Encryption Schemes and Key Exchange Protocols in the Certificateless Setting

by

Georg Lippold

Dipl.-Inf. (Diploma in Computer Science) (University of Bremen) – 2006

Thesis submitted in accordance with the regulations for the Degree of Doctor of Philosophy

Information Security Institute
Faculty of Science and Technology
Queensland University of Technology

November 6, 2010
Keywords

certificateless encryption, certificateless key encapsulation, certificateless key agreement, security models, IND-CPA, IND-CCA2, generic conversion, ID-based, standard model, random oracle model, bilinear pairings, lattices, trapdoor algorithms
Abstract

The contributions of this thesis fall into three areas of certificateless cryptography. The first area is encryption, where we propose new constructions for both identity-based and certificateless cryptography. We construct an \( n \)-out-of-\( n \) group encryption scheme for identity-based cryptography that does not require any special means to generate the keys of the trusted authorities that are participating. We also introduce a new security definition for chosen ciphertext secure multi-key encryption. We prove that our construction is secure as long as at least one authority is uncompromised, and show that the existing constructions for chosen ciphertext security from identity-based encryption also hold in the group encryption case. We then consider certificateless encryption as the special case of 2-out-of-2 group encryption and give constructions for highly efficient certificateless schemes in the standard model. Among these is the first construction of a lattice-based certificateless encryption scheme.

Our next contribution is a highly efficient certificateless key encapsulation mechanism (KEM), that we prove secure in the standard model. We introduce a new way of proving the security of certificateless schemes based that are based on identity-based schemes. We leave the identity-based part of the proof intact, and just extend it to cover the part that is introduced by the certificateless scheme. We show that our construction is more efficient than any instanciation of generic constructions for certificateless key encapsulation in the standard model.

The third area where the thesis contributes to the advancement of certificateless cryptography is key agreement. Swanson showed that many certificateless key agreement schemes are insecure if considered in a reasonable security model. We propose the first provably secure certificateless key agreement schemes in the strongest model for certificateless key agreement. We extend Swanson's definition for certificateless key agreement and give more power to the adversary. Our new schemes are secure as long as each party has at least one uncomprom-
mised secret. Our first construction is in the random oracle model and gives the adversary slightly more capabilities than our second construction in the standard model. Interestingly, our standard model construction is as efficient as the random oracle model construction.
2.2.1 Negligible Function ........................................ 19
2.2.2 Computational Diffie-Hellman Problem .................. 19
2.2.3 Decisional Diffie-Hellman Problem ....................... 19
2.2.4 Computational Bilinear Diffie-Hellman Problem ........ 20
2.2.5 Decisional Bilinear Diffie-Hellman Problem ............ 21
2.2.6 Learning With Error Problem ............................ 21
2.2.7 The Twin Bilinear Diffie-Hellman Trapdoor Theorems . 22

2.3 Definitions of Cryptographic Primitives .................... 25
2.3.1 Strong Randomness Extractor ............................ 25
2.3.2 Pseudorandom Function Family ........................... 26
2.3.3 Target Collision Resistant Hash Function ............... 26
2.3.4 Waters’ Hash ............................................ 27

2.4 A Short Introduction to Provable Security ................ 28

2.5 Signatures, Message Authentication Codes, and Encapsulation Schemes ............................................. 30

2.6 Encryption .................................................. 34
2.6.1 Public Key Encryption ..................................... 35
2.6.2 Identity-Based Encryption (IBE) .......................... 38
2.6.3 Hierarchical Identity-Based Encryption (HIBE) ......... 41

2.7 Key Encapsulation ............................................ 44
2.7.1 Key-Encapsulation Mechanisms ........................... 44
2.7.2 ID-based Key Encapsulation Mechanism ................ 45

2.8 Key Agreement ............................................... 47
2.8.1 The Canetti-Krawczyk Model for Key Agreement (CK Model) .................................................. 50
2.8.2 The Extended Canetti-Krawczyk Model for Key Agreement (eCK Model) .................................. 52
Public Key eCK Model ........................................ 52
ID-based eCK Model .......................................... 54

2.9 Conclusion .................................................. 55

3 Security Models for Certificateless Protocols ................. 57
3.1 Certificateless Cryptography ................................ 57
3.1.1 General Attacks Against Certificateless Cryptography . 58
3.2 Types of Certificateless Adversaries ....................... 59
6.1.3 Summary .............................................. 169
6.2 Certificateless Key Agreement in the Standard Model .......... 169
  6.2.1 An $e^2$CK Secure Certificateless AKE in the Standard Model 170
  6.2.2 Proving the Protocol Secure .............................. 172
  6.2.3 Proof of Security in the $e^2$CK Model ....................... 174
    Analysis of Game 1: ....................................... 179
    Analysis of Game 2: ....................................... 179
    Analysis of Game 3 ......................................... 180
    Analysis of Game 4: ....................................... 182
    Analysis of Game 5 ......................................... 184
    Analysis of Game 6 ......................................... 184
    Analysis of Game 7 ......................................... 185
    Combining Results ......................................... 186
  6.2.4 Summary ............................................... 187
6.3 Conclusion .................................................. 188

7 Conclusion and Future Work .................................. 191
  7.1 Contributions ............................................. 191
  7.2 Applications .............................................. 193
  7.3 Future Work ............................................... 194

Bibliography ................................................. 197
# List of Figures

1.1 Typical workflow in a public key infrastructure. .......................... 2
1.2 Typical workflow in ID-based cryptography. ............................. 4
1.3 Typical workflow in certificateless cryptography. ....................... 7

2.1 A diagram for the IND-CPA security game ............................. 36
2.2 A diagram for the IND-CCA2 security game ............................ 37
2.3 The Diffie-Hellman key exchange protocol ............................. 47

3.1 PK-AKE + ID-AKE ≠ CL-AKE ........................................ 81

4.1 The signature based construction for IND-CCA2 security from ID-based encryption by Boneh et al. [BCHK07]. ....................... 92
4.2 An IND-CCA2 secure multi-key encryption scheme based on the CHK construction. .................................................. 95
4.3 An ID-based IND-CCA2 secure multi-key encryption scheme . . 112
4.4 Efficient certificateless encryption in the standard model . . . . 117
4.5 Our lattice-based two-key KEM. ...................................... 124
4.6 Our lattice-based CL-HIBE ............................................. 128
4.7 The CLE version of the HIBE in [BBG05] ............................... 130

5.1 Our CCA secure CL-KEM. .............................................. 139
5.2 Overview of the proof of security for our CL-KEM. .................. 142

6.1 Our e²CK certificateless key agreement protocol in the random oracle model. .................................................. 153
6.2 Our e²CK secure certificateless key agreement protocol in the standard model. .................................................. 171
List of Tables

4.1 Overview of Chapter 4 .......................................... 86
4.2 Efficiency comparison of our generic construction with existing schemes .................................................. 133
5.1 Comparison of the Huang-Wong scheme with our scheme .... 141
6.1 Possible corrupt queries sorted by strategy ................... 158
6.2 Modified $H_1$ oracle .............................................. 159
6.3 Modified $H_3$ oracle suitable for twin bilinear Diffie-Hellman ... 160
6.4 Possible corrupt queries sorted by strategy ................... 175
6.5 Efficiency comparison of our CL-AKE schemes ............... 188
Glossary

TCR target collision resistant hash function. 26, 27

ppt probabilistic polynomial time. xix, 19–21, 30–37, 39, 40, 42, 43, 45, 50, 87, 88

BTE binary tree encryption. 124

CK model Canetti and Krawczyk model for key agreement [CK01]. 50, 60

CL certificateless. 69, 82, 121

CL-AKE certificateless authenticated key agreement protocol. 81

CL-KEM certificateless key-encapsulation mechanism. 69

CLE certificateless encryption scheme. 98

DEM data encapsulation mechanism. 44, 125

eCK model extended Canetti and Krawczyk model for key agreement [LLM07]. 50, 52, 54, 61, 73

HCLE hierarchical certificateless encryption scheme. 66, 119

HIBE hierarchical identity based encryption scheme. 42, 43, 84, 90, 109

IB-KEM identity-based key encapsulation mechanism. 45, 46

IBE identity-based encryption scheme. 39, 41, 84, 90, 92, 104, 109

ID-AKE identity-based authenticated key agreement protocol. 81

ID-based identity-based. 3, 4, 6, 8, 40, 54–56, 58, 73, 82, 89, 90
IND-CCA2 secure  secure against adaptive chosen ciphertext attacks based on indistinguishability. 37, 40, 84, 88, 90, 118, 120

IND-CCA2 security  indistinguishable based security against adaptive chosen ciphertext attacks. 35, 36, 40, 84, 88, 89

IND-CPA secure  secure against chosen plaintext attacks based on indistinguishability. 36, 40, 84, 90

IND-CPA security  indistinguishable based security against adaptive chosen plaintext attacks. 35, 40, 88

KEM  key encapsulation mechanism. 11, 26, 44, 45, 60, 69, 121, 125, 167, 172

KGC  key generation centre. 41, 55, 60, 73

MAC  message authentication code. 31, 90, 91, 115

MK-MSK-IND-CPA secure  multi-key masking IND-CPA secure. 106–108

MK-MSK-SID-IND-CPA secure  multi-key masking selective-identity IND-CPA secure. 109

MSK-IND-CPA secure  masking IND-CPA secure. 105, 106, 108

MSK-SID-IND-CPA secure  masking selective-identity IND-CPA secure. 109

PK-AKE  public-key-based authenticated key agreement protocol. 81

PKI  public key infrastructure. 2

PRF  pseudorandom function. 26

SID-IND-CCA2 secure  secure against selective-identity adaptive chosen ciphertext attacks. 41, 191

SID-IND-CCA2-CLE  selective identity IND-CCA2 secure certificateless encryption scheme. 65, 66

SID-IND-CCA2-HCLE  selective identity IND-CCA2 secure hierarchical certificateless encryption scheme. 67, 69
SID-IND-CPA secure secure against selective-identity adaptive chosen plaintext attacks. 40, 94, 109, 118, 191

SID-IND-CPA-CLE selective identity IND-CPA secure certificateless encryption scheme. 65, 66

SID-IND-CPA-HCLE selective identity IND-CPA secure hierarchical certificateless encryption scheme. 67, 68

TA trusted authority. 55
Declaration

The work contained in this thesis has not been previously submitted for a degree or diploma at any higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed: ............................................. Date: .........................
The following papers have been published or presented, and contain material based on the content of this thesis.


Acknowledgements

I would like to thank my supervisors Colin Boyd and Juanma González Nieto for their guidance, support, and friendship during the last 44 months. Without Colin offering me a PhD scholarship and living allowance after my graduation, this thesis would not have been possible. I would like to thank the members of the review panels of my thesis, especially Rodney Wolff for encouraging me to showcase my work, Douglas Stebila for increasing the overall quality of the thesis by a significant amount, and Willy Susilo and Liqun Chen for the final review and their invaluable comments on how to improve my thesis further.

I would also like to thank all of the members of the Information Security Institute at QUT for their friendship, support, and the nice atmosphere that everyone contributed to. I would particularly like to mention Ejaz Ahmed, Mufeed Al Mashrafi, Ali Alhamdan, Hani Alzaid, James Birkett, Mark Brannagan, Chai-Wen Chuah, Andrew Clark, Craig Costello, Ed Dawson, Vik Tor Goh, Choudary Gorantla, Marianne Hirschbichler, Husseyin Hisil, Roheena Khan, Lakshmi Kuppusamy, Andrew Marrington, Adrian McCullagh, George Mohay, Long Ngo, Quan Pham, Ken Radke, Jothi Rangasamy, Jason Reid, Jared Ring, Farzad Salim, Bradley Schatz, Leonie Simpson, Jason Smith, Douglas Stebila, Donald Sun, Suriadi Suriadi, Sui-Guan Teo, Suratose Tritilanunt, and Reeza Z’Aba. I would like to thank the support staff at the ISI, Elizabeth, Christine, Gleb, Edward, Matt, and James: you all made my life a lot easier. I would like to thank Denise Goya for interest in my work, correcting some flaws, and implementing it in the Brazilian mobile health system Borboleta. I would like to thank David Jao and Jonathan Katz for feedback on my work and helpful discussions. I would like to thank Michael Hortmann, Dmitry Feichtner-Kozlov, Helge Holden, and Jouko Vaananen for the opportunity to work for the European Mathematical Society. I would like to thank Ed Dawson, Christian Cachin, and Shai Halevi for supporting my efforts to publish the videos from Crypto 2009, and David Pointcheval
and Christian Cachin for support in publishing the videos of PKC 2010. I would like to thank Colin Boyd, Guy Gable, Leonie Simpson, Colin Fidge, Malcolm Corney, Lauren May, and Juanma González Nieto for giving me the opportunity to teach in various subjects.

Privately, I would like to thank my wife for her unconditional love, encouragement and support during our time in Australia. I would like to thank our daughter for her inspiration, for exploring the world with us, and for her smiles and her laughs. I would like to thank my mother and my stepfather and my mother-in-law and my sister-in-law for their unconditional love, their visits, and their support. I would like to thank my sister, her boyfriend, and their child for their love. I would like to thank all our German friends and relatives who stopped over and reminded us of home. I would like to thank all the other guys from home for staying in touch. I would like to thank my godson and his family for keeping in touch. In particular, I would like to thank all our more intimate friends here in Brisbane for their friendship and the good times we had together.
Notation

We briefly introduce the mathematical notation and the graphical elements used in drawings that we will use throughout the thesis.

Mathematical Notation

For a set $A$ we write $a \in A$ to specify an element from the set. If we draw an element randomly from the set, we write $a \triangleleft A$. We express the probability that an event $x$ happens by $\Pr[x]$. A vector is denoted by $\mathbf{a}$, a matrix by $\mathbf{A}$. Sometimes we transpose a vector, i.e. transform a vector from a column to a row. For $\mathbf{a}$ or a matrix $\mathbf{A}$ we let $\mathbf{a}^t$ denote the transposition of $\mathbf{a}$, and $\mathbf{A}^t$ denote the transposition of $\mathbf{A}$. For an $n \times m$ matrix $\mathbf{A}$ we write $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ where $\mathbf{a}_i$ denotes the $i$th column vector of $\mathbf{A}$. An algorithm/function that takes inputs $a$ and $b$ and outputs a value $c$ is written as $c \leftarrow \text{Algorithm name}(a, b)$. In some algorithms it is necessary to compare two values. To compare $a$ with $b$ we write $a \equiv b$. Some algorithms run in probabilistic polynomial time, which is commonly abbreviated as probabilistic polynomial time ($\text{ppt}$). Some algorithms have a security parameter $k$ that is used in the setup algorithm to generate the other parameters used by subsequent algorithms accordingly. In the security experiments associated with cryptographic schemes, the success probability of an adversary is related to this security parameter. Sometimes we also write $1^k$ to represent a string of 1’s of length $k$, which is given as a security parameter. Generally speaking, the larger the security parameter $k$ is, the longer it takes for an algorithm to run. However, since the algorithms that we describe are $\text{ppt}$ algorithms, the runtime increases only polynomially. Simultaneously, the success probability of an adversary against the algorithm decreases exponentially. Therefore using larger security parameters makes sense. To concatenate two strings, we write $\mathbf{a} \circ \mathbf{b}$. Similarly, to append an element to a vector, we also
write $a \circ b$. Sometimes we use multiple elements from the same set. We write $a, b, c \in \mathbb{Z}^3$ meaning $a, b, c \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$. In cyclic groups, we use the multiplicative notation. For a positive integer $k$, $[k]$ denotes the set $\{1, \ldots, k\}$; $[0]$ is the empty set. We use $\lambda$ to represent the empty string.

## Diagrams

In workflow diagrams like in Figure 1.1, 1.2, and 1.3, we use arrows to indicate actions of a party. Colors are used to indicate which elements of the diagram belong to which party, all elements associated with one party use the same color. If an element is controlled by two parties, then we use a gradient to indicate which parties have control over the element. Parties are depicted by either male or female user icons. The trusted authority is always depicted by two intertwined hands. Keys (public and private) are symbolized by key symbols. Dashed lines indicate that keys match each other (public key and corresponding private key). Certificates are symbolized by a certificate icon, and identities by an identity card. Lists are depicted as a document containing two columns of text.

Diagrams for security experiments, like Figure 2.1, 2.2, and 5.2, use arrows (sometimes dashed) to indicate communication between parties, letter icons to indicate messages, and letter icons decorated with keys to indicate encrypted messages. Private information of a party and decisions made by a party are inside the respective party’s box.

To depict encryption and signature operations in our diagrams (for example in Figure 4.1, 4.2, 4.3, and 4.4) we use arrows starting at at the key used for the operation and labeled with the operation name. An arrow collects all the necessary information needed by the operation by joining with other arrows, and ends at the output of the operation. Colors are used as follows. Messages and elements derived from a message are green, schemes under control of the party which encrypts the message are blue, and information needed for the encryption provided by third parties (like public keys) is in red (associated with a trusted authority) and yellow (associated with the intended recipient). The final ciphertext is highlighted by orange circles around the elements of the ciphertext. The remaining symbols are used similar to the diagrams for security experiments.

Key agreement schemes (for example Figure 2.3, 6.1, 6.2) use arrows to indicate the exchanged messages. Information belonging to each party is listed
below the respective party.
Chapter 1

Introduction

Public key cryptography is a relatively new technology. Public interest in the subject was sparked by the work of Diffie and Hellman [DH76] who introduced the first public key key agreement scheme and Rivest, Shamir and Adleman [RSA78] who constructed the first public key encryption scheme. The name public key cryptography stems from the fact that a cryptographic key that is generated in such a scheme always has two parts: one part that can be publicly distributed and that everyone can use (the public key), and one part of the key has to be kept secret by the person who generated it (the private key). The public key is used to send messages securely to the person that generated the key pair by other parties or to verify digital signatures generated by the person who generated the key pair. The private key is used to decrypt messages or to generate digital signatures.

Since the invention of public key cryptography, cryptography has become an everyday commodity in the lives of many people having access to advanced technology, such as computers and cellphones. They use cryptography without even knowing why it is there, what it is good for, or what the principles behind cryptography are. However, in most scenarios, cryptography is used to guarantee confidentiality, authenticity and/or integrity of data transmitted from one identity to another identity (and vice versa). With the help of trusted third parties, cryptography has enabled users that never had contact before to interact securely and confidentially. Currently, there are three known ways that a trusted third party can mediate trust between two identities: public key infrastructures,
identity-based cryptography and certificateless cryptography.

1.1 Public Key Infrastructure using Certificates

Public key infrastructure is historically the oldest version of trust mediation. The workflow associated with a public key infrastructure (PKI) is shown in Figure 1.1. The trusted third party (also named trusted authority (TA)) in a public key infrastructure certifies copies of the public key of a given identity. To be able to do this, all identities must have and trust the public key of the trusted third party. The trusted third party thus vouches for the connection between a public key and a given identity, usually by adding personal identification details to the public key and digitally signing the resulting document using the private key of the trusted third party. In the context of PKI, the document is then named a certificate, its authenticity can be checked using the public key of the trusted third party (which is implicitly trusted by all users of the PKI). A formal definition of a certificate is given in the X.509 Standard [CSF+08].

