of a single encryption, which is split over multiple threads, and at the level of the parallel encryptions, which are split over the blocks. When performing a single encryption, one needs to access all elements r_i^b, h_j^b and e_k^b. Each block (block index denoted with the superscript b) is responsible for doing a single encryption/decryption operation. To make storing e_k as efficient as possible, each thread is responsible of computing 4 coefficients of e, which implies that each thread writes only one `long`.

For the normal ternary polynomials, the algorithm executed by each thread is presented in Algorithm 8. There is an implicit 'outer loop' that iterates over k (the parallel threads). The middle loop (over i) selects the element from r^b and then uses simple branching and addition (no multiplications).

Algorithm 6 shows the algorithm for the product-form ternary polynomials. The implicit outer loop is the same, but the computation inside is completely different. The computation of $r_2 \star h$ is split over all threads and the results are stored (packed) in shared memory. Unlike the other convolutions in Algorithm 6, all threads need all indices of $r_2 \star h$ and not just the $k \ldots k+3$-th coefficients.

Since r_1, r_2 and r_3 are stored using indices, the convolution algorithm is different from that used for ordinary polynomials. Algorithm 7 describes part of such a convolution. Again, only 4 coefficients of the result are computed, which matches the division of the convolution among the threads.

5.5 Memory Access

Since the convolutions are very simple operations, using only addition and some index-keeping operations, the memory access will be the bottleneck. One of the solutions is to explicitly cache the elements of r and h in registers (the GPU does not have a cache for global memory). Especially for r this turns out to be a good strategy, since each `long` contains 32 coefficients, thereby reducing the number of accesses to global memory with a factor 32. For h no significant benefits were observed, so the caching was omitted. The main reason is that the packed coefficients of h are less often accessed (many of the r_i are zero) and they are accessed in a more or less random pattern, so caching them for more than one iteration (over i in Algorithm 8) makes no sense. There is however a benefit from executing multiple threads in parallel: when thread t accesses h_j, thread $t+1$ will at the same time access h_{j+4}, which is always stored in the next `long`. This means that memory access is coalesced, although bad alignment of the memory blocks will prevent the full speedup.

For product-form polynomials the number of memory accesses is much lower: the space used to store r is smaller. As r_1,r_2 and r_3 are accessed only once, this means a drop in memory access from 296 to 48 bytes per block. The number of accesses to h also goes down: only the convolutions $r_2 \star h$ and $r_3 \star h$ need access to h. r_2 and r_3 each have 10 non-zero coefficients, giving a total of 20 accesses to h for each element in the result, so $20 \times 1171 = 23420$ `long`s per block, compared to $2Nd_r = 248252$ `long`s per block for ordinary polynomials.

Note that the access to e is coalesced, since each thread accesses a consecutive `long`.

5.6 Branching

Almost no divergent branching occurs during the execution of the algorithms. In the case of normal polynomials branching on r_i is not divergent, as each thread has the same value for i. The only divergent branches are for the modulo computation. There is one aspect when using product-form polynomials in Algorithm 6 that might cause a performance hit: the thread synchronization.

Since the intermediate result t_{shared} is shared among all threads, all threads should wait for the completion of that computation.

6 Results

In this section the results of the GPU implementations are compared to a simple CPU implementation in C and other implementations found in the literature. The CPU tests were performed on an Intel Core2 Extreme CPU, running at 3.00GHz. This processor has four cores, but only one of these cores is used as the CPU implementation is not parallel. The GPU simulations were performed on a GTX280. To verify that all implementations were correct, the output was verified (with success) against a reference implementation in Magma [6].

Table 1 shows the results expressed as milliseconds per operation (or operations per second). Results for different h_i are, obviously, only available for the GPU when doing multiple (20000) operations in parallel. The times in Table 1 are the minimal times over 10 identical experiments. All results are expressed as wall clock time, since this is the only way to be able to compare CPU and GPU. Taking the minimum time ensures that clearing of the cache or context switches do not bias the results. Clearing of the cache and context switches depend heavily on the environment in which the program is used, so it would not be fair to include these in the measurements. Overall, the GPU times had a small variance, so the difference between average time and minimal time was negligible. The time for copying data from main to GPU memory is included in the GPU performance figures.

The CPU implementation does not use any optimizations like bit packing and just consists of a few nested loops. The CPU implementation only performs one single encryption/decryption. Despite the fact that the CPU implementation is not optimized, we use it as a rough basis for comparison for the GPU version. The available performance results for previous implementations are for different (less secure) parameter sets, which makes it very hard to compare.

From Table 1 it is clear that encryption and decryption have roughly the same performance: the extra element-wise operations for decryption do not take much time. This is also the reason that decryption was not implemented separately for product-form ternary polynomials, since it would show the same performance. Encryption with the same h is slightly faster than using different h^i, although an explanation for this has not been found [2].

[2] The opposite result was expected. As h was not stored in constant memory, there should be no benefit from caching.

Figure 1 shows the subsequent gain in performance when increasing the number of parallel encryptions (for ordinary polynomials). Around 2^{11} encryptions the GPU approaches its maximum performance, larger numbers of parallel encryptions yield only a slight improvement in the number of operations per second.

Table 1 shows that for all implementations, product-form polynomials are much faster, as expected by the lower number of memory accesses in Section 5.5. The performance increases by almost a factor 10 compared to ordinary polynomials. Again a small difference is observed between encryption with the same and different h^i.

Table 2 compares the CPU and GPU implementations with previous work on NTRU and to some RSA and ECC implementations. A note of caution is due, since the previous NTRU implementations use much lower security parameters and because the platforms that are used are totally different. Also note that the amount of data encrypted per operation is different. As a very rough extrapolation to convert the results for the other NTRU implementations to the security level of our implementation one can use the $O(N^2)$ asymptotic performance of NTRU. This drastically lowers the performance measures for the other NTRU implementations, ignoring even the increase of q and d_r. For applications with a focus on high throughput (many op/s), the CUDA implementation for product-form polynomials outperforms all other NTRU implementations (taking the higher security parameters and amount of data into account). The implementation with product-form polynomials gives a speed of more than 200 000 encryptions per second or 41.8 MByte/s. For applications that need to perform a small number of encryptions with low latency, the parallelization of CUDA does not give much speedup compared to the CPU. However, when comparing NTRU with RSA and ECC, the speedup is large: up to 1300 times faster than 2048-bit RSA and 117 times faster than ECC NIST-224 when comparing the number of encryptions per second (or up to 1113 times faster than 2048-bit RSA when comparing the data throughput). In addition, the security level of NTRU is much higher: when extrapolating to RSA and ECC with $k = 256$ bit security, this would add an extra factor of around 10 for ECC and around 400 for RSA (assuming $O(N^3)$ operations for RSA and ECC, where N is the length of a message block). So, in this extrapolation, NTRU has a speedup of five orders of magnitude compared to RSA and three orders of magnitude compared to ECC. The results listed for RSA encryption on CPU are operations with a small public key ($e = 17$), which allows for further optimization that has not been done for the RSA GPU implementation.

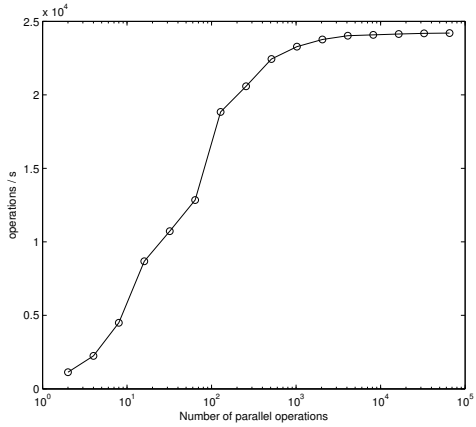


Figure 1: NTRU encryption operations per second using ordinary polynomials and the same h ($N = 1171, q = 2048, p = 3$).

7 Conclusion

In this paper NTRU encryption/decryption was implemented for the first time on GPU. Several design choices, such as the NTRU parameters sets, are compared. The exact implementation is analysed in detail against the CUDA platform, explaining the impact of every choice by looking at the underlying effects on branching, memory access, blocks & threads... Although the programming is done in C, the CUDA model has its own specific ins and outs that take some time to learn, making a good implementation not very straightforward.

Many external factors, like power consumption, cost, reprogrammability, context (latency vs throughput), space... besides the speed of the cipher influence the choice of platform. In areas in which reprogrammability, cost and throughput are important and power consumption is of lesser importance, a GPU implementation is a very good option.

For 2^{16} encryptions a peak performance of around 218 000 encryptions/s (or 4.58×10^{-6} s/encryption) is reached, using product-form polynomials. This corresponds to a theoretical data throughput of 47.8 MB/s. The GPU performs at its best when performing a large number of parallel NTRU operations. Parallel NTRU implementations could serve well on servers processing many secured connections or in various attack strategies in which many (partial)

encryption operations are needed. A single NTRU operation on GPU is still faster than a (simple) CPU implementation, but the speedup is limited. Even then a GPU might be interesting to simply move load off the CPU.

Comparing NTRU to other cryptosystems like RSA and ECC shows that NTRU, at a high security level, is much faster than RSA (around five orders of magnitude) and ECC (around three orders of magnitude). Even when only performing a single operation NTRU is still faster by around a factor of 35 for 2048 bit RSA and 3 for ECC NIST-244. Because of the ways NTRU can be parallelized, NTRU also clearly outperforms RSA and ECC for high-throughput applications. So, both for low-latency (single operation) and high-throughput (multiple operations) applications NTRU on GPU outperforms RSA and ECC.

References

[1] 1363 Working Group of the C/MSC Committee. IEEE P1363.1 Standard Specification for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, 2009. Available at `http://grouper.ieee.org/groups/1363/`.

[2] A. C. Atici, L. Batina, J. Fan, I. Verbauwhede, and S. B. Örs. Low-cost implementations of NTRU for pervasive security. In *ASAP*, pages 79–84. IEEE Computer Society, 2008.

[3] D. V. Bailey, D. Coffin, A. J. Elbirt, J. H. Silverman, and A. D. Woodbury. NTRU in Constrained Devices. In Çetin Kaya Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 262–272. Springer, 2001.

[4] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for Key Management, NIST Special Publication 800-57, 2007.

[5] D. J. Bernstein, H.-C. Chen, M.-S. Chen, C.-M. Cheng, C.-H. Hsiao, T. Lange, Z.-C. Lin, and B.-Y. Yang. The Billion-Mulmod-Per-Second PC. In *SHARCS*, pages 131–144, 2009.

[6] W. Bosma, J. Cannon, and C. Playoust. The Magma Algebra System I: The User Language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.

[7] H. Cohen and G. Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, volume 34 of *Discrete Mathematics and Its Applications*. Chapman & Hall/CRC, 2005.

[8] Consortium for Efficient Embedded Security. Efficient embedded security standards 1: Implementation aspects of NTRU and NSS, Version 1. 2002.

[9] D. L. Cook, J. Ioannidis, A. D. Keromytis, and J. Luck. CryptoGraphics: Secret Key Cryptography Using Graphics Cards. In A. Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 334–350. Springer, 2005.

[10] W. Dai. Crypto++: benchmarks. `http://www.cryptopp.com/benchmarks.html`.

[11] European Network of Excellence in Cryptology ECRYPT. Lightweight Asymmetric Cryptography and Alternatives to RSA. 2005.

[12] European Network of Excellence in Cryptology ECRYPT. ECRYPT Benchmarking of Asymmetric Systems (eBATS). `http://www.ecrypt.eu.org/ebats/`, 2007.

[13] S. Fleissner. GPU-Accelerated Montgomery Exponentiation. In Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, editors, *International Conference on Computational Science (1)*, volume 4487 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 2007.

[14] C. Gentry. Key Recovery and Message Attacks on NTRU-Composite. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2001.

[15] O. Harrison and J. Waldron. AES Encryption Implementation and Analysis on Commodity Graphics Processing Units. In P. Paillier and I. Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 209–226. Springer, 2007.

[16] J. Hoffstein and J. H. Silverman. Random small Hamming weight products with applications to cryptography. *Discrete Applied Mathematics*, 130(1):37–49, 2003.

[17] N. Howgrave-Graham. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In A. Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007.

[18] N. Howgrave-Graham, P. Q. Nguyen, D. Pointcheval, J. Proos, J. H. Silverman, A. Singer, and W. Whyte. The Impact of Decryption Failures on the Security of NTRU Encryption. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. Springer, 2003.

[19] Intel. Intel Pentium 4 - SL8Q9 Datasheet, 2008.

[20] P. Karu and J. Loikkanen. Practical Comparison of Fast Public-key Cryptosystems. In *Telecommunications Software and Multimedia Lab. at Helsinki Univ. of Technology, Seminar on Network Security*, 2001.

[21] S. Manavski. CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography. In *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*, pages 65 –68, nov. 2007.

[22] A. Moss, D. Page, and N. P. Smart. Toward Acceleration of RSA Using 3D Graphics Hardware. In S. D. Galbraith, editor, *IMA Int. Conf.*, volume 4887 of *Lecture Notes in Computer Science*, pages 364–383. Springer, 2007.

[23] Nvidia. Compute Unified Device Architecture Programming Guide, 2007.

[24] Nvidia. GeForce GTX280 - GeForce GTX 200 GPU Datasheet, 2008.

[25] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[26] M. Settings. Password crackers see bigger picture. *Network Security*, 2007(12):20, 2007.

[27] R. Szerwinski and T. Güneysu. Exploiting the Power of GPUs for Asymmetric Cryptography. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 79–99. Springer, 2008.

A Code Listings

Algorithm 6: Pseudo-code for a single NTRU encryption (product-form polynomials)

1: $b \leftarrow$ blockID
2: $k \leftarrow 4 * \text{threadID}$
3: Allocate $e_{\text{temp}}[0 \dots 3] \leftarrow 0$
4: Allocate $t_{\text{shared}}[0 \dots N-1]$
5: $t_{\text{shared}}[k \dots k+3] \leftarrow Convolve(h^b, r^b_{2,+}, r^b_{2,-}, k, t_{\text{shared}}[k \dots k+3])$
6: Synchronize threads
7: $e_{\text{temp}}[0 \dots 3] \leftarrow Convolve(t_{\text{shared}}, r^b_{1,+}, r^b_{1,-}, k, e_{\text{temp}}[0 \dots 3])$
8: $e_{\text{temp}}[0 \dots 3] \leftarrow Convolve(h^b, r^b_{3,+}, r^b_{3,-}, k, e_{\text{temp}}[0 \dots 3])$
9: **for** $l = 0$ to 3 **do**
10: $\quad e^b_{k+l} \leftarrow m^b_{k+l} + e_{\text{temp}}[l] \mod q$
11: **end for**

Algorithm 7: Pseudo-code for a single product-form convolution. $Convolve(h, r^+, r^-, k, t)$

Require: h: an ordinary polynomial,
r^+, r^-: the positions of the $+1$ and -1 elements in the polynomial r,
t: result of the convolution,
k: offset of the results that need to be calculated.
Ensure: $t[k \dots k+3] = \{h \star r\}_{k \dots k+3}$
1: $k \leftarrow 4 * \text{threadID}$
2: **for** $l = 0$ to $d_{r-1} - 1$ **do**
3: $\quad i \leftarrow r^+_l$
4: $\quad$ **for** $\delta_k = 0$ to 3 **do**
5: $\quad\quad t[k+\delta_k] \leftarrow t[k+\delta_k] + h_{(k+\delta_k - i \mod N)}$
6: $\quad$ **end for**
7: **end for**
8: **for** $l = 0$ to $d_{r-1} - 1$ **do**
9: $\quad i \leftarrow r^-_l$
10: $\quad$ **for** $\delta_k = 0$ to 3 **do**
11: $\quad\quad t[k+\delta_k] \leftarrow t[k+\delta_k] - h_{(k+\delta_k - i \mod N)}$
12: $\quad$ **end for**
13: **end for**
14: **return** $t[k \dots k+3] \mod q$

Algorithm 8: Pseudo-code for a single NTRU encryption (ordinary polynomials)

$b \leftarrow$ blockID
$k \leftarrow 4 * \text{threadID}$
Allocate $e_{\text{temp}}[0 \dots 3] \leftarrow 0$
for $i = 0$ to 10 **do**
    for $l = 0$ to 3 **do**
        if $P(i) \neq P(i-1)$ **then**
            $r_{\text{cache}} \leftarrow r^b_{\text{packed},i}$
        end if
        $r_{\text{elem}} \leftarrow r_i$ (from r_{cache})
        $j \leftarrow k + l - i \mod N$
        if $r_{\text{elem}} = 1$ **then**
            $e_{\text{temp}}[l] \leftarrow e_{\text{temp}}[l] + h^b_j$
        end if
        if $r_{\text{elem}} = -1$ **then**
            $e_{\text{temp}}[l] \leftarrow e_{\text{temp}}[l] - h^b_j$
        end if
    end for
end for
for $l = 0$ to 3 **do**
    $e^b_{k+l} \leftarrow m^b_{k+l} + e_{\text{temp}}[l] \mod q$
end for

Algorithm 9: Pseudo-code for a single NTRU Decryption

Require: F: the private key ($f = 1 + p \star F$)
e: the encrypted message
$k \leftarrow 4 * \text{threadID}$
Execute Algorithm 8, taking $m = 0$, $r = F$ and $h = e$ and obtaining $t[0 \dots 3]$.
for $l = 0$ to 3 **do**
    $t[l] \leftarrow t[l] + (t[l] \ll 1) + e_{k+l}$
    $tmp \leftarrow t[l] - p * ((p^{-1} \mod q) * t[l] \gg \log_2 q)$
    $(t[l] > q) \Rightarrow (tmp \leftarrow tmp + 1)$
    $t[l] \leftarrow tmp$
end for

	Encryption (different h^i)		Encryption (same h^i)		Decryption	
	μs/op	op/s	μs/op	op/s	μs/op	op/s
Ordinary						
CPU	-	-	$10.5 \cdot 10^3$	(95)	$10.5 \cdot 10^3$	(95)
GPU, 1 op.	-	-	$1.75 \cdot 10^3$	-	$1.87 \cdot 10^3$	-
GPU, 20000 ops	41.3	24 213	40.0	25 025	41.1	24 331
Product-form						
CPU	-	-	$0.31 \cdot 10^3$	(3225.8)	-	-
GPU, 1 op.	-	-	$0.16 \cdot 10^3$	-	-	-
GPU, $\sim 2^{16}$ ops	4.58	218 204	4.51	221 845	-	-

Table 1: Performance comparison of NTRU on an Intel Core2 CPU and a Nvidia GTX280 GPU using ordinary and product-form ternary polynomials ($N = 1171, q = 2048, p = 3$).

Platform		(N, q, p)	**Enc/s**	**Dec/s**	bit/op
FPGA [3]	Xilinx Virtex 1000EFG860 @ 50 MHz	$(251, 128, X + 2)$	$193 \cdot 10^3$	-	251
Palm [3]	Dragonball @ 20 MHz (C)	Product form	21	11	
Palm [3]	Dragonball @ 20 MHz (ASM)	$(k < 80)$	30	16	
ARM C [3]	ARM7TDMI @ 37 MHz		307	148	
FPGA [2]	Xilinx Virtex 1000EFG860 @ 500kHz	$(167, 128, 3)$ $(k \ll 80)$	18	8.4	250
C	Intel Core2 Extreme @ 3.00GHz	$(1171, 2048, 3)$	95	95	1756
CUDA	GTX280 (1 op)	$(k = 256$ [1])	571	546	
CUDA	GTX280 (20000 ops)		$24 \cdot 10^3$	$24 \cdot 10^3$	
C	Intel Core2 Extreme @ 3.00GHz	$(1171, 2048, 3)$	$3.22 \cdot 10^3$	-	1756
CUDA	GTX280 (1 op)	Product form	$6.25 \cdot 10^3$	-	
CUDA	GTX280 ($\sim 2^{16}$ ops)	$(k = 256$ [1])	$218 \cdot 10^3$	-	
RSA comparison					
CUDA [27]	Nvidia 8800GTS	1024 bit		813	1024
C++ [10]	Intel Core2 @ 1.83GHz	$(k = 80$ [4])	$(14 \cdot 10^3)$	657	1024
CUDA [27]	Nvidia 8800GTS	2048 bit		104	2048
C++ [10]	Intel Core2 @ 1.83GHz	$(k = 112$ [4])	$(6.66 \cdot 10^3)$	168	2048
ECC comparison					
CUDA [27]	Nvidia 8800GTS (PointMul)	ECC NIST-224		$1.41 \cdot 10^3$	
C [12]	Intel Core2 @ 1.83 GHz (ECDSA)	$(k = 112$ [4])		$1.86 \cdot 10^3$	

Table 2: Comparison of several NTRU, RSA and ECC implementations. The chosen parameter set and claimed security level (k) is listed for all ciphers. The number of operations per second is listed, together with the amount of data encrypted/decrypted per operation (excluding all padding, headers...)

Publication

Parallel Shortest Lattice Vector Enumeration on Graphics Cards

Publication Data

Jens Hermans, Michael Schneider, Johannes Buchmann, Bart Preneel, and Frederik Vercauteren. Parallel Shortest Lattice Vector Enumeration on Graphics Cards. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 52–68. Springer, 2010.

Contributions

- Principal author together with Michael Schneider.

Parallel Shortest Lattice Vector Enumeration on Graphics Cards*

Jens Hermans †[1], Michael Schneider[2], Johannes Buchmann[2], Frederik Vercauteren ‡[1], and Bart Preneel[1]

[1] Katholieke Universiteit Leuven - ESAT/SCD-COSIC and IBBT
{Jens.Hermans,Frederik.Vercauteren,Bart.Preneel}@esat.kuleuven.be

[2] Technische Universität Darmstadt
{mischnei,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. In this paper we present an algorithm for parallel exhaustive search for short vectors in lattices. This algorithm can be applied to a wide range of parallel computing systems. To illustrate the algorithm, it was implemented on graphics cards using CUDA, a programming framework for NVIDIA graphics cards. We gain large speedups compared to previous serial CPU implementations. Our implementation is almost 5 times faster in high lattice dimensions.

Exhaustive search is one of the main building blocks for lattice basis reduction in cryptanalysis. Our work results in an advance in practical lattice reduction.

Keywords: Lattice reduction, ENUM, parallelization, graphics cards, CUDA, exhaustive search

1 Introduction

Lattice-based cryptosystems are assumed to be secure against quantum computer attacks. Therefore these systems are promising alternatives to

*The work described in this report has in part been supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

†Research assistant, sponsored by the Fund for Scientific Research - Flanders (FWO).

‡Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

factoring or discrete logarithm based systems. The security of lattice-based schemes is based on the hardness of special lattice problems. Lattice basis reduction helps to determine the actual hardness of those problems in practice. In the past few years there has been increased attention to exhaustive search algorithms for lattices, especially to implementation aspects. In this paper we consider parallelization and special hardware for the exhaustive search.

Lattice reduction is the search for short and orthogonal vectors in a lattice. The algorithm used for lattice reduction in practice today is the BKZ algorithm of Schnorr and Euchner [SE91]. It consists of two main parts, namely an exhaustive search ('enumeration') for shortest, non-zero vectors in lower dimensions and the LLL algorithm [LLL82] for the search for short (not *shortest*) vectors in high dimensions. The BKZ algorithm is parameterized by a blocksize parameter β, which determines the blocksize of the exhaustive search algorithm inside BKZ.

Algorithms for exhaustive search were presented by Kannan [Kan83] and by Fincke and Pohst [FP83]. Therefore, the enumeration is sometimes referred to as KFP-algorithm. Kannan's algorithm runs in $2^{\mathcal{O}(n \log n)}$ time, where n denotes the lattice dimension. Schnorr and Euchner presented a variant of the KFP exhaustive search, which is called ENUM [SE91]. Roughly speaking, enumeration algorithms perform a depth first search in a search tree that contains all lattice vectors in a certain search space, i.e., all vectors of Euclidean norm less than a specified bound. The main challenge is to determine which branches of the tree can be cut off to speed up the exhaustive search. Enumeration is always executed on lattice bases that are at least LLL reduced in a preprocessing step, as this reduces the runtime significantly compared to non-reduced bases.

The LLL algorithm runs in time polynomial in the lattice dimension and therefore can be applied in high lattice dimensions ($n > 1000$). The runtime of all known exhaustive search algorithms is exponential in the dimension, and therefore can only be applied in blocks of smaller dimension ($n \lessapprox 70$). With this, the runtime of BKZ increases exponentially in the blocksize β. As in BKZ, enumeration is executed very frequently, it is only practical to choose blocksizes up to 50. For high blocksize, our experience shows that ENUM takes 99% of the time of BKZ.

There are numerous works on parallelization of LLL [Vil92, HT98, RV92, Jou93] [Wet98, BW09]. Parallel versions of lattice enumeration were presented in the masters theses of Pujol [Puj08] and Dagdelen [Dag09] (in french and german language, respectively). Both approaches are not suitable for GPU, since they require dynamic creation of new threads, which is not possible for GPUs.

Being able to parallelize ENUM means to parallelize the second (more time consuming) building block of BKZ, which reduces the runtime of the most promising lattice reduction algorithm in total.

As a platform for our parallel implementation we have chosen graphical processing units (GPUs). Because of their design to perform identical operations on large amounts of graphical data, GPUs can run large numbers of threads in parallel, provided the threads execute similar instructions. We can take advantage of this design and split up the ENUM algorithm over several identical threads. The computation power of GPU rises faster than that of CPUs over the last years, with respect to floating point operations per second (GFlops). This trend is not supposed to stop, therefore using GPUs for computation will be a useful model also in the near future.

Our Contribution. In this paper we present a parallel version of the enumeration algorithm of [SE91] that finds a shortest, non-zero vector in a lattice. Since the enumeration algorithm is tree-based, the main challenge is splitting the tree in some way and executing subtree enumerations in parallel. We use the CUDA framework of NVIDIA for implementing the algorithm on graphics cards. Because of the choice for GPUs, parallelization and splitting are more difficult than for a CPU parallelization. Firstly we explain the ideas of how to parallelize enumeration on GPU. Secondly we present some first experimental results. Using the GPU, we reduce the time required for enumeration of a random lattice in dimensions higher than 50 by a factor of almost 5. We are using random lattices in the sense of Goldstein and Mayer [GM03] for testing our implementation.

The first part of this paper, namely the idea of parallelizing enumeration, can also be applied on multicore CPU. The idea of splitting the search tree into parts and search different subtrees independently in parallel is also applicable on CPU, or other parallel computing frameworks. As mentioned above, BKZ is only practical using blocksizes up to 50. As our GPU version of the enumeration performs best in dimensions n greater than 50, we would expect to speed up BKZ with high blocksizes only.

In contrast to our algorithm, Pujol's idea [Puj08] is to predict the number of enumeration steps in a subtree beforehand, using a volume heuristic. If the number of expected steps in a subtree exceeds some bound, the subtree is split recursively, and enumerated as different threads. Dagdelen [Dag09] bounds the height of subtrees that can be split recursively. Both ideas differ from our approach, as we use a real-time scheduling; when a subtree enumeration has exceeded a specified number of enumeration steps it is stopped, to balance the

load of all GPU kernels. This fits best into the SIMD structure of GPUs, as both existing approaches lead to a huge number of diverging subthreads.

Structure of the Paper. In Section 2 we introduce the necessary preliminaries on lattices and GPUs. We discuss previous lattice reduction algorithms and the applications of lattices in cryptography. The GPU (CUDA) programming model is shortly introduced, explaining in more detail the memory model and data types which are important for our implementation. Section 3 explains our parallel enumeration algorithm, starting from the ENUM algorithm of Schnorr and Euchner and ending with the iterated GPU enumeration algorithm. Section 4 discusses the results obtained with our algorithm.

2 Preliminaries

A lattice is a discrete subgroup of $\mathbb{R}^d$. It can be represented by a basis matrix $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ $(n \leq d)$. We call $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\}$ the lattice spanned by the column basis vectors $\mathbf{b}_i \in \mathbb{R}^d$ $(i = 1 \ldots n)$. The dimension n of a lattice is the number of linear independent vectors in the lattice, i.e. the number of basis vectors. When $n = d$ the lattice is called full dimensional.

The basis of a lattice is not unique. Every unimodular transformation $\mathbf{M}$, i.e. integer transformation with $\det \mathbf{M} = \pm 1$, turns a basis matrix $\mathbf{B}$ into a second basis $\mathbf{MB}$ of the same lattice.

The determinant of a lattice is defined as $\det(\mathcal{L}(\mathbf{B})) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$. For full dimensional lattices we have $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$. The determinant of a lattice is invariant of the choice of the lattice basis, which follows from the multiplicative property of the determinant and the fact that basis transformations have determinant ± 1.

The length of a shortest vector of a lattice $\mathcal{L}(\mathbf{B})$ is denoted $\lambda_1(\mathcal{L}(\mathbf{B}))$ or in short λ_1 if the lattice is uniquely determined.

The Gram-Schmidt algorithm computes an orthogonalization of a basis. It is an efficient algorithm that outputs $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ with $\mathbf{b}_i^*$ orthogonal and $\mu_{i,j}$ such that $\mathbf{B} = \mathbf{B}^* \cdot [\mu_{i,j}]$, where $[\mu_{i,j}]$ is an upper triangular matrix consisting of the Gram-Schmidt coefficients $\mu_{i,j}$ for $1 \leq j \leq i \leq n$. The orthogonalized matrix $\mathbf{B}^*$ is not necessarily a basis of the lattice.

2.1 Lattice Basis Reduction

Problems. Some lattice bases are more useful than others. The goal of lattice basis reduction (or in short lattice reduction) is to find a basis consisting of short and almost orthogonal lattice vectors. More exactly, we can define some (hard) problems on lattices. The most important one is the *shortest vector problem* (SVP), which consists of finding a vector $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| = \lambda_1(\mathcal{L}(\mathbf{B}))$. In most cases, the Euclidean norm $\|\cdot\|_2$ is considered. As the SVP is $\mathcal{NP}$-hard (at least under randomized reductions) [Din02, Kho05, RR06] people consider the approximate version γ-SVP, that tries to find a vector $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$.

