

# The Cryptographic Security of the German Electronic Identity Card

Vom Fachbereich Informatik der  
Technischen Universität Darmstadt genehmigte

## Dissertation

zur Erlangung des Grades  
Doktor rerum naturalium (Dr. rer. nat.)

von

**M.Sc. Inf. Dipl.-Math. Özgür Dagdelen**

geboren in Darmstadt, Deutschland



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Referenten: Prof. Dr. Marc Fischlin  
Technische Universität Darmstadt  
Prof. Dr. Miroslaw Kutylowski  
Wroclaw University of Technology

Tag der Einreichung: 2. Mai 2013  
Tag der mdl. Prüfung: 13. Juni 2013

Hochschulkennziffer: D 17

Darmstadt 2013

## Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Darmstadt, \_\_\_\_\_  
Date Signature

## Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Darmstadt, \_\_\_\_\_  
Datum Unterschrift

## Wissenschaftlicher Werdegang

- Oktober 2003 - Mai 2006.** Studium der Informatik an der Technischen Universität Darmstadt zur Erlangung des Grades "Bachelor of Science"
- Oktober 2004 - Mai 2009.** Studium der Mathematik an der Technischen Universität Darmstadt zur Erlangung des Grades "Diplom Math."
- Oktober 2006 - Mai 2009.** Studium der Informatik an der Technischen Universität Darmstadt zur Erlangung des Grades "Master of Science"
- Seit Juni 2009.** Wissenschaftlicher Mitarbeiter in der erst Emmy-Noether- und später Heisenberg-Forschungsgruppe "Minicrypt" und "Cryptoplexity" an der Technischen Universität Darmstadt

# List of Publications

Some ideas and figures have appeared previously in my following publications and manuscripts:

## Peer Reviewed Conference and Workshop Publications

- [ADK10] Sami Alsouri, Özgür Dagdelen, and Stefan Katzenbeisser. “Group-based Attestation: Enhancing Privacy and Management in Remote Attestation” *3rd International Conference on Trust and Trustworthy Computing - TRUST 2010.*, Lecture Notes in Computer Science, Volume 6101, pages 63-77, Springer-Verlag, 2010.
- [DF10] Özgür Dagdelen and Marc Fischlin. “Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents” *13th International Conference on Information Security - ISC 2010*, Lecture Notes in Computer Science, Volume 6531, pages 54-68, Springer-Verlag, 2010.
- [DS10] Özgür Dagdelen and Michael Schneider. “Parallel Enumeration of Shortest Lattice Vectors” *16th International Euro-Par Conference - Parallel Processing - Euro-Par (2) 2010*, Lecture Notes in Computer Science, Volume 6272, pages 211-222, Springer-Verlag, 2010.
- [BBD<sup>+</sup>10] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. “Redactable Signatures for Tree-structured Data: Definitions and Constructions” *8th International Conference on Applied Cryptography and Network Security - ACNS 2010*, Lecture Notes in Computer Science, Volume 6123, pages 87-104, Springer-Verlag, 2010.
- [MDCE11] Mohammed Meziani, Özgür Dagdelen, Pierre-Louis Cayrel, and Sidi Mohamed El Yousfi Alaoui. “S-FSB: An Improved Variant of the FSB Hash Family” *Information Security and Assurance - ISA 2011*, Communications in Computer and Information Science, Volume 200, pages 132-145, Springer-Verlag, 2011.

- [KSD<sup>+</sup>11] Po-Chun Kuo, Michael Schneider, Özgür Dagdelen, Jan Reichelt, Johannes Buchmann, Chen-Mou Cheng, and Bo-Yin Yang. “Extreme Enumeration on GPU and in Clouds - - How many Dollars you need to break SVP Challenges”. *Cryptographic Hardware and Embedded Systems - CHES 2011*, Lecture Notes in Computer Science, Volume 6917, pages 176-191, Springer-Verlag, 2011.
- [BDF<sup>+</sup>11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. “Random Oracles in a Quantum World” *Advances in Cryptology - ASIACRYPT 2011*, Lecture Notes in Computer Science, Volume 7073, pages 41-69, Springer-Verlag, 2011.
- [CSD<sup>+</sup>12] Lassaad Cheikhrouhou, Werner Stephan, Özgür Dagdelen, Marc Fischlin, and Markus Ullmann. “Merging the Cryptographic Security Analysis and the Algebraic-logic Security Proof of PACE” *GI SICHERHEIT 2012 Sicherheit - Schutz und Zuverlässigkeit*, Lecture Notes in Informatics, Volume 195, pages 83-94, GI, 2012.
- [BDF12] Christina Brzuska, Özgür Dagdelen, and Marc Fischlin. “TLS, PACE, and EAC: A Cryptographic View at Modern Key Exchange Protocols” *GI SICHERHEIT 2012 Sicherheit - Schutz und Zuverlässigkeit*, Lecture Notes in Informatics, Volume 195, pages 71-82, GI, 2012.
- [BDFK12b] Jens Bender, Özgür Dagdelen, Marc Fischlin, and Dennis Kügler. “The PACE|AA Protocol for Machine Readable Travel Documents, and its Security” *16th International Conference on Financial Cryptography and Data Security - FC 2012*, Lecture Notes in Computer Science, volume 7397, pages 344-358, Springer-Verlag, 2012.
- [BDFK12a] Jens Bender, Özgür Dagdelen, Marc Fischlin, and Dennis Kügler. “Domain-Specific Pseudonymous Signatures for the German Identity Card” *15th International Conference on Information Security - ISC 2012*, Lecture Notes in Computer Science, Volume 7483, pages 104-119, Springer-Verlag, 2012.
- [ADV<sup>+</sup>12] Sidi Mohamed El Yousfi Alaoui, Özgür Dagdelen, Pascal Véron, David Galindo, and Pierre-Louis Cayrel. “Extended Security Arguments for Signature Schemes” *5th International Conference on Cryptology in Africa - AFRICACRYPT 2012*, Lecture Notes in Computer Science, Volume 7374, pages 19-34, Springer-Verlag, 2012.
- [DMV13] Özgür Dagdelen, Payman Mohassel, and Daniele Venturi. “Rate-Limited Secure Function Evaluation: Definitions and Constructions” *16th International Conference on Practice and Theory in Public-Key Cryptography - PKC 2013*, Lecture Notes in Computer Science, Volume 7778, pages 461-478, Springer-Verlag, 2013.
- [DFG<sup>+</sup>13b] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia

Azzurra Marson, Arno Mittelbach and Cristina Onete. “A Cryptographic Analysis of OPACITY” *to appear at ESORICS 2013*, 2013. <http://eprint.iacr.org/2013/234.pdf>.

### Non-Refereed Articles

- [**ADDV12**] Giuseppe Ateniese, Özgür Dagdelen, Ivan Damgård, and Daniele Venturi. “Entangled Cloud Storage” *Cryptology ePrint Archive*, Report 2012/511, 2012. <http://eprint.iacr.org/>.
- [**DF12**] Özgür Dagdelen and Marc Fischlin. “Unconditionally-Secure Universally Composable Password-Based Key-Exchange based on One-Time Memory Tokens” *Cryptology ePrint Archive*, Report 2012/537, 2012. <http://eprint.iacr.org/>.
- [**DFG13a**] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. “The Fiat-Shamir Transformation in a Quantum World” *Cryptology ePrint Archive*, Report 2013/245, 2013. <http://eprint.iacr.org/>.

## Acknowledgments

It is impossible to put into words my gratitude for my supervisor Marc Fischlin. During my graduation, he's become my personal hero; I consider myself lucky, and I am truly grateful, for being able to siphon a little of his genius into my empty glass. Special thanks also go to Johannes Buchmann, whose powerful presentations and lectures because in cryptography first piqued my interest towards this topic.

I'd also like to give heartfelt thanks to those colleagues who became important to me throughout these last four years; I've grown quite fond of you all! More specifically, I am deeply grateful to (sorted in the order I've met them) Anja Lehmann, Dominique Schröder, Cristina Onete, Christina Brzuska, Paul Baecher, Pooya Farshin, Giorgia Azzurra Marson, Tommaso Gagliardoni, and Arno Mittelbach. Apart from the Cryptoplexity gang, I've also shared great times and research ideas with the following folks, whom I am also very grateful to: Bertram Poettering, Andreas Peter, Heike Schröder, Sidi Mohamed El Yousfi Alaoui, Mohammed Meziani, and Michael Schneider. I collected a pack of experience together with you, guys. You've been an important part of the most amazing time of my life.

I would also like to thank my co-authors, with whom I've had a lot of fruitful discussions: Payman Mohassel, Guiseppe Ateniese, Ivan Damgard, and Sami Alsouri. Special thanks go to Daniele Venturi. He is not only a colleague and fellow researcher, but become one of my best, "powerful" friends, a brother from another mother.

For their support I also thank the German Federal Office for Information Security (BSI), in particular to Dennis Kügler, Jens Bender, and Markus Ullmann. I am grateful you had confidence in my abilities to work on the interesting and challenging topic of the German Identity Card. Without them, this thesis would not have existed,

Last but not least, I would like to very gratefully dedicate this work to my family and to my soulmate Fatima Sialiti. My success is due to them: to their patience, sacrifices, and never-ceasing support, not just during these four

years, but my entire life. They are my strength and compass in life, and always there for me. I hope you are proud of me and will always remain so. You are always in my heart.

Özgür Dagdelen



## Abstract

In November 2010, the German government started to issue the new electronic identity card (eID) to its citizens. Besides its original utilization as a 'visual' identification document, the eID card can be used by the cardholder to prove one's identity at border control and to enhance security of authentication processes over the Internet, with the eID card serving as a token to reliably transmit personal data to service providers or terminals, respectively. To this end, the German Federal Office for Information Security (BSI) proposed several cryptographic protocols now deployed on the eID card.

The Password Authenticated Connection Establishment (PACE) protocol secures the wireless communication between the eID card and the user's local card reader, based on a cryptographically weak password like the PIN chosen by the card owner. Subsequently, the Extended Access Control (EAC) protocol is executed by the chip and the service provider to mutually authenticate and agree on a shared secret session key. This key is then used in the secure channel protocol, called Secure Messaging (SM). Finally, an optional protocol, called Restricted Identification (RI), provides a method to use pseudonyms such that they can be linked by individual service providers, but not across different service providers (even not by malicious ones).

This thesis consists of two parts. First, we present the above protocols and provide a rigorous analysis on their security from a cryptographic point of view. We show that the German eID card provides reasonable security for authentication and exchange of sensitive information allaying concerns regarding its usage.

In the second part of this thesis, we introduce two possible modifications to enhance the security of these protocols even further. Namely, we show how to (a) add to PACE an additional efficient chip authentication step, and (b) augment RI to allow also for signatures under pseudonyms.



# Zusammenfassung

In November 2010 wurde der neue elektronische Personalausweis (nPA) in Deutschland eingeführt. Neben der ursprünglichen Verwendung als 'optisches' Identifikationsdokument, kann ein Ausweiseigentümer den nPA verwenden, um sich in Grenzkontrollen auszuweisen oder die Sicherheit bei dem Authentisierungsvorgängen im Internet zu verstärken. Der nPA wirkt hier als Mittel, um zuverlässig persönliche Daten zu einem Dienstanbieter oder dem Terminal zu transferieren. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) führte mehrere kryptographische Protokolle ein, welche nun im nPA eingesetzt werden.

Das Password Authenticated Connection Establishment (PACE) Protokoll, welches auf kryptographisch schwachen Passwörtern basiert, wie die von Nutzern ausgewählten PIN, sichert die kabellose Kommunikation zwischen dem nPA und dem Lesegerät des Nutzers. Anschließend wird das Extended Access Control Protokoll zwischen der Chipkarte und dem Diensteanbieter ausgeführt, um sich gegenseitig zu authentifizieren und sich auf einen gemeinsamen geheimen Sitzungsschlüssel zu einigen. Dieser Schlüssel wird dann in dem Protokoll für sichere Kanäle mit dem Namen Secure Messaging (SM) verwendet. Schlussendlich bietet ein optionales Protokoll namens Restricted Identification (RI) eine Methode an, Pseudonyme zu verwenden, welche innerhalb individuellen Diensteanbieter wiedererkennbar sind, allerdings anonym gegenüber anderen (selbst böswilligen) Diensteanbieter bleiben.

Diese Dissertation besteht aus zwei Teilen. Zum einen präsentieren wir die oben genannten Protokolle und liefern eine gründliche Sicherheitsanalyse aus kryptographischen Gesichtspunkt. Wir zeigen, dass der nPA eine vernünftige Sicherheit für eine Authentifikation und den Austausch von sensiblen Daten bietet, welche jegliche Bedenken den neuen elektronischen Personalausweis aus Sicherheitsgründen nicht zu nutzen, ausräumen sollte.

Im zweiten Teil dieser Dissertation führen wir zwei mögliche Änderungen ein, welche die Sicherheit der Protokolle noch weiter erhöhen könnten. Genauer: wir zeigen wie (a) dem PACE Protokoll eine zusätzliche Chip Authentifizierung

hinzugefügt werden kann, und (b) das RI Protokoll so erweitert werden kann, dass Nutzer Signaturen unter ihren Pseudonymen erstellen können.

# Contents

<b>Abstract</b>	<b>ix</b>
<b>1. Introduction</b>	<b>1</b>
<b>1. Introduction</b>	<b>3</b>
1.1. The Protocols . . . . .	5
1.2. Security Analysis . . . . .	7
1.3. Future Enhancements . . . . .	9
1.4. Further Remarks . . . . .	10
1.5. Contributions of this Thesis . . . . .	12
1.6. Outline of this Thesis . . . . .	14
<b>2. Definitions</b>	<b>17</b>
2.1. General Notation . . . . .	17
2.2. Number-Theoretic Assumptions . . . . .	19
2.3. Basic Cryptographic Primitives . . . . .	22
<b>3. Security Model</b>	<b>29</b>
3.1. Proving Security in Cryptography . . . . .	29
3.2. Security Model for Authenticated Key Agreement . . . . .	31
3.2.1. The BR and BPR Security Model . . . . .	32
3.2.2. The eCK Security Model . . . . .	37
3.3. Security Model for Secure Channels . . . . .	38
3.4. Discussion on the Security Models . . . . .	41
3.4.1. Achieved Security Properties in the B(P)R Model . . . . .	41
3.4.2. Achieved Security Properties in the eCK Model . . . . .	42
3.4.3. Achieved Security Properties in the AKE+SC Model . . . . .	43

<b>II. The Protocols used by the German eID Card</b>	<b>45</b>
<b>4. The Password Authenticated Connection Establishment Protocol</b>	<b>47</b>
4.1. Protocol Description . . . . .	49
4.2. On the Security of PACE . . . . .	52
4.2.1. Results by Bender <i>et al.</i> . . . . .	52
4.2.2. Results by Coron <i>et al.</i> . . . . .	54
4.2.3. Further Security Studies on PACE . . . . .	55
<b>5. The Extended Access Control (EAC) Protocol</b>	<b>57</b>
5.1. Protocol Description . . . . .	59
5.1.1. Terminal Authentication . . . . .	59
5.1.2. Chip Authentication . . . . .	61
5.1.3. Extended Access Control . . . . .	62
5.1.4. Remarks . . . . .	63
5.2. Security Analysis . . . . .	66
5.3. Discussion . . . . .	73
5.3.1. EAC and the eCK Model . . . . .	73
5.3.2. On the Role of the Components . . . . .	75
5.3.3. On the Composition of the MRTD-Protocols . . . . .	77
<b>6. Secure Messaging</b>	<b>79</b>
6.1. Protocol Description . . . . .	79
6.2. Security Analysis . . . . .	80
<b>7. Restricted Identification</b>	<b>85</b>
7.1. Protocol Description . . . . .	86
7.2. Security Analysis . . . . .	88
7.2.1. Cross-Domain Anonymity . . . . .	88
7.2.2. Domain-Specific Linkability . . . . .	93
<b>8. On the Choice of Cryptographic Primitives</b>	<b>95</b>
8.1. Cryptographic Primitives . . . . .	95
8.2. Further Security Requirements . . . . .	99
8.2.1. Hardware Security . . . . .	100
8.2.2. Personalization . . . . .	100

<b>III. Future Enhancements</b>	<b>103</b>
<b>9. The PACE AA Protocol for MRTD, and its Security</b>	<b>105</b>
9.1. Protocol Description . . . . .	108
9.2. Security Model Adaptations . . . . .	110
9.3. Security Analysis . . . . .	113
9.3.1. Security Assumptions . . . . .	113
9.3.2. Security as a Key-Exchange Protocol . . . . .	116
9.3.3. Security against Impersonation . . . . .	119
9.4. A Deniable Schnorr Variant . . . . .	124
9.4.1. Defining Deniability . . . . .	124
9.4.2. Deniability of Our Protocol . . . . .	125
<b>10. Domain-Specific Pseudonymous Signatures</b>	<b>129</b>
10.1. Introduction . . . . .	129
10.2. Definitions . . . . .	133
10.2.1. Cross-Domain Anonymity . . . . .	134
10.2.2. Unforgeability . . . . .	138
10.2.3. Seclusiveness . . . . .	139
10.3. Construction . . . . .	141
10.4. Security Analysis . . . . .	142
10.4.1. Cross-Domain Anonymity . . . . .	143
10.4.2. Unforgeability . . . . .	145
10.4.3. Seclusiveness . . . . .	148
<b>Bibliography</b>	<b>153</b>



**Part I.**

**Introduction**



# 1. Introduction

In more than 100 countries around the world, national identity cards are issued to their citizens. These identity cards can be used as travel documents — equivalent to passports — (as long as they are accepted in the destination country) as well as a proof of identity within their home country. Only few nations do not require their citizens to own and use an identity card, e.g., Australia, Japan, and United States. Usually, identity cards are more compact in their physical form than passports and the handy size makes them easier to carry around in a daily life. Additionally, more information about the card owner is often displayed on identity cards than passports reveal, such as the home address, height of the person, and eye color.

Nowadays, traditional identity cards (made of paper-core laminated at both sides) are replaced and updated more and more by identity cards in which an electronic RFID chip is embedded. These electronic identity (eID) cards allow one not only to identify and authenticate oneself to a person or to a machine which has physical access to the card, but also to a remote machine far from the eID card and its owner. For this, the card owner requires a card reader, which communicates with the eID card over a wireless channel, and Internet connection. Obviously, online authentication — that is, over the Internet — is not an option when the card is used as a travel document; however, certain services which require a person to identify via an identity card or passport, can now be accessed through the Internet. This includes services like eGovernment, eBanking, or eCommerce. For instance, one can open a new bank account without being physically present in the bank. Preannouncing, in Cebit 2013 in Hannover, the biw-bank provided an automatic cash dispenser to German citizens where one was able to withdraw money only using his/her eID card [Hei13]. What services are provided exactly and under which conditions they can be utilized is determined by the corresponding countries.

In Germany, any person aged 16 or older must possess an identity card or a passport. In 1951, Germany issued for the first time (traditional) identity cards to their citizens containing, among others, the name, birth date and birth place of the card owner, as well as an ID picture and a handwritten signature.

The paper-made identity card is currently being gradually replaced by an electronic identity card, a process that started in November 2010. The German eID card displays all the information included in the older paper cards, but in addition stores biometric data (i.e., fingerprints) of the card owner and a digital copy of his or her picture in the RFID chip on the card. This hardware chip allows to communicate with a card reader over a short distance through a wireless connection. Optionally, one can store information for electronic signatures on the card, possibly provided by a private company. Two years after its introduction, there were 45 companies and departments offering more than 87 applications or services for eID users [Hei12].

Identity cards require protection against copying, cloning, and forgery in order to prevent that one misuses an identity card or impersonates a different person — possibly a non-existent person. For this reason, during the production of these cards, identity-card issuers use countermeasures which allow to physically check the correctness and validity of the identity card at hand. For instance, on the German eID card, parts are written with color-changing ink, and the card includes 2D and 3D holographic security elements, and fluorescent elements which luminesce in different colors under ultraviolet light. Also, parts of the card's memory, where the secret key material is stored, should be resistant against tampering and no one, not even the card owner, should be able to read from or write on this memory.

As the authentication can now take place online over the Internet, all messages sent and received by the eID card must be protected against eavesdroppers and malicious parties which try to illicitly access sensitive information about the card owner and/or his or her transactions. Also, the card owner must choose a personal identification number (PIN) which needs to be entered to the card reader in order to authenticate. In order to secure the card's authentication over the Internet, several cryptographic protocols are implemented on the German eID card. These protocols aim to provide authenticity, integrity, and privacy of user data. In this thesis, we investigate the security of these protocols from a cryptographic point of view. That is, we look at the mathematical abstraction of these protocols (as defined in the specifications [BSI10]) and give security arguments under reasonable assumptions.

The security of the actual implementation is not part of our analysis, and it must thus be studied further. Also, the design choices of the protocols are not part of this thesis. Nonetheless, in the second part of the present work, we propose future extensions to facilitate the authentication process and provide

additional security features while making as little modification to existing protocols as possible.

Next, we introduce the cryptographic protocols used by the German eID card and present our results on their security.

### 1.1. The Protocols

The cryptographic protocols deployed on the German eID card build a secure communication between the chip card, and a card reader or a terminal, respectively. In Figure 1.1 we depict an overview of all the relevant cryptographic protocols. Here, we briefly introduce these protocols and refer to the subsequent chapters for a detailed description and the corresponding security analysis.

#### THE PASSWORD-AUTHENTICATED CONNECTION ESTABLISHMENT PROTOCOL.

Through ISO/IEC JTC1 SC17 WG3/TF5 [ISO10] the International Civil Aviation Organization (ICAO) has adopted the Password Authenticated Connection Establishment (PACE) protocol [BSI10] to secure the contactless communication between machine-readable travel documents (including identity cards) and a reader. Roughly, the protocol generates a secure Diffie–Hellman key out of a low-entropy password — the PIN for the German eID card — which the owner has to enter at the reader, or which is transmitted through a read-out of the machine-readable zone. This (hashed) Diffie–Hellman key is subsequently used to secure the communication through the Secure Messaging protocol which we describe below.

**THE EXTENDED ACCESS CONTROL PROTOCOL.** The Extended Access Control (EAC) protocol was proposed initially in 2005 by the German Federal Office for Information Security (BSI) for the electronic passports (ePASS). It provides a secure key establishment between a chip card and a terminal, using a public-key infrastructure. The new version of EAC, recommended and introduced in November 2010 for the German ID card, is presented in this thesis (with some slight simplifications for the sake of presentation, but without violation of security properties of the overall protocol). EAC serves the purpose of limiting access to the sensitive data stored on the chip card (e.g., fingerprints). The

## 1. Introduction

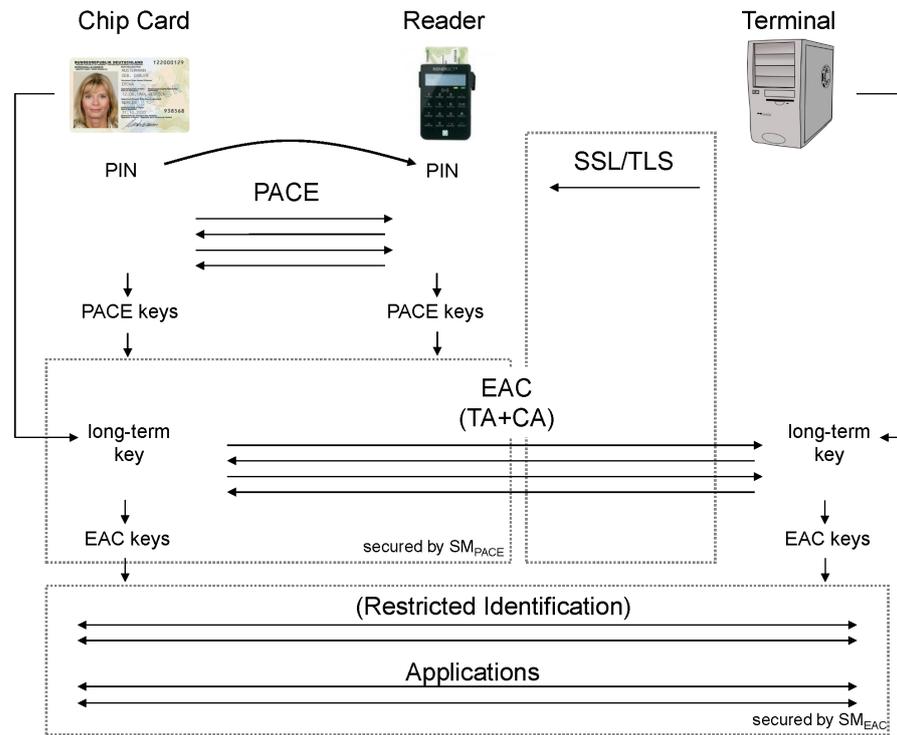


Figure 1.1.: Overview of the cryptographic protocols

BSI integrated EAC in the German electronic identity (eID) card to have a complete protection of all recorded personal data.

The EAC protocol consists of two phases: the Terminal Authentication (TA) protocol which is a challenge-response protocol in which the terminal signs a random challenge (and an ephemeral public key) with its certified signing key; and the Chip Authentication (CA) protocol in which both parties derive a Diffie-Hellman key from the terminal's ephemeral key and the chip's static, certified key, and where the chip finally computes a message authentication code to authenticate.

Note that the PACE protocol should be executed first; then the communication between the card and the terminal is secured through the EAC protocol. We note that the reader and the terminal should also be connected securely through, say, SSL/TLS, before the keys between the chip and the terminal are derived in the EAC protocol. We comment on security issues related to

protocol composition in Chapter 5.

**SECURE MESSAGING.** The Secure Messaging (SM) protocol as specified in [BSI10] constructs an end-to-end integrity-preserving secure channel, which requires a secret key shared by the chip card and the reader or terminal, respectively, and enables the parties to communicate securely over an insecure channel. In particular, SM provides confidentiality and authenticity for the exchanged messages. It uses the Encrypt-then-Authenticate paradigm which implements a secure channel if the input keys are uniformly distributed [CK01].

The Secure Messaging protocol is run twice. First, the messages between the chip card and the card reader are secured using the session key derived by PACE. Second, the keys derived by running EAC serve as input to SM to provide end-to-end security. In Figure 1.1 we denote by  $SM_{PACE}$  the SM protocol with input the PACE keys, and, respectively,  $SM_{EAC}$  denotes the protocol run for EAC keys.

**RESTRICTED IDENTIFICATION.** The overall design of the identity card includes another protocol, called Restricted Identification (RI). In this optional protocol, card holders can use domain-specific pseudonyms to interact with service providers such that (a) a service provider can recognize pseudonyms of individual cards and use this information for the service (domain-specific linkability), and (b) different service providers cannot link interactions of one user in their respective domains (cross-domain anonymity).

The Restricted Identification protocol is run as soon as a secure channel is built between the chip card and the corresponding service provider by running  $SM_{EAC}$ . Afterwards, any card application can be run as long as pertaining communication is secured and confidential.

## 1.2. Security Analysis

In this thesis we investigate the security of all the aforementioned protocols from a cryptographic point of view. As such, we classify the protocols according to their purpose and identify strong security models capturing the desired security properties. We also propose two novel protocols enhancing and updating the previous protocols. First, we present our results about the security

of cryptographic protocols implemented currently on the German eID card. We remark that this thesis focuses on the *security* analysis of cryptographic protocols. Performance analysis and design choices for these protocols are not investigated here.

In [BFK09] it has been shown that the PACE protocol achieves the widely accepted security notion of password-based authenticated key agreement of Bellare, Pointcheval, and Rogaway [BPR00], in its strong form stated by Abdalla *et al.* [AFP05]. This holds under a variant of the Diffie–Hellman assumption, assuming secure cryptographic building blocks, and idealizing the underlying block cipher, as well as the hash function. Following the initial analysis of PACE [BFK09], further studies were published regarding the security of the PACE protocol [CS10, CGIP12, CSD<sup>+</sup>12].

Similarly to PACE, the EAC protocol also allows two parties to agree on a secret shared key; however, rather than relying on a secret password as input, the security of the EAC protocol is based on public-key infrastructure. All chip cards and terminals possess a unique secret value whose corresponding public value is revealed in order to authenticate and derive a session key. A party’s secret value remains secret throughout lifetime. We analyze the EAC protocol as an authenticated key-exchange protocol in the strong security model by Bellare and Rogaway [BR94a]. We show that the protocol is secure in this model, assuming that the underlying cryptographic primitives are secure (signatures, certification, and message authentication codes). Similarly to PACE, the deployed hash function for key derivation is assumed to behave ideally, and a Diffie–Hellman assumption is required. We mention that the security of the earlier version of EAC, implemented on electronic passports, was assessed in [MVV07, PPW08, SBPV08]. The revised version — that is, the up-to-date and currently deployed version in the German eID card — was modified accordingly. The security of the latest version of EAC was an unresolved question previous to our analysis.<sup>1</sup>

We give a monolithic security analysis on the Secure Messaging protocol. That is, we define an appropriate security model where the secure channel protocol is executed after the key agreement took place. To this end, we build our model upon on the Bellare-Rogaway framework for authenticated key exchange, which allows us to use known results on the security of the underlying key-exchange protocol (in our case: PACE and EAC). We prove the SM protocol secure in this monolithic security model, which tells us that SM

---

<sup>1</sup>In [Nit09], the author briefly discusses whether attacks on previous protocols apply to EAC. However, he does not provide a rigorous security analysis.

implements an end-to-end integrity-preserving confidential channel between the participants. In particular, our security proof implies immediately that not only the single components are secure, but their compositions as well.

Restricted Identification (RI) allows users to use domain-specific pseudonyms. From a security and privacy point of view, these pseudonyms should be unique within a domain (domain-specific linkability) and moreover, two service providers should be unable to link pseudonyms of the same card across domains (cross-domain anonymity). In this context, we are first to introduce formal security notions for cross-domain anonymity and prove RI secure in this model. We also argue on the domain-specific linkability.

### 1.3. Future Enhancements

Besides investigating the security of the protocols used by the German eID card, we further propose two new cryptographic protocols building on the aforementioned protocols. In particular, we introduce the following enhancements for PACE and RI, respectively.

**PACE WITH ACTIVE AUTHENTICATION.** As already mentioned, PACE is introduced and standardized to secure the communication between a reader and machine-readable travel documents in general. It was implemented for the first time in the German eID card; however, it was designed primarily to replace the Basic Access Control (BAC) protocol for electronic passports. An optional protocol, called Active Authentication (AA), was introduced for electronic passports in order to prevent passport cloning (and identity theft). In this thesis we show how to merge PACE and AA in a single protocol, namely PACE|AA, which preserves all the security guarantees of the two separate protocols and, moreover, it improves the computation complexity — AA is run (almost) for free — when compared to the sequential execution of these protocols. We provide a rigorous security analysis in a security model slightly different from the one used for the analysis of PACE.

Active Authentication introduces a potential threats to privacy, as discussed in [MVV07, BPSV08a, BPSV08b, BSI10]. A terminal is able from the interaction with the passport to derive a proof for others *that* an interaction took place. Our protocol avoids this pitfall. We show that PACE|AA provides *deniable*

*authentication* [DDNoo]. This roughly means that a terminal could have generated the purported proof (essentially a transcript itself) from public data. Consequently, a passport owner can deny any interaction and claim that the terminal itself has fabricated this conversation.

**DOMAIN-SPECIFIC PSEUDONYMOUS SIGNATURES.** Although the concept of Restricted Identification in [BSI10] currently only supports recognition of pseudonyms, it can be easily extended to provide additional functionalities, by, for example, allowing users to output signatures under their pseudonyms. In this thesis we propose a novel signature scheme which builds upon RI. As such we identify several required security properties for such signatures, which we roughly define as follows:

**Cross-Domain Anonymity.** As for RI, signatures generated under user's pseudonym should also not allow service providers to link transactions of users across domains.

**Unforgeability.** As for common digital signature schemes, the standard security notion of unforgeability must also hold for our domain-specific pseudonymous signatures. That is, no one can generate signatures on behalf of an honest card except the legitimate card.

**Seclusiveness.** This property guarantees that no one is able to generate signatures under a pseudonym which is not (yet) assigned such that the signature is accepted by the terminal.

We are first to give a formal security definition of these security properties for pseudonym generated signatures. We also propose a construction satisfying all security requirements, which, moreover, builds upon RI enabling an easy integration into cryptosystems (e.g., the German eID card) where RI is already implemented.

### 1.4. Further Remarks

**MEDIA COMMENTS ON THE GERMAN eID CARDS.** Since the introduction of the novel German electronic identity cards, several organizations scrutinized the security of the card in several attack scenarios. In this paragraph we comment on these ostensible potential threats and discuss how they relate to the results in this thesis.

In August 2010, the Chaos Computer Club (CCC) criticized one of the three possible card readers handed out to German citizens [Hei10b, Hei10a, Hei10d]. They claim that the basic reader, which uses the keyboard from the PC rather than its own keyboard, is inherently unsafe. In their attack scenario they assume that an adversary is able to install on the card owner's PC a keylogger (without the owner noticing this). Then, if the legitimate owner of the card enters his/her PIN to authenticate to the card reader, this PIN is forwarded to the adversary, which can re-authenticate illicitly if the owner forgets to take the card from the reader.

Clearly, if an adversary possesses the card (which remains close to the reader) and if the PIN is revealed (through keylogging software on the owner's PC), there is no secret information that can prevent impersonation attempts. This attack, however, does not contradict our results since we do not — and indeed cannot — give any security guarantees if all secrets needed to authenticate are disclosed. Nonetheless, the adversary is not able to read sensitive data from the card, as this information is sent encrypted over the channel. We stress here that in order to apply this attack a trojan has to be installed on the PC and thus, if one keeps one's PC protected through a firewall, with an updated virus-scanner, this attack is not a potential threat. Also note that authentication through the eID card and the PIN still provides stronger security than traditional credentials, like username and password, to authenticate to a service provider.

In November 2010, Jan Schejbal described in his blog [Jan10] an attack against the eID application software, called AusweisApp, where he exploits its update routine in order to inject a wrong uncertified AusweisApp [Hei10c]. This does not affect the security of the eID card directly. That is, no immediate abuse of the eID card is possible and also no sensitive data can be retrieved from the card. However, the compromised PC might leak sensible information. To this end, BSI reacted to this post and updated the AusweisApp accordingly, with no subsequent security leak (to the present day). The subsequent phishing attack by Jan Schejbal [Hei11] works similarly to his previous approach, luring the card owner to a webpage masquerading as a certified service provider. The adversary thus learns the PIN, given that the basic card reader is used. Similar arguments as for attack by the CCC apply here. That is, if all secrets of an eID card leak, no secure authentication can be constructed on top of it. This attack, though, works under particular setup assumptions on the user's PC, and can be eliminated by updating the corresponding software.

We infer from previous paragraphs that in order to assess the security of

the eID card not only the protocols itself needs to be secure but also their implementation and its periphery.

**OTHER NATIONAL ELECTRONIC IDENTITY CARDS.** Germany is not the only country moving to identity cards with an embedded electronic chip. In fact, many European countries are going, or have already started, to use national electronic identity cards. This is the case, for instance, in Belgium, Estonia, Finland, Italy, the Netherlands, Portugal, Spain, and Sweden. Also outside Europe, eID cards are expected to be deployed (or are already being used) in, e.g., Brazil, Indonesia, Malaysia, the Philippines, Afghanistan, Bahrain, Oman, Qatar, Saudi Arabia, the UAE, and many more countries.

These electronic cards are different in many ways. First, the information displayed on the cards varies from nation to nation. For instance, in Europe biometric data is contained in only four countries, i.e., Italy, Portugal, Spain, and Germany. Also, each country uses different protocols to ensure secure communication and identification. In this thesis we focus only on the security of the protocols deployed in the German eID card. However, we stress that standardization (which is planned for Europe) can lead to much more applicability and stronger security if nations agree on the choice of protocol implemented on the card. Clearly, without provable security no protocol should be chosen and implemented. To this end, we hope that our result encourages authorities to favor the protocols proposed by BSI, which are partially already standardized for deployment on machine-readable travel documents.

### 1.5. Contributions of this Thesis

The major focus of this thesis is to investigate how secure the cryptographic protocols deployed on the German eID card are. In addition, based on these protocols we propose two novel protocols providing stronger security and improved efficiency. We summarize the major results of this thesis below:

**Security of the Extended Access Control Protocol.** We give a positive security proof of the EAC protocol under a generally accepted number-theoretic assumption (namely Gap Diffie-Hellman) and assuming the security of the underlying building primitives. To this end, we select an appropriate and approved, strong security model for authenticated key

exchange, i.e., the Bellare and Rogaway model [BR94a]. Moreover, we look at further security properties, such as forward secrecy and leakage of internal secrets. The following theorem captures informally our result on the security of EAC:

**Theorem 1 (Security of EAC - informal)** *The Extended Access Control protocol is a secure authenticated key-exchange protocol in the security model of Bellare and Rogaway as long as the underlying certification, message authentication code, and respectively, signature scheme, and the compression function are secure, the hash function behaves ideally, and the Gap Diffie–Hellman assumption holds.*

**Security of the Secure Messaging Protocol.** Key-exchange protocols are subsequently used in the secure channel protocol (constructed in the German eID card by the SM protocol). We first formally define the security model which captures all required and desired security properties. In particular, we formulate a monolithic security definition and assess the security of SM when it is composed with the key-exchange protocols PACE and EAC, as this is the intended procedure for the German eID card. We stress that composing two secure protocols do not yield a secure composition. We discuss this issue further in Section 3.3. Subsequently, we show that Secure Messaging provides all required security properties as a secure channel even if executed after PACE or EAC. Our main result can be stated as:

**Theorem 2 (Security of SM - informal)** *The Secure Messaging protocol is a secure channel protocol when using the keys derived from PACE or EAC, as long as the underlying message authentication code and encryption scheme are secure.*

**Security of the Restricted Identification Protocol.** The optional RI protocol allows a user to identify to a service provider through a pseudonym. For this protocol we first assess its security with respect to the two properties claimed in BSI specification, namely, cross-domain anonymity and domain-specific linkability. To this end, to the best of our knowledge, we give the first concrete security model for cross-domain anonymity in the literature. In this model we prove that RI is cross-domain anonymous, i.e.:

**Theorem 3 (Cross-Domain Anonymity of RI - informal)** *The Restricted Identification protocol is cross-domain anonymous as long as the Decisional Diffie–Hellman assumption holds.*

We also comment on domain-specific linkability and provide positive results in the context of RI security.

**Introduction of the PACE|AA Protocol.** We propose a novel protocol which is essentially a merge of the PACE and Active Authentication (AA) protocols. AA is the alternative to the EAC protocol implemented on electronic passports. We describe how the PACE and AA protocols can be merged in a natural way such that it preserves the security properties of the component protocols. We prove the following result:

**Theorem 4 (Security of PACE|AA - informal)** *The PACE|AA protocol is a secure password-based authenticated key-exchange protocol resistant to impersonation attacks as long as, the specific signature scheme is secure in a robust sense, and the certification and message authentication scheme are secure.*

We also prove that one variant of our proposed protocol is deniable, i.e., a chip card can deny that any interaction with a terminal took place. This is a privacy property not provided by the original Active Authentication protocol.

**Introduction of Domain-Specific Pseudonymous Signatures.** We introduce a novel signature scheme which allows a user to sign arbitrary messages on behalf of pseudonyms. In particular, our signature scheme is built upon Restricted Identification such that it allows easy integration in the given architecture and provides additional security properties. That is, we show that our scheme provides cross-domain anonymity, unforgeability, and seclusiveness. Informally, the two latter properties guarantee that no one can sign messages on behalf of pseudonyms of other honest parties or of (still) unassigned pseudonyms.

**Theorem 5 (Security of our Signature Scheme - informal)** *Our domain-specific pseudonymous signature scheme is cross-domain anonymous, unforgeable, and seclusive as long as the Decisional Diffie–Hellman assumption, and, respectively, the Discrete Logarithm assumption hold. In addition, we assume that the hash function behaves ideally.*

## 1.6. Outline of this Thesis

The structure of this thesis is as follows:

- Chapter 2.** In this chapter we introduce the basic definitions and notations necessary to formally define and analyze cryptographic protocols used in the German electronic identity card. Moreover, we introduce number-theoretic assumptions and cryptographic primitives which serve as building blocks for the more complex protocols and resp. for the proofs we show in the next chapters.
- Chapter 3.** Here, we explain the general structure and strategy employed in the security analysis of a cryptographic protocol. This way of reasoning will often recur and influence our results in the subsequent chapters. We also show one way of modeling the security of authenticated key exchange and secure channel protocols.
- Chapter 4.** In this chapter, we describe the PACE protocol in detail. We also recall all the results on PACE to date. We stress that these security analyses are not part of this thesis' contribution. Nevertheless, these results must form a part of any complete treatment of the other protocols used by the German identity card.
- Chapter 5.** We give a detailed description of the EAC protocol where we subdivide the protocol into two parts, namely Terminal Authentication and Chip Authentication. Afterwards, we analyze the security of EAC as an authenticated key-exchange protocol.
- Chapter 6.** In this chapter we describe the Secure Messaging protocol, and prove its security if it is run with the keys from the key-exchange protocols PACE or EAC.
- Chapter 7.** Here, we describe the Restricted Identification protocol and provide a rigorous security analysis of it. In particular, we show that RI satisfies cross-domain anonymity and offers domain-specific linkability.
- Chapter 8.** We summarize the assumptions on the cryptographic primitives and whether they are conformed according to the specifications in order to assure secure usage of the German eID card for authentication. Then, we comment on further requirements, beyond cryptography, for the security of the German eID card.
- Chapter 9.** This chapter presents the PACE|AA protocol (variants). We discuss the security model for password-based authenticated key exchange and impersonation resistance. After discussing the relevant (number-theoretic and cryptographic) security assumptions, we demonstrate our security results. Finally, we discuss our deniable version and its security.
- Chapter 10.** Here, we propose an extension to the Restricted Identification solution where users are able to sign messages on behalf on their pseudonyms. We identify three essential security properties, namely cross-domain anonymity, unforgeability, and seclusiveness. We show that our

## 1. Introduction

---

proposed domain-specific pseudonymous signature scheme satisfies all these properties, making it a reasonable upgrade to the RI protocol.

## 2. Definitions

In this chapter we introduce the basic definitions and notations necessary to formally define and analyze the cryptographic protocols used in the German electronic identity card. Moreover, we introduce number-theoretic assumptions and cryptographic primitives which serve as building blocks for the more complex protocols we analyze in the next chapters. For a broader and more detailed introduction to cryptography encompassing important primitives, cryptographic assumptions, and their instantiations we refer to the books of Goldreich [Golo1, Golo4], and Katz and Lindell [KL07]. Experienced readers familiar with cryptography may skip most of this chapter.

**READER'S ROADMAP.** In Section 2.1, we begin introducing the general notation which we use throughout this dissertation. In Section 2.2 we formalize number-theoretic assumptions that the protocols in the German eID card rely on. These computational assumptions have been believed to hold — i.e., no computationally efficient algorithm is known to exist that breaks the corresponding problem — for decades by cryptographers and mathematicians. In Section 2.3, we give formal definitions of cryptographic primitives and their security requirements, and describe their purpose when deployed in a more complex construction.

### 2.1. General Notation

Throughout this dissertation, we denote the security parameter by  $\lambda$  (or in unary notation  $1^\lambda$ ). Cryptographic primitives and protocols adjust their specific parameters (e.g, key sizes, choice of a mathematical group, etc.) according to the security parameter. A function is called *negligible* (in the security parameter  $\lambda$ ) if it decreases faster than the inverse of every polynomial in  $\lambda$ , for  $\lambda$  sufficiently large. More formally:

## 2. Definitions

---

**Definition 2.1.1 (Negligible Functions)** A function  $\varepsilon : \mathbb{N} \mapsto \mathbb{R}$  is called negligible, written  $\varepsilon(\lambda) \approx 0$ , if for all polynomials  $p : \mathbb{N} \mapsto \mathbb{R}^+$  there exists an  $N \in \mathbb{N}$  such that for all  $\lambda \geq N$  it holds  $\varepsilon(\lambda) \leq \frac{1}{p(\lambda)}$ .

Consequently, if a function  $\delta(\cdot)$  is non-negligible, then there exists a polynomial  $p(\cdot)$  whose inverse decreases faster than function  $\delta$ . Examples for negligible functions are  $2^{-\lambda}$ ,  $2^{-\sqrt{\lambda}}$ , or  $\lambda^{-\log \lambda}$ . Let  $\varepsilon_1, \varepsilon_2$  be two negligible functions and  $p_1, p_2$  be two polynomial functions. It is easy to verify the validity of the following relations:

- The functions defined as follows are non-negligible:

$$1/(p_1(\lambda) + p_2(\lambda)), \quad 1/(p_1(\lambda) \cdot p_2(\lambda)), \quad \varepsilon_1(\lambda) + 1/p_1(\lambda).$$

- The functions defined as follows are negligible:

$$\varepsilon_1(\lambda) + \varepsilon_2(\lambda), \quad \varepsilon_1(\lambda) \cdot \varepsilon_2(\lambda), \quad \varepsilon_1(\lambda)/p_1(\lambda).$$

For an algorithm  $\mathcal{A}$ , the value  $y \leftarrow \mathcal{A}(x)$  denotes the output of  $\mathcal{A}$  on input  $x$ ; if  $\mathcal{A}$  uses randomness, then  $\mathcal{A}(x)$  is a random variable. Also,  $\mathcal{A}^{\mathcal{O}}$  denotes that  $\mathcal{A}$  has access to oracle  $\mathcal{O}$ . An algorithm  $\mathcal{A}$  is probabilistic polynomial-time (PPT) if  $\mathcal{A}$  is randomized — uses internal random coins — and, for any input  $x \in \{0, 1\}^*$ , the computation of  $\mathcal{A}(x)$  terminates in at most  $\text{poly}(|x|)$  steps.