This approach has certain advantages and certain drawbacks. We see it as an advantage that the user is able to generate the key pair on its own and can be sure that no other party will know the corresponding private key, not even the trusted third party. However, certificate management is somewhat
tedious: since it is possible that the private key of the user or the authority becomes exposed to third parties, the validity of a given certificate has to be checked with the trusted authority (TA) each time the public key is used for a cryptographic operation. There are many approaches to how this can be done, the most prominent are Certificate Revocation Lists (CRL’s) [CSF+08] and the Online Certificate Status Protocol (OCSP) [MAM+99]. Using a certificate for encryption thus requires a lot of effort: the signature of the trusted authority has to be verified, and the validity of the certificate has to be checked online (and possibly the validity of the validation list has to be checked as well). Certificate Revocation Lists can become very large and take time to download and parse correctly. Furthermore, it is necessary to have synchronous clocks to ensure that a certificate is not used beyond its lifespan. We omitted clock sources from Figure 1.1, 1.2, and 1.3, although they would have to be present in all three diagrams for completeness. We omitted the clocks because it makes the diagrams easier to follow and helps distinguishing the major differences. Gentry [Gen03] compiled a compelling list of drawbacks commonly associated with public key infrastructures using certificates.

### 1.2 Identity-Based Cryptography

In 1984, Shamir [Sha85] proposed an alternative to the tedious workflow of a certificate-based public key infrastructure, named identity-based (ID-based) cryptography. A diagram of a typical ID-based workflow is given in Figure 1.2. A central trusted authority (TA) publishes parameters and a public key that is used to encrypt data to all users that trust the authority. Each user is issued a private key that matches the user’s identity and allows the user to decrypt and/or sign messages. Due to its ability to provide users with private keys, the trusted authority is also named key generation centre (KGC). The user’s identity, e.g. a social security number or an email address, is directly used for encryption or signature verification. In 1984, Shamir was only able to provide a signature scheme for identity-based cryptography and it took until 2000 for Sakai, Oghishi and Kasahara [SOK00] to propose the first identity-based key agreement schemes based on the mathematical primitive of bilinear pairings. In 2001, Boneh and Franklin [BF01] and Cocks [Coc01] published the first identity-based encryption (IBE) schemes. Cocks’ scheme was based on quadratic residues and had the
drawback that ciphertexts were huge. Boneh and Franklin’s IBE was based on bilinear pairings with a proof of security and was efficient.

Since then, many ID-based schemes have been published, lately focusing on ID-based constructions with stronger security models and on constructions that use new mathematical primitives. Many of the primitives of public key cryptography have been realized as ID-based primitives. The first identity-based encryption (IBE) schemes were the work of Boneh and Franklin [BF03] based on El-Gamal like encryption using pairings, Cocks [Coc01] based on quadratic residues, and Sakai and Kasahara [SK03] also based on bilinear pairings, but slightly more efficient since encryption did not need any pairings. However, a proof of security was missing and when the proof was written by Chen and Cheng [CC05], it used a stronger assumption than the Boneh and Franklin construction. This work was in the random oracle model. We discuss models for provable security in more detail in Section 2.4.

Boneh and Boyen then focused on constructions in the standard model. Their first construction was selective-identity secure [BB04a] (so an adversary had to declare in advance which identity would be attacked), and was quickly followed by fully secure construction [BB04b]. However, the construction was not very efficient. Waters [Wat05] showed how to construct fully secure and efficient ID-based encryption schemes. Boneh, Goh and Hamburg [BGH07] constructed an efficient IBE based on quadratic residues in 2007.
In 2002, Gentry and Silverberg [GS02] constructed the first hierarchical identity-based encryption (HIBE) scheme. The benefit of hierarchical constructions is that the root authority does not have to generate all the private keys used in the scheme. Private key generation in identity-based encryption is expensive. After Canetti, Halevi, and Katz [CHK04] showed that hierarchical identity-based encryption schemes can be used together with strong one-time signature schemes to construct chosen-ciphertext secure identity-based encryption in the standard model, interest sparked in hierarchical ID-based schemes in the standard model. Boneh, Boyen, and Goh [BBG05] constructed selective-identity secure hierarchical ID-based encryption with constant size ciphertexts in the standard model. Boyen and Waters [BW06] improved the construction to allow anonymity. At Crypto 2009, Waters [Wat09] was able to give a construction for fully secure HIBE and IBE schemes under simple assumptions.

Chen et al. [CCMLS05] and Boyen, Mei, and Waters [BMW05] constructed the first identity-based key encapsulation mechanisms in 2005. The Chen et al. construction is related to the Sakai-Kasahara scheme [SK03] and proven in the random oracle model. The Boyen, Mei, and Waters construction is related to the Boneh-Boyen encryption scheme [BB04b] and in the standard model. Key encapsulation mechanisms are usually more efficient than encryption schemes, but have a different security model. Kiltz and Galindo [KG06a,GK06] improved the efficiency of previous constructions. Bentahar et al. [BFMLS08] showed how to obtain generic constructions for identity-based key encapsulation mechanisms in the random oracle model.

Key agreement protocols in the ID-based setting were first studied by Smart [Sma02]. Chen and Kudla [CK03] improved Smart’s protocol and were able to prove it secure in the Bellare-Rogaway model for authenticated key agreement [BR93a]. McCullagh and Barreto [MB05] proposed a key agreement protocol that used the Sakai-Kasahara method for generating private keys for identities and is more efficient than the Chen and Kudla construction. However, a proof of security is missing. Chen, Cheng and Smart [CCS07] were able to prove the security of ID-based key agreement without having to rely on gap assumptions (see also Okamoto and Pointcheval [OP01] and Kudla and Paterson [KP05] for gap assumptions). Huang and Cao [HC09] were able to prove the first identity-based protocol that was provably secure in the eCK model (see also Section 2.8.2 for security models for key agreement protocols).
Identity-based signature schemes were proposed initially by Shamir [Sha85]. Shamir’s construction was based on groups of composite order. Guillou and Pointcheval [GQ88] use a similar construction, but used smartcards as identities. Cha and Cheon [CC03] provided the first ID-based signature scheme based on the gap Diffie-Hellman assumption, which formed a natural complement of the Boneh-Franklin IBE. Hess [Hes02] gave a construction that was based directly on the Diffie-Hellman assumption, which helped improve the tightness of the proof. Barreto et al. [BLMQ05] gave more efficient constructions for signatures that use private keys similar to the Sakai-Kasahara encryption schemes. Waters [Wat05] was able to construct the first identity-based signature scheme in the standard model. Paterson and Schuldt [PS06] were able to give a more efficient construction for ID-based signatures in the standard model.

Although ID-based schemes are appealing in terms of efficiency and manageability, user privacy is one of the major concerns in ID-based cryptography. Since the key generation centre can compute the private key for every user, it can use this escrow ability to monitor a user’s private communication and fake signatures.

1.3 Certificateless Cryptography

To enhance a users privacy, Al-Riyami and Paterson [ARP03] proposed the idea of certificateless cryptography. The goal of certificateless cryptography is to have the easy manageability that comes with ID-based cryptography while simultaneously having the privacy of public-key-based schemes. A typical workflow in certificateless cryptography is shown in Figure 1.3. Certificate-based cryptography proposed by Gentry [Gen03] has a similar approach to user privacy; Al-Riyami and Paterson [ARP05] and Libert and Quisquater [LQ06] showed that certificate-based schemes can be readily constructed from certificateless schemes. The enhanced privacy of certificateless schemes stems from the fact that every user generates their own public key pair and publishes it (e.g. in an online directory). Encryption then uses the ID-based public key and the public key generated by the user simultaneously. Then, both the ID-based private key and the user generated private key are used for decryption, thus key escrow is not an issue any more. However, due to the fact that a certificateless scheme uses multiple keys, that also enjoy different trust levels, security is more difficult to model. We will
1.3. Certificateless Cryptography

The first certificateless encryption scheme was proposed by Al-Riyami and Paterson [ARP03]. Al-Riyami and Paterson [ARP05] improved the efficiency of their initial construction, however they introduced security flaws that were addressed by Zhang and Feng [ZF05]. Baek, Safavi-Naini and Susilo [BSNS05] were able to construct a certificateless encryption scheme without pairings. However, they had to relax the original security model that was defined by Al-Riyami and Paterson. Furthermore, it was not possible for the user to generate a public key without having previously obtained a private key from the trusted authority. This prevents some of the workflows that are associated with identity-based encryption and previous constructions of certificateless encryption, for example the ability to encrypt messages into the future. These workflows are prohibited because all identities used for encryption in identity-based settings include a timestamp to mitigate the effect of compromised secret keys. Therefore, private keys have to change frequently, and accordingly certificateless keys have to be updated frequently as well. Sun, Zhang and Baek [SZB07] were later able to get back to the original security model proposed by Al-Riyami and Paterson [ARP03] without using pairings, however the workflow limitations were kept. Liu, Au and Susilo [LAS07] introduced the somewhat artificial notion of a “Denial of Decryption” attack that we discuss in more detail in Section 3.1.1. To mitigate the attack, they introduced the idea of self generated certificates. However, these self-generated certificates require the user to obtain an ID-based private key.
before the certificateless public key can be generated. This implies that standard workflows associated with ID-based and certificateless cryptography are not available, as discussed above. Lai and Kou [LK07] were able to construct self-generated certificate schemes without using bilinear pairings. However, their proof had some shortcomings that were fixed by Wang, Huang, and Yang [WHY08].

Au et al. [AMC07] showed that many certificateless schemes are vulnerable to attacks originating from a malicious trusted authority, especially in the strong security model originally proposed by Al-Riyami and Paterson. Following the attack description by Au et al. [AMC07], Hwang, Liu, and Chow [HLC08] constructed the first scheme that resisted this attack in the standard model. Certificateless encryption schemes for threshold encryption were discussed by [YCD08] in the standard model.

Other certificateless encryption schemes are the construction by Shi, Li, and Shi [SLS08] who used an approach similar to the Sakai-Kasahara construction to construct more efficient certificateless encryption schemes. Chow, Roth, and Rieffel [CRR08] focused on the connection between certificateless encryption and time-released encryption and proposed the first schemes for both paradigms in the standard model. In a recent development, Lai et al. [LDLK09] showed how to construct certificateless encryption based on RSA.

Generic constructions for certificateless encryption were first observed by Yum and Lee [YL04a] in the random oracle model. Libert and Quisquater [LQ06] analysed some flaws in Yum and Lee’s work and proposed new generic constructions for certificateless construction in the random oracle model. Somewhat generic constructions were also proposed by Cheng et al. [CCLC07], who showed how to construct generic certificateless encryption from identity-based encryption and public-key-based key agreement schemes in the random oracle model. Generic constructions for certificateless encryption in the standard model were proposed in a realistic security model by Huang and Wong [HW07a], and by Dent [Den08, Section 3.5]. Dent, Libert and Paterson [DLP08] presented inefficient generic constructions for certificateless encryption in the standard model, although in a stronger security model. They also published a concrete scheme in the standard model that was very efficient but not related to their generic construction. A very detailed overview of certificateless encryption schemes can be found in Dent [Den08].

Many papers appeared on certificateless signature schemes [YL04b, HSMZ05,
We will not discuss these papers in more detail, because we are not going to propose new certificateless signature schemes in this work. In our opinion, certificateless signatures offer no new security guarantees compared to ID-based signatures. This stems from the fact that signatures are added to plaintext messages, and therefore privacy is not a concern. Since adversaries in certificateless cryptography are allowed to replace the public key of a user, security of the signature must rely on the identity-based part of the certificateless key. Therefore, it seems as if certificateless signatures do not offer any improvement over identity-based signatures.

Many certificateless key agreement schemes were published in the literature \[Zh05, WCW06, MT06, XWSX08, HKK07, WZ07, LWZ08, WCB08, WZ08, CZQ+09\]. However, none of the proposed schemes had a proof of security in a security model that actually reflected the abilities of a certificateless adversary as Swanson and Jao \[SJ09\] observed. In Chapter 6 we present the first provably secure certificateless key agreement schemes in the literature today. Geng and Zhang \[GZ09b\] claim to have a provably secure key agreement protocol without pairings, however a proof of security has not been published. Their protocol has the same workflow limitations as other certificateless schemes that do not use pairings.

Prior to this work, certificateless key encapsulation mechanisms were studied only by Huang and Wong \[HW07b\] who provide an inefficient generic construction for certificateless key encapsulation in the standard model, and by Bentahar et al. \[BFMLS08\] who gave a more efficient construction, although in the random oracle model. We study efficient constructions for certificateless key encapsulation in the standard model in Chapter 5.

### 1.4 Provable security

This thesis focuses on advancing the research in certificateless cryptography. Since the work of Goldwasser and Micali \[GM84\], research in public key crypto-
graphy tried to prove that breaking a cryptographic protocol implies solving a problem in polynomial time that is deemed to be unsolvable in polynomial time. Bellare and Rogaway [BR93b] made proving cryptographic protocols easier by introducing the idea of a random oracle that is available to all parties. This approach is well known as the random oracle model (ROM). Since there are no random oracles in the real world, random oracles in cryptographic protocols are instantiated through hash functions. There has been some debate [CGH04] about the value of proofs in the random oracle model, but most scientists agree that a protocol that is proven in the random oracle model is better than a protocol that has no proof of security at all. Protocols that are proven without the use of the random oracle model are referred to as proven in the standard model, and are generally regarded higher because their security can be proven without having to rely on imaginary tools. However, protocols that are proven in the standard model are generally based on stronger complexity assumptions and are usually not as efficient as protocols in the random oracle model. It seems as if nowadays protocols proven in the random oracle model are more prevalent when it comes to using them in real life, mostly due to the improved efficiency of random oracle model protocols.

To make protocol design easier to understand and protocols more readily available, protocols are often composed of existing cryptographic sub-protocols, e.g. an encryption scheme that is secure against modification of the ciphertext may be composed of an encryption scheme without security against modification and a signature of the sender of the message. Then, the proof of security of the composed protocol treats the building blocks as “black boxes” and shows that breaking the composed protocol breaks the security of at least one of the building blocks. Thus, if the building blocks are secure, then the composed protocol is secure, too. However, composed protocols are often less efficient than protocols that are specifically designed for a purpose. Since the sub-protocols may use the same cryptographic primitive, e.g. the same hash function, it may happen that security cannot be guaranteed any more. This is due to the fact that most proofs in the random oracle model treat each random oracle independently. If a random oracle is used multiple times due to the composition of protocols, then the security guarantees of the individual protocols may not hold any more, as in some proofs the adversary needs to be rewound which is not practical any more in a composition where the same oracle is used twice.
1.5 Our contributions

Our research is in certificateless cryptography, mostly in the standard model. After a literature review that focused on certificateless protocols, we found research questions in the areas of certificateless encryption, certificateless key encapsulation and certificateless key agreement.

The generic constructions for certificateless encryption that we surveyed were both in the random oracle model [YL04a, LQ06] and in the standard model [DLP08, HW07a, Den08]. However, the constructions in the standard model are inefficient when used with lattices instead of pairings, due to the way that messages are typically encoded in lattice-based settings. Furthermore, there are very efficient constructions for chosen ciphertext security from identity-based encryption schemes (e.g. Boneh et al. [BCHK07]). However, the existing literature does not attempt to make these techniques available for certificateless encryption schemes. The results of the literature review led to the following research questions.

- Is it possible to construct efficient lattice-based certificateless encryption schemes in the standard model?

- Can the constructions for chosen ciphertext security from identity-based encryption also hold in the certificateless case?

This research results in a new generic construction for certificateless encryption in the standard model, that is also efficient when instantiated with lattice-based encryption schemes. We discuss the results in more detail in Chapter 4.

There is only one construction of a certificateless key encapsulation mechanism (KEM) by Huang and Wong [HW07b] in the standard model today. This is a generic construction which is not very efficient, due to the fact that a public key encryption scheme is used in the construction, which imposes considerable overhead in a construction for a KEM whose goal is to be computationally more efficient than an encryption scheme. The results of the literature review led to the following research question.

- Is it possible to obtain a direct construction of a certificateless KEM with a proof of security in the standard model that is more efficient than existing generic constructions?
This research results in a new and efficient construction for certificateless key encapsulation in the standard model. We also propose a new set of algorithms for certificateless KEMs, which lead to more efficient operation. We discuss the results in more detail in Chapter 3 and Chapter 5.

When we analyzed certificateless key agreement protocols in the literature, we found that none had a proof of security in a reasonable security model that considered the specific abilities of a certificateless adversary against the protocol correctly. A security model that considers the abilities of a certificateless adversary correctly was proposed by Swanson and Jao [SJ09, Swa08] in their analysis of flaws in certificateless key agreement protocols. However, they did not try to prove any protocol secure in their model. The results of the literature review led to the following research questions.

- Is the security model for certificateless key agreement that was proposed by Swanson and Jao [SJ09, Swa08] suitable to prove security of a certificateless key agreement protocol?
- Can we construct certificateless key agreement protocols with proofs of security?

This research results in two new construction for certificateless key agreement protocols, both in the random oracle model, and in the standard model. We also improve the model for certificateless key agreement given by Swanson and Jao. The results of our research are discussed in more detail in Chapter 3 and in Chapter 6.

1.6 Outline

The thesis and its contributions are structured as follows.

Chapter 2: This chapter explains the theoretical background that is used in the following chapters. First, the mathematical primitives used in the protocols that are discussed in the following chapters are explained. Then, we continue to define the complexity assumptions which are the foundations of the proofs in the protocols that we discuss later. An overview of the cryptographic primitives that are building blocks of our protocols follows, and we conclude with definitions of cryptographic protocols in the public-key-based and identity-based setting. One major contribution of this chapter
is the introduction of two new bilinear trapdoor theorems in analogy to the trapdoor theorems published by Cash, Kiltz, and Shoup [CKS08]. The definitions from this chapter form the basis of the security definitions for certificateless protocols that we develop in Chapter 3.

Chapter 3: We introduce the security models for the three types of certificateless protocols (encryption, key encapsulation, key agreement) that we focus our research on. The models for key encapsulation and key agreement are new contributions. This chapter also explains the different types of adversaries that are commonly assumed in certificateless protocols, and explains the general strategy that we will use when proving our protocols secure. This chapter includes the security models for certificateless encryption, certificateless key agreement and certificateless key encapsulation.

Chapter 4: We propose a new way of using identity-based encryption (IBE) schemes to obtain encryptions to one identity under multiple authorities. We show that the existing constructions to obtain chosen ciphertext security (IND-CCA2 security) from identity-based encryption as described by Boneh et al. [BCHK07] also hold in this setting. We then propose a very efficient way to do parallel encryption in a certain class of blinding/masking IND-CPA secure schemes. Almost all constructions for identity-based encryption fall into this class. Using the parallel encryption technique and the multi-authority encryption scheme in the special case of two authorities (one being the user, the other being the trusted authority), we construct highly efficient chosen ciphertext (IND-CCA2) secure certificateless encryption from chosen plaintext (IND-CPA) secure IBE schemes. We finish the chapter by giving two explicit constructions for certificateless schemes in the standard model, one based on pairings, the other based on lattices. We compare these schemes to the results obtained by previous constructions.

Chapter 5: This chapter focuses on certificateless key encapsulation mechanisms (CL-KEM). We give the first direct construction for a certificateless KEM that is secure in the standard model and compare it to existing generic constructions for certificateless KEMs.

The contents of this chapter have appeared in the following publication. Georg Lippold, Colin Boyd, and Juan Manuel González Nieto. Efficient certificateless kem in the standard model. In Donghoon Lee and Seokhie
Chapter 6: We consider two certificateless key agreement protocols in this chapter. The first protocol is in the random oracle model, and is the first protocol for certificateless key agreement in the literature that has a proof of security. The second protocol is a generic construction for certificateless key agreement in the standard model that uses certificateless KEM schemes.