Other important problems like the *closest vector problem* (CVP) that searches for a nearest lattice vector to a given point in space, its approximation variant γ-CVP, or the *shortest basis problem* (SBP) are listed and described in detail in [MG02].

Algorithms. In 1982 Lenstra, Lenstra, and Lovász [LLL82] introduced the LLL algorithm, which was the first polynomial time algorithm to solve the approximate shortest vector problem in higher dimensions. Another algorithm is the BKZ block algorithm of Schnorr and Euchner [SE91]. In practice, this is the algorithm that gives the best solution to lattice reduction so far. Their paper [SE91] also introduces the enumeration algorithm (ENUM), a variant of the Fincke-Pohst [FP83] and Kannan [Kan83] algorithms. The ENUM algorithm is the fastest algorithm in practice to solve the exact shortest vector problem using complete enumeration of all lattice vectors in a suitable search space. It is used as a black box in the BKZ algorithm. The enumeration algorithm organizes linear combinations of the basis vectors in a search tree and performs a depth first search above the tree.

In [PS08] Pujol and Stehlé analyze the stability of the enumeration when using floating point arithmetic. In [HS07], improved complexity bounds for Kannan's algorithm are presented. This paper also suggests some better preprocessing of lattice bases, i.e., the authors suggest to BKZ reduce a basis before running enumeration. This approach lowers the runtime of enumeration. In this paper we consider both LLL and BKZ pre-reduced bases. [AKS01] show how to solve SVP using a randomized algorithm in time $2^{\mathcal{O}(n)}$, but their algorithm requires exponential space and is therefore impractical. The papers [NV08] and [MV10] present improved sieving variants, where the Gauss-sieving algorithm of [MV10] is shown to be really competitive to enumeration algorithms in practically interesting dimensions.

Several LLL variants were presented by Schnorr [Sch03], Nguyen and Stehlé [NS05], and Gama and Nguyen [GN08a]. The variant of [NS05] is implemented in the `fpLLL` library of [CPS], which is also the fastest public implementation of ENUM algorithms. Koy introduced the notion of a primal-dual reduction in [Koy04]. Schnorr [Sch03] and Ludwig [BL06] deal with random sampling reduction. Both are slightly different concepts of lattice reduction, where primal-dual reduction uses the dual of a lattice for reducing and random sampling combines LLL-like algorithms with an exhaustive point search in a set of lattice vectors that is likely to contain short vectors.

The papers [SE91, SH95] present a probabilistic improvement of ENUM, called tree pruning. The idea is to prune subtrees that are unlikely to contain shorter vectors. As it leads to a probabilistic variant of the enumeration algorithm, we do not consider pruning techniques here.

In [GN08b] Gama and Nguyen compare the `NTL` implementation [Sho] of floating point LLL, the deep insertion variant of LLL and the BKZ algorithm. It is the first comprehensive comparison of lattice basis reduction algorithms and helps understanding their practical behavior.

In [Vil92, HT93, RV92] the authors present parallel versions for n and n^2 processors, where n is the lattice dimension. In [Jou93] the parallel LLL of Villard [Vil92] is combined with the floating point ideas of [SE91]. In [Wet98] the authors present a blockwise generalization of Villards algorithm. Backes and Wetzel worked out a parallel variant of the LLL algorithm for multi-core CPU architectures [BW09]. For the parallelization of lattice reduction on GPU the authors are not aware of any previous work.

Applications. Lattice reduction has applications in cryptography as well as in cryptanalysis. The foundation of some cryptographic primitives is based on the hardness of lattice problems. Lattice reduction helps determining the practical hardness of those problems and is a basis for real world application of those hash functions, signatures, and encryption schemes. Well known examples are the SWIFFT hash functions of Lyubashevsky et al. [LMPR08], the signature schemes of [LM08, GPV08, Lyu09, Pei09a], or the encryption schemes of [AD97, Pei09b, SSTX09]. The NTRU [HPS98, otCC09] and GGH [GGH97] schemes do not provide a security proof, but the best attacks are also lattice based.

There are also attacks on RSA and similar systems, using lattice reduction to find small roots of polynomials [CNS99, DN00, May10]. Low density knapsack cryptosystems were successfully attacked with lattice reduction [LO85]. Other applications of lattice basis reduction are factoring numbers and computing discrete logarithms using diophantine approximations [Sch91]. In Operations

Research, or generally speaking, discrete optimization, lattice reduction can be used to solve linear integer programs [Len83].

2.2 Programming Graphics Cards

A Graphical Processing Units (GPUs) is a piece of hardware that is specifically designed to perform a massive number of specific graphical operations in parallel. The introduction of platforms like CUDA by NVIDIA [Nvi07a] or CTM by ATI [AMD06], that make it easier to run custom programs instead of limited graphical operations on a GPU, has been the major breakthrough for the GPU as a general computing platform. The introduction of integer and bit arithmetic also broadened the scope to cryptographic applications.

Applications. Many general mathematical packages are available for GPU, like the BLAS library [NVI07b] that supports basic linear algebra operations.

An obvious application in the area of cryptography is brute force searching using multiple parallel threads on the GPU. There are also implementations of AES [CIKL05, Man07, HW07] and RSA [MPS07, SG08, Fle07] available. GPU implementations can also be used (partially) in cryptanalysis. In 2008, Bernstein et al. use parallelization techniques on graphics cards to solve integer factorization using elliptic curves [BCC$^+$09]. Using NVIDIA's CUDA parallelization framework, they gained a speed-up of up to 6 compared to computation on a four core CPU. However, to date, no applications based on lattices are available for GPU.

Programming Model. For the work in this paper the CUDA platform will be used. The GPUs from the Tesla range, which support CUDA, are composed of several multiprocessors, each containing a small number of scalar processors. For the programmer this underlying hardware model is hidden by the concept of SIMT-programming: Single Instruction, Multiple Thread. The basic idea is that the code for a single thread is written, which is then uploaded to the device and executed in parallel by multiple threads.

The threads are organized in multidimensional arrays, called blocks. All blocks are again put in a multidimensional array, called the *grid*. When executing a program (a grid), threads are scheduled in groups of 32 threads, called *warps*. Within a warp threads should not diverge, as otherwise the execution of the warp is serialized.

Memory Model. The Tesla GPUs provide multiple levels of memory: registers, shared memory, global memory, texture and constant memory. Registers and shared memory are on chip and close to the multiprocessor and can be accessed with low latency. The number of registers and shared memory is limited, since the number available for one multiprocessor must be shared among all threads in a single block.

Global memory is off-chip and is not cached. As such, access to global memory can slow down the computations drastically, so several strategies for speeding up memory access should be considered (besides the general strategy of avoiding global memory access). By coalescing memory access, e.g. loading the same memory address or a consecutive block of memory from multiple threads, the delay is reduced, since a coalesced memory access has the same cost as a single random memory access. By launching a large number of blocks the latency introduced by memory loading can also be hidden, since other blocks can be scheduled in the meantime.

The constant and texture memory are cached and can be used for specific types of data or special access patterns.

Instruction Set. Modern GPUs provide the full range of (32 and) 64 bit floating point, integer and bit operations. Addition and multiplication are fast, other operations can, depending on the type, be much slower. There is no point in using other than 32 or 64 bit numbers, since smaller types are always cast to larger types. Most GPUs have a specialized FMAD instruction, which performs a floating point multiplication followed by an addition at the cost of only a single operation. This instruction can be used during the BKZ enumeration.

One problem that occurs on GPUs is the fact that today GPUs are not able to deal with higher precision than 64 bit floating point numbers. For lattice reduction, sometimes higher bit sizes are required to guarantee the correct termination of the algorithms. For an n-dimensional lattice, using the floating point LLL algorithm of [LLL82], one requires a precision of $\mathcal{O}(n \log B)$ bits, where B is an upper bound for the length of the d-dimensional vectors [NS05]. For the L^2 algorithm of [NS05], the required bit size is $\mathcal{O}(n \log_2 3)$, which is independent of the norm of the input basis vectors. For more details on the floating point LLL analysis see [NS05] and [NS06].

In [PS08] the authors state that for enumeration algorithms double precision is suitable up to dimension 90, which is beyond the dimensions that are practical today. Therefore enumeration should be possible on actual graphics cards, whereas the implementation of LLL-like algorithms will be more complicated

and require some multi-precision framework.

3 Parallel Enumeration on GPU

In this section we present our parallel algorithm for shortest vector enumeration in lattices. In Subsection 3.1 we briefly explain the ENUM algorithm of Schnorr and Euchner [SE91], which was used as a basis for our algorithm. Next, we present the basic idea for multi-thread enumeration in Subsection 3.2. Finally, in Subsection 3.3, we explain our parallel algorithm in detail.

The ENUM algorithm of Schnorr-Euchner is an improvement of the algorithms from [Kan83] and [FP83]. The ENUM algorithm is the fastest one today and also the one used in the `NTL` [Sho] and `fpLLL` [CPS] libraries. Therefore we have chosen this algorithm as basis for our parallel algorithm.

3.1 Original ENUM Algorithm

The ENUM algorithm enumerates over all linear combinations $[x_1, \ldots, x_n] \in \mathbb{Z}^n$ that generate a vector $\mathbf{v} = \sum_{i=1}^{n} x_i \mathbf{b}_i$ in the search space (i.e., all vectors $\mathbf{v}$ with $\|\mathbf{v}\|$ smaller than a specified bound). Those linear combinations are organized in a tree structure. Leafs of the tree contain full linear combinations, whereas inner nodes contain partly filled vectors. The search for the tree leaf that determines the shortest lattice vector is performed in a depth first search order. The most important part of the enumeration is cutting off parts of the tree, i.e. the strategy which subtrees are explored and which ones cannot lead to a shorter vector.

Let i be the current level in the tree, $i = 1$ being at the bottom and $i = n$ at the top of the tree (c.f. Figure 1). Each step in the enumeration algorithm consists of computing an *intermediate* squared norm l_i, moving one level up or down the tree (to level $i' \in \{i-1, i+1\}$) and determining a new value for the coordinate $x_{i'}$.

Let $r_i = \|\mathbf{b}_i^*\|^2$. We define $l_i = l_{i+1} + y_i^2 r_i$ with $y_i = x_i - c_i$ and $c_i = -\sum_{j=i+1}^{n} \mu_{j,i} x_j$. So, for a certain choice of coordinates $x_i \ldots x_n$ it holds that $l_k \geq l_i$ (with $k < i$) for all coordinate vectors $\mathbf{x}$ that end with the same coordinates $x_i \ldots x_n$. This implies that the intermediate norm l_i can be used to cut off infeasible subtrees. If $l_i > A$, with A the squared norm of the shortest vector that has been found so far, the algorithm will increase i and move up inside the tree. Otherwise, the algorithm will lower i and move down in the

tree. Usually, as initial bound A for the length of the shortest vector, one uses the norm of the first basis vector.

The next value for $x_{i'}$ is selected in an interval of length $\sqrt{\frac{A-l_{i'+1}}{r_{i'}}}$ centered at $c_{i'}$. The interval is enumerated according to the zig-zag pattern described in [SE91]. Starting from a central value $\lfloor c_{i'} \rceil$, ENUM will generate a sequence $\lfloor c_{i'} \rceil + 1, \lfloor c_{i'} \rceil - 1, \lfloor c_{i'} \rceil + 2, \lfloor c_{i'} \rceil - 2, \ldots$ for the coordinate $x_{i'}$. To be able to generate such a pattern, helper vectors $\mathbf{\Delta x} \in \mathbb{Z}^n$ are used. We do not require to store $\mathbf{\Delta^2 x}$ as in the orginal algorithm [SE91, PS08], as the computation of the zigzag pattern is done in a slightly different way as in the original algorithm. For a more detailed description of the ENUM algorithm we refer to [PS08].

3.2 Multi-Thread Enumeration

Roughly speaking, the parallel enumeration works as follows. The search tree of combinations that is explored in the enumeration algorithm can be split at a high level, distributing subtrees among several threads. Each thread then runs an enumeration algorithm, keeping the first coefficients fixed. These fixed coefficients are called *start vectors*. The subtree enumerations can run independently, which limits communication between threads. The top level enumeration is performed on CPU and outputs start vectors for the GPU threads.

When the number of postponed subtrees is higher than the number of threads that we can start in parallel, then we copy the start vectors to the GPU and let it enumerate the subtrees. After all threads have finished enumerating their subtrees we proceed in the same manner: caching start vectors on CPU and starting a batch of subtree enumerations on GPU. Figure 1 illustrates this approach. The variable α defines the region where the initial enumeration is performed. The subtrees where GPU threads work are also depicted in Figure 1.

If a GPU subtree enumeration finds a new optimal vector, it writes back the coordinates $\mathbf{x}$ and the squared norm A of this vector to the main memory. The other GPU threads will directly receive the new value for A, which will allow them to cut away more parts of the subtree.

Early Termination. The computation power of the GPU is used best when as many threads as possible are working at the same time. Recall that the GPU uses warps as the basic execution units: all threads in a warp are running the same instructions (or some of the threads in the warp are stalled in the case of branching).

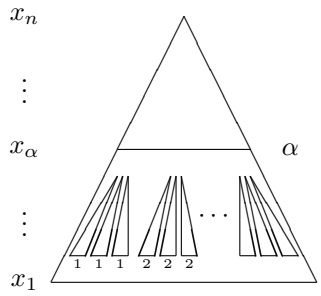


Figure 1: Illustration of the algorithm flow. The top part is enumerated on CPU, the lower subtrees are explored in parallel on GPU. The tiny numbers illustrate which subtrees are enumerated in the same iteration.

In general, more starting vectors than there are GPU threads are uploaded in each run of the GPU kernel. This allows us to do some load balancing on the GPU, to make sure all threads are busy. To avoid the GPU being stalled by a few long running subtree enumerations, the GPU stops when just a few subtrees are left. We call this process, by which the GPU stops some subtrees even though they are not finished, *early termination*.

At the end of Section 3.3 details are included on the exact way early termination and our load balancing algorithm works. For now it suffices to know that, because of early termination, some of the subtree enumerations are not finished after a single launch of the GPU kernel. This is the main reason why the entire algorithm is iterated several times. At each iteration the GPU launches a mix of enumerations: new subtrees (start vectors) from the top enumeration and subtrees that were not finished in one of the previous GPU launches.

3.3 The Iterated Parallel ENUM Algorithm

Algorithm 10 shows the high-level layout of the GPU enumeration algorithm. Details concerning the updating of the bound A, as well as the write-back of newly discovered optimal vectors have been omitted. The actual enumeration is also not shown: it is part of several subroutines which are called from the main algorithm.

The whole process of launching a grid of GPU threads is iterated several times (line 10), until the whole search tree has been enumerated either on GPU or CPU.

In line 10, the top of the search tree is enumerated, to generate a set S of starting vectors $\mathbf{x}_k$ for which enumeration should be started at level α. More

Algorithm 10: High-level Iterated Parallel ENUM Algorithm

Input: $\mathbf{b}_i (i = 1, \ldots, n)$, A, α, n

```
1  Compute the Gram-Schmidt orthogonalization of b_i
2  while true do
3  |  S = {(x_k, Δx_k, L_k = α, s_k = 0)}_k ← Top enum: generate at most
   |  NUMSTARTPOINTS − #T vectors
4  |  R = {(x̄_k, Δx_k, L_k, s_k)}_k ← GPU enumeration, starting from S ∪ T
5  |  T ← {R_k : subtree k was not finished}
6  |  if #T < CPUTHRESHOLD then
7  |  |  Enumerate the starting points in T on the CPU.
8  |  |  Stop
9  |  end
10 end
```

Output: $(x_1, \ldots, x_n)$ with $\|\sum_{i=1}^{n} x_i \mathbf{b}_i\| = \lambda_1(\mathcal{L})$

detailed, the top enumeration in the region between α and n outputs distinct vectors

$$\mathbf{x}_k = [0, \ldots, 0, x_{k,\alpha}, \ldots, x_{k,n}] \qquad \text{for } k = 1 \ldots \text{NUMSTARTPOINTS} - \#T\,.$$

The top enumeration will stop automatically if a sufficient number of vectors from the top of the tree have been enumerated. The rest of the top of the tree is enumerated in the following iterations of the algorithm.

Line 10 performs the actual GPU enumeration. In each iteration, a set of starting vectors and starting levels $\{\mathbf{x}_k, L_k\}$ is uploaded to the GPU. These starting vectors can be either vectors generated by the top enumeration in the region between α and n (in which case $L_k = \alpha$) or the vectors (and levels) written back by the GPU because of early termination, so that the enumeration will continue. In total NUMSTARTPOINTS vectors (a mix of new and old vectors) are uploaded at each iteration. For each starting vector $\mathbf{x}_k$ (with associated starting level L_k) the GPU outputs a vector

$$\bar{\mathbf{x}}_k = [\bar{x}_{k,1}, \ldots, \bar{x}_{k,\alpha-1}, x_{k,\alpha}, \ldots, x_{k,n}] \qquad \text{for } k = 1 \ldots \text{NUMSTARTPOINTS}$$

(which describes the current position in the search tree), the current level L_k, the number of enumeration steps s_k performed and also part of the internal state of the enumeration. This state $\{\bar{\mathbf{x}}_k, \mathbf{\Delta x}_k, L_k\}$ can be used to continue the enumeration later on. The vectors $\mathbf{\Delta x}_k$ are used in the enumeration to generate the zig-zag pattern and are part of the internal state of the enumeration [SE91]. This state is added to the output to be able to efficiently restart the enumeration at the point it was terminated.

Line 10 will select the resulting vectors from the GPU enumeration that were terminated early. These will be added to the set T of *leftover* vectors, which will be relaunched in the next iteration of the algorithm. If the set of leftover vectors is too small to get an efficient GPU enumeration, the CPU takes over and finishes off the last part of the enumeration. This final part only takes limited time.

GPU Threads and Load Balancing. In Section 3.2 the need for a load balancing algorithm was introduced: all threads should remain active and to ensure this, each thread in the same warp should run the same instruction. One of the problems in achieving this, is the length difference of each subtree enumeration. Some very long subtree enumeration can cause all the other threads in the warp to become idle after they finish their subtree enumeration.

Therefore the number of enumeration steps that each thread can perform on a subtree is limited by $\mathbf{M}$. When $\mathbf{M}$ is exceeded, a subtree enumeration is forced to stop. After this, all threads in the same warp will reinitialise: they will either continue the previous subtree enumeration (that was terminated by reaching $\mathbf{M}$) or they will pick a new starting vector of the list $S \cup T$ delivered by the CPU. Then the enumeration starts again, limited to $\mathbf{M}$ enumeration steps.

In our experiments, NUMSTARTPOINTS was around 20-30 times higher than NUMTHREADS, which means that on average every GPU thread enumerated 20-30 subtrees in each iteration. $\mathbf{M}$ was chosen to be around 50-200.

4 Experimental Results

In this section we present some results of the CUDA implementation of our algorithm. For comparison we used the highly optimized ENUM algorithm of the `fpLLL` library in version 3.0.11 from [CPS]. `NTL` does not allow to run ENUM as a standalone SVP solver, but [Puj08] and the ENUM timings of [GN08b] show that `fpLLL`'s ENUM runs faster than `NTL`'s (the bit size of the lattice bases used in [GN08b] is higher than what we used, therefore a comparison with those timings is to be drawn carefully).

The CUDA program was compiled using `nvcc`, for the CPU programs we used `g++` with compiler flag `-O2`. The tests were run on an Intel Core2 Extreme CPU X9650 (using one single core) running at 3 GHz, and an NVIDIA GTX

280 graphics card. We run up to 100000 threads in parallel on the GPU. The code of our program can be found online.[3]

We chose random lattices following the construction principle of [GM03] with bit size of the entries of $10 \cdot n$. This type of lattices was also used in [GN08b] and [NS06]. We start with the basis in Hermite normal form and LLL-reduce them with $\delta = 0.99$. At the end of this section, we present some timings using BKZ-20 reduced bases, to show the capabilities of stronger pre-reduction.

Both algorithms, the enum of `fpLLL` (run with parameter *-a svp*) and our CUDA version, always output the same coefficient vectors and therefore a lattice vector with shortest possible length. We compare now the throughput of GPU and CPU concerning enumerations steps. Section 3.1 gives the explanation what is computed in each enumeration step. On the GPU, up to 200 million enumeration steps per second can be computed, while similar experiments on CPU only yielded 25 million steps per second. We choose $\alpha = n - 11$ for our experiments, this shapes up to be a good choice in practice. Table 1 and Figure 2 illustrate the experimental results. The figure shows the runtimes

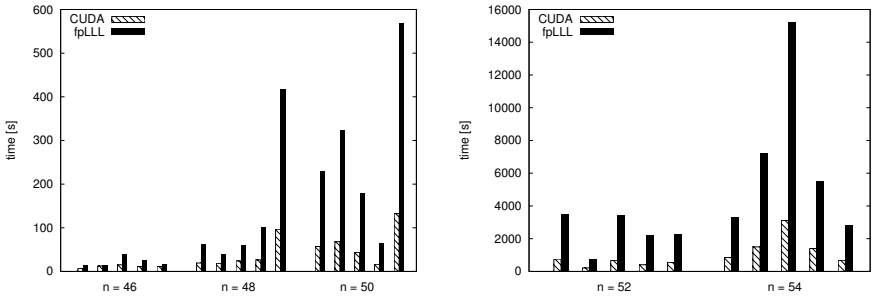


Figure 2: Timings for enumeration. The graph shows the time needed for enumerating five different random lattices in each dimension n. It compares the ENUM algorithm of the `fpLLL`-library with our parallel CUDA version.

of both algorithms when applied to five different lattices of each dimension. One can notice that in dimension above 44, our CUDA implementation always outperforms the `fpLLL` implementation.

Table 1 shows the average value over all five lattices in each dimension. Again one notices that the GPU algorithm demonstrates its strength in dimensions above 44, where the time goes down to 22% in dimensions 54 and 56 and

[3] `http://homes.esat.kuleuven.be/~jhermans/gpuenum/index.html`

Table 1: Average time needed for enumeration of LLL pre-reduced lattices in each dimension n. The table presents the percentage of time that the GPU version takes compared to the `fpLLL` version.

n	40	42	44	46	48	50	52	54	56	60
`fpLLL` - ENUM	0.96s	2.41s	17.7s	22.0s	136s	273s	2434s	6821s	137489s	-
CUDA - ENUM	2.75s	4.29s	11.7s	11.4s	37.0s	63.5s	520s	1504s	30752s	274268s
	286%	178%	66%	52%	27%	23%	21%	22%	22%	-

down to 21% in dimension 52. Therefore we state that the GPU algorithm gains big speedups in dimensions higher than 45, which are the interesting ones in practice. In dimension 60, `fpLLL` did not finish the experiments in time, therefore only the average time of the CUDA version is presented in the table.

Table 2 presents the timing of the same bases, pre-reduced using BKZ algorithm with blocksize 20. The time of the BKZ-20 reduction is not included in the timings shown in the table. For dimension 64 we changed α (the subtree dimension) from the usual $n - 11$ to $\alpha = n - 14$, as this leads to lower timings in high dimensions. First, one can notice that both algorithms run much faster when using stronger pre-processing, a fact that was already mentioned in [HS07]. Second, we see that the speedup of the GPU version goes down to 13% in the best case (dimension 62).

Table 2: Average time needed for enumeration of BKZ-20 pre-reduced lattices in each dimension n. The time for pre-reduction is omitted in both cases.

n	48	50	52	54	56	58	60	62	64
`fpLLL` - ENUM	2.96s	7.30s	36.5s	79.2s	190s	601s	1293s	7395s	15069s
CUDA - ENUM	3.88s	5.42s	16.9s	27.3s	56.8s	119s	336s	986s	4884s
	131%	74%	46%	34%	30%	20%	26%	13%	32%

As pruning would speed up both the serial and the parallel enumeration, we expect the same speedups with pruning.

It is hard to give an estimate of the achieved speedup compared to the number of threads used: since GPUs have hardware-based scheduling, it is not possible to know the number of active threads exactly. Other properties, like memory access and divergent warps, have a much greater influence on the performance and cannot be measured in thread counts or similar figures. When comparing only the number of double fmadds, the GTX 280 should be able to do 13 times more fmadd's than a single Core2 Extreme X9650.[4] Based on our results we fill only 30 to 40% of the GPUs ALUs. Using the CUDA Profiler, we determine

[4] A GTX280 can do 30 double fmadds in a 1.3GHz cycle, a single Core2 core can do 2 double fmadds in every two 3GHz cycle, which gives us a speedup of 13 for the GTX280.

that in our experiments around 12% of branches was divergent, which implies a loss of parallelism and also some ALUs being left idle. There is also a high number of warp serializations due to conflicting shared and constant memory access. The ratio warp serializations/instructions is around 35%.

To compare CPUs and GPUs, we can have a look at the cost of both platforms in dollardays, similar to the comparison in [BCC+09]. We assume a cost of around $2200 for our CPU (quad core) + 2x GTX295 setup. For a CPU-only system, the cost is only around $900. Given a speedup of 5 for a GPU compared to a CPU, we get a total speedup of 24 (4 CPU cores + 4 GPUs) in the $2200 machines and only a speedup of 4 in the CPU-only machine, assuming we can use all cores. This gives $225 \cdot t$ dollardays for the CPU-only system and only $91 \cdot t$ dollardays for the CPU+GPU system, where t is the time. This shows that even in this model of expense, the GPU implementation gains an advantage of around 2.4.

5 Further Work

Further improvements are possible using multiple CPU cores. Our implementation only uses one CPU core for the top enumeration and the rest of the outer loop of the enumeration. During the subtree enumerations on the GPU, the main part of the algorithm, the CPU is not used. When the GPU starts a batch of subtree enumerations it would be possible to start threads on the CPU cores as well. We expect a speedup of two compared to our actual implementation using this idea.

It is possible to start enumeration using a shorter starting value than the first basis vectors norm. The Gaussian heuristic can be used to predict the norm of the shortest basis vector λ_1. This can lead to enormous speedups in the algorithm. We did not include this improvement into our algorithm so far to get comparable results to `fpLLL`.

Acknowledgments

We thank the anonymous referees for their valuable comments. We thank Özgür Dagdelen for creating some of the initial ideas of parallelizing lattice enumeration and Benjamin Milde, Chen-Mou Cheng, and Bo-Yin Yang for the nice discussions and helpful ideas.

We would like to thank EcryptII[5] and CASED[6] for providing the funding for the visits during which this work was prepared. Part of the work was done during the authors' visit to Center for Information and Electronics Technologies, National Taiwan University.

References

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Annual Symposium on the Theory of Computing — STOC 1997*, pages 284–293, 1997.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the Annual Symposium on the Theory of Computing — STOC 2001*, pages 601–610. ACM Press, 2001.

[AMD06] Advanced Micro Devices. ATI CTM Guide. Technical report, 2006.

[BCC+09] Daniel J. Bernstein, Tien-Ren Chen, Chen-Mou Cheng, Tanja Lange, and Bo-Yin Yang. ECM on graphics cards. In *Advances in Cryptology — Eurocrypt 2009*, volume 5479 of *LNCS*, pages 483–501, 2009.

[BL06] Johannes Buchmann and Christoph Ludwig. Practical lattice basis sampling reduction. In *Algorithmic Number Theory Symposium — ANTS 2006*, volume 4076 of *LNCS*, pages 222–237. Springer-Verlag, 2006.

[BW09] Werner Backes and Susanne Wetzel. Parallel lattice basis reduction using a multi-threaded Schnorr-Euchner LLL algorithm. In *15th International European Conference on Parallel and Distributed Computing — Euro-Par*, 2009.

[CIKL05] Debra L. Cook, John Ioannidis, Angelos D. Keromytis, and Jake Luck. Cryptographics: Secret key cryptography using graphics cards. In *Topics in Cryptology — Cryptographer's Track, RSA Conference — CT-RSA 2005*, pages 334–350, 2005.

[5] `http://www.ecrypt.eu.org/`
[6] `http://www.cased.de`

[CNS99] Christophe Coupé, Phong Q. Nguyen, and Jacques Stern. The effectiveness of lattice attacks against low exponent RSA. In *Public-Key Cryptography — PKC*, volume 1560 of *LNCS*, pages 204–218. Springer-Verlag, 1999.

[CPS] David Cadé, Xavier Pujol, and Damien Stehlé. fpLLL - a floating point LLL implementation. Available at Damien Stehlé's homepage at école normale supérieure de Lyon, `http://perso.ens-lyon.fr/damien.stehle/english.html`.

[Dag09] Özgür Dagdelen. Parallelisierung von Gitterbasisreduktionen. Masters thesis, TU Darmstadt, 2009.

[Din02] Irit Dinur. Approximating SVP_∞ to within almost-polynomial factors is NP-hard. *Theoretical Computer Science*, 285(1):55–71, 2002.

[DN00] Glenn Durfee and Phong Q. Nguyen. Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt '99. In *Advances in Cryptology — Asiacrypt 2000*, volume 1976 of *LNCS*, pages 14–29. Springer-Verlag, 2000.

[Fle07] Sebastian Fleissner. GPU-Accelerated Montgomery Exponentiation. In *International Conference on Computational Science — ICCS*, volume 4487 of *LNCS*, pages 213–220. Springer-Verlag, 2007.

[FP83] U. Fincke and Michael Pohst. A procedure for determining algebraic integers of given norm. In *European Computer Algebra Conference 1983*, volume 162 of *LNCS*, pages 194–202. Springer-Verlag, 1983.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology — Crypto 1997*, volume 1294 of *LNCS*, pages 112–131. Springer-Verlag, 1997.

[GM03] Daniel Goldstein and Andrew Mayer. On the equidistribution of hecke points. *Forum Mathematicum 2003, 15:2*, pages 165–189, 2003.

[GN08a] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *Proceedings of the Annual Symposium on the Theory of Computing — STOC 2008*, pages 207–216. ACM Press, 2008.