By  $\mathbb{N}$  and  $\mathbb{Z}$  we denote the set of natural and integer numbers, respectively. For  $n \in \mathbb{N}$ , let  $[n] \stackrel{\text{def}}{=} \{1, 2, \dots, n\}$ . The order of a group  $\mathcal{G}$  is denoted by  $\text{ord}\{\mathcal{G}\}$ . We denote by  $\langle g \rangle$  with  $g \in \mathcal{G}$  the (sub)group generated by  $g$ . All protocols within the eID card use cyclic groups of prime order  $p$ . Indeed, the finite additive cyclic group  $\mathbb{Z}_p \stackrel{\text{def}}{=} \{0, 1, \dots, p-1\}$  has prime order  $p$ , i.e.,  $\text{ord}\{\mathbb{Z}_p\} = p$ . The multiplicative group  $\mathbb{Z}_p^*$  consists of the non-zero elements of  $\mathbb{Z}_p$  (given  $p$  is prime). Furthermore, if the protocols run on elliptic curves, then the parameters  $\mathcal{G} = (a, b, p, q, g, \lambda)$  give the description of an elliptic curve  $y^2 = x^3 + ax + b \pmod{p}$  where  $\langle g \rangle$  is a group of prime order  $q$ .

If  $\mathcal{D}$  is a probability distribution or random variable, we denote by  $d \leftarrow \mathcal{D}$  the process of sampling a value  $d$  randomly according to  $\mathcal{D}$ . In case  $S$  is a set, then  $s \leftarrow_R S$  means that the value  $s$  is sampled according to a uniformly random distribution over the set  $S$ .

Let  $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$  and  $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$  be two distribution ensembles. We say  $X$  and  $Y$  are  $\varepsilon$ -computationally indistinguishable if for every polynomial-time

distinguisher  $\mathcal{A}$  there exists a function  $\varepsilon$  such that:

$$|\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]| \leq \varepsilon(\lambda).$$

If  $\varepsilon(\lambda)$  is negligible, we simply say  $X$  and  $Y$  are (computationally) indistinguishable (and we write  $X \approx Y$ ).

## 2.2. Number-Theoretic Assumptions

The security of cryptographic protocols is often based on number-theoretic assumptions, i.e., the hardness of computational problems.

Constructions secure against even unbounded adversaries (thus not relying on computationally hard problems) guarantee information-theoretic security. Even if such constructions are preferable over computationally-secure protocols from a security point of view, they often require significantly more memory or they lack efficiency, such that usually they cannot be implemented on a resource-restricted platform, such as the chip card within electronic identity cards. Moreover, some cryptographic primitives or protocols cannot be built to be information-theoretically secure. An direct example is information-theoretically secure key agreement when parties do not share a secret and communicate over a public channel, which is, interceptable by anyone [Mau93]. For these reasons, the protocols presented and analyzed in this thesis, i.e., the protocols deployed on the electronic identity card, are based on problems that are presumably hard-to-solve by an efficient bounded adversary.

All the protocols involved in the German electronic identity card base their security on the hardness of either the discrete-logarithm (DL) problem or on variants of the Diffie–Hellman (DH) problem [DH76]. These assumptions have been the subject of study for many decades such that nowadays we can very well assess their hardness. Consequently, we are able to provide the exact, concrete security of cryptosystems based on these problems.

We start by recalling the DL problem, whose hardness is required for proving the security of the Restricted Identification protocol (cf. Chapter 7), and its extension (cf. Chapter 10). We also note that, throughout the thesis, we use the multiplicative notation for group operations. It is understood that, if working with elliptic curves, multiplications correspond to additions and exponentiations to multiplications. Moreover, we consider a randomized algorithm  $\mathcal{I}$ , called an instance generator, which upon input a security parameter  $\lambda$  (in

## 2. Definitions

---

unary), runs in polynomial time in  $\lambda$  and outputs a (cyclic) group  $\mathcal{G}$  of order  $p_\lambda$  with group generator  $g$ .

**Definition 2.2.1 (DL Hardness)** *The Discrete Logarithm problem (relative to an instance generator  $\mathcal{I}$ ) is  $(t, \varepsilon)$ -hard if any algorithm  $\mathcal{A}$ , running in time  $t$ , makes the following experiment output 1 with probability at most  $\varepsilon$ :*

$(\mathcal{G}, g) \leftarrow \mathcal{I}(\lambda)$   
pick  $a \leftarrow \mathbb{Z}_p$   
let  $a' \leftarrow \mathcal{A}(\mathcal{G}, g, g^a)$   
output 1 iff  $a = a'$

We let  $\text{Adv}^{\text{DL}}(t)$  denote (a bound on) the value  $\varepsilon$  for which the DL problem is  $(t, \varepsilon)$ -hard.

In 1976, Diffie and Hellman [DH76] proposed a novel protocol where parties derive a shared key while no passive adversary, merely eavesdropping the network, learns any information about this key. Its security is based on a new number-theoretic assumption, named after the authors.

We denote by  $\text{DH}(g^a, g^b)$  the *Diffie–Hellman value* of  $g^a$  and  $g^b$ , i.e.,  $\text{DH}(g^a, g^b) \stackrel{\text{def}}{=} g^{ab}$ . We sometimes write  $\text{DH}(g^a, b)$  to emphasize that a party contributes with its secret  $b$  to compute the corresponding Diffie–Hellman value  $\text{DH}(g^a, g^b)$ .

The computational Diffie–Hellman (CDH) assumption is formally defined as follows.

**Definition 2.2.2 (CDH Hardness)** *The Computational Diffie–Hellman (CDH) problem (relative to an instance generator  $\mathcal{I}$ ) is  $(t, \varepsilon)$ -hard if any algorithm  $\mathcal{A}$ , running in time  $t$ , makes the following experiment output 1 with probability at most  $\varepsilon$ :*

$(\mathcal{G}, g) \leftarrow \mathcal{I}(\lambda)$   
pick  $a, b \leftarrow \mathbb{Z}_p$   
let  $Z \leftarrow \mathcal{A}(\mathcal{G}, g, g^a, g^b)$   
output 1 iff  $Z = \text{DH}(g^a, g^b) = g^{ab}$

We let  $\text{Adv}^{\text{CDH}}(t)$  denote (a bound on) the value  $\varepsilon$  for which the CDH problem is  $(t, \varepsilon)$ -hard.

The computational Diffie–Hellman assumption states that no algorithm can output the DH value relative to a group generator and to public shares. The decisional variant of this assumption, called Decisional Diffie–Hellman (DDH) [Bra93, Bon98], says that no efficient algorithm is able to even distinguish a genuine DH value from a randomly chosen group element, given the same inputs. The following definition captures the DDH assumption formally.

**Definition 2.2.3 (DDH Hardness)** *The Decisional Diffie–Hellman (DDH) problem (relative to an instance generator  $\mathcal{I}$ ) is  $(t, \epsilon)$ -hard if any algorithm  $\mathcal{A}$  running in time  $t$ , makes the following experiment output 1 with probability at most  $\epsilon$ :*

$$\begin{aligned} &(\mathcal{G}, g) \leftarrow \mathcal{I}(\lambda) \\ &\text{pick } a, b, c \leftarrow \mathbb{Z}_p \text{ and } d \leftarrow_{\mathcal{R}} \{0, 1\} \\ &\text{let } d' \leftarrow \mathcal{A}(\mathcal{G}, g, g^a, g^b, g^{dab+(1-d)c}) \\ &\text{output 1 iff } d = d' \end{aligned}$$

We let  $\text{Adv}^{\text{DDH}}(t)$  denote (a bound on) the value  $\epsilon$  for which the DDH problem is  $(t, Q, \epsilon)$ -hard.

To show security against active adversaries, possibly contributing to the computation of DH values, we need a stronger assumption than CDH, denoted Gap Diffie–Hellman (GDH) problem [OP01, BLS01]. This assumption basically states that solving the computational DH problem (outputting the DH value of  $g^a, g^b$ , given these shares) is still hard, even if an adversary can test whether arbitrary triples  $(g^x, g^y, g^z)$  form a DH triple. This is formally done by giving the adversary access to a decisional DH-oracle  $\text{DDH}(X, Y, Z)$  which returns 1 iff  $\text{DH}(X, Y) = Z$ , and 0 otherwise.

**Definition 2.2.4 (GDH Hardness)** *The Gap Diffie–Hellman (GDH) problem (relative to an instance generator  $\mathcal{I}$ ) is  $(t, Q, \epsilon)$ -hard if any algorithm  $\mathcal{A}$ , running in time  $t$  and making at most  $Q$  oracle queries, makes the following experiment output 1 with probability at most  $\epsilon$ :*

$$\begin{array}{l|l} (\mathcal{G}, g) \leftarrow \mathcal{I}(\lambda) & \\ \text{pick } a, b \leftarrow \mathbb{Z}_p & \\ \text{let } Z^* \leftarrow \mathcal{A}^{\text{DDH}(\cdot, \cdot, \cdot)}(\mathcal{G}, g, g^a, g^b) & \\ \text{output 1 iff } Z^* = \text{DH}(g^a, g^b) = g^{ab} & \end{array} \quad \left| \quad \begin{array}{l} \text{If } \mathcal{A} \text{ queries } \text{DDH}(X, Y, Z), \\ \text{return 1 iff } \text{DH}(X, Y) = Z. \end{array} \right.$$

## 2. Definitions

---

We let  $\text{Adv}^{\text{GDH}}(t, Q)$  denote (a bound on) the value  $\varepsilon$  for which the GDH problem is  $(t, Q, \varepsilon)$ -hard.

For the security proof of the protocols, we actually need a slightly weaker requirement where the adversary can only query the DDH oracle on inputs of the form  $(g^a, B, C)$  or  $(A, g^b, C)$ , i.e., where one of the values is given by parts of the input.

### 2.3. Basic Cryptographic Primitives

In this section we introduce the cryptographic primitives used by the protocols in the German eID card. For sake of clarity, we sometimes omit saying explicitly that messages are elements from the message spaces (analogously, ciphertexts, keys, etc.) and simply assume it implicitly.

**DIGITAL SIGNATURES.** A (digital) signature allows a party to provide a publicly-verifiable proof that a certain message is approved by it. It is the digital analogue of a hand written signature. Roughly, anyone who can produce signatures on behalf of a party must know its corresponding secret material. This provides a means of guaranteeing data authenticity. Digital signatures are important tools in key-exchange protocols, where parties can verify that their partner in one session is a legitimate one. The following definition captures digital signatures formally.

**Definition 2.3.1 (Signature Scheme)** A (digital) signature scheme consists of three PPT algorithms (SKGen, Sig, SVf) defined as follows.

**Key Generation.** Upon input the security parameter  $1^\lambda$ , the probabilistic algorithm SKGen outputs a key pair  $(\text{sk}, \text{pk})$  where sk (resp. pk) denotes the signing (resp. public verification) key.

**Signature.** Upon input a signing key sk and a message  $m$ , the probabilistic algorithm Sig outputs a signature  $\sigma$ .

**Verification.** Upon input the verification key pk, a message  $m$ , and a signature  $\sigma$ , the deterministic algorithm SVf outputs either 1 (= valid) or 0 (= invalid).

We require correctness of the verification, i.e., the verifier must always accept genuine signatures. More formally, for any security parameter  $\lambda$ , any

$(sk, pk) \leftarrow \text{SKGen}(1^\lambda)$ , for any message  $m$ , any signature  $\sigma \leftarrow \text{Sig}(sk, m)$ , we must have  $\text{SVf}(pk, m, \sigma) = 1$ .

From a signature scheme, we require that no outsider is able to forge a signer's signature. Formally, this property is called unforgeability against adaptively-chosen-message attacks (unf-cma) and is defined as follows.

**Definition 2.3.2 (UNF-CMA Security.)** A (digital) signature scheme  $\mathcal{S} = (\text{SKGen}, \text{Sig}, \text{SVf})$  is  $(t, Q, \epsilon)$ -unforgeable against adaptively-chosen-message attacks if for any algorithm  $\mathcal{A}$  with runtime  $t$  and making at most  $Q$  queries to its signing oracle  $\mathcal{S}$ , the probability that the following experiment returns 1 is at most  $\epsilon$ .

$(sk, pk) \leftarrow_R \text{SKGen}(\lambda)$ $(m^*, \sigma^*) \leftarrow_R \mathcal{A}^{\mathcal{S}(\cdot)}(pk)$ Return 1 iff $\text{SVf}(pk, m^*, \sigma^*) = 1$ and $m^* \notin M$ .	Set $M = \emptyset$ . If $\mathcal{A}$ queries $\mathcal{S}(m)$ , add $m$ to $M$ , and return $\text{Sig}(sk, m)$ .
--	--

The probability is taken over all coin tosses of  $\text{SKGen}$ ,  $\text{Sig}$ , and  $\mathcal{A}$ .

We let  $\text{Adv}_{\mathcal{S}}^{\text{unf-cma}}(t, Q)$  denote (a bound on) the value  $\epsilon$  for which the scheme  $\mathcal{S}$  is  $(t, Q, \epsilon)$ -unforgeable.

**CERTIFICATION SCHEMES.** The protocols within the German eID card make use of a Public-Key Infrastructure (PKI). A certification authority ( $\mathcal{CA}$ ) assigns all users — chip cards and terminals — secret and public key material and provides them with a certificate on their identity and on the public key. The certificate binds a public key to a corresponding identity. If a certificate verifies, one is assured that the underlying public key has been generated by the trusted  $\mathcal{CA}$  only.

Formally, we consider a certification scheme as a signature scheme where the signer is a certification authority ( $\mathcal{CA}$ ). We denote such a scheme as a tuple  $\mathcal{CA} = (\text{CKGen}, \text{Certify}, \text{CVf})$ . The key generation algorithm, executed by a  $\mathcal{CA}$ , outputs a pair of keys  $(sk_{\mathcal{CA}}, pk_{\mathcal{CA}}) \leftarrow \text{CKGen}(\lambda)$ ; given the identity information relative to user  $\mathcal{U}$  (to specify: for sure, a static public key  $pk_{\mathcal{U}}$ ), the  $\mathcal{CA}$  embeds  $\mathcal{U}$ 's credentials into a certificate  $cert_{\mathcal{U}} \leftarrow \text{Certify}(sk_{\mathcal{CA}}, pk_{\mathcal{U}}, \dots)$ . Anyone can validate the certificate  $cert_{\mathcal{U}}$  by checking whether the algorithm  $\text{CVf}(pk_{\mathcal{CA}}, cert_{\mathcal{U}})$  outputs 1.

## 2. Definitions

---

Similarly to digital signatures, we denote by  $\text{Adv}_{\mathcal{C}, \mathcal{A}}^{\text{unf-cma}}(t, Q)$  a (bound on the) value  $\varepsilon$  for which no attacker in time  $t$  can forge a certificate (while making at most  $Q$  certifying queries).

**ENCRYPTION SCHEME.** Encryption schemes allow a user to send messages to a recipient over an insecure communication channel while preserving the confidentiality of the message. Roughly speaking, an encryption of a message does not reveal any information about the message and only the person who knows the corresponding secret key can recover the encrypted plaintext. The protocols involved in the German eID card use symmetric encryption schemes, where both the encryption of messages and the decryption of ciphertexts require the secret key. The following definition captures symmetric encryption schemes formally.

**Definition 2.3.3 (Symmetric Encryption Scheme)** *A symmetric encryption scheme consists of three polynomial-time algorithms  $\mathcal{E} = (\text{KGen}, \text{Enc}, \text{Dec})$  defined as follows.*

**Key Generation.** *Upon input the security parameter  $1^\lambda$ , the probabilistic algorithm  $\text{KGen}$  outputs a secret key  $\text{sk}$ .*

**Encryption.** *Upon input the secret key  $\text{sk}$  and a message  $m$ , the probabilistic algorithm  $\text{Enc}$  outputs a ciphertext  $c$ .*

**Decryption.** *Upon input the secret key  $\text{sk}$  and a ciphertext  $c$ , the deterministic algorithm  $\text{Dec}$  outputs a message  $m$ .*

An encryption scheme should be complete, i.e., for any security parameter  $\lambda$ , any  $\text{sk} \leftarrow \text{KGen}(1^\lambda)$ , for any message  $m$ , and any  $c \leftarrow \text{Enc}(\text{sk}, m)$  we have  $\text{Dec}(\text{sk}, c) = m$ .

The security of the protocols in the German eID card rely on the standard security of the underlying encryption primitive, namely the indistinguishability (of encryptions) under chosen-plaintext attacks (ind-cpa) [GM84].

**Definition 2.3.4 (IND-CPA Security.)** *A symmetric encryption scheme  $\mathcal{E} = (\text{KGen}, \text{Enc}, \text{Dec})$  is  $(t, Q, \varepsilon)$ -indistinguishable under chosen-plaintext attacks (or simply  $(t, Q, \varepsilon)$ -ind-cpa secure) if for any algorithm  $\mathcal{A}$  with runtime  $t$  and making at most  $Q$  queries to its encryption oracle  $\text{E}$ , the advantage  $\text{Adv}_{\mathcal{E}}^{\text{ind-cpa}}(t, Q)$  is at most  $\varepsilon$  where*

$$\text{Adv}_{\mathcal{E}}^{\text{ind-cpa}}(\mathcal{A}) = \Pr \left[ \text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind-cpa-1}}(\lambda) \right] - \Pr \left[ \text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind-cpa-0}}(\lambda) \right].$$

The experiments  $\text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind-cpa-}b}(\lambda)$  for  $b \in \{0, 1\}$  are defined as follows.

<p><b>Experiment</b> <math>\text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind-cpa-}b}(\lambda)</math></p> <p><math>\text{sk} \leftarrow_R \text{KGen}(\lambda)</math></p> <p><math>d \leftarrow \mathcal{A}^{\text{E}(\cdot)}</math></p> <p>Output <math>d</math>.</p>	<p>If <math>\mathcal{A}</math> queries <math>\text{E}(m_0, m_1)</math>,</p> <p>return <math>\text{Enc}(\text{sk}, m_b)</math>;</p>
--	--

The probability is taken over all coin tosses of  $\text{KGen}$ ,  $\text{Enc}$ , and  $\mathcal{A}$ .

Many symmetric encryption schemes take as a building block *blockciphers* — or simply ciphers — which are a family of permutations. Roughly speaking, a blockcipher  $\mathcal{C}$  consists of two PPT algorithms  $(\text{Enc}, \text{Dec})$  where both algorithms take as input a key of size  $k$  and an input of block size  $n$  and output an  $n$ -bit long bitstring. Correctness requires that for all keys  $\kappa \in \{0, 1\}^k$  and all strings  $m \in \{0, 1\}^n$  we have  $\text{Dec}(\kappa, \text{Enc}(\kappa, m)) = m$ . A blockcipher is essentially a symmetric encryption scheme for a fixed “maximum” message length. If needed, one could pad an  $n$ -string to a message to satisfy the length condition.

In order to allow easier proofs, ciphers are sometimes modeled as an ideal cipher. In the ideal-cipher model, ciphers act like a random permutation, and hence, are hard to invert, and output a uniformly distributed ciphertext from the ciphertext space. The ideal-cipher model is commonly used and dates back to Shannon [Sha49]. Also, the protocols PACE and PACE|AA are proven secure in the ideal-cipher model.

**MESSAGE AUTHENTICATION CODES.** As is the case for digital signature schemes, message authentication codes (MAC) provide data authenticity; however, unlike signature schemes, the verification of a tag — the analog of a signature for MAC — requires the secret key. That is, message authentication codes are signature schemes with private verification. The following definition captures message authentication codes formally.

**Definition 2.3.5 (Message Authentication Codes)** *A message authentication code consists of three efficient algorithms  $(\text{MKGen}, \text{MAC}, \text{MVf})$  defined as follows.*

**Key Generation.** *Upon input the security parameter  $1^\lambda$ , the probabilistic algorithm  $\text{MKGen}$  outputs a secret (tagging) key  $\text{sk}$ .*

## 2. Definitions

**Tagging.** Upon input a secret key  $sk$  and a message  $m$ , the probabilistic algorithm  $MAC$  outputs a tag  $T$ .

**Verification.** Upon input the secret key  $sk$ , a message  $m$ , and a tag  $T$ , the deterministic algorithm  $MVf$  outputs either 1 (= valid) or 0 (= invalid).

The completeness of the message authentication code demands that for any security parameter  $\lambda$ , any key  $sk \leftarrow_R \text{MKGen}(1^\lambda)$ , and any message  $m$ , the value  $T \leftarrow MAC(sk, m)$  makes  $MVf(sk, m, T)$  return 1.

We require that the message authentication code  $\mathcal{M}$  is unforgeable under adaptively chosen-message attacks. That is, the adversary is granted oracle access to  $MAC(sk, \cdot)$  and  $MVf(sk, \cdot, \cdot)$  for random key  $sk$  and wins if it can eventually output a tag  $T^*$  on a message  $m^*$  which has not been sent previously to  $MAC$ , and for which  $MVf$  returns 1.

The following definition captures this property formally.

**Definition 2.3.6 (UNF-CMA Security.)** A message authentication code  $\mathcal{M} = (\text{MKGen}, \text{MAC}, \text{MVf})$  is  $(t, q_m, q_v, \epsilon)$ -unforgeable under adaptively chosen-message attacks if for any algorithm  $\mathcal{A}$  with runtime  $t$  and making at most  $q_m$  (resp.  $q_v$ ) queries to its tagging (resp. verifying) oracle, the probability that the following experiment returns 1 is at most  $\epsilon$ .

$sk \leftarrow_R \text{MKGen}(\lambda)$ $(m^*, T^*) \leftarrow_R \mathcal{A}^{\text{MAC}(sk, \cdot), \text{MVf}(sk, \cdot, \cdot)}(\lambda)$ <p style="margin: 0;">Return 1 iff <math>MVf(sk, m^*, T^*) = 1</math> and <math>m^* \notin M</math>.</p>	<p style="margin: 0;">Set <math>M = \emptyset</math>.</p> <p style="margin: 0;">If <math>\mathcal{A}</math> queries <math>MAC(sk, m)</math>, add <math>m</math> to <math>M</math>, and return <math>MAC(sk, m)</math>.</p> <p style="margin: 0;">If <math>\mathcal{A}</math> queries <math>MVf(sk, m, T)</math>, return <math>MVf(sk, m, T)</math>.</p>
---	---

The probability is taken over all coin tosses of  $\text{MKGen}$ ,  $\text{MAC}$ , and  $\mathcal{A}$ .

We let  $\text{Adv}_{\mathcal{M}}^{\text{unf-cma}}(t, q_m, q_v)$  denote (a bound on) the value  $\epsilon$  for which the scheme  $\mathcal{M}$  is  $(t, q_m, q_v, \epsilon)$ -unforgeable.

We note that some security models for message authentication codes (e.g., [KL07]) do not allow the adversary to query a verification oracle. However, we stick to our model allowing the distinction of “impersonation attacks” ( $q_m = 0$  and  $q_v = 1$ ), “substitution attacks” ( $q_m = 1$  and  $q_v = 1$ ), and even attacks featuring large  $q_m$  and  $q_v$  values, as described in lecture notes on “Modern Cryptography” by Bellare [Bel12].

**KEY DERIVATION AND HASH FUNCTIONS.** The key-exchange protocols within the German eID card invoke a key-derivation function enabling both parties to derive a uniformly-random distributed key on input a DH value which is only known to the participants of the protocol instance. The same property is desirable for cryptographic hash functions. Informally, hash functions are functions that map a string of arbitrary length into a string of fixed length, and, in addition, terminate in polynomial time in the input length. While key-derivation functions should provide a uniformly distributed output, hash functions must provide mainly (second-) preimage resistance and collision-resistance. In practice, key derivation functions are implemented by hash functions. For that reason, we only introduce here the definition of hash functions and sketch the security requirements. We assume throughout the thesis that the hash functions have predefined fixed domain and image space depending on the security parameter.

**Definition 2.3.7** A function  $h : \{0,1\}^m \mapsto \{0,1\}^n$  is called a hash function if the following conditions are fulfilled:

1.  $h$  is efficiently computable (i.e., in PPT), and
2.  $h$  is compressing, i.e.,  $m > n$ .

The following security properties are requested by a hash function.

**One-Wayness (Pre-Image Resistance).** For a value  $y$  uniformly sampled from  $\{0,1\}^n$ , it should be computationally hard to find a pre-image  $x \in \{0,1\}^m$  such that  $h(x) = y$ .

**Second Pre-Image Resistance.** For a value  $x$  uniformly sampled from  $\{0,1\}^m$ , it should be computationally hard to find a second pre-image  $x' \in \{0,1\}^m$  such that  $h(x) = h(x')$ .

**Collision Resistance.** It should be computationally hard to find two values  $x, x' \in \{0,1\}^m$  such that  $h(x) = h(x')$  and  $x \neq x'$ .<sup>1</sup>

Similarly to the ideal-cipher model, to facilitate security proofs, hash functions and key-derivation functions are sometimes modeled by ideal functions. In the random-oracle model (ROM) [BR93] one assumes the existence of a publicly-available function  $H$  that acts like a truly random function.

The random oracle is widely recognized as relevant for security proofs of complex cryptographic protocols and, in particular, extensively used for

---

<sup>1</sup>Here, we implicitly assume that the hash function is drawn uniformly from a family of hash function because otherwise a collision can be hardcoded in the machine of an adversary.

practical protocols, e.g., RSA-FDH [BR93, BR96], RSA-OAEP [BR94b, BF06], RSA-PSS [BR96], TLS [MSW08, GMP<sup>+</sup>08, BFS<sup>+</sup>13], and SSH [Wil11]. Through simplified proofs, the ROM allows one to construct more efficient protocols and primitives; as opposed to protocols which are provably secure in the standard model (i.e., no idealized function/primitive). However, there exists cryptographic schemes which are provably secure in the ROM, but insecure once the idealized function is instantiated with *any* hash function [CGH98], these constructions are rather artificial and security in the ROM shows that in order to break the protocol, one must exploit some weaknesses of the hash function [KM07]. The relation between the random-oracle model and the ideal-cipher model is studied in [CPS08, HKT11].

In this thesis, we analyze the security of the protocols in the random-oracle model.

**COMPRESSION FUNCTIONS.** The Extended Access Control protocol makes use of a compression function  $\text{Compr}$ . For a compression function  $\text{Compr}$  we assume either that this function is injective (errorless compression), or at least second-preimage resistant. In any case, it should be hard to find another preimage to a given random image mapping to the same value. For instance, this property is fulfilled by injective, as well as collision-resistant functions, even though second-preimage resistance imposes a weaker condition on the function than collision-resistance. We discuss in Section 5.3 that even this requirement can be weakened even further, provided that collisions are “appropriately intertwined” with the Diffie–Hellman problem. For example, in the case of elliptic curves, a projection of the public key to the x-coordinate, suggested in [BS10], remains secure as well.

**Definition 2.3.8 (Second-Preimage Resistance)** *For a function  $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$  let  $\text{Adv}_{\mathcal{H},\mathcal{D}}^{\text{SecPre}}(\mathcal{A})$  denote the probability that algorithm  $\mathcal{A}$ , given input  $m \leftarrow \mathcal{D}$  (drawn according to distribution  $\mathcal{D}$ ), outputs  $m'$  such that  $m \neq m'$  and  $\mathcal{H}(m) = \mathcal{H}(m')$ .*

We usually demand that  $\text{Adv}_{\mathcal{H},\mathcal{D}}^{\text{SecPre}}(\mathcal{A})$  is negligible, and then call the function second-preimage resistant.

## 3. Security Model

In this chapter, we present how to appropriately model the security of authenticated key exchange (AKE) and secure channel (SC) protocols. Essentially, an AKE protocol is secure if agreed session keys are private (i.e., only known to the intended partner) and not reproducible by different parties even if an adversary interacts actively, e.g., by tampering exchanges messages as well as by eavesdropping. The first property provides confidentiality of agreed session keys, while the latter ensures key authenticity. On the other hand, an SC protocol should provide the confidentiality and integrity of the exchanged messages in the channel.

**READER'S ROADMAP.** In Section 3.1 we explain the general procedure of analyzing the security of cryptographic protocols. The understanding of this section is essential for the rest of the thesis. In Section 3.2 we introduce two approved security models for authenticated key agreement, namely the model by Bellare and Rogaway (BR) [BR94a] and extended Canetti-Krawczyk (eCK) model [LLM07], under which EAC is analyzed. For password-based key-exchange protocols we recall the Bellare-Pointcheval-Rogaway (BPR) model [BPR00]. Then, in Section 3.3, we present our security model for secure channels. Finally, in Section 3.4, we discuss the security properties implied by a security proof in the respective security model.

### 3.1. Proving Security in Cryptography

The security of proposed cryptographic protocols must be analyzed before the protocols are deployed in practice. This methodology is called alias provable security. Attaining the goal of provable security for cryptosystems involves providing a mathematical guarantee that the protocol is secure. As is common in mathematics, this is accomplished under some assumptions. We attain this by means of the following points:

### 3. Security Model

---

- a precise description of the protocol
- a precise description of the class of adversaries
- a precise description of the security model
- a precise description of the success condition
- a security proof that no adversary in the specified class succeeds to break this protocol in the security model under the predefined success condition.

There are various ways of proving security. In this thesis, we investigate the computational security of protocols, i.e., we show that no adversary succeeds in attacking the security protocol — here, AKE and SC — with “reasonable” advantage in “reasonable” time. A common approach is to base security proofs on theoretical experiments, called “games”. We gradually reduce the adversary’s attack strategies against the protocol showing that if the adversary succeeds by using such strategies, it can break the security of underlying assumptions. This process is repeated until the adversary can only perform trivial attacks. Note that it is essential to prove that the probability for each excluded strategy to succeed is negligible; otherwise, we would exclude realistic (and practical) adversaries.

Typically, we model an adversary through an algorithm — formally by a Turing machine — which is probabilistic and terminates in polynomial time (in a given security parameter). Security models provide an abstract interpretation of the actual implementation of protocol, and should be, therefore, adequately formulated. This issue already applies when defining the notion of security.

Before stating that a protocol is secure, one needs first to define precisely the sought-after security goals. A model must then encompass all these security properties. Security models describe how an adversary can interact with the honest participants of the cryptographic protocol and, potentially, grant the adversary additional power in order to capture stronger (even non-existent) adversaries. In the early years of cryptography, one tailored a (unique) security model for each cryptosystem to assess its security. However, it is much more practical (and less fault-prone) to have a single generic security model, which can be reused for many cryptosystems and, which is recognized as accurate. For instance, [BR94a] or [BPR00], are models of these kind, which provide an established security notion for (password-based) key-exchange protocols (cf. Section 3.2). However, it might be still necessary to modify this generic model to some extent, since due to its broad application spectrum, special instantiations of protocols might not be captured in the model.

The security of a cryptosystem is often proven by means of *reductions*. Here it is shown that if a successful adversary against the cryptosystem exists, then one can use this adversary to solve a different (often number-theoretic) problem, which is assumed to be hard to solve. This hardness assumption is justified after people have investigated the problem for many years. Examples are the Diffie–Hellman or the discrete logarithm assumptions (cf. Section 2.2). Since we assume that these problems cannot be solved efficiently (neither today, nor for many years), we can exclude the existence of such adversaries by reasoning through mathematical logic.

## 3.2. Security Model for Authenticated Key Agreement

Bellare and Rogaway [BR94a] were the first to provide a profound model to analyze AKE protocols (this is the BR model). Security according to their notion provides strong guarantees, ensuring that the derived keys remain secure even in presence of active adversaries and multiple concurrent executions. We analyze the Extended Access Control protocol implemented in the German eID card (cf. Chapter 5) in the BR model.

Though the BR model provides strong security guarantees, alternative models subsequently proposed to ensure even further desirable security properties. The most prominent alternatives stem from Canetti and Krawczyk [CK01], mainly augmenting the BR model by modeling leakage of session states during the execution (CK model), and from LaMacchia *et al.* [LLM07] extending the CK model (eCK model) to also include forward secrecy and key-compromise impersonation resilience (KCI). The former property guarantees that session keys are still protected even if the adversary later learns long-term secrets like a signature key. KCI, on the other hand, ensures that leaking the long-term secret does not cause the party to spuriously believe its talking to a different party. Although seemingly stronger, all these models do not form a strict hierarchy, due to technical details [CBH05, BCNP08, Cre09b, Cre09a].

In cryptography, security models are divided into two classes depending on the approach and methodology they use. All aforementioned security models follow the *game-based* approach. Here, one considers an adversary and its “game”, which the adversary aims to win (in order to break the corresponding security property). For key agreement one declares an adversary successful if it can distinguish (with significant advantage) a session key derived by an honest participant from a truly random key, drawn from the key space. The

### 3. Security Model

---

game-based approach is usually to divide the intuitive properties of a scheme into separate games, and thus, proofs are often easier to follow. For example, in authenticated key-exchange protocols one can capture the properties key secrecy and impersonation resistance by two separate security games.

In contrast, *simulation-based* security models capture many security properties at once and work as follows. First, the protocol to be analyzed is modeled by an ideal functionality which by definition satisfies all security requirements required for this protocol. Then, in order to prove security, one shows that any successful real adversary against the protocol can be translated into a likewise successful ideal adversary (also called simulator) against the ideal functionality. This implies that the only information an adversary has from the real protocol is the permitted leakage modeled in the ideal world; in other words, the real protocol behaves (almost) ideally. Prominent examples for simulation-based security models are the “Universally Composability” (UC) framework by Canetti [Can01], the model for reactive systems by Backes, Pfitzmann and Waidner [BPW04], Shoup’s security model, purpose-built for key exchange [Sho99], and the abstract cryptography framework by Maurer and Renner [MR11].

Unfortunately, the above mentioned models do not cover AKE protocols which are password-based, such as PACE or our novel proposal PACE|AA. For this reason, Bellare, Pointcheval, and Rogaway [BPR00] propose a variant of the BR security model particularly for password-based key-exchange protocols. Later, Abdalla, Fouque, and Pointcheval [AFP05] extended the model of Bellare *et al.* [BPR00], proposing a real-or-random security model (cf. Section 3.2.1). The BR model [BR94a] and BPR model [BPR00] are closely related, though. Hence, we cover both models in a single description in the following subsection. Nevertheless, we do explicitly mention the differences between them.

We also mention that Canetti *et al.* [CHK<sup>+</sup>05] (and subsequently [ACCP08]) do give stronger simulation-based formalizations for password-based key exchange in the UC framework. Only a few password-based AKE protocols, however, could be proven secure in that model [CHK<sup>+</sup>05, GMR06, ACCP08, GK10, KV11, DF12].

#### 3.2.1. The BR and BPR Security Model

**ATTACK MODEL.** Consider a set of honest participants, also called users. Each participant may run several instances of the key agreement protocol, and

the  $j$ -th instance of a user  $U$  is denoted by  $U_j$  or  $(U, j)$ . Each user holds a long-term key pair  $(sk, pk)$  and we assume that the public key is registered with a certification authority (e.g., some approved organization for identity cards). The certification authority checks the well-formedness of the keys, e.g., that they belong to an approved group. To obtain a session key, the protocol  $P$  is executed between two instances of the corresponding users.

Upon successful termination we assume that an instance  $U_j$  outputs a session key  $\mathcal{K}$ , the session ID  $sid$ , and a user ID  $pid$  identifying the intended partner. In the case of the EAC protocol we will assume that the partner identity is determined through the certificates exchanged during the protocol. By contrast, in the case of PACE the partner identity is set to  $\emptyset$  for anonymity reasons. We note that the session ID usually contains the entire transcript of the communication but, for efficiency reasons, in the PACE, PACE|AA and EAC protocol it only contains a fraction thereof. We discuss the implications of our approach in more detail in corresponding sections of the protocols.

We consider security against active attacks where the adversary's goal is to distinguish between genuine keys, derived in executions between honest parties, and random keys. This is formalized by allowing a (single) test query in which the adversary either sees the genuine key of the session, or a randomly and independently chosen key (real-or-random). In the BR model, it suffices to consider only a single test query, since the case for multiple test queries for many sessions follows by a hybrid argument [AFP05], decreasing the adversary's advantage by a factor equal to the number of test queries. In contrast, in the BPR model one must allow many test queries explicitly.

The adversary can access user instances through an oracle, basically providing the interface of the protocol instance. By assumption, the adversary is in full control of the network, i.e., it decides upon message delivery. In the BR model, initially, the adversary is given all (registered) public keys of the users. These users are called *honest* whereas the other users, for which the adversary subsequently registers chosen public keys, are called *adversary-controlled*.<sup>1</sup>

In the password-based BPR model, the honest users are initially given their secret password (chosen uniformly from a dictionary of size  $N$ ). Before each interaction, this password is transmitted privately to the partner without leaking any information about it to the adversary. This captures entering the PIN in the case of PACE.

---

<sup>1</sup>We remark that the adversary may register public keys already chosen by honest parties on behalf of adversary-controlled users.

The adversary can make the following queries to its oracles:

$\text{Execute}(U, i, U', j)$  causes the honest users  $U$  and  $U'$  to run the protocol for (fresh) instances  $i$  and  $j$ . The final output is the transcript of a protocol execution. This query simulates a passive attack where the adversary merely eavesdrops on the network.

$\text{Send}(U, i, m)$  causes the instance  $i$  of honest user  $U$  to proceed with the protocol when receiving message  $m$ . The output is the message generated by  $U$  on input  $m$  and depends on the state of the instance. This query simulates an active attack where the adversary pretends to be the partner instance.

$\text{Reveal}(U, i)$  returns the session key of the input instance. The query is answered only if the session key was generated, the instance has terminated in accepting state, and the user is not controlled by the adversary. This query models the case when the session key has been leaked. We assume without loss of generality that the adversary never queries about the same instance twice.

$\text{Corrupt}(U)$  enables the adversary to obtain the party's long-term key  $sk$ . This happens in the so-called *weak-corruption* model. In the *strong-corruption* model the adversary also obtains the state information of all instances of user  $U$ . The corrupt queries model a total break of the user and allow us to model forward secrecy. Henceforward, user  $U$  is considered to be adversary-controlled.

$\text{Test}(U, i)$  is initialized with a random bit  $b$ . Assume the adversary makes a test query about  $(U, i)$  during the attack and that the instance has terminated in accepting state, having computed a secret session key  $\mathcal{K}$ . Then, the oracle returns  $\mathcal{K}$  if  $b = 0$  or a random key  $\mathcal{K}'$  from the domain of keys if  $b = 1$ . If the instance has not terminated yet, or has not accepted, or the user is adversary-controlled, then the oracle returns  $\perp$ .

This query determines the adversary's success in telling apart a genuine session key from an independent random key. In the BPR model, we assume without loss of generality that the adversary never queries about the same instance twice, and does not query a partner instance for an already tested instance (where the partners are defined below). Consequently, all answers by the oracle are consistent for  $b$  without further checking. In the BR model, we assume that the adversary only makes a single Test-query during the attack.

$\text{Register}(U^*, pk^*)$  allows the adversary to register a public key  $pk^*$  on behalf of a new user (identity)  $U^*$ . The user is immediately considered to be

adversary-controlled.

In addition, since we work in the random-oracle model, the attacker may also query a hash function oracle. In case of PACE and in PACE|AA the adversary may additionally query an ideal-cipher oracle in order to encrypt messages and decrypt ciphertexts, both chosen adaptively by it.

We assume that the adversary always knows if an instance has terminated and/or accepted. This seems to be inevitable since the adversary can send further messages (in subsequent protocols) to check for the status. We also assume that the adversary learns the session id and the partner id immediately for accepting runs.

**DIFFERENCES BETWEEN THE BR AND BPR MODEL.** As mentioned above, in the BR model we can restrict ourselves to a single test query when analyzing protocols based on asymmetric cryptography; it increases the success probability of an adversary by a factor of at most “#test-queries” when compared to multiple queries.

In the case of the password-based BPR model we must allow multiple queries to the test oracle. This is the most significant difference to the model of Abdalla *et al.* [AFP05]. For simplicity reasons we denote the BPR model with multiple queries again by BPR. In the BPR model for PACE the Register-oracle becomes obsolete because there are no certified public keys involved and the active adversary can simulate users on its own (by choosing the password) through an active attack.

**PARTNERS, CORRECTNESS AND FRESHNESS.** We say that instances  $U_i$  and  $U'_j$  are *partnered* if both instances have terminated in an accepting state with the same output for *sid* and each *pid* identifies the other party as the alleged partner. Instance  $U_i$  is called a partner to  $U'_j$  and vice versa. Any untampered execution between honest users should be partnered and, in particular, they should end up with the same key (this correctness requirement ensures the minimal functional requirement of a key agreement protocol).

Neglecting forward secrecy for the moment, an instance  $(U, i)$  is called *fresh* if  $U$  is not controlled by the adversary, there has been no  $\text{Reveal}(U, i)$ -query at any point, neither has there been a  $\text{Reveal}(U', j)$ -query where party  $U'_j$  is a partner to  $U_i$ , nor is  $(U, i)$  partnered with an adversary-controlled party, nor has a user been corrupted; else, the instance is called *unfresh*. In other words,

### 3. Security Model

---

fresh executions require that the session key has not been leaked (by either partner) and that no Corrupt-query took place.

To capture forward secrecy we refine the notion of freshness and further demand from a fresh instance  $(U, i)$ , as before, the following: the session key has not been leaked through a Reveal-query; for each Corrupt( $U$ )-query there has been no subsequent Test( $U, i$ )-query involving  $U$ , or, if so, then there has been no Send( $U, i, m$ )-query for this instance at any point. In this case we call the instance *fs-fresh*, else *fs-unfresh*. This notion captures the fact that it should not help the adversary to corrupt some party after the test query, and even if corruptions take place before test queries, then executions between honest users are still protected (before or after a Test-query).

**AKE SECURITY.** The adversary  $\mathcal{A}$  eventually outputs a bit  $b'$ , trying to predict the bit  $b$  of the Test-oracle. We say that the adversary wins if  $b = b'$  and instance  $(U, i)$  in the test query is fresh (resp. fs-fresh). Ideally, this probability should be close to  $1/2$ , implying that the adversary cannot do significantly better than guessing.

To measure the resources of the adversary we denote by

- $t$  the number of steps of the adversary, i.e., its running time, (counting also all the steps required by honest parties)
- $q_e$  the maximal number of initiated executions (bounded by the number of Send- and Execute-queries),
- $q_h$  the number of adversarial queries to the hash oracle,
- $q_c$  the number of adversarial queries to the ideal-cipher oracle.

We often write  $Q = (q_e, q_h)$  or  $Q = (q_e, q_h, q_c)$ , depending on whether we consider an ideal cipher, and say that  $\mathcal{A}$  is  $(t, Q)$ -bounded.

Define now the AKE advantage of an adversary  $\mathcal{A}$  for a key agreement protocol  $P$  by

$$\begin{aligned} \mathbf{Adv}_P^{\text{ake}}(\mathcal{A}) &:= 2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1 \\ \mathbf{Adv}_P^{\text{ake}}(t, Q) &:= \max \left\{ \mathbf{Adv}_P^{\text{ake}}(\mathcal{A}) \mid \mathcal{A} \text{ is } (t, Q)\text{-bounded} \right\}. \end{aligned}$$

The forward secrecy version is defined analogously and denoted by  $\mathbf{Adv}_P^{\text{ake-fs}}(t, Q)$ .

For a secure protocol in the BR model we expect that the advantage is close to 0 — formally, the advantage is smaller than a negligible function in the security parameter of the AKE protocol. For password-based protocols and the BPR model we merely expect that the advantage is close to  $q_e/N$ , where  $q_e$  is the number of executions where the adversary takes actively part, and  $N$  the size of the password dictionary. The reason lies in the fact that  $N$  here is usually much smaller than, for example, the space of cryptographic keys, and that the adversary could find the correct password in the executions by purely-guessing. Indeed, one can easily construct a trivial adversary in the BPR model which guesses passwords and has  $q_e/N$  advantage. In this spirit we demand from a secure password-based protocol that this trivial attack is the best possible one.

### 3.2.2. The eCK Security Model

The BR model is a strong security model providing confidentiality of agreed session keys and authenticity (i.e., at most one partner will hold the same derived keys). In addition, one can show forward secrecy when adapting the freshness notion. However, as pointed out by LaMacchia *et al.* [LLM07], some attack scenarios are not considered in the BR model. For this reason, LaMacchia *et al.* [LLM07] proposed the eCK model, an extension of the AKE model by Canetti and Krawczyk [CK01].

The eCK model modifies the BR model in such way that the adversary can derive information about the users' secrets, and its winning condition. The idea is that the adversary can now also get information about the internal secrets of the users within a session, i.e., the randomness, but without compromising the long-term secret key nor the session key. This fine-grained distinction allows us to model further desirable security properties, as we discuss in Section 3.4.

**ATTACK MODEL.** The adversary in the eCK model obtains essentially access to two additional oracles:

**LongTermReveal( $U$ )** returns the long-term secret key of the user  $U$ . Querying this oracle does not change user's state, i.e., an honest user  $U$  remains honest even if its long-term key is revealed through this oracle.

**EphKeyReveal**( $U, i$ ) returns the ephemeral secret key of the user  $U$  in instance  $i$ . This encompasses all internal random coins in this session, but neither the long-term secret key nor the derived session key (if it exists) are divulged. We denote an ephemeral secret key by  $esk_U$ , where the index determines the user.

The AKE security is defined analogously to the BR model requiring a fresh instance as input to the Test-oracle. In order to define freshness (or, respectively, clean [CK01]) here, let  $sid$  be the session identity of a completed session by participant  $U$  and  $sid^*$  the corresponding session of partner  $U'$ , which needs not necessarily exist. Then, we call a session  $sid$  fresh if *none* of the following conditions is satisfied:

- $U$  or  $U'$  is adversary-controlled. This rules out trivial attacks where the malicious participant follows honestly the protocol specification, computes the session key, and picks this instance or the one of the partner as the challenge.
- The adversary  $\mathcal{A}$  requested the session key of the session  $sid$  or  $sid^*$  (if it exists) via a Reveal-query. As before — in the BR model —, a disclosed session key cannot be tested.
- The session  $sid^*$  exists, and  $\mathcal{A}$  asked for either  $sk_U$  and  $esk_U$  or  $sk_{U'}$  and  $esk_{U'}$  (via oracle queries). This corresponds to the situation where an adversary already has knowledge of all necessary information to compute the session key itself.
- The session  $sid^*$  does not exist, and  $\mathcal{A}$  asked for either  $sk_{U'}$  or both  $sk_U$  and  $esk_U$  (via oracle queries). This comes close to the previous case except that the adversary acts as  $U'$  with the knowledge of the long-term key  $sk_{U'}$ .