The protocols discussed in this chapter have appeared in the following publications.


Chapter 7: We summarize the contributions of the thesis in this chapter. We explore avenues for further research, discuss an application of our work, and discuss some open problems.
Chapter 2

Theoretical Background

This chapter explains the theoretical background that is used in the following chapters. Firstly, the mathematical primitives used in the protocols that are discussed in the following chapters are explained. Then, we continue to define the complexity assumptions which are the foundations of the proofs in the protocols that we discuss later. An overview of the cryptographic primitives that are building blocks of our protocols follows, and we conclude with definitions of cryptographic protocols in the public-key-based and identity-based setting. These definitions are then used as the basis of the security definitions for certificateless protocols that are developed in Chapter 3.

In Section 2.6, 2.7, and 2.8, we review the standard definitions for cryptographic protocols, both in the public-key-based setting and in the identity-based setting. We also discuss the security experiments that are associated with the respective protocols. Comparing the definitions, it is evident that the security model for ID-based encryption differs from the security model for public key encryption in the following ways. First, there is a key derivation algorithm to derive ID-based keys from the master key. Then, the security experiment is limited to an identity for which the adversary did not ask for the secret key (the target identity ID*:). The adversary is allowed to learn the secret keys for all other identities, and is thus able to decrypt their ciphertexts or try to perform collusion attacks with their secret keys. Due to the enhanced possibilities of the adversary it seems as if the adversary for an ID-based scheme is more powerful than the adversary for a public key scheme, where an adversary is challenged
against a single public key that is completely random and not derived from a common master key.

As we will see in Chapter 3, certificateless protocols always have two secret keys for a user (the ID-based secret key and the user generated secret key), thus the security model for certificateless encryption differs from the security model for both ID-based encryption and standard public key encryption, because the adversary is even allowed to partially corrupt the target identity. That means that the adversary may learn either the ID-based secret key or the user generated secret key of the target identity $ID^*$. Therefore, the adversary is even more powerful than in the ID-based setting.

## 2.1 Definitions of Mathematical Primitives

In this section we describe mathematical primitives that the protocols in the following chapters use as their basis. Of particular importance are bilinear pairings and lattices.

### 2.1.1 Minimal Entropy

Given a set $A$, the minimal entropy measures the highest probability to draw a certain element from $A$. This definition is used in the key agreement protocol that we present in Section 6.2.

**Definition 2.1. Min-Entropy [GKR04]**

Let $\chi$ be a probability distribution over $A$. The min-entropy of $\chi$ is the value

$$\text{min-ent}(\chi) = \min_{x \in A : \Pr_x[x] \neq 0} (\log_2(\Pr_x[x]))$$

If $\chi$ has min-entropy $t$, then for all $x \in A : \Pr_x[x] \leq 2^{-t}$.

### 2.1.2 Bilinear Pairing

The primitive of a bilinear pairing fueled the research into identity cryptography since the ground-breaking paper by Boneh and Franklin [BF01]. We employ bilinear pairings in the key agreement protocol in Section 6.1, in the key encapsulation mechanism discussed in Chapter 5, and in one of the protocols surveyed in our construction of certificateless encryption in Chapter 4.
Definition 2.2. Bilinear Pairing [BF01]

Let $G$ and $G_T$ be groups of prime order $p$. A bilinear pairing is a map $e : G \times G \rightarrow G_T$ between the groups $G$ and $G_T$ that satisfies the following properties:

**Bilinear** We say that a map $e : G \times G \rightarrow G_T$ is bilinear if $e(g^a, g^b) = e(g, g)^{ab}$ for all $g \in G$ and $a, b \in \mathbb{Z}_p$.

**Non-degenerate** We say that $e$ is non-degenerate if it does not send all pairs in $G \times G$ to the identity in $G_T$. Since $G$ and $G_T$ are groups of prime order $p$, it follows that if $g \in G$ is a generator of $G$, then $e(g, g)$ is a generator of $G_T$.

**Computable** There is an efficient algorithm to compute $e(g, h)$ for any $g, h \in G$.

The only known way to compute pairings is on elliptic curves. Pairings were first described by Weil [Wei40]. Miller [Mil04] developed an efficient algorithm to compute pairings on elliptic curves. Galbraith, Paterson, and Smart [GPS08] give a short introduction to pairings on elliptic curves. Our key agreement protocol in Section 6.1 relies on the flexibility offered by “Type I” or symmetric pairings as Galbraith et al. classify them. The key encapsulation mechanism discussed in Chapter 5 could also be implemented using asymmetric pairings, however this would make the algorithm descriptions longer and more confusing. Furthermore, it does not give any new insights into the security of the constructions.

### 2.1.3 Lattices

The work by Ajtai [Ajt99] established lattices as a cryptographic primitive. Since then, Regev [Reg09] showed that solving the learning with error problem (discussed in Definition 2.9) implies a quantum solution to worst-case lattice problems. Current classical polynomial time algorithms for lattice based cryptography yield only mildly subexponential approximation factors. Since there are currently no known quantum algorithms for lattice problems that outperform classical algorithms, one can conjecture that solving lattice problems is also hard on quantum computers. This result fueled research in lattice cryptography, since classical cryptography may become “trivial” to solve given a quantum computer that has sufficiently many bits. We employ lattices in one of the encryption
schemes presented in Chapter 4, and thus present the first lattice based certificateless encryption scheme in the standard model. We now proceed to define lattices.

**Definition 2.3. Lattice [Pei09a]**

A lattice $\Lambda \subseteq \mathbb{Z}^m$ is defined as the set of all integer linear combinations of $m$ linearly independent basis vectors $B = \{b_1, \ldots, b_m\} \subset \mathbb{Z}^m$:

$$\Lambda = \mathcal{L}(B) = \left\{ Bc := \sum_{i \in [m]} c_i b_i : c \in \mathbb{Z}^m \right\}.$$

When $m \geq 2$, there are infinitely many bases that generate the same lattice.

We focus on a subset of the integer lattices modulo $q$. Let $n \geq 1$ and $q \geq 2$ be integers; the dimension $n$ is the main cryptographic security parameter, all other parameters are implicitly defined by $n$. An $m$-dimensional lattice from the family is specified relative to the additive group $\mathbb{Z}_q^n$ by a parity check matrix $A \in \mathbb{Z}_q^{n \times m}$. The associated lattice is defined as

$$\Lambda^\perp(A) = \left\{ x \in \mathbb{Z}^m \left| Ax = \sum_{j \in [m]} x_j a_j \equiv 0 \mod q \right. \right\} \subseteq \mathbb{Z}^m.$$

An introduction to lattice-based cryptography can be found in Micciancio and Regev [MR09].

### 2.2 Definitions of Computational Problems

Cryptographic protocols are based on problems that are believed to be hard to solve in polynomial time. In the proofs of the protocols presented in this thesis (Chapter 4, Chapter 5, Chapter 6), we will show how a successful adversary against the protocol can be used to solve one of these hard problems in polynomial time. From this fact we reason that there is no polynomial time adversary against the protocols presented in this thesis. The protocols presented in this work are based on the some previously established computational problems which we discuss in this section. Additionally, we introduce the additive and the multiplicative trapdoor test described in Section 2.2.7, which are new contributions and are particularly useful for the key agreement protocol presented in Section 6.1.
2.2. Definitions of Computational Problems

2.2.1 Negligible Function

In the proofs for the cryptographic schemes presented in Chapter 4, 5, and 6, we try to show that an adversary has negligible probability of breaking these schemes. The definition of a negligible function was first formalized by Impagliazzo, Levin, and Luby [ILL89].

Definition 2.4. Negligible Function [ILL89]

A function \( p : \mathbb{N} \rightarrow \mathbb{N} \) is negligible with respect to a resource class \( R \) if for all \( r \in R \), for almost all \( n \), \( p(n) \leq 1/r(n) \).

A more intuitive way to describe a negligible function is the following. A function \( \epsilon(k) \) is called negligible in the parameter \( k \) if for every \( c > 0 \) there exists a \( k_c > 0 \) such that for all \( k > k_c \), \( \epsilon(k) < k^{-c} \).

2.2.2 Computational Diffie-Hellman Problem

The computational Diffie-Hellman problem is used in the key agreement protocol in the random oracle model discussed in Section 6.1.

Definition 2.5. Computational Diffie-Hellman Problem [DH76]

The computational Diffie-Hellman (CDH) assumption states that given \( \{g^a, g^b\} \in \mathbb{G}^2 \) it is hard to compute \( g^{ab} \in \mathbb{G} \), where \( \mathbb{G} \) is a cyclic group of prime order \( p \). Let \( Z \) be an algorithm that takes as input the pair \( \{g^a, g^b\} \in \mathbb{G}^2 \), and outputs an element \( T \in \mathbb{G} \). We define the CDH advantage of \( Z \) to be

\[
\text{Adv}_{Z, \mathbb{G}}^{\text{CDH}}(k) = \Pr \left[ a, b \leftarrow Z_p : Z(g^a, g^b) = g^{ab} \right]
\]

We assume that \( \text{Adv}_{Z}^{\text{CDH}}(k) \) is negligible in \( k \) for all \( \text{PPT} \) adversaries \( Z \).

2.2.3 Decisional Diffie-Hellman Problem

The decisional Diffie-Hellman problem is used in the key agreement protocol in the standard model discussed in Section 6.2. The decisional Diffie-Hellman problem is not harder than the computational Diffie-Hellman problem. The decisional Diffie-Hellman problem was first studied by Boneh [Bon98], we use the definition by Boyd et al..
Definition 2.6. Decisional Diffie-Hellman (DDH) Assumption [BCGP08]
Let $G$ be a cyclic group of order $p$ generated by an element $g$. Consider the set $G^3 = G \times G \times G$ and the following two probability distributions over it:

$$\mathcal{R}_G = \{(g^a, g^b, g^c) : (a, b, c) \xleftarrow{\$} \mathbb{Z}_p\}$$

and

$$\mathcal{DH}_G = \{(g^a, g^b, g^{ab}) : (a, b) \xleftarrow{\$} \mathbb{Z}_p\}$$

We say the decisional Diffie-Hellman (DDH) assumption holds over $G = \langle g \rangle$ if the two distributions $\mathcal{R}_G$ and $\mathcal{DH}_G$ are indistinguishable by all polynomial-time adversaries $\mathcal{Z}$. More precisely, for $k = \lceil \log_2(p) \rceil$

$$\text{Adv}^\text{DDH}_{\mathcal{Z}, G}(k) = \left| \Pr[\mathcal{Z}(\rho) = 1 | \rho \xleftarrow{\$} \mathcal{DH}_G] - \Pr[\mathcal{Z}(\rho) = 1 | \rho \xleftarrow{\$} \mathcal{R}_G] \right|$$

is negligible in $k$.

2.2.4 Computational Bilinear Diffie-Hellman Problem

The computational bilinear Diffie-Hellman problem is used in the key agreement protocol in the random oracle model discussed in Section 6.2. It was formally described by Boneh and Franklin [BF03], and named “Weil Diffie-Hellman Assumption”. However, the name “Computational Bilinear Diffie-Hellman Assumption” is now commonly used in the literature.

Definition 2.7. Computational Bilinear Diffie-Hellman (CBDH) Assumption [BF03]

The computational bilinear Diffie-Hellman (CBDH) assumption states that given \{g^a, g^b, g^c\} $\in G^3$ it is hard to compute $e(g, g)^{abc} \in G_T$. Let $\mathcal{Z}$ be an algorithm that takes as input a triple \{g^a, g^b, g^c\} $\in G^3$, and outputs an element $T \in G_T$. We define the CBDH advantage of $\mathcal{Z}$ to be

$$\text{Adv}^\text{CBDH}_{\mathcal{Z}, G, e}(k) = \Pr[a, b, c \xleftarrow{\$} \mathbb{Z}_p : \mathcal{Z}(g^a, g^b, g^c) = e(g, g)^{abc}]$$

We assume that $\text{Adv}^\text{CBDH}_{\mathcal{Z}}(k)$ is negligible in $k$ for all PPT adversaries $\mathcal{Z}$. 
2.2. Definitions of Computational Problems

2.2.5 Decisional Bilinear Diffie-Hellman Problem

The decisional bilinear Diffie-Hellman problem is used in the encryption scheme in the standard model discussed in Section 4.6.2.

Definition 2.8. Decisional Bilinear Diffie-Hellman (DBDH) Assumption [BB04b]

The decisional Bilinear Diffie-Hellman (DBDH) assumption states that given \( \{g^a, g^b, g^c\} \in G^3 \) it is hard to distinguish \( e(g, g)^{abc} \in G_T \) from a random element \( R \xleftarrow{\$} G_T \). Let \( Z \) be an algorithm that takes as input a 4-tuple \( \{g^a, g^b, g^c, T\} \in G^3 \times G_T \), and outputs a bit \( b \in \{0, 1\} \), indicating a guess of whether \( T \stackrel{?}{=} e(g, g)^{abc} \).

We define the dBDH advantage of \( Z \) to be

\[
\text{Adv}_{Z,G,e}^{\text{dBDH}}(k) = \left| \Pr \left[ a, b, c \xleftarrow{\$} Z_p : Z(g^a, g^b, g^c, T) = \left( T \stackrel{?}{=} e(g, g)^{abc} \right) \right] - \frac{1}{2} \right|
\]

We assume that \( \text{Adv}_{Z}^{\text{dBDH}}(k) \) is negligible in \( k \) for all PPT adversaries \( Z \).

2.2.6 Learning With Error Problem

The learning with error problem is used in the encryption scheme in the standard model discussed in Section 4.6.1.

Definition 2.9. Learning With Error Problem (LWE) [CHK09]

Let \( q \geq 2 \) be an integer and let \( \chi \) be a probability distribution on \( \mathbb{Z}_q \). For \( s \in \mathbb{Z}_q^n \), define \( A_{a,\chi} \) to be the distribution on \( \mathbb{Z}_q^n \times \mathbb{Z}_q \) induced by choosing \( a \xleftarrow{\$} \mathbb{Z}_q^n \), \( x \xleftarrow{\$} \chi \), and outputting \( (a, a^t s + x) \) (here, \( a^t \) means the transposition of \( a \)). The learning with error problem \( \text{LWE}_{q,\chi} \) is defined by the following experiment. The adversary \( Z \) must distinguish between the following oracles drawing \( m = \text{poly}(n) \) times.

The first oracle selects \( s \) uniformly at random from \( \mathbb{Z}_q^n \), which remains fixed. To respond to a query, the oracle draws a sample from \( A_{a,\chi} \) and returns it to the adversary. The other oracle draws uniform samples over \( \mathbb{Z}_q^n \times \mathbb{Z}_q \) and returns them to the adversary. We define the LWE advantage of \( Z \) to be

\[
\text{Adv}_{Z}^{\text{LWE}}(k) = \left| \Pr \left[ Z(\{(a_1, b_1), \ldots, (a_m, b_m)\}) = 1 | \{(a_1, b_1), \ldots, (a_m, b_m)\} \xleftarrow{\$} A_{a,\chi} \right] - \Pr \left[ Z(\{(a_1, b_1), \ldots, (a_m, b_m)\}) = 1 | \{(a_1, b_1), \ldots, (a_m, b_m)\} \xleftarrow{\$} \mathbb{Z}_q^n \times \mathbb{Z}_q \right] \right|
\]

We assume that \( \text{Adv}_{Z}^{\text{LWE}}(k) \) is negligible in \( k \) for all PPT adversaries \( Z \).
2.2.7 The Twin Bilinear Diffie-Hellman Trapdoor Theorems

The twin Diffie-Hellman trapdoor test was first presented by Cash, Kiltz and Shoup [CKS08] at Eurocrypt 2008. The idea is to construct a trapdoor that allows testing whether a value generated by an adversary is the value that the simulator is looking for. We present the original theorem given by Cash, Kiltz and Shoup first, then we present and prove two new variants of the theorem that we use in the key agreement protocol presented in Section 6.1.

**Theorem 2.1 (Trapdoor Test [CKS08]).** Let $e : G \times G \rightarrow G_T$ be a bilinear pairing, where $G, G_T$ are two cyclic groups of prime order $p$. Let $g \in G$ be a generator of $G$. Suppose $B_1 \in G$ and $y, z \in \mathbb{Z}_p$ are mutually independent random variables. Define $B_2 := g^y / B_1^z$. Further, suppose that $A, C$ are random variables in $G$ and $T_1, T_2$ are random variables in $G_T$, each of which is defined as some function of $B_1$ and $B_2$. Then we have:

1. $B_2$ is uniformly distributed over $G$.
2. $B_1$ and $B_2$ are independent.
3. If $B_1 = g^{b_1}$ and $B_2 = g^{b_2}$, then the probability that the truth value of
   \[ T_1^z \cdot T_2 \overset{?}{=} e(A, C)^y \]  
   (2.1)

   does not agree with the truth value of
   \[ T_1 \overset{?}{=} e(A, C)^{b_1} \land T_2 \overset{?}{=} e(A, C)^{b_2} \]  
   (2.2)

   is at most $1/p$; moreover, if Equation (2.2) holds, then Equation 2.1 certainly holds.

See Cash, Kiltz and Shoup [CKS08] for a proof.

Additionally we define the “Additive double BDH Trapdoor Test” and the “Multiplicative double BDH Trapdoor Test” to prove security of the key agreement scheme in the random oracle model in Section 6.1.

**Theorem 2.2 (Additive Double BDH Trapdoor Test).** Let $e : G \times G \rightarrow G_T$ be a bilinear pairing, where $G, G_T$ are two cyclic groups of prime order $p$. Let $g \in G$ be a generator of $G$. Suppose $B_1, D_1 \in G, y_1, y_2, z \in \mathbb{Z}_p$ are mutually independent
random variables. Define $B_2 := g^{y_1}/B_1^z$ and $D_2 := g^{y_2}/D_1^z$. Further, suppose that $A, C$ are random variables in $G$ and $T_1, T_2$ are random variables in $G_T$, each of which is defined as some function of $(A, C, B_1, D_1)$ and $(A, C, B_2, D_2)$ respectively. Then we have:

(i) $B_2$ and $D_2$ are uniformly distributed over $G$, as is $B_1 \cdot D_2$.

(ii) $B_1$ and $B_2$ are independent, $D_1$ and $D_2$ are independent, $B_2$ and $D_2$ are independent, and $B_1 \cdot D_1$ and $B_2 \cdot D_2$ are independent.

(iii) If $B_1 = g^{b_1}, B_2 = g^{b_2}, D_1 = g^{d_1}, D_2 = g^{d_2}$, then the probability that the truth value of

$$T_1^z \cdot T_2^z = e(A, C)^{y_1+y_2}$$

(2.3)

does not agree with the truth value of

$$T_1^z = e(A, C)^{b_1+d_1} \wedge T_2^z = e(A, C)^{b_2+d_2}$$

(2.4)

is at most $1/p$; moreover, if Equation (2.4) holds, then Equation (2.3) certainly holds.