[GN08b] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *Advances in Cryptology — Eurocrypt 2008*, volume 4965 of *LNCS*, pages 31–51, 2008.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Annual Symposium on the Theory of Computing — STOC 2008*, pages 197–206. ACM Press, 2008.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory Symposium — ANTS 1998*, volume 1423 of *LNCS*, pages 267–288, 1998.

[HS07] Guillaume Hanrot and Damien Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. In *Advances in Cryptology — Crypto 2007*, volume 4622 of *LNCS*, pages 170–186. Springer-Verlag, 2007.

[HT93] Christian Heckler and Lothar Thiele. A parallel lattice basis reduction for mesh-connected processor arrays and parallel complexity. In *IEEE Symposium on Parallel and Distributed Processing — SPDP*, pages 400–407. IEEE Computer Society Press, 1993.

[HT98] Christian Heckler and Lothar Thiele. Complexity analysis of a parallel lattice basis reduction algorithm. *SIAM J. Comput.*, 27(5):1295–1302, 1998.

[HW07] Owen Harrison and John Waldron. AES Encryption Implementation and Analysis on Commodity Graphics Processing Units. In *Cryptographic Hardware and Embedded Systems — CHES 2007*, volume 4727 of *LNCS*, pages 209–226. Springer-Verlag, 2007.

[Jou93] Antoine Joux. A fast parallel lattice reduction algorithm. In *Proceedings of the Second Gauss Symposium*, pages 1–15, 1993.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the Annual Symposium on the Theory of Computing — STOC 1983*, pages 193–206. ACM Press, 1983.

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005.

[Koy04] Henrik Koy. Primale-duale Segment-Reduktion. http://www.mi.informatik.uni-frankfurt.de/research/papers.html, 2004.

[Len83] Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[LLL82] Arjen Lenstra, Hendrik Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LM08] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *Theory of Cryptography Conference — TCC 2008*, LNCS, pages 37–54. Springer-Verlag, 2008.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifft: A modest proposal for fft hashing. In *Fast Software Encryption — FSE 2008*, LNCS, pages 54–72. Springer-Verlag, 2008.

[LO85] J. C. Lagarias and Andrew M. Odlyzko. Solving low-density subset sum problems. *Journal of the ACM*, 32(1):229–246, 1985.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology — Asiacrypt 2009*, volume 5912 of *LNCS*, pages 598–616. Springer-Verlag, 2009.

[Man07] Svetlin A. Manavski. Cuda Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography. In *IEEE International Conference on Signal Processing and Communications — ICSPC*, pages 65–68. IEEE Computer Society Press, 2007.

[May10] Alexander May. Using LLL-reduction for solving RSA and factorization problems. In Phong Q. Nguyen and Brigitte Vallée, editors, *The LLL algorithm*, pages 315–348. Springer, 2010.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[MPS07] Andrew Moss, Dan Page, and Nigel P. Smart. Toward Acceleration of RSA Using 3D Graphics Hardware. In *IMA International Conference*, volume 4887 of *LNCS*, pages 364–383. Springer-Verlag, 2007.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proceedings of the Annual Symposium on Discrete Algorithms — SODA 2010*, 2010.

[NS05] Phong Q. Nguyen and Damien Stehlé. Floating-point LLL revisited. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *LNCS*, pages 215–233. Springer-Verlag, 2005.

[NS06] Phong Q. Nguyen and Damien Stehlé. LLL on the average. In *Algorithmic Number Theory Symposium — ANTS 2006*, volume 4076 of *LNCS*, pages 238–256. Springer-Verlag, 2006.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *J. of Mathematical Cryptology*, 2(2), 2008.

[Nvi07a] Nvidia. Compute Unified Device Architecture Programming Guide. Technical report, 2007.

[NVI07b] NVIDIA. CUBLAS Library, 2007.

[otCC09] 1363 Working Group of the C/MM Committee. IEEE P1363.1 Standard Specification for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, 2009. Available at `http://grouper.ieee.org/groups/1363/`.

[Pei09a] Chris Peikert. Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive, Report 2009/359, 2009. `http://eprint.iacr.org/`.

[Pei09b] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the Annual Symposium on the Theory of Computing — STOC 2009*, pages 333–342, 2009.

[PS08] Xavier Pujol and Damien Stehlé. Rigorous and efficient short lattice vectors enumeration. In *Advances in Cryptology — Asiacrypt 2008*, volume 5350 of *LNCS*, pages 390–405. Springer-Verlag, 2008.

[Puj08] Xavier Pujol. Recherche efficace de vecteur court dans un réseau euclidien. Masters thesis, ENS Lyon, 2008.

[RR06] Oded Regev and Ricky Rosen. Lattice problems and norm embeddings. In *Proceedings of the Annual Symposium on the Theory of Computing — STOC 2006*, pages 447–456. ACM Press, 2006.

[RV92] Jean-Louis Roch and Gilles Villard. Parallel gcd and lattice basis reduction. In *Proceedings of the Second Joint International Conference on Vector and Parallel Processing*, volume 634 of *LNCS*, pages 557–564. Springer-Verlag, 1992.

[Sch91] Claus-Peter Schnorr. Factoring integers and computing discrete logarithms via diophantine approximations. In *Advances in Cryptology — Eurocrypt 1991*, volume 547 of *LNCS*, pages 281–293, 1991.

[Sch03] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *20th Annual Symposium on Theoretical Aspects of Computer Science — STACS 2003*, volume 2607 of *LNCS*, pages 146–156. Springer-Verlag, 2003.

[SE91] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *FCT '91: Proceedings of the 8th International Symposium on Fundamentals of Computation Theory*, pages 68–85. Springer-Verlag, 1991.

[SG08] Robert Szerwinski and Tim Guneysu. Exploiting the Power of GPUs for Asymmetric Cryptography. In *Cryptographic Hardware and Embedded Systems — CHES 2008*, volume 5154 of *LNCS*, pages 79–99. Springer-Verlag, 2008.

[SH95] Claus-Peter Schnorr and Horst Helmut Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Advances in Cryptology — Eurocrypt 1995*, volume 921 of *LNCS*, pages 1–12. Springer-Verlag, 1995.

[Sho] Victor Shoup. Number theory library (NTL) for C++. `http://www.shoup.net/ntl/`.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *Advances in Cryptology — Asiacrypt 2009*, LNCS. Springer-Verlag, 2009.

[Vil92] Gilles Villard. Parallel lattice basis reduction. In *International Symposium on Symbolic and Algebraic Computation — ISSAC*, pages 269–277. ACM Press, 1992.

[Wet98] Susanne Wetzel. An efficient parallel block-reduction algorithm. In *Algorithmic Number Theory Symposium — ANTS 1998*, volume 1423 of *LNCS*, pages 323–337. Springer-Verlag, 1998.

Publication

On the Claimed Privacy of EC-RAC III

Publication Data

Junfeng Fan, Jens Hermans, and Frederik Vercauteren. On the Claimed Privacy of EC-RAC III. In Siddika Berna Ors Yalcin, editor, *RFIDSec*, volume 6370 of *Lecture Notes in Computer Science*, pages 66–74. Springer, 2010.

Contributions

- Analysis of privacy models.
- One of the attacks

On the Claimed Privacy of EC-RAC III *

Junfeng Fan, Jens Hermans †, and Frederik Vercauteren ‡

Department of Electrical Engineering - COSIC
K.U.Leuven and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. In this paper we show how to break the most recent version of EC-RAC with respect to privacy. We show that both the ID-Transfer and ID&PWD-Transfer schemes from EC-RAC do not provide the claimed privacy levels by using a man-in-the-middle attack. The existence of these attacks voids the presented privacy proofs for EC-RAC.

Keywords: RFID, Protocols, EC-RAC, Privacy

1 Introduction

Radio Frequency Identification (RFID) is a technology that has great potential. It can be used in supply chains, access control, product authentication and so on. The study on RFID has mainly two branches: design of RFID-specific protocols and implementation of security components. The former focuses on design and analysis of cryptographic schemes that can meet various requirements in terms of security and privacy. The latter focuses on low-cost and secure implementations of cryptographic primitives such as hash functions and Public Key Cryptography (PKC).

The EC-RAC (ECDLP Based Randomized Access Control) protocol is a cryptographic protocol designed for RFID systems. It was designed to offer anonymity, which is not offered by conventional ECDLP based protocols such

*This work was supported in part by K.U. Leuven-BOF (OT/06/40), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by FWO project G.0300.07, by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

†Research assistant, sponsored by the Fund for Scientific Research - Flanders (FWO).

‡Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

as the Schnorr [6] and the Okamoto [5] protocol. It was also carefully designed to "minimize the computation workload of a tag" [3]. The first version of the EC-RAC protocol [3] was broken in [7] and [1], while the second version of EC-RAC [4] was broken in [8]. In this paper, we examine the third version of EC-RAC [2] (EC-RAC III) and we show that it does not provide the claimed privacy properties.

The ID&Pwd-Transfer protocols (protocol 2,3) are broken by a (wide) man-in-the-middle attack, and a tag can be traced by the attacker. Since our attacks on the ID&Pwd-Transfer scheme do not require access to the tag's secrets, not even *wide-weak* privacy is provided by the protocols. *Narrow-weak* privacy might be provided by these protocols, but no formal proof for this is included. Also the ID-transfer protocol does not provide the claimed *wide-strong* privacy. An attacker that knows the identity of a certain tag, can always identify this tag using a man-in-the-middle attack. The highest privacy levels that could be provided by the ID-Transfer scheme are *narrow-strong* privacy or *wide-destructive*, although no formal proof for this exists.

The remainder of the paper is structured as follows: in Section 2 we introduce the different versions of EC-RAC in detail and discuss the vulnerabilities of EC-RAC I and EC-RAC II. Section 3 introduces the privacy model of Vaudenay, which is used throughout this paper. In Section 4 we present our attacks on the various schemes of EC-RAC III and discuss the impact on the claimed privacy properties of the protocol.

2 The EC-RAC Protocols

The basic setup considered in this paper is a world consisting of several tags and a single reader (or multiple connected to a central server). The reader/server is assumed trusted and the goal of the protocols is to authenticate the tag to the reader and, at the same time, protect the identity of the tag. Intuitively, it should be impossible for an adversary to impersonate a tag and it should be impossible for the adversary to derive any information on the identity of tags involved.

2.1 EC-RAC I/II and related attacks

The first version of the EC-RAC protocol was proposed in [3]. EC-RAC consists of several sub-protocols: ID-transfer, Pwd-Transfer and server authentication. The ID-transfer protocol allows the tag to identify itself to the server, the Pwd-

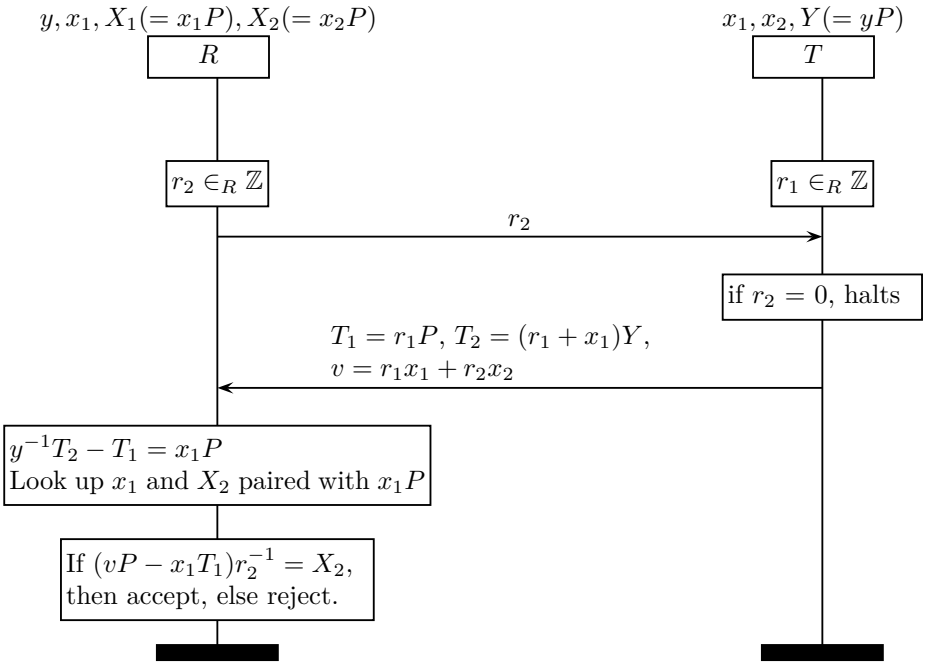
Transfer protocol allows the tag to authenticate to the server. The two can be combined into the Id&Pwd-Transfer protocol. Figure 1 shows the ID&Pwd-Transfer protocol of EC-RAC I. Upper case symbols denote elliptic curve points, lower case symbols denote scalars.

This scheme was broken in [7] and [1], which show that a tag could be traced by an attacker using a quality-time attack [7]. If an attacker runs the protocol twice with the same r_2, collecting $\{v, T_1\}$ and $\{v', T_1'\}$, she can then derive

$$(v - v')^{-1}(T_1 - T_1') = x_1^{-1}P$$

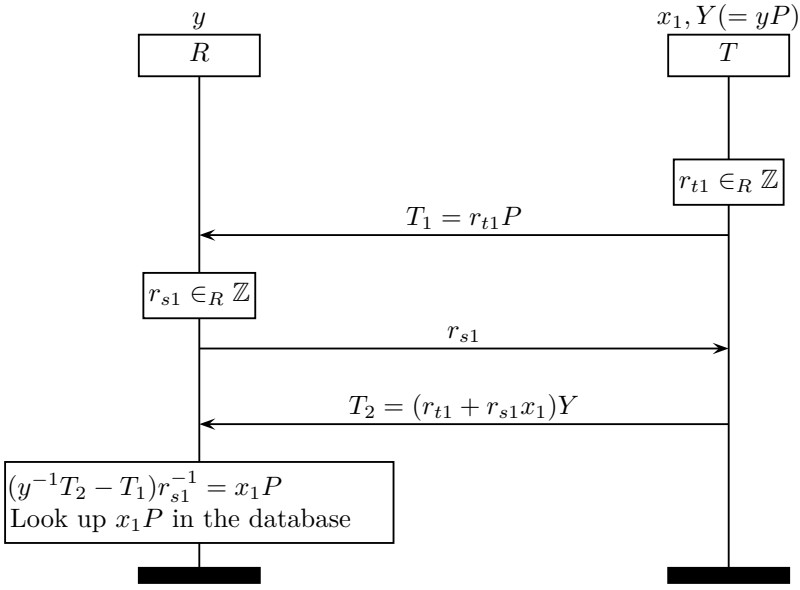
which is a unique attribute of a tag. This unique attribute can then be used to identify the tag.

Figure 1: ID&Pwd-Transfer protocol from EC-RAC I [3]



EC-RAC II [4] introduced three different sub-protocols: ID-transfer, Pwd-Transfer and server authentication. These sub-protocols were combined into several protocols. Figure 2 shows the ID transfer protocol.

Figure 2: ID-Transfer protocol from EC-RAC II [4]



EC-RAC II was broken in [8]. The ID-transfer scheme was broken with respect to untraceability using a man-in-the-middle attack, in which the attacker uses a previous, valid execution of the protocol to modify the communication. If the reader accepts the modified values, the attacker can identify the previously eavesdropped tag.

One of the fundamental problems is that protocols, which in isolation are secure and/or untraceable, are not necessarily secure and/or privacy preserving when combined. The ID&Pwd-Transfer protocols were broken with respect to tag-to-server authentication, allowing the attacker to impersonate a tag. The main cause of this attack is the reuse of the same randomness for both the ID- and Pwd-Transfer sub-protocol.

2.2 EC-RAC III

In [2] Lee, Batina, Singelée and Verbauwhede present an improved version of EC-RAC. The paper [2] claims that the ID-transfer protocol (protocol 1 from [2]) and the ID&Pwd-Transfer protocol (protocol 3 from [2]) provide *wide-strong* privacy (see Section 3 for definition).

Let P be a generator of the elliptic curve group. Every tag has two private-public key pairs $x_1, X_1 = x_1P$ and $x_2, X_2 = x_2P$. In this case x_1 serves as the identity of the tag and is also known by the reader. The reader has a private-public key pair $y, Y = yP$.

Figure 3 shows the ID-transfer protocol from [2]. This protocol should identify the tag as x_1 in a secure and *wide-strong* privacy preserving way. The main difference with the previous versions of the protocol is the introduction of the non-linearity $\dot{r}_s = x(r_sP)$, with $x(\cdot)$ the x-coordinate function for an elliptic curve point.

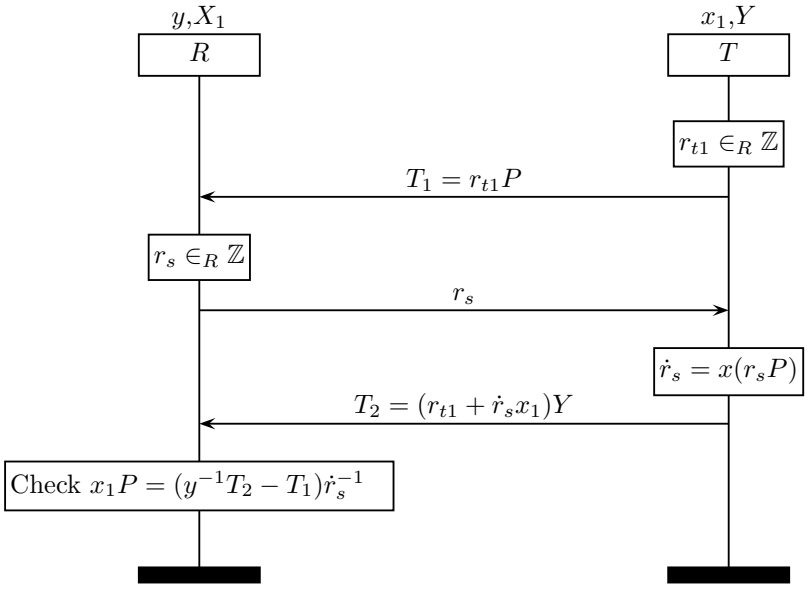
Figure 4 shows the ID&Pwd-Transfer protocol from [2]. In addition to the reader identifying the tag correctly as x_1, it also authenticates the tag using the public-private key pair $x_2, X_2 = x_2P$. (Note that the secret x_1 is known to both the tag and the reader and cannot be used for authentication.)

3 Privacy Models

Throughout this paper we use the privacy model from Vaudenay [9]. This model describes several oracles available to the attacker. For a complete list we refer to the original paper. Basically the attacker has the ability to perform a man-in-the-middle attack on any tag that is within its vincinity: it can influence all communication between tag and reader. The attacker also gets the result of the authentication of a tag, i.e. whether the reader accepts the tag or not. The attacker can also draw (at random) and free tags, moving them in and out of its range. During all of these interactions the attacker has to use a virtual identity to refer to the tags in its vincinity, i.e. it does not need to know the real identity to interact with a chosen tag. Finally the attacker can corrupt tags, reading out the entire internal state of the tag.

A *strong* attacker is allowed to use all the oracles available. A *destructive* attacker cannot use a tag anymore after it has been corrupted, i.e. corruption destroys the tag. In case of a *forward* attacker, the attacker can only do other corruptions after the first corruption. No protocol interactions are allowed after the first corrupt. A *weak* attacker does not have the ability to corrupt tags.

Figure 3: ID-transfer protocol (Protocol 1) from [2]

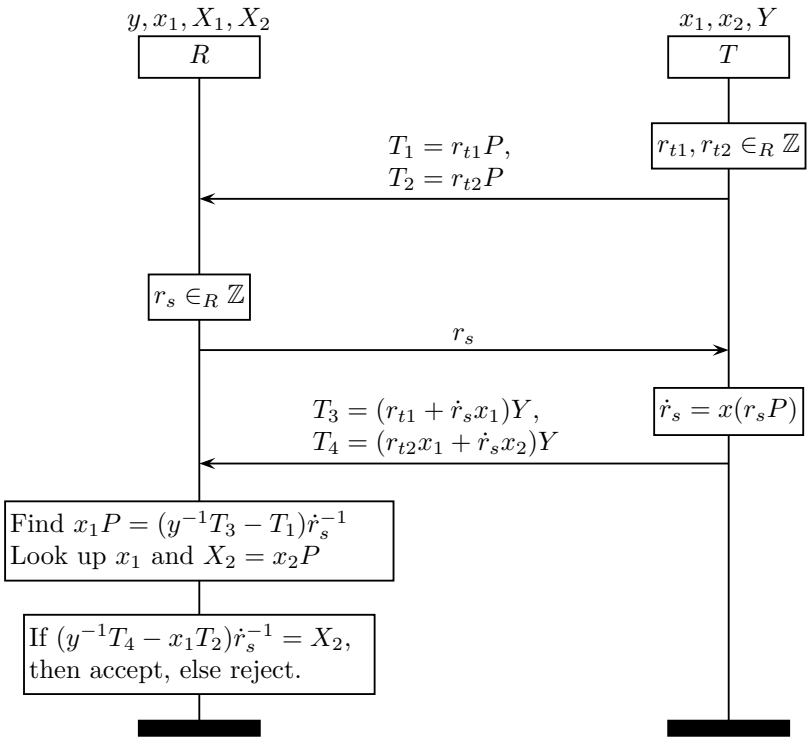


Orthogonal to these four attacker classes there is the notion of *wide* and *narrow* attackers. A *wide* attacker has access to the result of the verification by the server while a *narrow* attacker does not.

Definition 1. (Simplified version of Definition 6 from [9]) Privacy - A protocol is called P-private, with P an adversary class from above (strong, destructive,...), if all adversaries belonging to the class P are trivial.

Intuitively, an adversary is called trivial if it produces the same output, even when all protocol oracles are blinded (i.e. the attacker does not 'use' the communication captured during the protocol run to determine its output). Since the attacks presented in this paper allow tracing of tags, they clearly violate the privacy property, because the output of the attacker depends on information from the protocol runs that the attacker executes. As such, we do not require any detailed elements of the privacy definition used by Vaudenay.

Figure 4: ID&Pwd-Transfer protocol (protocol 3) from [2]



The equations below show the most important relations between the privacy notions above:

$$\begin{array}{ccccccc} \text{Wide Strong} & \Rightarrow & \text{Wide Destructive} & \Rightarrow & \text{Wide Forward} & \Rightarrow & \text{Wide Weak} \\ \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ \text{Narrow Strong} & \Rightarrow & \text{Narrow Destructive} & \Rightarrow & \text{Narrow Forward} & \Rightarrow & \text{Narrow Weak} \end{array}$$

In this case $A \Rightarrow B$ means that if the protocol is A-private it implies that the protocol is B-private. It should be obvious that a protocol that is e.g. *Wide Strong* private will also belong to all other privacy classes above, that only allow weaker adversaries.

Besides privacy the protocol should also offer authentication of the tag. We refer to this property as the *security* of the protocol.

Definition 2. (Simplified version of Definition 4 from [9]) Security - We consider any adversary in the class *strong*. The adversary wins if the reader identifies an uncorrupted legitimate tag, but the tag and the reader did not have a matching conversation. The RFID Scheme is called secure if the success probability of any such adversary is negligible.

4 Attacks on the Protocols

The main flaw in the ID&Pwd-Transfer scheme is the fact that the "hash" of the challenge, i.e. $\dot{r}_s$ does not mask all of the secret keys x_1 and x_2. Indeed, in the response T_4, the x_1 part is only masked by the randomness r_{t2}.

4.1 First Attack

The first attack exploits the fact that it is possible to force $\dot{r}_s$ to become 0. Indeed, note that the protocol does not verify whether r_s is a multiple of the order of P. As such, it is possible for an attacker impersonating a reader to send $r_s = k \cdot \mathrm{ord}(P)$ to the tag, who will then compute $\dot{r}_s = x(r_s P) = 0$ and therefore return $T_3 = r_{t1}Y$ and $T_4 = r_{t2}x_1 Y$. Using the messages ($T_1 = r_{t1}P$, $T_2 = r_{t2}P$, $T_3 = r_{t1}Y$, $T_4 = r_{t2}x_1 Y$), it is then possible to mount a man-in-the-middle attack on a second communication to test whether the same tag from the first run is present or not. This attack is described in Figure 5 where the tag's secret keys are now denoted by x'_1 and x'_2.

The adversary adds T_1 and T_2 to the messages T'_1 and T'_2 obtained from the unknown tag and forwards these to the reader. The reader responds with a nonce r'_s, which the attacker simply forwards to the tag. The tag responds with valid messages T'_3 and T'_4 which the attacker uses to obtain $T''_3 = T'_3 + T_3$ and $T''_4 = T'_4 + T_4$ and sends these to the reader. The reader then computes

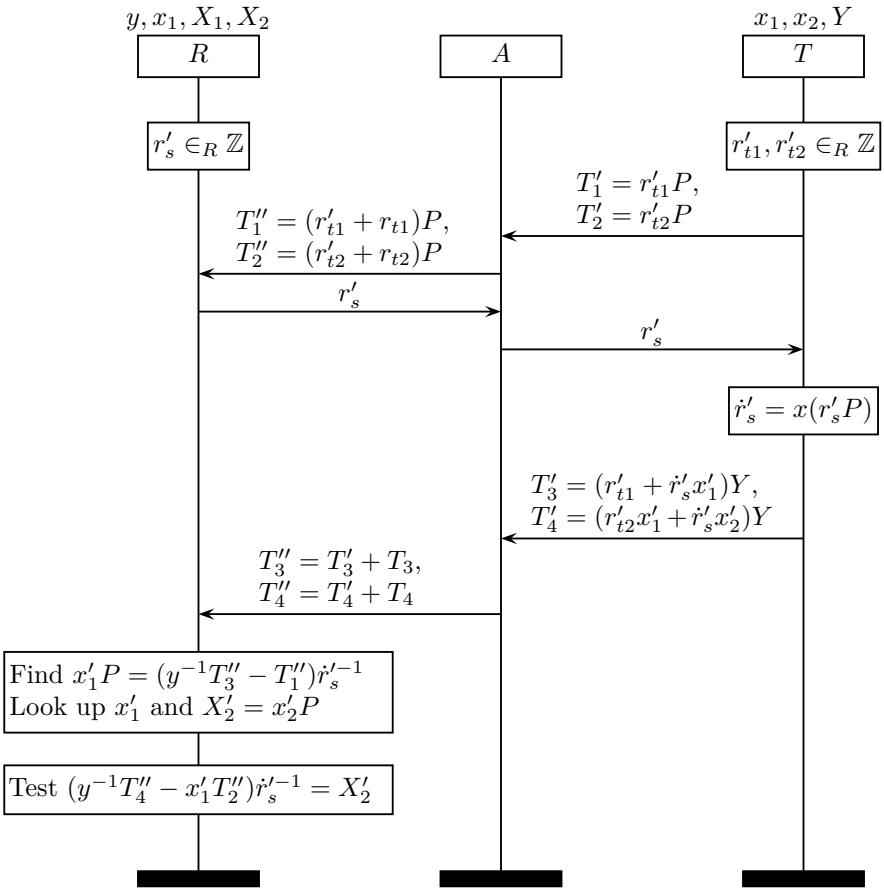
$$(y^{-1}T''_3 - T''_1)\dot{r}'^{-1}_s = (r_{t1} + r'_{t1} + \dot{r}'_s x'_1 - r_{t1} - r'_{t1})\dot{r}'^{-1}_s P = x'_1 P\,,$$

and looks up x'_1 and $X'_2 = x'_2 P$. Note that this step always verifies. The reader then tests whether $(y^{-1}T''_4 - x'_1 T''_2)\dot{r}'^{-1}_s = X'_2$, which is equivalent with

$$(r'_{t2}x'_1 + \dot{r}'_s x'_2 + r_{t2}x_1 - x'_1(r'_{t2} + r_{t2}))\dot{r}'^{-1}_s P = x'_2 P\,.$$

The test will succeed if and only if $x_1 = x'_1$, i.e. if the tag is the same as the one from the first run.

Figure 5: Man-in-the-middle attack on protocols 2 and 3



4.2 Second Attack

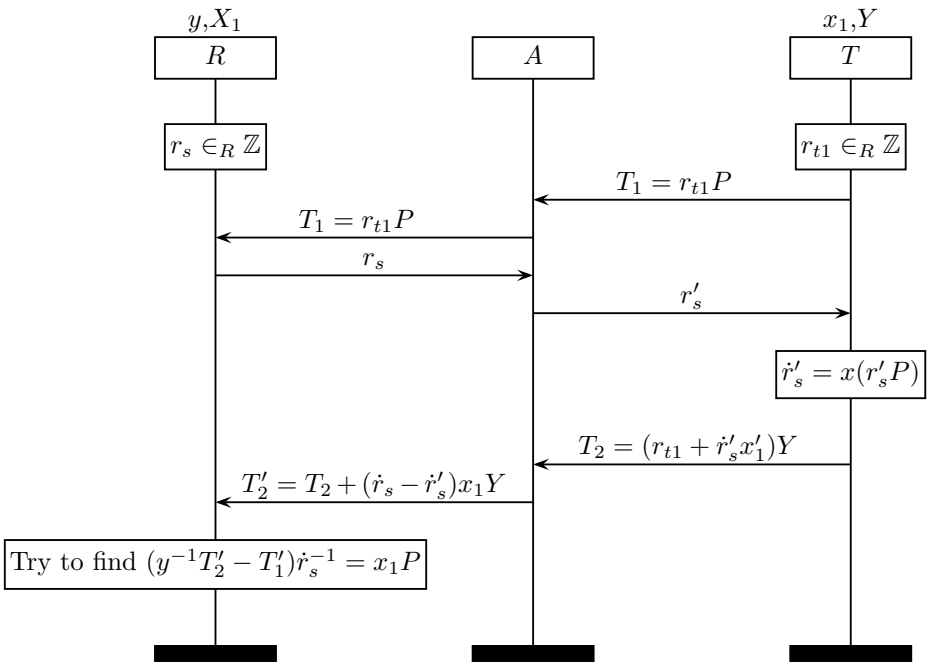
The second attack even works when the tag adds an extra verification that $\dot{r}_s \neq 0$. Note that the first attack worked because the attacker obtained ($T_1 = r_{t1}P$, $T_2 = r_{t2}P$, $T_3 = r_{t1}Y$, $T_4 = r_{t2}x_1Y$), so it suffices to explain how such a tuple can be obtained when the tag verifies whether $\dot{r}_s \neq 0$. In fact, obtaining such a tuple is trivial by querying the tag twice with the same r_s and subtracting

the results, since the parts involving $\dot{r}_s$ will cancel out. As such we obtain a valid tuple ($T_1^* = r_{t1}^* P$, $T_2^* = r_{t2}^* P$, $T_3^* = r_{t1}^* Y$, $T_4^* = r_{t2}^* x_1 Y$), which can then be used in the first attack.