Note that, as opposed to the BR model, sessions where, for instance, the long-term key of a protocol participant is disclosed to the adversary, remain fresh (unless the ephemeral secret of that party is leaked to the adversary). Even more, an adversary can test a session for which it knows all long-term secret keys of the parties. Any (partial) leakage in the BR model, which is captured by Corrupt-queries, makes the corresponding session unrefresh.

### 3.3. Security Model for Secure Channels

In this section, we define the security model for secure channels (SC) when executed after an AKE protocol; our approach here is monolithic, i.e., we

do not separate the two component protocols. In particular, we consider the secure composition of both protocols. If one is only interested in the security of the secure channel, then one simply assumes that the session keys, which are input to the secure channel are sampled uniformly from the key space and distributed, accordingly, among the users. Note that a modular security investigation where the security of each component is assessed separately, does not necessarily remain secure once the protocols are composed (see [BFWW11] on composability of key-exchange protocols). Nevertheless, composed protocols with a positive proof in a monolithic security definition are by definition secure in the given composition, even though providing the security statement is usually more challenging.

The security model follows the BR model for key exchange as discussed in the previous section, but with the following modifications. We consider executions of the AKE protocol together with the SC protocol, and the adversary may initiate the Secure Channel phase after the key exchange by querying  $\text{Send}(U, i, c)$  with  $c = \emptyset$  (the AKE protocol is then considered completed). In order to protect the transmitted data (instead of the session keys), we modify the Test-oracle as follows. Instead of obtaining a genuine or random session key (according to  $b$ ) by asking the Test-oracle on  $(U, i)$ , the adversary may test the SC protocol: it eavesdrops on the communication of either the genuine secured execution between two honest participants for this phase ( $b = 0$ ) or a secured “dummy” execution where trivial messages consisting of 0-bits are exchanged ( $b = 1$ ). The adversary is even allowed to choose the outgoing messages  $m$  (resp.  $m'$ ) of both participants. If the adversary cannot tell apart those executions, the composition of the key-exchange protocol with the subsequent secure channel protocol provides confidentiality of the messages.

Note that we demand from a secure channel even more than confidentiality of exchanged messages. In fact, a secure channel should prevent adversaries from modifying, replaying, or adding messages (and more). For this reason, we require in addition that honest parties only answer to a received cryptogram  $c$  if it is generated by the intended communication partner (as defined in the AKE protocol) *and* this cryptogram  $c$  was not delivered before. Put differently, only if an adversary passes cryptograms to the parties (leaving them untouched), do honest participants react to them.

Formally, we modify the Test-oracle as follows.

**Test** $(U, i, m, U', j, m')$ . The oracle is again instantiated with a random and secret bit  $b \in \{0, 1\}$ . The oracle outputs  $\perp$  if either (i) the AKE execution between  $(U, i)$  and  $(U', j)$  is not completed, or (ii) there is no

session key  $\mathcal{K}$  for one of these instances  $(U, i)$  or  $(U', j)$ , or (iii) the parties are unpartnered or dishonest. In addition, both parties must not have received already a message in the secure channel. Now, the oracle generates the communication for outgoing messages  $m$  and  $m'$  as specified in the secure channel protocol. The partners always receive these honestly generated cryptograms as opposed to the adversary. If  $b = 1$ , the oracle returns the adversary a communication generated by exchanging encryptions of 0-string messages of same length. Otherwise, it receives the transcript of the genuine communication.

In addition, we prohibit the use of Reveal-queries. This is mandatory since an adversary could query the Reveal-oracle after the secure channel is initiated. In that case, the Reveal-oracle could potentially return keys for the dummy communication and disclose the secret bit  $b$  trivially. A rigorous discussion on this fact is given in [BFS<sup>+</sup>13].

The security requirements for authenticated key exchange including the definition of fresh sessions carries over to secure channels. An adversary is declared to be successful if all test sessions are fresh and

**Forgery:** a  $\text{Send}(U, i, c)$  to an honest instance  $U_i$  with  $c \neq \emptyset$  does not return a  $\perp$  symbol where  $c$  was not generated by  $U_i$ 's partner before (as a response to a Send-query);

**Replay:** a  $\text{Send}(U, i, c)$  to an honest instance  $U_i$  does not return a  $\perp$  symbol where the same cryptogram was sent before with a response distinct from  $\perp$ ;

**Confidentiality:** the adversary's guess  $b'$  equals  $b$ .

We define the advantage of an adversary in the authenticated key agreement protocol with subsequent secure channel (AKE+SC) as its success probability minus the pure guessing probability  $1/2$ . Note that an adversary who outputs a forgery or performs a replay attack with probability  $\varepsilon$  can be converted to an adversary against the AKE+SC experiment with success probability of  $\frac{1}{2} + \frac{\varepsilon}{2}$ : this adversary simply checks if the forgery or replay attack was successful and, otherwise, returns a random bit  $b'$ .

We define now the AKE+SC advantage of an adversary  $\mathcal{A}$  for AKE protocol  $P_1$  and SC protocol  $P_2$  against the AKE+SC security experiment by

$$\begin{aligned} \text{Adv}_{P_1, P_2}^{\text{ake+sc}}(\mathcal{A}) &:= |2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1| \\ \text{Adv}_{P_1, P_2}^{\text{ake+sc}}(t, Q) &:= \max \left\{ \text{Adv}_{P_1, P_2}^{\text{ake+sc}}(\mathcal{A}) \mid \mathcal{A} \text{ is } (t, Q)\text{-bounded} \right\}. \end{aligned}$$

**Remark.** We stress that the above security model does not capture attacks where an adversary intercepts transmitted cryptograms and forwards them in a modified order. However, as we will show in Chapter 6, the Secure Messaging protocol, which implements the secure channel protocol for the German eID card, prevents those attacks by injecting sequence numbers into the header protected by the cryptogram.

### 3.4. Discussion on the Security Models

In this section we informally describe all desired security properties provided by the key-exchange models B(P)R and eCK, and by the secure channel model from the previous section. We start with the B(P)R model.

#### 3.4.1. Achieved Security Properties in the B(P)R Model

Roughly, security in the B(P)R model guarantees:

- confidentiality of agreed session keys
- resistance against so-called dictionary attacks (online and offline), where an adversary tries to retrieve the password by exhaustive search (BPR model)
- forward secrecy
- authenticity (i.e., there exists only one partner sharing the same session key)

*Confidentially* of agreed session keys means that an adversary is not able to learn information in the key agreement protocol regarding an established session key. Since in B(P)R, the winning condition is that an adversary can distinguish a session key from a random key with significant probability, an unsuccessful adversary does not even learn a single bit of the session key. Thereby, confidentiality is implicitly given by a security proof in the B(P)R security model.

*Dictionary attacks* can be divided into offline and online attacks. In an offline dictionary attack an adversary tries passwords which are compliant to the observed communication or where it actively participated *after* the communication terminated. If the password-based protocol (with a low-entropy password) allows one to perform such attacks, an adversary could try all possible passwords in a reasonable time to break the protocol's security.

### 3. Security Model

---

The BPR security model gives security guarantees against such adversaries, because, otherwise, the following strategy would break the protocol's security in the BPR model. An adversary would eavesdrop an honest communication or would take actively part in the protocol, and, afterwards, run the offline dictionary attack to search for a working password (or at least to rule out a set of possible passwords). Again, it initiates an interaction to this user and picks one of the derived passwords. The success probability to learn the shared secret key is clearly higher than the pure guessing probability of  $1/N$ .

In online attacks, an adversary needs to take part actively in the protocol in order to test a password guess for correctness. Security in the BPR model guarantees that an adversary is able to test only a single password per protocol execution.

*Forward secrecy* entails that loss of a long-term key should not compromise already-distributed session keys and, in addition, future session keys remain private if one only eavesdrops on the corresponding communication. The B(P)R model captures this property (if requested) by modeling freshness with respect to a *Corrupt-query*. This query reveals the long-term secrets of a user. B(P)R allows *Corrupt-queries* if it takes place *after* the test session or if the test session was only eavesdropped by the adversary, i.e., the communication is untouched by the adversary.

*Authenticity* is provided in the B(P)R model by restrictions on test queries with respect to partners — the adversary is not permitted to query the *Reveal-oracle* for a partner instance of a tested session. If an adversary is able to initiate another protocol execution, where two participants agree on the same session key, as different users in a previous session, in such a way that those parties are not partnered, then one can easily query the *Reveal-oracle* for one session and answer test queries to the other instance. Obviously, in that case an adversary can distinguish the key from a random string.

#### 3.4.2. Achieved Security Properties in the eCK Model

The eCK model provides several nice features in addition to the security properties entailed by the BR model. The following three properties are satisfied through a security proof in the eCK model.

- key-compromise impersonation resistance
- (weak) forward secrecy
- resistance against leakage of internal information

*Key-Compromise Impersonation (KCI)*, as considered in [JV96], covers attacks in which the adversary first receives the long-term secret key of a user, and takes advantage of this knowledge to impersonate others to this party. In the eCK model, sessions remains fresh even if an adversary obtained one's long-term key via a LongTermReveal-query. If one successfully launches a KCI attack, then one succeeds in the eCK model as well. Hence, a positive security result in the eCK model excludes the possibility of KCI attacks.

*Forward Secrecy* is captured in the eCK model. A session remains fresh (and therefore, the session key confidential) as long as the adversary misses one of the four secrets (long-term and ephemeral secret key of each party) in a session of honest users, or misses one of three secrets in case the adversary pretended to a party and talked to an honest user (where there is only the long-term secret of the intended partner but no ephemeral key). Therefore, similarly as in BR, forward secrecy is by definition captured. In BR, however, forward secrecy is proven explicitly when considering *fs-fresh* sessions.

*Resistance against leakage of internal information* is a strong security property. The adversary can get information about the internal state of a session and the derived session key should still remain confidential (unless the adversary also elicits the long-term secret key of the corresponding user). This is, for example, a difference compared to the model of Canetti and Krawczyk [CK01] where one can specify more restrictively that the session-state reveal leaks less information, and where such sessions with a state reveal are not considered fresh anymore.

#### 3.4.3. Achieved Security Properties in the AKE+SC Model

The AKE+SC security model for secure channels where the session keys are derived by an authenticated key-exchange protocol builds an end-to-end integrity-secure confidential channel.

*Confidentiality* of transmitted messages says basically that, without knowledge of the session key, all messages exchanged through the channel are hidden. An adversary learns no information about the exchanged messages, not even a single bit. This is captured by the fact that adversaries able to distinguish a genuine communication from the "dummy" communication with sufficient advantage, succeed in the security model. Consequently, a security proof in the AKE+SC model implies confidentiality.

### 3. Security Model

---

The *integrity* of messages over the channel is preserved as well. In particular, this means that no adversary can modify, replay, or add messages in the channel. An adversary that successfully injects forged or replayed messages through the channel, is, by definition, declared successful in breaking the security of the secure channel. Hence, any modifications on messages within the channel are detected.

## **Part II.**

# **The Protocols used by the German eID Card**



## 4. The Password Authenticated Connection Establishment Protocol

Through ISO/IEC JTC1 SC17 WG3/TF5 [ISO10] the International Civil Aviation Organization (ICAO) has adopted the Password Authenticated Connection Establishment (PACE) protocol [BSI10] to secure the contactless communication between machine-readable travel documents (MRTD) including identity cards, and a reader. PACE was primarily developed to replace the Basic Access Control (BAC) mechanism [ICA06] for MRTD.

BAC is currently the de facto standard for almost all electronic passports. It provides basic security against skimming and eavesdropping attacks. However, BAC does not provide sufficient security for today's and future threats. More precisely, BAC realizes (password-based) authentication using only symmetric-key mechanisms. Even though, symmetric-key operations are much faster than asymmetric ones, the password strength bounds the entropy (or strength) of the session keys derived from a password-based key-exchange protocol. For electronic passports, the password is basically the Machine Readable Zone (MRZ) which offers only limited strength (entropy). As the MRZ consists of the date of birth, expiry date, and a relatively short serial number, the maximum entropy of the MRZ is estimated at 36 bits according to [LG07]. This is clearly not enough entropy for the established session keys.

To overcome this issue and to offer cryptographically strong keys as output of the key-exchange protocol, the German Federal Office for Information Security (BSI) proposed the PACE protocol in 2007 [BSI10]. This protocol works even for weak passwords (= low entropy), and, therefore, is suitable for the use in electronic identity cards, like the German eID. Indeed, PACE is implemented in all German eID cards to ensure secure communication between the chip card and the terminal or card reader depending on whether the communication takes place during border control — the ePass scenario — or during online authentication. The password of the eID card depends on the application. If the card acts as a passport, and connects to a reader at a border control, the MRZ defines the password. Otherwise, the card owner enters his

Personal Identification Number (PIN) — of low entropy — to the card reader to authenticate. If the PIN is blocked, for instance through mistyping, one can use the PIN Unblock Key (PUK). In exceptional cases (e.g., at a police check), the reader is authorized to read the Card Access Number (CAN) as the password printed or displayed on the card.

In the remainder of this chapter, we consider the execution of PACE with a (card) reader, as this is the more general case and possibly the one that is more susceptible to attacks. Roughly, the PACE protocol comprises four phases:

1. In the *randomization* phase, the chip card sends an encrypted nonce to the reader. For correct encryption and decryption the knowledge of the password  $\pi$  is required.
2. In the *mapping* phase, both the chip card and the reader derive a random generator of a group by executing an interactive protocol Map2Point. There are several options for the employed Map2Point protocol, and address these later in this chapter.
3. In the *key establishment* phase, the participants execute the Diffie–Hellman protocol on ephemeral keys, using the group generator obtained from the mapping phase.
4. In the *key confirmation* phase, the chip and the reader exchange and verify authentication tokens, thereby, ensuring that the partner knows the session key.

Several works provide rigorous security analyses of the PACE protocol [BFK09, CS10, CGIP12, CSD<sup>+</sup>12], and thus, furnish evidence about the security of the PACE protocol. In 2009, Bender *et al.* [BFK09] proved the security of PACE as a (password-based) key-exchange protocol in the BPR model [BPR00] (cf. Section 3.2.1). The authors gave a generic result, i.e., they proved security based on the security of chosen mapping Map2Point. While [BFK09] showed the security of the DH2Point instantiation of Map2Point— i.e, the mapping is realized by a DH exchange—, Coron *et al.* [CGIP12] examined the instantiation Hash2Point. The mapping Hash2Point maps a nonce directly to a point in an elliptic curve, which can be instantiated by hash functions proposed in [SvdWo6, BCI<sup>+</sup>10, Ica09].

In [CS10], a (symbolic) algebraic-logic security proof of PACE in the Dolev-Yao (DY) model [DY81] has been carried out in the Verification Support Environment (VSE) tool, yielding a machine generated proof of all its security properties. The DY model is based on a message algebra given by cryptographic operations and equations. Cheikhrouhou *et al.* [CSD<sup>+</sup>12] describe the

integration of the cryptographic security analysis of PACE in [BFK09] and the algebraic-logic security proof in [CS10].

We emphasize explicitly that the results of this chapter, namely the design of the PACE protocol and its security analyses are not part of the thesis' contribution. We merely describe and recall given security results on PACE for sake of completeness. The thesis should provide the reader with a comprehensive and rigorous study on all protocols used by the German identity card.

**READER'S ROADMAP.** In Section 4.1, we give a detailed protocol description of PACE, including all options for the mapping function Map2Point. Then, in Section 4.2, we recall more formally the security results on PACE.

## 4.1. Protocol Description

As mentioned before, the Password Authenticated Connection Establishment protocol can be divided into four phases: randomization, mapping, key establishment, and key confirmation. Next, we describe these phases in more detail. The complete protocol is depicted in Figure 4.1.

When the chip card connects to a card reader, the user is asked to enter a password. This password  $\pi$  is derived from one of the following passwords: the PIN, PUK, CAN or the MRZ. If MRZ is chosen, the hashed MRZ sets the password. Otherwise, the input characters are converted to octets using the ISO/IEC 8859-1 character set. Both the chip card and card reader are initialized with authenticated group parameters  $\mathcal{G} = (a, b, p, q, g, \lambda)$  describing a subgroup of order  $q$ , generated by  $g$ , of an elliptic curve with parameters  $a, b, p$  for security parameter  $\lambda$ .

In the *randomization* phase, the chip card chooses a random nonce  $s \in \{0, 1\}^\ell \subseteq \mathbb{Z}_q$ . This nonce is then encrypted through a cipher  $\mathcal{C}$  using as key  $K_\pi := \mathcal{H}(0||\pi)$ , i.e., the password  $\pi$  is hashed in order to suit in size for an encryption key. The card sends this encrypted nonce over to the reader, which recovers  $s$  by decrypting under  $K_\pi$ .

In the *mapping* phase, the participants jointly generate a random generator  $\hat{g}$  which becomes the group generator for the subsequent DH step. For this purpose, they execute the interactive protocol Map2Point which takes as input

#### 4. The Password Authenticated Connection Establishment Protocol

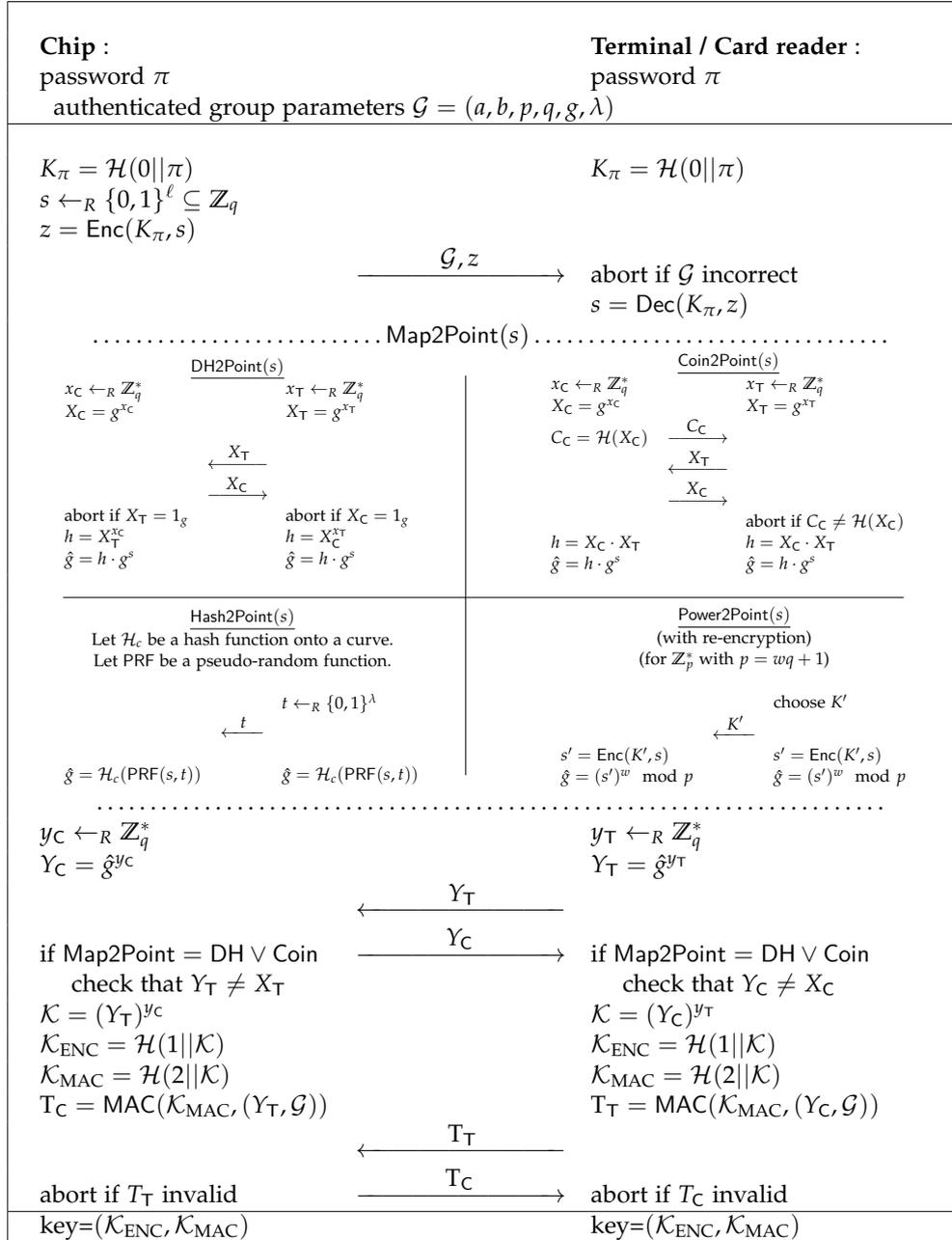


Figure 4.1.: The Password Authenticated Connection Establishment protocol

the nonce  $s$ . One of the following four options for Map2Point is chosen when PACE is executed:

- DH2Point.** The chip card and the reader perform a Diffie-Hellman execution. For this, both choose ephemeral exponents  $x_C$  and  $x_T$ , and exchange their public part  $X_C = g^{x_C}$  and  $X_T = g^{x_T}$ , respectively. Both parties have to verify that the received DH part is not the neutral element  $1_g$  of the group generated by  $g$ . The resulting DH value  $h = \text{DH}(X_C, X_T)$ , multiplied by  $g^s$ , is the new generator  $\hat{g}$ .
- Coin2Point.** In the coin-flipping protocol Coin2Point the chip card first sends a commitment for its input  $X_C$ , and reveals  $X_C$  after the reader sends its input  $X_T$  to the card. The commitment  $C_C$  on  $X_C$  binds  $X_C$  to the chip  $C$  such that the chip card cannot change its input once the commitment is delivered to the reader. Both inputs, together with  $g^s$ , are then multiplied such that the generator  $\hat{g}$  is set to  $\hat{g} = g^{x_C + x_T + s}$ .
- Hash2Point.** This mapping assumes the availability of a hash function  $\mathcal{H}_c$  which maps a string directly to a point in the elliptic curve. Possible instantiations are given in [SvdWo6, BCI<sup>+</sup>10, Ica09]. Now both parties apply a pseudo-random function PRF on input a nonce  $t$  chosen by the reader using as key the nonce  $s$ . The output of  $\text{PRF}(s, t)$  is then hashed to a curve point  $\hat{g}$  using  $\mathcal{H}_c$ .
- Power2Point.** This mapping works only for groups over  $\mathbb{Z}_p^*$  where  $p$  is defined as  $p = wq + 1$ , with  $q$  prime and  $w$  having large prime factors. Now, the reader chooses a new key  $K'$  for the cipher  $\mathcal{C}$  and encrypts the nonce  $s$  using  $K'$  such that  $s' = \text{Enc}(K', s)$ . Both parties set  $\hat{g} = (s')^w \bmod p$  after the reader handed over the key  $K'$ . Since  $w$  has large factors and the nonce  $s'$  looks uniform —  $\mathcal{C}$  is ind-cpa secure —  $\hat{g}$  is distributed statistically close to uniform.

In the *key establishment* phase, the chip card and the reader perform a DH exchange. To compute their public part of the DH exchange the parties raise  $\hat{g}$  to their ephemeral keys  $y_C$  and  $y_T$ , respectively. Both parties, however, check that they receive a “fresh” input to DH. That is, neither  $\hat{g}^{y_C}$  nor  $\hat{g}^{y_T}$  appeared in the mapping phase. Otherwise, the parties abort the execution. The parties derive the session key(s)  $\mathcal{K}_{\text{ENC}}$  and  $\mathcal{K}_{\text{MAC}}$  by applying the hash function  $\mathcal{H}$  on input  $(1||\mathcal{K})$  and  $(2||\mathcal{K})$  where  $\mathcal{K}$  is the deduced Diffie-Hellman value, i.e.,  $\mathcal{K}_{\text{ENC}} = \mathcal{H}(1||\mathcal{K})$  and  $\mathcal{K}_{\text{MAC}} = \mathcal{H}(2||\mathcal{K})$ . At this point, the parties know the session key only if they used to correct PIN and know their ephemeral keys.

In the *key confirmation* phase, both parties now explicitly confirm knowledge of the session key. For this, they authenticate the public values  $(\hat{g}^{y_C}, \mathcal{G})$  and

$(\hat{g}^{y_C}, \mathcal{G})$ , respectively, using as key  $\mathcal{K}_{\text{MAC}}$ . If verification fails (= outputs 0), parties abort the execution.

### 4.2. On the Security of PACE

In this section, we recall and comment on given security analyses of the PACE protocol.

#### 4.2.1. Results by Bender *et al.*

Bender *et al.* [BFK09] prove the security of the PACE protocol in the BPR model (cf. Section 3.2.1). The authors show that the PACE protocol is a strongly-secure password-based key-exchange protocol under the following assumptions:

- The encryption scheme and the hash function used in PACE behave ideally, i.e., the result hold in the random-oracle and ideal-cipher model.
- The message authentication scheme  $\mathcal{M}$  is unforgeable against adaptively chosen message attacks.
- A newly-introduced assumption, namely the so-called gPACE-DH problem is hard, holds for the chosen mapping function Map2Point.

In order to prove security, the authors introduce a new number-theoretic assumption, called the general password-based chosen-element DH (gPACE-DH) problem, which is related to the DH assumption. It allows to reason about the security of the PACE protocol independently of the respective choice for Map2Point. The gPACE-DH problem captures the fact that in PACE the adversary may contribute to the input of DH either directly, via  $y_C$  (resp.  $y_T$ ), or indirectly, via the mapping into a group generator  $\hat{g}$  including the choice of nonce  $s$  and the ephemeral values in Map2Point. Consequently, the gPACE-DH problem states that no efficient adversary is able to compute the final DH value even if the adversary has a strong impact on the respective DH execution. In particular, this means that the hashed DH value, which defines the session key(s), looks random to an adversary, and hence, the adversary cannot win in the BPR model by distinguishing a session key from a randomly sampled key. Formally, gPACE-DH is defined as follows.

**Definition 4.2.1 (gPACE-DH Hardness [BFK09, Def. 4.3])** *The general password-based chosen-element DH problem is  $(t, N, Q, \epsilon)$ -hard (with respect to*

Map2Point) if for any algorithm  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$  running in time  $t$ , the probability that the following experiment returns 1 is at most  $\frac{1}{N} + \varepsilon$ .

**Experiment**  $\text{Exp}_{\text{Map2Point}, N, Q, \mathcal{A}}^{\text{gPACE-DH}}(\lambda)$

pick authenticated group parameters  $\mathcal{G} = (a, b, p, q, g, \lambda)$

$(s_1, \dots, s_n, \text{st}) \leftarrow_{\mathcal{R}} \mathcal{A}_0(\mathcal{G}, N)$  with  $s_1, \dots, s_n$  pairwise distinct

pick  $y_{\top} \leftarrow_{\mathcal{R}} \mathbb{Z}_q$  and  $k \leftarrow_{\mathcal{R}} [N]$

let  $\hat{g}$  be the local output of the honest party in an run of  $\text{Map2Point}(s_k)$ ,  
where  $\mathcal{A}_1(\text{st})$  controls the other party (updating state information  $\text{st}$ ).

$(Y_C, K_1, \dots, K_Q) \leftarrow \mathcal{A}_2(\text{state}, \hat{g}^{y_{\top}})$

Return 1

iff  $Y_C \neq 0$  and  $K_i = Y_C^{y_{\top}}$  for some  $i \in [Q]$ .

The probability is taken over the choice of  $\mathcal{G}$ ,  $y_{\top}$ , all coin tosses of  $\text{Map2Point}$ , and  $\mathcal{A}$ .

We let  $\text{Adv}_{\text{Map2Point}}^{\text{gPACE-DH}}(t, N, Q)$  denote (a bound on) the value  $\varepsilon$  for which the gPACE-DH problem is  $(t, Q, N, \varepsilon)$ -hard.

Bender *et al.* show that if one breaks the DL problem, then one can build an adversary against the gPACE-DH problem. The converse is not known to hold in general; however, the authors prove the equivalence of gPACE-DH and DH in the generic group model (introduced in [Sho97]) and under the assumption that  $\hat{g}$  is defined as  $\hat{g} = g^s \cdot h$  for randomly chosen group element  $h$ . Both  $\hat{g}$  and  $h$  are given to  $\mathcal{A}_2$  in the experiment.

The following theorem provides concrete bounds on the successful probability of attacking the PACE protocol with respect to the generic protocol  $\text{Map2Point}$ .

**Theorem 4.2.2 ([BFK09, Thm. 5.1])** *In the ideal-cipher and random-oracle models, we have:*

$$\begin{aligned} \text{Adv}_{\text{PACE}}^{\text{ake}}(t, Q) &\leq \frac{q_e}{N} + q_e \cdot \text{Adv}_{\text{Map2Point}}^{\text{gPACE-DH}}(t^*, N, q_h) \\ &\quad + q_e \cdot \text{Adv}_{\mathcal{M}}^{\text{unf-cma}}(t^*, 2q_e, 2q_e) + \frac{2q_e N^2 + 8q_e^2 N + q_c q_e}{\min\{q, |\text{Range}(\mathcal{H})|\}} \end{aligned}$$

for  $t^* = t + O(\lambda(q_e^2 + q_h^2 + q_c^2 + \lambda))$  and  $Q = (q_e, q_c, q_h)$ , where  $\lambda$  denotes the security parameter,  $q = q(\lambda)$  the group order,  $q_e$  the number of protocol executions launched by the adversary,  $q_h$  (resp.  $q_c$ ) denotes the number of queries made to the random oracle (resp. to the ideal cipher). We denote by  $N$  the size of the dictionary the passwords are drawn from.

In other words, in order to learn any information about the established session key an adversary can merely guess the password as long as the gPACE-DH is hard with respect to Map2Point and the message authentication scheme is unforgeable. Beside the generic result on the security of PACE, the authors additionally show the hardness of the gPACE-DH when Map2Point is instantiated by DH2Point.

To be precise, the above theorem holds for a slightly modified PACE protocol. In fact, the authors apply the following modification to PACE. In the key-establishment phase, both parties additionally derive a MAC key  $\mathcal{K}'_{\text{MAC}} = \mathcal{H}(3||\mathcal{K})$  which in turn sets the respective MAC key in the key confirmation. This is necessary to prove security in the BPR model. As the BPR model requires that the corresponding session keys for secure key-exchange protocols are indistinguishable from random values, any use of the session key in the protocol execution itself allows an adversary to test the challenge key by checking its consistency with the protocol transcript.

A similar strategy of modification is recommended in [BPR00] and is applied ever since, when parties authenticate the protocol transcript already with the established session key, like in [DFG<sup>+</sup>13b]. Nonetheless, the security of the modified protocol carries over to the original protocol if the message structure in the key confirmation phase is different than in the actual payload sent through the secure channel. Since the channel is also implemented by secure messaging, and due to the similarity between the BPR, and respectively, the BR model, our monolithic analysis for EAC and Secure Messaging in the following two chapters immediately carries over. Hence, roughly speaking, the security implications of Theorem 4.2.2 hold for the original PACE as well.

### 4.2.2. Results by Coron *et al.*

Coron *et al.* [CGIP12] extend the work by Bender *et al.* [BFK09] and show that the PACE protocol is secure when the Map2Point protocol is instantiated by Hash2Point. The specific instantiation of PACE is called PACE v2 Integrated Mapping and is standardized in [ISO10]. Their results rely on the Gap Chosen-Base Diffie-Hellman (GCBDH) assumption. Informally, the GCBDH assumption states that on input  $(g, g^a, g^b)$  for uniformly sampled  $a, b \in [\text{ord}\{\langle g \rangle\}]$  it is hard to compute a tuple  $(h, h^{1/a}, h^{1/b})$  even given a DDH oracle — a similar statement as in GDH (cf. Definition 2.2.4).

In their security proof, the authors of [CGIP12] assume that Map2Point (= Hash2Point) behaves as a random oracle mapping to a point in the curve. However, they show that if the hash function  $\mathcal{H}_c$  is simulatable and its output is statistically close to uniform, then the security results carries over. Indeed, as the authors show, the functions defined in [SvdWo6, Ica09] fulfill the required properties. Recently, [BCI<sup>+</sup>10] proposed a hash function mapping to elliptic curves points, which also fulfills the above requirements.

Note that the key derivation function is still modeled as a random oracle, and, thus, their results also hold in the random oracle.

### 4.2.3. Further Security Studies on PACE

Concurrently to [BFK09], Cheikhrouhou and Stephan gave a (symbolic) algebraic-logic security proof of PACE [CS10], in the Dolev-Yao (DY) model [DY81], which has been carried out in the Verification Support Environment (VSE) tool, yielding a machine-generated proof of all its security properties. The DY model is based on a message algebra given by cryptographic operations and equations. The operations define the syntax of protocol messages while the equations formalize in an abstract way which computations honest participants must perform during the protocol execution. A subset of these operations is available for the adversary to analyze observed messages and to synthesize new ones. This attacker model is very powerful in the sense of an unrestricted access (of the adversary) to the communication lines. On the other hand, it limits the abilities of the adversary in attacking cryptography by assuming that exactly the algebraic (or symbolic) computations can be carried out by the adversary.

In follow-up work [CSD<sup>+</sup>12], Cheikhrouhou *et al.* describe an approach to merge the cryptographic security analysis in [BFK09] and the algebraic-logic security proof in [CS10], aiming at a reliability improvement in the security of PACE. Their merging approach is based on an adequate formalization of the BR model (cf. Section 3.2.1) allowing to attribute public bit strings (values) to symbolic expressions for corresponding applications of the cryptographic functions. Sequences of alternating intruder queries and protocol oracle responses (BR traces) are associated in this way with a symbolic structure that allows one to define DY computations in the BR model.

In [BDF12], Brzuska *et al.* survey the recent cryptographic developments for building secure channels, especially for TLS and the German identity card. In

#### 4. The Password Authenticated Connection Establishment Protocol

---

particular, the authors comment on the security of PACE and EAC building on the results of [BFK09] and our security analysis on EAC in Chapter 5.

## 5. The Extended Access Control (EAC) Protocol

The Extended Access Control (EAC) protocol was proposed in 2005 by the German Federal Office for Information Security (BSI), for the German passports (ePASS). It is meant to provide secure key establishment between a chip card and a terminal, using a public-key infrastructure. The latest version of EAC, recommended for the German ID card, is presented in this thesis (with some slight simplifications for the sake of presentation, but without alteration of security properties of the overall protocol). EAC enables the terminal to access sensitive data on the card (e.g., stored finger prints). The BSI integrated EAC in the German ID card in order to ensure the full protection of all recorded personal data.

The EAC protocol consists of two phases: Terminal Authentication (TA) which is a challenge-response protocol in which the terminal signs a random challenge (and an ephemeral public key) with its certified signing key; and Chip Authentication (CA), in which both parties derive a Diffie–Hellman key from the terminal’s ephemeral key and the chip’s static certified key; in the second part the chip ends by computing and sending a message authentication code in order to authenticate.

We note that the EAC key-exchange protocol is just one component in the security framework for identity cards and passports. Another sub protocol is Password Authenticated Connection Establishment (PACE) [BSI10], which ensures a secure key exchange between the card and a reader (which sits in-between the card and the terminal). The PACE protocol is executed first, and then the communication between the card and the terminal is secured through the EAC protocol. We note that the reader and the terminal should also be connected securely through, say, SSL/TLS, before the keys in the EAC protocol are derived by the chip and the terminal. We comment on security issues related to the composition at the end of this chapter.

**ANALYZING THE EAC PROTOCOL.** We analyze the EAC protocol as an authenticated key-exchange protocol in the BR security model (cf. Section 3.2.1). We show that the protocol is secure in this model, assuming that the underlying cryptographic primitives are secure (signatures, certification, and message authentication codes), that the deployed hash function for key derivation behaves as a random oracle, and assuming the gap Diffie–Hellman problem [BLS01] is hard. Recall that the latter assumption says that it is infeasible to solve the *computational* Diffie–Hellman problem even if one has access to a *decisional* Diffie–Hellman oracle. This assumption is equivalent to the standard computational DH assumption for pairing-friendly elliptic curves since the *decisional* Diffie–Hellman problem is easy in such groups [JN03].

Our analysis is in terms of concrete security, identifying exactly how weaknesses of the EAC protocol relate to attacks on the underlying assumptions and primitives. We note that we cannot prove the EAC protocol secure in the eCK model (cf. Section 3.2.2). This is mainly because the chip card does not use ephemeral secrets due to its limited resources; consequently, forward secrecy cannot be achieved without further assumptions (like tamper-resistant hardware). However, we still show that the EAC protocol achieves key-compromise impersonation resilience (cf. Section 3.4.2).

**RELATED WORK.** We like to mention that the security of the earlier version of EAC, implemented on the electronic passports, received some attention in e.g., [MVV07, PPW08, SBPV08]. The security of the latest version — that is, the up-to-date version, currently deployed in the German eID card — remained, to the best of our knowledge, open until our results. In [Nit09], the author briefly discusses whether attacks on previous protocols apply to the latest EAC. However, he does not provide a rigorous security analysis.

**READER’S ROADMAP.** In Section 5.1 we describe the EAC protocol. We subdivide here the description into the Terminal Authentication and Chip Authentication protocols. We elaborate on the security of the EAC protocol in Section 5.2. We discuss further security properties in Section 5.3.

## 5.1. Protocol Description

The EAC protocol, or more precisely, the composition of the Terminal Authentication (TA) protocol and the Chip Authentication (CA) protocol, allows mutual authentication between a terminal and a chip and the establishment of an authenticated and encrypted connection. The following subsections present the two main components of the Extended Access Control protocol. Afterwards we define the EAC protocol and comment on deviations in the presentation here, compared to the original protocol.

### 5.1.1. Terminal Authentication

Terminal Authentication requires the terminal to present a proof of authority, thereby allowing the chip to check whether the terminal is allowed to access sensitive data. This proof works with the use of a Public Key Infrastructure (PKI). Terminals may be operated by an official domestic document verifier or by a foreign document verifier. As document verifier (DV), such verifiers are certified by the Country Verifying Certification Authority (CVCA) of the issuing country.

In addition, terminals are given a certificate specifying the access authorization and a signed public key. By a challenge-response step, the terminal convinces the chip to allow it permission to read (some of) the chip's data. In this step the terminal signs a nonce sent by the chip together with its compressed ephemeral key, which is used in the following Chip Authentication protocol. This step, therefore, somewhat "connects" the TA and the CA phases.

From a cryptographic point of view, the TA protocol requires a compression function  $\text{Compr} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ , and a secure signature scheme. A signature scheme consists of three efficient algorithms  $\mathcal{S} = (\text{SKGen}, \text{Sig}, \text{SVf})$  to generate a key pair  $(\text{sk}, \text{pk}) \leftarrow \text{SKGen}(1^\lambda)$  such that  $s \leftarrow \text{Sig}(\text{sk}, m)$  allows to sign arbitrary messages  $m$  with the secret key  $\text{sk}$ , and such that the signatures can be subsequently verified via the verification algorithm and the public key  $\text{pk}$ , returning a bit  $d \leftarrow \text{SVf}(\text{pk}, m, s)$ . The scheme should be (perfectly) correct in the sense that for any  $(\text{sk}, \text{pk}) \leftarrow \text{SKGen}(1^\lambda)$ , any message  $m$ , any  $s \leftarrow \text{Sig}(\text{sk}, m)$  we always have  $\text{SVf}(\text{pk}, m, s) = 1$ . For formal definitions of signature and compression schemes together with their security requirements we refer to Chapter 2.

We also assume a certification authority  $\mathcal{CA}$ , modeled similarly to the signature scheme, with algorithms  $\mathcal{CA} = (\text{CKGen}, \text{Certify}, \text{CVf})$ , but where we call the “signing” algorithm *Certify*. This is in order to indicate that certification may be done by means other than signatures. We assume that the keys  $(sk_{\mathcal{CA}}, pk_{\mathcal{CA}})$  of the  $\mathcal{CA}$  are generated at the outset and that  $pk_{\mathcal{CA}}$  is distributed securely to all parties (including the adversary). We also often assume that the certified data is part of the certificate. We note that the  $\mathcal{CA}$  may, as usual, check for correctness of the data it certifies, e.g., verifying well-formedness of certified keys or checking the identity of a key owner; however, in order to model security threats more realistically, we allow the adversary to register arbitrary (well-formed) keys.

In the Terminal Authentication protocol the terminal  $T$  and the chip  $C$  perform the following steps:

1.  $T$  sends a certificate chain to  $C$ , including its certificate, along with the certificates of the DV and CVCA.
2.  $C$  is able to verify the certificate chain of CVCA, DV, and the certificate  $cert_T$  of the terminal. Then,  $C$  extracts  $T$ 's public key  $pk_T$  from the certificate.<sup>1</sup>
3.  $T$  generates an ephemeral Diffie–Hellman key pair  $(esk_T, epk_T, D_C)$  for domain  $D_C$  and sends the compressed ephemeral public key  $\text{Compr}(epk_T)$  to  $C$ .
4.  $C$  randomly chooses a nonce  $r_1 \leftarrow_R \{0, 1\}^\lambda$  and sends it to  $T$ .
5.  $T$  signs the identifier  $ID_C$  of  $C$  along with the nonce  $r_1$  and the compressed public key  $\text{Compr}(epk_T)$ , i.e.,

$$s = \text{Sig}(sk_T, (ID_C, r_1, \text{Compr}(epk_T))) .$$

The signature  $s$  is sent to  $C$  by  $T$ .

6.  $C$  checks if  $\text{SVf}(pk_T, m, s) = 1$  for  $m = (ID_C, r_1, \text{Compr}(epk_T))$ , using the static public key  $pk_T$  of the terminal.

**REMARKS.** We do not consider auxiliary data sent by the terminal as specified in [BS10], since it neither offers any additional security nor hurts the security for the key exchange. The auxiliary data is eventually later used by eID card applications once a secure channel is built; however, the specifications for the protocol do not indicate or restrict the later use of this data. Still, the delivery

---

<sup>1</sup>For the sake of simplicity, we will sometimes use  $cert_T$  and mean therewith the whole certificate chain, including the certificates of CVCA and DV.

of auxiliary data is insignificant for key secrecy, and omitting this data from the description above facilitates the analysis and understanding of the TA protocol.

The static domain parameter  $D_C$  contains the (certified) group description for which the chip and terminal execute the Diffie–Hellman computations. This domain parameter is known to the terminal from PACE, which is run before the EAC protocol. The identifier of the chip  $ID_C$  is defined by the compressed ephemeral public key  $\text{Compr}(epk_C)$  used in the PACE protocol before the Terminal Authentication is invoked. Thus, one establishes a link between the PACE protocol and Terminal Authentication; however, decoupling protocols like PACE and EAC eases the analysis of EAC. The composition of the protocols does not lead to any significant advantage with respect to the BR model, since active adversaries are potentially able to control the card reader and act genuinely for the PACE and SSL/TLS protocol executions.

Further, we assume that the parties abort an execution whenever they receive an unexpected message including either wrong format or a false sequence of messages. This holds also for the following protocols.

### 5.1.2. Chip Authentication

The Chip Authentication protocol provides an authenticity check of chips, as well as a secure session key for encryption and integrity-preservator of subsequently transmitted messages. Unlike TA, there is no challenge-response action, as this leads to certain privacy breaches; see Chapter 9 for more details. Instead, the chip computes the Diffie–Hellman value with its static key and the ephemeral key chosen by the terminal, and both parties then hash this value together with a random nonce. Thereby, the chip obtains the session key for encryption and authentication (and an extra key for authentication in the key confirmation phase). Now, the chip computes a message authentication code over the ephemeral public key of the terminal, using the additional authentication key as the secret, and sends this authentication token to the terminal. The terminal can verify the authenticity by checking the validity of the token with the newly derived key.

In this step we need a message authentication code, which, similarly to signature schemes, is modeled by a tuple  $\mathcal{M} = (\text{MKGen}, \text{MAC}, \text{MVf})$  of efficient algorithms (cf. Definition 2.3.5). We also let  $\mathcal{H}_1$ ,  $\mathcal{H}_2$ , and  $\mathcal{H}_3$  denote hash functions modeled as random oracles. For implementations we assume that

## 5. The Extended Access Control (EAC) Protocol

---

we are given a random oracle  $\mathcal{H}$  and then set  $\mathcal{H}_i(\cdot) = \mathcal{H}(\langle i \rangle \parallel \cdot)$  for some fixed-length encoding  $\langle i \rangle$  of  $i = 1, 2, 3$ .

In the Chip Authentication protocol, the terminal T and the chip C perform the following steps:

1. C sends T its static public key  $pk_C$  and the domain parameters  $D_C$ , together with a certificate  $cert_C$  for  $pk_C$ .<sup>2</sup>
2. After T has checked the validity of  $cert_C$ , T sends C its ephemeral public key  $epk_T$  from the TA phase.
3. C applies the compression function  $Compr$  to the received ephemeral public  $epk_T$  and compares this to the compressed public key received during the Terminal Authentication execution.
4. Both C and T compute the shared key as  $\mathcal{K} = \text{DH}(epk_T, sk_C)$  or resp.  $\mathcal{K} = \text{DH}(pk_C, esk_T)$ .<sup>3</sup>
5. C picks a random  $r_2 \leftarrow_R \{0, 1\}^\lambda$  and derives session keys, computing

$$\mathcal{K}_{\text{ENC}} = \mathcal{H}_1(\mathcal{K}, r_2), \quad \mathcal{K}_{\text{MAC}} = \mathcal{H}_2(\mathcal{K}, r_2), \quad \mathcal{K}'_{\text{MAC}} = \mathcal{H}_3(\mathcal{K}, r_2)$$

where we assume that  $\mathcal{H}_3$  generates the same distribution on keys as  $\text{MKGen}(1^\lambda)$ . Afterwards, the chip C prepares an authentication token  $T = \text{MAC}(\mathcal{K}'_{\text{MAC}}, (epk_T, D_C))$  and sends it to T, along with  $r_2$ .

6. After receiving  $r_2$  the terminal T is able to derive the session keys as well, by computing the secret keys

$$\mathcal{K}_{\text{ENC}} = \mathcal{H}_1(\mathcal{K}, r_2), \quad \mathcal{K}_{\text{MAC}} = \mathcal{H}_2(\mathcal{K}, r_2), \quad \mathcal{K}'_{\text{MAC}} = \mathcal{H}_3(\mathcal{K}, r_2).$$

Finally, the validity of the authentication token T is checked by T with  $\mathcal{K}'_{\text{MAC}}$ .