Proof. This proof is similar to Cash, Kiltz and Shoup’s [CKS08] trapdoor test proof. Observe that $y_1 + y_2 = z(b_1 + d_1) + (b_2 + d_2)$. It is easy to verify that $B_2 \cdot D_2$ is uniformly distributed over $G$, and that $B_1 \cdot D_1$, $B_2 \cdot D_2$, and $z$ are mutually independent, from which (i) and (ii) follow. To prove (iii), condition on fixed values of $B_1 \cdot D_1$ and $B_2 \cdot D_2$. In the resulting conditional probability space, $z$ is uniformly distributed over $Z_p$, while $(b_1 + d_1), (b_2 + d_2), e(A, C), T_1$ and $T_2$ are fixed. If Equation (2.4) holds, then by multiplying together the two equations in Equation (2.4), we see that Equation (2.3) certainly holds. Conversely, if Equation (2.4) does not hold, we show that Equation (2.3) holds with probability at most $1/p$. Observe that Equation (2.3) is equivalent to

$$\left( \frac{T_1}{e(A, C)^{b_1+d_1}} \right)^z \equiv e(A, C)^{b_2+d_2} \frac{T_2}{T_2}.$$  

(2.5)

It is not hard to see that if $T_1 = e(A, C)^{b_1+d_1}$ and $T_2 \neq e(A, C)^{b_2+d_2}$, then Equation (2.5) certainly does not hold. This leaves us with the case $T_1 \neq e(A, C)^{b_1+d_1}$. But in this case, the left hand side of Equation (2.5) is a random element of $G_T$ (since $z$ is uniformly distributed in $Z_p$), but the right hand side is a fixed element of $G_T$. Thus, Equation (2.5) holds with probability $1/p$ in this case. \qed
Theorem 2.3 (Multiplicative Double BDH Trapdoor Test). Let \( e : G \times G \to G_T \) be a bilinear pairing, where \( G, G_T \) are two cyclic groups of prime order \( p \). Let \( g \in G \) be a generator of \( G \). Suppose \( B_1, C_1 \in G, y_1, y_2, z \in \mathbb{Z}_p \) are mutually independent random variables. Define \( B_2 := g^{y_1}/B_1^z \) and \( C_2 := g^{y_2}/C_1^z \). Further, suppose that \( A \) is a random variables in \( G \) and \( T_1, T_2 \) are random variables in \( G_T \), each of which is defined as some function of \((A, B_1, C_1)\) and \((A, B_2, C_2)\). Then we have:

(i) \( B_2 \) and \( C_2 \) are uniformly distributed over \( G \), and \( e(B_2, C_2) \) is uniformly distributed over \( G_T \).

(ii) \( B_1 \) and \( B_2 \) are independent and \( C_1 \) and \( C_2 \) are independent and \( B_2 \) and \( C_2 \) are independent, and \( e(B_1, C_1) \) and \( e(B_2, C_2) \) are independent.

(iii) If \( B_1 = g^{b_1}, B_2 = g^{b_2}, C_1 = g^{c_1}, C_2 = g^{c_2} \), then the probability that the truth value of

\[
\frac{T_1 z^2}{T_2} \overset{?}{=} \frac{e(A, g)^{y_1 y_2}}{e(A, C_2)^{y_1} e(A, B_1)^{y_2}} \tag{2.6}
\]

does not agree with the truth value of

\[
T_1 \overset{?}{=} e(A, g)^{b_1 c_1} \land T_2 \overset{?}{=} e(A, g)^{b_2 c_2} \tag{2.7}
\]

is at most \(2/p\), moreover, if Equation (2.7) holds, then Equation (2.6) certainly holds.

Proof. Observe that \( y_1 y_2 = (zb_1 + b_2)(zc_1 + c_2) = z^2 b_1 c_1 + zb_1 c_2 + zb_2 c_1 + b_2 c_2 \). It is easy to verify that \( e(B_2, C_2) \) is uniformly distributed over \( G_T \), and that \( e(B_1, C_1), e(B_2, C_2), z \) are mutually independent, from which (i) and (ii) follow. To prove (iii), condition on fixed values of \( e(B_1, C_1) \) and \( e(B_2, C_2) \). In the resulting conditional probability space, \( z \) is uniformly distributed over \( \mathbb{Z}_p \), while \( b_1 c_1, b_2 c_2, A, T_1 \) and \( T_2 \) are fixed. If Equation (2.7) holds, then by multiplying together the two equations in Equation (2.7), we see that Equation (2.6) certainly holds. Conversely, if Equation (2.7) does not hold, we show that Equation (2.6) holds with probability at most \(2/p\). Observe that Equation (2.6) is equivalent to

\[
\left( \frac{T_1}{e(A, g)^{b_1 c_1}} \right)^z = \frac{e(A, g)^{b_2 c_2}}{T_2} \tag{2.8}
\]

It is not hard to see that if \( T_1 = e(A, g)^{b_1 c_1} \) and \( T_2 \neq e(A, g)^{b_2 c_2} \), then Equation
2.3 Definitions of Cryptographic Primitives

(2.8) certainly does not hold. This leaves us with the case $T_1 \neq e(A, g)^{b_1c_1}$. But in this case, the left hand side of Equation (2.8) is the square of a random element of $G_T$. Since $z$ is uniformly distributed in $\mathbb{Z}_p$, $z^2$ is uniformly distributed over half of $\mathbb{Z}_p$ as half of the elements of $\mathbb{Z}_p$ are quadratic residues. On the other hand, the right hand side of 2.8 is a fixed element of $G_T$. Thus, Equation (2.8) holds with probability $2/p$ in this case.

We note that if this test was implemented with $B_2 = g^{y_1}/g^{z_1b}$ and $C_2 = g^{y_2}/g^{z_2c}$, then the probability that Equation (2.7) holds would be $1/p^2$. We use $z$ instead of $z_1$ and $z_2$ because we need Theorem 2.2 (the additive double BDH trapdoor test) simultaneously in Chapter 6. In the key-agreement protocol in Section 6.1, we will use these two theorems to prove that our scheme is secure.

2.3 Definitions of Cryptographic Primitives

In this section, we define some of the cryptographic primitives that we use in the protocols described in the following chapters. Most of the definitions presented here are taken from previous work.

2.3.1 Strong Randomness Extractor

A strong randomness extractor maps randomness contained in $n$ bits to $k$ bits, and is parametrised by the choice of a random hash function. We use a strong randomness extractor in the key agreement protocol in the standard model discussed in Section 6.2.

Definition 2.10. Strong Randomness Extractor [NZ96]

A family of efficiently computable hash functions $\mathcal{H} = \{h_\kappa : \{0,1\}^n \to \{0,1\}^k | \kappa \in \{0,1\}^d\}$ is called a strong $(m, \epsilon)$-randomness extractor, if for any random variable $X$ over $\{0,1\}^n$ that has min-entropy at least $m$, if $\kappa$ is chosen uniformly at random from $\{0,1\}^d$, and $R$ is chosen uniformly at random from $\{0,1\}^k$, the two distributions $\langle \kappa, h_\kappa(X) \rangle$ and $\langle \kappa, R \rangle$ have statistical distance $\epsilon$, that is

$$\frac{1}{2} \sum_{x \in \{0,1\}^k} |\Pr[h_\kappa(X) = x] - \Pr[R = x]| = \epsilon$$
Chapter 2. Theoretical Background

To implement the randomness extraction function, one could apply the work of Dodis et al. [DGH+04] to use a block cipher or a hash function as a randomness extractor.

2.3.2 Pseudorandom Function Family

We use a pseudorandom function family in the key agreement protocol in the standard model discussed in Section 6.2. Pseudorandom functions were first introduced by Goldreich, Goldwasser and Micali [GGM86], however they were named poly-random collections. We use the definition given by Boyd et al. [BCGP08].

Definition 2.11. Pseudorandom Function Family (PRF) [BCGP08]

Let \( \mathcal{F} = \{f_s\}_{s \in S} \) be a family of functions for security parameter \( k \in \mathbb{N} \) and with seed \( s \in S = S(k) \). Let \( C \) be an adversary that is given oracle access to either \( F_s \) for \( s \leftarrow K \) or a truly random function \( \text{Rand} \) with the same domain and range as the functions in \( \mathcal{F} \). \( \mathcal{F} \) is said to be pseudorandom if \( C \)'s advantage in distinguishing whether it has access to a random member of \( \mathcal{F} \) or a truly random function is negligible in \( k \), for all polynomial-time adversaries \( C \). That is,

\[
\text{Adv}_{\mathcal{F},C}^{p-rand}(k) = \left| \Pr[C^{F_s}(\cdot)(k) = 1] - \Pr[C^{\text{Rand}}(\cdot)(k) = 1] \right|
\]

(2.9)

is negligible in \( k \).

Functions that are proven to be pseudorandom include CBC-MAC [BKR00] (provided the underlying block cipher is a secure pseudorandom permutation family and the input length is constant) and HMAC [Bel06] (provided the compression function is a pseudorandom function (PRF)).

2.3.3 Target Collision Resistant Hash Function

We use a target collision resistant hash function (TCR) as one of the building blocks in our key encapsulation mechanism discussed in Chapter 5. Rogaway and Shrimpton [RS04] discuss the historical development of notions of security for hash functions. According to them, Bellare and Rogaway [BR97] first used the term TCR. The notion of a TCR is called “eSec” by Rogaway and Shrimpton [RS04]. We use the definition of a TCR given by Kiltz and Galindo [KG06b],
2.3. Definitions of Cryptographic Primitives

because the KEM discussed in Chapter 5 uses the work of Kiltz and Galindo as a building block.

**Definition 2.12. Target Collision Resistant Hash Function (TCR)**[KG06b]

Let \( F = (TCR_s)_{s \in S} \) be a family of hash functions for security parameter \( k \) and with seed \( s \in S \) where \( S \) is parametrized by the security parameter \( k \). \( F \) is said to be collision resistant if, for a hash function \( TCR = TCR_s \) with \( s \leftarrow S \), it is infeasible for an efficient adversary to find two distinct values \( x \neq y \) such that \( TCR(x) = TCR(y) \).

The notion of a target collision resistant hash function (TCR) is strictly weaker. The adversary against a target collision resistant hash function is supplied with a randomly drawn hash function \( TCR = TCR_s \) and a randomly chosen element \( x \). The task of the adversary is to find a \( y \) such that \( TCR(x) = TCR(y) \). Note that the adversary may not select \( x \), and is thus limited with respect to collision resistant hash functions. Target collision resistant hash functions are sometimes also called universal one-way hash functions. In the following we assume that TCR’s exist and define the advantage of any efficient polynomial time adversary \( M \) against a randomly chosen hash function \( TCR = TCR_s \) as

\[
Adv_{TCR, M}^{\text{hash-tcr}}(k) = \Pr[y \leftarrow M(TCR(\cdot), x) | TCR(y) = TCR(x)]
\]

The hash function \( TCR \) is said to be target collision resistant if the advantage for all \( M \) against \( TCR \) is negligible in \( k \).

Naor and Yung [NY89] and Rompel [Rom90] give efficient constructions for target collision resistant hash functions from arbitrary one-way hash functions.

2.3.4 Waters’ Hash

We use Waters’ hash function as one of the building blocks in our key encapsulation mechanism discussed in Chapter 5. It was first described in Waters’ identity-based encryption scheme [Wat05].

**Definition 2.13. Waters’ Hash [KG06b]**

On input of an integer \( n \), the randomized hash key generator \( HGen(G) \) chooses \( n+1 \) random group elements \( h_0, h_1, \ldots, h_n \in G \) and returns \( h = (h_0, h_1, \ldots, h_n) \)
as the public description of the hash function. The hash function $H : \{0, 1\}^n \rightarrow G^*$ is evaluated on a string $ID = (ID_1, \ldots, ID_n) \in \{0, 1\}^n$ as the product

$$H(ID) = h_0 \prod_{i=1}^{n} h_{ID_i}^i.$$

### 2.4 A Short Introduction to Provable Security

In the following sections, we will prove the security of multiple protocols. To prove the security of a protocol, we make certain assumptions. In particular, we assume that if there exists an adversary that has an advantage against the protocol, then the adversary has to interact with the protocol. Furthermore, we assume that we can isolate this adversary and simulate the protocol for the adversary. We also assume that the adversary has limited computational resources, and that the main quality of the adversary is that it can break the protocol in probabilistic polynomial time. With probabilistic polynomial time we mean that the adversary has access to a source of randomness and using this source of randomness is able to break the protocol in polynomial time.

To prove the protocol secure, we construct a simulator that interacts with the adversary. The simulator is given a problem that is according to current scientific standards not solvable in polynomial time, for example one of the problems listed in Section 2.2. An instance of the protocol is then constructed around the given hard problem, and we show that this instance is indistinguishable from any other randomly generated instance of the protocol. In one run of the protocol, that is also not distinguishable from any other run of the protocol, the hard problem is then injected. It is hoped that the adversary will attack the protocol run that contains the hard problem, and present its successful attack by disclosing some information about the protocol instance that cannot not be computed otherwise. If that happens, then the simulator can use the information presented by the adversary to obtain a solution to the hard problem. Notice that then the adversary must have solved the problem to be able to present the information, and that the adversary runs in probabilistic polynomial time. But according to current scientific standards, the problem is not solvable in polynomial time, therefore it follows that no probabilistic polynomial time adversary against the protocol exists.

The above describes the general idea behind a security proof in cryptographic
protocols. As we explained before, the security proof is an abstraction of a protocol run in the real world. To make it easier to model the real world, a number of actions (later we usually call them queries) that the adversary may make in its interaction with the protocol is specified. In particular, it is allowed for the adversary to corrupt parties and learn their secrets. Since these secrets allow to execute the protocol in polynomial time (they provide the necessary trapdoor for the party executing the protocol to run the protocol in polynomial time), it is not considered a break of the protocol if the adversary simply learns all secrets that a party executing the protocol has and then presents its result.

We assume that although the adversary may learn the secrets of any party, an adversary has an advantage against a protocol only if the party that it attacks has at least one secret that the adversary does not know. Otherwise it would be trivial to break the protocol.

Furthermore, the simulator allows the adversary access to oracles. An oracle is a source of information that the adversary cannot compute on its own. One type of oracle provides the adversary with information so that the adversary can check that the protocol behaves as expected. Examples for this type of oracle are decryption oracles that allow the adversary to decrypt arbitrary ciphertexts and learn the corresponding plaintext, or session key reveal oracles that allow the adversary to learn the key used in a session between two parties in a key agreement protocol. This type of oracle exists both in the random oracle model and in the standard model.

Then there is another type of oracles, and this type is specific to the random oracle model and makes security proofs in the random oracle model easier. A way to think of this type of oracle is to think of a table consisting of key-value pairs, where the adversary can ask for any key, but the returned value is unpredictable. In fact, in the random oracle model, the simulator controls the value that is returned. Since it is prescribed in the protocol run to request values from the table (or the “random oracle”) during certain stages of the protocol, these requests allow the simulator to monitor the computations of the adversary and to extract data that the adversary computed (the keys). Furthermore, it allows the simulator to inject values into the computation of the adversary by returning values that look random but have a special meaning for the simulator. For example, the simulator can use the random oracle to inject values that are related to the hard problem that the simulator would like to solve. In a real
world protocol run these random oracles are then replaced by hash functions, since random oracles do not exist in reality and would be impractical to use in a protocol as the oracle would have to be queried during each execution of the protocol. As we said before, random oracles (or tables that can be manipulated by the simulator during the protocol run) are specific to the random oracle model. The standard model has only oracles that provide information to the adversary that can be computed from exchanged messages during the protocol run and the private keys of the parties that participated in the protocol run (the first type of oracles discussed above). As already discussed in the introduction, the scientific community values proofs in the standard model higher than proofs in the random oracle model, because random oracles do not exist in reality. However, protocols that are proven in the random oracle model are usually computationally more efficient than protocols in the standard model, and the proofs are easier to write.

Examples for security games between a simulator and an adversary are in Figure 2.1, 2.2, and 5.2. More detailed proofs of security can be found in Chapter 4, 5, and 6.

2.5 Signatures, Message Authentication Codes, and Encapsulation Schemes

We do not develop any new signature schemes, message authentication codes (MACs), or encapsulation schemes in this thesis, but make use of these primitives in the encryption schemes presented in Chapter 4.

Signature schemes allow signing a message using a public/private key pair. The private key is for signing, the public key for verifying the signature. If the signature is sent with the message, a receiver who trusts the public key can verify that the message was unmodified during transit and originated from the sender that the public key belongs to. The definition of a signature scheme was given by Goldwasser, Micali and Rivest [GMR88]. Our construction in Chapter 4 extends the work by Boneh et al. [BCHK07]. In the following we present the definitions given by Boneh et al. [BCHK07] for consistency with their work.

Definition 2.14. Signature Scheme [BCHK07]
A signature scheme is a triple of PPT algorithms $\text{Sig} = (G, \text{Sign}, \text{Verify})$ such that:

- The randomized key generation algorithm $G$ takes as input the security
parameter $1^k$ and outputs a verification key $vk$ and a signing key $sk$. We write $(vk, sk) \leftarrow G(1^k)$.

- The possibly randomized signing algorithm $\text{Sign}$ takes as input a signing key $sk$ and a message $m$ (in some implicit message space), and outputs a signature $\sigma$. We write $\sigma \leftarrow \text{Sign}(sk, m)$.

- The possibly randomized signature verification algorithm $\text{Verify}$ takes as input a verification key $vk$, a message $m$, and a signature $\sigma$, and outputs a bit $b \in \{0, 1\}$ (where $b = 1$ signifies “acceptance” and $b = 0$ signifies “rejection”). We write $b \leftarrow \text{Verify}(vk, m, \sigma)$.

We require that $\forall (vk, sk) \leftarrow G : \text{Verify}(vk, m, \text{Sign}(m, sk)) = 1$.

**Definition 2.15. Strong One-Time Signature Scheme** [BCHK07]  
A signature scheme $\text{Sig}$ is a strong one-time signature scheme if the success probability of any PPT adversary $\mathcal{M}$ in the following game is negligible in the security parameter $k$:

1. $(vk, sk) \leftarrow G(1^k)$. $\mathcal{M}$ is given $1^k$ and $vk$.

2. $\mathcal{M}(1^k, vk)$ may do one of the following:

   (a) $\mathcal{M}$ may output a pair $(m^*, \sigma^*)$ and halt. In this case $(m, \sigma)$ are undefined.

   (b) $\mathcal{M}$ may output a message $m$, and is then given in return a signature $\sigma \leftarrow \text{Sign}(m, sk)$. Following this, $\mathcal{M}$ outputs $(m^*, \sigma^*)$.

We say the adversary succeeds if $\text{Verify}(m^*, \sigma^*) = 1$ but $(m^*, \sigma^*) \neq (m, \sigma)$ (assuming $m$ and $\sigma$ are defined). We stress that the adversary may succeed even if $m^* = m$.

Message authentication codes as discussed for example in [BKR00] rely on a shared symmetric key to allow all persons who know the shared key to verify the integrity of a message. We present the definition by Boneh et al. [BCHK07] for ease of understanding in Chapter 4.

**Definition 2.16. Message Authentication Code (MAC)** [BCHK07]  
A message authentication code (MAC) is a pair of PPT algorithms $(\text{Mac}, \text{Verify})$ such that:
• The tagging algorithm $\text{Mac}$ takes as input a key $sk \in \{0,1\}^k$ (where $k$ is the security parameter) and a message $m$. It outputs a tag $\text{tag}$ by computing $\text{tag} \leftarrow \text{Mac}_{sk}(m)$.

• The verification algorithm $\text{Verify}$ takes as input a key $sk \in \{0,1\}^k$, a message $m$, and a tag $\text{tag}$. It outputs a bit $b \in \{0,1\}$ where $b = 1$ signifies “acceptance” and $b = 0$ signifies “rejection”. We write this as $b \leftarrow \text{Verify}_{sk}(m, \text{tag})$.