4.3 Third Attack

The third attack shows that the ID-transfer scheme (protocol 1 from [2]) is not wide-strong. A *strong* attacker is able to read a tag's ID x_1 without destroying the tag. We will now show how a *strong* attacker can track a particular tag using a man-in-the-middle attack.

Figure 6: Man-in-the-middle attack on protocol 1



This attack is described in Figure 6. By definition of *strong*, the attacker knows x_1 of a certain tag. In order to test if a random tag is the corrupted one, she

plays a man-in-the-middle attack as follows. The attacker replaces the value r_s with another random value r'_s and replaces $T_2 = (r_{t1} + \dot{r}'_s x'_1)Y$ by

$$T'_2 = T_2 + (\dot{r}_s - \dot{r}'_s)x_1 Y = (r_{t1} + \dot{r}'_s(x'_1 - x_1) + \dot{r}_s x_1)Y$$

The reader will accept this only if $x_1 = x'_1$ (provided $\dot{r}'_s \neq 0$, which the attacker can assure). This allows the attacker to identify the tag x_1 upon acceptance by the reader. The ID-transfer protocol is thus not *wide-strong* private. Since our attacker is both *wide* and *strong*, the ID-transfer might be *narrow-strong* private or *wide-destructive* private, although no proof for this is given in the original paper.

5 Conclusions

In this paper we have shown three successful attacks on the latest version of EC-RAC [2]. We prove that the ID&PWD-Transfer scheme is not *wide-strong* private and is not even *wide-weak* private. The highest possible privacy level that might be achieved by the ID&PWD-Transfer scheme is *narrow-strong* privacy.

We also prove that the ID-transfer scheme is not *wide-strong* private as claimed and can be at most *wide-destructive* or *narrow-strong* private.

References

[1] J. Bringer, H. Chabanne, and T. Icart. Cryptanalysis of EC-RAC, a RFID identification protocol. In *CANS*, volume 5339 of *Lecture Notes in Computer Science*, pages 149–161. Springer, 2008.

[2] Y. K. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-cost untraceable authentication protocols for RFID. In S. Wetzel, C. Nita-Rotaru, and F. Stajano, editors, *WISEC*, pages 55–64. ACM, 2010.

[3] Y. K. Lee, L. Batina, and I. Verbauwhede. EC-RAC (ECDLP Based Randomized Access Control): Provably Secure RFID authentication protocol. In *IEEE International Conference on RFID 2008*, pages 97–104, Las Vegas,NA,USA, 2008. IEEE.

[4] Y. K. Lee, L. Batina, and I. Verbauwhede. Untraceable RFID Authentication Protocols: Revision of EC-RAC. In *IEEE International Conference on RFID 2009*, pages 178–185, Orlando,FL,USA, 2009. IEEE.

[5] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.

[6] C. P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, New York, NY, USA, 1989. Springer.

[7] T. van Deursen and S. Radomirovic. Attacks on RFID protocols. Cryptology ePrint Archive, Report 2008/310, 2008. `http://eprint.iacr.org/`.

[8] T. van Deursen and S. Radomirovic. Untraceable RFID protocols are not trivially composable: Attacks on the revision of EC-RAC. Cryptology ePrint Archive, Report 2009/332, 2009. `http://eprint.iacr.org/`.

[9] S. Vaudenay. On privacy models for RFID. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2007.

Publication

A New RFID Privacy Model

Publication Data

Jens Hermans, Andreas Pashalidis, Frederik Vercauteren, and Bart Preneel. A New RFID Privacy Model. In Vijay Atluri and Claudia Diaz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 568–587. Springer, 2011.

Contributions

- Principal author.

A New RFID Privacy Model *

Jens Hermans †, Andreas Pashalidis, Frederik Vercauteren ‡, and Bart Preneel

Department of Electrical Engineering - COSIC
Katholieke Universiteit Leuven and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. This paper critically examines some recently proposed RFID privacy models. It shows that some models suffer from weaknesses such as insufficient generality and unrealistic assumptions regarding the adversary's ability to corrupt tags. We propose a new RFID privacy model that is based on the notion of indistinguishability and that does not suffer from the identified drawbacks. We demonstrate the easy applicability of our model by applying it to multiple existing RFID protocols.

Keywords: RFID, authentication, identification, privacy model

1 Introduction

As Radio Frequency Identification (RFID) systems are becoming more common (for example in access control [2, 29], product tracking [2], e-ticketing [26, 29], electronic passports [18]), managing the associated privacy and security concerns becomes more important [33]. Since RFID tags are primarily used for authentication purposes, 'security' in this context means that it should be infeasible to 'fake' a legitimate tag. 'Privacy', on the other hand, means that adversaries should not be able to identify, trace, or link tag appearances.

*This work was supported in part by (a) the Research Council K.U.Leuven: GOA TENSE (GOA/11/007), (b) the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), (c) the 'Trusted Architecture for Securely Shared Services' (TAS3) project, supported by the 7th European Framework Programme with contract number 216287, and (d) the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

†Research assistant, sponsored by the Fund for Scientific Research - Flanders (FWO).

‡Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

Several models for privacy and security in the context of RFID systems have been proposed in the literature. In this paper, we critically examine some of these models. In particular, we focus on *general* models [1]. For some of these models we show that, despite their intended generality, it remains unclear how to apply them to protocols other than the protocol in the context of which they were proposed. Other existing models do not support adversaries that can tamper with tags. However, considering such adversaries is important because, as low-cost devices, tags are hardly protected against physical tampering. In particular, it has been shown that side-channel attacks may enable an adversary to extract secrets from the tag [17, 20, 21, 25], and so-called 'reset' attacks force the tag to re-use old randomness [4, 10, 15]. The adversary can mount reset attacks by inducing power drops or by otherwise influencing the physical environment of the tag. Adversaries that can tamper with tags are therefore realistic.

Subsequently we propose a new model that borrows concepts from previous models, including virtual tag references, the corruption model that Vaudenay [31] introduced and the notion of 'narrow' and 'wide' adversaries to construct a new model. We believe that the new model is easier to apply. Also note that, although presented as a model for RFID privacy, it is not limited to the RFID setting; the model may also apply to other setups, in which the participants should not be identifiable or linkable.

Structure of the paper Section 2 introduces the basic definitions for RFID systems and some notation. Section 3 discusses a selection of existing models, their underlying assumptions, their usability, and some further technicalities. Section 4 presents our model for RFID privacy which is then applied to some of the stronger existing RFID protocols in Section 5. In the appendices, our model is extended to a multi-indistinguishability setup, which allows multi-bit challenges. Mutual authentication is also discussed there.

2 Definitions

Throughout this paper we use a common model for RFID systems, similar to the definitions introduced in [9,31]. An RFID system consists of a set of tags $\mathcal{T}$, and a reader R. Each tag is identified by an identifier ID. The memory of the tags contains a state S, which may change during the lifetime of the tag. The

[1] We do not discuss some of the early proposals that were made in the context of one specific protocol.

tag's ID may or may not be stored in S. Each tag is a transponder with limited memory and computation capability.

Tags can also be corrupted: the adversary has the capability to extract secrets and other parts of the internal state from the tags it chooses. The reader R consists of one or more transceivers and a central database. The reader's task is to identify legitimate tags (i.e. to recover their IDs), and to reject all other incoming communication. The reader has a database that contains for every tag, its ID and a matching secret K.

Definition 1 (RFID Framework [31])**.** An RFID scheme consists of the following algorithms:

- $SetupReader(1^k)$: setup the reader by generating the necessary keys, depending on the security parameter k. The function returns the public and private keys of the reader. Public keys are assumed to be publicly released by the algorithm, private keys are stored in the reader.
- $SetupTag(ID)$: return the tag specific secret K and the initial state S of the tag. The pair (ID, K) will be stored in the reader, the state S in the tag. Note that K is not necessarily stored in the tag, but the definition of the protocol might include K in the state S.
- $Protocol$: a polynomial-time interactive protocol between a reader and a tag. The reader ends with a tape output.

All the models discussed below fit the above general RFID system definition.

A function $f : \mathbb{N} \to \mathbb{R}$ is called 'polynomial' in the security parameter $k \in \mathbb{N}$ if $f(k) = O(k^n)$, with $n \in \mathbb{N}$. It is called 'negligible' if, for every $c \in \mathbb{N}$ there exists an integer k_c such that $f(k) \leq k^{-c}$ for all $k > k_c$. We denote a negligible function by ϵ.

If $\mathcal{T}$ is a set, $t \in_R \mathcal{T}$ means that t is chosen uniformly at random from $\mathcal{T}$. $|\mathcal{T}|$ denotes the cardinality of the set. If $\mathcal{A}$ is an algorithm, then $\mathcal{A}^{\mathcal{O}}$ denotes the fact that $\mathcal{A}$ has access to the oracle $\mathcal{O}$.

3 Existing Privacy Models

This section discusses certain existing RFID privacy models. Most models feature a correctness (no false negatives), security (no false positives) and privacy definition.

Note that covering all existing models would exceed the scope of this paper by far. Many models, including the ones introduced in [3,8,11,14,16,19,30] do not allow corrupted tags to be traced. We have selected two such models [14, 19] for further discussion, in addition to the stronger models of Vaudenay [31] and Canard et al. [9].

3.1 Vaudenay

Several concepts from the privacy model introduced by Vaudenay [31] are used in our model. We therefore present this in detail.

Adversarial Model

The adversary of the Vaudenay model has the ability to influence all communication between a tag and the reader and can therefore perform man-in-the-middle attacks on any tag that is within its range. It may also obtain the result of the authentication of a tag, i.e. whether the reader accepts or rejects the tag. The adversary may also 'draw' (at random) tags and then 'free' them again, moving them inside and outside its range. During these interactions the adversary has to use a virtual identifier (not the tag's real ID) in order to refer to the tags that are inside its range. Finally the adversary may corrupt tags, thereby learning their entire internal state.

The above interactions take place over eight oracles that the adversary may invoke: `CreateTag(ID)`, `DrawTag(distr)` $\rightarrow$ $(vtag)$, `Free(`$vtag$`)`, `Launch` $\rightarrow$ π, `SendReader(`m, π`)` $\rightarrow$ m', `SendTag(`$m, vtag$`)` $\rightarrow$ m', `Result(`π`)` $\rightarrow$ x and `Corrupt(`$vtag$`)`. $vtag$ denotes a virtual tag reference, π a protocol instance, $distr$ a polynomially bounded sampling algorithm, m and m' messages and ID a tag ID. For a complete definition of the oracles the reader is referred to [31].

The Vaudenay model divides adversaries into different classes, depending on restrictions regarding their use of the above the oracles. In particular, a *strong* adversary may use all eight oracles without any restrictions. A *destructive* adversary is not allowed to use a tag after it has been corrupted. This models situations where corrupting a tag leads to the destruction of the tag. A *forward* adversary can only do other corruptions after the first corruption. That is, no protocol interactions are allowed after the first corrupt. A *weak* adversary does not have the ability to corrupt tags. Orthogonal to these four attacker classes there is the notion of *wide* and *narrow* adversary. A *wide* adversary has access to the result of the verification by the server while a *narrow* adversary does not.

Due to their generality, the above restrictions can be used perfectly in other privacy models. Throughout the paper we will frequently refer to strong, destructive, forward, weak and wide/narrow adversaries.

The equations below show the most important relations between the above privacy notions:

$$\begin{array}{ccccccc} \text{Wide Strong} & \Rightarrow & \text{Wide Destructive} & \Rightarrow & \text{Wide Forward} & \Rightarrow & \text{Wide Weak} \\ \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\ \text{Narrow Strong} & \Rightarrow & \text{Narrow Destructive} & \Rightarrow & \text{Narrow Forward} & \Rightarrow & \text{Narrow Weak} \end{array}$$

In this case $A \Rightarrow B$ means that if the protocol is A-private it implies that the protocol is B-private. A protocol that is *Wide Strong* private, for example, obviously also belongs to all other privacy classes, that only allow weaker adversaries.

Privacy, Security and Correctness

In general, an RFID protocol should satisfy (a) correctness (a 'real' tag is always accepted), (b) security (fake tags are rejected) and (c) privacy (tags cannot be identified or traced). Privacy is defined by means of the notion of a 'trivial' adversary. Intuitively, a trivial adversary does not 'use' the communication captured during the protocol run to determine its output.

Definition 2 (Blinder, trivial adversary - Simplified version of Definition 7 from [31])**.** A Blinder $\mathcal{B}$ for an adversary $\mathcal{A}$ is a polynomial-time algorithm which sees the messages that $\mathcal{A}$ sends and receives, and simulates the `Launch`, `SendReader`, `SendTag` and `Result` oracles to $\mathcal{A}$. The blinder does not have access to the reader tapes. A blinded adversary $\mathcal{A}^{\mathcal{B}}$ is an adversary who does not use the `Launch`, `SendReader`, `SendTag` and `Result` oracles.

An adversary $\mathcal{A}$ is trivial if there exists a blinder $\mathcal{B}$ such that $|\Pr(\mathcal{A}\,\text{wins}) - \Pr(\mathcal{A}^{\mathcal{B}}\,\text{wins})|$ is negligible.

Intuitively, an adversary is called trivial if, even when blinded, it still produces the same output. Such an adversary does not 'use' the communication captured during the protocol run in order to determine its output. Note that a blinded adversary is not the same as a simulator typically found in security proofs: the blinder is separate from the adversary and has no access to the adversary's tape. The blinder just receives incoming queries from the adversary and has to respond either by itself or by forwarding the queries to the system.

We are now ready to present the privacy definition.

Definition 3 (Privacy - Simplified version of Definition 6 from [31])**.** The privacy game between the challenger and the adversary consists of two phases:

1. Attack phase: the adversary issues oracle queries according to applicable restrictions
2. Analysis phase: the adversary receives the table that maps every *vtag* to a real tag ID. Then it outputs `true` or `false`.

The adversary wins if it outputs `true`. A protocol is called P-private, where P is an adversary class (strong, destructive, ...), if and only if all winning adversaries that belong to the class P are trivial.

Besides privacy the protocol should also offer authentication of the tag. We refer to this property as the *security* of the protocol.

Definition 4 (Security - Simplified version of Definition 4 from [31])**.** We consider any adversary in the class *strong*. The adversary wins if the reader identifies an uncorrupted legitimate tag, but the tag and the reader did not have a matching conversation. The RFID scheme is called secure if the success probability of any such adversary is negligible.

Definition 5 (Correctness - Definition 1 from [31])**.** An RFID scheme is correct if its output is correct except with negligible probability for any polynomial-time experiment which can be described as follows:

1. set up the reader
2. create a number of tags including a subject one named ID
3. execute a complete protocol between reader and tag ID

The output is correct if and only if $Output = \perp$ and tag ID is not legitimate or $Output = ID$ and tag ID is legitimate.

In a follow-up paper [24] to the Vaudenay paper, the concept of mutual authentication for RFID is defined. The tag simply outputs a boolean, indicating whether or not the reader was accepted. The authors extend the security definition by adding a criterion for reader authentication.

Discussion

The paper of Vaudenay inspired many authors to formulate derived RFID privacy models or to evaluate the (Paise-)Vaudenay model [7, 9, 12, 13, 22–24, 27, 28]. Although Vaudenay's privacy model is perhaps the strongest and most complete, it contains some flaws with respect to strong privacy.

Vaudenay's proof of the statement that 'strong privacy is impossible' uncovers some of these flaws. This proof assumes a destructive private protocol. By definition, for every destructive adversary, there exists a blinder. This includes the adversary that (a) creates one real tag, (b) corrupts this tag right away, (c) starts a protocol using either the state from the corrupted tag or from another fake tag. In the end, the blinder has to answer the `Result` oracle. Obviously, the adversary knows which tag was selected and knows which result to expect. However, since the blinder has no access to this random coin of the adversary, it must be able to distinguish a real and a fake tag just by looking at the protocol run from the side of the reader. The proof then uses this blinder to construct a strong adversary. Since all strong adversaries are also destructive, this proves the impossibility of strong privacy.

Obviously, this proof only works because the blinder is separated from the adversary. In later work [32], Vaudenay corrects the inconsistency in the model and shows that strong privacy is indeed possible. In this new approach, the blinder is given access to the random coin flips of the adversary. The issue with a separate blinder is exploited multiple times by Armknecht et al. in [1]. Using this property the authors show the impossibility of reader authentication combined with respectively narrow forward privacy (if `Corrupt` reveals the temporary state of tags) and narrow strong privacy (if `Corrupt` only reveals the permanent state of tags).

Independent from this correction, Ng et al. [22] also identified the problems with strong privacy. They propose a solution, based on the concept of a 'wise' adversary that does not make any 'irrelevant' queries to the oracles i.e. queries to which it already knows the answer. The authors claim that, if the protocol does not generate false negatives, then a wise adversary never calls the `Result` oracle. Given the vague definition of wise adversaries it is hard to verify these claims. The existence of attacks which exploit false positives [5] however, suggests that the general claim that `Result` is not used by a wise adversary is incorrect. Based on this questionable general claim, the authors further identify an IND-CPA-based protocol as being strong private, without giving a formal proof. [2]

[2]Note that the original security proof (i.e. no false positives) by Vaudenay requires IND-CCA2 encryption, so using only IND- CPA encryption would require a new security proof. The `Result` may therefore serve as a decryption oracle.

3.2 Canard et al.

Model

The model of Canard et al. [9] builds on the work of Vaudenay, so the definition of oracles is quite similar. For the privacy definition the model requires the adversary to produce a non-obvious link between virtual tags.

Definition 6. $(vtag_i, vtag_j)$ is a non-obvious link if $vtag_i$ and $vtag_j$ refer to the same ID and if a 'dummy' adversary, who only has access to `CreateTag`, `Draw`, `Free`, `Corrupt`, is not able to output this link with a probability better than $1/2$. [3]

One major difference with respect to Vaudenay's model is that a 'dummy' adversary is used instead of a blinded adversary. This avoids some of the issues surrounding the use of a blinder, because a 'dummy' adversary can also access its own random tape, while a blinder cannot access the adversary's random tape.

The definition requires the adversary to output a non-obvious link. A protocol is said to be untraceable if, for every adversary $\mathcal{A}$, it is possible to construct a 'dummy' adversary $\mathcal{A}_d$ such that $|\mathbf{Succ}_{\mathcal{A}}^{Unt}(1^k) - \mathbf{Succ}_{\mathcal{A}_d}^{Unt}(1^k)| \leq \epsilon(k)$.

Discussion

While the work certainly has its merit in formalizing and fixing the Vaudenay model (by using a dummy adversary instead of a blinder), the model of Canard et al. lacks generality because it focuses on non-trivial links. Other relevant properties, which do not imply the leakage of a non-trivial link, are not considered a privacy breach. For example, the cardinality of the set of active tags can be leaked without leaking a non-trivial link. Because of the limited scope of untraceability, we are not using this model.

[3] It is unclear why the authors use the probability threshold $1/2$, since one would expect some dependency on the total number of non-obvious links. One slightly different interpretation is that a 'dummy' adversary cannot determine if a given non-obvious candidate link $vtag_i$, $vtag_j$ is a link in reality or not.

3.3 Deng, Li, Yung and Zhao

Model

Deng et al. presented their RFID Privacy Framework in [14].

The correctness ('adaptive completeness') definition used by Deng et al. is more elaborate than Vaudenay's definition. In particular, it allows the adversary to execute multiple complete protocol runs. This captures 'desynchronization' attacks where the adversary communicates a number of times with a tag (without involvement of the reader), in order to desynchronize the tag's state such that it will no longer be recognised by the reader.

The security definition considers both tag-to-reader and reader-to-tag authentication. The definition is similar to Vaudenay's since it requires matching sessions at reader and tag side. In Deng et al.'s model the last message is always sent by the reader, so an adversary could just prevent the tag from finishing the protocol by dropping this last message. Deng et al. therefore define the notion of 'matching sessions' such that last message attacks do not breach security. Vaudenay omits an exact definition of 'matching sessions', and therefore issues like the last message attack are not captured.

While the correctness and security definitions of Vaudenay and Deng et al. appear to be, to a large extent, equivalent, there is a significant discrepancy in the privacy definitions. Firstly, there is no notion of virtual tags in Deng et al.'s model; instead the adversary can refer to all tags using their real identifiers. Secondly, the adversary cannot create new tags. Thirdly, Deng et al. apply a zero-knowledge proof instead of Vaudenay's blinder construction. Informally stated, in the zero-knowledge experiment, the adversary (in the real world) consists of these phases:

1. Standard interaction using the oracles.
2. Select one tag at random (the 'challenge' tag) from the set of clean (non-corrupted and non-active) tags.
3. Interaction using the oracles, except that the adversary can only interact with the non-clean tags and the challenge tag. Moreover, the challenge tag cannot be corrupted.
4. Output a view from the previous step and the index of the challenge tag.

The simulated world is the same, except that, in the third phase, the adversary cannot access the challenge tag. If all PPT adversaries can

be simulated such that the output of the adversary and simulator are computationally/statistically indistinguishable, then the protocol is considered zk-private. This implies that for all adversaries the output can actually be derived without interacting with the challenge tag (as the simulator does).

Discussion

Because of the very specific restrictions imposed in the third phase, this model is significantly weaker than Vaudenay's. Firstly, the model focuses on deriving information about a specific challenge tag (selected by the adversary), while in Vaudenay's model *any* statement that reveals information on the underlying identity of any of the tags is considered a privacy breach. Secondly, the adversary's ability to corrupt tags is limited. In Vaudenay's (corrected) strong privacy model one could prove that a protocol satisfies the privacy definition even if the 'challenge' tag is corrupted. The restriction that the challenge tag must be clean is, according to the authors, introduced to ensure that the tag is not stuck halfway a protocol run. Otherwise one can trivially distinguish the challenge tag by checking whether or not it responds to the remainder of the protocol run. Since a protocol run takes only a short timespan, obviously linking two protocol messages from the same run to the same tag should not be considered a privacy breach. However, we believe that, for the purposes of excluding this as a privacy breach, the concept of virtual tags is more suitable than overly limiting the adversary's corruption abilities in this manner.

The zero-knowledge private protocol proposed in [14] uses a counter as the tag state. The value of this counter is incremented after each protocol run completed by the tag. Obviously, this protocol does not satisfy the privacy definition if the adversary can corrupt the targeted tag, because the adversary learns the value of the counter (and the key) and, by decrementing the value of the counter, it can identify previous protocol runs of the targeted tag. The model in [14] has however been specifically tuned to disallow corruption of the challenge tag, which is a rather unrealistic assumption and thus undermines the significance of the claims that follow from its application.

The security and correctness definitions are more rigorous than Vaudenay's, so they can be a valuable alternative to them.

Experiment $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}$:

1. Setup:
   - Generate n random keys key_i.
   - Initialize the reader with the random key_i.
   - Create n tags, each with a key_i.
2. Phase (1): Learning
   - $\mathcal{A}$ can interact with a polynomial number of calls to the system, but can only issue `SetKey` on $n-2$ tags, leaving at least 2 uncorrupted tags
3. Phase (2): Challenge
   - $\mathcal{A}$ selects two uncorrupted tags T_0 and T_1. Both are removed from the set of tags.
   - One of these tags (T_b, the challenge tag) will be selected at random by the challenger.
   - $\mathcal{A}$ can make a polynomial number of calls to the system, but cannot corrupt the challenge tag T_b.
   - $\mathcal{A}$ outputs a guess bit $g \in \{0,1\}$.

Figure 1: Privacy experiment from [19].

3.4 Juels-Weis

Model

The Juels-Weis model [19] is based on the notion of indistinguishability. The model does not feature a `DrawTag` query and the `Corrupt` query is replaced by a `SetKey` query, which returns the current secret of the tag and allows the adversary to set a new secret. Figure 3.4 shows a simplified version of the privacy game. The protocol is considered private if $\forall\mathcal{A}$, $\Pr\left[\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}} \text{guesses } b \text{ correctly}\right] \leq \frac{1}{2} + \epsilon$

Discussion

The Juels-Weis model is one of the few models that are based on a simple indistinguishability game instead of the notion of simulatability. The model is limited by the fact that the challenge tags cannot be corrupted. In terms of the model in [31] it would be a Weak adversary with regard to the challenge tags. For example, attacks in which the adversary links together executions of a tag that have taken place prior to its corruption are not possible in the Juels-Weis model because of this.

The model from [16] is very similar, with the difference that the privacy is defined as distinguishing the reply from a real tag from a random reply.

3.5 Bohli-Pashalidis

Model

Unlike the previous models, the Bohli-Pashalidis model [6] is not an RFID-specific model. Unfortunately, it captures only privacy properties; properties like security and correctness are not covered. The model considers a set of users (with unique identifiers) $\mathcal{U}$, whose size is at least polynomial in a security parameter. There is no formal difference between different types of player, like there is with tag and reader in most RFID models. The system $\mathcal{S}$ can be invoked with input batches $(u_1, \alpha_1), (u_2, \alpha_2), \ldots, (u_c, \alpha_c) \in (\mathcal{U}, A)^c$, consisting of pairs of user identifiers and 'parameters' and will output a batch $((e_1, \ldots e_c), \beta)$, with the outputs e_i from each system invocation and a general output β, applying to the batch as a whole. Users can also be corrupted, revealing their internal state to the adversary.

The authors investigate the properties of the function $f \in \mathcal{F}$, where $\mathcal{F} = \{f : \{1, 2, \ldots, n\} \to \mathcal{U}\}$ is the space of functions that map the serial number of each output element to the user it corresponds to. In the Strong Anonymity (SA) setting, no information should be revealed to the adversary about the function f, guaranteeing the highest level of privacy. Several weaker notions (which reveal *some* information on f) are defined and the relations among notions are examined.

In the RFID setting the batch properties are currently not considered, although this would be an interesting extension, since some localization protocols are based on batch invocations of a large set of RFID tags. For simplicity we restrict ourselves to the Bohli-Pashalidis model for online systems. For these systems, where all batches have size one (i.e. the system never waits for multiple inputs until it produces some output), the only two applicable distinct notions are Strong Anonymity (SA) and Pseudonymity (PS).

The adversarial model is based on indistinguishability. The adversary can cause different users to invoke the system using different parameters (e.g. messages) in both a left and right world with the $\texttt{Input}((u_0, \alpha_0), (u_1, \alpha_1))$ oracle. Based on a bit b, selected by the challenger, the system will be invoked with the user-data pair (u_b, α_b). That is, the adversary itself defines the functions $f_0, f_1 \in \mathcal{F}$, for respectively the left and right world. The adversary can also corrupt users. At the end of the game the adversary has to output a guess bit g. The adversary

wins the game if $g = b$. By imposing restrictions on f_0 and f_1, the authors investigate different levels of privacy.

Definition 7. A privacy protecting system $\mathcal{S}$ is said to unconditionally provide privacy notion X, if and only if the adversary $\mathcal{A}$ is restricted to invocations (u_0, α_0) and (u_1, α_1) such that f_0 and f_1 are X-indistinguishable for all invocations and for all such adversaries $\mathcal{A}$, it holds that $\mathbf{Adv}^X_{\mathcal{S},\mathcal{A}}(k) = 0$.

Similar definitions for computational ($\mathcal{A}$ is polytime in k and $\mathbf{Adv}^X_{\mathcal{S},\mathcal{A}}(k) \leq \epsilon(k)$) and statistical privacy are available.

Discussion

Due to its generality, and due to the fact that it is not meant to cover security properties, the Bohli-Pashalidis model needs non-trivial adaptations in order to apply to RFID setting. In its current form, the model does not support multi-pass protocols, where linking two messages from the same protocol run is not a privacy breach. Moreover there is no distinction between tags that need to be protected, and the reader for which privacy is not an issue. An interesting question is whether the strictly binary distinguishing game (only one bit of randomness in the challenge) provides enough flexibility compared to other models, like Vaudenay's, where there are multiple bits of randomness that are to be guessed.

4 Our Model

4.1 Adversarial Model & Privacy

We use the setup from Definition 1. We assume a central reader R and a set of tags $\mathcal{T} = \{T_1, T_2, \ldots, T_i\}$. $\mathcal{T}$ is initially empty, and tags are added dynamically by the adversary. The reader maintains a database of tuples (ID_i, K_i), one for every tag $T_i \in \mathcal{T}$. Moreover, every tag T_i stores an internal state S_i.

Let $\mathcal{A}$ denote the adversary, which can adaptively control the system $\mathcal{S}$. $\mathcal{A}$ interacts with $\mathcal{S}$ through a set of oracles. The experiment that the challenger sets up for $\mathcal{A}$ (after the security parameter k is fixed) proceeds as follows:

$\mathbf{Exp}^b_{\mathcal{S},\mathcal{A}}(k)$:

1. $b \in_R \{0, 1\}$

2. $SetupReader(1^k)$
3. $g \leftarrow \mathcal{A}^{\texttt{CreateTag},\texttt{Launch},\texttt{DrawTag},\texttt{Free},\texttt{SendTag},\texttt{SendReader},\texttt{Result},\texttt{Corrupt}}()$
4. Return $g == b$.