### 5.1.3. Extended Access Control

After the introduction of the Terminal Authentication and Chip Authentication protocols, we define the Extended Access Control protocol. See Figure 5.1 for an overview. The EAC scheme can be viewed as an authenticated key-exchange protocol, where the parties output  $\mathcal{K}_{\text{ENC}}, \mathcal{K}_{\text{MAC}}$  as the session key(s),

<sup>2</sup>Formally, the chip card also sends here some further data, which is irrelevant for the security of EAC as an AKE protocol.

<sup>3</sup>Recall from Section 2.2 that  $\text{DH}(g^a, b) = g^{ab}$  for group element  $g^a$  and exponent  $b$ . We overload the function and occasionally also write  $\text{DH}(g^a, g^b) = g^{ab}$ , where  $g$  is clear from the context.

the session id consists of the authenticated values  $(epk_T, pk_C, r_2, D_C)$  in the final messages, and the partner id is assumed to be empty for the chip card (see the remarks below).

#### 5.1.4. Remarks

Some remarks about the changes we made, with respect to the original protocol in [BS10] and to underlying assumptions are in order.

**SESSION AND PARTNER IDs.** We substitute the common definition of session IDs, which include the whole transcript in *sid* by a “more loose” version. In order to allow the parties not to store the transcript data for each execution — see [FL10] for solutions to this problem — we define the session IDs as the ephemeral public key of the terminal  $epk_T$ , the chip’s static key  $pk_C$ , and the nonce  $r_2$  chosen by the chip (the domain  $D_C$  is also implicitly included). This loose partnering approach, however, may allow an adversary to run a man-in-the-middle attack making the honest parties assume they communicate with someone else, even though they hold the same key.

We note that the terminal identifies the chip (i.e., its public key) as a partner in *pid*, whereas the chip outputs an empty partner ID. The latter is necessary if the adversary can register adversary-controlled terminals (as is the case in our model), because such a terminal could basically perform a man-in-the-middle attack, substituting the honest terminal’s data in the TA phase by its own. If the adversary cannot register terminals, then the chip can output the certified public key  $pk_T$  as the reliable partner ID.

**THE FINAL AUTHENTICATION STEP.** The original scheme uses the output key  $\mathcal{K}_{MAC}$  for the MAC computations (token) in the key-agreement protocol. This version, however, may not be provably secure in our security model. The reason is that with the Test-query the adversary obtains a random or the genuine session key, including  $\mathcal{K}_{MAC}$ . Then, the adversary can easily test whether this received  $\mathcal{K}_{MAC}$ , together with  $epk_T$ , matches the transmitted value. For the general analysis, we therefore suggest to derive an ephemeral MAC key  $\mathcal{K}'_{MAC}$  as  $\mathcal{K}'_{MAC} = \mathcal{H}_3(\mathcal{K}, r_2)$  and use this key for authentication. A similar strategy is used in the formal analysis of PACE [BFK09]. As mentioned in Chapter 4, as long as the messages input to the MAC scheme in EAC and

## 5. The Extended Access Control (EAC) Protocol

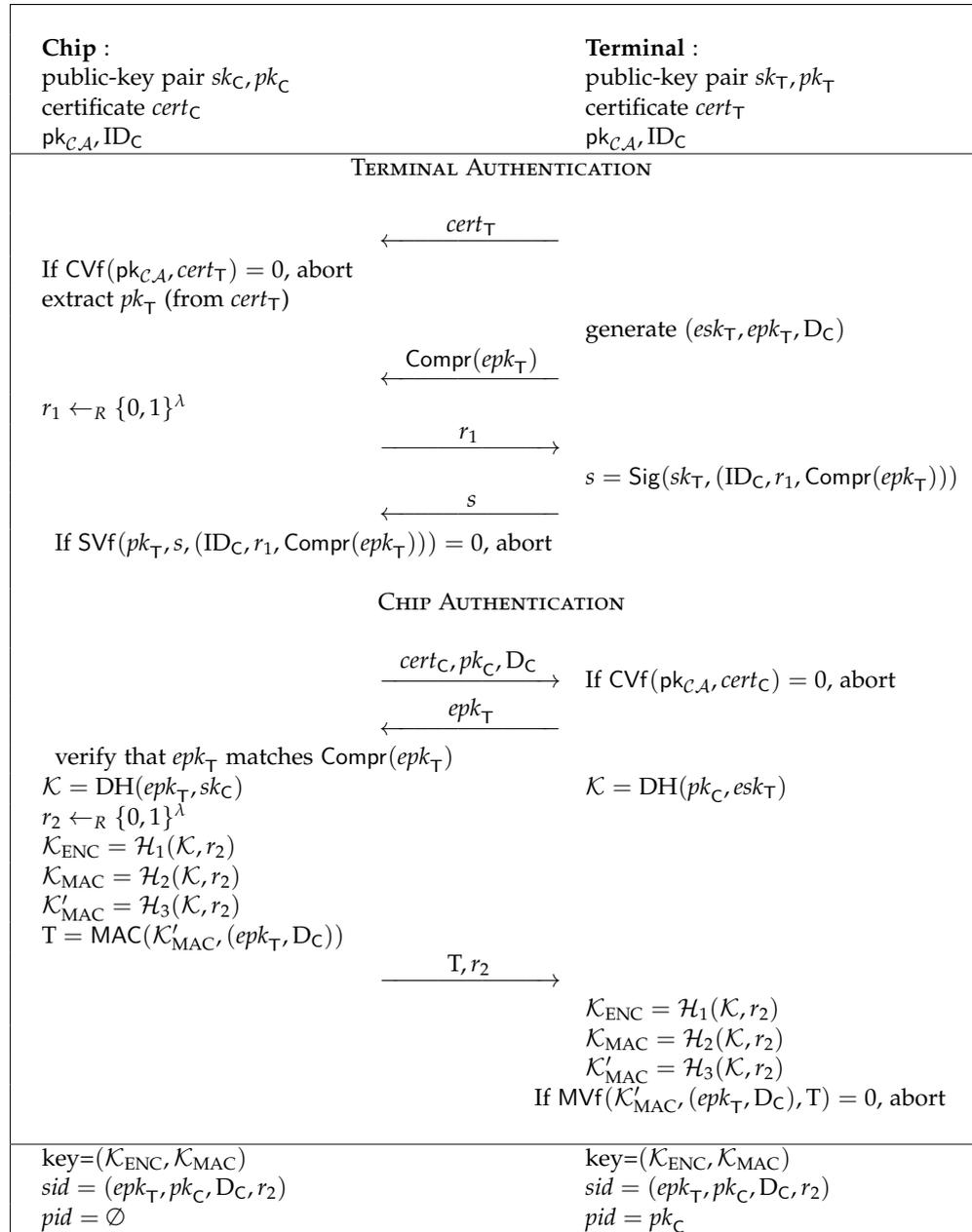


Figure 5.1.: The Extended Access Control protocol

in the subsequent secure channel have different structure — and thus, are different — the security of modified EAC carries over to the original one.

**DIFFERENT VERSIONS OF EAC PROTOCOL.** In [BSI10], the German Federal Office for Information Security (BSI) proposes two versions of EAC. Both versions give implicit authentication of the data stored on the chip. The second version additionally provides explicit authentication of the chip. This is realized by the authentication token in Steps 5 and 6 of the Chip Authentication protocol. We present and analyze the second version of EAC since it allows the composition order executing the Chip Authentication protocol at the end, and this version is currently implemented in the German electronic ID card since November 2010.

**ENCRYPTION BY PACE.** Basically, the whole communication between the chip and the terminal is secured by the session key obtained in the previously-run PACE protocol instance combined with presumably SSL/TLS. Running PACE ensures a secure communication between card reader and the chip. The communication between card reader and the terminal should be secured by using SSL/TLS. For our analysis, we do not even use these additional provisions of security means; the EAC protocol alone is strong enough.

**PASSIVE AUTHENTICATION.** In [BSI10], the BSI recommends to insert a Passive Authentication step between the Terminal Authentication and the Chip Authentication protocols. In this step the chip's certificate (data security object) issued by the document signer is transmitted to the terminal to verify the authenticity of the chip. However, Passive Authentication cannot detect cloning. Therefore, the execution of the Chip Authentication protocol is necessary. To simulate the original workflow of EAC, we substitute the Passive Authentication step by adding a certificate  $cert_C$  to the first message of the CA when the chip card sends its static public key. This reflects the essential part for AKE security.

**STATIC PUBLIC KEY OF THE CHIP CARD.** The static public key  $pk_C$  of the chip card is shared by a large set of chip cards. This gives the cardholder some kind of anonymity, as this is essentially the only information the terminal obtains about the specific chip card (resp. cardholder). Thus, revealing the static public key allows the cardholder to still remain hidden within a (sufficiently large)

group. All chip cards sharing the same static public key are essentially the same card in our analysis, since the session IDs contain the chip's static public key (and no other information about the card).

## 5.2. Security Analysis

This section gives a security analysis for the Extended Access Control protocol.

**Theorem 5.2.1** *In the random-oracle model we have*

$$\begin{aligned}
 Adv_{EAC}^{ake}(t, Q) &\leq q_e \left( Adv_{Compr, \mathcal{KG}}^{SecPre}(t) + Adv_{\mathcal{M}}^{unf-cma}(t^*, 1, q_e) + Adv_S^{unf-cma}(t^*, q_e) \right) \\
 &\quad + 2q_e^2 \cdot Adv^{GDH}(t^*, (2q_e + 1)(q_h + q_e)) + Adv_{\mathcal{CA}}^{unf-cma}(t^*, q_e) \\
 &\quad + \binom{q_e}{2} \cdot \left( \frac{1}{q} + 2^{-\lambda+1} \right),
 \end{aligned}$$

where  $\lambda$  denotes the security parameter (and bit length of nonces),  $t^* = t + O(\lambda \cdot q_e q_h \log(q_e q_h))$ ,  $q = q(\lambda)$  the group order,  $Q = (q_e, q_h)$  the number of executions and hash computations, respectively, and  $\mathcal{KG}$  the algorithm which generates an ephemeral public key for the terminal.

*Proof.* The proof of correctness, i.e., that untampered executions between honest parties yield the same accepting output, is straightforward from the correctness of the underlying primitives. Therefore, it remains to show the AKE security property.

We show security by using a common game-based approach, called game-hopping, gradually changing the original attack  $\text{GAME}_0$  (with random test bit  $b$ ) via experiments  $\text{GAME}_1; \text{GAME}_2; \dots$  to a game where the adversary's success probability to predict  $b$  is bounded by the guessing probability of  $\frac{1}{2}$ . Each transition from  $\text{GAME}_i$  to  $\text{GAME}_{i+1}$  will change the adversary's probability only slightly (depending on cryptographic assumptions); thus, in the end we prove that the success probability in the original attack cannot be significantly larger than  $\frac{1}{2}$ . (Formally, we can condition on all "bad" events ruled out in the previous games not to happen.)

**DESCRIPTION OF GAME<sub>0</sub>.** This corresponds to the original attack on the EAC protocol.

**DESCRIPTION OF GAME<sub>1</sub>.** This game works as GAME<sub>0</sub>, but aborts in the case of collisions in the compression function Compr.

We abort the experiment (declaring the adversary to lose) whenever there is an execution in which the honest terminal chooses a key  $epk_{\top}$  such that there is another execution in which an honest terminal chooses (resp. the adversary sends) a distinct ephemeral public key  $epk'_{\top}$  with the same compressed value  $\text{Compr}(epk_{\top}) = \text{Compr}(epk'_{\top})$ . This immediately yields an algorithm finding a second-preimage for Compr (running in the same time as the AKE adversary). To this end, we consider all the (at most)  $q_e$  public keys  $epk_{\top}$  of honest terminals in executions to be generated via  $\mathcal{KG}$  at the outset of the attack, and we assume that one of them is determined at random for the second-preimage attack. With a probability of  $1/q_e$  the adversary (or an honest terminal) finds a collision for this key.

Letting GAME<sub>*i*</sub> also denote the event that the adversary successfully predicts the test bit  $b$  in GAME<sub>*i*</sub> we thus have

$$\Pr[\text{GAME}_0] \leq \Pr[\text{GAME}_1] + q_e \cdot \mathbf{Adv}_{\text{Compr}, \mathcal{KG}}^{\text{SecPre}}(t).$$

**DESCRIPTION OF GAME<sub>2</sub>.** This game works as GAME<sub>1</sub>, but aborts in case of collisions among  $(epk_{\top}, r_1)$  values chosen (resp. received) by honest terminals.

We abort if there are two executions in which honest terminals receive the same value  $r_1$  (possibly chosen by the adversary) and use the same ephemeral key  $epk_{\top}$ . Since we have at most  $q_e$  executions (and thus at most  $\binom{q_e}{2}$  execution pairs), and since the ephemeral keys are chosen at random from a set of size  $q$ , we have

$$\Pr[\text{GAME}_1] \leq \Pr[\text{GAME}_2] + \binom{q_e}{2} \cdot \frac{1}{q}.$$

In particular, in combination with the assumption that the compressed ephemeral keys of different keys are distinct, no honest terminal issues a signature for the same message  $m = (\text{ID}_C, r_1, \text{Compr}(epk_{\top}))$ .

## 5. The Extended Access Control (EAC) Protocol

---

**DESCRIPTION OF GAME<sub>3</sub>.** This game works as GAME<sub>2</sub>, but aborts in case of collisions among the received compressed value  $h$  and chosen nonce  $r_1$  by honest chip cards.

We now abort if there are two executions where the honest chip cards receive the compressed value  $h$  (possibly chosen by the adversary) and use the same nonce  $r_1$ . Since we again have at most  $q_e$  executions and the values  $r_1$  are chosen at random from a set of size  $2^\lambda$  we have

$$\Pr[\text{GAME}_2] \leq \Pr[\text{GAME}_3] + \binom{q_e}{2} \cdot 2^{-\lambda}.$$

Note that there may of course be pairs  $(h, r_1)$  appearing in an execution with an honest chip and an honest terminal, e.g., in a matching conversation or if the adversary runs a man-in-the-middle attack. However, the previous steps imply that there is at most one such combination.

**DESCRIPTION OF GAME<sub>4</sub>.** This game works as GAME<sub>3</sub>, but aborts if there exists an adversary  $\mathcal{A}$  who is able to forge a valid signature on behalf of an honest terminal.

That is, assume that there exists an execution such that the honest chip card receives a valid signature for  $m = (\text{ID}_C, r_1, \text{Compr}(epk_\top))$  under the (certified) key  $pk_\top$  of an honest terminal, but such that the terminal has never signed this message before.<sup>4</sup>

Next, we show that this would yield a successful forger against the signature scheme. More precisely, we construct an adversary  $\mathcal{A}_S$  against the underlying signature scheme as follows. Adversary  $\mathcal{A}_S$  receives as input a public key  $pk$  of the signature scheme, picks a random index  $i \leftarrow \{1, 2, \dots, q_e\}$ , and runs the attack of  $\mathcal{A}$  by simulating all other parties faithfully, including the certification authority (note that there can be at most  $q_e$  active honest terminals). Only for the  $i$ -th honest terminal  $\mathcal{A}_S$  sets  $pk_\top := pk$  and calls its signature oracle whenever it is supposed to sign a message for this terminal (all other steps of this terminal follow the game description). When the adversary at some point submits a valid signature  $s^*$  for a message  $m^*$  to an honest chip under the key of our predicted terminal, where  $m^*$  has not been signed before, adversary  $\mathcal{A}_S$  stops and outputs  $m^*, s^*$ .

---

<sup>4</sup>Note that for honest terminals the pairs  $(\text{Compr}(epk_\top), r_1)$  are unique according to the previous games. Thus, it cannot happen that the terminal has issued, say,  $i$  signatures for such values, but  $i + 1$  signatures appear.

For the analysis note that the simulation here is identically distributed as  $\text{GAME}_3$ , independently of the choice  $i$  of  $\mathcal{A}_S$ . Hence, if the adversary in this game at some point forges a signature for *some* honest terminal, then  $\mathcal{A}_S$  forges a signature in the unforgeability experiment with the same probability (times a loss factor  $1/q_e$  for making the right guess):

$$\Pr[\text{GAME}_3] \leq \Pr[\text{GAME}_4] + q_e \cdot \mathbf{Adv}_S^{\text{unf-cma}}(t + O(\lambda \cdot q_e \log q_e), q_e),$$

where the running time of  $\mathcal{A}_S$  is only negligibly higher than of  $\mathcal{A}$ 's where the overhead is due to maintaining the simulation.

**DESCRIPTION OF  $\text{GAME}_5$ .** This game works as  $\text{GAME}_4$ , but aborts if the adversary in some execution submits a new long-term public key with a valid certificate such that this key has not been registered with the authority before.

In this case the adversary must forge a certificate for another public key. This can easily be turned into a successful forger against the  $\mathcal{CA}$  scheme but, as opposed to the previous case, we do not need to predict the corresponding execution. Thus,

$$\Pr[\text{GAME}_4] \leq \Pr[\text{GAME}_5] + \mathbf{Adv}_{\mathcal{CA}}^{\text{unf-cma}}(t + O(\lambda), q_e).$$

**DESCRIPTION OF  $\text{GAME}_6$ .** This game works as  $\text{GAME}_5$ , but aborts if the adversary makes a hash query for a Diffie–Hellman key computed by an honest chip, allegedly in an execution with an honest terminal.

More specifically, we abort if there exists an honest chip  $C$  that accepts in a session with  $\text{sid} = (epk_T, pk_C, r_2, D_C)$  such that a partnered  $T$  (which has sent  $epk_T$ ) is honest, and where the adversary at some point makes a query  $(\text{DH}(epk_T, sk_C), r_2)$  to the function  $\mathcal{H}_i$  for  $i \in \{1, 2, 3\}$ .

Then we show how to solve the GDH problem. That is, we build an adversary  $\mathcal{A}_{\text{gdh}}$  with access to a DDH oracle as follows. Algorithm  $\mathcal{A}_{\text{gdh}}$  gets as input  $\mathcal{G}, g$  and  $g^a, g^b$ . It initially guesses a chip index  $i \leftarrow \{1, 2, \dots, q_e\}$  and a terminal session indexed as  $j \leftarrow \{1, 2, \dots, q_e\}$ . Algorithm  $\mathcal{A}_{\text{gdh}}$  injects  $g^a$  in  $pk_C$  for the  $i$ -th chip  $C$ , and  $g^b$  into  $epk_T$  for the  $j$ -th execution with an honest terminal. It then invokes  $\mathcal{A}$ 's attack, playing all other parties, with exceptions concerning the implicit evaluations of the hash function:

- When the adversary makes a hash query  $(K, r_2)$  to the  $\mathcal{H}_i$  algorithm,  $\mathcal{A}_{\text{gdh}}$  checks if  $\text{DDH}(g^a, g^b, K) = 1$  with the help of its DDH oracle and, if so, stops with output  $K$ . Else,  $\mathcal{A}_{\text{gdh}}$  checks if there is already an entry  $(\text{HONEST}, \{A, B\}, r_2, y)$  for  $\mathcal{H}_i$  such that  $\text{DDH}(A, B, K) = 1$ . If so, it returns  $y$  to the adversary; if not, it checks if there is an entry  $(\text{MALICIOUS}, K, r_2, y)$  and returns  $y$  in this case; in any other case it picks a random  $y$  and stores  $(\text{MALICIOUS}, K, r_2, y)$  in the list for  $\mathcal{H}_i$  and hands  $y$  over to  $\mathcal{A}$ .
- When an honest party is supposed to compute a key  $K = \text{DH}(A, B)$  and to make hash queries about  $(K, r_2)$  in an execution where it knows either  $\log_g A$  or  $\log_g B$  (i.e., the part under control of the simulated honest party is not in  $\{g^a, g^b\}$ ), then it computes  $K$  as requested and returns  $y$  such that there is an entry  $(\text{HONEST}, \{A', B'\}, r_2, y)$  with  $\text{DDH}(A', B', K) = 1$ , or an entry  $(\text{MALICIOUS}, K, r_2, y)$  (and else inserts  $(\text{HONEST}, \{A, B\}, r_2, y)$  for a new random value  $y$ ). If the honest party cannot derive  $K$ , that is, its part equals  $g^a$  or  $g^b$ , then it checks the lists for entries  $(\text{HONEST}, \{A, B\}, r_2, y)$  and afterwards for entries  $(\text{MALICIOUS}, K, r_2, y)$  with  $\text{DDH}(A, B, K) = 1$  (and adds an entry to the HONEST list if there are no entries at all).

Basically, entries of the form  $(\text{HONEST}, \{A, B\}, r_2, y)$  provide a more fine-grained maintenance of hash queries and answers, and are thus consistent with the adversary's view.

It remains to argue that, if the honest user identifies an honest terminal  $T$  as the partner, then the adversary likely hits our trap, represented by the injected values (and cannot, for example, use an adversarially-chosen  $\text{epk}_T$  on behalf of  $T$  for which it can easily compute the DH key and make the hash query). It follows from the unforgeability of the signature scheme that the adversary must make a query to the random oracle about a DH key from an ephemeral key  $\text{epk}_T$  chosen by  $T$ ; otherwise, the chip would abort after checking the signature in the TA phase. Hence, we inject the values at the right execution with probability at least  $1/q_e^2$ . In this case, however, if  $\mathcal{A}$  makes such a hash query,  $\mathcal{A}_{\text{gdh}}$  has found a solution to the Gap Diffie–Hellman problem, running in time  $t + O(\lambda \cdot q_e \log q_e)$  and making at most  $(2q_e + 1) \cdot (q_h + q_e)$  DDH queries:

$$\begin{aligned} & \Pr[\text{GAME}_5] \\ & \leq \Pr[\text{GAME}_6] + q_e^2 \cdot \mathbf{Adv}^{\text{GDH}}(t + O(\lambda \cdot q_e q_h \log q_e q_h), (2q_e + 1)(q_h + q_e)). \end{aligned}$$

**DESCRIPTION OF GAME<sub>7</sub>.** This game works as GAME<sub>6</sub>, but aborts if the adversary makes a hash query for a Diffie–Hellman key computed by an honest terminal, allegedly in an execution with an honest chip.

More specifically, we abort if there exists an honest terminal T that accepts in a session with  $sid = (epk_T, pk_C, r_2, D_C)$  and  $pid = pk_C$  such that a partnered C is honest and has registered  $pk_C$ , and where the adversary at some point makes a query  $(DH(pk_C, esk_T), r_2)$  to the function  $\mathcal{H}_i$  for any  $i \in \{1, 2, 3\}$ .

The fact that this cannot significantly decrease the adversary’s success probability follows as in the previous game. Note that the adversary cannot, for example, send a false key  $pk_C^* \neq pk_C$  with a valid certificate for chip C, since the certification scheme is unforgeable, i.e.,

$$\Pr[\text{GAME}_6] \leq \Pr[\text{GAME}_7] + q_e^2 \cdot \mathbf{Adv}^{\text{GDH}}(t + O(\lambda \cdot q_e q_h \log q_e q_h), (2q_e + 1)(q_e + q_h)).$$

Note that in order to argue about key secrecy, it is insufficient merely to require that an honest chip never uses a DH key (computed with an honest terminal) known by the adversary without making a hash query (this was our strategy in the previous steps). Indeed, the adversary *could* produce the same DH key and  $r_2$ -value with the chip in an execution with a malicious terminal, then make a Reveal request for this session, and easily distinguish the key of the other execution from random. In the following games we show that this is unlikely.

**DESCRIPTION OF GAME<sub>8</sub>.** This game works as GAME<sub>7</sub>, but aborts if there are  $r_2$ -collisions for honest chip cards.

We abort if there are two executions where (possibly distinct) honest chip cards pick the same value  $r_2$ . Since we again have at most  $q_e$  executions and the values  $r_2$  are chosen uniformly at random from a set of size  $\{0, 1\}^\lambda$ , we have

$$\Pr[\text{GAME}_7] \leq \Pr[\text{GAME}_8] + \binom{q_e}{2} \cdot 2^{-\lambda}.$$

It follows that, for any accepting executions of honest chip cards, there are no identical pairs  $(K, r_2)$  from which the keys are derived. We next show that this is also true on the terminal side, assuming the unforgeability of the MAC.

## 5. The Extended Access Control (EAC) Protocol

---

**DESCRIPTION OF GAME<sub>9</sub>.** This game works as GAME<sub>8</sub>, but aborts if there are two honest parties with both accepting sessions yielding  $(K, r_2)$ , but which are not partnered.

First note that, according to GAME<sub>8</sub>, the two parties cannot be two honest chips. We next argue that, in any other case, such  $(K, r_2)$  collisions require forging a MAC. To this end, we first show that an adversarially-controlled chip must send a distinct pair  $(epk'_T, D'_C)$  with a valid MAC.

Assume that there are two honest terminals (or an honest chip and an honest terminal) with accepting executions resulting in the same pair  $(K, r_2)$ , but such that the executions have different session identifiers  $(epk_T, pk_C, r_2, D_C) \neq (epk'_T, pk'_C, r_2, D'_C)$ ; note that the partner identities  $pk_C, pk'_C$  output by terminals are determined by the session IDs. Then it must hold in particular that  $(epk_T, D_C) \neq (epk'_T, D'_C)$ . Else, since the Diffie–Hellman keys match and the certification authority verifies the well-formedness of the registered keys — and the key must be registered according to GAME<sub>5</sub> — this would imply that  $pk_C = pk'_C$ . This, however, would mean identical session and resp. partner identities. According to GAME<sub>8</sub>, however, no two sessions run by honest chip cards use the same  $(K, r_2)$  to compute the MACs for both  $(epk_T, D_C) \neq (epk'_T, D'_C)$ , implying that an adversarially-controlled chip card must create one of the valid MACs.

We finally need to argue that sending a valid MAC for a new pair  $(epk'_T, D'_C)$  contradicts the unforgeability of the MAC scheme. According to GAME<sub>7</sub> the adversary does not make a hash query about  $(K, r_2)$ . Hence, it follows that the key  $K'_{MAC}$  is an unknown random value for the adversary, which is also used by at most one honest chip card in at most one session to compute a MAC for either  $(epk_T, D_C)$  or the distinct value  $(epk'_T, D'_C)$ ; all other derived keys are based on a different  $r_2$  values and are thus independent, since they are hashed by the random oracle.

The above can be now easily used to construct a successful forger against the message authentication code  $\mathcal{M}$ . That is, simulate (perfectly) all the steps of honest parties in GAME<sub>7</sub>, but guess in advance an execution. In this execution the value  $K'_{MAC} = \mathcal{H}_3(K, r_2)$  of the chip card is not used to compute the MAC  $T$ , but one calls an external MAC-oracle instead. If there is another execution in which a (simulated) honest party, or the adversary sends a MAC for the same values  $(K, r_2)$ , then output the corresponding MAC for a value  $(epk'_T, D'_C)$  which verifies as the forgery attempt. This requires only a single call to the MAC oracle and at most  $q_e$  verification queries, and the initial guess is correct

with probability at least  $1/q_e$ . Thus,

$$\Pr[\text{GAME}_8] \leq \Pr[\text{GAME}_9] + q_e \cdot \mathbf{Adv}_{\mathcal{M}}^{\text{forge}}(t + O(\lambda), 1, q_e).$$

In the final game it follows that for any session in the *fresh* Test-query the adversary cannot make a hash query, nor a Reveal-query to an *unpartnered* honest party for the same pair  $(K, r_2)$  from which the keys are derived. Hence, in the random-oracle model the derived keys in the Test-session are indistinguishable from independent and random ones, and the adversary can only guess  $b$ .  $\square$

**HIDING THE CHIP'S DATA.** Note that, without certificates for malicious terminals, the adversary can never make a chip card reveal any of its data<sup>5</sup> in another protocol run (except for some randomness  $r_1$ ). This follows from  $\text{GAME}_4$  above, where it is shown that a chip card aborts, unless the adversary can forge signatures on behalf of an honest terminal. Since the communication in previous executions is supposed to be encrypted (via PACE and SSL/TLS), an outsider cannot obtain the chip card's data.

## 5.3. Discussion

In this section we put the security guarantees provided by protocol in the BR model into the perspective of the eCK security model of LaMacchia *et al.* [LLM07], which in turn extends the model of Canetti and Krawczyk [CK01]. We also discuss some specific steps in the EAC protocol which are necessary to ensure security in the BR model, and briefly discuss questions related to the composition of the security protocols for identity cards.

### 5.3.1. EAC and the eCK Model

**SECURITY PROPERTIES OF EAC.** The eCK model covers several security properties not captured by the BR model (cf. Section 3.4); however, as shown in [CBH05, Cre11] these models are neither formally, nor practically comparable. We cannot, indeed, prove the security of the EAC protocol in the eCK model

<sup>5</sup>By data we refer to personal data (e.g., name, date of birth, or address) and particularly sensitive information (i.e., information for biometric identification). In addition, we include the chip's certificate to what we call chip data.

directly. This is mainly because the chip card does not use ephemeral secrets (due to its limited resources) and, consequently, forward secrecy cannot be achieved without further assumptions (like, for instance, tamper-resistant hardware). We next discuss which of the three security properties — key-compromise impersonation (KCI) resilience, forward secrecy, and leakage-resilience — the EAC protocol achieves. We start with forward secrecy and leakage-resilience:

**Forward Secrecy.** It is obvious that, whenever the card discloses its long term secret key, or the terminal its ephemeral key, the adversary can easily deduce the session key. On the other hand, assuming that the hardware of the eID card is leakage-resistant, and that the terminal immediately and securely deletes ephemeral keys when they are no longer required (and that these keys cannot be leaked in the meanwhile), previous sessions cannot be attacked anymore. We also remark that, if the adversary only knows the terminal's long-term secret key, but lacks knowledge of the terminal's ephemeral key and of the card's long-term secret, then the session key of an execution between honest parties is still secret.

**Leakage-Resilience.** The card only uses two internal random strings  $r_1$  and  $r_2$ , which are later sent in clear. Leaking this information does not harm the security of the protocol, even if leaked in advance. Leaking the terminal's ephemeral key, however, does breach security. Thus, it is recommended to hedge terminals against leakage of ephemeral keys, whereby this risk of leakage can be minimized.

Next, we discuss that the EAC protocol ensures resilience against KCI. This, in particular, implies that the following attacks are infeasible:

**Leakage of Card Secrets.** Assume that the adversary gets hold of the (long-term) secret key of a chip card. Then the adversary still cannot read the user data from the card, or make the card accept in a terminal impersonation attack (without knowledge of the terminal's long-term secret), because the chip card would abort prematurely unless the signature in the TA phase is valid.

**Leakage of Terminal Secrets.** Assume now that the adversary has obtained the long-term secret key of a terminal. This still does not allow the adversary to successfully complete a protocol run with the terminal on behalf of an honest chip card. The reason is that the adversary cannot compute the derived Diffie–Hellman key (and therefore, the derived

session key) without knowledge of the card's long-term secret. It must thus forge the final message authentication code.

### 5.3.2. On the Role of the Components

**ON THE ROLE OF  $r_2$ .** We first show that without having the (public) randomness  $r_2$  in the protocol, a security proof in the BR model would be impossible, i.e., we show a successful attack in this case. More generally, assume that  $r_2$  is a random string of  $\ell$  bits only. We show an easy attack which breaks the security of a key with advantage roughly  $2^{-\ell+1}$ . Hence, for small values of  $\ell$  this probability significantly exceeds the acceptable bound.

Our attack works as follows. The adversary registers a malicious terminal  $T^*$  and then observes a session between a chip card  $C$  and an honest terminal  $T$ . Let  $epk_T$  be the terminal's ephemeral public key in this execution. The adversary now lets the card interact with  $T^*$ , where the adversary uses the same key  $epk_T$  as in the other execution; all other steps are performed using the known secret key of  $T^*$ .

Note that the card's key part is identical in both executions, and thus the derived DH keys are also identical. If the key part chosen by the card matches in the two executions – which happens with probability  $2^{-\ell}$  – then the adversary obtains the same session key as for the execution between  $C$  and  $T$  with a Reveal-query for the session between  $C$  and  $T^*$ ; it can then test the session between  $C$  and  $T$  and distinguish the obtained key from a random key with probability almost 1 (there is a small chance that the random key accidentally matches the session key, in which case the adversary cannot deduce the secret bit  $b$  correctly). In case the  $r_2$  values do not match (with probability  $1 - 2^{-\ell}$ ), the adversary outputs a random guess. This yields the claimed advantage.

**ON THE ROLE OF  $\text{Compr}(epk_T)$ .** In case of elliptic curves, [BSI10] proposes the projection  $\mathcal{P}_x$  of the point into the x-coordinate as the choice for a compression function. This function is obviously neither collision resistant nor second-preimage resistant, since each points  $P = (x, y)$  and  $-P = (x, -y)$  map to the same value. However, this is the *only* other point on the curve with the same x-coordinate. This procedure, however, is (according to Theorem 5.2.1) still secure for the function  $\text{Compr} = \mathcal{P}_x$ :

**Corollary 5.3.1** *In the random-oracle model we have for  $\text{Compr} = \mathcal{P}_x$ :*

$$\begin{aligned}
 \text{Adv}_{\text{EAC}}^{\text{ake}}(t, Q) &\leq q_e \left( \text{Adv}_{\mathcal{M}}^{\text{unf-cma}}(t^*, 1, q_e) + \text{Adv}_{\mathcal{S}}^{\text{unf-cma}}(t^*, q_e) \right) \\
 &\quad + 2q_e^2 \cdot \text{Adv}^{\text{GDH}}(t^*, 2(2q_e + 1)(q_h + q_e)) + \text{Adv}_{\mathcal{CA}}^{\text{unf-cma}}(t^*, q_e) \\
 &\quad + \binom{q_e}{2} \cdot \left( \frac{1}{q} + 2^{-\lambda+1} \right),
 \end{aligned}$$

where  $\lambda$  is the security parameter,  $t^* = t + O(\lambda \cdot q_e q_h \log(q_e q_h))$ ,  $q = q(\lambda)$  the group order, and  $Q = (q_e, q_h)$  the number of executions and hash computations, respectively.

In contrast to Theorem 5.2.1 the term  $\text{Adv}_{\text{Compr}, \mathcal{KG}}^{\text{SecPre}}$  for second preimage resistance of Compr is dropped, but consequently the number of queries to the DDH oracle in the GDH experiment doubles.

*Proof.* The proof is carried out analogously to the proof of Theorem 5.2.1; we sketch here only the game modifications.

**DESCRIPTION OF GAME<sub>1</sub>.** This game is omitted.

Games 2 to 5 remain unchanged.

**DESCRIPTION OF GAME<sub>6</sub>.** In this game we originally aimed to solve the GDH problem by inserting given values  $g^a, g^b$  in the value  $pk_C$  of a (randomly chosen) honest chip card and in the ephemeral key  $epk_T$  of a (randomly chosen) honest terminal. Via the DDH oracle we then checked if the adversary at some point queries its random oracle on the Diffie–Hellman key  $K$  for these values. In this step we used the fact that, by the unforgeability of the signature scheme, the adversary can only use previously signed values  $\text{Compr}(epk_T)$  and thus, because of the second-preimage resistance, only values  $epk_T$  generated by the honest terminal. We concluded that the adversary must, with sufficiently large probability, use the injected values  $g^a, g^b$  such that we break the GDH problem.

With a compression function like  $\mathcal{P}_x$ , which is not second-preimage resistant, the argument above is no longer valid, though. The adversary could now send a different value  $epk'_T$  instead of  $epk_T$  with  $\text{Compr}(epk'_T) = \text{Compr}(epk_T)$ , i.e.,

in our case one with identical  $x$ -coordinate. Luckily, for  $\mathcal{P}_x$  the corresponding Diffie–Hellman key  $K'$  to  $epk'_T$  would be the inverse of the key  $K$  for  $epk_T$ . Hence, we can check with two queries to the DDH oracle both possibilities,  $K$  and  $-K$ , and return the corresponding value. Put differently, we are still able to solve the GDH problem for this compression function  $\mathcal{P}_x$ .

The remaining games stay unchanged for  $\text{Compr} = \mathcal{P}_x$ . □

More abstractly, for the security of the EAC protocol we require that the compression function  $\text{Compr}$  is either second preimage resistant, or that for  $g^a, g^b$ , and DH key  $K$ , for any  $g^c$  with  $\text{Compr}(g^b) = \text{Compr}(g^c)$  and for the corresponding DH key  $K'$  of  $g^a, g^c$ , one can efficiently compute  $K$  and the number of such  $g^c$  with identical compression values is polynomial in  $\lambda$ .

### 5.3.3. On the Composition of the MRTD-Protocols

Recall that the EAC protocol is only one of the security mechanisms related to identity cards and machine-readable travel documents (see again Figure 1.1 in Chapter 1). EAC already provides strong AKE security guarantees on its own, without relying on the security of PACE and SSL/TLS. This is advantageous since an adversary might gain control of the reader, such that the communication of the EAC protocol is actually visible to the adversary.

Still, PACE, for example, has been shown to be forward-secure [BFK09]. If, in addition, the choices for the SSL/TLS protocol (negotiated in the handshake phase) also provide forward secrecy, then learning, say, the card's long-term secret key and its PIN may not immediately lead to a security breach of previous executions; such communication between card and terminal is then still protected by the forward secrecy of PACE and SSL/TLS.



## 6. Secure Messaging

In this chapter we give the description of the Secure Messaging (SM) protocol aiming for an end-to-end integrity-secure channel between a chip card and a terminal. Afterwards, we discuss its security and, in particular, analyze the composition of all protocols within the German ID card, i.e., PACE, EAC, and, subsequently, SM. We elaborate on the security of the optional Restricted Identification protocol in Chapter 7.

**READER'S ROADMAP.** In Section 6.1 we describe the Secure Messaging protocol, and prove its security as a secure channel in Section 6.2.

### 6.1. Protocol Description

The Secure Messaging protocol follows the Encrypt-then-Authenticate paradigm with the use of a counter (Send Sequence Counter, SSC), and is based on AES (resp. 3DES) and on the session keys derived by running PACE and EAC, respectively. The length of the value SSC is set according to the negotiated blockcipher (i.e., 64 Bits for 3DES and 128 Bits for AES), and the counter is incremented by each execution, both for Command and Response. According to [CK01] this implements a secure channel *if the keys are sampled uniformly from a sufficiently large enough set*.

In the following we give a short description of the SM protocol, considering only the security-relevant components. The SM protocol supports 3DES, as well as AES to encrypt and authenticate. Let  $(\text{Enc}, \text{Dec})$  and  $(\text{MAC}, \text{MVf})$  be the corresponding encryption and authentication schemes,  $\mathcal{K}_{\text{ENC}}$  and  $\mathcal{K}_{\text{MAC}}$  the session keys derived by running PACE or EAC, respectively,  $IV$  the seed of the underlying blockcipher, and let  $AUX$  (resp.  $AUX'$ ) be auxiliary data

(like header information) whose size is adapted to the blockcipher. A message  $m$  is protected by the following means.

$$\begin{aligned} C &\leftarrow \text{Enc}(\mathcal{K}_{\text{ENC}}, IV, m) \\ &\quad \text{with } IV = \text{Enc}(\mathcal{K}_{\text{ENC}}, SSC) \text{ if AES is used,} \\ &\quad \text{or } IV = 0 \text{ (3DES)} \\ T &\leftarrow \text{MAC}(\mathcal{K}_{\text{MAC}}, SSC || AUX || C || AUX') \end{aligned}$$

We assume that the message  $m$  is padded according to the used blocklength. In the case of AES, the MAC is truncated to 8 Bytes. Then, one sends  $(AUX, C, T, AUX')$ .

The receiver first checks the validity of the MAC (i.e., whether  $\text{MVf}(\mathcal{K}_{\text{MAC}}, SSC || AUX || C || AUX', T) = 1$ ) using its local counter  $SSC$ , and, afterwards, decrypts  $m \leftarrow \text{Dec}(\mathcal{K}_{\text{ENC}}, C)$ . If no error occurs, the message  $m$  is accepted (considered as authentic and unmodified), and the receiver increments its counter by 1; otherwise, the receiver aborts.

### 6.2. Security Analysis

The Encrypt-then-Authenticate paradigm for secure communication is well established and rigorously analyzed: Canetti and Krawczyk [CK01] show that the Authenticate-then Encrypt paradigm using counters implements a secure channel as long as (i) the keys are “secure”, (ii) the encryption scheme is secure against chosen-plaintext attacks, and (iii) the message authentication code is unforgeable (against adaptively chosen messages attacks). The requirements (ii) and (iii) follow immediately from the security analysis of CMAC [BR00] and CBC-Encryption for a fixed  $IV$  or  $IV = \text{Enc}(\mathcal{K}_{\text{ENC}}, SSC)$  for counter  $SSC$  [BDJR97] (for messages adjusted to blocklength, even for truncated MACs). To the best of our knowledge there is no security analysis for the case of retail-mode of the MAC suggested by 3DES. We stress that one should be careful when using 3DES in addition to a truncated MAC (see [HP04] for an overview).

In order to make a statement about the security of the composition between the secure channel and the authenticated key-exchange protocol, one can rely on the recent results of Brzuska *et al.* [BFS<sup>+</sup>13]. Roughly, the authors show that one obtains a secure channel if the key-exchange protocol is proven secure in the B(P)R model, and if one proves that the cryptographic primitives

(encryption and message authentication codes) remain secure when the keys used in these protocols are not chosen by the adversary. However, we give here an alternative approach, i.e., the direct proof, that the EAC protocol combined with Secure Messaging *does* provide the desired security, namely, it satisfies the AKE+SC security definition (cf. Section 3.3).

The security of the composition EAC+SM follows immediately from the security of EAC, because we proved the security of EAC secure without relying on the fact that the communication was secured by PACE and SSL/TLS (cf. Section 5.2). Together with the Secure Messaging protocol, one obtains a secure channel such that for subsequent eID applications or protocols, only authorized terminals have access to the data inside the chip card. Even in the ePASS scenario of the eID card, where in addition to the chip's static public key, some further data groups are transmitted in the passive authentication step, the transmission only occurs if a terminal has successfully authenticated before.

Note that even given our security result on EAC, PACE is not redundant. It still provides protection against skimming attacks, in particular, in the ePASS application, and implements a simple basic access control. In addition, as mentioned in Section 5.3.3, it provides forward secrecy for the entire protocol.

In the following we show that EAC with Secure Messaging (SM) is secure in the sense of AKE+SC (cf. Definition 3.3), if the underlying encryption scheme  $\mathcal{E}$  and the MAC scheme  $\mathcal{M}$  are secure.

**Theorem 6.2.1** *We have*

$$\mathbf{Adv}_{EAC,SM}^{ake+sc}(t, Q) \leq \mathbf{Adv}_{EAC}^{ake}(t, Q) + 2q_e \cdot \mathbf{Adv}_{\mathcal{M}}^{unf-cma}(t, \ell, \ell) + \mathbf{Adv}_{\mathcal{E}}^{ind-cpa}(t, \ell q_e),$$

where  $\ell$  denotes the maximum number of exchanged messages and  $q_e$  the number of protocol executions.

*Proof.* Again we show security in the common game-based approach of game-hopping, gradually changing the original attack  $\text{GAME}_0$  (with random test bit  $b$ ) via experiments  $\text{GAME}_1; \text{GAME}_2; \dots$  to a game where the adversary's success probability to predict  $b$  is bounded by the guessing probability of  $\frac{1}{2}$ . Each transition from  $\text{GAME}_i$  to  $\text{GAME}_{i+1}$  will change the adversary's probability only slightly (depending on cryptographic assumptions), thus showing that the success probability in the original attack cannot be significantly larger than

$\frac{1}{2}$ . (Formally, we can condition on all “bad” events ruled out in the previous games not to happen.)

Next, we describe the games.

**DESCRIPTION OF GAME<sub>0</sub>.** This corresponds to the original attack on the protocol.

**DESCRIPTION OF GAME<sub>1</sub>.** This game works as GAME<sub>0</sub>, but we replace all session keys  $\mathcal{K}_{\text{ENC}}, \mathcal{K}_{\text{MAC}}$  in all instances (and the corresponding partner instances involved in a Test-query) by random and independent keys.

We show that the adversary’s success probability decreases thereby by at most a term  $\text{Adv}_{\text{EAC}}^{\text{ake}}(t, Q)$ . Thus,  $|\Pr[\text{GAME}_0] - \Pr[\text{GAME}_1]|$  is bounded by this advantage.

We construct an adversary  $\mathcal{B}$  with black-box access to  $\mathcal{A}$  against the underlying key-exchange protocol EAC as follows. Initially, algorithm  $\mathcal{B}$  picks a value  $b^* \leftarrow_R \{0, 1\}$ , and then simulates the attack of  $\mathcal{A}$ : the steps on EAC are simulated by the oracles given to  $\mathcal{B}$ , whereas the steps of Secure Messaging are computed by  $\mathcal{B}$  itself and handed over to  $\mathcal{A}$ . With the help of its Test-oracle, algorithm  $\mathcal{B}$  provides honest users with (genuine or real) session keys. Analogously, if  $\mathcal{A}$  queries its Test-oracle, then  $\mathcal{B}$  uses previously provided keys or queries its Test-oracle for this instance to obtain the session key; for the partner (which is determined by the *pid*) no Test-query is required afterwards.

Adversary  $\mathcal{B}$  outputs  $b' = 1$  if and only if  $\mathcal{A}$  was successful (in the AKE+SC experiment with bit  $b^*$ ). In particular,  $\mathcal{B}$  can easily check, whether there has been a forgery or a replay attack, or if a correct prediction was made.

Note that for the analysis,  $\mathcal{A}$  must not ask Reveal– and Corrupt-queries such that all instances (except the ones controlled by the adversary) remain fresh. Consequently, all queries by  $\mathcal{B}$  to its Test-oracle are fresh. We have

$$\begin{aligned} \Pr[b' = b] &= \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ loses} \mid b = 0] + \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ wins} \mid b = 1] \\ &= \frac{1}{2} - \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ wins} \mid b = 0] + \frac{1}{2} \cdot \Pr[\mathcal{A} \text{ wins} \mid b = 1] \\ &= \frac{1}{2} + \frac{1}{2} (\Pr[\text{GAME}_0] - \Pr[\text{GAME}_1]) , \end{aligned}$$

and hence,

$$\Pr[\text{GAME}_0] \leq \Pr[\text{GAME}_1] + \mathbf{Adv}_{\text{EAC}}^{\text{ake}}(t, Q).$$

**DESCRIPTION OF GAME<sub>2</sub>.** This game works as GAME<sub>1</sub>, but we abort when  $\mathcal{A}$  is successful in forging, or in performing a replay-attack.