We require $\forall (sk, m) : \text{Verify}_{sk}(m, \text{Mac}_{sk}(m)) = 1$.

Definition 2.17. Strong One-Time Message Authentication Code $[\text{BCHK07}]$

A message authentication code $(\text{Mac}, \text{Verify})$ is a strong one-time message authentication code if the success probability of any PPT adversary $\mathcal{M}$ in the following game is negligible in the security parameter $k$.

1. A random key $sk \in \{0,1\}^k$ is chosen.

2. $\mathcal{M}(1^k)$ may do one of the following:
   (a) $\mathcal{M}$ may output $(m^*, \text{tag}^*)$. In this case $(m, \text{tag})$ are undefined.
   (b) $\mathcal{M}$ may output a message $m$ and is then given in return a tag $\text{tag} \leftarrow \text{Mac}_{sk}(m)$. Following this, $\mathcal{M}$ outputs $(m^*, \text{tag}^*)$.

We say the adversary succeeds if $\text{Verify}_{sk}(m^*, \text{tag}^*) = 1$ but $(m^*, \text{tag}^*) \neq (m, \text{tag})$ (assuming that $(m, \text{tag})$ are defined). We stress that the adversary may succeed even if $m^* = m$.

An encapsulation scheme is similar to a weak commitment scheme. In a commitment scheme, the sender chooses a string and sends a commitment to the string to the verifier. At a later time, the sender opens the commitment and reveals the string. Using the commitment, the verifier can verify that the sender disclosed the correct string and not some arbitrary string. The first commitment scheme was proposed by Blum $[\text{Blu82}]$. The notion was formalized and a practical scheme was proposed by Halevi and Micali $[\text{HM96}]$. The notion of an encapsulation scheme was formalized by Boneh et al. $[\text{BCHK07}]$. In an encapsulation scheme, the sender commits to a random string as opposed to a string chosen by the sender in a commitment scheme.
Definition 2.18. Encapsulation Scheme [BCHK07]

An encapsulation scheme $\Pi$ is a triple of PPT algorithms $(\text{Init}, S, R)$ such that

- $\text{Init}$ takes as input the security parameter $1^k$ and outputs a string $\text{pub}$.
- $S$ takes as input $1^k$ and $\text{pub}$ and outputs $(r, \text{com}, \text{dec})$ with $r \in \{0, 1\}^k$. We refer to $\text{com}$ as the commitment string and to $\text{dec}$ as the decommitment string.
- $R$ takes as input $(\text{pub}, \text{com}, \text{dec})$ and outputs $r \in \{0, 1\}^k \cup \{\bot\}$, where $\bot$ indicates an invalid commitment.

We require that for all $\text{pub}$ output by $\text{Init}(1^k)$ and for all $(r, \text{com}, \text{dec})$ output by $S(1^k, \text{pub})$, we have $R(\text{pub}, \text{com}, \text{dec}) = r$.

A concrete encapsulation scheme is given by Boneh et al. [BCHK07]. They propose an encapsulation scheme based on any universal one-way hash function (UOWHF) family $\{H_s : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{k}\}$ (where $k_1 \geq 3k$ is a function of the security parameter $k$). The scheme works as follows:

- $\text{Init}$ chooses a hash function $h$ from a family of pairwise-independent hash functions mapping $k_1$-bit strings to $k$-bit strings, and also chooses at random a key $s$ defining UOWHF $H_s$. It outputs $\text{pub} = (h, s)$.
- The encapsulation algorithm $S$ takes $\text{pub}$ as input, chooses a random $x \in \{0, 1\}^{k_1}$, and then outputs $(r = h(x), \text{com} = H_s(x), \text{dec} = x)$.
- The recovery algorithm $R$ takes as input $((h, s), \text{com}, \text{dec})$ and outputs $h(\text{dec})$ if $H_s(\text{dec}) = \text{com}$, and $\bot$ otherwise.

We use this encapsulation scheme in the encryption scheme proposed in Section 4.6.2.

Since an encapsulation scheme has similar functionality to a commitment scheme, it also satisfies the notions of “hiding” and “binding”. The “hiding” requirement of the encapsulation scheme means that given $\text{pub}, \text{com}$, and $r$, $r$ should be indistinguishable from random. The “binding” requirement means that a $\text{com}$ output by $S$ can be opened only to one single value of $r$. We formalize this notion now.
Definition 2.19. Hiding and Binding Properties of an Encapsulation Scheme [BCHK07]

An encapsulation scheme $\Pi$ is said to be hiding if the success probability of any PPT adversary $\mathcal{M}$ is negligible in the following game:

1. $\text{pub} \leftarrow \text{Init}(1^k)$.
2. $r_0$ is randomly drawn from $\{0, 1\}^k$: $r_0 \leftarrow \{0, 1\}^k$.
3. $(r_1, \text{com}, \text{dec})$ are constructed using $S$: $(r_1, \text{com}, \text{dec}) \leftarrow S(1^k, \text{pub})$.
4. A bit $b$ is randomly chosen, and $\mathcal{M}$ is given $(r_b, \text{com}, \text{pub})$.
5. Finally, $\mathcal{M}$ outputs a guess $b'$ for $b$.

We say that the adversary $\mathcal{M}$ succeeds if $b' = b$, and denote the probability of this event by $\Pr_{\mathcal{M},\Pi}^{\text{hiding}}[\text{Succ}]$. The adversary’s advantage is defined as $\text{Adv}_{\mathcal{M},\Pi}^{\text{hiding}}(k) = \left| \Pr_{\mathcal{M},\Pi}^{\text{hiding}}[\text{Succ}] - \frac{1}{2} \right|$.

An encapsulation scheme $\Pi$ is said to be binding if the success probability of any PPT adversary $\mathcal{M}$ is negligible in the following game:

1. $\text{pub} \leftarrow \text{Init}(1^k)$.
2. $(r, \text{com}, \text{dec})$ are constructed using $S$: $(r, \text{com}, \text{dec}) \leftarrow S(1^k, \text{pub})$.
3. $\mathcal{M}$ is given $(\text{pub}, \text{com}, \text{dec})$.
4. Finally, $\mathcal{M}$ outputs a $\text{dec}'$ so that $\text{dec}' \neq \text{dec}$.

We say that the adversary $\mathcal{M}$ succeeds if $\text{dec}' = \text{dec}$, and denote the probability of this event by $\Pr_{\mathcal{M},\Pi}^{\text{binding}}[\text{Succ}]$. The adversary’s advantage is defined as $\text{Adv}_{\mathcal{M},\Pi}^{\text{binding}}(k) = \left| \Pr_{\mathcal{M},\Pi}^{\text{binding}}[\text{Succ}] - \frac{1}{2} \right|$.

Boneh et al. [BCHK07, Section 7.2] describe a very efficient encapsulation scheme based on a universal one-way hash function family.

2.6 Encryption

An encryption scheme is used to send a message from a sender to a receiver using only the receiver’s public key. An adversary against an encryption scheme supplies two messages and then gets back an encryption of one of the messages under the public key. The adversary has to distinguish which of the two messages was encrypted. We will now formalize this notion.
2.6. Encryption

2.6.1 Public Key Encryption

We begin by describing the algorithms of a public key encryption scheme. As already discussed in the introduction, Diffie and Hellman [DH76] were the first to discuss the idea of a public key encryption scheme. Rivest, Shamir and Adleman [RSA78] published the first public key encryption scheme (RSA). However, RSA as described by Rivest et al. [RSA78] is a deterministic algorithm. Goldwasser and Micali [GM84] described the idea of a probabilistic public key encryption scheme, however their encryption scheme had a ciphertext expansion of approximately 1:1000 (one bit was encrypted in a message the size of an RSA modulus). ElGamal [Eig85] published a practical encryption scheme that is still used today. Bellare and Rogaway [BR94] then showed how to encrypt with RSA in a probabilistic way. We use the definition by Boneh et al. [BCHK07] to describe the basic functionality a user would expect from an encryption scheme.

**Definition 2.20. Public Key Encryption Scheme [BCHK07]**

A public key encryption scheme is a triple of PPT algorithms \((\text{Gen}, \text{Enc}, \text{Dec})\) such that:

- **The randomized key generation algorithm** \(\text{Gen}\) **takes as input a security parameter** \(1^k\) **and outputs a public key** \(pk\) **and a secret key** \(sk\). We write \((pk, sk) \leftarrow \text{Gen}(1^k)\).

- **The randomized encryption algorithm** \(\text{Enc}\) **takes as input a public key** \(pk\) **and a message** \(m\) (in some implicit message space) and outputs a ciphertext \(C\). We write \(C \leftarrow \text{Enc}(pk, m)\).

- **The (possibly randomized) decryption algorithm** \(\text{Dec}\) **takes as input a ciphertext** \(C\) **and a secret key** \(sk\). It returns a message or the special symbol \(\perp\) which is not in the message space and indicates an invalid ciphertext. We write \(m \leftarrow \text{Dec}(sk, C)\).

We require \(\forall (pk, sk) \leftarrow \text{Gen}(1^k) : \text{Dec}(sk, \text{Enc}(pk, m)) = m\).

The previous definition does not specify any security guarantees. There are two commonly used definitions for security in encryption schemes (see also Bellare and Rogaway [BDPR98] for a more detailed discussion) namely indistinguishable based security against adaptive chosen plaintext attacks (IND-CPA security) and indistinguishable based security against adaptive chosen ciphertext attacks.
Chapter 2. Theoretical Background

Figure 2.1: A diagram for the IND-CPA security game

(IND-CCA2 security). A diagram showing the security experiment for IND-CPA security is given in Figure 2.1. IND-CCA2 security defines the best currently known security level for public key encryption. Decryption (or, according to the definition of IND-CCA2 security, distinguishability) of a message sent in a public key encryption scheme should be infeasible even if the plaintext of all other messages can be recovered. A diagram showing the IND-CCA2 security experiment is given in Figure 2.2.

Definition 2.21. IND-CPA Secure Public Key Encryption

A public-key encryption scheme $\Pi$ is secure against chosen plaintext attacks based on indistinguishability (IND-CPA secure) if the advantage of any PPT adversary $\mathcal{M}$ in the following game is negligible in the security parameter $k$:

1. $(pk, sk) \leftarrow \text{Gen}(1^k)$. Adversary $\mathcal{M}$ is given $pk$.

2. At some point, $\mathcal{M}$ outputs two messages $m_0, m_1 : |m_0| = |m_1|$. A bit $b$ is randomly chosen (we write $b \overset{\$}{\leftarrow} \{0,1\}$), and the adversary is given a challenge ciphertext $C^* = \text{Enc}(pk, m_b)$.

- Adversary with access to public key cannot distinguish between encryption of either of two messages of his choice.
2.6. Encryption

• Adversary with access to public key and decryption oracle cannot distinguish between encryption of either of two messages of his choice.

• Adversary must not ask for a decryption of the challenge ciphertext.

Figure 2.2: A diagram for the IND-CCA2 security game

3. Finally, $\mathcal{M}$ outputs a guess $b'$ for $b$.

We say that the adversary $\mathcal{M}$ succeeds if $b' = b$, and denote the probability of this event by $\Pr^{\text{IND-CPA}}_{\mathcal{M},\Pi}[\text{Succ}]$. The adversary’s advantage is defined as

$$\text{Adv}^{\text{IND-CPA}}_{\mathcal{M},\Pi}(k) = |\Pr^{\text{IND-CPA}}_{\mathcal{M},\Pi}[\text{Succ}] - \frac{1}{2}|$$

Definition 2.22. IND-CCA2 Secure Public Key Encryption [BCHK07]

A public-key encryption scheme $\Pi$ is secure against adaptive chosen ciphertext attacks based on indistinguishability (IND-CCA2 secure) if the advantage of any PPT adversary $\mathcal{M}$ in the following game is negligible in the security parameter $k$:

1. $(pk, sk) \leftarrow \text{Gen}(1^k)$. Adversary $\mathcal{M}$ is given $pk$.

2. The adversary may make polynomially-many queries to a decryption oracle $\text{Dec}_{sk}(\cdot)$. 
3. At some point, \( \mathcal{M} \) outputs two messages \( m_0, m_1 : |m_0| = |m_1| \). A bit \( b \) is randomly chosen (we write \( b \overset{\$}{\leftarrow} \{0,1\} \)), and the adversary is given a challenge ciphertext \( C^* = \text{Enc}(pk, m_b) \).

4. \( \mathcal{M} \) may continue to query its decryption oracle \( \text{Dec}_{sk}(\cdot) \), except that it may not request the decryption of \( C^* \).

5. Finally, \( \mathcal{M} \) outputs a guess \( b' \) for \( b \).

We say that the adversary \( \mathcal{M} \) succeeds if \( b' = b \) and denote the probability of this event by \( \Pr^{\text{IND-CCA2 PKE}}_{\mathcal{M}, \Pi}[\text{Succ}] \). The adversary’s advantage is defined as

\[
\text{Adv}^{\text{IND-CCA2 PKE}}_{\mathcal{M}, \Pi}(k) = \left| \Pr^{\text{IND-CCA2 PKE}}_{\mathcal{M}, \Pi}[\text{Succ}] - \frac{1}{2} \right|
\]

### 2.6.2 Identity-Based Encryption (IBE)

We now proceed to define the functionality that would be expected from an identity encryption scheme. These schemes feature an additional user key derivation function \( \text{KeyDer} \), with which an ID-specific secret key can be derived from the master secret key.

This functionality reduces the amount of work necessary for key management compared to public key cryptography, which was discussed in Chapter 1. Since encryption can be performed to arbitrary identities, it is common practice to simply append a timestamp (e.g. a week number, or any granularity the sender deems necessary) to an identity during encryption. This makes sure that the receiver has to get a recent key from the key generation centre. Therefore there is less necessity for revocation lists in ID-based encryption. It is of course necessary for the sender to send the full identity that the message was encrypted under in plaintext along with the message as otherwise decryption will fail in many cases. However, even if the sender forgot to send this information, the trusted authority is in many schemes able to recover the message from the ciphertext using the master secret key. However, if the trusted authority does this, and discloses the result to any user, this effectively defeats the purpose of identity-based cryptography. This is because then any user could approach the trusted authority and claim that a message was encrypted under an unknown identity and request decryption of the message. Since most encryption schemes use hashing to derive an encryption key from an identity, it is hard to falsify the claim. This then would open a backdoor that allows decryption of arbitrary
messages for any user. Therefore, this approach is not recommended, and all necessary information needed for decryption has to be transmitted along with the ciphertext.

As we will see in Chapter 3, a certificateless encryption scheme also features all of the functionality of an ID-based scheme with the addition of a user key generation algorithm that is similar to the setup algorithm of a public key only scheme.

The first identity-based encryption scheme was proposed by Boneh and Franklin [BF03]. We give a recent definition by Boneh et al. [BCHK07] for the functionality of an identity-based encryption scheme.

**Definition 2.23. Identity-Based Encryption (IBE) Scheme [BCHK07]**

An identity-based encryption scheme (IBE) for identities of length \( n \) (where \( n \) is a polynomially-bounded function) is a tuple of PPT algorithms \( \text{IBE} = (\text{Setup}, \text{KeyDer}, \text{IBE Enc}, \text{IBE Dec}) \) such that:

- **The randomized setup algorithm** \( \text{Setup} \) takes as input a security parameter \( 1^k \). It outputs a master public key \( \text{mpk} \) and a master secret key \( \text{msk} \) and system parameter \( \text{param} \). We write \( (\text{param}, \text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^k) \).

- **The (possibly randomized) key derivation algorithm** \( \text{KeyDer} \) takes as input the master secret key \( \text{msk} \), an identity \( \text{ID} \) and the system parameters \( \text{param} \). It returns the corresponding decryption key \( d_{\text{ID}} \) (also named ID-based private key) for identity \( \text{ID} \). We write \( d_{\text{ID}} \leftarrow \text{KeyDer}(\text{param}, \text{msk}, \text{ID}) \).

- **The randomized encryption algorithm** \( \text{IBE Enc} \) takes as input the master public key \( \text{mpk} \), the system parameters \( \text{param} \), the identity \( \text{ID} \) and the message \( m \). It outputs a ciphertext \( C \). We write \( C \leftarrow \text{IBE Enc}(\text{param}, \text{mpk}, \text{ID}, m) \).

- **The (possibly randomized) decryption algorithm** \( \text{IBE Dec} \) takes as input the system parameters \( \text{param} \), the decryption key \( d_{\text{ID}} \), an identity \( \text{ID} \) and the ciphertext \( C \). It returns the message \( m \) or the symbol \( \perp \) (which is not in the message space). We write \( m \leftarrow \text{IBE Dec}(\text{param}, d_{\text{ID}}, \text{ID}, C) \).

We require that \( \forall (\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^k); \ \forall \text{ID} \in \{0, 1\}^n : \text{IBE Dec}(\text{param}, \text{KeyDer}(\text{param}, \text{msk}, \text{ID}), \text{ID}, \text{IBE Enc}(\text{param}, \text{mpk}, \text{ID}, m)) = m. \)
With respect to security notions for ID-based encryption, we discuss the weaker notion of IND-CPA security, where the adversary is allowed to request secret keys for all but the target identity, but has no access to a decryption oracle for the target identity at all. IND-CCA2 security in the ID-based case is discussed next and gives the adversary access to a decryption oracle for the target identity as well (except for the challenge ciphertext). The reason for discussing IND-CPA security is that all ID-based schemes that are IND-CPA secure can be converted to IND-CCA2 secure public key only encryption schemes (without an ID-based component) with the techniques explained by Boneh et al. [BCHK07]. These techniques require only selective-identity secure ID-based encryption schemes to start with. Selective-identity (SID) secure means that the adversary has to declare the challenge identity in the security experiment before he is given the public parameters of the experiment. On the other hand, there are fully secure ID-based schemes. In fully secure ID-based schemes the adversary selects the challenge identity at the time when the challenge ciphertext is requested. Below we give definitions for the weaker security level of selective identity secure ID-based encryption, because that is all that is needed for the Boneh, Canetti, Halevi and Katz transformations that we will study in more detail in Chapter 4. We note that all fully secure ID-based schemes are of course also selective identity secure, but not vice versa. The SID-IND-CPA and SID-IND-CCA2 security notions for certificateless encryption will be discussed in Chapter 3. To make it easier for the reader to follow the presentation in Chapter 4, we use the definitions for SID-IND-CPA secure encryption schemes by Boneh et al. [BCHK07], since we build upon the results by Boneh et al. in Chapter 4.

Definition 2.24. SID-IND-CPA IBE [BCHK07]
An identity-based encryption scheme $\text{IBE} = (\text{Setup}, \text{KeyDer}, \text{IBE Enc}, \text{IBE Dec})$ for identities of length $n$ is secure against selective-identity adaptive chosen plaintext attacks (SID-IND-CPA secure) if the advantage of any PPT adversary $\mathcal{M}$ in the following game is negligible in the security parameter $k$:

1. $\mathcal{M}(1^k)$ outputs a target identity $ID^* \in \{0,1\}^{n(k)}$.
2. $(mpk, msk) \leftarrow \text{Setup}(1^k)$. The adversary is given $mpk$.
3. The adversary $\mathcal{M}$ may make polynomially-many queries to an oracle $\text{KeyDer}_{msk}(\cdot)$, except that it may not request a secret key corresponding to the target identity $ID^*$. 
2.6. Encryption

The adversary is allowed to query this oracle repeatedly using the same identity; if KeyDer<sub>msk</sub> is randomized, then a different secret key may possibly be returned each time.