At the beginning of the experiment, the challenger picks a random bit b. The adversary $\mathcal{A}$ subsequently interacts with the challenger by means of the following oracles:

- `CreateTag(ID)` $\rightarrow T_i$: on input a tag identifier ID, this oracle calls $SetupTag(ID)$ and registers the new tag with the server. A reference T_i to the new tag is returned. Note that this does not reject duplicate IDs.
- `Launch()` $\rightarrow \pi, m$: this oracle launches a new protocol run, according to the protocol specification. It returns a session identifier π, generated by the reader, together with the first message m that the reader sends. Note that this implies that our model does not support tag-initiated protocols.
- `DrawTag(`T_i`,`T_j`)` $\rightarrow vtag$: on input a pair of tag references, this oracle generates a virtual tag reference, as a monotonic counter, $vtag$ and stores the triple $(vtag, T_i, T_j)$ in a table $\mathcal{D}$. Depending on the value of b, $vtag$ either refers to T_i or T_j. If one of the two tags T_i or T_j is already referenced in the table (i.e. is already passed to a `DrawTag` without being released with a `Free`), then this oracle returns $\perp$. Otherwise, it returns $vtag$.
- `Free(vtag)`$_b$: on input $vtag$, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$. If $b = 0$, it resets the tag T_i. Otherwise, it resets the tag T_j. Then it removes the entry $(vtag, T_i, T_j)$ from $\mathcal{D}$. When a tag is reset, its volatile memory is erased. The non-volatile memory, which contains the state S, is preserved.
- `SendTag(vtag,`m`)`$_b \rightarrow m'$: on input $vtag$, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$ and sends the message m to either T_i (if $b = 0$) or T_j (if $b = 1$). It returns the reply from the tag (m'). If the above triple is not found in $\mathcal{D}$, it returns $\perp$.
- `SendReader(`π`,` m`)` $\rightarrow m'$: on input π, m this oracle sends the message m to the reader in session π and returns the reply m' from the reader (if any) is returned by the oracle. [4]
- `Result(`π`)`: on input π, this oracle returns a bit indicating whether or not the reader accepted session π as a protocol run that resulted in successful authentication of a tag. If the session with identifier π is not finished yet, or there exists no session with identifier π, $\perp$ is returned.

[4] If no active session π exists, the reader is likely to return $\perp$.

- `Corrupt`(T_i): on input a tag reference T_i, this oracle returns the complete internal state of T_i. [5] Note that the adversary is not given control over T_i.

According to the above experiment description, the challenger presents to the adversary the system where either the 'left' tags T_i (if $b = 0$) or the 'right' tags T_j (if $b = 1$) are selected when returning a virtual tag reference in `DrawTag`. The function $f_0 \in \mathcal{F}$ (where $\mathcal{F} = \{f : \{1, 2, \ldots, n\} \rightarrow \mathcal{T}\}$, see Section 3.5) maps the `DrawTag` invocations (referenced by an index k) to the tag T_i, which was passed as first argument to `DrawTag`. Similarly, f_1 maps invocation serial numbers to the second argument to `DrawTag`. f_0 and f_1 therefore describe the 'left' and the 'right' world, respectively.

$\mathcal{A}$ queries the oracles a number of times and, subsequently, outputs a guess bit g. We say that $\mathcal{A}$ wins the privacy game if and only if $g = b$, i.e. if it correctly identifies which of the worlds was active. The advantage of the adversary is defined as

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}}(k) = \left| Pr\left[\mathbf{Exp}^0_{\mathcal{S},\mathcal{A}}(k) = 1\right] + Pr\left[\mathbf{Exp}^1_{\mathcal{S},\mathcal{A}}(k) = 1\right] - 1 \right| \qquad (1)$$

4.2 Security, Correctness, Privacy

Since our model focuses on privacy, the correctness and security property are not discusses further. Both the Vaudenay and Deng et al. security and correctness definition can be used combined with the new privacy definition, without compatibility issues (also see Section 3.1 and Section 3.3).

The adversary restrictions, as defined in Section 3.1, also apply to our privacy definition. Depending on the acceptable usage of the `Corrupt` oracle, an adversary in our model is either Strong, Destructive (`Corrupt` destroys a tag), Forward (after the first `Corrupt` only further corruptions are allowed), or Weak (no `Corrupt` oracle) adversaries. Depending on the allowed usage of the `Result` oracle, there exist Narrow (no `Result` oracle) and Wide adversaries. X is used to denote one of these privacy notions.

Definition 8 (Privacy)**.** An RFID system $\mathcal{S}$, is said to unconditionally provide privacy notion X, if and only if for all adversaries $\mathcal{A}$ of type X, it holds that $\mathbf{Adv}^X_{\mathcal{S},\mathcal{A}}(k) = 0$. Similarly, we speak of computational privacy if for all polynomial time adversaries, $\mathbf{Adv}^X_{\mathcal{S},\mathcal{A}}(k) \leq \epsilon(k)$

[5] Both the volatile and non-volatile state is returned. For multi-pass protocols it might be necessary to relax this to only the non-volatile state; to force the adversary to only corrupt tags T_i that are currently not drawn; or to use the concept of X^+ privacy, as discussed in Section 4.3.

We also define X^+ privacy notion variants, where X refers to the basic privacy notion and $+$ to the notion that arises when the corruption abilities of the adversary are further restricted (see [6]). Formally, an RFID system is said to be X^+ private if it is X private and if, for all adversaries, $f_0 \approx_{\hat{T}} f_1$. Here, $f_0 \approx_{\hat{T}} f_1$ means that $\forall i$ such that $f_0(i) \in \hat{T}$ or $f_1(i) \in \hat{T}$, it holds that $f_0(i) = f_1(i)$, where $\hat{T}$ denotes the set of corrupted tags. This implies that, whenever a tag is corrupted at some point during the privacy game, it always has to be drawn simultaniously in both the left and the right world using a `DrawTag`(T_i, T_i) query with identical arguments.

4.3 Motivation and Comparison

Our proposed model is based on the well-studied notion of (left-or-right) indistinguishability. This avoids the issues with less well-studied concepts such as blinders that the Vaudenay model suffers from (see Section 3.1). Moreover, since several cryptographic schemes have proven security properties based on indistinguishability games (e.g. IND-CPA, IND-CCA, IND-CCA2...), this is likely to simplify the proofs using our model when using these schemes as building blocks.

Note that the Juels-Weis model from Section 3.4 also uses a traditional indistinguishability setup. However, the model requires the adversary to distinguish one out of two selected tags in the final phase. The disadvantage of this approach is that it does not take into account other properties that might leak privacy (e.g. cardinality) and that it limits the use of tag corruption. The Vaudenay model did introduce some crucial tools like virtual tag references and the corruption types that are still required.

Modelling Details There are certain notable differences of our model when compared to the Bohli-Pashalidis model [6] and the other models discussed in Sect. 3:

- The introduction of `CreateTag`$(\cdot)$: since the set of tags is not predefined we allow the adversary to dynamically create new tags.
- `DrawTag`$(\cdot, \cdot)$ and `Free`$(\cdot)$ are used to introduce the concept of virtual tags. This concept is needed since otherwise `SendTag`$(\cdot, \cdot)$ would have to accept two tag/message pairs (and select one of them based on the value of b). In this case it would be trivial to determine the bit b for multi-pass protocols, simply by using different tags for each pass of the protocol if $b = 0$ and the same tag if $b = 1$. The protocol would only succeed if

$b = 1$, thus allowing detection of b. Hence, it is crucial that the same tag is always used within a certain protocol run, which can be ensured by using virtual tag identifiers.

- `Free(·)` clears the volatile memory of tag, in order to avoid attacks that depend on leaving a tag hanging in a temporary state. Such an attack is described in [24].
- A separate communication oracle for tags and reader is used, since the reader is not considered as an entity whose privacy can be compromised.
- `Corrupt(·)`: corruption is done with respect to a tag, not a virtual tag. If `Corrupt(·)` would accept a vtag, then determining the bit b becomes trivial by performing the following attack:
  - $vtag_a \leftarrow$ `DrawTag(`T_1`,`T_2`)`
  - $C_a \leftarrow$ `Corrupt(`$vtag_a$`)`
  - `Free(`$vtag_a$`)`
  - $vtag_b \leftarrow$ `DrawTag(`T_1`,`T_3`)`
  - $C_b \leftarrow$ `Corrupt(`$vtag_b$`)`

  If $C_a = C_b$ then $b = 0$, otherwise $b = 1$.

  We believe that it is realistic to assume that one has the tag identifier T_i when corrupting a tag, since corruption implies having physical access to a tag.

  Note that stateful protocols (which update their state after a protocol run) do not satisfy our privacy definition. By issuing a `Corrupt(`T_i`)` query before and after a protocol run, one can always identify whether or not the tag has been active. For such protocols, one could use the significantly weaker X^+ privacy notions.
- In the current setup `Corrupt(`T_i`)` reveals the full internal state of the tag, i.e. both its volatile and non-volatile parts. This follows [1] where it is shown that, if corruptions reveal the volatile state, then the resulting privacy notions are stronger. Single-pass protocols (e.g. challenge-response) do not suffer from any issues, since the volatile memory is typically erased after sending the reply, and hence all computations are confined to the invocation of the `SendTag` oracle. Multi-pass protocols on the contrary, typically require storage of data in between `SendTag` invocations. Because corruption yields the entire internal state, one could make additional assumptions on the corruption abilities of the adversary by restricting corruption to the non-volatile state. An even stronger restriction would be to allow only corruption of tags that are not drawn in either the left or right world; or use the X^+ privacy notions.

5 Evaluating Existing Protocols

This section evaluates several protocols (or classes of protocols) using our privacy model. For security and correctness results we refer to the original papers.

Several protocol 'prototypes' based on symmetric cryptography are evaluated by Ng et al. in [23] with respect to Vaudenay's privacy model. Since none of these protocols attain wide-forward privacy, we expect them to behave the same in our model. For this reason, these protocols are not discussed further.

5.1 Vaudenay's Public Key Protocol

Figure 2 shows the public key protocol presented by Vaudenay. The reader sends out a random number a and the tag encrypts this challenge, combined with the shared secret K and tag ID under the public key K_P of the reader. The reader can decrypt the tag's reply and verify the shared secret K in its database. The protocol relies on the encryption being IND-CPA to achieve narrow-strong Vaudenay-privacy and IND-CCA2 to achieve security and forward privacy. However, this protocol is wide-strong private under our model, if the underlying encryption is IND-CCA2.

Theorem 1. *If the encryption used in the protocol from Figure 2 is IND-CPA, then the protocol is strong private for narrow adversaries (i.e. adversaries that do not use the* `Result` *query).*

Proof. Given an adversary $\mathcal{A}$ that wins the privacy game with non-negligible advantage, we show how to create an adversary $\mathcal{A}'$ that wins the IND-CPA game with non-negligible advantage.

The adversary $\mathcal{A}'$ runs the adversary $\mathcal{A}$ and answers all oracle queries from $\mathcal{A}$ by simply simulating the system $\mathcal{S}$, with the following exceptions:

- The public key K_P of the reader is the public key of the IND-CPA game.
- `SendTag`: retrieve the tag references T_i and T_j from the table using the virtual tag identity $vtag$. For these two tags, it generates the messages $m_0 = ID_i||K_i||a$ and $m_1 = ID_j||K_j||a$. The two messages m_0, m_1 are forwarded to the IND-CPA oracle, which returns the encryption under K_P of one of the messages.

At the end of the game $\mathcal{A}'$ outputs whatever guess $\mathcal{A}$ outputs. The privacy game is perfectly simulated for the inner adversary $\mathcal{A}$.

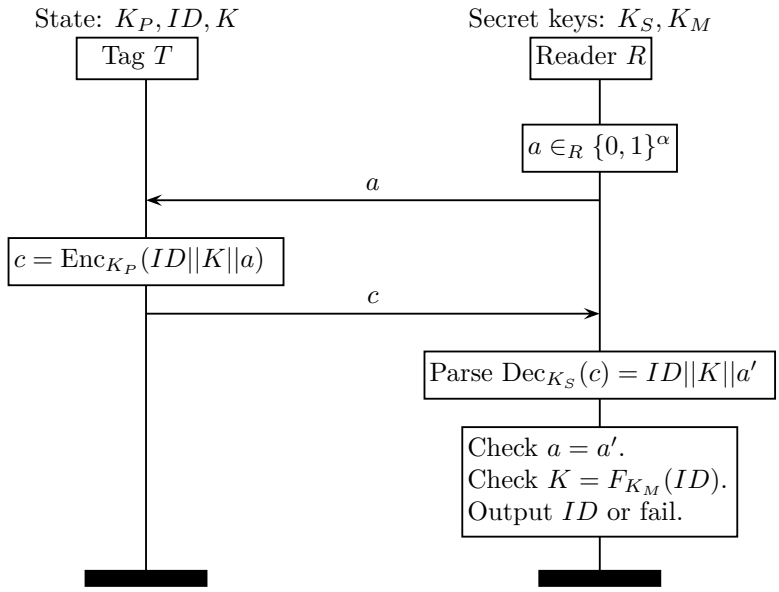


Figure 2: Public key RFID protocol from [31]

Assume that $\mathcal{A}$ breaks privacy, i.e. it can distinguish the left and right world, then $\mathcal{A}'$ wins the IND-CPA game. Since IND-CPA with only one call to the encryption oracle is equivalent to IND-CPA with multiple calls to the encryption oracle, this proves the (narrow) privacy of the protocol. □

The results from Lemma 8 in [31] still hold, provided the security and correctness definitions from Vaudenay are used. So, based on these results, the protocol above is also wide forward private.

Theorem 2. *If the encryption used in the protocol from Figure 2 is IND-CCA2, then the protocol is strong private for wide adversaries.*

Proof. The proof is similar to the proof for Theorem 1 above. When receiving a `Result` query, the adversary proceeds as follows. It first compares the ciphertext c to a list of outputs generated by the encryption oracle from the IND-CPA game (which are used in the `SendTag` oracle). If it matches one of these, `true` is returned. Otherwise, the result oracle forwards the ciphertext to the IND-CCA decryption oracle and receives the matching plaintext m. The plaintext is then parsed and verified, just as the reader would do. This game gives the same result as the IND-CPA game described in Theorem 1. □

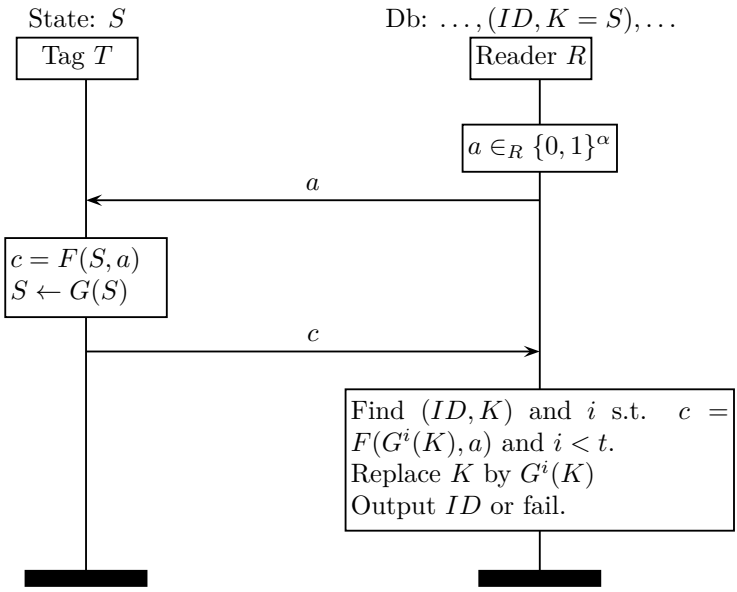


Figure 3: RO protocol from [31]

5.2 RO-based Protocol

Another (weaker) protocol from [31], shown in Figure 3, makes use of two random oracles F and G. The protocol uses an updating state S, which is shared by both tag and reader. The reader sends out a random number a and the tag computes a reply by applying F on the state S and a. The state is afterwards updated using G. Obviously, such a protocol cannot be (narrow) strong private, since the tag can trivially be traced after being corrupted.

Theorem 3. *The protocol shown in Figure 3 is narrow-destructive private.*

Proof. Assume that the challenge bit $b = 0$. We simulate the `SendTag` oracle by returning a random value c. There will never be a `SendTag` query to a corrupted tag, since tags are destroyed after corruption. This way we obtain a 'random' world that is indistinguishable from the 'left' world obtained when $b = 0$, provided the adversary makes no calls to F and G identical to the queries inside the `SendTag` oracle when $b = 0$. The probability of this happening is however negligible. By applying the same argument to the adversary execution when $b = 1$, we show that the adversary cannot distinguish between the two worlds. □

6 Conclusion

Several RFID privacy models were critically examined with respect to their assumptions, practical usability and other issues that arise when applying their privacy definition to concrete protocols. We have shown that, while some models are based on unrealistic assumptions, others are impractical to apply. We presented a new RFID privacy model, that, based on the classic notion of indistinguishability, combines the benefits of existing models while avoiding their identified drawbacks. By proving it for a concrete protocol, we show that the notion of (wide) strong privacy can be achieved under our model. Since the privacy model is based on an indistinguishability game, we can fall back on a wide range of existing proof techniques, making the model quite straightforward to use in practice.

Acknowledgements

The authors would like to thank Elena Andreeva, Junfeng Fan, Sebastian Faust, and Roel Peeters for the frequent meetings and discussions; and the anonymous reviewers for their comments and suggestions.

References

[1] F. Armknecht, A.-R. Sadeghi, A. Scafuro, I. Visconti, and C. Wachsmann. Impossibility Results for RFID Privacy Notions. *Transactions on Computational Science*, 11:39–63, 2010.

[2] Atmel Corporation. Innovative Silicon IDIC solutions, 2007. `http://www.atmel.com/dyn/resources/prod_documents/doc4602.pdf`.

[3] G. Avoine, E. Dysli, and P. Oechslin. Reducing Time Complexity in RFID Systems. In B. Preneel and S. E. Tavares, editors, *SAC*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306. Springer, 2005.

[4] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification Protocols Secure against Reset Attacks. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511. Springer, 2001.

[5] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In H. Krawczyk, editor,

CRYPTO, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

[6] J.-M. Bohli and A. Pashalidis. Relations Among Privacy Notions. In R. Dingledine and P. Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 362–380. Springer, 2009.

[7] J. Bringer, H. Chabanne, and T. Icart. Efficient zero-knowledge identification schemes which respect privacy. In W. Li, W. Susilo, U. K. Tupakula, R. Safavi-Naini, and V. Varadharajan, editors, *ASIACCS*, pages 195–205. ACM, 2009.

[8] M. Burmester, T. Le, and B. Medeiros. Provably secure ubiquitous systems: Universally composable RFID authentication protocols. In *Proceedings of the 2nd IEEE/CreateNet International Conference on Security and Privacy in Communication Networks (SECURECOMM.* IEEE Press, 2006.

[9] S. Canard, I. Coisel, J. Etrog, and M. Girault. Privacy-preserving rfid systems: Model and constructions. Cryptology ePrint Archive, Report 2010/405, 2010. `http://eprint.iacr.org/`.

[10] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[11] I. Damgård and M. Østergaard. RFID Security: Tradeoffs between Security and Efficiency. Cryptology ePrint Archive, Report 2006/234, 2006. `http://eprint.iacr.org/`.

[12] P. D'Arco, A. Scafuro, and I. Visconti. Revisiting DoS Attacks and Privacy in RFID-Enabled Networks. In S. Dolev, editor, *ALGOSENSORS*, volume 5804 of *Lecture Notes in Computer Science*, pages 76–87. Springer, 2009.

[13] P. D'Arco, A. Scafuro, and I. Visconti. Semi-Destructive Privacy in DoS-Enabled RFID systems. *RFIDSec*, 2009.

[14] R. H. Deng, Y. Li, M. Yung, and Y. Zhao. A New Framework for RFID Privacy. In D. Gritzalis, B. Preneel, and M. Theoharidou, editors, *ESORICS*, volume 6345 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2010.

[15] V. Goyal and A. Sahai. Resettably Secure Computation. In A. Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2009.

[16] J. Ha, S.-J. Moon, J. Zhou, and J. Ha. A New Formal Proof Model for RFID Location Privacy. In S. Jajodia and J. López, editors, *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 2008.

[17] M. Hutter, J.-M. Schmidt, and T. Plos. RFID and Its Vulnerability to Faults. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 363–379. Springer, 2008.

[18] I.C.A. Organization. Machine Readable Travel Documents, Doc 9303, Part 1 Machine Readable Passports, 5th edn, 2003.

[19] A. Juels and S. A. Weis. Defining Strong Privacy for RFID. In *PerCom Workshops*, pages 342–347. IEEE Computer Society, 2007.

[20] T. Kasper, D. Oswald, and C. Paar. New Methods for Cost-Effective Side-Channel Attacks on Cryptographic RFIDs. *RFIDSec*, 2009.

[21] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[22] C. Y. Ng, W. Susilo, Y. Mu, and R. Safavi-Naini. RFID Privacy Models Revisited. In S. Jajodia and J. López, editors, *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2008.

[23] C. Y. Ng, W. Susilo, Y. Mu, and R. Safavi-Naini. New Privacy Results on Synchronized RFID Authentication Protocols against Tag Tracing. In M. Backes and P. Ning, editors, *ESORICS*, volume 5789 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2009.

[24] R.-I. Paise and S. Vaudenay. Mutual Authentication in RFID: Security and Privacy. In *ASIACCS*, pages 292–299, Tokyo, Japan, 2008. ACM Press.

[25] T. Plos. Evaluation of the Detached Power Supply as Side-Channel Analysis Countermeasure for Passive UHF RFID Tags. In M. Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 444–458. Springer, 2009.

[26] A.-R. Sadeghi, I. Visconti, and C. Wachsmann. User Privacy in Transport Systems Based on RFID E-Tickets. In C. Bettini, S. Jajodia, P. Samarati, and X. S. Wang, editors, *PiLBA*, volume 397 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[27] A.-R. Sadeghi, I. Visconti, and C. Wachsmann. Anonymizer-Enabled Security and Privacy for RFID. In J. A. Garay, A. Miyaji, and A. Otsuka,

editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 134–153. Springer, 2009.

[28] A.-R. Sadeghi, I. Visconti, and C. Wachsmann. Efficient RFID security and privacy with anonymizers. *RFIDSec*, 2009.

[29] N. Semiconductors. MIFARE. `http://www.mifare.net/`.

[30] T. Van Le, M. Burmester, and B. de Medeiros. Universally composable and forward-secure RFID authentication and authenticated key exchange. In *ASIACCS*, ASIACCS, pages 242–252, New York, NY, USA, 2007. ACM.

[31] S. Vaudenay. On Privacy Models for RFID. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2007.

[32] S. Vaudenay. Invited talk at RFIDSec, 2010.

[33] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In D. Hutter, G. Müller, W. Stephan, and M. Ullmann, editors, *SPC*, volume 2802 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2003.

A Extending the Model

In a typical indistinguishability-based security/privacy definition, a challenger picks a random bit b and then offers a set of well-defined interfaces over which an adversary $\mathcal{A}$ can interact with the challenger. In 'left-or-right' security/privacy definitions, in particular, the interface specification requires that $\mathcal{A}$ provides *a pair* of identically formatted inputs to the challenger. The value of b can be interpreted as indicating in which of two possible configurations the challenger operates, namely the 'left' or the 'right' configuration, and $\mathcal{A}$'s job is to determine this configuration.

It is possible to generalise left-or-right indistinguishability such that, the challenger picks one out of 2^n possible configurations, giving us an n-indistinguishability game, with adversary $\mathcal{A}_n$. Suppose there is a system $\mathcal{S}$ that, if invoked with some parameter α (taken from a system-specific parameter space A), produces an output $\mathcal{S}(\alpha)$. The challenger chooses a positive number n, such that n is polynomial in k and generates an n-bit vector $\hat{b} = (\hat{b}_1, \ldots, \hat{b}_n)$ uniformly at random. Finally, it offers an interface over which $\mathcal{A}_n$ may query the challenger with triplets of the form $(i, \alpha_0, \alpha_1) \in \{1, \ldots, n\} \times A \times A$. On input such a triple, the challenger outputs $\mathcal{S}(\alpha_{\hat{b}_i})$.

At the end of the game, $\mathcal{A}_n$ outputs a guess $\hat{g}$ for $\hat{b}$, and we say that it wins the game if $\hat{g} = \hat{b}$. If there exists some $\mathcal{A}_n$ such that $\Pr(\mathcal{A}_n \text{wins}) > 1/2^n + \epsilon$, where ϵ is any function that is non-negligible in k, then we say that $\mathcal{A}_n$ has 'non-negligible advantage' and that $\mathcal{S}$ is not secure.

In general, it is unclear whether or not n-indistinguishability implies 1-indistinguishability. In principle, a system could be secure if the adversary has to identify a string from a space that is exponentially large in k, but may fail security if the adversary just needs to identify a single hidden bit.

Lemma 1 (1-indistinguishability implies n-indistinguishability)**.** *If a system $\mathcal{S}$ satisfies 1-indistinguishability then $\mathcal{S}$ also satisfies n-indistinguishability.*

Proof. We construct an 1-indistinguishability adversary $\mathcal{A}$ that uses an n-indistinguishability adversary $\mathcal{A}_n$ as a black box. $\mathcal{A}$ proceeds as follows. First, it uniformly at random chooses two n-bit vector κ and λ such that $\kappa \neq \lambda$. Then it offers the interface (i, α_0, α_1) to $\mathcal{A}_n$. For each (i, α_0, α_1) received from $\mathcal{A}_n$, $\mathcal{A}$ forwards the query $(\alpha_{\kappa_i}, \alpha_{\lambda_i})$ to the challenger, and returns the challenger's output. By forwarding the queries this way, $\mathcal{A}$ simulates $\hat{b} = \kappa$ if $b = 0$, and $\hat{b} = \lambda$ if $b = 1$ for $\mathcal{A}_n$. In the rest of the proof $\hat{b}$ will denote the κ if $b = 0$ and λ if $b = 1$, $\bar{\hat{b}}$ will denote the κ if $b = 1$ and λ if $b = 0$. Accordingly, and given $\mathcal{A}_n$'s guess $\hat{g}$, $\mathcal{A}$ outputs the guess $b = 0$ if $\hat{g} = \kappa$, $b = 1$ if $\hat{g} = \lambda$, or simply a uniformly at random selected bit otherwise.

Consider the $2^n \times 2^n$ matrix P with elements $p_{i,j} = \Pr(\mathcal{A}_n \text{ outputs } j \mid \hat{b} = i)$. That is, P contains the probabilities that $\mathcal{A}_n$ outputs any possible value $\hat{g}$, conditional on the value of $\hat{b}$; the element at row number i and column number j is the probability that $\mathcal{A}_n$ outputs $\hat{g} = j$ (encoded as a bit vector), given the challenge bit vector has the value $\hat{b} = i$ (encoded as a bit vector). Note that, for all $0 \leq i \leq 2^n$, $\sum_j p_{i,j} = 1$.

For any given choice of a pair (κ, λ), the probability that $\mathcal{A}_n$ wins (i.e. that it outputs $\hat{g} = \hat{b}$) is $1/2(p_{\kappa,\kappa} + p_{\lambda,\lambda})$. Similarly, the probability that it ouputs $\hat{g} = \bar{\hat{b}}$ is $1/2(p_{\kappa,\lambda} + p_{\lambda,\kappa})$. Averaging over all possible choices of (κ, λ) we obtain

$$\Pr(\mathcal{A}_n \text{ wins}) = \frac{1}{2^n(2^n - 1)} \sum_{\substack{\kappa,\lambda \in \{0,1\}^n \\ \kappa \neq \lambda}} \frac{1}{2}(p_{\kappa,\kappa} + p_{\lambda,\lambda}) = \frac{\mathcal{D}}{2^n} \tag{2}$$

$$\Pr(\text{err}) = \frac{1}{2^n(2^n - 1)} \sum_{\substack{\kappa,\lambda \in \{0,1\}^n \\ \kappa \neq \lambda}} \frac{1}{2}(p_{\kappa,\lambda} + p_{\lambda,\kappa}) = \frac{2^n - \mathcal{D}}{2^n(2^n - 1)}, \tag{3}$$

where $\mathcal{D} = \sum_{i=1}^{2^n} p_{i,i}$ is the trace of P. By construction of our $\mathcal{A}$, we have

$$\Pr(\mathcal{A} \text{ wins}) = \Pr(\mathcal{A}_n \text{ wins}) + 1/2(1 - \Pr(\mathcal{A}_n \text{ wins}) - \Pr(\text{err})) \tag{4}$$

and substituting Equations 2 and 3 into Equation 4, we obtain

$$\Pr(\mathcal{A} \text{ wins}) = \frac{1}{2} + \frac{2^n(\mathcal{D} - 1)}{2^{n+1}(2^n - 1)}. \tag{5}$$

By assumption we have that $\Pr(\mathcal{A}_n \text{ wins}) > 1/2^n + \epsilon$ for all functions ϵ that are negligible in k. Hence, $\Pr(\mathcal{A}_n \text{ wins}) = 1/2^n + \delta$ for some non-negligible positive $\delta \leq 1 - 1/2^n$. In terms of the elements in P, we have $\mathcal{D} = 1 + 2^n\delta$ and when substituting this into Equation 5 we obtain $\Pr(\mathcal{A} \text{ wins}) = \frac{1}{2} + \frac{2^n\delta}{2(2^n-1)} > 1/2 + \delta/2$. Hence, $\mathcal{A}$'s advantage is non-negligible. □

Unlike standard hybrid arguments, the advantage δ is at most divided by 2, when going from an n-bit distinguisher to a 1-bit distinguisher.

B Mutual Authentication

Since our model is not based anymore on the blinder construction of Paise-Vaudenay [24], none of the impossibility results of [1] apply. It is straightforward to modify the proof from Section 5.1 to the mutual authentication protocol based on IND-CCA encryption from Section 6.3 in [24].