We show that if this event is triggered, one can win the unforgeability experiment against the MAC scheme. Note first that the local counter SSC maintained by the parties guarantees that a replay-attack can only occur if a forgery (of the current counter number) occurs. For this reason, it suffices to restrict our analysis to the case that such a forgery *does* occur.

One can easily transform an adversary  $\mathcal{A}$  against the unforgeability property of the AKE+SC protocol into an adversary  $\mathcal{B}$  against the unforgeability of the MAC. First,  $\mathcal{B}$  must guess one of the (maximum)  $2q_e$  user instances and then ask the external MAC- and MVf-oracles to simulate every computation on the key  $\mathcal{K}_{\text{MAC}}$  as  $\text{MAC}(\mathcal{K}_{\text{MAC}}, \cdot)$ . Now, if the adversary successfully outputs a forgery in AKE+SC, then in particular the MAC sent in the protocol must be valid, and neither generated, nor sent by an honest participant before. In total, one requires at most  $\ell$  queries to the MAC- and MVf-oracles; thus it holds that,

$$\Pr[\text{GAME}_1] \leq \Pr[\text{GAME}_2] + 2q_e \cdot \mathbf{Adv}_{\mathcal{M}}^{\text{unf-cma}}(t, \ell, \ell).$$

**DESCRIPTION OF GAME<sub>3</sub>.** This game works as GAME<sub>2</sub>, but we replace all encryptions in instances from the Test-queries by encryptions of  $0 \dots 0$ . The indistinguishability property of the encryption scheme implies that through this replacement the probability from GAME<sub>2</sub> to GAME<sub>3</sub> decreases by a factor of at most  $\mathbf{Adv}_{\mathcal{E}}^{\text{ind-cpa}}(t, \ell q_e)$  because there are at most  $q_e$  test sessions, and each session consists of at most  $\ell$  exchanged messages. Hence, we have,

$$\Pr[\text{GAME}_2] \leq \Pr[\text{GAME}_3] + \mathbf{Adv}_{\mathcal{E}}^{\text{ind-cpa}}(t, \ell q_e).$$

At this point the Test-oracle acts independently of the bit  $b$ , and, thus, the adversary has a success probability of outputting the correct bit of at most  $1/2$ .  $\square$

Finally, note that as remarked in the security model for secure channels (cf. Section 3.3), the local counter SSC prevents that cryptograms are received and accepted by parties in a unintended order.



## 7. Restricted Identification

The German electronic identity card allows the card owner to identify to a service under a pseudonym. This is accomplished via the Restricted Identification (RI) protocol proposed by the German Federal Office for Information Security (BSI) in [BSI10], and it is intended, especially, for usage in authentication over the Internet. The Restricted Identification protocol is optional, supposed to be executed after the secure channel is built through Secure Messaging and the application or requested service. Note that, even though the chip card is authenticated to a service provider by means of its certified public key, this key is still shared among a large group of chip cards; thus, the chip card is not uniquely identified.

For this reason, Restricted Identification introduces pseudonyms generated based on the owner's identity and the service provider's public domain value. In order to be used in online authentication (with respect to privacy and security) these pseudonyms have to comply with the following requirements:

- One desired feature of online authentication is the ability of recognizing a card. The card holder should be able to maintain a state and history for a specific service. Hence, a terminal within a domain (e.g. which is associated to a service provider), is able to profile cards (resp. users). This property is called *domain-specific linkability*. Moreover, we require that the pseudonym under which a chip card authenticates to a terminal is unique (in order to prevent Sybil attacks<sup>1</sup> and identity transfer).
- However, pseudonyms should only be linkable within a domain. Two service providers (associated to different domains) should be unable to link interactions of the same user. We call this property *cross-domain anonymity*.
- For privacy reasons, full disclosure of the user's identity is unacceptable. A card holder should not reveal more information than necessary for the specific service. For instance, an age verification should only reveal

---

<sup>1</sup>In Sybil attacks, a malicious party can illicitly acquire many pseudonyms. In our scenario, a malicious card could identify to the same terminal using many different pseudonyms.

the age or even merely that the owner is above 18 years old. As such, the authentication process should be privacy-friendly.

BSI addressed all of the aforementioned requirements when they designed the Restricted Identification (RI) protocol for the German electronic identity card. Roughly, pseudonyms output by RI are derived using only the unique secret key  $sk_{ID}$  of a card and the certified domain-specific public key  $dpk$  of a service provider. More precisely, the card holder sends the hash of the Diffie-Hellman value  $DH(dpk, sk_{ID})$  to the terminal. The terminal then checks if the pseudonym  $dsnym = \mathcal{H}(DH(dpk, sk_{ID}))$  is in the blacklist of this sector (for revocation reasons).<sup>2</sup>

**ANALYZING THE RI PROTOCOL.** We first provide the formal security model, capturing cross-domain anonymity. We then show that under the DDH assumption, the Restricted Identification protocol is cross-domain anonymous. Finally, we comment on the domain-specific linkability.

**RELATED WORK.** In [KKKK12], Kutyłowski *et al.* analyze the Restricted Identification protocol and show unlinkability under a variant of the DH problem, called the Linking Diffie-Hellman problem. Unfortunately, the authors do not provide a formal definition of unlinkability. We bridge this definitional gap, and introduce the notion of cross-domain anonymity capturing the desired privacy requirement for eID cards.

**READER'S ROADMAP.** In Section 7.1 we describe the Restricted Identification protocol; and its security analysis follows in Section 7.2.

### 7.1. Protocol Description

The Restricted Identification protocol allows a card to generate pseudonyms specific to a domain. Roughly, this is accomplished by performing a one-sided Diffie-Hellman exchange (i.e., only the terminal sends its public key part), and

---

<sup>2</sup>To be precise, the pseudonym is computed as  $dsnym = \mathcal{H}(DH(dpk, sk_{ID}), D)$  where  $D$  denotes certified domain parameters sent by the terminal. These domain parameters are distinct from the domain parameters used in PACE and EAC, which are provided by the chip card. For simplicity, we assume that  $D$  is implicitly included in  $dpk$ .

the pseudonym is essentially defined by the hash of the corresponding DH value. Since a malicious card could send any value to the terminal, pretending that this value is the respective pseudonym, the RI protocol must be executed after Chip Authentication and within Secure Messaging, to ensure that the chip card is trusted and honest. In addition, the trustworthiness of the public domain key is ensured since Terminal Authentication is performed before including the step where the terminal certificate (containing the public domain key) is sent. Hence, the interaction within the Restricted Identification protocol is performed within the secure channel using the key material offered by the EAC protocol.

Computing the domain-specific pseudonym, merely requires a hash function and a DH value computation. As in the case of EAC, RI is based on a Public Key Infrastructure (PKI) including a Document Verifier and a Country Verifying Certification Authority (CVCA). We assume that the public domain keys  $dpk$  are certified by a certification authority  $\mathcal{CA}$ . We assume that the keys  $(sk_{\mathcal{CA}}, pk_{\mathcal{CA}})$  of the  $\mathcal{CA}$  are generated at the outset and that  $pk_{\mathcal{CA}}$  is securely distributed to all parties (including the adversary).

We also assume that the chip identifier  $pk_{ID}$  and the corresponding secret key  $sk_{ID}$  are unique. In the specification of RI [BSI10], the exact generation of these keys is not described. We assume that there exists an unknown generator  $IdGen$  generating key pairs for chip cards. Note, only the secret key  $sk_{ID}$  is stored on the chip card. This condition is required since the specification demands that the chip card does not know its public identifier  $pk_{ID}$ . Also, the terminal does not know the corresponding secret key (if any exists) to its public domain key  $dpk$  generated by an unknown generator  $DGen$ . These keys are externally used to generate revocation lists (i.e., blacklists), and only the required keys are implemented on the card and, respectively, sent to the service providers. Blacklists are obtained by the Document Verifier.

In the Restricted Identification protocol the terminal T and the chip C perform the following steps:

1. T sends its certified public domain key  $dpk$  to C.<sup>3</sup>
2. C is able to verify the validity of the public domain key  $dpk$  using  $pk_{\mathcal{CA}}$ . The chip C computes the domain-specific pseudonym as:

$$dsnym = \mathcal{H}(\text{DH}(dpk, sk_{ID})) ,$$

---

<sup>3</sup>As mentioned before, for simplicity reasons, we omit the domain parameters  $D$  here. This does not affect our security result on the original RI protocol.

- and sends  $dsnym$  over to T.
3. T checks whether the received domain-specific pseudonym  $dsnym$  is listed in the blacklist. If so, T rejects and C is not identified.

### 7.2. Security Analysis

In this section we look at the security of the RI protocol. We first give a formal definition of cross-domain anonymity. We define the model with respect to the protocol at hand instead of giving a modular and generic model. Nonetheless, our model can easily be generalized to work on other protocols of the same kind as RI. Afterwards, we prove in this model that RI is indeed cross-domain anonymous. Finally, we look at the domain-specific linkability of RI.

#### 7.2.1. Cross-Domain Anonymity

**THE SECURITY MODEL.** Cross-domain anonymity protects against linking pseudonyms across different domains — within one domain pseudonyms are supposed to be linkable. We define cross-domain anonymity via a game between the adversary and a left-or-right oracle which, given two public chip identifiers and a domain, generates a domain-specific pseudonym for either the left or the right chip (identifier) in the domain, according to a secret random bit  $b$ . We allow the adversary to make adaptive calls to this left-or-right oracle. The adversary's goal is to predict  $b$  significantly beyond the pure guessing probability (condition (a) below).

In addition to challenge queries to the left-or-right oracle, the adversary may decide to: blacklist domain-specific pseudonyms, ask for domain-specific pseudonyms via the pseudonym-generating oracle PG, or corrupt users. The pseudonym-generating oracle PG takes as input a chip identifier  $pk_{ID}$  and a domain public key  $dpk$ , and outputs the corresponding domain-specific pseudonym  $dsnym$  under domain  $dpk$ . To exclude trivial attacks, we must take into account that domain-specific pseudonyms are in principle linkable by the domain holder. Hence, in “transitivity” attacks where the adversary asks the left-or-right oracle first about a domain-specific pseudonym for chip identifiers  $pk_{ID}^0, pk_{ID}^1$  and then for  $pk_{ID}^0, pk_{ID}^2$  for the same domain, but  $pk_{ID}^1 \neq pk_{ID}^2$ , the pseudonyms would point to the same chip identifier if and only if the oracle uses the left pseudonym. We thus exclude such queries in condition (b) below.

We require another case to be excluded. Namely, the generation of pseudonyms through PG immediately tells the adversary the connection between the chip identifier and the resulting domain-specific pseudonym. In other words, if an adversary queries on input one of the tested identifiers and the tested domain, the output will trivially enable to win. However, the adversary should be able to query PG on other inputs, e.g., with input one of the tested cards, but a different domain. This is captured in condition (c) below.

Finally, since we allow adaptive corruptions of chip cards, these cards may not be asked to the left-or-right oracle at any time (condition (d) below). Knowing the chip's secrets immediately allows to compute any domain-specific pseudonym for this chip card. Comparing computed pseudonyms with the output of the left-or-right oracle can reveal  $b$  trivially.

The model below assumes, to the advantage of the adversary, that all public chip identifiers and all domain keys are known by the adversary at the outset; while Restricted Identification provides terminals only its respective domain key, we additionally give the adversary the public chip identifiers. As usual, we measure the adversary's running time including also all steps of honest parties.

In the definition, we assume that domain-specific pseudonyms are unique within a domain; this is satisfied by definition in Restricted Identification since chip identifiers  $pk_{ID}$  are unique and hence, we have domain-specific uniqueness.<sup>4</sup>

In the following definition,  $ID$  denotes the list of public chip identifiers  $pk_{ID}$ ,  $D$  the list of domain keys,  $B$  the blacklist containing domain-specific pseudonyms,  $P$  the list of generated pseudonyms,  $C$  the corrupted chip cards, and  $LR$  denotes the set of queries made to the left-or-right oracle.

**Definition 7.2.1 (Cross-Domain Anonymity)** *The Restricted Identification protocol is  $(n, d, t, Q, \epsilon)$ -cross-domain anonymous with  $Q = (q_c, q_p, q_t)$  if for any algorithm  $\mathcal{A}$  running in time  $t$  and making at most  $q_c$  queries to the corruption oracle,  $q_p$  queries to the pseudonym-generating oracle, and  $q_t$  queries to the left-or-right oracle, the probability that the following experiment returns 1 is at most  $\epsilon$ :*

**Experiment**  $CD - Anon_{\mathcal{A}}(\lambda, n, d)$

---

<sup>4</sup>This might not be true given that potentially the hash function collides on two different inputs. However, if the underlying hash function is collision resistant, the uniqueness assumption is reasonable (i.e., collisions occur extremely rarely).

## 7. Restricted Identification

$b \xleftarrow{\$} \{0, 1\}$   
 Set  $ID, D, B, C, P, LR = \emptyset$   
 $(pk_{ID}, sk_{ID})_{1, \dots, n} \leftarrow_R \text{IdGen}(1^\lambda)$  // no collisions among  $pk_{ID}$   
 $(dpk)_{1, \dots, d} \leftarrow_R \text{DGen}(1^\lambda)$  // no collisions among  $dpk$   
 $ID = (pk_{ID})_{1, \dots, n}; D = (dpk)_{1, \dots, d}$   
 $d \leftarrow \mathcal{A}^{\text{Corrupt}, B', \text{PG}, \text{LoR}}(ID, D)$   
 Return 1 iff  
 (a)  $d = b$ ,  
 (b)  $\forall (\{pk_{ID}^0, pk_{ID}^1\}, dpk), (\{pk_{ID}^2, pk_{ID}^3\}, dpk) \in LR$ , we have  
 $(\{pk_{ID}^0, pk_{ID}^1\} = \{pk_{ID}^2, pk_{ID}^3\}) \vee (\{pk_{ID}^0, pk_{ID}^1\} \cap \{pk_{ID}^2, pk_{ID}^3\} = \emptyset)$ ,  
 (c)  $\forall (pk_{ID}, dpk) \in P \nexists pk'_{ID}: (\{pk_{ID}, pk'_{ID}\}, dpk) \in LR$ , and  
 (d)  $\forall pk_{ID} \in C \nexists pk'_{ID}, dpk: (\{pk_{ID}, pk'_{ID}\}, dpk) \in LR$ .

If  $\mathcal{A}$  queries  $\text{Corrupt}(pk_{ID})$   
 on input  $pk_{ID} \in ID$ :  
 – set  $C \leftarrow C \cup \{pk_{ID}\}$   
 – return corresponding  $sk_{ID}$

If  $\mathcal{A}$  queries  $\text{PG}(pk_{ID}, dpk)$   
 on input  $pk_{ID} \in ID \setminus B$  and  $dpk \in D$ :  
 – set  $P \leftarrow P \cup \{(pk_{ID}, dpk)\}$   
 – return  $dsnym = \mathcal{H}(\text{DH}(dpk, pk_{ID}))$

If  $\mathcal{A}$  queries  $B'(pk_{ID}, dpk)$   
 on input  $pk_{ID} \in ID$  and  $dpk \in D$ :  
 – set  $B \leftarrow B \cup \{dsnym\}$   
 with  $dsnym = \mathcal{H}(\text{DH}(dpk, pk_{ID}))$

If  $\mathcal{A}$  queries  $\text{LoR}(pk_{ID}^0, pk_{ID}^1, dpk)$   
 on input  $pk_{ID}^0, pk_{ID}^1 \in ID \setminus B$  and  $dpk \in D$ :  
 – set  $LR \leftarrow LR \cup \{(\{pk_{ID}^0, pk_{ID}^1\}, dpk)\}$   
 – return  $dsnym_b = \mathcal{H}(\text{DH}(dpk, pk_{ID}^b))$

The probability is taken over all coin tosses of  $\text{IdGen}$ ,  $\text{DGen}$ , and  $\mathcal{A}$ , and over the choice of  $b$ .

**THE SECURITY PROOF.** Informally, the protocol is cross-domain anonymous because domain-specific pseudonyms appear to the adversary to be random under the decisional Diffie–Hellman assumption (cf. Definition 2.2.3). Below we write  $t' \approx t$  to denote the fact that  $t'$  is essentially the same as  $t$ , except for some minor administrative overhead.

**Theorem 7.2.2** *Assume the DDH problem is  $(t, \epsilon)$ -hard. Then, the Restricted Identification protocol is  $(n, d, t', Q, \epsilon')$ -cross-domain anonymous with  $Q = (q_c, q_p, q_t)$ , where  $\epsilon' \leq nd\epsilon$  and  $t' \approx t$ .*

*Proof.* In order to proof security we use the ideas of pseudorandom synthesizers of Naor and Reingold [NR97]. Assume the original attack of the adversary on our protocol. In a first game hop we replace the actual DH

values  $\text{DH}(pk_{ID}, \text{dpk})$  for the computation of the pseudonyms generated in PG and resp. LoR queries by random group elements (but in a consistent way). That is, whenever we are supposed to output  $\text{dsnym}$  for some fresh identity with public key  $pk_{ID}$  or a fresh domain  $\text{dpk}$  we instead output  $\text{dsnym}'$ , which is the hash of a new random group element  $g' \in \mathcal{G}$ . Else, if  $pk_{ID}$  has been used before in combination with  $\text{dpk}$ , either in PG request or an LoR-query, we reuse the previously generated random value  $\text{dsnym}'$ .

We claim that this hop cannot noticeably increase the adversary's success probability, by the DDH assumption. To this end, we briefly recall the notion of a pseudorandom synthesizer in [NR97]. Pseudorandom synthesizers for DH pairs are  $a \times b$  matrices, with the rows labeled by values  $g^{x_i}$  and the columns labeled by  $g^{r_j}$ , and with entries at position  $i, j$  set to  $g^{x_i r_j}$ . One such is indistinguishable from an  $a \times b$  matrix of independent and random group elements, even if the row and column labels  $g^{x_i}$  and  $g^{r_j}$  are given. In a sense, the matrix entries are correlated, but still look random. As discussed in [NR97] this holds in our case under the DDH assumption. In fact, it allows for a reduction to the DDH problem with a loss of a factor  $ab$  where, in our case, after the initial corruption, there are at most  $ab \leq nd$  entries of honest users.

Note that if an adversary is unable to notice when the actual DH values  $\text{DH}(pk_{ID}, \text{dpk})$  for computing pseudonyms are replaced by random group elements, then this also holds when the adversary only receives the hash of these elements. This in particular means that the security of the hash function does not affect the cross-domain anonymity. Kutyłowski *et al.* [KKKK12] also observe that the hash function does not strengthen the security of the protocol.

In the next game hop, we always use the left public chip identifier  $pk_{ID}^0$  in LoR queries, independently of the value of  $b$ . We stress that, in case  $b = 1$ , this does not change the adversary's success probability at all. Assume from now on that  $b = 0$ . Note that each LoR-query with input  $(pk_{ID}^0, pk_{ID}^1, \text{dpk})$  is answered by a random element, just as it would be for  $b = 1$ . All other LoR-queries involving  $\text{dpk}$  can only be about the same pair  $(pk_{ID}^0, pk_{ID}^1)$  in this order, in reverse order  $(pk_{ID}^1, pk_{ID}^0)$ , or for distinct entries. In the first case, we would answer again consistently with the (wrong) random element, in the second case, we would switch to the other random element, and in the third case use an independent random value. This behavior, however, is identical to the case  $b = 1$  from the adversary's point of view. Similarly, the adversary cannot generate any pseudonym by querying PG for  $(pk_{ID}^0, \text{dpk})$  nor  $(pk_{ID}^1, \text{dpk})$  without losing (this is the case captured by condition (c)). It

follows that such pseudonym generations do not depend on the bit  $b$ . Hence, the probability of the experiment returning 1 does not change.

Hence, the adversary's success probability is independent of  $b$ , and the adversary cannot win with probability more than  $\frac{1}{2}$ . Collecting all probabilities from the game hops yields the claimed bound. Note that the adversary cannot use the oracles to its advantage, as any chip identifier input to the oracles cannot be queried to the left-or-right oracle.  $\square$

**REMARK.** Theorem 7.2.2 shows that under the DDH assumption, the Restricted Identification protocol is cross-domain anonymous. However, the protocol is analyzed in the standalone model, meaning that the security statements hold only if the protocol is executed alone; i.e., not before, after, or during other protocols. In fact, we have to look at the concrete eID scenario where RI is executed in the secure channel right after the execution of EAC.

We observe that during the Chip Authentication phase, the chip sends information about itself in form of a public key  $pk_C$ . Service providers from different domains could link exactly this key and, consequently, link the pseudonyms, as well. This implies that cross-domain anonymity of RI is lost when applied after the Chip Authentication protocol. For this reason, the BSI proposed to use group keys for the chip's public key  $pk_C$ . That is, several chip cards share the same public key, and, thus, the chip cards identity is hidden in a group of chip cards. Certainly, the size of these groups need to be large enough, since otherwise, the probability to link the correct chip cards is not negligibly small. It is, however, always possible, to exclude a connection of cards, if the public keys differ. This issue relaxes the hardness of cross-domain anonymity for RI when used in practice. Large groups for public keys provide sufficient "obfuscation", though.

Note also that if all chip cards share the same public key and an adversary manages to disclose the secret key by breaking into the hardware, then all identity cards must be revoked. This is because the certified public key is the only information about an eID card which appears during the authentication and can be used for revocation. Hence, if all cards share the same public key, no card can authenticate after suspicion of such leakage. Putting it differently, the size of the group which share the same public key is a tradeoff between security and privacy. The exact size of groups is not stated in the specification [BSI10]. Recently, in [HKKK12], Hanzlik *et al.* built an authenti-

cation mechanism based on Chip Authentication and Restricted Identification without the need of group keys.

### 7.2.2. Domain-Specific Linkability

Domain-specific linkability requires that terminals recognize identity cards within a domain without learning any information about the card holder. At the same time no chip card can use multiple pseudonyms for the same domain. We argue in the following why Restricted Identification achieves this property.

Restricted Identification implements exactly one secret chip identifier  $sk_{ID}$  into the chip card. In addition, this identifier is unique among all chip cards already distributed, still within the generation phase, or even in future identity cards. Hence, a domain-specific pseudonym for a chip card in a domain  $dpk$  is unique as long as the hash function is collision-resistant. The Diffie–Hellman values  $DH(dpk, sk_{ID})$  for a static domain  $dpk$  are distinct if the underlying group element is a generator and all  $sk_{ID}$ 's are distinct.

As mentioned, there is only one secret chip identifier embedded in the chip. This would imply that the chip card can only generate one pseudonym per each domain. However, the chip cards do not need to authenticate the transmitted pseudonym towards the terminal. In order to trust the pseudonym sent by a card, the card must have successfully executed the Chip Authentication protocol. Similar arguments hold for the public domain key sent by the terminal. Hence, the Restricted Identification must be executed only after the EAC protocol and during Secure Messaging, in order to guarantee the authenticity of the messages within RI.

Note that in the EAC protocol the chip card must send its public key  $pk_C$  in the Chip Authentication phase. As argued in the case of cross-domain anonymity, many chip cards have to share the same keys. This is also implemented in the German identity cards as described in the specification [BSI10].<sup>5</sup>

To this end, the cards possess each one unique secret identifier which yields one unique domain-specific pseudonym. Thus, the terminal always recognizes a card. We still need to show that no information about the card leaks during an execution of the Restricted Identification protocol.

---

<sup>5</sup>To be precise, privileged terminals also have access to an chip-individual key. The definition of “privilege” can be found in the specifications [BSI10].

## 7. Restricted Identification

---

Computing domain-specific pseudonyms merely require the secret chip identifier, which is hidden in the value  $DH(dpk, sk_{ID})$ . In fact, this value is hidden in an even stronger way since any adversary would need to compute first the exact preimage of the pseudonym with respect to the used hash function  $\mathcal{H}$ , before attempting to break the CDH assumption. Thus, the card does not reveal any information other than the domain-specific pseudonym, which moreover looks random with respect to pseudonyms on other domains (due to cross-domain anonymity).

The above arguments give strong evidence that Restricted Identification preserves domain-specific linkability in a privacy-friendly way.

## 8. On the Choice of Cryptographic Primitives

In the previous chapters we analyzed the cryptographic protocols within the German electronic identity card. The security results hold under number-theoretic assumptions, and independently of the specific instantiations of cryptographic primitives used in the protocols. We merely require that these schemes (like the blockcipher or digital signature scheme) are secure in their respective security model. As the number-theoretic assumptions we use are established and known to hold for decades, we rather investigate whether the choice of primitives according to the specification [BSI10] preserves security allowing us to conclude that the cryptographic protocols run by the German electronic identity card represent a strongly secure, composed set of protocols from a cryptographic point of view.

**READER'S ROADMAP.** In Section 8.1 we summarize the necessary requirements for the cryptographic primitives and whether they are attained by the instantiations suggested in the specifications [BSI10]. This would ensure the secure usage of the German eID card for authentication. Finally, we comment on further requirements, beyond cryptography, for the security of the German eID card.

### 8.1. Cryptographic Primitives

The protocols used in the German eID card use several building blocks (cryptographic primitives), including hash function, compression function, digital signature schemes, etc. Depending on the role they play, different properties may be required: for example, RI requires the underlying hash function to be collision-resistant (for domain-specific linkability), whereas PACE and EAC requires an ideal hash function behaving as a random oracle. We recall the

requirements for the respective primitives and whether they are met by the chosen instantiations of primitives as proposed in the specifications.

**Hash Functions.** All protocols, PACE, EAC and RI, use hash functions. For instance, PACE and EAC employ hash functions to deduce the session keys from an intermediate value, unknown and nonproducible, for an adversary. The RI protocol outputs domain-specific pseudonyms by using a hash function. The deployed hash functions must thus provide at least collision resistance, i.e., it is computationally hard to find two preimages which map to the same hash value. In fact, PACE and EAC require an even stronger requirement, i.e., the hash function behaves as a random oracle. Formally, we define the advantage of an adversary  $\mathcal{A}$  to find a collision for the hash function  $\mathcal{H}$  by  $\text{Adv}_{\mathcal{H}}^{\text{coll}}(\mathcal{A})$ ; for our purposes, we require that  $\text{Adv}_{\mathcal{H}}^{\text{coll}}(\mathcal{A}) \approx 0$ .

According to the specifications, SHA-1 or SHA-2 (with all parameter choices) must be used as hash functions. SHA-1 and SHA-2 are standardized in NIST FIPS PUB 180-3 [NIS]. SHA-1 is deployed in many embedded systems due to its efficiency. However, trust in SHA-1 has been diminished since several works cryptanalyzed round-reduced SHA-1 [WYY05, Mano8, AS09]. Nonetheless, there is no concrete, efficient algorithm implementing an attack against the collision resistance of SHA-1. In fact, as shown in [Sat05], a computer system worth \$10 million still requires 127 days to find real collision.<sup>1</sup> SHA-2 (even if Keccak [BDPA11] as SHA-3 is announced and standardized by NIST in 2012) is currently widely deployed and assumed to be secure. Current state-of-the-art cryptanalysis on (the round-reduced) SHA-2 can be found in [GLRW10, BLMN11, LIS12]. Due to its stronger collision-resistance, SHA-2 is a favored choice when using the German eID card. We note that none of these schemes are proven to behave as a random oracle, and in fact, one can distinguish those schemes from a random oracle (due to extension attacks [CDMP05]). However, this does not yield an immediate attack but tells one that in order to attack the protocol this “difference” to underlying hash function must be exploited.

**Compression Functions.** In EAC a compression function is applied to the ephemeral public key  $pk_{\top}$  of the terminal. This function should be quasi-injective, i.e., it is either injective or second-preimage resistant, or, alternatively, all decompressed ephemeral keys are related to the

---

<sup>1</sup>Even though, this observation was made eight years ago, still no significant improvement to break the collision resistance of SHA-1 exists. Nonetheless, the increasing CPU power has to be taken into account when assessing the security against state-of-the-art computation devices.

DH problem as described in Section 5.3 (e.g., by the projection into the x-coordinate of a point in an elliptic curve).

According to the specifications, if the underlying group is a finite field, SHA-1 is used to compress the public keys and is sufficiently preimage-resistant since as argued above collision resistance holds, and thus implies preimage resistance. In Section 5.3 we have shown that the projection to the x-coordinate for subgroups in an elliptic curve is still secure for the compression function.

**Digital Signatures.** In EAC, the terminal signs, along with other inputs, its compressed ephemeral DH value, together with a nonce chosen by the chip card. Through the signature, the terminal in some sense shows that it knows the corresponding static secret key to the public verification key.<sup>2</sup> Hence, the signature ensures a proof of validity for the terminal. The signature scheme must be unforgeable, i.e, no outsider must be able to produce a signature on behalf of another terminal. Formally, in the security proofs, we denote the advantage of an adversary in forging a signature for the signature scheme  $\mathcal{S}$  by  $\text{Adv}_{\mathcal{S}}^{\text{unf-cma}}(t, Q)$ . We demand that  $\text{Adv}_{\mathcal{S}}^{\text{unf-cma}}(t, Q) \approx 0$  for polynomial-time adversaries.

BSI suggest the use of RSA-PSS [RSA02] as specified in RFC 3447 [JK03] to implement the signature scheme. Many works [Jono1, Coro2, CM09] show the security of RSA-PSS, and, thus meeting the above requirement. We note that the original scheme PSS proposed by Bellare and Rogaway [BR93] is significantly different than the one in the specifications [RSA02]. Nonetheless, both signature schemes are unforgeable against chosen-message attacks in the random-oracle model and under a computational assumption known as the RSA assumption.

**Certificates.** Chip cards, as well as terminals, possess certificates, which are issued by the certification authority  $\mathcal{CA}$  and which unequivocally (and permanently) bind the static public key (together with some data irrelevant wrt. security) to the respective owner. The terminals, in addition, possess authorization certificates saying which information from a chip card they are allowed to read. Such certificates should be issued only by the certification authority, and any later modification on a certificate invalidates it. Also, we assume that revocation works in a secure way; this process is out of scope in this thesis. Formally, in the security proofs, we denote the probability for an adversary to output a valid certificate

<sup>2</sup>In fact, the terminal shows that it can generate signatures on behalf of the identity the public key is associated to. If the signature scheme is unforgeable, it roughly means that it must have known the underlying secret key in order to come up with a valid signature on a “fresh” message.

for an arbitrary static public key and a terminal by  $\text{Adv}_{\mathcal{CA}}^{\text{unf-cma}}(t, Q)$ . We demand that  $\text{Adv}_{\mathcal{CA}}^{\text{unf-cma}}(t, Q) \approx 0$  for polynomial-time adversaries.

The German eID card uses card certificates in the card verifiable certificate format (CVC) standardized as part of ISO/IEC 7816 [ISO04a, ISO04b, ISO05a]. Since it is an international standard for electronic identity cards with contacts, this certification scheme should have been verified with respect to their security.

**Message Authentication Codes.** In both PACE and EAC, authorization tokens are sent, confirming the knowledge of the same session key. For this purpose, message authentication codes (MAC) are used; these are typically implemented by blockciphers or hash functions. We require that an adversary against a MAC can produce a valid tag for a new message with only negligible probability (even if it may receive tags for chosen messages). Formally, in the security proofs, this advantage is denoted by  $\text{Adv}_{\mathcal{M}}^{\text{unf-cma}}(t, Q)$ . We demand that  $\text{Adv}_{\mathcal{M}}^{\text{unf-cma}}(t, Q) \approx 0$  for polynomial-time adversaries.

The German eID card deploys two versions of MACs. If 3DES authentication is chosen, the chosen MAC scheme is that of algorithm 3 in ISO/IEC 9797-1 [ISO99] (with blockcipher *DES* and  $IV = 0$ ). This scheme is also known as ANSI retail MAC [ANS86], and is run in a special CBC-MAC mode. This CBC mode has been widely scrutinized, and many documents can help avoid potential loopholes in implementations. Highly optimized and well-trusted implementations of CBC-MAC have been around for years. CBC-MACs in general are known to be provably secure under certain assumptions [BDJR97, BKRo0, PRo0]. The exact MAC algorithm used in the German eID card is subject to the attacks in [PvO99, Mit03, CKMo0]; however, these attacks remain rather theoretical and cannot be applied in reasonable time. In particular, the specification enforces the use of serial numbers, which hampers forgery attacks.

If the AES mode is used, AES in CMAC-mode as specified in NIST SP 800-38B [NISo7] is implemented. CMAC was invented by Black and Rogaway in [BRo0], and is provably secure [BRo0, IKo3, JJVo2] if the underlying blockcipher acts as a random permutation. AES is included in the ISO/IEC 18033-3 standard [ISO05b] and, similarly to 3DES, it is widely deployed, optimized, and subject to much cryptanalysis [RSVCo9, BKNo9, BKo9, BKR11]. Nonetheless, all known attacks are either too expensive (in running time) or only work under assumptions that are unrealistic in practice.

**Encryption Schemes.** In PACE the first message is an encrypted nonce. This

nonce must be transferred confidentially to the card reader or terminal, respectively. This is accomplished by using a symmetric encryption scheme which only decrypts correctly if the card reader enters the correct PIN. For this reason, the encryption scheme must guarantee that an adversary without knowledge of the PIN can neither decrypt correctly an encrypted nonce, nor even differentiate it from a different encrypted nonce. Furthermore, the encryption scheme should be collision resistant, i.e., there is no ciphertext  $z$ , such that for two possible PINs the ciphertext  $z$  decrypts to the same value  $s$ . The first two properties are captured by the term  $\text{Adv}_{\mathcal{E}}^{\text{ind-cpa}}(\mathcal{A})$ , and we demand that  $\text{Adv}_{\mathcal{E}}^{\text{ind-cpa}}(\mathcal{A}) \approx 0$  for any polynomial-time adversary  $\mathcal{A}$ .

In the German eID card, the PACE protocol implementation employs the encryption scheme standardized in ISO/IEC 10116 [ISO06] in CBC-mode. The blockcipher is assumed to be secure and cryptanalysis on this standard remains impractical [PY04, Mit05]. The ind-cpa security of the scheme is shown in [BDJR97]; however, it is unknown whether the encryption scheme behaves as an ideal cipher, a fact assumed in the security proof of PACE [BFK09]. We refer to [Rog11] for a summary of more sophisticated attacks on blockciphers, similar to the one used in PACE.

**Random Number Generation.** The random number generator used to generate nonces or, respectively, as input in the run of cryptographic operations must produce (pseudo-)random values.

BSI recommends the pseudo-random number generators of the class DRG.3, DGR.4, PTG.2, PTG.3, and NTG.1 from [KS11a]. See also [SK02] for some evaluation criteria for these generators.

## 8.2. Further Security Requirements

Besides the cryptographic security of the primitives, we further require additional security properties of the German electronic identity card. Indeed, the underlying hardware must effectively protect the sensitive data and the secret keys of the card owner. Moreover, since online authentication via the German eID card requires a PIN entered by the user, there should be sufficient entropy when choosing this PIN. For instance, taking one's birthday as a PIN is certainly not sufficiently strong for a password. In the following, we summarize several further requirements.

### 8.2.1. Hardware Security

**Security against Unauthorized Read-Out.** The hardware of the German eID card must prevent reading-out the static secret key. Otherwise, the security of the cryptographic protocols implemented on it cannot be ensured. In particular, an adversary might be able to access the content of secured communication of previous sessions if given access to the card (thus contradicting the forward secrecy). We demand similar hardware-security for the terminal. We refer to possible consequences of hardware vulnerabilities in the discussion on forward secrecy in Sections 5.3.1 and 5.3.3.

**Side-Channel Resistance.** The hardware must be side-channel resistant, i.e., the German eID card must be protected against adversaries who attack the actual implementation of cryptographic protocols, rather than aim to break the mathematical structures and security. Such adversaries learn sensitive information by timing attacks [Koc96], (differential) power analysis attacks [KJJ99], or by electromagnetic radiation [GMO01, QSo1]. Our cryptographic analysis implicitly assumes side-channel resistance; however, this issue should be explicitly studied for the German eID card. Side-channel analysis on the (German) electronic passport (which has a architecture similar to the eID card) can be found in [KOP11] which says that side-channel analysis does not provide sufficient gain for an adversary.

**Secure Erasure (Terminal).** The terminal must securely delete the secret ephemeral key used in the EAC protocol immediately after the computation of the session key (particularly, it has to securely remove it from memory). If this is not done, then the forward secrecy of the protocols is exposed.

**Cryptographic Building Blocks.** Several cryptographic building blocks are implemented on the RFID-chip in the German eID card. These must be correctly implemented and they must be provably secure in the sense of the previously mentioned cryptographic security requirements.

### 8.2.2. Personalization

It is assumed that every eID card is assigned to exactly one person. For this reason, a personalization procedure is applied. While the secure handling of chip cards and their corresponding secrets are quite important, we also list some further necessary security requirements below.

**The Choice of the PIN.** The owner of the card must choose, among other things, the password (PIN) for the online authentication. It is assumed that the owner selects a random 6-digit string as a PIN. In particular, one should not choose obvious combinations (for instance, his or her birthday) which would make the search for the correct PIN easier for a potential adversary. In addition, this password should be neither re-used in unauthorized environments, nor visible to third parties. Under these conditions, the entropy of the PIN is maximized.

**Nondisclosure of the PIN.** The owner of the card is sworn to secrecy of the password, and should disclose the PIN only to dedicated bodies, such as border control authorized staff.

**Chip Card Manufacture.** During the manufacture of the card, at a specific time, a static secret key is placed into the non-readable memory in the German eID card. This procedure must not leak any information about the secret key. This includes the generation of the keys.



## **Part III.**

# **Future Enhancements**



## 9. The PACE|AA Protocol for MRTD, and its Security

In Chapter 4, we introduced the Password Authenticated Connection Establishment (PACE) protocol [BSI10], which secures the contactless communication between machine-readable travel documents (including identity cards), and readers. Roughly, the protocol generates a secure Diffie–Hellman key out of a low-entropy password, which the owner of the passport has to enter into the reader, or which is transmitted through a read-out of the machine-readable zone. The Diffie–Hellman key is subsequently used to secure the communication. In Chapter 4, we presented the PACE protocol and discussed the results of previous security analyses on PACE. In [BFK09], it has been shown that the PACE protocol achieves the widely accepted security notion of password-based authenticated key agreement by Bellare, Pointcheval, and Rogaway [BPR00], in its strong form given by Abdalla *et al.* [AFP05]. This holds under a variant of the Diffie–Hellman assumption, assuming secure cryptographic building blocks, and idealizing the underlying block cipher and hash function.

According to the specifications for the German eID card, the PACE protocol should be followed by the Extended Access Control (EAC) authentication steps, called Terminal Authentication (TA) and Chip Authentication (CA), run with high-entropy, certified keys. This should ensure that access for either party is granted based on strong cryptographic keys (i.e., not relying on low-entropy passwords only). In the specifications of the ICAO 9303 standard [ICA06] for the border control scenario, the normative document about machine-readable travel documents, however, only a passive passport authentication is mandatory, where the passport essentially sends merely its (authenticated) data. Active Authentication (AA) of the passport, implemented through a signature-based challenge-response protocol, is only optional. If AA is not enforced, this potentially allows an attacker to bypass authentication by cloning valid passports. Even if AA is used, then the (plain) challenge-response protocol introduces a potential threat to privacy, as discussed in [BSI10] (see

also [BPSVo8b, BPSVo8a, MVV07]). Namely, if the terminal can encode a time stamp or the location into the challenge, then the signature on that challenge can be used as a proof towards third parties about the location or time of the border check. In this sense, the passport cannot deny this interaction. This problem has been explicitly addressed in the European Chip Authentication protocol (where a message authentication code for a shared key is used for the challenge-response step instead).

**COMBINING PACE AND AA.** We argue that, on the chip's side, we can re-use some of the (secret) data in the PACE step for the AA step, thus saving the exponentiation for the signature in AA on the chip's side, getting Active Authentication (almost) for free.

To understand our technique, one needs to take a closer look at the PACE protocol. The PACE protocol first maps the short password to a random group element through an interactive sub-protocol Map2Point, followed by a Diffie–Hellman key exchange step for this group element, and concludes with an authentication step. While the latter steps are somewhat canonical, the Map2Point step can be instantiated by different means, and it allows a modular design. The most common instantiations rely on another Diffie–Hellman step (run internally in the German identity card), or on hashing into elliptic curves as proposed by Icart [Ica09] and Brier *et al.* [BCI<sup>+</sup>10]. The security proof for PACE [BFK09] holds for general Map2Point protocols satisfying some basic security properties.

Our improvement works for the Diffie–Hellman-based Map2Point protocol as implemented on the German identity cards. One reason is that the chip can re-use the secret exponent it uses in the Diffie–Hellman step of the Map2Point protocol. We discuss two alternatives how to carry out the AA step with this exponent more efficiently, one based on DSA signatures and the other one using Schnorr signatures. We note that the idea applies more generally to other discrete-log-based signature schemes. The challenge in the new AA step is now the authentication data sent by the terminal in the PACE step.

**SECURITY OF THE COMBINED PROTOCOL.** Whenever secret data is used throughout several sub-protocols, great care must be taken not to spoil the security of the overall protocol. We thus show that sharing the data between the PACE protocol and the new AA sub-protocol preserves the desirable security properties. More precisely, we show that:

- 
- In the combined PACE|AA protocol we still achieve the security of a password-based authenticated key-exchange protocol (and thus showing that the deployment of the randomness in the extra AA step does not violate the security of the PACE protocol), and
  - the overall protocol still authenticates the chip securely (in a high-entropy sense), even when many concurrent executions of PACE|AA take place. To this end, we define a strong security model for authentication, essentially only excluding trivial attacks, e.g., if the adversary gets possession of the secret key, or simply relays information in executions.<sup>1</sup>

It follows that the PACE|AA protocol achieves the previous security standards of the individual protocols, but comes with a clear efficiency improvement. We note that the underlying assumptions are essentially the same as for PACE and for AA, i.e., besides the common assumptions about secure encryption, digital signatures, and MAC algorithms, we reduce the security of the combined protocol to the security of PACE (as an authenticated key-exchange protocol) and to a variant of the security of Schnorr and resp. DSA signatures (where the adversary now also gets access to a decisional Diffie–Hellman oracle and can decide which message should be signed after seeing the first half of the signature).

**A DENIABLE SCHNORR VERSION.** As explained before, for privacy reasons it may be important that the terminal cannot derive from the interaction with the passport or identity card, a proof that convinces others *that* an interaction took place. Put differently, the protocol should provide *deniable authentication* [DDNoo]. In other words, a user must be able to argue that the terminal could have generated its view of the protocol run by itself from the public data, without communicating with the passport. Thus, the passport holder can deny that any actual interaction occurred and claim that the terminal has fabricated this conversation.

We note that the previously discussed signature-based protocols do not support deniability. The reason is that the terminal cannot create a signature on behalf of the passport (i.e., under its public key) without the signing key — or without communicating with the actual chip. For the (ordinary) AA variant, the terminal is allowed to encode *any* information into the challenge; in our improved combinations, namely PACE|AA, the challenge is “only” a MAC

---

<sup>1</sup>Relaying of information can be prevented by distance-bounding protocols, but there is a question as to whether these can be securely implemented in practice (for more information on distance-bounding, we refer the reader to [BC93, Ger10, DFKO11, FO13]).

computed over data provided by the passport and the shared Diffie–Hellman key. Whether this allows to encode information or not depends on the MAC.

Our proposed deniable variant does not rely on Schnorr signatures, but rather (in some sense) on the interactive Schnorr identification scheme for honestly chosen challenges. This identification scheme is deniable because one can simulate the interaction via a well-known zero-knowledge simulator.<sup>2</sup> Interestingly, our variant is essentially as efficient as the signature based one, but comes with the advantage of deniability.

**READER’S ROADMAP.** Section 9.1 presents the PACE|AA protocol (variants). In Section 9.2 we discuss the security model for authenticated key exchange and impersonation resistance. In Section 9.3 we discuss the relevant (number-theoretic and cryptographic) security assumptions, before we present our security results. Finally, in Section 9.4 we discuss our deniable version and its security.

## 9.1. Protocol Description

Figure 9.1 illustrates the PACE|AA protocol with both authentication options (DSA- and resp. Schnorr signature-based authentication) at the end. The scheme itself uses a block cipher  $\mathcal{C}(K_\pi, \cdot) : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$  and a hash function  $\mathcal{H}$ , with values  $1, 2, \dots$  in fixed-length encoding prepended to make evaluations somewhat independent.

The chip already holds a certificate  $cert_C$  for its public key  $pk_C$  under the authority’s public key  $pk_{CA}$ , and (authenticated) group parameters  $\mathcal{G} = (a, b, p, q, g, \lambda)$  describing a subgroup of order  $q$ , generated by  $g$ , of an elliptic curve for parameters  $a, b, p$  and security parameter  $\lambda$ . Then the parties run the PACE protocol, with the chip sending a nonce encrypted under the password, running the Diffie–Hellman based Map2Point protocol to derive another generator  $\hat{g}$  on which another Diffie–Hellman key exchange is then performed. In this Map2Point step, the chip uses a secret exponent  $x_C$ , chosen

---

<sup>2</sup>This property, however, is not known to work for the DSA case; this is why we restrict ourself to the Schnorr scheme. Note also that Schnorr signatures are also somewhat simulatable, but only if one programs the random oracle hash function; this, however, is not admissible for the notion of deniability. We nonetheless still use a hash function in the solution but use programmability only to prove the unforgeability/impersonation-resistance property, not deniability.