4. At some point, \( M \) outputs two messages \( m_0, m_1 : |m_0| = |m_1| \). A bit \( b \in \{0,1\} \) is randomly chosen and the adversary is given the challenge ciphertext \( C^* \leftarrow \text{IBE Enc}(\text{param}, \text{ID}^*, m_b) \).

5. \( M \) may continue to query KeyDer<sub>msk</sub>(·) as above. Finally \( M \) outputs a guess \( b' \) for \( b \).

We say that \( M \) succeeds if \( b' = b \), and denote the probability event by

\[
\text{SID–IND–CPA IBE}_{\text{IBE}}([\text{Succ}]).
\]

The adversary’s advantage is defined as

\[
\text{Adv}^{\text{SID–IND–CPA IBE}}_{\text{IBE}}(k) = \left| \text{SID–IND–CPA IBE}_{\text{IBE}}([\text{Succ}]) - \frac{1}{2} \right|.
\]

Definition 2.25. SID–IND–CCA2 IBE [BCHK07]

The SID–IND–CPA IBE definition can be extended to an identity-based encryption scheme IBE that is secure against selective-identity adaptive chosen ciphertext attacks (SID–IND–CCA2 secure) by allowing the adversary \( M \) additional access to an oracle IBE Dec(·) such that for any ciphertext \( C \), IBE Dec(\( C \)) returns IBE Dec(\( d_{\text{ID}^*}, C \)) where \( d_{\text{ID}^*} \) is the secret key associated with the target identity \( \text{ID}^* \). The adversary has access to this oracle throughout the game but may not submit the challenge ciphertext \( C^* \).

2.6.3 Hierarchical Identity-Based Encryption (HIBE)

Hierarchical identity-based encryption (HIBE) was first introduced by Gentry and Silverberg [GS02]. Since the key generation centre (KGC) in identity-based encryption must generate the private keys for all of its users, it experiences high work loads (depending on the lifetime of ID-based private keys, these keys may have to be issued for every message received by a user). To ease the load that the key generation centre experiences, a hierarchical structure is employed, and each user gets its keys from the hierarchical node that is directly above it on
the path to the root node. This key delegation structure allows reduction of the work load that the root node receives.

Furthermore, the techniques described by Boneh et al. [BCHK07] give constructions for SID-IND-CCA2 secure \((l-1)\)-level hierarchical encryption schemes from existing \(l\)-level SID-IND-CPA secure hierarchical identity based encryption scheme (HIBE) schemes. We will show in Chapter 4 that these techniques are also applicable to certificateless encryption.

**Definition 2.26. Hierarchical Identity-Based Encryption (HIBE) [BCHK07]**

An \(l\)-level HIBE scheme for identities of length \(n\) (where \(l, n\) are polynomially-bounded functions) is a tuple of PPT algorithms \(HIBE = (\text{Setup}, \text{KeyDer}, \text{HIBE Enc}, \text{HIBE Dec})\) such that:

- **The randomized setup algorithm** \(\text{Setup}\) takes as input a security parameter \(1^k\). It outputs a master public key \(mpk\) and a master secret key \(msk\) and system parameter \(param\), defined by \(k, l = l(k), n = n(k)\). We write \((param, mpk, msk) \leftarrow \text{Setup}(1^k)\).

- **The (possibly randomized) key derivation algorithm** \(\text{KeyDer}\) takes as input an identity vector \(v = (ID_1, \ldots, ID_j) \in \{0,1\}^n\)\(^{<l}\), the ID-based secret key \(d_v\) (we define \(d_\emptyset = msk\)), the system parameters \(param\), and a string \(r \in \{0,1\}^n\). It returns the decryption key \(d_{v \circ r}\) for identity \(v \circ r\). We write \(d_{v \circ r} \leftarrow \text{KeyDer}(param, d_v, v, r)\).

- **The randomized encryption algorithm** \(\text{HIBE Enc}\) takes as input the master public key \(mpk\), the system parameters \(param\), the identity vector \(v \in \{0,1\}^n\)^{\leq l} and the message \(m\). It outputs a ciphertext \(C\). We write \(C \leftarrow \text{HIBE Enc}(param, mpk, v, m)\).

- **The (possibly randomized) decryption algorithm** \(\text{HIBE Dec}\) takes as input the system parameters \(param\), the decryption key \(d_v\), an identity vector \(v\) and the ciphertext \(C\). It returns the message \(m\) or the symbol \(\perp\) (which is not in the message space). We write \(m \leftarrow \text{HIBE Dec}(param, d_v, v, C)\).

We require that \(\forall (mpk, msk) \leftarrow \text{Setup}(1^k); \forall v \circ r \in \{0,1\}^n\)^{\leq l} : \text{HIBE Dec}(param, KeyDer(param, d_v, v, r), v \circ r, \text{HIBE Enc}(param, mpk, v \circ r, m)) = m.

In the security definitions for HIBE schemes we focus again on the selective-identity secure case.
2.6. Encryption

Definition 2.27. SID-IND-CPA HIBE \[BCHK07\]
An l-level hierarchical identity-based encryption scheme \( \text{HIBE} = (\text{Setup}, \text{KeyDer}, \text{HIBE Enc}, \text{HIBE Dec}) \) for identities of length \( n \) is selective ID secure against chosen-plaintext attacks (SID-IND-CPA secure) if the advantage of any \( \text{PPT} \) adversary \( \mathcal{M} \) in the following game is negligible in the security parameter \( k \):

1. Let \( l = l(k), n = n(k) \). \( \mathcal{M}(1^k) \) outputs a target identity \( v^* \in \{0,1\}^{n(k)} \).\(<l \).
2. \((mpk,msk) \leftarrow \text{Setup}(1^k)\). The adversary is given mpk.
3. The adversary \( \mathcal{M} \) may make polynomially-many queries to an oracle \( \text{KeyDer}_{msk} (\cdot) \) for any ID-vector \( v \), as long as \( v \) is not a prefix of the target ID-vector \( v^* \). The adversary is given the secret key \( d_v \) correctly generated for \( v \) using msk and KeyDer (possibly repeatedly).

The adversary is allowed to query this oracle repeatedly using the same identity; if \( \text{KeyDer}_{msk} \) is randomized, then a different secret key may possibly be returned each time.

4. At some point, \( \mathcal{M} \) outputs two messages \( m_0, m_1 : |m_0| = |m_1| \). A bit \( b \in \{0,1\} \) is randomly chosen and the adversary is given the challenge ciphertext \( C^* \leftarrow \text{IBE Enc}(\text{param}, v^*, m_b) \).

5. \( \mathcal{M} \) may continue to query \( \text{KeyDer}_{msk}(\cdot) \) as above. Finally \( \mathcal{M} \) outputs a guess \( b' \) for \( b \).

We say that \( \mathcal{M} \) succeeds if \( b' = b \), and denote the probability event by \( \text{SID--IND--CPA} \text{ HIBE}_{\mathcal{M}, \text{HIBE}} [\text{Succ}] \).

The adversary’s advantage is defined as
\[
\text{Adv}_{\mathcal{M}, \text{HIBE}}^{\text{SID--IND--CPA} \text{ HIBE}}(k) = \left| \text{SID--IND--CPA} \text{ HIBE}_{\mathcal{M}, \text{HIBE}} [\text{Succ}] - \frac{1}{2} \right|
\]

Definition 2.28. SID-IND-CCA2 HIBE \[BCHK07\]
The SID-IND-CPA HIBE definition can be extended to a hierarchical identity-based encryption scheme HIBE that is selective identity secure against chosen ciphertext attacks (SID-IND-CCA2) by allowing the adversary \( \mathcal{M} \) additional access to an oracle \( \text{HIBE Dec}(\cdot) \) such that for any ciphertext \( C \), \( \text{HIBE Dec}(C) \)
returns HIBE Dec($d_{v^*}, C$) where $d_{v^*}$ is the secret key associated with the target identity $v^*$. The adversary has access to this oracle throughout the game but may not submit the challenge ciphertext $C^*$.

2.7 Key Encapsulation

The idea of a key encapsulation mechanism (KEM) was first introduced by Shoup [Sho01, Sho06]. A KEM is a simple form of encryption that transports a random key used for symmetric encryption from a sender to a receiver. The symmetric encryption algorithm is referred to as a data encapsulation mechanism (DEM). This way of encrypting a message is known as the KEM-DEM framework [CS04]. The certificateless KEM that we describe in Chapter 5 can be used with a DEM, another use is in the key agreement scheme that we present in Chapter 6. An adversary against a key encapsulation scheme is said to have an advantage if it can distinguish between the key that is actually encapsulated and a key randomly drawn from the distribution of valid keys. We will now formalize this notion.

2.7.1 Key-Encapsulation Mechanisms

Definition 2.29. Key-Encapsulation Mechanism

A key encapsulation mechanism (KEM) scheme $\text{KEM} = (\text{KEM Setup}, \text{KEM Encap}, \text{KEM Decap})$ consists of three polynomial-time algorithms.

- Via $(pk, sk) \leftarrow \text{KEM Setup}(1^k)$ the randomized setup algorithm produces a key pair for security parameter $k \in \mathbb{N}$.
- Via $(C, K) \leftarrow \text{KEM Encap}(pk)$ a sender creates a random session key $K$ and a corresponding ciphertext $C$.
- Via $K \leftarrow \text{KEM Decap}(sk, C)$ the possessor of secret key $sk$ decapsulates ciphertext $C$ to get back the session key $K$.

Associated to the scheme is a key space $\text{KeySp}$. For consistency, we require that for all $k \in \mathbb{N}$, and all $(C, K) \leftarrow \text{KEM Encap}(pk)$, we have $\Pr[\text{KEM Decap}(sk, C) = K] = 1$, where the probability is taken over the choice of $(pk, sk) \leftarrow \text{KEM Setup}(1^k)$, and the coins of all the algorithms in the expression above.
Definition 2.30. IND-CCA-KEM
The security game for a KEM scheme against an adversary $M$ is associated with the following experiment:

\[
\text{Exp. } \text{Challenge}^\text{KEM-CCA}_{KEM,M}(k) : \\
(pk, sk) \leftarrow \text{KEM Setup}(k) \\
(state) \leftarrow M^\text{KEM Decap}(\cdot)(\text{find}, mpk) \\
K_0^* \leftarrow \text{KeySp}; (C^*, K_1^*) \leftarrow \text{KEM Enc}(pk) \\
\gamma \leftarrow \{0, 1\}; K^* = K^*_\gamma \\
\gamma' \leftarrow M^\text{KEM Decap}(\cdot)(\text{guess}, K^*, C^*, state) \\
\text{Return } \gamma = \gamma'
\]

with the restriction that after $M$ received $C^*$, $M$ is not allowed to query oracle $\text{KEM Decap}()$ for $C^*$. State : Exp is some internal state information of adversary $M$ and can be any (polynomially bounded) string.

The advantage an adversary $M$ has against a KEM scheme is therefore expressed by

\[
\text{Adv}^\text{KEM}_{M}(k) = \left| \Pr \left[ \text{Exp. } \text{Challenge}^\text{KEM-CCA}_{KEM,M}(k) = 1 \right] - 1/2 \right|
\]

2.7.2 ID-based Key Encapsulation Mechanism

Definition 2.31. Identity-Based Key-Encapsulation Mechanism (IB-KEM) [KG09]
An identity-based key encapsulation mechanism (IB-KEM) scheme $\text{IB-KEM} = (\text{IB-KEM Setup}, \text{IB-KEM KeyDer}, \text{IB-KEM Enc}, \text{IB-KEM Dec})$ consists of four PPT algorithms.

- Via $(mpk, msk) \leftarrow \text{IB-KEM KeyGen}(1^k)$ the randomized key-generation algorithm produces master keys for security parameter $k \in \mathbb{N}$.
- Via $sk_{ID} \leftarrow \text{IB-KEM KeyDer}(msk, ID)$ the key generation centre (KGC) computes the secret key for identity $ID$.
- Via $(C, K) \leftarrow \text{IB-KEM Enc}(mpk, ID)$ a sender creates a random session key $K$ and a corresponding ciphertext $C$ with respect to identity $ID$. 
Chapter 2. Theoretical Background

- Via $K \leftarrow \text{IB-KEM Dec}(sk_{ID}, C)$ the possessor of secret key $sk_{ID}$ decapsulates ciphertext $C$ to get back the session key $K$.

Associated to the scheme is a key space $\text{KeySp}$. For consistency, we require that for all $k \in \mathbb{N}$, all identities $ID$, and all $(C, K) \leftarrow \text{IB-KEM Enc}(mpk, ID)$, we have $\Pr[\text{IB-KEM Dec}(\text{IB-KEM KeyDer}(msk, ID), C) = K] = 1$, where the probability is taken over the choice of $(mpk, msk) \leftarrow \text{IB-KEM KeyGen}(1k)$, and the coins of all the algorithms in the expression above. Let $\text{IB-KEM} = (\text{IB-KEM KeyGen}, \text{IB-KEM KeyDer}, \text{IB-KEM Enc}, \text{IB-KEM Dec})$ be an IB-KEM with associated key space $\text{KeySp}$.

**Definition 2.32. IND-CCA-IB-KEM [KG09]**

The security game for an IB-KEM scheme is associated with the following experiment:

\[
\text{Exp. Challenge}^{\text{IB-KEM-CCA}}_{\text{IB-KEM}, M}(k) : \\
(mpk, msk) \xleftarrow{\$} \text{IB-KEM Setup}(k) \\
,ID^*; state \xleftarrow{\$} M(\text{IB-KEM KeyDer}(.), \text{IB-KEM Dec}(\cdot))(\text{find}, mpk) \\
K_0^* \xleftarrow{\$} \text{KeySp}; (C^*, K_1^*) \xleftarrow{\$} \text{IB-KEM Enc}(mpk, ID^*) \\
\gamma \xleftarrow{\$} \{0, 1\}; K^* = K_\gamma^* \\
\gamma' \xleftarrow{\$} M(\text{IB-KEM KeyDer}(.), \text{IB-KEM Dec}(\cdot))(\text{guess}, K^*, C^*, state) \\
\text{Return } \gamma = \gamma'
\]

The oracle $\text{IB-KEM KeyDer}(ID)$ returns $sk_{ID} \leftarrow \text{IB-KEM KeyDer}(msk, ID)$ with the restriction that $M$ is not allowed to query oracle $\text{IB-KEM KeyDer}(\cdot)$ for the target identity $ID^*$. The oracle $\text{IB-KEM Dec}(ID, C)$ first computes $sk_{ID} \leftarrow \text{IB-KEM KeyDer}(msk, ID)$ as above and then returns $K \leftarrow \text{IB-KEM Dec}(sk_{ID}, C)$ with the restriction that in the guess stage $M$ is not allowed to query oracle $\text{IB-KEM Dec}(\cdot, \cdot)$ for the tuple $(ID^*, C^*)$. $state$ is some internal state information of adversary $M$ and can be any (polynomially bounded) string.

The advantage an adversary $M$ has against an IB-KEM scheme is therefore expressed by

\[
\text{Adv}^{\text{IB-KEM}}_M(k) = |\Pr \left[ \text{Exp. Challenge}^{\text{IB-KEM-CCA}}_{\text{IB-KEM}, M}(k) = 1 \right] - 1/2|
\]
2.8 Key Agreement

A key agreement protocol enables two parties to agree on a common secret session key over a public channel, where neither party can predict the outcome of the key before the protocol starts. An adversary against a key agreement scheme has to distinguish a correct session key from a random session key.

First, we illustrate the requirements for key agreement schemes with the Diffie-Hellman key agreement protocol [DH76] on a group of prime order $p$ as an example. Then, we continue to discuss the formal models for key agreement.

We describe the basic Diffie-Hellman key agreement protocol. Given a cyclic group $G$ of prime order $p$ that has a generator $g$, the Diffie-Hellman key agreement protocol between two users ($A$ and $B$) works as follows.

- User $A$ picks a random element $a \leftarrow \mathbb{Z}_p$ and sends $g^a$ to user $B$.
- User $B$ picks a random element $b \leftarrow \mathbb{Z}_p$ and sends $g^b$ to user $A$.
- The key that both agree upon is $g^{ab}$ which can be computed by $A$ from $g^b$ knowing $a$, and by $B$ from $g^a$ knowing $b$.

Assuming that it is hard to compute $g^{ab}$ given $g^a$ and $g^b$, thus implying that the discrete logarithm problem in $G$ is hard, nobody but $A$ and $B$ will know the shared key.

The Diffie-Hellman key exchange protocol has some drawbacks, that were addressed by later authors (for example by Bellare, Canetti, and Krawczyk [BCK98]). Since the Diffie-Hellman key agreement protocol was the first published protocol in public key cryptography, later developments for key management like public key infrastructures were not known. Furthermore, there was no
formal model for key agreement. Therefore, many things that we would expect from a modern key agreement protocol were left unspecified, especially how often keys would be generated and how trust between users of the protocol would be mediated. Therefore, there are two ways to interpret the protocol from today’s point of view. One interpretation, that parties always generate new public and private keys whenever they want to share a key with someone else results in the problem that the parties do not know with whom they share a key. So an adversary can impersonate any party to any user. This is known as “basic impersonation attack”, which is formalized below.

On the other hand, if we assume that User $A$ has a static public key $g^a$ and User $B$ has a static public key $g^b$, and that the users trust each other’s keys, then they would always use the same (static) key $g^{ab}$ for communication. As soon as the private key of either party is compromised, the shared key is exposed and all previous communication can be deciphered. To protect against this threat, the notion of “forward secrecy” has been developed, which is also formalized below.

To protect against these attacks, a combination of certified keys and session keys may be used. User $A$ then has a private key $a \in \mathbb{Z}_p$ and a matching public key $g^a$ and generates a fresh ephemeral key $a' \xleftarrow{\$} \mathbb{Z}_p$ for each session. Then $g^a, g^a'$ are sent over to User $B$ and $B$ responds with $g^b, g^b'$ in a similar manner. The session key is then computed as $g^{ab}, g^{a'b'}$.

However, this protocol is still susceptible to a key compromise impersonation attack: if $A$’s private key $a$ is compromised by an adversary $M$, then $M$ can claim to be any party. Assume that $M$ wants to impersonate $B$ to $A$. $M$ does not know $B$’s private key $b$. But $M$ is free to choose $b'$ during the run of the protocol and can thus compute $g^{a'b'}$. Knowing $A$’s private key $a$, $M$ can also compute $g^{ab}$ from $B$’s public key $g^b$. Thus, $M$ can impersonate any party to $A$. To protect against this threat, the notion of resistance to “key compromise impersonation” (KCI) has been developed, which we formalize below. In many two-party protocols, an additional Diffie-Hellman of the ephemeral keys and the long term keys is used to protect against KCI. That is, $A$ having private key $a$ and ephemeral key $a'$ publishes $g^a$ and $g^a'$ and gets from $B$ the public key $g^b$ and the ephemeral public key $g^{b'}$. Besides computing $g^{ab}$ and $g^{a'b'}$, both parties also compute $g^{ab}$ and $g^{a'b}$. This protects from key compromise impersonation because the adversary who knows $a$ but not $a'$ can then no longer impersonate $B$ to $A$, since he cannot compute $g^{a'b}$.
We use similar techniques in our pairing-based certificateless key agreement protocol in Section 6.1.

The following security properties are commonly required of key establishment protocols in general.

**Resistance to Basic Impersonation Attacks.** An adversary who does not know the private key of party $A$ should not be able to impersonate $A$.