Publication

Wide Strong Private RFID Identification based on Zero-Knowledge

Publication Data

Roel Peeters and Jens Hermans. Wide Strong Private RFID Identification based on Zero-Knowledge, 2012. In submission.

Contributions

- Security and privacy proofs for the protocols.
- Co-development of protocols with Roel Peeters

Wide Strong Private RFID Identification based on Zero-Knowledge

Roel Peeters and Jens Hermans*

Department of Electrical Engineering – COSIC
KU Leuven and IBBT
Kasteelpark Arenberg 10/2446, 3001 Heverlee, BELGIUM
firstname.lastname@esat.kuleuven.be

Abstract. We present the first wide-strong RFID identification protocol that is based on zero-knowledge. Until now this notion has only been achieved by schemes based on IND-CCA2 encryption. Rigorous proofs in the standard model are provided for the security and privacy properties of our protocol. Furthermore our protocol is the most efficient solution presented in the literature. Using only Elliptic Curve Cryptography (ECC), the required circuit area can be minimized such that our protocol even fits on small RFID tags. Concerning computation on the tag, we only require two scalar-EC point multiplications.

Keywords. RFID, Private Identification, Zero-Knowledge, Elliptic Curve Cryptography.

1 Introduction

RFID tags are deployed in various consumer applications such as physical access tokens, car keys, contactless payment systems and electronic passports. For these applications, it is crucial that the underlying protocols protect not only security but also the (location) privacy of the end user. Yet, all communication with RFID tags can easily be eavesdropped or modified, tags respond to any query and RFID tags can be corrupted, which renders these vulnerable to attacks. On top of this, an adversary can typically learn the outcome of the identification protocol. Successful identifications result in an unlocked door, unlocked car or processed payment; while failure has no outcome.

*Jens Hermans is a research assistant, sponsored by the Fund for Scientific Research - Flanders (FWO).

Privacy of RFID identification protocols is evaluated in terms of achieved privacy notions. The notion of strong privacy provides the strongest privacy guarantees: no adversary actively interacting with the tags and the reader is able to infer any information on a tag's identity from tag communication, even when given all secrets stored on the tag. The notion of wide-strong privacy corresponds to strong privacy against adversaries that also learn the outcome of the protocol.

Our goal is to design and evaluate an RFID identification protocol with the strongest possible privacy guarantees, i.e. wide-strong. This privacy notion cannot be achieved when considering only symmetric identification protocols [27], where some cryptographic secret is shared between tag and reader. Additionally, Damgård and Pedersen [11] showed that privacy for RFID symmetric identification protocols, comes at the cost of a non-scalable lookup procedure at the reader. Examples of symmetric RFID identification protocols can be found in [5, 13, 18]. The main reason behind using symmetric identification protocols is the perception that public-key cryptography requires either too much time, power or circuit area to implement on low-cost devices. However, Lee *et al.* [21] and Hein *et al.* [16] showed that public key cryptography, in particular Elliptic Curve Cryptography (ECC), can be realized on RFID tags. Previously, wide strong privacy has only been achieved by schemes relying on an IND-CCA2 encryption scheme (or variants of such schemes) [10, 27]. Our scheme only needs an ECC architecture without additional components typically required for IND-CCA2 encryption (e.g. hash function), resulting in a smaller hardware footprint, which is a great improvement.

Outline

Section 2 first introduces the required definitions. An overview of relevant previously proposed private RFID protocols is given in Sect. 3. In Sect. 4, we propose our protocol and analyze its security and privacy properties. We also propose an optimized version of our protocol. Section 5 takes into account some implementation considerations and compares the different protocols.

2 Definitions

We consider a system comprised of multiple tags and only one reader, where a tag and the reader carry out an identification protocol. Each tag stores a state and the reader keeps a database of all valid tags, to which tags can be dynamically added by the adversary. More general the reader could be a central

back-end server that is connected to multiple readers, however tags can only identify to one server. Adversaries are allowed to communicate with all tags and the reader. For privacy, only the content of the exchanged messages is taken into account, not the physical characteristics of the radio links as studied by Danev *et al.* [12] which should be dealt with at the hardware level.

In this section, we will first give a short overview of the selected privacy model and the different privacy notions. Then the properties of a private identification protocol are discussed. Finally we give an overview of the necessary number-theoretic assumptions.

2.1 Privacy Model

The privacy model of Vaudenay [27] was one of the first and most complete privacy models that featured the notion of strong privacy. This model is based on simulatability; for the strongest privacy notions a separate blinder between the adversary and the oracles is required. Vaudenay shows that wide strong privacy cannot be achieved in this model by using a specific feature of the blinder. Armknecht *et al.* [2] later pointed out some issues of this model with regard to the blinder. Canard *et al.* [10] also proposed a simulation based model that resolves these issues by introducing a trivial adversary. However, their model is less general, as the focus is on finding non-trivial links between messages communicated by the same tag. The Juels-Weis model [19] is a well-known privacy model based on indistinguishability. This model lacks generality since it does not allow the adversary to corrupt challenge tags. Hermans *et al.* [17] provide a general privacy model for RFID based on indistinguishability; it is more robust and easier to apply. For these reasons we selected this model, a brief description of the model is given below. For more details on the different RFID privacy models and a comparison between these, the reader is referred to [17].

Privacy Model of Hermans et al. [17]

The intuition behind the RFID privacy model of Hermans *et al.* is that privacy is guaranteed if an adversary cannot distinguish with which one of two RFID tags (of its choosing), he is interacting through a set of oracles.

Privacy is defined as a distinguishability game (or experiment $\mathbf{Exp}$) between a challenger and the adversary. This game is defined as follows. First the challenger picks a random challenge bit b and then sets up the system $\mathcal{S}$ with

a security parameter k. Next, the adversary $\mathcal{A}$ can use a subset (depending on the privacy notion) of the following oracles to interact with the system:

- `CreateTag`$(ID) \rightarrow T_i$: on input a tag identifier ID, this oracle creates a tag with the given identifier and corresponding secrets, and registers the new tag with the reader. A reference T_i to the new tag is returned. Note that this does not reject duplicate IDs.
- `Launch`$() \rightarrow \pi$: this oracle launches a new protocol run on the reader R_j, according to the protocol specification. It returns a session identifier π, generated by the reader.
- `DrawTag`$(T_i, T_j) \rightarrow vtag$: on input a pair of tag references, this oracle generates a virtual tag reference, as a monotonic counter, $vtag$ and stores the triple $(vtag, T_i, T_j)$ in a table $\mathcal{D}$. Depending on the value of b, $vtag$ either refers to T_i or T_j. If T_i is already references as the left-side tag in $\mathcal{D}$ or T_j as the right-side tag, then this oracle also returns $\perp$ and adds no entry to $\mathcal{D}$. Otherwise, it returns $vtag$.
- `Free`$(vtag)_b$: on input $vtag$, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$. If $b = 0$, it resets the tag T_i. Otherwise, it resets the tag T_j. Then it removes the entry $(vtag, T_i, T_j)$ from $\mathcal{D}$. When a tag is reset, its volatile memory is erased. The non-volatile memory, which contains the state S, is preserved.
- `SendTag`$(vtag, m)_b \rightarrow m'$: on input $vtag$, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$ and sends the message m to either T_i (if $b = 0$) or T_j (if $b = 1$). It returns the reply from the tag (m'). If the above triple is not found in $\mathcal{D}$, it returns $\perp$.
- `SendReader`$(\pi, m) \rightarrow m'$: on input π, m this oracle sends the message m to the reader in session π and returns the reply m' from the reader (if any) is returned by the oracle.
- `Result`(π): on input π, this oracle returns a bit indicating whether or not the reader accepted session π as a protocol run that resulted in successful authentication of a tag. If the session with identifier π is not finished yet, or there exists no session with identifier π, $\perp$ is returned.
- `Corrupt`(T_i): on input a tag reference T_i, this oracle returns the complete internal state of T_i. Note that the adversary is not given control over T_i.

By using the `DrawTag` oracle the adversary can arbitrarily select which tags to interact with. Based upon the challenge bit b the system that the challenger presents to the adversary will behave as either the 'left' tags T_i or the 'right'

tags T_j. After $\mathcal{A}$ called the oracles, it outputs a guess bit g. The outcome of the game will be $g == b$, *i.e.*, 0 for an incorrect and 1 for a correct guess. The adversary wins the privacy game if it can distinguish correctly the 'left' from the 'right' world being executed. The advantage of the adversary $\mathbf{Adv}_{\mathcal{S},\mathcal{A}}(k)$ is defined as:

$$\left|Pr\left[\mathbf{Exp}^0_{\mathcal{S},\mathcal{A}}(k)=1\right]+Pr\left[\mathbf{Exp}^1_{\mathcal{S},\mathcal{A}}(k)=1\right]-1\right| .$$

2.2 Privacy Notions

Strong attackers are allowed to use all the oracles available. *Forward* attackers are only allowed to do other corruptions after the first corruption, protocol interactions are no longer allowed. *Weak* attackers cannot corrupt tags.

Independently of these classes, there is the notion of *wide* and *narrow* attackers. A *wide* attacker is allowed to get the result from the reader, *i.e.*, whether the identification was successful or not; while a *narrow* attacker does not.

The privacy notions are related as follows:

$$\begin{array}{ccccc} \text{Wide-Strong} & \Rightarrow & \text{Wide-Forward} & \Rightarrow & \text{Wide-Weak} \\ \Downarrow & & \Downarrow & & \Downarrow \\ \text{Narrow-Strong} & \Rightarrow & \text{Narrow-Forward} & \Rightarrow & \text{Narrow-Weak} \end{array}$$

$A \Rightarrow B$ means that if the protocol is A-private, it implies that the protocol is B-private. It should be obvious that a protocol that is wide-strong private will also belong to all other privacy classes, that only allow weaker adversaries.

We also define X^* privacy notion variants, where X refers to the basic privacy notion and $*$ to the notion that arises when the corruption abilities of the adversary are further restricted with respect to the `Corrupt` oracle. The restricted `Corrupt` oracle will only return the non-volatile state of the tag. This restriction allows to exclude trivial privacy attacks on multi-pass protocols, that require the tag to store some information in volatile memory during the protocol run.

2.3 Private Identification Protocol

A private identification protocol has the following three properties: correctness, soundness and privacy. Correctness and soundness are necessary to establish the security of the identification protocol. Privacy will ensure that all parties cannot infer any information on the tag's identity from the protocol messages, except the reader for which the tag is identifying to.

A function $f : \mathbb{N} \to \mathbb{R}$ is called 'polynomial' in the security parameter $k \in \mathbb{Z}$ if $f(k) = O(k^n)$, with $n \in \mathbb{N}$. It is called 'negligible' if, for every $c \in \mathbb{N}$ there exists an integer k_c such that $f(k) \leq k^{-c}$ for all $k > k_c$.

Correctness ensures that a legitimate tag is always accepted by a reader.

Definition 1. Correctness. A scheme is correct if the identification of a legitimate tag only fails with negligible probability.

Soundness is the property that a fake tag is not accepted by the reader. We only consider adversaries that cannot interact with the tag they try to impersonate during the identification protocol (i.e., we do not consider relay or concurrent attacks). Concurrent attacks are impossible in the RFID setting, since tags can only participate in one session at the time. To avoid relay attacks, distance bounding protocols can be deployed. Rasmussen *et al.* [23] proposed an implementation of such a protocol with analog components that is suitable for RFID tags. The following definition differs from most models as we do not require matching conversations, but impersonation resistance as in [7] is sufficient.

Definition 2. Soundness. A scheme is resistant against impersonation attacks if no polynomially bounded strong adversary succeeds, with non-negligible probability, in being identified by a verifier as the tag it impersonates. Adversaries may interact with the tag they want to impersonate prior to, and with all other tags prior to and during the protocol run. All tags, except the impersonated tag, can be corrupted by the adversary.

In a more general setting, a tag could be allowed to identify privately to multiple readers (not connected to the same central back-end server). In such a setting one RFID tag can be used to gain access to multiple independent locations, *e.g.*, office and home. However, even for a subset of corrupted readers, the adversary should not gain an advantage in authenticating as a valid tag to an uncorrupted reader. In this setting there is a clear advantage for protocols that provide extended soundness, since the tag can use the same private/public key pair to identify to each reader.

Definition 3. Extended Soundness. Identical to Def. 2, but the adversary is also given the secret key of the reader and the full reader database.

Definition 4. Privacy. A privacy protecting protocol, modeled by the system $\mathcal{S}$, is said to computationally provide privacy notion X, if and only if for all polynomially bounded adversaries $\mathcal{A}$, it holds that $\mathbf{Adv}^X_{\mathcal{S},\mathcal{A}}(k) \leq \epsilon$, for negligible ϵ.

The model does not adequately capture insider-attacks that aim to destroy tag privacy, which were recently introduced by Van Deursen *et al.* [26]. For these attacks the adversary controls one valid (insider) tag and has access to the `Result` oracle. As such, they were able to link other valid tags in certain protocols. Since this attack requires access to the `Result` oracle, only the wide privacy notions are affected. Obviously, privacy in presence of a wide-strong privacy attacker implies privacy against an inside attacker, since the former is allowed knowledge of tags' internal states (hence has access to insider tags). For wide-weak and wide-forward private protocols, privacy in presence of insider-attackers needs to be evaluated separately.

2.4 Number-theoretical Assumptions

Our proposed protocol is based on Elliptic Curve Cryptography, hence we make use of additive notation. Points on the curve are represented by capital letters while scalars are represented by lowercase letters.

The $\texttt{xcoord}(\cdot)$ function is the ECDSA conversion function [8], which comes almost for free when using elliptic curves. Assuming an elliptic curve $\mathbb{E}$ with prime order ℓ over $\mathbb{F}_p$, then for a point $Q = \{q_x, q_y\}$ with $q_x, q_y \in [0 \dots p-1]$, $\texttt{xcoord}(Q)$ maps Q to $q_x \mod \ell$. We define $\texttt{xcoord}(O) = 0$, where O is the point at infinity.

Discrete Logarithm

Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ and let A be a given arbitrary element of $\mathbb{G}_\ell$. The discrete logarithm (DL) problem is to find the unique integer $a \in \mathbb{Z}_\ell$ such that $A = aP$. The DL assumption states that it is computationally hard to solve the DL problem.

One More Discrete Logarithm

The one more discrete logarithm (OMDL) problem was introduced by Bellare *et al.* [3]. Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ. Let $\mathcal{O}_1()$ be an oracle that returns random elements $A_i = a_i P$ of $\mathbb{G}_\ell$. Let $\mathcal{O}_2(\cdot)$ be an oracle that returns the discrete logarithm of a given input base P. The OMDL problem is to return the discrete logarithms for each of the elements obtained from the m queries to $\mathcal{O}_1()$, while making strictly less than m queries to $\mathcal{O}_2(\cdot)$ (with $m > 0$).

x-Logarithm

Brown and Gjøsteen [9] introduced the x-Logarithm (XL) problem: given an elliptic curve point, determine whether its discrete logarithm is congruent to the x-coordinate of an elliptic curve point. The XL assumption states that it is computationally hard to solve the XL problem. Brown and Gjøsteen also provided some evidence that the XL problem is almost as hard as the DDH problem.

Diffie-Hellman

Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ and let aP, bP be two given arbitrary elements of $\mathbb{G}_\ell$, with $a, b \in \mathbb{Z}_\ell$. The computational Diffie-Hellman (CDH) problem is, given P, aP and bP, to find abP. The 4-tuple $\langle P, aP, bP, abP \rangle$ is called a Diffie-Hellman tuple. Given a fourth element $cP \in \mathbb{G}_\ell$, the decisional Diffie-Hellman (DDH) problem is to determine if $\langle P, aP, bP, cP \rangle$ is a valid Diffie-Hellman tuple or not. The DDH assumption states that it is computationally hard to solve the DDH problem.

Oracle Diffie-Hellman

Abdalla *et al.* [1] introduced the ODH assumption:

Definition 5. Oracle Diffie-Hellman (ODH) Given $A = aP, B = bP$, a function H and an adversary $\mathcal{A}$, consider the following experiments:

Experiment $\mathbf{Exp}^{odh}_{H,\mathcal{A}}$:

- $\mathcal{O}(Z) := H(bZ)$ for $Z \neq \pm A$
- $g = \mathcal{A}^{\mathcal{O}(\cdot)}(A, B, H(C))$
- Return g

The value C is equal to abP for the $\mathbf{Exp}^{odh-real}_{H,\mathcal{A}}$ experiment, chosen at random in $\mathbb{G}_\ell$ for the $\mathbf{Exp}^{odh-random}_{H,\mathcal{A}}$ experiment.

We define the advantage of $\mathcal{A}$ violating the ODH assumption as:

$$\left| \Pr\left[\mathbf{Exp}^{odh-real}_{H,\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{odh-random}_{H,\mathcal{A}} = 1\right] \right|.$$

The ODH assumption consists of the plain DDH assumption combined with an additional assumption on the function $H(\cdot)$. The idea is to give the adversary access to an oracle $\mathcal{O}$ that computes bZ, without giving the adversary the ability to compute bA, which can then be compared with C. To achieve this one restricts the oracle to $Z \neq \pm A$, and moreover, only $H(bZ)$ instead of bZ is released, to prevent the adversary from exploiting the self-reducibility of the DL problem.[1] The crucial property that is required for $H(\cdot)$ is one-wayness. In the following part we use a one-way function based on the DL assumption. We define the function $H(Z) := \texttt{xcoord}(Z)P$.

Theorem 1. *The function $H(\cdot)$ is a one-way function under the DL assumption.*

3 Previously Proposed Protocols

In this section, we give an overview of previously proposed protocols that are based on public key cryptography. Each of these protocols is correct, sound and achieves narrow-strong privacy.

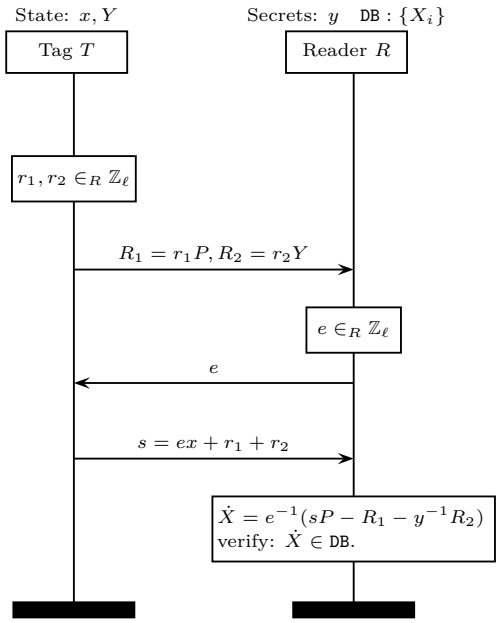
3.1 Zero Knowledge Based Protocols

The zero knowledge based protocols are proofs of knowledge for a specific verifier (reader) with public key $Y = yP$. The prover (tag) proves knowledge of the private key $x \in \mathbb{Z}_\ell$, which is the discrete logarithm of the corresponding public key $X = xP$, for P a publicly agreed-on generator of $\mathbb{G}_\ell$. The public key X of the tag will serve as its identity and has been registered with the reader.

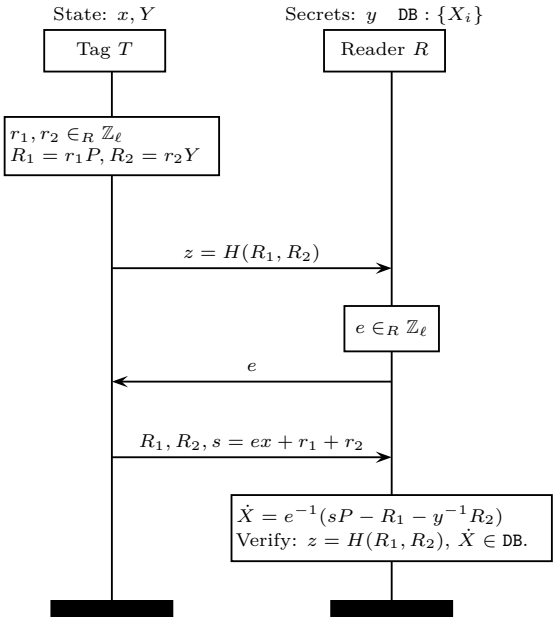
Randomized Schnorr was proposed by Bringer *et al.* [6] (see Fig. 1(a)). It achieves extended soundness and narrow-strong* privacy. This protocol requires only two scalar-EC point multiplications at the tag side.

Randomized Hashed GPS was later proposed by Bringer *et al.* [7] (see Fig. 1(b)). The protocol has extended soundness and narrow-strong* privacy. The authors also claim wide-PI-forward* privacy, i.e., wide-forward* privacy even when the list of registered tags' identities is known. This approach requires two scalar-EC point multiplications and the evaluation of a hash function, for which additional hardware will be needed.

[1] The adversary can set $Z = rA$ for a known r and compute $r^{-1}(bZ) = bA$.



(a) Randomized Schnorr [6]



(b) Randomized Hashed GPS [7]

Figure 1: Zero knowledge based protocols.

Privacy-wise, both protocols suffer from the adversary having complete freedom over the exam e it sends to the tag and the fact that the final message from the tag s contains a term that is linearly dependent on this exam and the secret of the tag x. For this reason these protocols cannot be wide-strong private.[2] Furthermore, there exist a linear relation between the commitments (R_1, R_2) and the answer s. This, together with the above, makes that Randomized Schnorr cannot be wide-weak private.[3] Both protocols are also vulnerable to insider-attacks.[4]

3.2 Public Key Encryption Based Protocols

For these protocols, the reader has a public/private key pair (PK, pk). The identities ID of tags that registered are stored in the reader's database. The tag and reader share a symmetric key K.

Vaudenay's Public Key Protocol [27] (see Fig. 2(a)) requires the tag to compute the public key encryption of one message. This cryptosystem needs to be secure against adaptive chosen ciphertext attacks (IND-CCA2) to have a secure identification scheme that achieves narrow-strong and wide-forward privacy. When evaluating this protocol in the privacy model of Hermans *et al.* [17], this protocol achieves wide-strong privacy. One of the most efficient IND-CCA2 cryptosystems in the standard model is DHIES [1]. This cryptosystem requires two scalar-EC point multiplications, one evaluation of a hash function, one evaluation of a MAC algorithm and the invocation of symmetric encryption scheme per encryption.

Hash ElGamal Based Protocol was proposed by Canard *et al.* [10] (see Fig. 2(b)). This protocol is secure, narrow-strong private and future untraceable. It is unclear how future untraceability (as defined by Canard *et al.* [10]) and wide-strong privacy relate to each other, however, these seem to be closely

[2] An attacker in the middle sends $e - 1$ to the virtual tag and responds to the reader with $s + x$. For a correct guess of the tag's identity with known internal state x, the `result` oracle returns 1.

[3] For an observed protocol run π_0, an adversary can test, using the `result` oracle, that the current virtual tag is the tag of π_0. The adversary mounts a Man-In-The-Middle attack, sends to the reader $(R_1 + R_{1,0}, R_2 + R_{2,0})$, challenges the tag with $e - e_0$ and returns to the reader $s + s_0$.

[4] Similar to the above. The attacker sends the exam e_0 to the virtual tag in protocol run π_1. When subtracting the answers $s_0 - s_1$, the tag specific part should cancel out. The attacker starts a protocol run π_2 between its insider tag (with private key x') and the reader. The attacker sets $R_1 = R_{1,0} - R_{1,1}$, $R_2 = R_{2,0} - R_{2,1}$ and replies with $s' = s_0 - s_1 + e_2 x'$.

related. It makes use of a cryptosystem that is secure against chosen plaintext attacks (IND-CPA), Hash ElGamal; and a MAC algorithm. This scheme is more efficient than Vaudenay's public key scheme since the underlying encryption does not need to be IND-CCA2. Note that the combination of a MAC and IND-CPA encryption used in this specific protocol in fact provides IND-CCA2 encryption for the type of plaintext messages used [20]. The tag is required to compute two scalar-EC point multiplications, one evaluation of a hash function and one evaluation of a MAC algorithm.

Neither protocol achieves extended soundness. The tag and the reader need to store some shared (secret) data. These shared data consist of an identifier ID and a shared secret key K. Both protocols achieve wide-strong privacy and soundness can also be proven under the more strict definition of matching conversations. Wide-strong privacy rules out insider attacks on privacy.

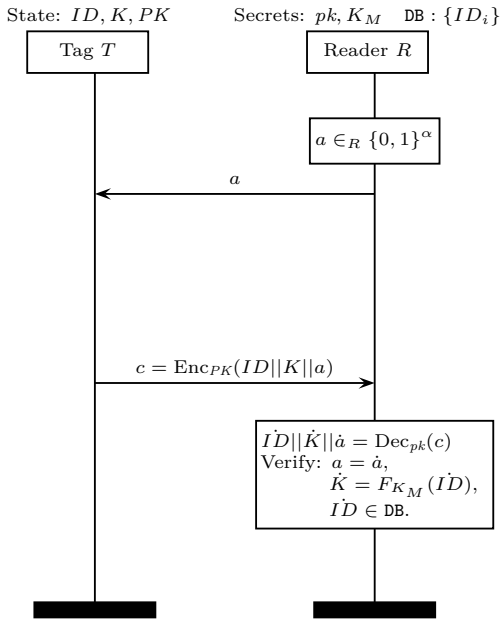
4 A New Protocol

Our proposed protocol is a modified version of the Schnorr identification protocol [24]. The original protocol is proven secure by Bellare and Palacio [4] under the OMDL assumption. This protocol consists of three passes: commit, exam and response. A consequence of having a three pass protocol is that only the X* privacy notions can be reached.

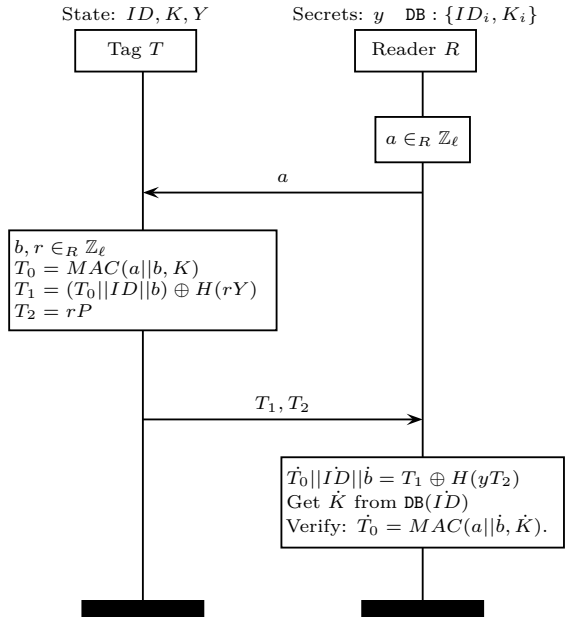
Our starting point is a variant of the Schnorr identification protocol, where the exam of the reader is applied to the tag's randomness instead of its secret. This variant is equivalent to the original protocol, except for the case that $e = 0$. In the original Schnorr identification protocol this results in the adversary learning the tag's randomness while in the variant the adversary will learn the tag's secret. This situation can easily be avoided by having the tag check that e is not equal to 0.

Privacy is ensured by introducing a blinding factor d that can only be computed by the tag and the reader. The blinding factor is also applied to the exam e and added to the response. This blinding factor only depends on input of the tag and the public key of the reader, which is known to the tag. As such an adversary cannot influence the value of this blinding factor. In contrast to previously proposed zero-knowledge based protocols (see Sect. 3.1), no factor is applied to the secret of the tag.

An overview of the proposed protocol is given in Fig. 3. The tag generates two random numbers r_1 and r_2, where the former is needed for extended soundness and the latter is used to ensure privacy. The tag commits to its randomness



(a) Vaudenay's Public Key RFID Protocol [27]



(b) Hash ElGamal Based Protocol [10]

Figure 2: Public key encryption based protocols.

by sending R_1, R_2 to the reader. The reader verifies that neither $R_1 = O$ nor $R_2 = O$, the point at infinity. The tag's response is $s = x + er_1 + d$, with d the blinding factor as computed by the tag. Note that the tag must check that $d, e \neq 0$.[5] The reader verifies by checking that a tag with public key $\dot{X} = (s - \dot{d})P - eR_1$, with $\dot{d}$ the blinding factor as computed by the reader, has been registered. The reader keeps a list of all incomplete sessions. If a session timeout occurs or the tag fails to identify for a given challenge, the session is also considered to be completed.

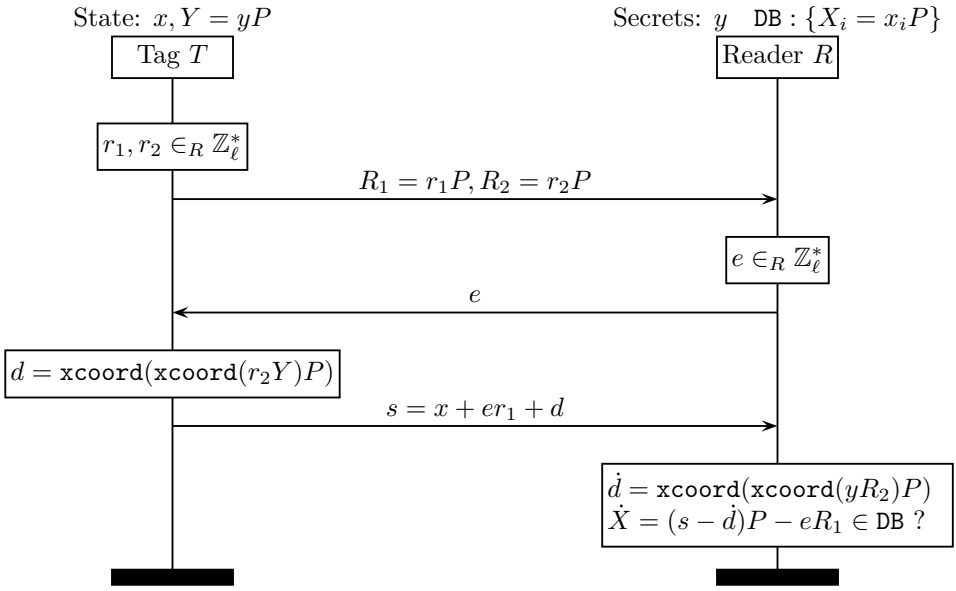


Figure 3: Private RFID identification protocol.