A :	PACE	B :
password $\pi$ secret $sk_C$ , public $pk_C = g^{sk_C}$ certificate $cert_C$ for $pk_C$ under $pk_{CA}$ authenticated group parameters $\mathcal{G} = (a, b, p, q, g, \lambda)$		password $\pi$  $pk_{CA}$
$K_\pi = \mathcal{H}(0  \pi)$ choose $s \leftarrow \{0, 1\}^\ell \subseteq \mathbb{Z}_q$ $z = \text{Enc}(K_\pi, s)$	$\xrightarrow{\mathcal{G}, z}$	$K_\pi = \mathcal{H}(0  \pi)$  abort if $\mathcal{G}$ incorrect $s = \text{Dec}(K_\pi, z)$ choose $x_T \leftarrow \mathbb{Z}_q^*$ $X_T = g^{x_T}$
choose $x_C \leftarrow \mathbb{Z}_q^*$ $X_C = g^{x_C}$	$\xleftarrow{X_T}$	
abort if $X_T \notin \langle g \rangle \setminus \{1\}$	$\xrightarrow{X_C}$	abort if $X_C \notin \langle g \rangle \setminus \{1\}$ $h = X_C^{x_T}$ $\hat{g} = h \cdot g^s$ choose $y_T \leftarrow \mathbb{Z}_q^*$ $Y_T = \hat{g}^{y_T}$
$h = X_T^{x_C}$ $\hat{g} = h \cdot g^s$ choose $y_C \leftarrow \mathbb{Z}_q^*$ $Y_C = \hat{g}^{y_C}$	$\xleftarrow{Y_T}$	
check that $Y_T \neq X_T$ $\mathcal{K} = (Y_T)^{y_C}$ $\mathcal{K}_{\text{ENC}} = \mathcal{H}(1  \mathcal{K})$ $\mathcal{K}'_{\text{SC}} = \mathcal{H}(2  \mathcal{K})$ $\mathcal{K}_{\text{MAC}} = \mathcal{H}(3  \mathcal{K})$ $\mathcal{K}'_{\text{MAC}} = \mathcal{H}(4  \mathcal{K})$ $T_A = \text{MAC}(\mathcal{K}'_{\text{MAC}}, (Y_T, \mathcal{G}))$	$\xrightarrow{Y_C}$	check that $Y_C \neq X_C$ $\mathcal{K} = (Y_C)^{y_T}$ $\mathcal{K}_{\text{ENC}} = \mathcal{H}(1  \mathcal{K})$ $\mathcal{K}'_{\text{SC}} = \mathcal{H}(2  \mathcal{K})$ $\mathcal{K}_{\text{MAC}} = \mathcal{H}(3  \mathcal{K})$ $\mathcal{K}'_{\text{MAC}} = \mathcal{H}(4  \mathcal{K})$ $T_B = \text{MAC}(\mathcal{K}'_{\text{MAC}}, (Y_C, \mathcal{G}))$
	$\xleftarrow{T_B}$	
abort if $T_B$ invalid	$\xrightarrow{T_A}$	abort if $T_A$ invalid
..... Version: Schnorr Signature .....		
$\sigma = x_C + \mathcal{H}(5  X_C, T_B) \cdot sk_C$	$\xrightarrow{\text{Send}(\mathcal{K}'_{\text{SC}}, (\sigma, cert_C))}$	recover and validate certificate abort if $g^\sigma \neq X_C pk_C^{\mathcal{H}(5  X_C, T_B)}$
..... Version: DSA Signature .....		
$r = X_C \bmod q$ $\sigma = x_C^{-1}(\mathcal{H}(5  T_B) + rsk_C)$	$\xrightarrow{\text{Send}(\mathcal{K}'_{\text{SC}}, (\sigma, cert_C))}$	recover and validate certificate $w = \sigma^{-1}$ $r = X_C$ $v = g^{w\mathcal{H}(5  T_B)} \cdot pk_C^{r w}$ abort if $v \neq X_C$
key= $(\mathcal{K}_{\text{ENC}}, \mathcal{K}_{\text{MAC}})$ sid = $(Y_C, Y_T, \mathcal{G})$ pid = $cert_C$		key= $(\mathcal{K}_{\text{ENC}}, \mathcal{K}_{\text{MAC}})$ sid = $(Y_C, Y_T, \mathcal{G})$ pid = $cert_C$

 Figure 9.1.: The PACE|AA protocol (all operations are modulo  $q$ )

at random, to send  $X_C = g^{x_C}$ . At this point the terminal has an ephemeral public key  $X_C$  for the exponent  $x_C$  of the card. The parties in the PACE protocol finally exchange message authentication codes  $T_A, T_B$ .

Roughly, the idea is now to re-use the secret exponent  $x_C$  in the Map2Point sub protocol on the chip's side for the signature generation, and then use the authentication value  $T_B$  of the terminal as the challenge on which the signature is computed. The chip then sends its certificate (along with the missing signature part) over the secure channel via a Send command, using the key  $\mathcal{K}'_{SC}$  derived from the Diffie–Hellman exchange. The reader may think for now of the secure channel as an authenticated encryption; however, we note that other channel instantiations work as well.

**INSTANTIATIONS.** There are essentially two possible instantiations of the PACE|AA protocol, depending on the chosen method of authentication. One is based on Schnorr signatures [Sch90] where the chip uses the values  $x_C$  and  $X_C$  as the (private resp. public) randomness and  $T_B$  as the challenge for creating the signature under the long-term signature key  $pk_C$ . We call this option *Active Authentication via Schnorr signatures*. Alternatively, the chip card might prove its authenticity by using DSA signatures, where again  $x_C$  and  $X_C$  are used as the randomness for the signature generation [Kra95]. This version is called *Active Authentication via DSA signatures*. We note that the computation of the final signatures requires only modular multiplications (and, in the case of DSA, an inversion) instead of exponentiations.

### 9.2. Security Model Adaptations

We use the real-or-random security model of Abdalla *et al.* [AFP05] which extends the model of Bellare *et al.* [BPR00] for password-based key-exchange protocols (cf. Section 3.2). Some changes are necessary, though, because we now incorporate a long-term signing key on the chip. These minor modifications follow next.

**ATTACK MODEL.** We consider the same attack model and winning condition as described in Section 3.2. In addition, the adversary can also win if it successfully manages to complete the chip authentication phase on behalf of an honest chip. Although using the same framework for such attacks we

define this additional attack scenario as an extra security property (namely impersonation resistance), because this step involves cryptographically strong keys whereas the password-based phase relies on low-entropy secrets only.

Similarly to the BR model from Section 3.2, the adversary can access user instances via oracles, basically providing the interface of the protocol instance (via the usual Send, Execute, Reveal, and Test commands, which send messages to parties, allows the observation of executions between honest parties, reveal the session key and resp. triggers a challenge for a test key). In addition, there exists a Corrupt-oracle in the model from [AFP05]. The adversary can gain control over a user during the execution by issuing a Corrupt-query, thus obtaining the secrets of an honest party. For the sake of convenience, we split these queries here into Corrupt.pw and Corrupt.key queries, where the former reveals the password only and the latter discloses the long-term key only (in case of a chip); in both cases, the other secret remains private. Note that these queries fully cover all Corrupt queries (since we work in the weak corruption model where the parties' internal states are not revealed upon corruption). An honest party becomes adversary-controlled if it does not have any secrets left (i.e., if the adversary issues both Corrupt-query types for a chip, or the Corrupt.pw-query for the terminal).

The adversary can make the following queries to the interface oracles other than those from [AFP05] or Section 3.2.1, respectively:

Corrupt.pw( $U$ ). The adversary obtains party  $U$ 's password  $\pi$ .

Corrupt.key( $U$ ). The adversary obtains party  $U$ 's cryptographic key  $sk$  (if it exists).

In addition, since the original PACE protocol was cast in the random-oracle and ideal-cipher models, where a random hash function oracle and an encryption/decryption oracle are available, the attacker may also query these oracles here. (We note that we only use the ideal cipher implicitly, through the reduction to the security to PACE.)

**PARTNERS, CORRECTNESS AND FRESHNESS.** Upon successful termination, we assume that an instance  $U_i$  outputs a session key  $k$ , the session ID  $sid$ , and a user ID  $pid$  identifying the intended partner (assumed to be empty in PACE for anonymity reasons, but containing the chip's certificate in the combined PACE|AA protocol). We note that the session ID usually contains the entire transcript of the communication, but, for efficiency reasons, in PACE it only contains a part thereof. This is inherited here. We say that instances  $U_i$  and

$U'_j$  are *partnered* if both instances have terminated in accepting state with the same output. In this case, the instance  $U_i$  is called a partner to  $U'_j$  and vice versa. Any untampered execution between honest users should be partnered and, in particular, the users should end up with the same key (this correctness requirement ensures the minimal functional requirement of a key agreement protocol).

Neglecting forward secrecy for a moment, an instance  $(U, i)$  is called *fresh* at the end of the execution if there has been no  $\text{Reveal}(U, i)$ -query at any point, no  $\text{Reveal}(U', j)$ -query where  $U'_j$  is a partner to  $U_i$ , and no corruptions (i.e., neither kind of  $\text{Corrupt}$ -query has been issued); else, the instance is called *unfresh*. In other words, fresh executions require that the session key has not been leaked (by either partner) and that no  $\text{Corrupt}$ -query took place.

To capture forward secrecy we refine the notion of freshness and further demand from a fresh instance  $(U, i)$  that the session key has not been leaked through a  $\text{Reveal}$ -query, and that for each  $\text{Corrupt.pw}(U)$ - or  $\text{Corrupt.key}(U)$ -query there has been no subsequent  $\text{Test}(U, i)$ -query involving  $U$ , or, if so, then there has been no  $\text{Send}(U, i, m)$ -query for this instance at any point.<sup>3</sup> In this case we call the instance *fs-fresh*, else *fs-unfresh*. The notion of forward secrecy captures the idea that it should not help if the adversary corrupts some party after the test query, and that even if corruptions do take place before test queries, then executions between honest users are still protected (before or after a  $\text{Test}$ -query).

Formally, AKE security is defined analogously as in Section 3.2.1, with the updated definition of freshness. Next, we define impersonation resistance.

**IMPERSONATION RESISTANCE.** This security notion stipulates that an adversary *successfully impersonates* an honest card, if an honest reader accepts in some session with session id  $sid$  and partner identity  $pid$ , but such that (a) either the intended partner  $U$  in  $pid$  is not adversary-controlled or the public key in  $pid$  has not been registered; (b) no  $\text{Corrupt.key}$  command to  $U$  has been issued before the reader has accepted, and (c) the session id  $sid$  has not appeared in any other accepting session. This roughly means that a adversary successfully manages to impersonate an honest chip or to make the reader

---

<sup>3</sup>In a stronger notion the adversary may even issue a  $\text{Corrupt.key}$  command for the user before the testing; however, due to the entanglement of the PACE and the AA protocols here our protocol does not achieve this, though.

accept a fake certificate, without knowing the long-term secret and without relaying data in a trivial man-in-the-middle attack.

Define now the IKE advantage (I stands for impersonation) of an adversary  $\mathcal{A}$  for a key agreement protocol  $P$  by

$$\begin{aligned} \mathbf{Adv}_P^{\text{ike}}(\mathcal{A}) &:= \Pr[\mathcal{A} \text{ successfully impersonates}] \\ \mathbf{Adv}_P^{\text{ike}}(t, Q) &:= \max \left\{ \mathbf{Adv}_P^{\text{ike}}(\mathcal{A}) \mid \mathcal{A} \text{ is } (t, Q)\text{-bounded} \right\} \end{aligned}$$

Note that we do not need to define a forward-secret version of impersonation resistance, as this attack only makes sense for future sessions.

### 9.3. Security Analysis

In this section, we discuss the security of the PACE|AA protocol when active authentication is done via Schnorr signatures; the case of DSA signatures follows analogously, because we do not use any specific properties of the underlying signature scheme (except for robust unforgeability). That is, we assume that the chip, holding public key  $pk_C = g^{sk_C}$  with certificate  $cert_C$ , signs the message  $X_T$  with key  $sk_C$  and randomness  $X_C$ . The signature is given by  $\sigma = x_C + c \cdot sk_C \pmod{q}$  for  $c = \mathcal{H}(5||X_C, T_B)$ . After the final authentication step of PACE, the chip sends (using an already secure channel) the values  $\sigma$  and  $cert_C$  to the reader, which verifies the signatures and the certificate (and aborts in case at least one of the verifications fails).

As noted in [BFK09], using the derived keys already in the key agreement step does not allow for a proof in the Bellare-Pointcheval-Rogaway model. We hence also use a variant such that the keys  $\mathcal{K}'_{SC}$  and  $\mathcal{K}'_{MAC}$  are independent from the keys output as the result of the key agreement.

#### 9.3.1. Security Assumptions

Our PACE|AA protocol makes use of the Schnorr and DSA signature scheme. For the Schnorr signature based solution we rely on the following version of unforgeability (cf. Definition 9.3.1) which (a) allows access to a decisional DH (DDH) oracle for the forger, and (b) considers access to a signer in an online/offline fashion, in the sense that the adversary may ask to see the public randomness part first before deciding on a message to be signed. Still, the goal

is to create a signature on a new message for which the signing has not been completed. We note that the proof in [PSoo] for Schnorr signatures still holds, assuming that computing discrete-logarithms relative to a DDH-oracle is hard. In particular, the hardness of this “gap discrete-log problem” is implied by the hardness of GDH. We call this security notion *robust unforgeability*, as it should still hold in presence of the DDH oracle and the delayed message choice.

**Definition 9.3.1 (Robust Unforgeability of Schnorr Signatures)** *The Schnorr signature scheme (relative to an instance generator  $\mathcal{I}$ ) is  $(t, Q, \varepsilon)$ -robustly-unforgeable with  $Q = (q_R, q_{ddh})$  if for any adversary  $\mathcal{A}$  running in total time  $t$ , making at most  $q_{ddh}$  DDH oracle queries and at most  $q_R$  init-queries to oracle  $\mathcal{O}$ , the probability that the following experiment returns 1 is at most  $\varepsilon$ :*

$  \begin{aligned}  (\mathcal{G}, g) &\leftarrow \mathcal{I}(\lambda) \\  \text{pick } sk &\leftarrow \mathbb{Z}_q \text{ and let } pk = g^{sk} \\  (m^*, \sigma^*) &\leftarrow \mathcal{A}^{\mathcal{O}(sk, \cdot), DDH}(\mathcal{G}, g, pk) \\  \text{parse } (c^*, s^*) &\leftarrow \sigma^* \\  \text{output 1 iff} \\  &c^* = \mathcal{H}(g^{s^*} pk^{c^*}, m^*) \\  &\text{and } m^* \notin M  \end{aligned}  $	$  \begin{aligned}  &\text{Set } id = 0 \text{ and } R, M = \emptyset. \\  &\text{If } \mathcal{A} \text{ queries } \mathcal{O}(sk, \text{init}), \\  &\quad \text{pick } r \leftarrow \mathbb{Z}_q, \\  &\quad \text{set } id = id + 1, \\  &\quad \text{add } (id, r) \text{ to } R, \\  &\quad \text{return } (id, g^r). \\  &\text{If } \mathcal{A} \text{ queries } \mathcal{O}(sk, (\text{complete}, id, m)), \\  &\quad \text{if } (id, r) \in R \text{ for some } r, \\  &\quad \quad \text{update } R \leftarrow R \setminus \{(id, r)\} \\  &\quad \quad \text{add } m \text{ to } M, \\  &\quad \quad \text{return } r + \mathcal{H}(g^r, m) \cdot sk \pmod q; \\  &\quad \text{else, return } \perp. \\  &\text{If } \mathcal{A} \text{ queries } DDH(X, Y, Z), \\  &\quad \text{return 1 iff } DH(X, Y) = Z.  \end{aligned}  $
---	---

We let  $\text{Adv}_{\text{Schnorr}}^{r\text{-forge}}(t, Q)$  be the maximal advantage for any adversary running in time  $t$ , making in total  $Q = (q_R, q_{ddh})$  queries.

As it turns out to be useful for the deniable version of our protocol, we remark that the proof of Pointcheval and Stern [PSoo] holds as long as the input to the hash oracle in the forgery is new, i.e., one can extract the discrete-logarithm of the public key even if the hash function in signature requests is evaluated on quasi unique inputs, and the forgery, too, uses a previously unqueried hash function input. For the notion of signature unforgeability this holds because each signature request uses a high-entropic random group element and the message  $m^*$  in the forgery cannot have been signed before. We take advantage of this fact for the deniable version of our protocol, where we insert  $(\mathcal{Y}_C, \mathcal{G})$

instead of  $(R, m)$  into the hash function for the random group element  $Y_C$  chosen by the chip card and respectively the signer. We also show that for the proof of impersonation resistance the adversary cannot re-use one of these values  $(Y_C, \mathcal{G})$  but needs to pick a new value  $Y_C$ ; thus implying the second property under the assumption of robust unforgeability.

For the DSA based solution we require an analogous assumption:

**Definition 9.3.2 (Robust Unforgeability of DSA Signatures)** *The DSA signature scheme (relative to an instance generator  $\mathcal{I}$ ) is  $(t, Q, \varepsilon)$ -robustly-unforgeable with  $Q = (q_r, q_{ddh})$  if for any adversary  $\mathcal{A}$  running in total time  $t$ , making at most  $q_{ddh}$  DDH oracle queries and at most  $q_r$  init queries to oracle  $\mathcal{O}$ , the probability that the following experiment returns 1 is at most  $\varepsilon$ :*

$(\mathcal{G}, g) \leftarrow \mathcal{I}(\lambda)$   
 pick  $sk \leftarrow \mathbb{Z}_q$  and let  $pk = g^{sk}$   
 $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(sk, \cdot), DDH}(\mathcal{G}, g, pk)$   
 parse  $(c^*, s^*) \leftarrow \sigma^*$   
 output 1 iff  
 $r^* = g^{w\mathcal{H}(m^*)} pk^{wr^*} \pmod q$   
 for  $w = (s^*)^{-1} \pmod q$ ,  
 and  $m^* \notin M$

Set  $id = 0$  and  $R, M = \emptyset$ .  
 If  $\mathcal{A}$  queries  $\mathcal{O}(sk, init)$ ,  
 pick  $k \leftarrow \mathbb{Z}_q$ ,  
 set  $id = id + 1$ ,  
 add  $(id, k)$  to  $R_L$ ,  
 return  $(id, k)$ .  
 If  $\mathcal{A}$  queries  $\mathcal{O}(sk, (complete, id, m))$ ,  
 if  $(id, k) \in R$  for some  $k$ ,  
 update  $R \leftarrow R \setminus \{(id, k)\}$   
 add  $m$  to  $M$ ,  
 return  $k^{-1}(\mathcal{H}(m) + g^k sk) \pmod q$ ;  
 else, return  $\perp$ .  
 If  $\mathcal{A}$  queries  $DDH(X, Y, Z)$ ,  
 return 1 iff  $DH(X, Y) = Z$ .

We let  $\mathbf{Adv}_{\text{DSA}}^{r\text{-forge}}(t, Q)$  be the maximal advantage for any adversary running in time  $t$ , making in total  $Q = (q_r, q_{ddh})$  queries.

It is currently not known if DSA signatures are still secure in the robust sense; likewise, it is not known whether DSA can be proven unforgeable in the usual sense of the notion. For an overview about the (limited) security results on DSA and its elliptic curve version see [Vau03]. In particular, it is not known that the additional DDH oracle or the offline/online kind of attack facilitates the task of breaking the signature scheme.

**SECURE CHANNELS.** A secure channel provides an end-to-end integrity-preserving confidential channel. Roughly, we demand that a secure channel hides messages (as do encryption schemes), but at the same time ensures the authenticity of sent messages (as in MAC schemes).

A secure channel  $\mathcal{SC} = (\text{KGen}, \text{Send}, \text{Rec})$  consists of: an algorithm for generating keys  $\text{KGen}$  (we assume in this thesis that the keys are random strings and that the hash function  $\mathcal{H}$  maps to such strings), a sending algorithm  $\text{Send}(k, m)$ , which wraps the message usually in an encrypted and authenticated container  $C$ , and a recovery algorithm  $\text{Rec}(k, C)$ , which, on input a key  $k$  and a container  $C$ , returns a message  $m$  or an error symbol  $\perp$ . We assume the usual notion of completeness, i.e., any faithfully wrapped message under any key is recovered by the corresponding recovery algorithm.

The security definition of secure channels is discussed in Section 3.3.

### 9.3.2. Security as a Key-Exchange Protocol

**Theorem 9.3.3** *The protocol PACE|AA (with Schnorr or DSA signatures) satisfies:*

$$\text{Adv}_{\text{PACE|AA}}^{\text{ake}}(t, Q) \leq \frac{q_e^2}{2q} + \text{Adv}_{\text{SC}}^{\text{lor}}(t^*, q_e, q_e) + \text{Adv}_{\text{PACE}}^{\text{ake}}(t^*, Q)$$

where  $t^* = t + O(kq_e^2 + kq_h^2 + kq_c^2 + k^2)$  and  $Q = (q_e, q_c, q_h)$ .

We remark that the runtime  $t^*$  includes the additional operations required to maintain lists and perform look-ups. Since PACE is secure (under cryptographic assumptions) it follows together with the security of the underlying encryption scheme that the PACE|AA scheme is secure as well.

The idea of the proof is roughly that the additional Schnorr signature does not violate the security of the underlying PACE protocol, since the signature is encrypted. This is shown through a reduction to the security of the original PACE protocol, mildly exploiting the structure of the original proof in [BFK09] and the properties of the Schnorr signature scheme. Intuitively, we show that in the PACE|AA protocol we can simulate the final transmission of the signature token by sending dummy values through the channel, because the keys used to secure this transmission are “as secure as” the PACE keys. That is, even though the strength of the keys is only password-protected (i.e., one can try to guess the low-entropy password), this is sufficient for our purpose, as we do not aim to achieve better security than that.

*Proof.* The proof uses the same game-hopping technique we used in previous theorems, gradually taking away adversarial success strategies and arguing that each modification cannot detract significantly to the overall success probability. In our proof we use the fact that the original proof of PACE in [BFK09] actually shows something stronger than indistinguishability of keys (from random), namely that computing the Diffie–Hellman key  $\mathcal{K}$  in an execution is hard (unless one knows or has guessed the password); In [BFK09], key indistinguishability then follows from this. We use this stronger property in the proof below and also consider the adversary against the PACE|AA protocol in this regard, i.e., we measure its success probability with respect to the probability of making a hash query about  $\mathcal{K}$  in a Test session (called target hash query).

**DESCRIPTION OF GAME<sub>0</sub>.** Corresponds to an AKE attack on the PACE|AA protocol (with the more fine-grained success notion).

**DESCRIPTION OF GAME<sub>1</sub>.** Abort GAME<sub>0</sub> if an honest chip card computes the same Diffie–Hellman key in two executions.

Note that, since the honest chip card always goes second for the Diffie–Hellman key exchange step, sending  $Y_C$ , the keys in such executions are random elements and the probability that such a collision occurs is thus at most  $\frac{1}{2}q_e^2/q$ .

**DESCRIPTION OF GAME<sub>2</sub>.** Change the previous game slightly such that when an honest chip card sends the encrypted signature, it picks and uses random and independent (independent of the hash function output) keys  $\mathcal{K}'_{SC}$  instead.

Note that the only difference between using a genuine signature and one generated with a random  $\mathcal{K}'_{SC}$  can occur if the adversary makes a target hash query since Reveal and Test sessions never output these keys and, as stipulated in GAME<sub>1</sub>, Diffie–Hellman keys are always distinct. It follows that the adversarial success can only decrease by the probability of making a target hash query in this new game.

**DESCRIPTION OF GAME<sub>3</sub>.** Change the game once more and replace channeled transmissions of the signatures sent by an honest chip by encryptions of 0-bits of the same length. At the same time, let any honest terminal reject any final message unless it has really been sent by the honest chip card in the same session.

Note that the length (of the signature part and the certificate) is known in advance. Note also that the probability of making a target hash query in GAME<sub>3</sub> cannot be significantly larger, by the distinguishing advantage of genuine transmissions from all-zero transmissions. To make this claim more formally, assume that we mount an attack on the left-or-right security of the (multi-user) encryption scheme by simulating the entire GAME<sub>2</sub> with two exceptions: (1) If an honest chip is supposed to send the signature and certificate, then we simply call the next transmission challenge oracle about the signature and the certificate components and about an all-zero message of the same length. Then, the challenge bit of the left-or-right oracle corresponds exactly to the difference between the two games. (2) If the adversary successfully modifies the final transmission of an honest chip card, but the honest terminal accepts the message, this also constitutes a security breach of the channel protocol. Hence, if the success probabilities of the adversary dropped significantly, we obtain a successful attacker against the secure channel scheme.

The final game can now be easily cast as an attack on the original PACE protocol. That is, if there is an attacker that is successful in GAME<sub>3</sub> (making a target hash query), then there is a straightforward attacker with the same probability to win against the original PACE protocol: this attacker runs the GAME<sub>3</sub>-adversary and simulates the additional signature steps by itself (i.e., creating keys and certificates); injects the values from the PACE protocol (i.e., relay the communication), but sends dummy values 0...0 through the channel on behalf of honest chip cards under independent random keys. It follows that the probability of making a target hash query in GAME<sub>3</sub> is also bounded by the PACE security.

Given that no target hash query is made, the advantage in the final game is now upper-bounded by the advantage against PACE. Note that the advantage of breaking PACE simultaneously covers both the case of target hash queries and of other security risks (thus, we do not need to account for the advantage of target hash queries and then of other attacks; the only other way left for the adversary to win is now to guess, thus resulting in a probability of 1/2). □

**ON FORWARD SECRECY.** Note that the PACE|AA protocol inherits the forward secrecy of PACE (when used as authenticated key-exchange protocol). That is, even if the adversary knows the password, then executions between honest parties remain protected. Since the security of PACE|AA essentially reduces to the security of PACE any successful attack against the forward secrecy of PACE|AA yields a successful attack against PACE; the other protocol steps do not violate this property.

### 9.3.3. Security against Impersonation

It remains to show that the protocol is IKE-secure. Here, we only rely on the unforgeability of certificates and MACs, and on the robust unforgeability of the Schnorr/DSA signature scheme.

**Theorem 9.3.4** *For the PACE|AA protocol (with Schnorr or DSA signatures) it holds:*

$$\begin{aligned} \mathit{Adv}_{\text{PACE|AA}}^{\text{ike}}(t, Q) \leq & \frac{q_e^2 + q_e q_h}{q} + \mathit{Adv}_{\{\text{Schnorr|DSA}\}}^{\text{r-forge}}(t^*, q_e) \\ & + 2q_e \cdot \mathit{Adv}_{\mathcal{M}}^{\text{forge}}(t^*, 2q_e, 2q_e) + \mathit{Adv}_{\text{CA}}^{\text{forge}}(t^*, q_e) \end{aligned}$$

where  $t^* = t + O(kq_e^2 + kq_h^2 + k^2)$  and  $Q = (q_e, q_h)$ .

The idea is to show first that the adversary cannot inject its own unregistered key (unless it breaks the unforgeability of the certification authority). Since any successful attack must then be for an uncorrupted party whose secret signing key was not revealed, it follows that the adversary must produce a signature under the (registered) public key of an honest user. Because the session id must be new and is partly signed via  $T_B$ , it follows that the adversary must forge Schnorr respectively DSA signatures in order to break the IKE property.

*Proof.* We again proceed in games. We can assume that the adversary (and all reductions below) know all passwords at the outset. For the adversary this can be achieved by issuing `Corrupt.pw` passwords in the beginning, while the reductions can choose the passwords themselves.

**DESCRIPTION OF GAME<sub>0</sub>.** Corresponds to the original attack.

**DESCRIPTION OF GAME<sub>1</sub>.** Abort if an honest reader accepts an unregistered key as valid.

Abort and declare the adversary to lose if it manages to make the honest reader accept an unregistered key in any execution. It follows straightforwardly from the unforgeability of certificates that this can decrease the adversary's success probability by at most a negligible term. It is straightforward to make this claim formally by simulating the attack (including the honest players, thus being able to decrypt the final message with the certificate forgery), and using an external certificate issuing oracles for registering the user's public keys.

**DESCRIPTION OF GAME<sub>2</sub>.** Abort if (possibly distinct) honest chip cards derive the same key  $\mathcal{K}$  in two executions.

Note that, once a chip selects  $Y_C$  at random in an execution, an honest or malicious reader has already sent  $Y_T$ . Hence, the key  $\mathcal{K}$  is a uniformly random element and the probability that it matches any of the previous  $i$  keys is at most  $i/q$ . Summing over all the at most  $q_e$  executions shows that the adversary's success probability can only drop by  $\frac{1}{2}q_e^2/q$ .

**DESCRIPTION OF GAME<sub>3</sub>.** Abort if there are collisions among the  $Y_T$  values of honest readers.

In case the same value  $Y_T$  chosen by (possibly distinct) honest readers appears in two executions also declare the adversary to lose. By the birthday bound and since there are at most  $q_e$  such values, this can only deduct  $\frac{1}{2}q_e^2/q$  from the success probability.

**DESCRIPTION OF GAME<sub>4</sub>.** Abort if a malicious reader submits a valid  $T_B$  in an execution with an honest chip card, and such that neither (A) the same valid  $T_B$  appears in an execution with an honest reader for the same session identifier, nor (B) does the adversary make a hash query to the key  $\mathcal{K}$  derived by the honest chip in the execution before.

Let  $sid = (Y_C, Y_T, \mathcal{G})$  be the session identifier in an execution with an honest chip card in which the adversary submits a valid  $T_B$  without having made a hash query beforehand, and such that  $T_B$  does not appear in a session with an honest reader for the same session identifier. Call this a target execution

and fix it for now. Let  $\mathcal{K}$  be the Diffie–Hellman key derived by the chip in this execution.

According to the previous games we can assume that all keys in executions with honest chip cards are unique, and that there is at most one execution with an honest reader in which the same *sid* is used (because the values  $Y_{\top}$  chosen by honest readers are distinct). Consider now all executions between the adversary (as a chip card) and honest readers in which the same values  $Y_C, \mathcal{G}$  as in the target session appear. Since the  $Y_{\top}$  values are unique, there is at most one session with the same key  $\mathcal{K}$  and the same pair  $(Y_C, \mathcal{G})$  — and this execution must then carry the same value  $Y_{\top}$  — and thus the same session identifier as the target session. In other words, for a successful attack in the target session the adversary must send a valid MAC for an unknown key  $\mathcal{K}$ , or for a new value  $(Y_C^*, \mathcal{G}^*)$  (or both). We can therefore derive a contradiction to the unforgeability of the MAC as follows.

Simulate an attack against the MAC scheme by running the entire PACE|AA protocol and the adversary. Pick at random a session among the at most  $q_e$  ones in advance and follow exactly the description of  $\text{GAME}_3$ , but with the following differences: Compute the key  $\mathcal{K}$  in the pre-selected session as before (abort if the session aborts before or if the party is corrupt), as well as the keys  $\mathcal{K}_{\text{ENC}}, \mathcal{K}_{\text{MAC}}, \dots$ , but not the key  $\mathcal{K}'_{\text{MAC}}$ . Proceed accordingly if the key appears in another session and ignore, too, when asked to compute  $\mathcal{K}'_{\text{MAC}}$ . When it comes to verification or to MAC computation under this key  $\mathcal{K}'_{\text{MAC}}$  in any session, call the external verification resp. MAC oracle instead. (Stop if a verification request for a new message is accepted.)

For the analysis note that the simulation is perfect if the adversary never makes a hash query for the target session. Also, if the adversary at some point submits a valid MAC  $T_B$  under the key  $\mathcal{K}$  in the target session, then we guess the right session in which this key appears for the first time with probability  $1/q_e$ . Given this, we successfully forge a MAC, i.e., submit a new message  $(Y_C, \mathcal{G})$  with a valid MAC to the verification oracle. To see this is a valid forgery note that for the key  $\mathcal{K}'_{\text{MAC}}$  in question we only compute MACs on behalf of honest readers, but then only for pairs  $(Y_C, \mathcal{G})$  different from the one used by the successful adversary resp. for a new key no MAC has been computed before (as discussed above). It follows that the adversary's loss when proceeding from  $\text{GAME}_3$  to  $\text{GAME}_4$  can only be  $q_e$  times the probability of forging a MAC.

**DESCRIPTION OF GAME<sub>5</sub>.** Abort if the adversary queries its hash function oracle about a Diffie–Hellman key  $\mathcal{K}$  in an execution with the honest chip card, before it is determined by the value  $Y_C$ .

Note that, once  $Y_T$  has been sent, the random and independent value  $Y_C$  in such an execution makes the key  $\mathcal{K}$  random and thus the probability of the adversary having queried  $\mathcal{H}$  about  $\mathcal{K}$  before is at most  $q_h/q$  times the number  $q_e$  of such keys.

**DESCRIPTION OF GAME<sub>6</sub>.** Abort if a malicious reader submits a valid  $T_B$  in an execution with the honest chip card, but such that  $T_B$  has been sent by an honest reader before in a session with a different session identifier.

According to the previous games, if the session identifier is distinct in the latter session, the adversary must query the random oracle about the key before being able to send a valid  $T_B$  in this session. In this case, however, we can apply an information-theoretic argument based on the unforgeability of the MAC. Recall that in executions with an honest chip card the key  $\mathcal{K}$  is unique. Fix one such execution for the moment and call this the target execution. Let  $\mathcal{T}_B$  be the set of (at most  $q_e$ ) values  $T_B$  sent in executions with honest readers. Then, in the target execution with values  $Y_C, Y_T, \mathcal{G}$ , the probability that the (unique) key  $\mathcal{K}$  hashes to a key  $\mathcal{K}'_{MAC}$  such that  $MVf(\mathcal{K}'_{MAC}, T_B, (Y_C, \mathcal{G})) = 1$  for some  $T_B \in \mathcal{T}_B$ , is negligible. This can be seen as follows: Since key  $\mathcal{K}$  is uniquely determined from the target execution before, the value  $\mathcal{K}'_{MAC}$ , when returned to the adversary upon a hash query about  $\mathcal{K}$ , is random and independent from all other keys. If, by chance,  $MVf(\mathcal{K}'_{MAC}, T_B, (Y_C, \mathcal{G})) = 1$  for a fixed  $T_B \in \mathcal{T}_B$ , then we can easily devise a forgery against the MAC scheme as follows. Mount an attack against the MAC scheme by simulating the security game, but record all values in  $\mathcal{T}_B$  and make a verification query about  $T_B, (Y_C, \mathcal{G})$  for all values  $T_B \in \mathcal{T}_B$  and for the values  $Y_C, \mathcal{G}$  after sending  $Y_C$  in an execution on behalf of an honest chip card. Note that the adversary cannot have made a hash query about this key  $\mathcal{K}$  before, according to the previous game. Hence, it follows that the key  $\mathcal{K}'_{MAC}$  is still an undetermined random value and the probability that a verification query in the simulation succeeds is exactly equal to the probability in the attack on the MAC (for an unknown random key).

**DESCRIPTION OF GAME<sub>7</sub>.** Abort if the adversary sends a valid signature on behalf of an honest chip card for a fresh session.

Abort if the adversary manages to send a valid (encapsulated) signature on behalf of an honest chip card to the honest reader (for a session with a fresh  $sid$ , which does not appear in another accepting execution) for the challenge value  $T_B$  in this execution. By the previous game the adversary cannot have a value  $T_B$  signed by an honest chip card, before sending it in an execution with the honest reader, unless the session identifiers match. In other words, in order to impersonate successfully, the adversary needs to send a valid signature for a new value  $T_B$ . This can be straightforwardly turned into an attack against the signature scheme, as discussed next.

Pick at the outset one of the at most  $q_u$  active users. Our forger against the signature scheme injects the given public key of the signature scheme as the chosen user's public key. Whenever the adversary invokes a run of this user then we call the token step of the Schnorr signature scheme to get a value  $X_C$ . Similarly, we run the token step in case of DSA signatures but take the output modulo  $q$ . We inject this value into the execution and later send a random value  $Y_C$  on behalf of the user. Note that we thus do not know the key  $\mathcal{K}$  in this execution, but according to the previous games we can check in executions with the adversary for a candidate among the hash queries via the Decisional Diffie–Hellman oracle for a candidate and the two values  $Y_C, Y_T$  from the execution — if we do not find any match we can simply reject. In executions with an honest reader we can actually derive the key from the reader's view on the execution. It follows that we can still compute the right key and then complete the signature token once we have to send the encrypted signature for  $T_B$ .

Note that the simulation is perfect from the adversary's view. Hence, if the adversary in  $\text{GAME}_7$  eventually convinces an honest reader to accept a signature for the value  $T_B$  in the execution, then it follows that this value has not been signed before (or the session identifiers are identical and the adversary cannot win then), we derive a successful forger against the signature scheme.

This concludes the description of the games. Note that in the final game the adversary cannot successfully impersonate anymore since the adversary cannot send valid signatures on behalf of honest chip cards and hence, cannot successfully run the protocol.  $\square$

## 9.4. A Deniable Schnorr Variant

Deniability basically demands that for any (possibly malicious) party on either side, there exists a simulator which produces the same output distribution as the malicious party, but without communicating with the honest party. This implies that the malicious party could have generated this data itself, without the help of the other party, and thus cannot use it as a proof towards a third party.

### 9.4.1. Defining Deniability

Unlike in the key-exchange setting, we now assume that only one chip card and one terminal are present (but may run many executions concurrently), and that the adversary controls either party from the beginning. No further corrupt queries are allowed. We note that deniability in the multi-user setting immediately follows via a hybrid argument if the parties' secret inputs are otherwise picked independently (as is the case here).

Since we work in the random-oracle model, we must account for a peculiarity due to the (non-)programmability of the hash function [Pas03]. Roughly, it is important that the distinguisher (receiving either the view of the malicious party or the simulated view) cannot distinguish these two random variables, even if it gets access to the same random oracle as the parties and the simulator. The distinguisher's access to the same hash function prevents the simulator from programming the hash values (as it would be the case for a real-world hash function).

**Definition 9.4.1 (Deniability)** *A password-based key-exchange protocol  $P$  is deniable in the random-oracle model if for any (possibly adversary-controlled) party with access to the random oracle  $\mathcal{H}$  there exists an efficient algorithm  $\mathcal{S}^{\mathcal{H}}$  such that, on input the party's secret and public input, as well as the other party's public input,  $\mathcal{S}^{\mathcal{H}}$  generates the same output distribution as the malicious party in concurrent executions of the protocol. That is, for any algorithm  $D^{\mathcal{H}}$ , the output of  $D^{\mathcal{H}}$  when receiving the public data of both parties and the secret input of the malicious party, in addition to either the output of  $\mathcal{S}^{\mathcal{H}}$  or the output of the malicious party, is indistinguishable in both cases. We write  $\text{Adv}_P^{\text{den}}(T, Q)$  for a bound on the difference  $|\Pr[D^{\mathcal{H}}(\mathcal{A}^{\mathcal{H}}) = 1] - \Pr[D^{\mathcal{H}}(\mathcal{S}^{\mathcal{H}}) = 1]|$ , where  $D^{\mathcal{H}}(\mathcal{A}^{\mathcal{H}})$  is the output of  $D$  (in time  $t^*$  with at most  $q_h^*$  hash queries) when run in the experiment with  $\mathcal{A}$  (running in*

time  $t$ , invoking at most  $q_e$  protocol executions and with at most  $q_h$  hash queries); analogously,  $D^{\mathcal{H}}(\mathcal{S}^{\mathcal{H}})$  is the output of  $D$  when run in the experiment with  $\mathcal{S}$  (running in time  $t'$  with at most  $q'_h$  hash queries). Let  $T = (t, t', t^*)$  and  $Q = (q_e, q_h, q'_h, q_h^*)$ .

Clearly, without loss of generality, it suffices that the distinguisher outputs a bit only. Ideally, the advantage should be small for any efficient  $D$  and  $\mathcal{A}$  and where the simulator's runtime characteristics is close to the adversary's. We note that there are even stronger notions of deniability, e.g., *online deniability* [DKSW09], where the distinguisher can communicate with the malicious party, resp. the simulator, while the protocol is executed. This notion, however, is much harder to achieve and not known to work here.

### 9.4.2. Deniability of Our Protocol

Our deniable version of the Schnorr scheme works as before, only that this time we hash  $(Y_C, \mathcal{G})$  instead of  $T_B$ . We call this protocol the *deniable Schnorr-based PACE|AA protocol*. Roughly, the idea is now that the chip chooses the challenge by itself. Given that the challenge is chosen beforehand and that it is independent of the first signature step, one can simulate the final signature part as in the interactive Schnorr identification protocol [Sch91]. We only need to take care that the other security properties are not violated through this.

Note that security as an AKE protocol follows as in the Schnorr-signature-based version (with the exact same bounds). It suffices to show impersonation resistance (which follows similarly to the case of signatures) and deniability. We note that our deniability simulator will actually need some assistance in form of a decisional Diffie–Hellman oracle (which, for the sake of fairness, we then also give the adversary and to the distinguisher). We comment that this does not trivialize the task, as such a decision oracle is not known to help compute discrete logarithms. Thus, the simulator cannot simply derive the chip card's secret key from the public key and use this key to provide deniability. We note that the query parameters  $Q$  thus take additional bounds  $q_{\text{DDH}}$ ,  $q'_{\text{DDH}}$ , and  $q_{\text{DDH}}^*$ .

**Theorem 9.4.2** *For the deniable Schnorr-based PACE|AA protocol it holds that:*

$$\begin{aligned} \text{Adv}_{\text{PACE|AA}}^{\text{ike}}(t, Q) &\leq \frac{2q_e^2 + q_e q_h}{q} + \text{Adv}_{\mathcal{CA}}^{\text{forge}}(t^*, q_e) \\ &\quad + 2q_e \cdot \text{Adv}_{\mathcal{M}}^{\text{forge}}(t^*, 2q_e, 2q_e) + \text{Adv}_{\text{Schnorr}}^{\text{r-forge}}(t^*, q_e) \end{aligned}$$

where  $t^* = t + O(kq_e^2 + kq_h^2 + k^2)$  and  $Q = (q_e, q_h)$ .

*Proof.* The proof is almost identical to the one of Theorem 9.3.4 (impersonation resistance). We start after the hop to  $\text{GAME}_6$  and make another game hop, aborting if in two executions with honest chip cards, the cards send the same value  $Y_C$ . Since these values are random group elements we only lose a term  $\frac{1}{2}q_e^2/q$ , analogously to the hop to  $\text{GAME}_3$  in the previous proof. Analogously, we can assume that there are no collisions among the values  $Y_C$  and  $Y_T$  picked by honest chip cards or terminals, respectively. This also decreases the adversary's success probability by at most  $\frac{1}{4q}(2q_e)^2$ , since there are at most  $\frac{1}{2}(2q_e)^2$  pairs and the probability that there is a collision among the values chosen by the chip card and terminal is at most  $1/2q$ .

Assume now that the adversary at some point successfully impersonates an honest chip card to an honest terminal, by using a pair  $(Y_C, \mathcal{G})$  in the hashing step of our Schnorr version, such that this pair has been used by an honest chip card before. Since the values  $Y_C$  are unique, there exists at most one such execution. In this execution the adversary must have sent a different value than  $Y_T$  in the successful impersonation, or else the session identifiers would be identical. It follows that both executions have distinct keys. It furthermore holds that  $Y_T \neq Y_C$ . We can now apply an argument analogously to the one in  $\text{GAME}_4$ ,  $\text{GAME}_5$ , and  $\text{GAME}_6$  to argue that the adversary cannot find a valid token  $T_A$  on behalf of the honest chip. However, we need to make the MAC query about  $\mathcal{K}'_{\text{MAC}}(Y_C, \mathcal{G})$  before to compute  $T_B$  on behalf of the honest terminal. But since  $Y_C$  and  $Y_T$  are then both picked by honest users, this implies by assumption that  $(Y_C, \mathcal{G})$  is different from  $(Y_T, \mathcal{G})$  in this execution; thus, the MAC  $T_B$  does not help to forge the MAC  $T_A$ .

Hence, we can assume that the adversary uses a fresh pair  $(Y_C, \mathcal{G})$ , not previously authenticated under the key of an honest chip card. This, however, contradicts the unforgeability of the special Schnorr identification version, as discussed after Definition 9.3.1.  $\square$

**Theorem 9.4.3** *The deniable Schnorr-based PACE|AA protocol is deniable in the random-oracle model if the MAC is unforgeable (and the adversary, simulator, and distinguisher are granted access to a decision Diffie Hellman oracle). That is, for any malicious chip or terminal  $\mathcal{A}$  (with access to the random oracle and a DDH oracle) there exists a simulator  $\mathcal{S}$  (with the same oracle access) such that for any distinguisher*

$D$  (with the same oracle access) we have

$$\mathbf{Adv}_{\text{PACE|AA}}^{\text{den}}(T, Q) \leq \frac{q_e^2}{q} + 2q_e \cdot \mathbf{Adv}_{\mathcal{M}}^{\text{forge}}(t', 2q_e, 2q_e)$$

where  $t' = t + t^* + O(kq_e^2 + kq_h^2 + k^2)$  and  $q'_h = q_h + q_h^*$  and  $q'_{\text{adh}} = q_h^2 + q_{\text{adh}} + q_{\text{adh}}^*$ .

*Proof.* It is easy to see that one can easily simulate the view of a malicious chip card, by just following the protocol on the terminal's side (since the password is considered a joint secret input it is easy to run the terminal's steps) and mount a black-box simulation of the malicious chip card, outputting whatever this party outputs. The output distribution is identical and the simulator makes the same calls to the hash functions as the honest party.

The more interesting case is that of a malicious terminal. We present our simulator  $\mathcal{S}^{\mathcal{H}}$  for this case. Recall that, this time, the simulator only has access to the chip card's public key  $pk_C$ , the group data, and the password (but not the chip's secret key). The simulator now proceeds as follows, running a black-box simulation of the adversarial terminal (playing the honest chip card). In each execution the simulator initially picks values  $x_C, y_C \leftarrow \mathbb{Z}_q$  and computes  $Y_C = g^{y_C}$ , as well as  $c = \mathcal{H}(Y_C, \mathcal{G})$  and  $X_C = pk_C^{-c} g^{x_C}$ . Note that both values are not computed according to the protocol description, but still have the same distribution. In particular, even though the simulator cannot compute the shared Diffie–Hellman key  $\mathcal{K}$  in the execution, it can later complete the signature generation by setting  $s\sigma = x_C$  (such that  $g^\sigma = X_C pk_C^{\mathcal{H}(Y_C, \mathcal{G})}$ ). For the other steps the simulator proceeds as the chip card would, using its knowledge of the password. However, when the simulator receives  $T_B$  from the malicious terminal, it searches (with the decisional Diffie–Hellman oracle) in the list of hash queries of the malicious terminal for queries about a key  $\text{DH}(Y_C, Y_T)$ . If no key is found then abort this execution; else use the found key-value  $\mathcal{K}$  to finish the execution (using the signature tokens as computed above). If the adversary stops, then let the simulator output the same value.