**Resistance to Unknown Key-Share (UKS) attacks.** An adversary $\mathcal{M}$ interferes with two honest parties $A$ and $B$ such that both parties accept the session and compute the same key. However, while $A$ thinks that the key is shared with $B$, $B$ is convinced that the key is shared with $\mathcal{M}$.

**Known Key Security.** Each run of a key agreement protocol between two parties $A$ and $B$ should produce a unique session key. A protocol should not become insecure if the adversary has learned some of the session keys [LMQ+03].

**Weak Perfect Forward Secrecy (wPFS).** A key-exchange protocol provides weak PFS (wPFS) if an attacker $\mathcal{M}$ cannot distinguish from random a key of any session for which the session and its matching session are clean\footnotemark even if $\mathcal{M}$ has learned the private keys of both peers to the session [Kra05a, Definition 22].

**Resistance to Key-Compromise Impersonation (KCI) attacks.** An attacker $\mathcal{M}$ that has learned the private key of party $\hat{A}$ succeeds in a key-compromise impersonation (KCI) attack against $\hat{A}$ if $\mathcal{M}$ is able to distinguish from random the session key of a complete session at $\hat{A}$ for which the session peer is uncorrupted and the session and its matching session (if it exists) are clean [Kra05a, Definition 20].

**Resistance to Disclosure of Ephemeral Secrets.** The protocol should be resistant to the disclosure of ephemeral secrets. The disclosure of an ephemeral secret should not compromise the security of sessions where the ephemeral secret was not used.

ID-based protocols usually require the following property in addition to these properties:

\footnotetext{1}{Roughly speaking, clean is the same as fresh in Definition 2.34.}
KGC forward secrecy The key generation centre (KGC) should be unable to compute the session key knowing all publicly available information.

The first formal treatment of key agreement protocols was given by Bellare and Rogaway [BR93a]. For this work, we focus on the extended Canetti and Krawczyk model for key agreement [LLM07] (eCK model) that was introduced by LaMacchia, Lauter and Mityagin and uses ideas presented in the Canetti and Krawczyk model for key agreement [CK01] (CK model), which is a successor of the Bellare and Rogaway model. We introduce the certificateless version of the eCK model in Chapter 3 and use it then in the key agreement protocols that are presented in Chapter 6. We give a short overview of the CK model, and then discuss the changes that are made in the eCK model.

2.8.1 The Canetti-Krawczyk Model for Key Agreement (CK Model)

We give a slightly shortened version of the CK model used by Boyd et al. [BCGP08] and refer the reader to their paper for details. In the Canetti and Krawczyk model for key agreement [CK01] (CK model), a protocol π is modeled as a collection of n programs running at different parties $P_1, \ldots, P_n$. Each program is an interactive probabilistic polynomial time (PPT) machine. A session is defined as an invocation of π at party $P_i$, and every party may have multiple sessions running concurrently. The communication network is controlled by the adversary $\mathcal{M}$, who is also a PPT machine. The adversary controls the message flow between the parties by activating a party $P_i$, which may be done in two ways:

1. An Establish Session($P_i, P_j, s$) request where $P_j$ is another party with whom the session is to be established, and $s$ is the session ID string which uniquely identifies a session between the participants.

2. By means of an Incoming Message($m$) with a message $m$ and a specified sender $P_j$.

A matching session is defined by having two sessions ($P_i, P_j, s$) and ($P'_i, P'_j, s'$) for that $P_i = P'_i$, $P_j = P'_j$ and $s = s'$. $s$ is a “magic string” known to both parties before the protocol starts. A session $s$ enters the state accepted at party $P_i$ as soon as party $P_i$ finished computing a session key for $s$. 

$\mathcal{M}$ can ask any of the following queries:

- **Corrupt($P_i$):** $\mathcal{M}$ learns the long term key of $P_i$.

- **Session Key($P_i, P_j, s$):** $\mathcal{M}$ learns the session key of an accepted session for $(P_i, P_j, s)$ at party $P_i$.

- **Session State($P_i, P_j, s$):** Returns the internal state information of party $P_i$ with respect to session $s$ with party $P_j$ but does not include the long term key of $P_j$.

- **Session Expiration($P_i, P_j, s$):** Erases the session key for session $s$ with partner $P_j$ from the internal memory of $P_i$.

- **Test Session($P_i, P_j, s$):** The simulator $B$ selects a random bit $b$. If $b = 1$, then the correct session key is returned. Otherwise, a randomly chosen key from the probability distribution of the key space of the protocol is returned. This query may only be asked to a session that is not exposed, where an exposed session is defined as a session that has been asked either
  - a session-state or session-key reveal query to this session or the matching session, or
  - a corrupt query to either partner before the session expires at that partner.

The goal of $\mathcal{M}$ is to guess the bit $b$ in the test session correctly. Thus the security of a key agreement protocol in the CK model is defined by

**Definition 2.33. Session Key Security**

A key establishment protocol $\pi$ is called session key (SK-) secure with perfect forward secrecy (PFS) if the following properties are satisfied for any adversary $\mathcal{M}$.

1. If two uncorrupted parties complete matching sessions then they both output the same key;

2. The probability that $\mathcal{M}$ guesses correctly the bit $b$ is no more than $\frac{1}{2}$ plus a negligible function in the security parameter.
We define the advantage of $\mathcal{M}$ to be

$$\text{Adv}_{\mathcal{M}}(k) = |2 \Pr[b = b'] - 1|.$$  

Hence the second requirement will be met if the advantage of $\mathcal{M}$ is negligible. Canetti and Krawczyk also provide a definition of SK-security without PFS. The only difference with respect to the above definition is that now the adversary is not allowed to expire sessions.

### 2.8.2 The Extended Canetti-Krawczyk Model for Key Agreement (eCK Model)

The Canetti-Krawczyk model has the drawback that the adversary is not allowed to ask session-state queries to the test session. This does not allow the adversary to learn any internal state of the parties participating in the test session. LaMacchia, Lauter and Mityagin [LLM07] enhanced the model by allowing partial corruption of the parties participating in the test session, and consequently named it eCK model. In summary, a protocol proven to be secure in the eCK model guarantees that the adversary cannot learn anything useful about the session key if each party participating at the session has at least one uncompromised secret.

#### Public Key eCK Model

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of parties. The protocol may be run between any two of these parties. For each party there exists a certified public key, and each party has a matching private key.

The adversary is in control of the network over which protocol messages are exchanged. LaMacchia et al. [LLM07] specify a session identifier as $\text{sid} = (\text{role}, \text{ID}, \text{ID}^*, \text{comm}_1, \ldots, \text{comm}_n)$. However, for easier session selection in the proofs of our protocol, we will use the notation $\Pi_{i,j}^t$. Here, $\Pi_{i,j}^t$ represents the $t^{th}$ protocol session which runs at party $i$ with intended partner party $j$.

A session $\Pi_{i,j}^t$ enters an accepted state when it computes a session key $SK_{i,j}^t$. Note that a session may terminate without ever entering into an accepted state.
The information of whether a session has terminated with acceptance or without acceptance is assumed to be public. The session $\Pi_{i,j}$ is assigned a partner ID $pid = (ID_i, ID_j)$. The session ID $sid$ of $\Pi_{i,j}$ at party $i$ is the transcript of the messages exchanged with party $j$ during the session. Two sessions $\Pi_{i,j}^t$ and $\Pi_{j,i}^u$ are considered matching if they have the same $pid$ and $sid$.

The game runs in two phases. During the first phase of the game, the adversary $\mathcal{M}$ is allowed to issue the following queries in any order:

- **Send($\Pi_{i,j}^t, x$):** If the session $\Pi_{i,j}^t$ does not exist, it will be created as initiator at party $i$ if $x = \lambda$, or as a responder at party $j$ otherwise. If the participating parties have not been initiated before, the respective private and public keys are created. Upon receiving the message $x$, the protocol is executed. After party $i$ has sent and received the last set of messages specified by the protocol, it outputs a decision indicating accepting or rejecting the session. In the case of one-round protocols, party $i$ behaves as follows:
  
  $x = \lambda$: Party $i$ generates an ephemeral value and responds with an outgoing message only.
  
  $x \neq \lambda$: If party $i$ is a responder, it generates an ephemeral value for the session and responds with an outgoing message $m$ and a decision indicating acceptance or rejection of the session. If party $i$ as an initiator, it responds with a decision indicating accepting or rejecting the session.

  We require $i \neq j$, i.e. a party will not run a session with itself. We focus on one-round protocols because all protocols discussed in Chapter 6 are one-round protocols.

- **Session key reveal($\Pi_{i,j}^t$):** If the session has not accepted, it returns $\bot$, otherwise it reveals the accepted session key. A session that was asked this query is also referred to as an “opened” session.

- **Reveal private key($i$):** Party $i$ responds with its secret key that corresponds to its public key.

- **Reveal ephemeral key($\Pi_{i,j}^t$):** Party $i$ responds with the ephemeral secret used in session $\Pi_{i,j}^t$. 
These reveal queries can be grouped into two groups: the reveal private key query tries to undermine the security of the public-key-based part of the scheme, and the reveal ephemeral key query tries to undermine the security of one particular session.

We define the state fully corrupt as a session that was asked all two types of reveal queries: the reveal private key and the reveal ephemeral key query. In particular, if there is no matching session, then the ephemeral key of the party that is simulated by the adversary is always revealed.

Once the adversary $\mathcal{M}$ decides that the first phase is over, it starts the second phase by choosing a fresh session $\Pi_{i,j}$ and issuing a $\text{Test}(\Pi_{i,j})$ query, where the fresh session and test query are defined as follows:

**Definition 2.34. Fresh session**

A session $\Pi_{i,j}$ is fresh if (1) $\Pi_{i,j}$ has accepted; (2) $\Pi_{i,j}$ is unopened (not being issued the session key reveal query); (3) the session state at neither party participating in this session is fully corrupted; (4) there is no opened session $\Pi_{j,i}$ which has a matching conversation to $\Pi_{i,j}$.

- $\text{Test}(\Pi_{i,j})$: The input session $\Pi_{i,j}$ must be fresh. A bit $b \in \{0, 1\}$ is randomly chosen. If $b = 0$, the adversary is given the session key, otherwise a session key from the distribution of valid session keys is randomly chosen and returned to the adversary.

After the $\text{Test}(\Pi_{i,j})$ query has been issued, the adversary can continue querying except that the test session $\Pi_{i,j}$ should remain fresh. We emphasize here that partial corruption is allowed as this is a benefit of the eCK security model.

At the end of the game, the adversary outputs a guess $\hat{b}$ for $b$. If $\hat{b} = b$, we say that the adversary wins. The adversary’s advantage in winning the game is defined as

$$\text{Adv}^\mathcal{M}(k) = \left| \Pr[\mathcal{M} \text{ wins}] - \frac{1}{2} \right|$$

**ID-based eCK Model**

The eCK model was extended to cover ID-based key agreement protocols by Huang and Cao [HC09]. We proceed to review the differences to the public key setting by amending the preliminaries for executing the protocol, listing the
queries that the adversary may ask, and refining the resulting revised definition for a fully corrupted session.

The setup algorithm is modified in the following way.

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of parties. The protocol may be run between any two of these parties. For each party there exists an identity public key that can be derived from its identifier. There is a KGC, sometimes also called trusted authority (TA), that issues identity-based private keys to the parties through a secure channel. The rest of the setup process is identical.

The set of allowed adversarial queries is modified as follows:

- **Send($\Pi_{i,j}^t, x$)**: The send query stays the same.

- **Reveal master key()**: The adversary is given access to the master secret key of the KGC.

- **Session key reveal($\Pi_{i,j}^t$)**: If the session has not accepted, it returns $\perp$, otherwise it reveals the accepted session key.

- **Reveal ID-based secret($i$)**: Party $i$ responds with its ID-based private key, e.g. $sH_1(ID_i)$.

- **Reveal ephemeral key($\Pi_{i,j}^t$)**: Party $i$ responds with the ephemeral secret used in session $\Pi_{i,j}^t$.

Again, we can group the key reveal queries into two types: the **reveal master key** and **reveal ID-based secret** queries try to undermine the security of the ID-based part of the scheme and thus the long-term keys, and the **reveal ephemeral key** query tries to undermine the security of one particular session.

We define the state **fully corrupt** as a session that was asked both types of reveal queries: the **reveal master key** or **reveal ID-based secret**, and the **reveal ephemeral key** query. In particular, if there is no matching session, then the ephemeral key of the party that is simulated by the adversary is always revealed.

The rest of the security definition is exactly as for the public-key-based part.

### 2.9 Conclusion

This chapter establishes the theoretical background that is used in the following chapters of the thesis. We give an overview of the mathematical primitives that
we will use in our constructions. We discuss the problems that the research community considers not to be solvable in polynomial time and that we will later use as the basis for the protocols that are developed in Chapter 4, 5 and 6. Of particular interest in this section are the new trapdoor test techniques that we develop and prove. We will use these new results in the security analysis of the key agreement protocol presented in Section 6.1. Additionally, we believe that they may be useful in proving other protocols based on bilinear pairings secure. In Section 2.3 we define some of the cryptographic primitives such as hash functions and randomness extraction functions that we use as building blocks in our protocols. The last three sections of this chapter review the standard security definitions for the cryptographic protocols that we will discuss in Chapter 4, 5 and 6, both in the public key setting and in the ID-based setting. This is a good basis to develop the security definitions for these protocols in the certificateless setting in the following chapter.
Chapter 3

Security Models for Certificateless Protocols

This chapter discusses the security models for certificateless encryption, key encapsulation, and key agreement schemes, that we use in the following chapters. We give an introduction into the security assumptions associated with certificateless cryptography and develop new definitions for certificateless key encapsulation and certificateless key agreement. Our definitions expand the security definitions given in the previous chapter to the certificateless case. New security definitions are necessary to reflect the enhanced abilities of an adversary against certificateless protocols. The definitions given in this chapter are the basis for the security analysis of the protocols that we present in the following chapters. We also explain the general proof strategy that we will use in the following chapters.

3.1 Certificateless Cryptography

Certificateless cryptography was introduced by Al-Riyami and Paterson [ARP03]. As explained in Chapter 1, certificateless cryptography tries to merge the best features from both standard public key cryptography and identity-based cryptography. The goal is to combine the ease of key management that is offered by ID-based cryptography with the privacy for the user that is guaranteed by traditional public key cryptography. Considering the features implied, it sounds as if certificateless schemes could always be achieved by simply combining a public-

key-based scheme with an ID-based scheme. The public-key-based scheme would then guarantee the user privacy but would not need certificates, since the ID-based scheme which is also used would guarantee that only the intended recipient is able to extract the information. Several authors showed for certificateless encryption [YL04a, LQ06, HW07a], certificateless signatures [YL04b], and certificateless key encapsulation mechanisms [BFMLS08, HW07b] that this approach works. Contrary to what would be expected, we show that a one-round certificateless key agreement protocol cannot be securely constructed by a natural combination of an ID-based key agreement protocol with a public-key-based key agreement protocol. This means that we cannot readily obtain certificateless key agreement protocols from existing ID-based and public-key-based protocols, but we have to construct and prove certificateless key agreement protocols specifically. This is what we will do in Chapter 6.

3.1.1 General Attacks Against Certificateless Cryptography

In a certificateless scheme, each user publishes a public key in a publicly writeable directory. Furthermore, all users trust one key generation centre and agree to use its parameters. Because the directory is publicly writeable, there exists the problem of “denial of decryption” attacks, first described by Liu, Au and Susilo [LAS07]. Denial of decryption means that an adversary consistently replaces a user’s public key with a random key, so that the user cannot decrypt. This scenario seems to be realistic only for misbehaving key generation centres and governments that do not want their citizens to communicate privately, because all other attackers do not gain much benefit from the attack, as they do not know the master private key matching the system parameters and can therefore not gain any information about the messages that are exchanged with the replaced certificateless public key. Furthermore, since the users will start to overwrite the changes with the correct key again, these modifications will not last long. There are encryption schemes that resist the denial of decryption attack [WHY08], however these schemes are computationally much more expensive than certificateless schemes that are not resistant to denial of decryption attacks. Furthermore, standard workflow supported in identity-based encryption like encrypting into the future and encrypting to identity strings that match the recipient, but for which the recipient did not obtain a valid private key yet are
not supported in these schemes. For these reasons, we do not consider schemes that resist denial of decryption attacks but focus on schemes that are secure in the standard model.

Furthermore, it is possible for the trusted authority to generate the system parameters in such a way that it can decrypt certificateless ciphertexts of either a specific user or of all users. Encryption schemes that are susceptible to this attack seem to be unable to achieve the security guarantees that were originally set forth in certificateless encryption, since certificateless encryption aims to disable the key escrow mechanisms that are inherent in identity-based encryption (for the very reason that the user does not want the authority to be able to read all of his/her messages). Au et al. [AMC+07] described this attack first and named it malicious KGC attack. It is applicable to various certificateless schemes which are either in the random oracle model or in the standard model (for example [ARP03, DLP08]). We show that the encryption schemes that we present in Chapter 4 are not affected by this attack.

3.2 Types of Certificateless Adversaries

In certificateless cryptography it is common to distinguish between two types of adversaries:

**Type I:** A Type I adversary represents an outsider adversary that does not have access to the secret master key of the key generation centre (KGC).

**Type II:** A Type II adversary represents an insider adversary that has access to the master secret key (e.g. a malicious KGC).

Both types of adversaries have the power to replace certificateless public keys, because the keys are not certified by any authority. However, an insider adversary must not replace the public key for the challenge identity before the challenge ciphertext has been constructed. The power to replace public keys results in the classification of certificateless schemes in mainly two types (see also Dent’s [Den08] discussion). The classification is based on the type of decryption oracle access that the adversary has:

**Strong security:** The adversary has access to a strong decryption oracle. This means that the adversary can obtain decryption of ciphertexts even if the
adversary chooses the certificateless public key used for encryption. Thus the decryption oracle can decrypt a ciphertext $C \in C$ even if the adversary replaced the certificateless public key that was used to generate the ciphertext and does not disclose the matching private key to the decryption oracle.

**Weak security:** The adversary has access to a *weak secret value decryption oracle* (Weak SV Decrypt oracle). The oracle can decrypt ciphertexts only if it is given all private keys necessary for decryption. If the adversary replaced a public key, then decryption is only possible if the adversary submits the private key matching the public key along with the decryption request.