The blinding factor contains $r_2Y = yR_2$. Given the CDH assumption, this value can only be computed when given either r_2 or y. To prevent an adversary of exploiting the self-reduciblity of the DL problem, this value is encapsulated in a one-way function. An obvious one-way function is a cryptographic hash function. However, to implement a cryptographic hash function on an RFID tag, additional logic is required. Current hash functions [25] require at least 50% of the circuit area of the most compact ECC implementation. For this reason we propose the following one-way function, that is build using only EC

[5] By appropriate selection of the elliptic curve (e.g. a curve without points $(0, y)$), checking $d \neq 0$ is not necessary if $R_2 \neq O$.

operations $H(r_2Y) = \texttt{xcoord}(r_2Y)P$. The value d is set to the x-coordinate of the EC point.

4.1 Analysis

The first two theorems deal with the security properties of the proposed protocol. The last theorem deals with the privacy properties of the proposed protocol.

Theorem 2. *The proposed protocol is correct according to Def. 1.*

Proof. Since $d = \texttt{xcoord}(\texttt{xcoord}(r_2Y)P) = \texttt{xcoord}(\texttt{xcoord}(yR_2)P) = \dot{d}$, it follows that $\dot{X} = (s - \dot{d})P - eR_1 = (x + er_1 + d - d)P - er_1P = X$. □

Theorem 3. *The proposed protocol has extended soundness according to Def. 2 under the OMDL assumption.*

Proof. Assume an adversary $\mathcal{A}$ that can break the extended soundness with non-negligible probability, i.e. that can perform a fresh, valid authentication with the verifier. Without loss of generality we will assume the target tag is known at the start of the game. [6] We construct an adversary $\mathcal{B}$ that wins the OMDL game as follows:

- Set $X = \mathcal{O}_1()$. X will be used as the public key of the target tag.
- $\mathcal{B}$ executes $\mathcal{A}$. During the first phase of $\mathcal{A}$, $\mathcal{B}$ simulates the `SendTag` oracles for the target tag as follows (all other oracles are simulated as per protocol specification):
  - On the first $\texttt{SendTag}(vtag)$ query of the i'th protocol run: return $R_{2,i} = r_{2,i}P$ with $r_{2,i} \in_R \mathbb{Z}_\ell$ and $R_{1,i} = \mathcal{O}_1()$.
  - On the second $\texttt{SendTag}(vtag, e_i)$ query of the i'th protocol run: set $d_i = \texttt{xcoord}(\texttt{xcoord}(r_{2,i}Y)P)$ and return $s_i = \mathcal{O}_2(X + d_iP + e_iR_{1,i})$
- During the second phase of $\mathcal{A}$, $\mathcal{B}$ proceeds as follows:

[6] Otherwise, the proof can be adapted by choosing the public keys of the tags as $X_i = \mathcal{O}_1()$. All tag queries are simulated as for the target tag, until the tag is corrupted. When corrupting a tag, call $\mathcal{O}_2(X_i)$ for that tag and use the result as private key for simulating all following queries to that tag. At the end of the game, use the $\mathcal{O}_2(\cdot)$ oracle to extract all remaining discrete logarithms, except for the target tag.

- On the first call of $\mathcal{A}$ to $\texttt{Result}(\pi)$, compute $d = \texttt{xcoord}(\texttt{xcoord}(yR_2)P)$ and store (s, d). Next, rewind $\mathcal{A}$ until right before the call to $\texttt{SendReader}(\pi, R_1, R_2)$. On the next call to $\texttt{SendReader}(\pi, R_1, R_2)$, return a new random e'.
- On the next call of $\mathcal{A}$ to $\texttt{Result}(\pi)$: compute $r_1 = {}^{(s-s')}/_{(e-e')}$ and $x = s - d - er_1$ return $(x, e_1^{-1}(s_1 - x - d_1), \ldots, e_k^{-1}(s_k - x - d_k))$.

The simulation by $\mathcal{B}$ is perfect during both phases. At the end of the game $\mathcal{B}$ will successfully win the OMDL with non-negligible probability, unless $s = s'$, which happens with negligible probability since both e and e' are randomly chosen after $R_2 \neq O$ is fixed. □

Theorem 4. *The proposed protocol is wide-strong* private according to Def. 4 under the ODH and the XL assumption.*

Proof. Assume an adversary $\mathcal{A}$ that wins the privacy game with non-negligible advantage. Using a standard hybrid argument [15, 29], we construct an adversary that breaks the ODH-assumption. We set $Y = B$. $\mathcal{B}_i$ plays the privacy game with $\mathcal{A}$. $\mathcal{B}_i$ selects a random bit $\tilde{b}$, which will indicate which world is simulated to $\mathcal{A}$. All oracles are simulated in the regular way, with the exception of the `SendTag` and `Result` oracle for the target tag:

- $\texttt{SendTag}(vtag)$:
  - $j \neq i$: Generate $r_1, r_2 \in_R \mathbb{Z}_\ell$. Take $R_1 = r_1P, R_2 = r_2P$. Return R_1 and R_2.
  - $j = i$: Generate $r_1 \in_R \mathbb{Z}_\ell$.Take $R_1 = r_1P, R_2 = A$. Return R_1 and R_2.
- $\texttt{SendTag}(vtag, e)$, j'th query: retrieve the tuple $(vtag, T_0, T_1)$ from the table $\mathcal{D}$. Take the key x for tag $T_{\tilde{b}}$.
  - $j < i$: Generate $r \in_R \mathbb{Z}_\ell$. Take $d = \texttt{xcoord}(H(rP))$. Return $s = x + er_1 + d$.
  - $j = i$: Take $d = \texttt{xcoord}(H(C))$. Return $s = x + er_1 + d$.
  - $j > i$: Take $d = \texttt{xcoord}(H(r_2Y))$. Return $s = x + er_1 + d$.
- $\texttt{Result}(\pi)$: If the received R_2 in session π matches A from the ODH problem take $\dot{d} = \texttt{xcoord}(H(C))$. If not, check if R_2 matches any of the R_2's generated during the first $i - 1$ `SendTag` queries. If so, use the r generated in that query and compute $\dot{d} = \texttt{xcoord}(H(rP))$. Otherwise, take $\dot{d} = \texttt{xcoord}(\mathcal{O}(R_2))$. Finally, compute $\dot{X} = (s - \dot{d})P - eR_1$. Check $\dot{X}$ with the database, return true if $\dot{X}$ is found, false otherwise.

At the end of the game $\mathcal{A}$ outputs its guess g for the privacy game. $\mathcal{B}_i$ outputs $(\tilde{b} == g)$.

The above simulation to $\mathcal{A}$ is perfect, since validation is done in the same way as the protocol specification. If $R_2 = A$, the oracle $\mathcal{O}(\cdot)$ cannot be used. However, in this case we know the corresponding value of d by directly using $H(C)$, which gives the same result.

We use $\mathcal{A}^i$ (with $i \in [1 \dots k]$) to denote the case that $\mathcal{A}$ runs with the first i `SendTag` queries random instances, and the other queries real instances. This is the case when $\mathcal{B}_{i+1}$ runs with a real ODH instance, or $\mathcal{B}_i$ with a random ODH instance.

By the hybrid argument we get that

$$\| \Pr\left[\mathcal{A}^0 \text{ wins}\right] - \Pr\left[\mathcal{A}^k \text{ wins}\right] \| \leq \sum \mathbf{Adv}_{B_i} .$$

Note that $\mathcal{A}^i$ wins if $\tilde{b} == g$.

In the case of $\mathcal{A}^0$, it is clear $\Pr\left[\mathcal{A}^0 \text{ wins}\right] = \Pr\left[\mathcal{A} \text{ wins}\right]$ since all oracles are simulated exactly as in the protocol definition.

In the case of $\mathcal{A}^k$, all `SendTag` queries are simulated with $r \in_R \mathbb{Z}_\ell$ and $d = \texttt{xcoord}(\texttt{xcoord}(rP)P)$. Under the XL assumption it follows that d is indistinguishable from a random value from the x-coordinate distribution and that d is independent of R_1, R_2 and e.

Since $s = x + er_1 + d$ and $R_1 = r_1 P$, it follows under the XL assumption that $(x + er_1 + d, e, R_1 = r_1 P)$, with d a random value from the x-coordinate distribution, is indistinguishable from $(\tilde{r}, e, R_1 = r_1 P)$, with $\tilde{r}$ a uniformly random value. Hence it follows that s is indistinguishable from a uniformly random value independent of x, as long as $e, d \neq 0$.

So $\mathcal{A}^k$ has probability $1/2$ of winning the privacy game, since it obtains no information at all on x from a tag.

$$\begin{aligned} \| \Pr\left[\mathcal{A}^0 \text{ wins}\right] - \Pr\left[\mathcal{A}^k \text{ wins}\right] \| &= \| \Pr\left[\mathcal{A} \text{ wins}\right] - \frac{1}{2} \| \\ &= \frac{1}{2} \mathbf{Adv}_{\mathcal{A}}^{privacy} \\ &\leq \sum \mathbf{Adv}_{B_i} \end{aligned}$$

It follows that at least one of the $\mathcal{B}_i$ has non-negligible probability to win the ODH game. □

4.2 Efficiency Optimisation

Only one random value r is generated by the tag ($r_1 = r_2$). As such, the tag has to compute one less scalar-EC point multiplication and has to transmit one less element. The blinding factor is changed to $d = \texttt{xcoord}(rY)$. This reduces the computational effort required from the tag with another scalar-EC point multiplication. The function to compute the blinding factor is no longer one-way for rY, however, the response s is. An overview of the protocol is given in Fig. 4.

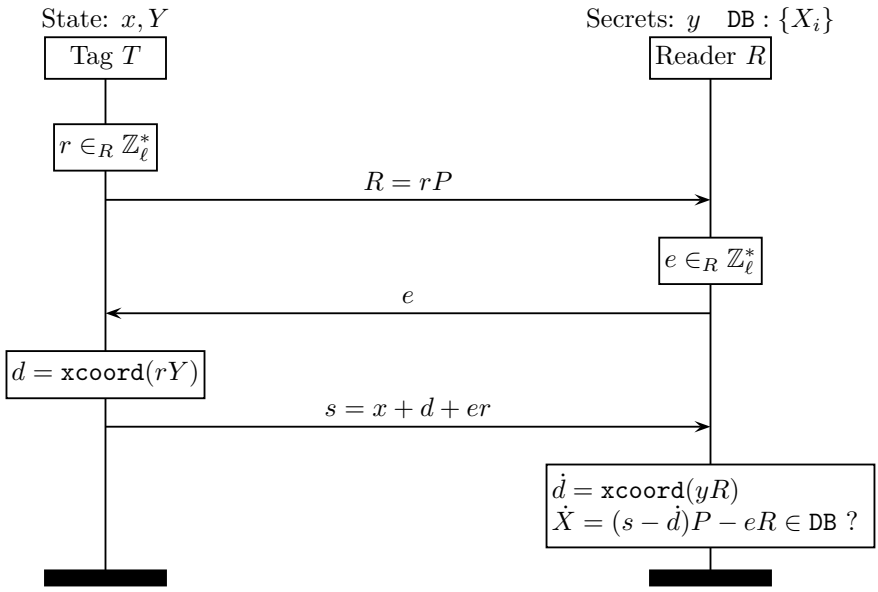


Figure 4: Optimised private RFID identification protocol.

Theorem 5. *The optimised protocol has extended soundness according to Def. 3 under the OMDL assumption.*

Proof. The proof is the same as the proof for the basic version of the protocol, except that $d = \texttt{xcoord}(rY)$ and $R = R_1 = R_2$. □

For privacy an extended ODH variant is required. The original ODH variant from Def. 5 gives direct access to an oracle for computing the blinding factor

d. This is no longer possible since $d = \texttt{xcoord}(C)$ does not involve a one-way function and would allow recovery of C.

Theorem 6. *The optimised protocol is wide-strong* private according to Def. 4 under the extended ODH assumption.*

The privacy of the optimised protocol can be shown under an extended ODH assumption where the adversary, in addition to $A = aP, B = bP, \texttt{xcoord}(C)P$ and the oracle $\mathcal{O}(Z)$, is also given an oracle $\mathcal{O}'(z) := \texttt{xcoord}(C) + za$ that can be called once with a $z \neq 0$.

A similar privacy proof as in Section 4.1 can be used, with different oracle calls in the `SendTag` and `Result` simulation. In this case $s = x + \mathcal{O}'(e)$ is used in the j'th `SendTag` for generating a reply if $j = i$. If $j < i$, a random r' is used to compute $d = \texttt{xcoord}(r'P)$.

If R matches one of the first $i - 1$ `SendTag` queries, then the random r' is used to compute $\dot{d} = \texttt{xcoord}(r'P)$. Otherwise, if $R \neq A$, then `Result` is simulated by using $\dot{d} = \mathcal{O}(R)$. If $R = A$, `Result` is simulated by directly computing $\dot{X} = sP - e \cdot \texttt{xcoord}(C)P - eR$.

5 Implementation Considerations

Our protocol requires the evaluation of scalar-EC point multiplications and the generation of a random number. For 80 bit security, we need an elliptic curve over a field that is approximately 160 bits in size. The protocol can be implemented on the architecture proposed by Lee *et al.* [22]. Their ECC coprocessor can be built with less than 15 kGEs (Gate Equivalent), consumes $\pm 13,8 \mu W$ of power and takes around 85 ms for one scalar-EC point multiplication. More recently, Wenger and Hutter [28] proposed an ECC coprocessor that only requires 9 kGEs, consumes $\pm 32,3 \mu W$ of power and takes around 286 ms for one scalar-EC point multiplication. Aside from the ECC coprocessor, circuit area is required for the ROM (Read Only Memory), RAM (Random Access Memory) and RNG (Random Number Generator).

5.1 Coupons

Several papers [7, 10] proposed to optimise their private RFID authentication protocols by means of precomputation. These precomputed values are stored in the form of coupons. When using coupons, the time needed by the tag to do the necessary computations drops. The most striking example is the

Randomized Hashed GPS: the tag does not need to compute complex scalar-EC point multiplications and evaluate a hash function anymore, instead only some simple scalar arithmetic is performed. Of course, the use of coupons comes with a price, i.e. storage also requires circuit area. As introduced by Girault *et al.* [14], the size of coupons can be minimized by not including the randomness in the coupons, but instead implementing a pseudo-random function with a seed on the tag to generate these random numbers when the coupons are used. But even so, only a limited number of coupons can be stored on the tag.[7] The question on how to securely get these coupons on the tag remains. These coupons can be generated by the tag itself, whenever energy is available. In this case, at the expense of having a slightly bigger design, private authentication protocols might be executed faster. Another option is that the coupons are generated by a third party and pushed on the tag. In this case, one can sometimes save on circuit area. For instance, the tag might only need to compute EC point additions or even only need scalar arithmetic. This approach has two disadvantages: first of all an attacker can quite easily mount a denial of service attack, since tags respond to any query; second, transferring these coupons securely is not straightforward. Lastly, it can be argued that strong privacy is not achievable when using coupons or a pseudo-random function instead of a true random number generator. Through the `Corrupt` oracle, the adversary learns the complete internal state of the tag, which also comprises coupons and/or the seed of the pseudo-random function. For these reasons we do not consider coupons.

5.2 Comparison

Now we will compare our protocol and its variants to previously proposed protocols, described in Sect. 3. A general overview of the protocols is given in Table 1.

Both the Randomized Schnorr and our proposed protocol benefit from a compact hardware design, only an ECC coprocessor is needed. The other protocols require additional hardware to evaluate a cryptographic hash function, which makes the design substantially larger. Recall that current hash functions [25] require at least 50% of the circuit area of the most compact ECC implementation.

The scalar-EC point multiplication is more complex than the evaluation of a hash/MAC. For a fair comparison between the performance of protocols that

[7] Abstracting away from the necessary control logic, one needs about one floating gate for each bit of storage. This means that we can only store 6-7 elements for a circuit area equivalent to 1kGE.

require the evaluation of a hash/MAC and protocols that do not, we assume the same total available circuit size. This means that our protocol can be implemented using a larger but faster ECC processor.

When also considering the more general setting, where a single tag can identify the end-user privately to multiple readers, the tags not only need to store an extra public key for every reader but also corresponding shared data, if any. In this setting there is a clear advantage for protocols that provide extended soundness, since the tag can use the same private/public key pair to identify to each reader.

6 Conclusions

This paper proposes a new wide-strong private RFID identification protocol. Unlike previous proposals, that are based on IND-CCA2 encryption, our protocol is based on zero-knowledge. Security and privacy of our protocol and all its optimised variant are proven in the standard model. Our protocol is the most efficient in its kind and can be implemented on RFID tags, using only Elliptic Curve Cryptography. This allows for a compact hardware design and requires minimal computational effort from the tag, namely two scalar-EC point multiplications. As an additional benefit, our protocols do not require any shared secrets between readers and tags.

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In D. Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[2] F. Armknecht, A.-R. Sadeghi, A. Scafuro, I. Visconti, and C. Wachsmann. Impossibility Results for RFID Privacy Notions. In M. Gavrilova, C. Tan, and E. Moreno, editors, *Transactions on Computational Science XI*, volume 6480 of *Lecture Notes in Computer Science*, pages 39–63. Springer, 2010.

[3] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology*, 16:185–215, 2003.

Table 1: Overview different proposed protocols.

Protocol	Strongest Privacy	Insider Private	Extended Soundness	Operations
Randomized Schnorr [6]	narrow-strong*	no	yes	2 EC mult
Randomized Hashed GPS [7]	narrow-strong* wide-forward*	no	yes	2 EC mult 1 hash
Vaudenay [27] + DHIES [1]	wide-strong	yes	no	2 EC mult 1 hash 1 MAC 1 symm enc
Hash ElGamal [10]	wide-strong	yes	no	2 EC mult 1 hash 1 MAC
Proposed Protocol (Sect. 4)	wide-strong*	yes	yes	4 EC mult
- optimised version (Sect. 4.2)	wide-strong*	yes	yes	2 EC mult

[4] M. Bellare and A. Palacio. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002.

[5] O. Billet, J. Etrog, and H. Gilbert. Lightweight Privacy Preserving Authentication for RFID Using a Stream Cipher. In S. Hong and T. Iwata, editors, *International Workshop — FSE*, volume 6147 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2010.

[6] J. Bringer, H. Chabanne, and T. Icart. Cryptanalysis of EC-RAC, a RFID Identification Protocol. In M. K. Franklin, L. C. K. Hui, and D. S. Wong, editors, *CANS*, volume 5339 of *Lecture Notes in Computer Science*, pages 149–161. Springer, 2008.

[7] J. Bringer, H. Chabanne, and T. Icart. Efficient Zero-Knowledge Identification Schemes which respect Privacy. In W. Li, W. Susilo, U. K. Tupakula, R. Safavi-Naini, and V. Varadharajan, editors, *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS, pages 195–205. ACM, 2009.

[8] D. R. Brown. Generic Groups, Collision Resistance, and ECDSA. *Designs, Codes and Cryptography*, 35(1):119–152, 2005.

[9] D. R. L. Brown and K. Gjøsteen. A Security Analysis of the NIST SP 800-90 Elliptic Curve Random Number Generator. In A. Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2007.

[10] S. Canard, I. Coisel, J. Etrog, and M. Girault. Privacy-Preserving RFID Systems: Model and Constructions. Cryptology ePrint Archive, Report 2010/405, 2010. `http://eprint.iacr.org/`.

[11] I. Damgård and M. Ø. Pedersen. RFID Security: Tradeoffs between Security and Efficiency. In T. Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2008.

[12] B. Danev, T. S. Heydt-Benjamin, and S. Čapkun. Physical-layer Identification of RFID Devices. In *USENIX*, pages 125–136. USENIX, 2009.

[13] H. Gilbert, M. J. Robshaw, and Y. Seurin. HB#: Increasing the Security and Efficiency of HB+. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2008.

[14] M. Girault, G. Poupard, and J. Stern. On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. *J. Cryptology*, 19:463–487, 2006.

[15] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, 2001.

[16] D. Hein, J. Wolkerstorfer, and N. Felber. *ECC Is Ready for RFID — A Proof in Silicon*, pages 401–413. Springer, Berlin, 2009.

[17] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A New RFID Privacy Model. In V. Atluri and C. Diaz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 568–587. Springer, 2011.

[18] A. Juels and S. A. Weis. Authenticating Pervasive Devices with Human Protocols. In V. Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

[19] A. Juels and S. A. Weis. Defining Strong Privacy for RFID. *ACM Trans. Inf. Syst. Secur.*, 13:7:1–7:23, November 2009.

[20] H. Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.

[21] Y. K. Lee, L. Batina, K. Sakiyama, and I. Verbauwhede. Elliptic Curve Based Security Processor for RFID. *IEEE Transactions on Computers*, 57(11):1514–1527, 2008.

[22] Y. K. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-Cost Untraceable Authentication Protocols for RFID. In C. Nita-Rotaru and F. Stajano, editors, *WISEC*, pages 55–64, Hoboken,NJ,USA, 2010. ACM.

[23] K. B. Rasmussen and S. Čapkun. Realization of RF Distance Bounding. In *USENIX*, pages 389–402. USENIX, 2010.

[24] C.-P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[25] SHA-3 Zoo. Overview of all Candidates for the Current SHA-3 Hash Competition Organized by NIST. `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`.

[26] T. van Deursen and S. Radomirović. Insider Attacks and Privacy of RFID Protocols. In S. Petkova-Nikova, A. Pashalidis, and G. Pernul, editors, *EUROPKI*, volume 7163 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2011.

[27] S. Vaudenay. On Privacy Models for RFID. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2007.

[28] E. Wenger and M. Hutter. A Hardware Processor Supporting Elliptic Curve Cryptography for Less Than 9 kGEs. In E. Prouff, editor, *CARDIS*, volume 7079 of *Lecture Notes in Computer Science*. Springer, 2011. in press.

[29] A. C.-C. Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *FOCS*, pages 80–91. IEEE Computer Society, 1982.

Publication

Private Yoking Proofs: Attacks, Models and New Provable Constructions

Publication Data

Jens Hermans and Roel Peeters. Private Yoking Proofs: Attacks, Models and new Provable Constructions. In Ingrid Verbauwhede, editor, *RFIDSec*, Lecture Notes in Computer Science. Springer, 2012. To appear.

Contributions

- Principal author together with Roel Peeters

Private Yoking Proofs: Attacks, Models and new Provable Constructions*

Jens Hermans and Roel Peeters

Department of Electrical Engineering - COSIC
KU Leuven and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. We present two attacks on the security of the private grouping proof by Batina *et al.* [1]. We introduce the first formal models for yoking proofs. One model incorporates the aspect *time*, ensuring that the grouping proofs were generated at a specific time. A more general variant only provides a proof that tags were together at some time. Based on these models we propose two new protocols to generate sound yoking proofs that can trivially be extended to multiple parties and that attain narrow-strong privacy.

1 Introduction

Juels [9] introduced the concept of yoking[1] proofs, also referred to as grouping proofs. These proofs allow the reader to claim afterwards (i.e. off-line) to a trusted party, that two RFID tags were scanned roughly at the same time and communicated to each other. Tags do not contain clocks and cannot communicate to each other directly, they communicate via the potentially untrusted reader. Note that these proofs give no guarantee that the RFID tags were physically close to each other, although close timing makes it harder for an adversary to obtain a yoking proof from two tags that are far apart. At the same time tag privacy should be considered: apart from the trusted party that is able to verify the yoking proof, no information should be gained on the tags' identities. Several papers have proposed constructions to generate yoking proofs, also generalising the setting to groupings of more than two tags.

*Joking with Yoking: Two Protocols in Front of a Circus :-)

[1]From the verb *to yoke*, meaning *to join together*.

Most proposed proof systems [3, 5, 9, 12–14] are based on symmetric cryptographic primitives. Lee *et al.* [11] and Hein *et al.* [7] showed that it is also possible to deploy public key cryptography on RFID tags, more specifically Elliptic Curve Cryptography. Towards tag privacy, symmetric cryptographic solutions are not scalable and only provide some basic privacy protection. Vaudenay [16] showed that public key cryptography is necessary to provide strong privacy guarantees for the tags such that no identifiable information leaks from the messages sent by the tags. Thus far, only Batina *et al.* [1] proposed two yoking proof systems that are based on public key cryptography. We will show two separate attacks on the security of their proposed protocols to generate yoking proofs.

One of the crucial aspects for grouping proofs is timing. Since a grouping proof is verified off-line by definition only a trusted party can assure the time the grouping proof took place. It's insufficient to simply submit the finished proof to the trusted party after finishing, since this does not prevent delaying the submission of the proof. The trusted party should actively participate in the protocol to avoid replaying and delaying. We present two security models: one that ensures timed grouping proofs, with trusted third party and one for non-timed grouping proofs without trusted third party. In the later case the proof only guarantees that the tags participated in a grouping proof without specifying any time or order.

Outline Section 2 presents two attacks on the Batina et al. In Sect. 3 we introduce the privacy and security model used throughout this paper. In Sect. 4 and Sect. 5 our new yoking proofs are proposed and their security and privacy is proven.

2 Attacks

Batina *et al.* [1] proposed two protocols to generate grouping proofs, one with colluding tag prevention and a basic one. Figure 1 describes the one with colluding tag prevention. The basic protocol, without colluding tag prevention, can be obtained by setting $r_s = 1$ in the protocol from Fig. 1. The proposed protocols build upon an authentication protocol, EC-RAC [10] for which the security is claimed to be related to the security of the Schnorr identification protocol [15].

We will now show how an adversary can break the security of these protocols, i.e. the adversary can generate a valid grouping proof that T_a and T_b were scanned together.

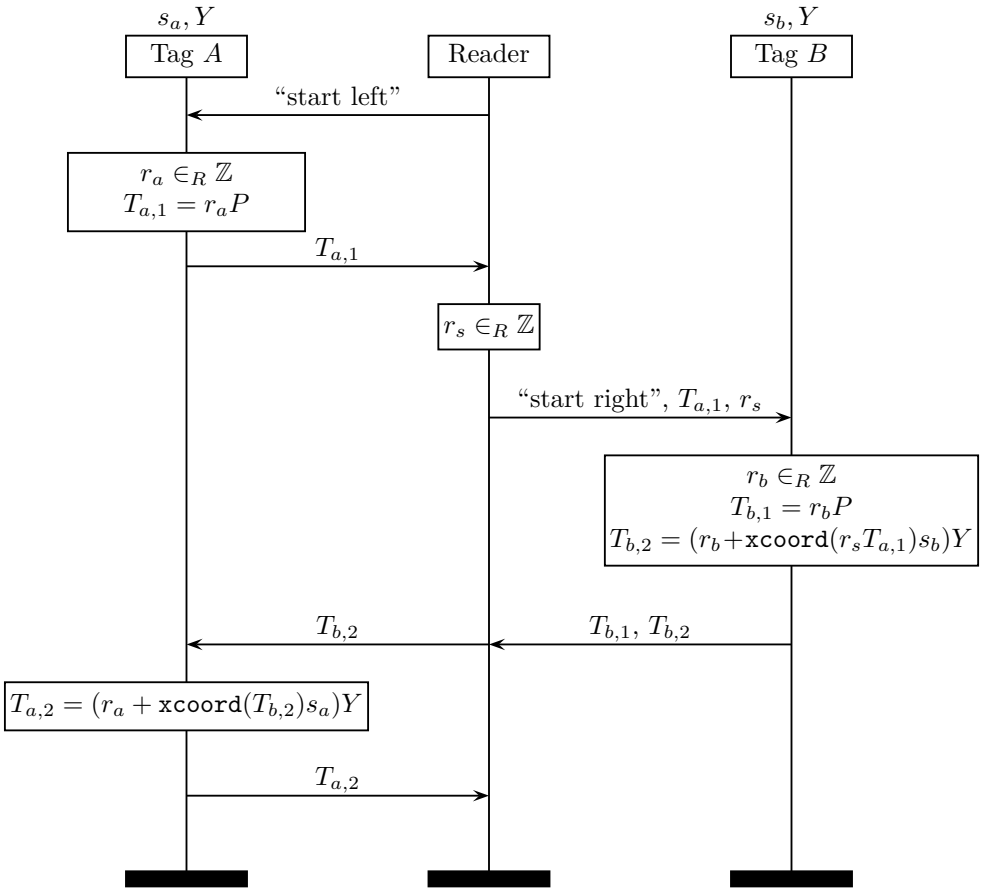


Figure 1: Two-party grouping-proof protocol with colluding tag prevention, proposed by Batina *et al.* [1].

2.1 First Attack

For authentication protocols, the temporal order of the messages is crucial. Authentication protocols consist of three stages: commit, exam and response. If the value of the exam is known before the prover needs to provide the commitment to its randomness (which is used later on for the response, the prover can construct a crooked proof). This can easily be shown for tag T_b. We are only interested in the value of the exam $\alpha = \texttt{xcoord}(r_s T_{a1})$, where $\texttt{xcoord}(P)$ returns the x-coordinate of the point $P = (x_P, y_P)$. Given this

tag's public key $S_b = s_bP$, the adversary can construct a valid response $T_{b,1} = rP - \alpha S_b$, $T_{b,2} = rY$ for $r \in_R \mathbb{Z}_l$. One can argue whether or not public keys of tags are known to the adversary, since the claimed privacy of the protocol implies that the adversary cannot learn the public key of an RFID tag from the exchanged messages, but we can definitely conclude that the value r_s, chosen by a genuine reader, does not provide any protection against colluding tags.