It remains to show that, with overwhelming probability, the malicious terminal cannot send a valid token unless it has queried the random oracle about the Diffie–Hellman key before. This follows as in the proof of Theorem 9.3.4 (impersonation resistance). There, in game hops from  $\text{GAME}_0$  to  $\text{GAME}_4$  it is shown that any such execution would yield a contradiction against the unforgeability of the MAC (some of the hops, namely to  $\text{GAME}_1$ ,  $\text{GAME}_3$ , and case (A) of  $\text{GAME}_4$ , do not apply here because only the chip card is honest).

Hence, unless the malicious terminal can forge MACs, the simulator will find a (unique) key in the list, such that the behavior of the simulator is indistinguishable from the one of the honest party. Additionally, the simulator only queries the hash function as the chip card would, allowing us to conclude that the output of  $D^H$  is also indistinguishable in both cases.  $\square$

# 10. Domain-Specific Pseudonymous Signatures

## 10.1. Introduction

In the optional protocol for the eID cards, namely Restricted Identification, card holders can use domain-specific pseudonyms to interact with service providers such that (a) a service provider can recognize pseudonyms of individual cards and use this information for the service (domain-specific linkability), and (b) different service providers cannot link interactions of one user in their respective domains (cross-domain anonymity). Although the concept of restricted identification in [BSI10] — and the Diffie–Hellman based solution — currently only support recognition of pseudonyms, it can be easily extended to provide additional functionality, by allowing users to create signatures under their pseudonyms. These signatures could then be used, for instance, to authenticate outgoing messages or transcripts under the card’s pseudonym.

**DOMAIN-SPECIFIC PSEUDONYMOUS SIGNATURES.** The security of the basic restricted identification (RI) protocol is shown in Chapter 7. We augment the RI protocol through signatures. To this end, we formally introduce the concept of *domain-specific pseudonymous signatures*.<sup>1</sup> In a sense, (domain-specific) pseudonymous signatures can be seen as a relaxed version of group signatures with a limited form of linkability: While group signature schemes, as formalized in [BMW03], provide a very strong form of anonymity, preventing an adversary to identify signers even when knowing the secret keys, pseudonymous signatures are designed specifically to allow a well-defined verifier to link signatures.

---

<sup>1</sup>Interestingly, the term “pseudonymous signatures” has occasionally already been mentioned in the literature, typically referring to regular signatures under pseudonyms [KBe12], but, to the best of our knowledge, such notions have never been considered formally so far, from a cryptographic point of view.

The group-signature ancestry also lays the ground work to the modelling of pseudonymous signatures' security. According to [BMW03], secure group signature schemes should guarantee full anonymity (as aforementioned), resistance to identifying the origin of signatures, and full traceability, i.e., the ability of the group manager to trace the origin of a signature with the help of some trapdoor information. The latter property implies, for example, unforgeability (because a forgery could not be traced) and non-frameability, i.e., an attack where a set of malicious parties, potentially including the group manager, could falsely blame an honest user of producing a signature that this user did not generate (which would again contradict traceability).

By contrast, in the case of restricted identification, we note that one of its goals is exactly to allow a service provider to link previously seen pseudonyms. We thus relax the full-anonymity requirement for pseudonymous signatures and only demand cross-domain anonymity, i.e., the inability to link signatures given to different service providers, even if the providers are malicious. Furthermore, since the restricted-identification scenario does not involve an authority like the group manager, to trace signatures, we revert to explicitly redefining unforgeability for our setting, i.e., we need to ensure that one cannot forge signatures on behalf of honest users.

We also introduce another property, demanding that it is infeasible to make the verifier accept a signature for an invalid domain-specific pseudonym. This assumes a white- or blacklisting approach. In the former case, valid pseudonyms are those in the white list; in the latter case, the verifier should not accept a blacklisted pseudonym. This property is not equivalent to unforgeability, which merely protects honest signers from forgeries under their name. While the additional property of rejecting invalid pseudonyms, which we call *seclusiveness*, follows in the case of group signatures from full-traceability, we need to state it explicitly in our setting without a tracing authority. Roughly, unforgeability together with seclusiveness provides a weaker, yet "best-we-can-hope-for" form of full-traceability for domain-specific pseudonymous signatures.

**OUR CONSTRUCTION.** The basic restricted identification protocol is roughly to give users a random value  $x_1$  and the service provider a certified group element  $R$ , such that the user will derive the domain-specific pseudonym as the Diffie–Hellman key  $I_R = R^{x_1}$  (or, to be precise, the hash value thereof). This value is then sent over a previously established secure channel to the service provider (but we ignore the channel part in our analysis).

To achieve the additional unforgeability and seclusiveness properties in the signature-augmented case, the user is instead given a random representation  $(x_1, x_2)$  of the authority's public key  $y$ , i.e.,  $y = g_1^x = g_1^{x_1} g_2^{x_2}$  for generators  $g_1, g_2 = g_1^z$ . The authority can create such representations easily with the knowledge of  $z = \log_{g_1} g_2$ . Each domain will again hold a certified group element  $R$ ; the user once more derives the domain-specific pseudonym as the Diffie–Hellman key  $I_R = R^{x_1}$  and transmits this value to the provider.

Obviously, if two malicious users could pool their distinct secrets  $(x_1, x_2)$  and  $(x'_1, x'_2)$  then they could recover the authority's secret key  $z$ . However, we note that these secrets are protected through the hardware of the identity card and are not available to users. In a sense, our security analysis relies on this fact, but at the same time even allows a single (malicious) user access to its secret key. In case of suspicion of leakage of the authority's secret key  $z$ , a new key can be generated, the old key can be revoked, but whitelisting can be used to mark the user keys under the old key as still valid.

In the signature-augmented version, the user additionally signs a message  $m$  by giving two intertwined non-interactive proofs of knowledge (where the message enters the hash evaluation to derive the challenge non-interactively in the random-oracle model). The first proof shows that the user knows the discrete logarithm  $x_1$  of  $I_R$  to base  $R$ , and the other proof shows that it additionally knows  $x_2$  such that  $(x_1, x_2)$  forms a representation of  $y = g_1^{x_1} g_2^{x_2}$ . The first proof is basically a Schnorr proof of knowledge (PoK) [Sch91]; the second one is an Okamoto kind of PoK [Oka93], but re-using the data from the Schnorr proof.

We show that the signature-augmented version of the restricted-identification protocol preserves the cross-domain anonymity according to our adapted notion for pseudonymous signatures. Furthermore, we show unforgeability (in the random-oracle model, and under the discrete log assumption). This property is ensured by the Schnorr PoK for  $x_1$ . Seclusiveness follows from the Okamoto proof, which shows that the pair  $(x_1, x_2)$  is a representation of  $y$  and has thus been issued by the authority. However, our construction requires that no two malicious users collaborate. Such collaboration is made harder in practice by hiding and protecting the secret of users within the secure hardware of the chip, as discussed above. Thereby, users can use their secrets; however, they cannot extract the keys to collude. Note that the basic version of the restricted identification protocol does not implement a signature functionality and thus cannot satisfy these two notions.

**RELATED WORK.** Domain-specific pseudonymous signatures can be seen as descendants of group signatures, with slightly weaker anonymity requirements, but with domain-specific verifiers. The related concept of ring signatures [RST01] can be viewed as “ad-hoc” group signatures without a central manager. For such schemes, rings of users can be formed at will, and one user can sign on behalf of the ring, but there is usually no mean to identify the actual signer later (i.e., signatures are not traceable). Domain-specific pseudonymous signatures provide stronger notions of traceability than ring signatures, but a weaker form of anonymity — the domain holder can link signatures.

We note that credential systems [Cha85, Bra00] are very similar to our pseudonymous signatures, but diverge in some important aspects. Most importantly, credential systems typically provide multi-show unlinkability, as opposed to our pseudonymous signatures. As such, solutions for multi-show credential systems are usually slightly more complex [CL01, CL02, CL04, PV04, BCKLo8, GNSN10]. For one-show solutions, where the user can use a credential (under a pseudonym) only once, domain-specific linkability — and therefore cross-domain anonymity — has not been considered before. We also note that the question of turning cross-domain pseudonymous signatures into fully-fledged multi-show credential systems is beyond the scope of the thesis here: the requirement of recognizing pseudonyms for domain holders is inherent in the application requirement.

Finally, we point out that our notion of domain-specific pseudonymous signatures is close to a recent proposal of Bernhard *et al.* [BFG<sup>+</sup>11] for defining direct anonymous attestation (DAA). Their security definition for DAA also resembles group signature security, but comes with a limited form of linkability: signatures of the same user in the same domain (called base there) must be publicly linkable. At the same time, one must be able to identify signatures given the user’s secret key, a requirement which we do not impose in our setting. Moreover, their scenario assumes that linkability must be enforced via cryptographic means, domain-specific pseudonymous signatures allow this by default through the pseudonyms. Besides minor technical differences concerning security in the presence of compromised keys or incorporating blacklisting, the main difference to their setting is that our model takes into account the extra layer of domain-specific pseudonyms and its unlinkability to the pseudonym layer.

Similarly, Wei [Wei05] also uses DAA as a motivation for his work on tracing-by-linking group signatures, but he considers a weaker form of anonymity,

where a signer's identity is only hidden up to at most a fixed number of generated signatures. Any additional signature enables to trace back the identity of the signer by a public algorithm. More formally, the approach also includes a notion of  $k$ -linkability which corresponds to (public) traceability and essentially means that more than  $k$  signatures are linkable and allow to identify the origin. It also contains a notion of non-slanderness which means that no group of malicious users (including the issuing authority) can sign more than  $k$  times such that it points to an honest user outside of the group. This primitive does not help in our scenario, because we demand unlinkability only among different domain sectors. Furthermore, our notion of seclusiveness, in contrast with non-slanderness (which addresses the linkability of more than  $k$  signatures), refers to the fact that one cannot produce any signature on behalf of honest users.

The idea of extending restricted identification to allow unlinkable sector signatures has appeared concurrently in [KS11b]. The scheme does not cover the issue of seclusiveness, though, and is less explicit about the underlying security model, e.g., it remains unclear if unlinkability holds for multiple signatures. Exploring such questions and providing sound models, including issues related to blacklisting or whitelisting, and to prove security according to these models for the (augmented) restricted identification protocol of the new German identity cards is our contribution here.

We also mention a very recent result from [BCP13], where the authors build a domain-specific signature scheme; opposed to our construction, the seclusiveness property holds for their construction even if several malicious chip cards collude and share their secrets. However, this scheme is more complicated in its structure and significantly less efficient (e.g., it requires many pairing computations).

## 10.2. Definitions

Below we first define our domain-specific pseudonymous signature scheme for static groups. Since both the secret keys of the users and the domain keys are chosen by the authority, we can imagine that a sufficiently large set is chosen at the outset, and individual entries only become active if required. We thus assume an algorithm  $\text{NymKGen}$  generating the group manager's key pair, as well as sufficiently many pseudonyms  $\text{nym}$  (and secret keys  $\text{gsk}[\text{nym}]$ ) and public domain keys  $\text{dpk}$ . Each pseudonym and its secret key

can now be combined via an algorithm  $\text{NymDSGen}$  to build a domain-specific pseudonym  $\text{dsnym} = \text{nym}[\text{dpk}]$  which, together with  $\text{gsk}[\text{nym}]$ , can be used to sign messages.

**Definition 10.2.1 (Domain-Specific Pseudonymous Signature)** *A domain-specific pseudonymous signature scheme is a collection of the following efficient algorithms  $\mathcal{NYS} = (\text{NymKGen}, \text{NymDSGen}, \text{NymSig}, \text{NymVf})$  defined as follows.*

$\text{NymKGen}(1^\kappa, 1^n, 1^d)$  *is a probabilistic algorithm which, on input a security parameter  $1^\kappa$  and parameters  $1^n, 1^d$  (both polynomial in  $\kappa$ ) outputs a pair  $(\text{gpk}, \text{gmsk})$  where  $\text{gpk}$  is the group public key and  $\text{gmsk}$  the secret key of the group manager, and outputs  $n$  (unique) pseudonyms  $\text{nym}$  with their corresponding secret keys  $\text{gsk}[\text{nym}]$ , and  $d$  domain descriptions  $\text{dpk}$ .*

$\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk})$  *is a deterministic algorithm which maps a pseudonym  $\text{nym}$  (and its secret key  $\text{gsk}[\text{nym}]$ ) and the domain  $\text{dpk}$  to a domain-specific pseudonym  $\text{dsnym} = \text{nym}[\text{dpk}]$ .*

$\text{NymSig}(\text{dsnym}, \text{gsk}[\text{nym}], \text{dpk}, m)$  *is a probabilistic algorithm which, on input a domain-specific pseudonym  $\text{dsnym}$ , a secret key  $\text{gsk}[\text{nym}]$ , a domain  $\text{dpk}$ , and message  $m$ , outputs the signature  $\sigma$  of  $m$  under  $\text{dsnym}$  for domain  $\text{dpk}$ .*

$\text{NymVf}(\text{gpk}, \text{dsnym}, \text{dpk}, m, \sigma, \mathcal{B})$  *is a deterministic algorithm which, on input a message  $m$  and a signature  $\sigma$  together with the group public key  $\text{gpk}$ , a domain-specific pseudonym  $\text{dsnym}$ , the domain's key  $\text{dpk}$ , and a list  $\mathcal{B}$ , outputs either 1 (=valid) or 0 (=invalid).*

*We assume the usual completeness property, i.e., for any honestly generated parameters, domain-specific pseudonyms, and signatures the verification algorithm accepts and outputs 1.*

Note that we can assume that the group manager uses some standard way of certification for the public keys  $\text{dpk}$  given out to registered verifiers, and that the signing and verification algorithms check the validity of the keys. We thus often omit this step from the description of protocols.

### 10.2.1. Cross-Domain Anonymity

Cross-domain anonymity ensures that pseudonyms across different domains cannot be linked — within one domain pseudonyms are meant to be linkable. We define cross-domain anonymity via a game between the adversary and a

left-or-right oracle which, given two pseudonyms, a message, and a domain, generates a signature for either the left or the right pseudonym in the domain, according to a secret random bit  $b$ . We assume that the adversary can make adaptive calls to this left-or-right oracle; this is necessary since we cannot apply a hybrid argument to reduce such multiple queries to a single one (as opposed to the case of full-anonymity for group signatures in [BMW03], where this is possible since anonymity even holds if the adversary knows the users' secret keys). The adversary's goal is to predict  $b$  significantly beyond the pure guessing probability (condition (a) below).

In addition to challenge queries to the left-or-right oracle, the adversary may decide to blacklist domain-specific pseudonyms, create additional signatures via NymSig, or corrupt users. For simplicity, we define a version for *static corruptions* where all corruptions are made at the outset, before any other oracle calls are made, and discuss the adaptive version briefly below. To exclude trivial attacks, we must take into account that domain-specific pseudonyms are in principle linkable by the domain holder. Hence, in "transitivity" attacks, where the adversary asks the left-or-right oracle first about a signature for domain-specific pseudonyms  $dsnym_0, dsnym_1$  and then for  $dsnym_0, dsnym'_1$  for the same domain, but  $dsnym_1 \neq dsnym'_1$ , the signatures would point to the same domain-specific pseudonym if and only if the oracle signs under the left pseudonym. We thus exclude such queries in condition (b) below.

We require another case to be excluded. Namely, the additional signatures generated through NymSig cannot hide the pseudonyms behind the signatures; this would require further means like anonymous signatures and would only work if signatures are not publicly verifiable [YWDW06, Fiso7]. Since such extra signatures for domain-specific pseudonyms would thus also allow to link the origin in the left-or-right queries to the pseudonyms, we must disallow the adversary from querying NymSig about domain-specific pseudonyms, which are also used in left-or-right queries (condition (c) below).

Our model assumes, to the advantage of the adversary, that all pseudonyms, all domain keys, and domain-specific pseudonyms are known at the outset; only the assignment of pseudonyms to domain-specific pseudonyms remains hidden. In the following,  $W$  denotes the set of all domain-specific pseudonyms  $dsnym$ ,  $D$  the set of domain keys,  $B$  the blacklist containing domain-specific pseudonyms,  $N$  the set of generated pseudonyms,  $C$  the corrupted chip cards,  $S$  the set of queried signatures, and  $LR$  denotes the set of queries made to the left-or-right oracle.

We assume that the relation of domains and domain-specific pseudonyms is known (see below for the motivation). All this is captured by giving the adversary the corresponding data as sets, that is,  $W \odot D$  being the set of all domain-specific pseudonyms and domains. The adversary will thus attack domain-specific pseudonyms from  $W$ . As is common, we measure the adversary's running time including also all steps of honest parties, and covering both phases of the adversary.

In the definition, we presume that domain-specific pseudonyms are unique within a domain; global uniqueness can then be trivially achieved by attaching the domain key to the pseudonym. Indeed, domain-specific uniqueness will be later ensured by the unforgeability property anyway.

**Definition 10.2.2 (Cross-Domain Anonymity)** *A domain-specific pseudonymous signature scheme  $\mathcal{NYS} = (\text{NymKGen}, \text{NymDSGen}, \text{NymSig}, \text{NymVf})$  is  $(n, d, t, Q, \varepsilon)$  cross-domain anonymous with  $Q = (q_c, q_s, q_t)$  if for any algorithm  $\mathcal{A}$  running in time  $t$ , and making at most  $q_c$  queries to the corruption oracle,  $q_s$  queries to the signing oracle, and  $q_t$  queries to the left-or-right oracle, the probability that the following experiment returns 1 is at most  $\varepsilon$ :*

**Experiment** CD – Anon $_{\mathcal{A}}^{\mathcal{NYS}}(\kappa, n, d)$

$b \xleftarrow{\$} \{0, 1\}$

$\text{LR}, \text{S}, \text{W}, \text{B}, \text{C}, \text{N}, \text{D} \leftarrow \emptyset$

$(\text{gpk}, \text{gmsk}, \{\text{gsk}[\text{nym}]\}_n, \{\text{nym}\}_n, \{\text{dpk}\}_d) \leftarrow \text{NymKGen}(1^\kappa, 1^n, 1^d)$

$\text{N} = \{\text{nym}\}_n$ , and  $\text{D} = \{\text{dpk}\}_d$

$\text{W} = \{\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk}) \mid \text{nym} \in \text{N}, \text{dpk} \in \text{D}\}$

$\text{W} \odot \text{D} = \{(\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk}), \text{dpk}) \mid \text{nym} \in \text{N}, \text{dpk} \in \text{D}\}$

$st \leftarrow \mathcal{A}^{\text{Corrupt}}(\text{gpk}, \text{W} \odot \text{D}, \text{N})$

$d \leftarrow \mathcal{A}^{\text{B}', \text{NymSig}', \text{LoR}}(st)$

Return 1 iff

(a)  $d = b$  and

(b) for any  $(\{\text{dsnym}_0, \text{dsnym}_1\}, \text{dpk}, m), (\{\text{dsnym}'_0, \text{dsnym}'_1\}, \text{dpk}, m') \in \text{LR}$

we have either  $\{\text{dsnym}_0, \text{dsnym}_1\} = \{\text{dsnym}'_0, \text{dsnym}'_1\}$

or  $\{\text{dsnym}_0, \text{dsnym}_1\} \cap \{\text{dsnym}'_0, \text{dsnym}'_1\} = \emptyset$ , and

(c) for any  $(\text{dsnym}, \text{dpk}, m) \in \text{S}$  there is no  $\text{dsnym}', m'$

such that  $(\{\text{dsnym}, \text{dsnym}'\}, \text{dpk}, m') \in \text{LR}$ .

If  $\mathcal{A}$  queries  $\text{Corrupt}(\text{nym})$  on input  $\text{nym} \in \text{N}$ :

– set  $\text{C} \leftarrow \text{C} \cup \{\text{nym}\}$

– return  $\text{gsk}[\text{nym}]$

If  $\mathcal{A}$  queries  $B'(\text{dsnym})$  on input  $\text{dsnym} \in W$ :

– set  $B \leftarrow B \cup \{\text{dsnym}\}$

If  $\mathcal{A}$  queries  $\text{NymSig}'(\text{dsnym}, \text{dpk}, m)$

on input on input  $\text{dsnym} \in W \setminus B$ ,  $\text{dpk} \in D$  and message  $m$ :

– set  $S \leftarrow S \cup \{(\text{dsnym}, \text{dpk}, m)\}$

– find  $\text{nym} \in N$  such that  $\text{dsnym} = \text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk})$

– return  $\text{NymSig}(\text{dsnym}, \text{gsk}[\text{nym}], \text{dpk}, m)$

If  $\mathcal{A}$  queries  $\text{LoR}(\text{dsnym}_0, \text{dsnym}_1, \text{dpk}, m)$

on input  $\text{dsnym}_0, \text{dsnym}_1 \in W \setminus B$ ,  $\text{dpk} \in D$ , and message  $m$ :

– set  $LR \leftarrow LR \cup \{(\{\text{dsnym}_0, \text{dsnym}_1\}, \text{dpk}, m)\}$

– find  $\text{nym}_0, \text{nym}_1 \in N \setminus C$  such that

$\text{dsnym}_i = \text{NymDSGen}(\text{nym}_i, \text{gsk}[\text{nym}_i], \text{dpk})$  for  $i = 0, 1$

– return  $\perp$  if no such  $\text{nym}_0, \text{nym}_1$  exist,

else return  $\text{NymSig}(\text{dsnym}_b, \text{gsk}[\text{nym}_b], \text{dpk}, m)$

The probability is taken over all coin tosses of  $\text{NymKGen}$ ,  $\text{NymSig}$ , and  $\mathcal{A}$ , and the choice of  $b$ .

We note that the adversary in our game always accesses oracles through their domain-specific pseudonyms. This is possible since  $\mathcal{A}$  knows the set  $W$  of such pseudonyms (but not the relation to the pseudonyms  $\text{nym}$ ). Having this list at the outset is motivated by the fact that the adversary can potentially collect such domain-specific pseudonyms when acting as a domain holder, where it gets to learn the domain-specific pseudonyms and signatures under these pseudonyms. Note that this also allows to link domain-specific pseudonyms  $\text{dsnym}$  to domain keys  $\text{dpk}$ , hence we give  $W \odot D$  as additional input. This also implies that we cannot grant the adversary access to another signature oracle which it can provide  $\text{nym}, \text{dpk}, m$  to get a signature for  $m$  under the corresponding  $\text{dsnym}$ ; it would be easy to check for the validity of the signature under the domain-specific pseudonym with the help of  $W$  and to link  $\text{dsnym}$  to  $\text{nym}$ . In other words, one can link pseudonyms to their domain-specific pseudonyms given a signature under the pseudonym for the respective domain.

For an adaptive version, the adversary may interleave Corrupt queries with the other oracle queries arbitrarily. Then, we must ensure that no nym in a Corrupt-query has appeared in an LoR-query before, declaring the adversary to lose if this is the case.

### 10.2.2. Unforgeability

Unforgeability of domain-specific pseudonymous signatures follows the basic paradigm for regular signatures: It should be infeasible to create a valid signature on behalf of an honest pseudonym for a previously unsigned message. This should even hold if the adversary knows the group manager's secret key (but not the user's secret key, else trivial attacks would be possible). Since for unforgeability we do not need to hide the link between pseudonyms and domain-specific pseudonyms, we assume that the adversary simply knows the tuples  $(\text{dsnym}, \text{nym}, \text{dpk})$  of domain-specific pseudonyms, and corresponding pseudonyms and domain keys. Below we say that the adversary wins if it manages to forge a signature under a domain-specific pseudonym  $\text{dsnym}^*$  which is potentially derived from *some* pseudonym  $\text{nym}$  in *some* domain  $\text{dpk}$ ; but the adversary does not to specify these values.

Our notion of unforgeability is weaker than the non-frameability property of group signatures in the sense that, even though the adversary may know the group manager's secret key, it must not collaborate with the group manager during generation. We again consider only the version of static corruptions, although here it is straightforward to capture adaptive corruptions by giving the adversary simply the corruption oracle in the second phase, too.

**Definition 10.2.3 (Unforgeability)** *A domain-specific pseudonymous signature scheme  $\mathcal{NYS} = (\text{NymKGen}, \text{NymDSGen}, \text{NymSig}, \text{NymVf})$  is  $(n, d, t, q, \epsilon)$ -unforgeable if any algorithm  $\mathcal{A}$ , running in time  $t$  and making at most  $q$  signing queries, makes the following experiment output 1 with probability at most  $\epsilon$ :*

**Experiment**  $\text{Unforge}_{\mathcal{A}}^{\mathcal{NYS}}(\kappa, n, d)$

$$\begin{aligned} & S, B, C, N, D \leftarrow \emptyset \\ & (\text{gpk}, \text{gmsk}, \{\text{gsk}[\text{nym}]\}_n, \{\text{nym}\}_n, \{\text{dpk}\}_d) \leftarrow \text{NymKGen}(1^\kappa, 1^n, 1^d) \\ & N = \{\text{nym}\}_n, \text{ and } D = \{\text{dpk}\}_d \\ & W \odot N \odot D = \{(\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk}), \text{nym}, \text{dpk}) \mid \text{nym} \in N, \text{dpk} \in D\} \\ & st \leftarrow \mathcal{A}^{\text{Corrupt}}(\text{gpk}, \text{gmsk}, W \odot N \odot D) \\ & (m^*, \sigma^*, \text{dsnym}^*) \leftarrow \mathcal{A}^{B', \text{NymSig}', \text{Corrupt}}(st) \end{aligned}$$

Output 1 iff there are  $\text{nym}^* \in N \setminus C$  and  $\text{dpk}^* \in D$  such that

- (a)  $\text{NymDSGen}(\text{nym}^*, \text{gsk}[\text{nym}^*], \text{dpk}^*) = \text{dsnym}^*$ , and
- (b)  $\text{NymVf}(\text{gpk}, \text{dsnym}^*, \text{dpk}^*, m^*, \sigma^*, B) = 1$ , and
- (c)  $(\text{dsnym}^*, \text{dpk}^*, m^*) \notin S$ .

If  $\mathcal{A}$  queries  $\text{Corrupt}(\text{nym})$  on input  $\text{nym} \in N$

- set  $C \leftarrow C \cup \{\text{nym}\}$
- return  $\text{gsk}[\text{nym}]$

If  $\mathcal{A}$  queries  $B'(\text{dsnym})$  on input  $\text{dsnym} \in W$

- set  $B \leftarrow B \cup \{\text{dsnym}\}$

If  $\mathcal{A}$  queries  $\text{NymSig}'(\text{dsnym}, \text{dpk}, m)$  on input  $\text{dsnym} \in W \setminus B$ ,  $\text{dpk} \in D$ , and message  $m$

- set  $S \leftarrow S \cup \{(\text{dsnym}, \text{dpk}, m)\}$
- find  $\text{nym} \in N$  such that  $\text{dsnym} = \text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk})$
- return  $\text{NymSig}(\text{dsnym}, \text{gsk}[\text{nym}], \text{dpk}, m)$

The probability is taken over all coin tosses of  $\text{NymKGen}$ ,  $\text{NymSig}$ , and  $\mathcal{A}$ .

Note that conditions (a) and (c) also imply that domain-specific pseudonyms of different users (within a domain) cannot collide, except with negligible probability. Otherwise, the adversary may corrupt one of the two parties, and any signature created under the domain-specific pseudonym of the one party would immediately constitute a forgery under the other party's pseudonym. (In the above model it would then be more appropriate to let the adversary in calls to  $\text{NymSig}'$  also specify  $\text{nym}$ , instead of searching for it; since the adversary knows the list  $W \odot N \odot D$  it can look this value up.)

### 10.2.3. Seclusiveness

Seclusiveness considers the case that the verifier would accept a signature under a domain-specific pseudonym which has not been created by the authority. Note that this assumes that only the blacklisted domain-specific pseudonyms are available, but not the universe of all created pseudonyms. This is indeed a valid assumption, following the suggestion in [BSI10] about revocation for pseudonyms through blacklists, with no intention for whitelists.

It is also clear that, unlike in case of unforgeability, we thus cannot allow the adversary to know the manager's secret key; else generating keys for additional users would be easy.

As in unforgeability, we again consider only the version of static corruptions. One captures adaptive corruptions by giving the adversary simply the corruption oracle in the second phase, too.

**Definition 10.2.4 (Seclusiveness)** *A domain-specific pseudonymous signature scheme  $\mathcal{NYS} = (\text{NymKGen}, \text{NymDSGen}, \text{NymSig}, \text{NymVf})$  is  $(n, d, t, Q, \varepsilon)$ -secluding with  $Q = (q_c, q_s)$  if any algorithm  $\mathcal{A}$ , running in time  $t$  and making at most  $q_s$  signing queries and  $q_c$  corruption queries, makes the following experiment output 1 with probability at most  $\varepsilon$ :*

**Experiment**  $\text{Sec}_{\mathcal{A}}^{\mathcal{NYS}}(\kappa, n, d)$

- $W, B, C, N, D \leftarrow \emptyset$
- $(\text{gpk}, \text{gmsk}, \{\text{gsk}[\text{nym}]\}_n, \{\text{nym}\}_n, \{\text{dpk}\}_d) \leftarrow \text{NymKGen}(1^\kappa, 1^n, 1^d)$
- $N = \{\text{nym}\}_n$  and  $D = \{\text{dpk}\}_d$
- $W = \{\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk}) \mid \text{nym} \in N, \text{dpk} \in D\}$
- $W \odot N \odot D$
- $\quad := \{(\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk}), \text{nym}, \text{dpk}) \mid \text{nym} \in N, \text{dpk} \in D\}$
- $st \leftarrow \mathcal{A}^{\text{Corrupt}}(\text{gpk}, W \odot N \odot D)$
- $(m^*, \sigma^*, \text{dsnym}^*) \leftarrow \mathcal{A}^{B', \text{NymSig}'}(st)$
- Output 1 iff there exists  $\text{dpk}^* \in D$  such that
  - (a)  $\text{NymVf}(\text{gpk}, \text{dsnym}^*, \text{dpk}^*, m^*, \sigma^*, B) = 1$
  - (b)  $\text{dsnym}^* \notin W$

If  $\mathcal{A}$  queries  $\text{Corrupt}(\text{nym})$  on input  $\text{nym} \in N$

- set  $C \leftarrow C \cup \{\text{nym}\}$
- return  $\text{gsk}[\text{nym}]$

If  $\mathcal{A}$  queries  $B'(\text{dsnym})$  on input  $\text{dsnym} \in W$

- set  $B \leftarrow B \cup \{\text{dsnym}\}$

If  $\mathcal{A}$  queries  $\text{NymSig}'(\text{dsnym}, \text{dpk}, m)$  on input  $\text{dsnym} \in W \setminus B$ ,  $\text{dpk} \in D$ , and message  $m$

- set  $S \leftarrow S \cup \{(\text{dsnym}, \text{dpk}, m)\}$
- find  $\text{nym} \in N$  such that  $\text{dsnym} = \text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk})$
- return  $\text{NymSig}(\text{dsnym}, \text{gsk}[\text{nym}], \text{dpk}, m)$

The probability is taken over all coin tosses of  $\text{NymKGen}$ ,  $\text{NymSig}$ , and  $\mathcal{A}$ .

### 10.3. Construction

The idea of the discrete-log based construction is as follows: The group manager will hold two generators  $g_1, g_2 = g_1^z$  with  $z \in \text{gmsk}$  for which it knows the discrete log with respect to each other. In addition, it will hold a public key  $y = g_1^x$ , such that it can easily compute many pairs  $(x_1, x_2)$  such that  $y = g_1^{x_1} g_2^{x_2}$  with the help of  $z$ ; this is the trapdoor property of such values [Ped92, BCC88]. Each user pseudonym  $\text{nym}$  will receive one of these pairs as its secret key  $\text{gsk}[\text{nym}]$ . The domain parameters are given by values  $\text{dpk} = g^r$ . A user can then compute the domain-specific pseudonym as  $\text{dsnym} = \text{dpk}^{x_1}$ .

To sign a message the user can then use common discrete-log based protocols (in the random-oracle model) to show that (a) it knows the discrete-log  $x_1$  of  $\text{dsnym}$  with respect to  $\text{dpk}$ , and (b) it knows a matching value  $x_2$  to this discrete logarithm  $x_1$  such that the pair forms a representation of  $y$ . Essentially, this is accomplished by running the non-interactive version of the Okamoto proof of knowledge [Oka93] for  $x_1, x_2$  and  $y$  to base  $g_1, g_2$ , where the  $x_1$ -part can simultaneously be used to show knowledge of  $x_1$  of  $\text{dsnym}$  with respect to base  $\text{dpk}$ . As usual, the message to be signed enters the hash computations.

**Construction 10.3.1** *The construction of the domain-specific pseudonymous signature scheme  $\mathcal{NYS} = (\text{NymKGen}, \text{NymDSGen}, \text{NymSig}, \text{NymVf})$  is as follows:*

$\text{NymKGen}(1^\kappa, 1^n, 1^d)$ : Let  $\mathcal{G} = \langle g \rangle$  be a (public) cyclic group of prime order  $q$ . We also assume a public hash function  $H$ , modeled as a random oracle in the security proofs. Choose  $z \in_R \mathbb{Z}_q$  randomly and calculate  $g_1 := g$  and  $g_2 := g^z$ . Define  $\text{gpk} := g_1^x$  for random  $x \in_R \mathbb{Z}_q$ . To generate the secrets for the pseudonyms choose  $n$  random elements  $x_{2,1}, \dots, x_{2,n} \in_R \mathbb{Z}_q^*$  and calculate  $x_{1,i} = x - z \cdot x_{2,i}$  for  $i = 1, 2, \dots, n$ . Define  $\text{gsk}[i] := (x_{1,i}, x_{2,i})$ . By  $x_j$  we denote the  $x_{j,i}$  when pseudonym  $i$  is clear from context. For the domain-parameters pick random  $r_1, \dots, r_d \in_R \mathbb{Z}_q^*$  and define  $\text{dpk}_i := g^{r_i}$  for  $i = 1, \dots, d$ . Store  $z$  in  $\text{gmsk}$ . (Note that once the values  $\text{gsk}[\cdot]$  have been output resp. given to the users, the group manager deletes them.)

$\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk})$ : Compute and output the domain-specific pseudonym  $\text{nym}[\text{dpk}] := \text{dpk}^{x_1}$ , which is also sometimes denoted as  $\text{dsnym}$  when  $\text{nym}$  and  $\text{dpk}$  are known from context.

## 10. Domain-Specific Pseudonymous Signatures

---

$\text{NymSig}(\text{dsnym}, \text{gsk}[\text{nym}], \text{dpk}, m)$ : Let  $a_1 = g_1^{t_1} \cdot g_2^{t_2}$  and  $a_2 = \text{dpk}^{t_1}$ , for random  $t_1, t_2 \in_R \mathbb{Z}_q$ . Compute  $c = H(\text{dpk}, \text{dsnym}, a_1, a_2, m)$ . Let  $s_1 = t_1 - c \cdot x_1$  and  $s_2 = t_2 - c \cdot x_2$ . Then, output  $\sigma = (c, s_1, s_2)$ . (Note that in the Restricted-Identification protocol (cf. Chapter 7) the user also sends  $\text{dsnym}$  which we can include here in the signature, in order to match the protocol description.)

$\text{NymVf}(\text{gpk}, \text{dsnym}, \text{dpk}, m, \sigma, B)$ : To verify a signature perform the following steps:

1. Parse  $(c, s_1, s_2) \leftarrow \sigma$ .
2. Let  $a_1 = y^c \cdot g_1^{s_1} \cdot g_2^{s_2}$  and  $a_2 = \text{dsnym}^c \cdot \text{dpk}^{s_1}$ .
3. Output 1 iff  $c = H(\text{dpk}, \text{dsnym}, a_1, a_2, m)$  and  $\text{dsnym} \notin B$ .

**REVOCAION MECHANISMS.** We presented our construction above in terms of blacklisting, revoking fraudulent domain-specific pseudonyms by listing them explicitly. Alternatively, and our definitions and constructions are robust in this regard, one can use a whitelisting approach to list valid entries only. To represent the whitelisting approach in our framework any delisted domain-specific pseudonym from  $W$  will be put in  $B$ , such that  $W \setminus B$  corresponds to the set of currently whitelisted entries. Checking for whitelisting thus corresponds to verifying that the entry is in  $W \setminus B$  in our framework.

Blacklisting and whitelisting is performed by calculating the domain-specific pseudonym  $\text{dsnym}$  for domain  $\text{dpk}$ , but without knowledge of the private keys  $x_1$  and  $x_2$ . One important difference between black- and whitelisting is that whitelisting allows to retain security even if the authority's secret key  $z$  is compromised. If whitelisting is used it is not strictly required to keep  $z$  secret. While an attacker would be able to construct valid private keys  $(x_1, x_2)$  corresponding to the group public key  $y$ , the corresponding pseudonyms would not be listed on the whitelist and thus, the signatures would be rejected. Therefore, the attacker would have to find corresponding private keys for given pseudonyms on the whitelist, which in turn would require to calculate discrete logarithms.

### 10.4. Security Analysis

Our proofs work in the random-oracle model, and thus, an adversary may also query a random hash function oracle. By  $q_h$ , we denote the maximum number of queries to hash function oracle made by the adversary.

### 10.4.1. Cross-Domain Anonymity

Informally, the protocol is cross-domain anonymous (with respect to static corruptions) because the domain-specific pseudonyms appear to be random to the adversary under the decisional Diffie–Hellman assumption. Below we write  $t' \approx t$  to denote the fact that  $t'$  is essentially the same running time as  $t$ , except for minor administrative overhead.

**Theorem 10.4.1** *Assume the DDH problem is  $(t, \varepsilon)$ -hard. Then, the domain-specific pseudonymous signature scheme  $\mathcal{NYS}$  of Section 10.3 is  $(n, d, t', Q, \varepsilon')$  cross-domain anonymous with  $Q = (q_c, q_s, q_t, q_h)$ , where*

$$\varepsilon' \leq nd\varepsilon + \frac{(q_s + q_t + q_h)(q_s + q_t)}{q^2}$$

and  $t' \approx t$ . This holds in the random-oracle model.

*Proof.* We combine the ideas of pseudorandom synthesizers of Naor and Reingold [NR97] with the simulation soundness of the non-interactive zero-knowledge proofs (aka. signatures) in the random-oracle model. That is, assume the original attack of the adversary on our protocol. In a first game hop we replace the actual signature computations in LoR and NymSig' queries by simulated signatures (via programming the random oracle). See [PSoo] for details. This strategy is valid if programming the hash values in such computations have not appeared in previous hash queries of the adversary, nor in previous signature queries. However, since the random group elements  $a_1, a_2$  enter the hash evaluations, the probability that an input collision occurs in any of the at most  $q_s + q_t$  signature generations, is at most  $(q_s + q_t + q_h)(q_s + q_t)/q^2$ . Given that no such collision occurs the simulation is perfectly indistinguishable, such that the adversary's success probability cannot increase by more than this term.

Note that after this game hop, we can create valid signatures on behalf of users without knowing the secret keys. In the next game hop, we replace the domain-specific pseudonyms  $dsnym$  in LoR queries by random group elements (but in a consistent way). That is, whenever we are supposed to use  $dsnym$  we instead use a new random element  $dsnym' \leftarrow \mathcal{G}$ , unless  $nym$  in combination with  $dpk$  has been used before, either in a signature request or an LoR-query, in which case we use again the previously generated random value  $dsnym'$ .

We claim that, by the DDH assumption, this hop cannot increase the adversary's success probability noticeably. To this end, we briefly recall the notion of a pseudorandom synthesizer in [NR97]. The pseudorandom synthesizer for the DH pairs is an  $a \times b$  matrix, with the rows labeled by values  $g^{x_i}$  and the columns labeled by  $g^{r_j}$ , and entries at position  $i, j$  set to  $g^{x_i r_j}$ , such that this matrix is indistinguishable from an  $a \times b$  matrix of independent and random group elements, even if the row and column labels  $g^{x_i}$  and  $g^{r_j}$  are given. In a sense, the matrix entries are correlated but still look random. As discussed in [NR97] this holds in our case under the DDH assumption. In fact, it allows for a reduction to the DDH problem with a loss of a factor  $ab$  where, in our case, after the initial corruption, there are at most  $ab \leq nd$  entries of honest users.

In the next game hop, we always use the left domain-specific pseudonym  $dsnym_0$  in LoR queries, independently of the value of  $b$ . We stress that, in case of  $b = 1$ , this does not change the adversary's success probability at all. Assume from now on that  $b = 0$ . Note that each LoR-query about  $(dsnym_0, dsnym_1, dpk, m)$  is answered by a random element, just as it would be for  $b = 1$ . All other LoR queries involving  $dpk$  can only be about the same pair  $(dsnym_0, dsnym_1)$  in this order, in reverse order  $(dsnym_1, dsnym_0)$ , or for distinct entries. In the first case, we would answer again consistently with the (wrong) random element, in the second case, we would switch to the other random element, and in the third case use an independent random value. This behavior, however, is identical to the case  $b = 0$  from the adversary's point of view. Similarly, the adversary cannot make any signature request for  $(dsnym_0, dpk)$  nor  $(dsnym_1, dpk)$  without losing. It follows that such signature requests do not depend on the bit  $b$ . Hence, the probability of the experiment returning 1 does not change.

In the final game, the adversary's success probability is independent of  $b$ , and the adversary cannot win with probability more than  $\frac{1}{2}$ . Collecting all probabilities from the game hops yields the claimed bound.  $\square$

**ANONYMITY OF RESTRICTED IDENTIFICATION.** Recall that in the basic version of the restricted identification protocol, the user merely shows the domain-specific pseudonym  $dsnym$  to the service provider (who checks that this value has not been revoked yet). Anonymity of this solution follows from the proof above under the DDH assumption alone, noting that we do not need to

simulate the additional signatures in the random-oracle model.<sup>2</sup>

### 10.4.2. Unforgeability

**Theorem 10.4.2** *Assume the DL problem is  $(t, \varepsilon)$ -hard on  $\mathcal{G}$ , then the domain-specific pseudonymous signature scheme  $\mathcal{NYS}$  of Section 10.3 is  $(n, d, t', Q, \varepsilon')$ -unforgeable with  $Q = (q_s, q_h)$ , where*

$$\varepsilon' \approx (2q)^{-1}(q_h - \sqrt{q_h} \sqrt{\delta\varepsilon + (2\delta/q)(q_s + q_h)^2 + q_h})$$

and  $t' \approx t$  with  $\delta = 4ndq^2$ . This holds in the random-oracle model.

*Proof.* We are given an adversary  $\mathcal{A}$  which wins in the unforgeability game of  $\mathcal{NYS}$ , in which  $\mathcal{A}$  outputs a fresh signature under domain-specific pseudonym  $\text{dsnym}$ . Similarly to the proof of Schnorr signatures [PSoo], we leverage the Forking Lemma, in order to obtain two related forgeries  $(c, s_1, s_2), (c', s'_1, s'_2)$  from which we can extract the witness (resp. discrete logarithm) from the challenge. The DL game asks for the discrete logarithm  $a$  of an element  $A := g_{dl}^a$  from a presumably DL-hard group  $\mathcal{G}_{dl}$ .

We describe first the framework for interacting with  $\mathcal{A}$ .

**Setup.** We obtain  $(\text{gpk}, \text{gmsk}, \{\text{gsk}[\text{nym}]\}_n, \{\text{nym}\}_n, \{\text{dpk}\}_d)$  by following the NymKGen algorithm on input  $(1^k, 1^n, 1^d)$  except that group  $\mathcal{G}$  is chosen as given by the DL game, i.e.,  $\mathcal{G} := \mathcal{G}_{DH}$ . We choose a random  $\text{nym}^*$  from  $\{\text{nym}\}_n$  and set  $\text{gsk}[\text{nym}^*] := *$  meaning that  $\text{gsk}[\text{nym}^*]$  is unknown. Similarly, we choose random  $\text{dpk}^* \in_R \{\text{dpk}\}_d$  and set  $\text{dpk}^* = g_{dl}$ . We replace all other domain parameters  $\text{dpk}_i \in \{\text{dpk}\}_d$  by setting  $\text{dpk}_i = g_{dl}^{r_i}$  where  $r_i$  is sampled uniformly from  $\mathbb{Z}_q^*$ . Next, we prepare the set

$W \odot N \odot D$

$$:= \{(\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk}), \text{nym}, \text{dpk}) \mid \text{nym} \in N, \text{dpk} \in D\}$$

for  $\mathcal{A}$ . However, we replace elements

$$(\text{NymDSGen}(\text{nym}^*, \text{gsk}[\text{nym}^*], \text{dpk}), \text{nym}^*, \text{dpk}) \in W \odot N \odot D$$

by  $(A, \text{nym}^*, \text{dpk})$  if  $\text{dpk} = \text{dpk}^*$ , or else by  $(A^{r_i}, \text{nym}^*, \text{dpk})$ . We give  $\mathcal{A}$  as input  $(\text{gpk}, z, W \odot N \odot D)$ .

<sup>2</sup>The specification actually lets the user send a hash value of  $\text{dsnym}$ . This does not affect the discussion, though.

**Hash Queries.** We answer with values uniformly sampled from the range of the hash function but keep being consistent and store input/output in a list. In case, we rewind the adversary, we remove all input/output tuples from the list of queries up to the point of rewinding.

**Signature Queries.** Upon input a domain-specific pseudonym  $\text{dsnym}$  and a message  $m$ , if there exists no element  $(\text{dsnym}, \text{nym}^*, *) \in W \odot N \odot D$ , we perform NymSig on these values honestly and return the output of NymSig. Else, we simulate a signature  $\sigma = (c, s_1, s_2)$  on  $m$  for domain-specific pseudonym  $\text{dsnym}$  by the setting the following.

- $s_1, s_2 \xleftarrow{\$} \mathcal{G}$  and  $c \xleftarrow{\$} \text{Range}(H)$
- $a_1 = y^c \cdot g_1^{s_1} \cdot g_2^{s_2}$  and  $a_2 = \text{dsnym}^c \cdot \text{dpk}^{s_1}$
- Program the random oracle such that upon input the tuple  $(\text{dpk}, \text{dsnym}, a_1, a_2, m)$ , it outputs  $c$ . If the oracle was queried before on that input, then we abort the simulation.

**Corrupt Queries.** Upon input pseudonym  $\text{nym} \neq \text{nym}^*$ , we output  $\text{gsk}[\text{nym}]$ . Upon input pseudonym  $\text{gpk}$ , we output  $z \in \text{gmsk}$ .

**Output.** At some point, adversary  $\mathcal{A}$  outputs a signature  $\sigma^* = (c, s_1, s_2)$  on message  $m$  under domain-specific pseudonym  $\text{dsnym}^*$ . If  $\text{dsnym}^*$  equals  $A$ , we rewind  $\mathcal{A}$  to the point where it queried the hashing oracle on input  $(\text{dpk}, A, y^c \cdot g_1^{s_1} \cdot g_2^{s_2}, A \cdot \text{dpk}^{s_1}, m)$  and output a different, but randomly chosen value  $c' \neq c$ . The Forking Lemma states that we can produce a second related signature  $(c', s'_1, s'_2)$  where  $a_1, a_2$  are equally defined in both signatures.

Next, we argue that the simulation of  $\mathcal{A}$ 's environment is perfect, and given a forgery from  $\mathcal{A}$ , leveraging the Forking Lemma we successfully derive the secret key  $x_1$  from pseudonym  $\text{nym}^*$  from the two related signatures and, consequently, find the discrete logarithm for our challenge.

We follow all steps of NymKGen compulsory except for the choice of  $\mathcal{G}$ ,  $\text{nym}^*$ , and domain public keys  $\text{dpk}$ . The group  $\mathcal{G}$  is randomly chosen in the unforgeability game of  $\mathcal{NYS}$ , as well as  $\mathcal{G}_{dl}$  in the DL experiment, and thus, both groups are indistinguishable. The domain descriptions  $\{\text{dpk}\}_d$  are chosen from a uniform distribution in  $\mathcal{G}$  (resp.  $\mathcal{G}_{dl}$ ) and since the generator of the DL instance  $g_{dl}$  is a random element, setting  $\text{dpk}^* = g_{dl}$  and  $\text{dpk}_i = g_{dl}^{r_i}$  for random  $r_i$ 's, is unnoticeable by  $\mathcal{A}$ . We also pick one random pseudonym  $\text{nym}^*$  in the hope that  $\mathcal{A}$  will forge on behalf of this pseudonym. Consequently, we are sure in case it happens that  $\mathcal{A}$  does not ask for  $\text{nym}^*$ 's secret key, otherwise,  $\mathcal{A}$  cannot win the unforgeability game. Thus, setting  $\text{gsk}[\text{nym}^*] = *$  is again unnoticed by  $\mathcal{A}$ .

Signature queries can be easily answered due to the fact that we know all secrets but the one of pseudonym  $\text{nym}^*$ . However, we stress that the simulation described above outputs a signature on an arbitrary message  $m$  under public key  $\text{dpk}$  for the domain-specific pseudonym  $\text{dsnym}$  which is shown in Lemma 10.4.3. We embed our DL challenge in  $\text{dsnym}^* := A$  where  $\text{dsnym}^*$  corresponds to  $\text{nym}^*$  and  $\text{dpk}^*$ , again in the hope that  $\mathcal{A}$  will forge on behalf of  $\text{nym}^*$  under domain description  $\text{dpk}^*$ . This simulation is indistinguishable as  $A$  is uniformly distributed as a domain-specific pseudonym  $\text{dsnym}$  is given that  $\mathcal{A}$  does not hold the discrete logarithm of  $\text{dpk}$  or secret key  $\text{gsk}[\text{nym}^*]$ , respectively.

**Lemma 10.4.3** *The distribution of NymSig and the simulation of signatures as described above is computationally indistinguishable.*

*Proof.* First, we show the validity of a simulated signature  $(c, s_1, s_2)$  and afterwards, we look at the distribution of the underlying elements  $c, s_1$  and  $s_2$ .

**Validity.** The case where  $\text{dsnym}$  is not derived by  $\text{nym}^*$  is trivial, since we use the pseudonym's secret key for signing a message. Therefore, we just have to look at the case  $\text{dsnym}$  corresponding to  $\text{nym}^*$ .

The verification of a signature  $\sigma = (c, s_1, s_2)$  outputs 1, if and only if  $c = H(\text{dpk}, \text{dsnym}, y^c \cdot g_1^{s_1} \cdot g_2^{s_2}, \text{dsnym}^c \cdot \text{dpk}^{s_1}, m)$ . We have programmed the random oracle such that on input  $(\text{dpk}, \text{dsnym}, a_1, a_2, m)$  with  $a_1 = y^c \cdot g_1^{s_1} \cdot g_2^{s_2}$  and  $a_2 = \text{dsnym}^c \cdot (\text{dpk})^{s_1}$  it outputs  $c$ . Indeed,  $a_1$  and  $a_2$  are exactly composed as required. Thus, we have that the oracle outputs the correct hash value  $c$  and the verification succeeds.

**Distribution.** The case where  $\text{dsnym}$  is not derived by  $\text{nym}^*$  is again trivial.

The signature consists of elements  $(c, s_1, s_2)$ . In the signing algorithm NymSig the value  $c$  is uniformly drawn from the challenge space,  $s_1 = t_1 - c \cdot x_1$  and  $s_2 = t_2 - c \cdot x_2$  are uniformly distributed in the group  $\mathbb{Z}_q$  because the elements  $t_1, t_2$  are picked randomly from  $\mathbb{Z}_q$ .

In the simulation,  $s_1$  and  $s_2$  are randomly picked from  $\mathbb{Z}_p$ , and, therefore, are indistinguishable. We also choose  $c$  randomly from the challenge space.

This proves the lemma. □

We have shown so far, that we provide a perfect simulation of adversary  $\mathcal{A}$ 's environment if no collision occurs during the signature simulation. This

probability is upper bounded by  $(2q_s + 2q_h)^2/q$ . We still need to show that we leverage the output by  $\mathcal{A}$  to solve a hard instance of the DL problem.

In case  $\mathcal{A}$  forges on behalf of pseudonym  $\text{nym}^*$  under domain  $\text{dpk}^*$ , we know that the domain-specific pseudonym  $\text{dsnym}$  is our embedded DL challenge  $A = g_{dl}^a$ . The probability that this event to be happens is at least  $1/nd$ . A second related forgery can be obtained by using the Forking Lemma where we rewind  $\mathcal{A}$  up to the point in which the random oracle was queried on  $(\text{dpk}, A, y^c \cdot g_1^{s_1} \cdot g_2^{s_2}, A^c \cdot \text{dpk}^{s_1}, m)$  with the corresponding output  $c$ . Now, we give  $\mathcal{A}$  a distinct value  $c'$  as output by the oracle. Let denote the signature obtained by  $\mathcal{A}$  by  $\sigma = (c, s_1, s_2)$ . Then, Forking Lemma guarantees that we obtain a related forgery  $\sigma' = (c', s'_1, s'_2)$  where the underlying commitments  $a_1, a_2$  and  $a'_1, a'_2$  (resp.  $t_1, t_2$  and  $t'_1, t'_2$ ) are equal with probability  $\varepsilon'(\varepsilon'/q_h + 1/q)$  where  $\varepsilon'$  is the success probability for the forger. Here, we use the bound given by [BNo6]. Note that commitments of our signature schemes are omitted in the final signature for efficiency reasons. In [PSoo], the Forking Lemma was introduced for signatures with commitments being part of the signatures, but the authors mentioned that this is merely for simplicity, and for efficiency reason one might omit the commitment or the challenge, respectively.

Given both signatures  $\sigma, \sigma'$  we extract the discrete-log  $a$  of  $\text{dsnym} = A = g^a$  as follows. Given  $s_1 = t_1 - c \cdot a$  and  $s'_1 = t_1 - c' \cdot a$ , we have  $a = (s_1 - s'_1)/(c' - c)$ . Hence, we found the solution  $a$  for the DL instance  $A$ .

We require that  $\mathcal{A}$  succeeds to forge on behalf of pseudonym  $\text{nym}^*$  under domain public key  $\text{dpk}^*$  in the first signature. In addition, we loose a tightness factor due to Forking Lemma, which yields the probability to find the discrete logarithm of  $A$  at most  $\varepsilon = \varepsilon'/nd \cdot (\varepsilon'/q_h + 1/q) - (2q_s + 2q_h)^2/q$  where  $\varepsilon'$  is the success probability of  $\mathcal{A}$ .  $\square$

### 10.4.3. Seclusiveness

As remarked before, seclusiveness only holds as long as the adversary does not get a hold of the group manager's secret key. By construction, this means that the adversary can thus only corrupt one user, else  $z$  becomes known. When considering blacklisting for our construction, we stipulate this below by requiring that the secrets are stored securely in hardware, or, respectively, that the number of corrupt requests  $q_c$  is at most 1. If whitelisting is used instead, then we do not require any bound on the number of corruptions the

adversary can make, since learning  $z$  does not help the adversary to compute a domain-specific dsnym which is listed in the (still trustworthy) whitelist.

**Theorem 10.4.4** *Assume the DL problem is  $(t, \varepsilon)$ -hard on  $\mathcal{G}$ , then the domain-specific pseudonymous signature scheme of Section 10.3 is  $(n, d, t', Q, \varepsilon')$ -secluding with  $Q = (q_c, q_s, q_h)$ , where  $q_c = 1$ ,*

$$\varepsilon' \approx (2q)^{-1}(q_h - \sqrt{q_h} \sqrt{q_h + 4q^2\varepsilon + 4q^2\delta})$$

and  $t' \approx t$  with  $\delta = 2(q_s + q_h)/q$ . This holds in the random-oracle model.

*Proof.* We are given an adversary  $\mathcal{A}$  which wins in the seclusiveness game of  $\mathcal{NYS}$ . Here,  $\mathcal{A}$  outputs a signature under a domain-specific pseudonym dsnym to no corresponding identity nym  $\in \{\text{nym}\}_d$ . Intuitively, the proof works as follows. We are asked for the discrete-log  $a$  of an element  $A := g_{dl}^a$  from a presumably DL-hard group  $\mathcal{G}_{dl}$ . We embed  $A$  in generator  $g_2$  such that the group's master key  $z \in \text{gmsk}$  equals  $a$ . We are able to generate one secret key pair  $(x_1, x_2)$  satisfying  $x = x_1 + zx_2$  for unknown  $x, \text{gmsk}$ . Using the signature given by  $\mathcal{A}$  we can extract a second pair  $(x_1, x_2)$ . Those two key pairs suffice to disclose  $z$  (resp.  $a$ ) and, thus, we solve the DL problem.

We describe first the framework for interacting with  $\mathcal{A}$ .

**Setup.** We set  $\mathcal{G} = \mathcal{G}_{dl}$ ,  $g_1 = g_{dl}$  and  $g_2 = A$ . We generate  $\{\text{nym}\}_n, \{\text{dpk}\}_d$  following NymKGen. We pick random elements  $x_1, x_2 \in_R \mathcal{G}_{dl}$  and set  $\text{gpk} = g_1^{x_1} g_2^{x_2}$ . After  $\mathcal{A}$  selected one pseudonym  $\text{nym}^*$  to corrupt, we answer by  $\text{gsk}[\text{nym}^*] := (x_1, x_2)$ . We pick  $n$  random elements  $x_{1,1}, \dots, x_{1,n}$  and set  $\text{gsk}[\text{nym}_i] = (x_{1,i}, *)$  for  $i = 1, \dots, n$  with  $\text{nym}_i \neq \text{nym}^*$ . Next, we prepare the set

$$W \odot N \odot D$$

$$:= \{(\text{NymDSGen}(\text{nym}, \text{gsk}[\text{nym}], \text{dpk}), \text{nym}, \text{dpk}) \mid \text{nym} \in N, \text{dpk} \in D\}$$

for  $\mathcal{A}$ . Note that in our construction we merely need the values  $x_{1,i}$  to derive the domain-specific pseudonyms. Hence, we give  $\mathcal{A}$  as input  $(\text{gpk}, W \odot N \odot D)$ .

**Hash Queries.** The simulation of the random oracle is identical to the simulation in the case of unforgeability.

**Signature Queries.** The simulation of signatures is identical to the simulation in the case of unforgeability. Again we program the random oracle such that upon input  $(\text{dpk}, \text{dsnym}, a_1, a_2, m)$ , the oracle outputs  $c$ . If the oracle was queried before on that input, then we abort the simulation.

**Output.** At some point, adversary  $\mathcal{A}$  outputs a signature  $\sigma^* = (c, s_1, s_2)$  on message  $m$  under domain-specific pseudonym  $\text{dsnym}^*$ . If  $\text{dsnym}^*$  equals  $A$ , we rewind  $\mathcal{A}$  to the point where it queried the hashing oracle on input  $(\text{dpk}, A, y^c \cdot g_1^{s_1} \cdot g_2^{s_2}, A^c \cdot \text{dpk}^{s_1}, m)$  and we output a different but randomly chosen value  $c' \neq c$ . The Forking Lemma states that we can produce a second related signature  $(c', s'_1, s'_2)$  where  $a_1, a_2$  are equally defined in both signatures.

Next, we argue that the simulation of  $\mathcal{A}$ 's environment is perfect, and given a forgery from  $\mathcal{A}$ , leveraging the Forking Lemma, we successfully derive a secret key  $(x_1^*, x_2^*)$  such that  $x = x_1^* + zx_2^*$  from the two related signatures and, consequently, find the discrete logarithm  $z = a$  for our challenge  $A$ .

The group  $\mathcal{G}$  is randomly chosen in the seclusiveness game of  $\mathcal{NYMS}$  as well as  $\mathcal{G}_{dl}$  in the DL experiment, and thus, both groups are indistinguishable. The values  $\{\text{nym}\}_n, \{\text{dpk}\}_d$  are computed compulsorily. The group public key  $\text{gpk}$  is obtained by selecting one secret pair in advance. All other secret keys cannot be specified since  $z$  is unknown and thus, we cannot produce key pairs. However, we can select randomly one part  $x_{1,i}$ , and leave the second part  $x_{2,i}$  as unknown.

The simulation above produces signatures indistinguishable from genuine signatures. We possess the first part of the secret keys, such that we are able to compute the domain-specific pseudonyms. The validity and distribution of the simulated signatures are proven in Lemma 10.4.3.

We have shown so far that we provide a perfect simulation of adversary  $\mathcal{A}$ 's environment if no collision in the signature simulation occurs. This probability is upper bounded by  $(2q_s + 2q_h)^2 / q$ . We still need to show that we leverage the output by  $\mathcal{A}$  to solve a hard instance of the DL problem.

At some point, adversary  $\mathcal{A}$  outputs a signature  $\sigma = (c, s_1, s_2)$  under domain-specific pseudonym  $\text{dsnym}$ . A second related forgery can be obtained by using the Forking Lemma where we rewind  $\mathcal{A}$  up to the point in which the random oracle was queried on  $(\text{dpk}, \text{dsnym}, y^c \cdot g_1^{s_1} \cdot g_2^{s_2}, \text{dsnym}^c \cdot \text{dpk}^{s_1}, m)$  with the corresponding output  $c$ . Now, we give  $\mathcal{A}$  a distinct value  $c'$  as output by the oracle. The Forking Lemma guarantees that we obtain a correlated forgery  $\sigma' = (c', s'_1, s'_2)$  where the underlying commitments  $a_1, a_2$  and  $a'_1, a'_2$  are equal. Given both signatures  $\sigma, \sigma'$  we extract the discrete-log  $a$  of  $g_2 = A = g_1^a$  as follows. Given

$$s_1 = t_1 - c \cdot x_1^* \quad s'_1 = t_1 - c' \cdot x_1^* \quad s_2 = t_2 - c \cdot x_2^* \quad s'_2 = t_2 - c' \cdot x_2^*$$

we have  $x_1^* = (s_1 - s'_1)/(c' - c)$  and  $x_2^* = (s_2 - s'_2)/(c' - c)$ . Hence, we found a second key pair  $(x_1^*, x_2^*)$  satisfying  $x = x_1^* + zx_2^*$ . We obtain  $z$  by  $z = (x_1^* - x_1)/(x_2 - x_2^*)$  where  $x_1, x_2$  is the secret key pair we chose in the setup phase for pseudonym  $\text{nym}^*$ . Hence, since  $z = a$ , we have found the solution  $a$  for the DL instance  $A$ . We require that  $\mathcal{A}$  succeeds to forge twice, which yields the probability to find the discrete logarithm of  $A$  at most  $\varepsilon = \varepsilon'(\varepsilon'/q_h + 1/q) - (2q_s + 2q_h)^2/q$  where  $\varepsilon'$  is the success probability of  $\mathcal{A}$ .  $\square$



## Bibliography

- [ACCP08] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Efficient Two-Party Password-Based Key Exchange Protocols in the UC Framework. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 335–351, San Francisco, CA, USA, April 7–11, 2008. Springer, Berlin, Germany.
- [ADDV12] Giuseppe Ateniese, Özgür Dagdelen, Ivan Damgard, and Daniele Venturi. Entangled cloud storage. *Cryptology ePrint Archive*, Report 2012/511, 2012. <http://eprint.iacr.org/>.
- [ADK10] Sami Alsouri, Özgür Dagdelen, and Stefan Katzenbeisser. Group-based attestation: Enhancing privacy and management in remote attestation. In *TRUST*, pages 63–77, 2010.
- [ADV<sup>+</sup>12] Sidi Mohamed El Yousfi Alaoui, Özgür Dagdelen, Pascal Véron, David Galindo, and Pierre-Louis Cayrel. Extended Security Arguments for Signature Schemes. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 12: 5th International Conference on Cryptology in Africa*, volume 7374 of *Lecture Notes in Computer Science*, pages 19–34, Ifrance, Morocco, July 10–12, 2012. Springer, Berlin, Germany.
- [AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84, Les Diablerets, Switzerland, January 23–26, 2005. Springer, Berlin, Germany.
- [ANS86] ANSI X9.19. American National Standard for Financial Institution Retail Message Authentication. ASC X9 Secretariat - American Bankers Association, 1986.

- [AS09] Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 70–89, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.
- [BBD<sup>+</sup>10] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable Signatures for Tree-Structured Data: Definitions and Constructions. In Jianying Zhou and Moti Yung, editors, *ACNS 10: 8th International Conference on Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 87–104, Beijing, China, June 22–25, 2010. Springer, Berlin, Germany.
- [BC93] Stefan Brands and David Chaum. Distance-Bounding Protocols (Extended Abstract). In Tor Helleseht, editor, *Advances in Cryptology – EUROCRYPT’93*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359, Lofthus, Norway, May 23–27, 1993. Springer, Berlin, Germany.
- [BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum Disclosure Proofs of Knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
- [BCI<sup>+</sup>10] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient Indifferentiable Hashing into Ordinary Elliptic Curves. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 237–254, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Berlin, Germany.
- [BCKLo8] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and Noninteractive Anonymous Credentials. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.
- [BCNPo8] Colin Boyd, Yvonne Cliff, Juan Gonzalez Nieto, and Kenneth G. Paterson. Efficient One-Round Key Exchange in the Standard Model. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors,

- ACISP 08: 13th Australasian Conference on Information Security and Privacy*, volume 5107 of *Lecture Notes in Computer Science*, pages 69–83, Wollongong, Australia, July 7–9, 2008. Springer, Berlin, Germany.
- [BCP13] Julien Bringer, Herve Chabanne, and Alain Patey. Collusion-Resistant Domain-Specific Pseudonymous Signatures. *Cryptology ePrint Archive*, Report 2013/182, 2013. <http://eprint.iacr.org/>.
- [BDF<sup>+</sup>11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random Oracles in a Quantum World. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69, Seoul, South Korea, December 4–8, 2011. Springer, Berlin, Germany.
- [BDF12] Christina Brzuska, Özgür Dagdelen, and Marc Fischlin. TLS, PACE, and EAC: A Cryptographic View at Modern Key Exchange Protocols. In *Sicherheit*, pages 71–82, 2012.
- [BDFK12a] Jens Bender, Özgür Dagdelen, Marc Fischlin, and Dennis Kügler. Domain-Specific Pseudonymous Signatures for the German Identity Card. In Dieter Gollmann and Felix C. Freiling, editors, *ISC 2012: 15th International Conference on Information Security*, volume 7483 of *Lecture Notes in Computer Science*, pages 104–119, Passau, Germany, September 19–21, 2012. Springer, Berlin, Germany.
- [BDFK12b] Jens Bender, Özgür Dagdelen, Marc Fischlin, and Dennis Kügler. The PACE|AA Protocol for Machine Readable Travel Documents, and Its Security. In Angelos D. Keromytis, editor, *FC 2012: 16th International Conference on Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 344–358, Kralendijk, Bonaire, February 27 – March 2, 2012. Springer, Berlin, Germany.
- [BDJR97] Mihir Bellare, Anand Desai, Eric Jorjani, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

- [BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The KECCAK reference, January 2011. <http://keccak.noekeon.org/>.
- [Bel12] Mihir Bellare. Lecture Notes on “Introduction to Modern Cryptography” (CSE 107), 2012.
- [BF06] Alexandra Boldyreva and Marc Fischlin. On the Security of OAEP. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 210–225, Shanghai, China, December 3–7, 2006. Springer, Berlin, Germany.
- [BFG<sup>+</sup>11] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N.P. Smart, and B. Warinschi. Anonymous attestation with user-controlled linkability. *Cryptology ePrint Archive*, Report 2011/658, 2011. <http://eprint.iacr.org/>.
- [BFK09] Jens Bender, Marc Fischlin, and Dennis Kügler. Security Analysis of the PACE Key-Agreement Protocol. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio Agostino Ardagna, editors, *ISC 2009: 12th International Conference on Information Security*, volume 5735 of *Lecture Notes in Computer Science*, pages 33–48, Pisa, Italy, September 7–9, 2009. Springer, Berlin, Germany.
- [BFS<sup>+</sup>13] C. Brzuska, M. Fischlin, N.P. Smart, B. Warinschi, and S.C. Williams. Less is more: relaxed yet composable security notions for key exchange. *International Journal of Information Security*, pages 1–31, 2013.
- [BFWW11] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of Bellare-Rogaway key exchange protocols. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 11: 18th Conference on Computer and Communications Security*, pages 51–62, Chicago, Illinois, USA, October 17–21, 2011. ACM Press.
- [BK09] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.

- [BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.
- [BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.
- [BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371, Seoul, South Korea, December 4–8, 2011. Springer, Berlin, Germany.
- [BLMN11] Alex Biryukov, Mario Lamberger, Florian Mendel, and Ivica Nikolic. Second-Order Differential Collisions for Reduced SHA-256. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 270–287, Seoul, South Korea, December 4–8, 2011. Springer, Berlin, Germany.
- [BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer, Berlin, Germany.
- [BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany.
- [BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 390–399, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.

- [Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 1998. Invited paper.
- [BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Germany.
- [BPSVo8a] Carlo Blundo, Giuseppe Persiano, Ahmad-Reza Sadeghi, and Ivan Visconti. Improved Security Notions and Protocols for Non-transferable Identification. In Sushil Jajodia and Javier López, editors, *ESORICS 2008: 13th European Symposium on Research in Computer Security*, volume 5283 of *Lecture Notes in Computer Science*, pages 364–378, Málaga, Spain, October 6–8, 2008. Springer, Berlin, Germany.
- [BPSVo8b] Carlo Blundo, Giuseppe Persiano, Ahmad-Reza Sadeghi, and Ivan Visconti. Resettable and Non-Transferable Chip Authentication for E-Passports. In *RFIDSec08*, 2008.
- [BPWo4] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A General Composition Theorem for Secure Reactive Systems. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 336–354, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.
- [BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.
- [BR94a] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO’93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Berlin, Germany.

- [BR94b] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT’94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, May 9–12, 1994. Springer, Berlin, Germany.
- [BR96] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT’96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Germany.
- [BR00] John Black and Phillip Rogaway. CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 197–215, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.
- [Bra93] Stefan A. Brands. An Efficient Off-line Electronic Cash System Based On The Representation Problem. Technical report, Centrum voor Wiskunde en Informatica (CWI), 1993.
- [Bra00] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy*. The MIT Press, 2000.
- [BSI10] BSI. Advanced Security Mechanism for Machine Readable Travel Documents Extended Access Control (EAC). Technical Report (BSI-TR-03110) Version 2.05 Release Candidate, Bundesamt fuer Sicherheit in der Informationstechnik, 2010.
- [Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press.
- [CBH05] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 585–604, Chennai, India, December 4–8, 2005. Springer, Berlin, Germany.

- [CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.
- [CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press.
- [CGIP12] Jean-Sébastien Coron, Aline Gouget, Thomas Icart, and Pascal Paillier. Supplemental access control (pace v2): Security analysis of pace integrated mapping. In *Cryptography and Security*, pages 207–232, 2012.
- [Cha85] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28, October 1985.
- [CHK<sup>+</sup>05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally Composable Password-Based Key Exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.
- [CKo1] Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Germany.
- [CKMoo] Don Coppersmith, Lars R. Knudsen, and Chris J. Mitchell. Key Recovery and Forgery Attacks on the MacDES MAC Algorithm. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 184–196, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.
- [CLo1] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional

- Anonymity Revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Germany.
- [CLo2] Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.
- [CLo4] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.
- [CMo9] Jean-Sébastien Coron and Avradip Mandal. PSS Is Secure against Random Fault Attacks. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 653–666, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.
- [Coro2] Jean-Sébastien Coron. Optimal Security Proofs for PSS and Other Signature Schemes. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Berlin, Germany.
- [CPSo8] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.
- [Creo9a] Cas J. F. Cremers. Session-state Reveal Is Stronger Than Ephemeral Key Reveal: Attacking the NAXOS Authenticated Key Exchange Protocol. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*:

- 7th International Conference on Applied Cryptography and Network Security*, volume 5536 of *Lecture Notes in Computer Science*, pages 20–33, Paris-Rocquencourt, France, June 2–5, 2009. Springer, Berlin, Germany.
- [Cre09b] Cas J.F. Cremers. Formally and practically relating the CK, CK-HMQV, and eCK security models for authenticated key exchange. *Cryptology ePrint Archive*, Report 2009/253, 2009. <http://eprint.iacr.org/>.
- [Cre11] Cas Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 11: 6th Conference on Computer and Communications Security*, pages 80–91, Hong Kong, China, October 17–21, 2011. ACM Press.
- [CS10] Lassaad Cheikhrouhou and Werner Stephan. Meilensteinreport: Inductive Verification of PACE. Technical report, Deutsches Forschungszentrum fuer Kuenstliche Intelligenz GmbH, 2010.
- [CSD<sup>+</sup>12] Lassaad Cheikhrouhou, Werner Stephan, Özgür Dagdelen, Marc Fischlin, and Markus Ullmann. Merging the cryptographic security analysis and the algebraic-logic security proof of pace. In *Sicherheit*, pages 83–94, 2012.
- [DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
- [DF10] Özgür Dagdelen and Marc Fischlin. Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *ISC 2010: 13th International Conference on Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 54–68, Boca Raton, FL, USA, October 25–28, 2010. Springer, Berlin, Germany.
- [DF12] Özgür Dagdelen and Marc Fischlin. Unconditionally-secure universally composable password-based key-exchange based on one-time memory tokens. *Cryptology ePrint Archive*, Report 2012/537, 2012. <http://eprint.iacr.org/>.

- [DFG<sub>13a</sub>] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat–Shamir Transformation in a Quantum World. *Cryptology ePrint Archive*, Report 2013/245, 2013. <http://eprint.iacr.org/>.
- [DFG<sup>+</sup><sub>13b</sub>] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzura Marson, Arno Mittelbach, and Cristina Onete. A Cryptographic Analysis of OPACITY. *Cryptology ePrint Archive*, Report 2013/234, 2013. <http://eprint.iacr.org/>.
- [DFKO<sub>11</sub>] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A Formal Approach to Distance-Bounding RFID Protocols. In Xuejia Lai, Jianying Zhou, and Hui Li, editors, *ISC 2011: 14th International Conference on Information Security*, volume 7001 of *Lecture Notes in Computer Science*, pages 47–62, Xi’an, China, October 25–29, 2011. Springer, Berlin, Germany.
- [DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [DKSW<sub>09</sub>] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. Composability and On-Line Deniability of Authentication. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 146–162. Springer, Berlin, Germany, March 15–17, 2009.
- [DMV<sub>13</sub>] Özgür Dagdelen, Payman Mohassel, and Daniele Venturi. Rate-Limited Secure Function Evaluation: Definitions and Constructions. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 461–478. Springer, Berlin, Germany, 2013.
- [DS<sub>10</sub>] Özgür Dagdelen and Michael Schneider. Parallel Enumeration of Shortest Lattice Vectors. In Pasqua Di<sub>1</sub>/<sub>2</sub>Ambra, Mario Guaracino, and Domenico Talia, editors, *Euro-Par 2010 - Parallel Processing*, volume 6272 of *Lecture Notes in Computer Science*, pages 211–222. Springer Berlin Heidelberg, 2010.
- [DY<sub>81</sub>] D. Dolev and A. C. Yao. On the security of public key protocols. In *SFCS '81*, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society.

- [Fis07] Marc Fischlin. Anonymous Signatures Made Easy. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 31–42, Beijing, China, April 16–20, 2007. Springer, Berlin, Germany.
- [FL10] Marc Fischlin and Anja Lehmann. Delayed-Key Message Authentication for Streams. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 290–307, Zurich, Switzerland, February 9–11, 2010. Springer, Berlin, Germany.
- [FO13] Marc Fischlin and Cristina Onete. Subtle kinks in distance-bounding: an analysis of prominent protocols. In *WISec*, pages 195–206, 2013.
- [Ger10] Gerhard P. Hancke. Distance bounding publication database, 2010. <http://www.rfidblog.org.uk/db.html>.
- [GK10] Adam Groce and Jonathan Katz. A new framework for password-based authenticated key exchange. Cryptology ePrint Archive, Report 2010/147, 2010. <http://eprint.iacr.org/>.
- [GLRW10] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 56–75, Singapore, December 5–9, 2010. Springer, Berlin, Germany.
- [GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
- [GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261, Paris, France, May 14–16, 2001. Springer, Berlin, Germany.
- [GMP<sup>+</sup>08] Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally Composable Security Analysis of TLS. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec 2008: 2nd International Conference on*

- Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 313–327, Shanghai, China, October 31 – November 1, 2008. Springer, Berlin, Germany.
- [GMR06] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A Method for Making Password-Based Key Exchange Resilient to Server Compromise. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 142–159, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.
- [GNSN10] Martin Gagné, Shivaramkrishnan Narayan, and Reihaneh Safavi-Naini. Threshold Attribute-Based Signcryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 154–171, Amalfi, Italy, September 13–15, 2010. Springer, Berlin, Germany.
- [Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.
- [Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
- [Hei10a] Heise Online. CCC zeigt Sicherheitsprobleme beim elektronischen Personalausweis auf [Update], September 2010. <http://www.heise.de/newsticker/meldung/CCC-zeigt-Sicherheitsprobleme-beim-elektronischen-Personalausweis-auf-Update-1083649.html>.
- [Hei10b] Heise Online. Elektronischer Personalausweis: Sicherheitsdefizite bei Lesegeräten [Update], August 2010. <http://www.heise.de/newsticker/meldung/Elektronischer-Personalausweis-Sicherheitsdefizite-bei-Lesegeraeten-Update-1064338.html>.
- [Hei10c] Heise Online. Neuer Personalausweis: Ausweis-App mit Lücken [2. Update], November 2010. <http://www.heise.de/newsticker/meldung/Neuer-Personalausweis-AusweisApp-mit-Luecken-2-Update-1133376.html>.

- [Hei10d] Heise Online - The H Security. CCC reveals security problems with German electronic IDs, September 2010. <http://www.h-online.com/security/news/item/CCC-reveals-security-problems-with-German-electronic-IDs-1094577.html>.
- [Hei11] Heise Online. Weitere Sicherheitslücke beim elektronischen Personalausweis, August 2011. <http://www.heise.de/security/meldung/Weitere-Sicherheitsluecke-beim-elektronischen-Personalausweis-1319432.html>.
- [Hei12] Heise Online. Zwei Jahre später: der neue Personalausweis, February 2012. <http://www.heise.de/newsticker/meldung/Zwei-Jahre-spaeter-der-neue-Personalausweis-1736387.html>.
- [Hei13] Heise Online. Geld abheben mit dem Personalausweis, March 2013. <http://www.heise.de/newsticker/meldung/Geld-abheben-mit-dem-Personalausweis-1815877.html>.
- [HKKK12] L. Hanzlik, K. Klucznik, P. Kubiak, and M. Kutyłowski. Restricted Identification without Group Keys. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1194–1199, 2012.
- [HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 89–98, San Jose, California, USA, June 6–8, 2011. ACM Press.
- [HP04] Helena Handschuh and Bart Preneel. Minding your MAC Algorithms. In *Information Security Bulletin 9(6)*, pages 213–221, 2004.
- [ICA06] ICAO. Machine Readable Travel Documents. Technical Report Doc 9303, Part 1 Machine Readable Passports, Sixth Edition, International Civil Aviation Organization (ICAO), 2006.
- [Ica09] Thomas Icart. How to Hash into Elliptic Curves. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 303–316, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

- [IKo3] Tetsu Iwata and Kaoru Kurosawa. Stronger Security Bounds for OMAC, TMAC, and XCBC. In Thomas Johansson and Subhamoy Maitra, editors, *Progress in Cryptology - INDOCRYPT 2003: 4th International Conference in Cryptology in India*, volume 2904 of *Lecture Notes in Computer Science*, pages 402–415, New Delhi, India, December 8–10, 2003. Springer, Berlin, Germany.
- [ISO99] ISO/IEC. Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher. Technical Report ISO/IEC 9797-1, International Organization for Standardization, Geneva, Switzerland, 1999.
- [ISO04a] ISO/IEC. Identification cards - Integrated circuit(s) cards with contacts - Part 6: Interindustry data elements for interchange. Technical Report ISO/IEC 7816-8, International Organization for Standardization, Geneva, Switzerland, 2004.
- [ISO04b] ISO/IEC. Identification cards - Integrated circuit(s) cards with contacts - Part 8: Security related interindustry commands. Technical Report ISO/IEC 7816-8, International Organization for Standardization, Geneva, Switzerland, 2004.
- [ISO05a] ISO/IEC. Identification cards - Integrated circuit(s) cards with contacts - Part 4: Organization, security and commands for interchange. Technical Report ISO/IEC 7816-4, International Organization for Standardization, Geneva, Switzerland, 2005.
- [ISO05b] ISO/IEC. Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers. Technical Report ISO/IEC 18033-3, International Organization for Standardization, Geneva, Switzerland, 2005.
- [ISO06] ISO/IEC. Information technology - Security techniques - Modes of operation for an n-bit block cipher. Technical Report ISO/IEC 10116, International Organization for Standardization, Geneva, Switzerland, 2006.
- [ISO10] ISO/IEC. Supplemental Access Control for Machine Readable Travel Documents. Technical Report ISO/IEC JTC1 SC17 WG3/TF5 Version 1.1, International Organization for Standardization, Geneva, Switzerland, 2010.

- [Jan10] Jan Schejbal. Personal Blog, November 2010. <http://janschejbal.wordpress.com/2010/11/09/ausweisapp-gehackt-malware-uber-autoupdate/>.
- [JJV02] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption – FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 237–251, Leuven, Belgium, February 4–6, 2002. Springer, Berlin, Germany.
- [JK03] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), February 2003.
- [JNo3] Antoine Joux and Kim Nguyen. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups. *Journal of Cryptology*, 16(4):239–247, September 2003.
- [Jon01] Jakob Jonsson. Security proofs for the RSA-PSS signature scheme and its variants. Cryptology ePrint Archive, Report 2001/053, 2001. <http://eprint.iacr.org/>.
- [JV96] Mike Just and Serge Vaudenay. Authenticated Multi-Party Key Agreement. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT’96*, volume 1163 of *Lecture Notes in Computer Science*, pages 36–49, Kyongju, Korea, November 3–7, 1996. Springer, Berlin, Germany.
- [KBe12] C.f. B. J. Koops, H. Buitelaar, and M. Lips eds. D5.4: Anonymity in electronic government: a case-study analysis of governments? identity knowledge. *FIDIS report*, Feb 2012.
- [KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Berlin, Germany.
- [KKKK12] Mirosław Kutylowski, Lukasz Krzywiecki, Przemysław Kubiak, and Michał Koza. Restricted Identification Scheme and Diffie-Hellman Linking Problem. In Liqun Chen, Moti Yung, and Liehuang Zhu, editors, *Trusted Systems*, volume 7222 of *Lecture*

- Notes in Computer Science*, pages 221–238. Springer Berlin Heidelberg, 2012.
- [KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.
- [KM07] Neal Koblitz and Alfred J. Menezes. Another Look at “Provable Security”. *Journal of Cryptology*, 20(1):3–37, January 2007.
- [Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO’96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Berlin, Germany.
- [KOP11] Timo Kasper, David Oswald, and Christof Paar. Side-channel analysis of cryptographic rfids with analog demodulation. In *RFIDSec*, pages 61–77, 2011.
- [Kra95] Kravitz, D. W. Digital signature algorithm. *Computer Engineering*, 44(5):6–17, 1995.
- [KS11a] Wolfgang Killmann and Werner Schindler. A proposal for: Functionality classes for random number generators. Technical Report (BSI AIS 20/31,) Version 2.0, Bundesamt fuer Sicherheit in der Informationstechnik, 2011.
- [KS11b] Mirosław Kutylowski and Jun Shao. Signing with Multiple ID’s and a Single Key. *IEEE Annual Symposium on Foundations of Computer Science*, pages 519–520, 2011.
- [KSD<sup>+</sup>11] Po-Chun Kuo, Michael Schneider, Özgür Dagdelen, Jan Reichelt, Johannes Buchmann, Chen-Mou Cheng, and Bo-Yin Yang. Extreme Enumeration on GPU and in Clouds - - How Many Dollars You Need to Break SVP Challenges -. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 176–191, Nara, Japan, September 28 – October 1, 2011. Springer, Berlin, Germany.
- [KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-Optimal Password-Based Authenticated Key Exchange. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597

- of *Lecture Notes in Computer Science*, pages 293–310, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.
- [LGo7] Dimitrios Lekkas and Dimitris Gritzalis. E-Passports as a Means Towards the First World-Wide Public Key Infrastructure. In *EuroPKI*, pages 34–48, 2007.
- [LIS12] Ji Li, Takanori Isobe, and Kyoji Shibutani. Converting Meet-In-The-Middle Preimage Attack into Pseudo Collision Attack: Application to SHA-2. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 264–286, Washington, DC, USA, March 19–21, 2012. Springer, Berlin, Germany.
- [LLMo7] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger Security of Authenticated Key Exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16, Wollongong, Australia, November 1–2, 2007. Springer, Berlin, Germany.
- [Mano8] Stephane Manuel. Classification and generation of disturbance vectors for collision attacks against SHA-1. *Cryptology ePrint Archive*, Report 2008/469, 2008. <http://eprint.iacr.org/>.
- [Mau93] Ueli M. Maurer. Protocols for Secret Key Agreement by Public Discussion Based on Common Information. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO’92*, volume 740 of *Lecture Notes in Computer Science*, pages 461–470, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Berlin, Germany.
- [MDCE11] Mohammed Meziani, Özgür Dagdelen, Pierre-Louis Cayrel, and Sidi Mohamed El Yousfi Alaoui. S-FSB: An Improved Variant of the FSB Hash Family. In Tai-hoon Kim, Hojjat Adeli, Rosslin John Robles, and Maricel Balitanas, editors, *Information Security and Assurance*, volume 200 of *Communications in Computer and Information Science*, pages 132–145. Springer Berlin Heidelberg, 2011.
- [Mito3] Mitchell, C.J. Key recovery attack on ANSI retail MAC. *Electronics Letters*, 39(4): 361 – 362, feb 2003.

- [Mit05] Chris J. Mitchell. Error Oracle Attacks on CBC Mode: Is There a Future for CBC Mode Encryption? In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC 2005: 8th International Conference on Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 244–258, Singapore, September 20–23, 2005. Springer, Berlin, Germany.
- [MR11] Ueli Maurer and Renato Renner. Abstract Cryptography. In Bernard Chazelle, editor, *The Second Symposium in Innovations in Computer Science, ICS 2011*, pages 1–21. Tsinghua University Press, January 2011.
- [MSWo8] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. A Modular Security Analysis of the TLS Handshake Protocol. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 55–73, Melbourne, Australia, December 7–11, 2008. Springer, Berlin, Germany.
- [MVV07] Jean Monnerat, Serge Vaudenay, and Martin Vuagnoux. About Machine-Readable Travel Documents – Privacy Enhancement Using (Weakly) Non-Transferable Data Authentication. *RFIDSEC '07*, 2007.
- [NIS] NIST. FIPS PUB 180-3 Secure Hash Standard (SHS).
- [NIS07] NIST. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Technical Report SP 800-38B, National Institute of Standards and Technology, 2007.
- [Nit09] Rishab Nithyanand. A survey on the evolution of cryptographic protocols in epassports. *Cryptology ePrint Archive*, Report 2009/200, 2009. <http://eprint.iacr.org/>.
- [NR97] Moni Naor and Omer Reingold. Number-theoretic Constructions of Efficient Pseudo-random Functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.
- [Oka93] Tatsuaki Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Berlin, Germany.

- [OP01] Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118, Cheju Island, South Korea, February 13–15, 2001. Springer, Berlin, Germany.
- [Pas03] Rafael Pass. On Deniability in the Common Reference String and Random Oracle Model. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Berlin, Germany.
- [Ped92] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO’91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Berlin, Germany.
- [PPWo8] Vijayakrishnan Pasupathinathan, Josef Pieprzyk, and Huaxiong Wang. An On-Line Secure E-Passport Protocol. In Liqun Chen, Yi Mu, and Willy Susilo, editors, *Information Security Practice and Experience*, volume 4991 of *Lecture Notes in Computer Science*, pages 14–28. Springer Berlin Heidelberg, 2008.
- [PRoo] Erez Petrank and Charles Rackoff. CBC MAC for Real-Time Data Sources. *Journal of Cryptology*, 13(3):315–338, 2000.
- [PSoo] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
- [PVo4] Giuseppe Persiano and Ivan Visconti. An Efficient and Usable Multi-show Non-transferable Anonymous Credential System. In Ari Juels, editor, *FC 2004: 8th International Conference on Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 196–211, Key West, USA, February 9–12, 2004. Springer, Berlin, Germany.
- [PvO99] Bart Preneel and Paul C. van Oorschot. On the Security of Iterated Message Authentication Codes. *IEEE Transactions on Information Theory*, 45(1):188–199, 1999.

- [PY04] Kenneth G. Paterson and Arnold Yau. Padding Oracle Attacks on the ISO CBC Mode Encryption Standard. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 305–323, San Francisco, CA, USA, February 23–27, 2004. Springer, Berlin, Germany.
- [QSo1] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
- [Rog11] Phillip Rogaway. Evaluation of Some Blockcipher Modes of Operation. Technical report, Cryptography Research and Evaluation Committees (CRYPTREC), 2011.
- [RSA02] RSA Laboratories. PKCS #1 v2.1. In *RSA Cryptography Standard*. RSA Security, June 2002.
- [RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565, Gold Coast, Australia, December 9–13, 2001. Springer, Berlin, Germany.
- [RSVC09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111, Lausanne, Switzerland, September 6–9, 2009. Springer, Berlin, Germany.
- [Sat05] Akashi Satoh. Hardware Architecture and Cost Estimates for Breaking SHA-1. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC 2005: 8th International Conference on Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 259–273, Singapore, September 20–23, 2005. Springer, Berlin, Germany.
- [SBPV08] Ahmad-Reza Sadeghi, Carlo Blundo, Giuseppe Persiano, and Ivan Visconti. Resettable and non-transferable chip authentication for e-passports. In *RFIDSec 2008*, July 2008.

- [Sch90] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO’89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.
- [Sch91] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
- [Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.
- [Sho97] Victor Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT’97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Germany.
- [Sho99] Victor Shoup. On Formal Models for Secure Key Exchange. Technical Report RZ 3120, IBM, 1999.
- [SK02] Werner Schindler and Wolfgang Killmann. Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 431–449, Redwood Shores, California, USA, August 13–15, 2002. Springer, Berlin, Germany.
- [SvdWo6] Andrew Shallue and Christiaane van de Woestijne. Construction of Rational Points on Elliptic Curves over Finite Fields. In Hess, Florian and Pauli, Sebastian and Pohst, Michael, editor, *Algorithmic Number Theory*, volume 4076 of *Lecture Notes in Computer Science*, pages 510–524. Springer Berlin Heidelberg, 2006.
- [Vau03] Serge Vaudenay. The Security of DSA and ECDSA. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 309–323, Miami, USA, January 6–8, 2003. Springer, Berlin, Germany.
- [Wei05] Victor K. Wei. Tracing-by-Linking Group Signatures. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors,

- 
- ISC 2005: 8th International Conference on Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 149–163, Singapore, September 20–23, 2005. Springer, Berlin, Germany.
- [Wil11] Stephen C. Williams. Analysis of the SSH Key Exchange Protocol. In Liqun Chen, editor, *13th IMA International Conference on Cryptography and Coding*, volume 7089 of *Lecture Notes in Computer Science*, pages 356–374, Oxford, UK, December 12–15, 2011. Springer, Berlin, Germany.
- [WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.
- [YWDW06] Guomin Yang, Duncan S. Wong, Xiaotie Deng, and Huaxiong Wang. Anonymous Signature Schemes. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 347–363, New York, NY, USA, April 24–26, 2006. Springer, Berlin, Germany.