This weakness can be mitigated by requiring that the tag T_b first has to send the commitment $T_{b,1}$ before being presented with the exam. However, the resulting proof, presented by the (potentially untrusted) reader to the verifier, contains no verifiable information on the temporal ordering of the messages, still allowing this attack.

2.2 Second Attack

For the second attack, no knowledge of public keys of the tags is required. In the first phase the adversary needs to collect a tuple (α, T_1, T_2) for which the following relations hold: $T_1 = rP$ and $T_2 = (r + \alpha s)Y$, for s the secret key of the target tag and r an unknown random number. To collect this tuple one can eavesdrop on the protocol with honest tags: $(\texttt{xcoord}(T_{b,2}), T_{a,1}, T_{a,2})$ and $(\texttt{xcoord}(r_s T_{a1}), T_{b,1}, T_{b,2})$. Since there is no reader authentication (and the reader can be untrusted), one can also query the tags actively to collect this attack tuple.

In the second stage one can trick a genuine reader to accept $T'_{b,1}, T'_{b,2}$ as coming from the target tag, for which the attacker only has a tuple (α, T_1, T_2). This means that one can generate arbitrary yoking proofs with respect to tag T_b. Let $\beta = \texttt{xcoord}(r'_s T'_{a1})$, then $T'_{b,1}$ and $T'_{b,2}$ are computed as follows:

$$T'_{b,1} = \gamma T_1 + \delta P \qquad T'_{b,2} = \gamma T_2 + \delta Y \quad \text{for} \quad \delta \in_R \mathbb{Z}_l \quad \text{and} \quad \gamma = \frac{\beta}{\alpha}\,.$$

Again, this attack is independent on the value of r_s.

3 Privacy and Security Model

In this paper we will use the privacy model from Hermans et al. [8]. We will also use the oracles defined in this privacy model for the security games.

3.1 Privacy Model of Hermans et al.

The intuition behind the RFID privacy model is that privacy is guaranteed if an adversary cannot distinguish with which one of two RFID tags (of its choosing), he is interacting through a set of oracles. A brief overview of these oracles is given in App. A.

Privacy is defined as a distinguishability game between a challenger and the adversary. This game is defined as follows. First the challenger picks a random challenge bit b and then sets up the system $\mathcal{S}$ with a security parameter k. Next, the adversary $\mathcal{A}$ can use a subset (depending on the privacy notion) of the following oracles to interact with the system: $\texttt{CreateTag}(ID)$, $\texttt{DrawTag}(T_i, T_j)$, $\texttt{Free}(vtag)_b$, $\texttt{SendTag}(vtag, m)_b$, $\texttt{SendReader}(\pi, m)$, $\texttt{Result}(\pi)$ and $\texttt{Corrupt}(T_i)$.

By using the `DrawTag` oracle the adversary can arbitrarily select which tags to interact with. Based upon the challenge bit b the system that the challenger presents to the adversary will behave as either the *left* tags T_i or the *right* tags T_j. After $\mathcal{A}$ called the oracles, it outputs a guess bit g.

In this paper the $\texttt{Result}(\pi)$ is not used, since the grouping proofs are validated off-line at a later stage. For the full privacy definition we refer the reader to [8].

For the protocols that require a trusted third party (TTP), we define the $\texttt{SendTTP}(m) \rightarrow m'$ oracle, to send a message m to the TTP and receive the reply m'.

Privacy Notions

All adversaries presented in this paper are *narrow strong* adversaries, which are allowed to use all the oracles available except the `Result` oracle.

We also define X^* privacy notion variants, where X refers to the basic privacy notion and $*$ to the notion that arises when the corruption abilities of the adversary are further restricted with respect to the `Corrupt` oracle. The restricted `Corrupt` oracle will only return the non-volatile state of the tag. This restriction allows to exclude trivial privacy attacks on multi-pass protocols, that require the tag to store some information in volatile memory during the protocol run.

3.2 Grouping Proof

A grouping proof protocol has the following two properties: correctness, soundness. Correctness and soundness are necessary to establish the security of the protocol.

A function $f : \mathbb{N} \to \mathbb{R}$ is called *polynomial* in the security parameter $k \in \mathbb{Z}$ if $f(k) = O(k^n)$, with $n \in \mathbb{N}$. It is called *negligible* if, for every $c \in \mathbb{N}$ there exists an integer k_c such that $f(k) \leq k^{-c}$ for all $k > k_c$.

Definition 1. Correctness. A scheme is correct if a legitimate grouping proof is rejected with negligible probability and all tags involved are identified correctly with overwhelming probability.

We make a distinction between timed grouping proofs and non-timed grouping proofs. For a timed grouping proof, the time at which the proof was generated is recorded and can be verified afterwards. For a protocol to achieve timed grouping proof soundness a trusted third party is required to provide timestamps.

Definition 2 (Timed Grouping Proof Soundness)**.** In the first phase of the soundness game the adversary may interact with all tags. After the first phase ends the challenger notes the current time t_1. In the second phase the adversary can also interact with all tags, except for one tag $T_c \in S$, where $S \subset \mathcal{T}$ is the set of tags for which a grouping proof is produced by the adversary. This tag T_c should also remain uncorrupted during the entire game. The adversary outputs a candidate grouping proof σ at the end of the second phase. A grouping proof scheme is sound if no polynomially bounded strong adversary, with non-negligible probability, is able to produce a valid grouping proof for a set of tags S with time $t_2 > t_1$.

The above definition ensures that even if all tags but one participating in the yoking protocol collude it remains impossible to construct a valid grouping proof without cooperation of all tags.

A non-timed proof is restricted to proving that the tags in question were together and completed the protocol. One cannot in any way determine from the yoking proof at what time this happened. As such, once a proof is produced it can be reused without limits.

Definition 3 (Non-Timed Grouping Proof Soundness)**.** In the first phase of the soundness game the adversary may interact with all tags, except T_a. This also implies that corrupting T_a is impossible in the first phase.

In the second phase the adversary cannot interact with any tag except T_a. The adversary outputs a candidate grouping proof σ at the end of the second phase.

A grouping proof scheme is sound if no polynomially bounded strong adversary, with non-negligible probability, is able to produce a valid grouping proof for the group of tags $S = \{T_a, T_b, \ldots\}$, even when allowed to corrupt T_a in the second phase.

During the entire game T_b cannot be corrupted.

By splitting the soundness game in two phases we ensure that at least two of the tags in the grouping proof cannot perform a yoking protocol together. In the first phase T_a cannot be used, but T_b can, while in the second phase only T_a can be used.

4 Yoking Proof with Trusted Party

Figure 2 presents our new protocol, which is based on the Randomised Schnorr protocol [4] to ensure soundness as well as tag privacy. The exam e is generated by the trusted time stamping authority (TTSA) after receiving the tags' commitments $R_{a1}, R_{a2}, R_{b1}, R_{b2}$. This ensures the proper ordering of the messages in the authentication protocol, necessary to avoid crooked proofs. Given the exam, each tag generates a response s_a, s_b. The TTSA finally signs all messages and the timestamp provided the final values s_a, s_b arrive before the session with the TTSA times out. The signature is returned to the reader, who stores the full grouping proof σ for later verification.

Note that neither the reader, nor the TTSA are able to learn the identity of the tags. The grouping proof can only be checked by the verifier with secret key y. The proof is verified as follows:

- $\texttt{verify}(s_{TTSA})$;
- $X_a = e^{-1}(s_a P - R_{a1} - y^{-1} R_{a2})$;
- $X_b = e^{-1}(s_b P - R_{b1} - y^{-1} R_{b2})$.

The public keys X_a, X_b can be checked in the database of the verifier. This ensures that tag T_a and tag T_b were scanned together at time ts.

The main cost for each tag is two scalar-EC point multiplications. The most complex operation, the signature, is performed by the TTSA.

Our protocol can easily be extended to multiple tags, at no additional cost for the RFID tags.

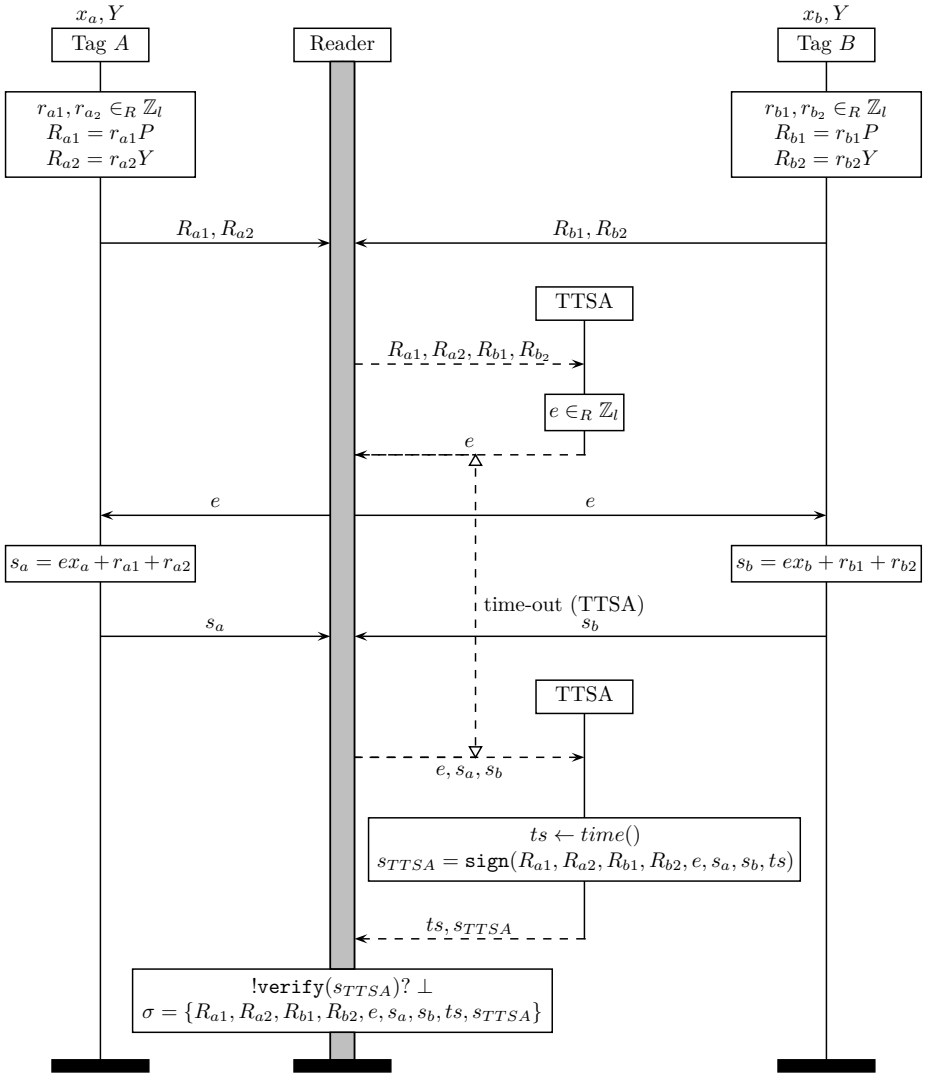


Figure 2: Two-party grouping-proof protocol with timestamp.

4.1 Security and Privacy

Grouping Proof Soundness

The soundness of the grouping proof is based on the one more discrete logarithm (OMDL) assumption, which was introduced by Bellare *et al.* [2]. Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ. Let $\mathcal{O}_1$ be an oracle that returns random elements $A_i = a_i P$ of $\mathbb{G}_\ell$. Let $\mathcal{O}_2(\cdot)$ be an oracle that returns the discrete logarithm of a given input base P. The OMDL problem is to return the discrete logarithms for each of the elements obtained from the m queries to $\mathcal{O}_1$, while making strictly less than m queries to $\mathcal{O}_2(\cdot)$.

Theorem 1. *The protocol from Fig. 2 is timed grouping proof sound under the OMDL assumption and the existential unforgeability of the signature scheme used by the TTSA.*

Proof. Assume an adversary $\mathcal{A}$ that forges the timed grouping proof.

We now construct an adversary $\mathcal{B}$ that breaks the unforgeability of the signature σ, or an adversary $\mathcal{B}$' that breaks the OMDL.

If $\mathcal{A}$ produces a σ with timestamp $t_2 > t_1$ this implies that either it communicated at time t_2 with the TTSA to produce σ or that $\mathcal{A}$ forged the signature. In the latter case we can easily use $\mathcal{A}$ to break the existential unforgeability of the signature scheme.

From now on we can assume that the messages $R_{a1}, R_{a2}, R_{b1}, R_{b2}, e,\ s_A,\ s_B$ where faithfully exchanged with the TTSA around time t_2 using the `SendTTS` oracle.

Let $X_A = \mathcal{O}_1()$. In the first phase $\mathcal{B}$' simulates the i'th pair of `SendTag` queries to tag T_a as follows:

- First $\texttt{SendTag}() \rightarrow R_{a1,i}, R_{a2,i}$: $R_{a1,i} = \mathcal{O}_1()$, $r_{a2,i} \in_R \mathbb{Z}_l, R_{a2,i} = r_{a2,i}Y$
- Second First $\texttt{SendTag}(e) \rightarrow s_A$: $s_{A,i} = \mathcal{O}_2(e_i X_A + R_{a1,i}) + r_{a2,i}$

In the second phase, the adversary $\mathcal{A}$ calls `SendTTS` with $R_{a1}, R_{a2}, R_{b1}, R_{b2}$. $\mathcal{B}$' simulates `SendTTS` by generating a random e after which $\mathcal{A}$ will call `SendTTS` with s_A and s_B. Upon receiving these, $\mathcal{B}$' rewinds $\mathcal{A}$ until the moment it calls `SendTTS` with $R_{a1}, R_{a2}, R_{b1}, R_{b2}$ and sends back a fresh e' after which $\mathcal{A}$ will send new s'_A and s'_B to the TTSA. $\mathcal{B}$' can now recover $x_A = (s'_A - s_A)/(e' - e)$, and returns $x_A, \{s_{A,i} - e_i x_A - r_{a2,i}\}_i$ to the OMDL challenger, thereby solving the OMDL problem. □

Privacy

The privacy of the protocol is based on the decisional Diffie Hellman (DDH) assumption. Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ. Let $a, b, r \in_R \mathbb{Z}_l$ and $A = aP, B = bP$. The DDH assumption states that is hard to distinguish between $(A, B, C = abP)$ and $(A, B, C = rP)$.

Theorem 2. *The protocol from Fig. 2 is narrow strong* private under the DDH assumption.*

Note that the protocol uses randomized Schnorr, which has been proven narrow strong private in [4]. Below we give a modified proof for the [8] model, using a standard hybrid argument [6,17].

Proof. For simplicity we will only consider a single execution of the protocol. A full proof can be obtained by using a standard hybrid argument.

Assume an adversary $\mathcal{A}$ that breaks narrow strong privacy. We will create a adversary $\mathcal{B}$ that breaks DDH (with $A = aP, B = bP$ and $C = abP$ or $C = rP$) which executes $\mathcal{A}$. $\mathcal{B}$ sets $Y = B$ at the beginning, chooses a random bit b and simulates the `SendTag` oracles for a single protocol run to $\mathcal{A}$ as follows:

- First `SendTag`$(vtag)$: select $r \in_R \mathbb{Z}_l$ and return $R_{a1} = r'P - A, R_{a2} = C$
- Second `SendTag`$(vtag, e)$: $s_A = ex_i + r'$ where x_i is either the secret key of tag T_i or T_j, depending on the tags passed to the `DrawTag` that generated $vtag$ and the random bit b.

At the end of the game $\mathcal{A}$ outputs a guess bit g. $\mathcal{B}$ outputs $(b == g)$ as output to the DDH challenger.

In case of a real DDH instance (i.e. $C = abP$) the simulation perfectly follows the real protocol, hence it follows that $\Pr[\mathcal{B}\,\text{wins}]_{realddh} = \Pr[\mathcal{A}\,\text{wins}]$. In case of a random DDH instance (i.e. $C = rP$) $\mathcal{A}$ only obtains randomized, independent data and as such $\Pr[\mathcal{B}\,\text{wins}]_{randomddh} = \frac{1}{2}$.

It follows that $\mathbf{Adv}_{\mathcal{B}} = \frac{1}{2}\mathbf{Adv}_{\mathcal{A}}$. □

5 Yoking Proof without Trusted Parties

In case no trusted parties are available we have to rely on some form of signature (or MAC) for validation of the grouping proof. We cannot rely on

authentication protocols since ordering of messages is not guaranteed when validation takes place off-line.

Figure 3 shows the proposed protocol to generate a yoking proof. In the first round, both tags T_a and T_b generate a one-time key pair for signing, with public key R_a and private key r_a. In the second round, both tags MAC both public keys with their permanent private key x_a. Note that one can also use a signature scheme instead of a MAC. In the final round, both tags sign the MAC's s_a, s_b using their one-time signing key.

One possible instantiation of the signature scheme is a Schnorr signature [15], which requires ECC and a hash function. The hash function can also be reused for the MAC function (or the MAC can be replaced with a signature). However, MAC functions and Schnorr signatures do not guarantee privacy. In Appendix B we show how to make privacy preserving signatures, which can replace the MAC function to ensure narrow strong privacy. Note that the final signature does not need to be privacy preserving as the signing key is freshly generated for every protocol run.

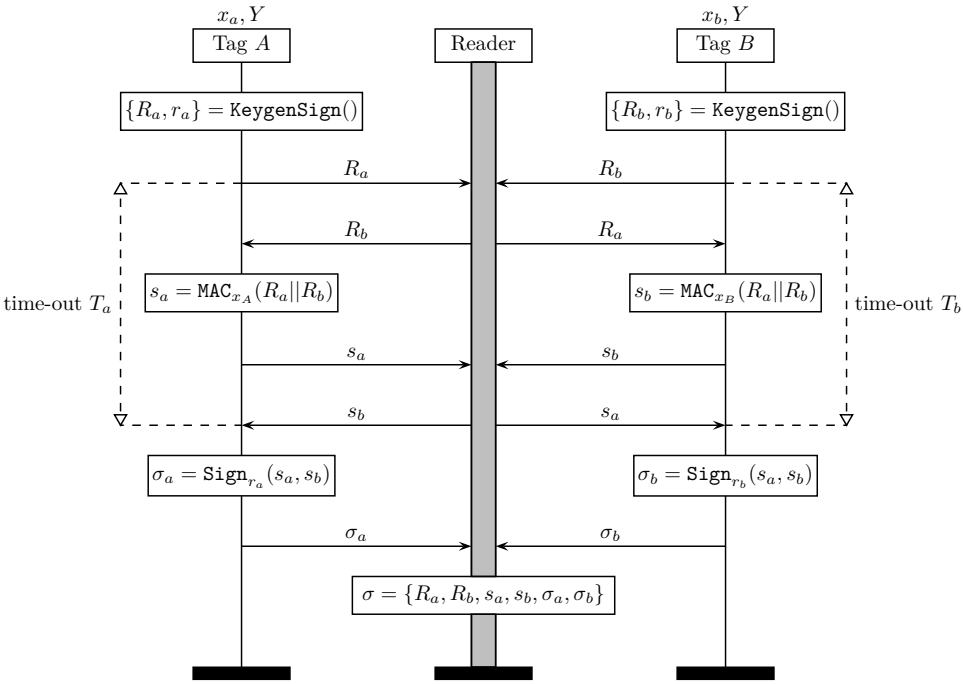


Figure 3: Two-party grouping-proof protocol without trusted party.

Our protocol can easily be extended to multiple tags, at the cost of additional communication. The computational overhead will remain small, since the number of signatures (and MACs) a tag needs to compute are independent of the number of tags in the grouping proof. However, since the messages that need to be signed (or on which the MAC algorithm needs to be deployed) increase in size, the computational effort will slightly raise.

5.1 Security proof

Theorem 3. *The protocol from Fig. 3 is non-timed grouping proof sound under the existential unforgeability of the MAC and the (one-time) signature scheme.*

Proof. Assume an adversary $\mathcal{A}$ that breaks the non-timed grouping proof soundness. We will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks either the existential unforgeability of the MAC or the signature scheme.

$\mathcal{B}$ runs $\mathcal{A}$ internally and simulates the grouping proof challenger to $\mathcal{A}$. At the start of the grouping proof game $\mathcal{B}$ sets $X_b = \texttt{KeygenMAC}$. During the first phase of the grouping proof challenge $\mathcal{B}$ simulates the `SendTag` oracle of T_b to $\mathcal{A}$ as follows:

- First $\texttt{SendTag}() \rightarrow R_b$: return $R_b = \texttt{KeygenSign}$.
- Second $\texttt{SendTag}(R_a) \rightarrow s_b$: return $s_b = \texttt{MAC}(R_a||R_b)$
- Third $\texttt{SendTag}(s_a) \rightarrow \sigma_b$: return $\sigma_b = \texttt{Sign}(s_a, s_b)$

All other oracle queries are simulated according to the protocol specification. In the second phase of the grouping proof game, $\mathcal{B}$ generates a random x_a and passes this to $\mathcal{A}$. At the end of the game $\mathcal{A}$ outputs a $\sigma = \{R_a, R_b, s_a, s_b, \sigma_a, \sigma_b\}$.

By assumption, σ is a valid grouping proof, implying that s_b is a valid MAC. If s_b was not requested during the first phase through the `MAC` oracle with $R_a||R_b$, this implies that s_b is a valid forgery and $\mathcal{B}$ breaks the existential unforgeability of the MAC scheme.

If, on the one hand, it was requested through a `MAC` oracle call, the definition of the simulation above by $\mathcal{B}$ to $\mathcal{A}$ implies that there also was a call to `KeygenSign`, which yielded the specified R_b. Since σ is a valid grouping proof, σ_b is valid signature on s_a, s_b using the private key matching to the public key R_b. If σ_b was not requested during the first phase through the `Sign` oracle, σ_b is a valid

forgery and $\mathcal{B}$ breaks the existential unforgeability of the one-time signature scheme.

If, on the other hand, σ_b was requested through a `Sign` oracle this implies that the full grouping proof presented by $\mathcal{A}$ took place in the first phase of the grouping proof challenge. This is impossible however, since x_a was only generated after the first phase. □

6 Conclusion

In this paper, we presented two attacks on the security of the yoking proofs as proposed by Batina *et al.* [1]. To ensure privacy of the RFID tags that take part in the protocol to generate a grouping proof, one should move away from the symmetric key cryptographic building blocks in favour of public key cryptography. Not only will this provide us with scalability at the verifier side, RFID tags will also have stronger privacy guarantees, i.e. narrow strong privacy. This paper introduced the first formal models of the security of yoking proofs. In the first model, time is taken into account, since for most use cases one is not only interested in two RFID tags being scanned together, but also when these tags were scanned together. In the second, we consider how to build a grouping proof without trusted third party. We provide for each model a protocol, for which both security and privacy are proven. Our proposed protocol with trusted timestamp authority is also the first one for which the verifier can upon verification of the yoking proof be absolutely sure that the tags were scanned at this point in time.

Acknowledgements

The authors would like to thank Fréderik Vercauteren for his valuable suggestion and interesting discussions. Additionally we appreciate the comments received from the anonymous reviewers.

This work was supported in part by the Research Council K.U.Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. In addition, this work was supported by the Flemish Government, IWT SBO MobCom and IWT Tetra EVENT. Jens Hermans is a research assistant, sponsored by the Fund for Scientific Research - Flanders (FWO).

References

[1] L. Batina, Y. K. Lee, S. Seys, D. Singelée, and I. Verbauwhede. Extending ECC-Based RFID Authentication Protocols to Privacy-Preserving Multi-Party Grouping Proofs. *Journal of Personal and Ubiquitous Computing*, 16(3):323–335, 2012.

[2] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology*, 16:185–215, 2003.

[3] L. Bolotnyy and G. Robins. Generalized "Yoking-Proofs" for a Group of RFID Tags. *MOBIQUITOUS*, pages 1–4, 2006.

[4] J. Bringer, H. Chabanne, and T. Icart. Cryptanalysis of EC-RAC, a RFID Identification Protocol. In *CANS*, volume 5339 of *Lecture Notes in Computer Science*. Springer, 2008.

[5] M. Burmester, B. de Medeiros, and R. Motta. Provably Secure Grouping-Proofs for RFID Tags. In G. Grimaud and F.-X. Standaert, editors, *CARDIS*, volume 5189 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2008.

[6] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, 2001.

[7] D. Hein, J. Wolkerstorfer, and N. Felber. ECC is Ready for RFID - A Proof in Silicon. In R. Avanzi, L. Keliher, and F. Sica, editors, *SAC*, volume 5381 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 2009.

[8] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A New RFID Privacy Model. In V. Atluri and C. Diaz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 568–587. Springer, 2011.

[9] A. Juels. "Yoking-Proofs" for RFID Tags. In *PERCOMW*, pages 138–143. IEEE Computer Society, 2004.

[10] Y. K. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-Cost Untraceable Authentication Protocols for RFID (extended version). In S. Wetzel, C. N. Rotaru, and F. Stajano, editors, *WiSec*, pages 55–64. ACM, 2010.

[11] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede. Elliptic Curve Based Security Processor for RFID. *IEEE Transactions on Computer*, 57(11):1514–1527, November 2008.

[12] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda. Solving the Simultaneous Scanning Problem Anonymously: Clumping Proofs for RFID Tags. In *SecPerU*. IEEE Computer Society Press, 2007.

[13] S. Piramuthu. On Existence Proofs for Multiple RFID Tags. In *SecPerU*, pages 317–320. IEEE, IEEE Computer Society Press, 2006.

[14] J. Saito and K. Sakurai. Grouping Proof for RFID Tags. In *AINA*, pages 621–624. IEEE Computer Society, 2005.

[15] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In G. Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[16] S. Vaudenay. On privacy models for RFID. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2007.

[17] A. C.-C. Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.

A Oracles Model Hermans et al.

The model of Hermans et al. [8] defines the following oracles for the privacy game:

- `CreateTag`$(ID) \rightarrow T_i$: on input a tag identifier ID, this oracle creates a tag with the given identifier and corresponding secrets, and registers the new tag with the reader. A reference T_i to the new tag is returned. Note that this does not reject duplicate IDs.
- `Launch`$() \rightarrow \pi$: this oracle launches a new protocol run on the reader R_j, according to the protocol specification. It returns a session identifier π, generated by the reader.
- `DrawTag`$(T_i,T_j) \rightarrow vtag$: on input a pair of tag references, this oracle generates a virtual tag reference, as a monotonic counter, *vtag* and stores the triple $(vtag, T_i, T_j)$ in a table $\mathcal{D}$. Depending on the value of b, *vtag* either refers to T_i or T_j. If T_i is already references as the left-side tag in $\mathcal{D}$ or T_j as the right-side tag, then this oracle also returns $\perp$ and adds no entry to $\mathcal{D}$. Otherwise, it returns *vtag*.

- $\texttt{Free}(vtag)_b$: on input $vtag$, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$. If $b = 0$, it resets the tag T_i. Otherwise, it resets the tag T_j. Then it removes the entry $(vtag, T_i, T_j)$ from $\mathcal{D}$. When a tag is reset, its volatile memory is erased. The non-volatile memory, which contains the state S, is preserved.
- $\texttt{SendTag}(vtag, m)_b \rightarrow m'$: on input $vtag$, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$ and sends the message m to either T_i (if $b = 0$) or T_j (if $b = 1$). It returns the reply from the tag (m'). If the above triple is not found in $\mathcal{D}$, it returns $\perp$.
- $\texttt{SendReader}(\pi, m) \rightarrow m'$: on input π, m this oracle sends the message m to the reader in session π and returns the reply m' from the reader (if any) is returned by the oracle.
- $\texttt{Result}(\pi)$: on input π, this oracle returns a bit indicating whether or not the reader accepted session π as a protocol run that resulted in successful authentication of a tag. If the session with identifier π is not finished yet, or there exists no session with identifier π, $\perp$ is returned.
- $\texttt{Corrupt}(T_i)$: on input a tag reference T_i, this oracle returns the complete internal state of T_i. Note that the adversary is not given control over T_i.

B Privacy Preserving Signatures

To obtain privacy preserving signatures with identification we make a slight modification to the Schnorr signature scheme [15]. The original Schnorr signature scheme works as follows:

- $r \in_R \mathbb{Z}_l$
- $e = H(M||rP)$, $s = ex + r$
- Output s, e.

In the modified scheme, rY, instead of e is provided together with s.

- $r \in_R \mathbb{Z}_l$
- $e = H(M||rP)$, $s = ex + r$
- Output s, rY.

The verifier can retrieve $rP = y^{-1}(rY)$ and as such compute e. By computing $e^{-1}(sP - rP) = X$, s is verified. By checking the database for a registered public key X, one obtains both identification and verification of the signature at the same time, provided the number of tags remains significantly lower than ℓ.

Privacy of this modified scheme can be shown under the DDH assumption. Existential unforgeability follows in the same way as for the Schnorr signature when the verifier is provided with y.

Curriculum Vitae

Jens Hermans was born on 9th October 1985 in Hasselt, Belgium. He received his Master's degree in Mathematical Engineering in July 2008 from KU Leuven, Belgium with the greatest distinction (*summa cum laude*) with the congratulations of the examination committee. For his Master's thesis on optimization of inland shipping he received an award from the Jos Schepens Memorial Fund. He served on the academic council and other university boards as a student delegate.

In October 2008, he joined the research group COSIC at the Department of Electrical Engineering (ESAT) of K.U.Leuven. His PhD research was sponsored by the Fund for Scientific Research, Flanders (FWO-Vlaanderen). He was a member of the visitation commission of the Flemish Interuniversity Council (VLIR) which evaluated the computer science and informatics programs at the flemish universities. He visited the Kryptographie und Computeralgebra research group at T.U.Darmstadt in July 2009 and the Department of Electrical Engineering at National Taiwan University in November 2009.

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering (ESAT)
Computer Security and Industrial Cryptography (COSIC)
Kasteelpark Arenberg 10, bus 2446
3001 Heverlee

KATHOLIEKE UNIVERSITEIT LEUVEN

ASSOCIATIE K.U.LEUVEN