The notion of strong security seems to be unrealistic, since one would expect that a decryption of a ciphertext that was encrypted under a random public key should not be possible. However, one easily gets the impression that strong security is the security notion to aim for in all certificateless schemes, since it would guarantee perfect simulatability. However, one of the most prevalent drawbacks of currently known strongly secure certificateless encryption schemes is that in these schemes it is possible for the key generation centre to attack users, because all currently known strongly secure encryption schemes are vulnerable to a malicious KGC attack discovered by Au et al. [AMC\textsuperscript{+}07]. The key generation centre does not generate the system parameters honestly in a malicious KGC attack, but in a way that it is able to compromise the privacy of one or even all users. This attack works even if the authority does not replace a user’s public key in the public directory. The exact attack and how many users are compromised depends on the specific certificateless scheme. However, a user of the certificateless encryption scheme cannot tell that this attack has taken place, because the attacks works even if the authority does not replace a users public key in the public directory. In the only published strongly secure scheme in the standard model by Dent et al. [DLP08], the trusted authority can even decrypt all ciphertexts if it is not honest. This almost puts as much trust in the trusted authority as in an authority in a “normal” ID-based scheme. The attack has been observed for certificateless encryption schemes only, and it is not known if it would also work for certificateless key encapsulation mechanisms and certificateless key agreement schemes in the strong security model. If we had to guess, we would say that the attack will probably also work for certificateless
KEMs and certificateless key agreement in the CK model without forward secrecy. However, the attack seems not to be mountable against certificateless key agreement schemes with forward secrecy (which are all key agreement schemes in the eCK model), because forward secrecy guarantees that the session key is indistinguishable from a random key even if the long term secrets are known. Intuitively it seems natural that an encryption scheme in the strong security model is vulnerable to the malicious KGC attack, especially in the standard model. During the simulation, the simulator must have some “backdoor” that can be exploited to decrypt ciphertexts that are encrypted under a public key chosen by the adversary. If there was no such “backdoor”, then the decryption of ciphertexts under a random public key should always fail, due to the security of the public key encryption scheme. In Au’s attack, the KGC simply exploits this backdoor for an attack. Dent et al. [DLP08] argue that the strong model for certificateless encryption is of theoretical interest because it also covers the weak security model. We think that due to the attack and its implications discussed above, the strong security model may even introduce additional vulnerabilities, especially if it is used in the standard model. We believe that the constructions that we prove in Chapter 4 are more secure than the construction proposed by Dent et al. [DLP08] for this very reason.

The generic construction for weakly secure certificateless encryption in the standard model published by Huang and Wong [HW07a] does not suffer from this vulnerability. In his survey on certificateless encryption schemes, Dent [Den08] remarks that the weak security model “seems to most realistically reflect the potential abilities of an attacker.” In the certificateless encryption schemes that we describe in Chapter 4 we also focus on weakly secure schemes. This has the benefit that our schemes are also not vulnerable to a malicious KGC attack.

To summarize our findings on the strong and weak security model for certificateless protocols, we can say that for certificateless encryption the weak security model seems to offer the best security guarantees, especially in the standard model. We think that random oracle model protocols may be more secure in reality, since the random oracle may be used to allow the simulation of the decryption oracle even for random public keys in the strong security model for certificateless cryptography. However, the functionality of a random oracle is not available in reality, so there may be strongly secure certificateless encryption schemes in the random oracle model that are secure in practice (although we are currently not
aware of any such protocols). Since there are no certificateless key encapsulation mechanisms that are proven in the strong security model, we cannot judge the situation here. However, since key encapsulation is usually seen as a more efficient form of encryption, we suppose that similar attacks may exist. We prove the KEM that we present in Chapter 5 secure in the weak certificateless setting. For certificateless key agreement, we consider the strong security model not necessarily harmful if the scheme has forward secrecy and present a key agreement scheme that has the strong security property in Section 6.1. We also present a key agreement scheme in the standard model with the weak security property in Section 6.2.

3.3 Security Model for Certificateless Encryption

In this section we discuss the security model for a certificateless encryption scheme and introduce the security model for hierarchical certificateless encryption. We discuss the differences from the security notions for encryption schemes that we presented in Section 2.6. Dent [Den08] discusses the security models of various certificateless encryption schemes. In the following, we focus on the description of certificateless encryption that is most widely used in the literature, and that allows workflows that match the workflows inherent in identity-based encryption (cf. Section 3.1.1).

3.3.1 Certificateless Encryption Scheme

Definition 3.1. Certificateless Encryption Scheme

A certificateless encryption scheme for identities ID ∈ {0, 1}^n is a tuple of probabilistic polynomial time (PPT) algorithms (CL-IBE Setup, CL-IBE KeyDer, CLE User KeyGen, CLE Enc, CLE Dec) such that:

CL-IBE Setup: On input 1^k where k ∈ N is a security parameter, it generates a master public/private key pair (mpk, msk) and system parameters param.

CL-IBE KeyDer: (Dent [Den08] calls this algorithm Extract Partial Private Key) On input msk, param and a user identity ID ∈ {0, 1}^*, it derives a
3.3. Security Model for Certificateless Encryption

user partial key / ID-based private key $d_{ID}$ from the master secret key $msk$ and the system parameters $param$.

**CLE User KeyGen:** On input $param$, it generates a user public/private key pair $(upk_{ID}, usk_{ID})$.

**CLE Enc:** takes as input $(param, mpk, upk_{ID}, ID, m)$ and outputs an encryption of the message $m$ as ciphertext $C$.

**CLE Dec:** takes as input $(param, (d_{ID}, usk_{ID}), ID, C)$ and decrypts $C$ to get back $m$, or outputs the special symbol $\perp$ indicating an invalid encryption.

We require that $\forall (mpk, msk) \leftarrow CL-IBE \text{Setup}(1^k)$ and $\forall ID \in \{0, 1\}^n$ and $\forall (upk_{ID}, usk_{ID}) \leftarrow CLE \text{User KeyGen}$ we have that $CLE \text{Dec}(param, CL-IBE \text{KeyDer}(param, msk, ID), ID, CLE \text{Enc}(param, mpk, upk_{ID}, ID, m)) = m$.

As we see, the main difference for certificateless encryption schemes from identity-based encryption schemes is the **CLE User KeyGen** algorithm. This algorithm is introduced to allow the user to generate a key pair and then publish the public key in an online directory. We consider two security definitions for certificateless encryption schemes. We consider IND-CPA security and IND-CCA2 security, as these are the standard definitions for encryption schemes as discussed in Section 2.6.

### 3.3.2 The Security Game for CLE

To model the security guarantees of a certificateless scheme correctly, we introduce the following model that is based on the requirements by Dent [Den08]. The adversary $\mathcal{M}$ has access to the following oracles:

**Reveal master key:** The adversary is given access to the master secret key.

**Reveal ID-based key(ID):** The adversary extracts the ID-based private key of party $ID$.

**Get user public key(ID):** The adversary obtains the certificateless public key for $ID$. If the certificateless key for the identity has not yet been generated, it is generated on the fly with **User KeyGen**.
Replace public key \((\text{ID}, \text{upk}_\text{ID})\): Party \text{ID}'s certificateless public key is replaced with \(\text{upk}_\text{ID}\) chosen by the adversary. All communication (encryption) for Party \text{ID} will use the new public key. In a production environment, the replace public key functionality can be used by a user to replace his public key on a regular basis to protect against accidental compromise of his private key.

Reveal user secret key \((\text{ID})\): The adversary extracts the secret value \(\text{usk}_\text{ID}\) that corresponds to the certificateless public key \(\text{upk}_\text{ID}\) for party \text{ID}. If the adversary issued a replace public key query for \text{ID} before, the special symbol \(\bot\) is returned.

Decrypt \((\text{ID}, \text{C})\): The adversary learns the decryption of \(\text{C}\) under \text{ID} or \(\bot\) if \(\text{C}\) is invalid. Since the encryption schemes presented in Chapter 4 are in the weak certificateless model, \(\bot\) is also returned if the adversary replaced the public key of \text{ID}. In the strong certificateless model, a decryption would be possible.

Decrypt \((\text{ID}, \text{C}, \text{usk}_\text{ID})\): The adversary learns the decryption of \(\text{C}\) under \text{ID} using the user secret key \(\text{usk}_\text{ID}\). The special symbol \(\bot\) will be returned if \(\text{C}\) is invalid or \(\text{usk}_\text{ID}\) does not match \(\text{upk}_\text{ID}\). This query is only necessary in the weak certificateless model.

Get challenge encryption \((\text{ID}^*, \text{m}_0, \text{m}_1)\): The adversary supplies two equal length messages \(\text{m}_0, \text{m}_1\) and requests a challenge encryption under the key for \(\text{ID}^*\). This marks the transition from \text{Oracles}_1 \text{ to Oracles}_2 \text{ in Experiment 3.1.} \text{ The simulator returns a challenge ciphertext as described in Experiment 3.1.}

To correctly model the security guarantees for adversaries against certificateless encryption schemes, the adversary has access to more oracles than in an ID-based scheme. Most notably the replace public key and the reveal user secret key queries correspond to the additional abilities of the adversary. Using the replace public key query, the adversary is able to replace the user generated public key. This models the fact that public keys in certificateless cryptography are not specifically protected, but usually reside in a publicly writeable directory. The reveal user secret key and reveal ID-based secret key queries model the ability of the adversary to corrupt a party only partially. Even if a party is partially
3.3. Security Model for Certificateless Encryption

corrupted, the security guarantees of a certificateless encryption scheme should still hold. The definition given below describes the security game for selective ID (SID) secure IND-CPA and IND-CCA2 security. IND-CPA security models security for an adversary that has no access to a decryption oracle, IND-CCA2 security focuses on an adversary that has access to a decryption oracle and can query the oracle adaptively during the game. We focus on SID security because the constructions that lead from IND-CPA security to IND-CCA2 security discussed in Chapter 4 focus on SID security. Of course, these constructions are also applicable to schemes that are fully secure, since any fully secure scheme can also deal with an adversary against a SID scheme (but not vice versa). We now proceed to give the definition.

Definition 3.2. SID-IND-CPA-CLE / SID-IND-CCA2-CLE

The security game for a selective identity IND-CPA secure certificateless encryption scheme (SID-IND-CPA-CLE) or a selective identity IND-CCA2 secure certificateless encryption scheme (SID-IND-CCA2-CLE) is associated with the following experiment, where CXX is either CPA or CCA2. These experiments differ only in the type of oracle access that the adversary has. The oracle access is specified below.

\[
\text{Exp. Challenge}^{\text{sid-ind-cxx-cle}(k)}: \\
\text{ID}^* \xleftarrow{\$} \mathcal{M}(n) \\
(\text{mpk}, \text{msk}) \xleftarrow{\$} \text{CLE IBE Setup}(k, \text{ID}^*) \\
(\text{upk}_{\text{ID}^*}, \text{usk}_{\text{ID}^*}) \xleftarrow{\$} \text{CLE User Keygen}(\text{param}) \\
(m_0, m_1, \text{state}) \xleftarrow{\$} \mathcal{M}^{\text{Oracles}_1}(\text{param}, \text{find}, \text{mpk}, \text{upk}_{\text{ID}^*}) \hspace{1cm} (3.1) \\
b \xleftarrow{\$} \{0, 1\} \\
C^* \xleftarrow{\$} \text{CLE Enc}(\text{param}, \text{mpk}, \text{upk}_{\text{ID}^*}, m_b, \text{ID}^*) \\
b' \xleftarrow{\$} \mathcal{M}^{\text{Oracles}_2}(\text{guess}, C^*, \text{state}) \\
\text{Return } b = b'
\]

The advantage an adversary \(\mathcal{M}\) has against a CLE scheme is therefore expressed by

\[
\text{Adv}^{\text{sid-ind-cxx-cle}(k)}_{\mathcal{M}}(k) = |\Pr[\text{Experiment Challenge}^{\text{sid-ind-cxx-cle}(k)}_{\text{CLE } \mathcal{M}}] - 1/2|
\]
For an outsider/Type I SID-IND-CPA-CLE adversary $\mathcal{M}$, Oracles$_1$ and Oracles$_2$ mean access to all oracles listed in Section 3.3.2 with the following limitations:

1. No reveal master key queries.
2. No decryption queries for ID$^*$.
3. Not both (reveal user secret key OR replace public key) AND reveal ID-based key oracles may be asked for ID$^*$.

For an insider/Type II adversary $\mathcal{M}$, Oracles$_1$ and Oracles$_2$ are subject to the following limitations:

1. Oracles$_1$ and Oracles$_2$ now includes reveal master key as allowed query,
2. no decryption queries for ID$^*$,
3. reveal user secret key must never be asked for ID$^*$,
4. Oracles$_1$ must not include replace public key for ID$^*$.

A Type I/II SID-IND-CCA2-CLE adversary has access to all the oracles that a SID-IND-CPA-CLE adversary has, but may also ask decryption queries for ID$^*$ with the exception that $C^*$ must never be submitted to a decryption oracle for ID$^*$.

### 3.3.3 Hierarchical Certificateless Encryption (HCLE)

Hierarchical identity-based encryption schemes were first discussed in Definition 2.26. Hierarchical ID-based encryption is useful to reduce the load that the key generation centre experiences, as key delegation to sub-authorities allows the sub-authorities to generate keys for their users. Then the root authority only has to generate keys for the sub-authorities occasionally. However, load balancing is not the only benefit that hierarchical ID-based encryption offers. Canetti, Halevi and Katz [CHK04] showed how to convert any selective-identity IND-CPA secure ID-based scheme into a IND-CCA2 secure public key scheme. Later, Boneh and Katz [BK05] described a more efficient technique. If these techniques are applied to an $l$-level selective identity IND-CPA secure hierarchical identity-based encryption scheme, then the resulting $l-1$ level hierarchical identity-based encryption scheme will be selective identity IND-CCA2 secure. In Chapter 4, we
convert both techniques to the certificateless case. This allows us to give generic
constructions for highly efficient certificateless encryption in the standard model.
However, we need to define hierarchical certificateless encryption schemes first.

Definition 3.3. Hierarchical Certificateless Encryption Scheme (HCLE)
A hierarchical certificateless encryption scheme (HCLE) for identities of length \( n \)
with depth \( l \) (where \( l \) and \( n \) are polynomially bounded) is a tuple of PPT algorithms
(\( \text{CL-HIBE Setup, CL-HIBE KeyDer, CLE User KeyGen, HCLE Enc, HCLE Dec} \)) such
that:

- **CL-HIBE Setup**: On input \( 1^k \) where \( k \in \mathbb{N} \) is a security parameter, it generates a
  master public/private key pair \( (\text{mpk,msk}) \) and system parameters \( \text{param} \).

- **CL-HIBE KeyDer**: On input \( d_v, \text{param} \) (where \( v \) is an identity vector in \( (\{0,1\}^n)^l \))
  and a user identity \( \text{ID} \in \{0,1\}^n \), it derives a partial / ID-based private
  key \( d_v \circ \text{ID} \) for the identity \( v \circ \text{ID} \) that is hierarchically below \( v \) from the secret
  key \( d_v \) and the system parameters \( \text{param} \). If \( |v| = 0 \), then \( d_v = \text{msk} \).

- **HCLE User KeyGen**: On input \( \text{param} \), it generates a user public/private key pair
  \( (\text{upk}_v, \text{usk}_v) \). In this work we focus only on certificateless schemes that do
  not have an ID-based component in the user key and do not require the
  master public key for user key generation. Thus neither ID nor mpk can
  be an input to this algorithm. Note that in these cases, \( \text{upk} \) is independent
  from \( \text{mpk} \).

- **HCLE Enc**: takes as input \( (\text{param,mpk,upk}_v,v,m) \) and outputs an encryption of
  the message \( m \) as ciphertext \( C \).

- **HCLE Dec**: takes as input \( (\text{param,}(d_v,\text{usk}_v),v,C) \) and decrypts \( C \) to get back \( m \),
  or outputs the special symbol \( \perp \) indicating an invalid encryption.

We require that \( \forall (\text{mpk,msk}) \leftarrow \text{CL-HIBE Setup}(1^k) \) and \( \forall v \in (\{0,1\}^n)^l \) and
\( \forall (\text{upk}_v, \text{usk}_v) \leftarrow \text{HCLE User KeyGen} \) we have that \( \text{HCLE Dec}(\text{param,}
\text{CL-HIBE KeyDer(\text{param, msk, ID}), ID, HCLE Enc(\text{param, mpk, upk}_v, ID, m)})
= m \).

A HCLE scheme differs from a HIBE scheme by the fact that users generate
their own key pair using the HCLE User KeyGen algorithm. To model the security
guarantees of a HCLE scheme correctly, we have to modify the security game
accordingly. We give the definition for both IND-CPA and IND-CCA selective identity hierarchical certificateless encryption schemes below.

**Definition 3.4. SID-IND-CPA-HCLE / SID-IND-CCA2-HCLE**

The security game for a selective identity IND-CPA secure hierarchical certificateless encryption scheme (SID-IND-CPA-HCLE) or a selective identity IND-CCA2 secure hierarchical certificateless encryption scheme (SID-IND-CCA2-HCLE) is associated with the following experiment, where CXX is either CPA or CCA2. These experiments differ only in the type of oracle access that the adversary has. The oracle access is specified below.

**Exp. Challenge** \[\text{sid-\text{ind-cpa-hcle}}(k) : \]

\[v^* \leftarrow \mathcal{M}(n)\]
\[(mpk, msk) \leftarrow \text{CLE HIBE Setup}(k, v^*)\]
\[(upk_{v^*}, usk_{v^*}) \leftarrow \text{HCLE User Keygen}(\text{param})\]
\[(m_0, m_1, \text{state}) \leftarrow \mathcal{M}_{\text{Oracles}_1}(\text{find, mpk, upk}_{v^*})\]
\[b \leftarrow \{0, 1\}\]
\[C^* \leftarrow \text{HCLE Enc}(\text{param, mpk, upk}_{v^*}, m_b, v^*)\]
\[b' \leftarrow \mathcal{M}_{\text{Oracles}_2}(\text{guess, C^*, state})\]

Return \(b == b'\)

The advantage an adversary \(\mathcal{M}\) has against a HCLE scheme is therefore expressed by

\[\text{Adv}^{\text{sid-ind-cpa-hcle}}(\mathcal{M}, k) = \left| \Pr \left[ \text{Experiment Challenge}_{\text{HCLE}}(\mathcal{M}, k) \right] - 1/2 \right|\]

For an outsider/Type I SID-IND-CPA-HCLE adversary \(\mathcal{M}\), \(\text{Oracles}_1\) and \(\text{Oracles}_2\) mean access to all oracles listed in Section 3.3.2 with the following limitations:

1. No reveal master key queries.
2. No decryption queries for \(v^*\).
3. Not both (reveal user secret key OR replace public key) AND reveal ID-based key oracles may be asked for \(v^*\).
For an insider/Type II adversary $\mathcal{M}$, $\text{Oracles}_1$ and $\text{Oracles}_2$ are subject to the following limitations:

1. $\text{Oracles}_1$ and $\text{Oracles}_2$ now includes reveal master key as allowed query,
2. No decryption queries for $v^*$.
3. reveal user secret key must never be asked for $v^*$,
4. $\text{Oracles}_1$ must not include replace public key for $v^*$.

A Type I/II SID-IND-CCA2-HCLE adversary has access to all the oracles that a SID-IND-CPA-CLE adversary has, but may also ask decryption queries for ID* with the exception that $C^*$ must never be submitted to a decryption oracle for ID*.

### 3.4 Security Model for Certificateless Key Encapsulation

The most prevalent use of key encapsulation mechanisms is to securely transport a randomly generated key that is used with a symmetric cipher to the intended recipient. This is known as the KEM/DEM framework (key encapsulation mechanism / data encapsulation mechanism). Because the messages transported are completely random, the security assumptions can be lowered. That means that in the security experiment, the adversary does not get to choose the message that is encapsulated but is given a key encapsulation generated by the simulator and either the correct key or a randomly drawn key from the distribution of valid keys. The adversary then has to decide whether the key is correct or random.

All published certificateless (CL) KEM schemes [HW07b, BFMLS08] focus on the weak security model. We will use this model for our work as well. We continue to define the algorithms that a certificateless key encapsulation mechanism uses and then continue to explain the security model for certificateless key encapsulation.

#### 3.4.1 Certificateless Key Encapsulation Mechanism

We use the definition by Huang and Wong [HW07b] for a certificateless key-encapsulation mechanism (CL-KEM). If the adversary replaced a user public key
with a key that is not based on the trusted authority’s public key, then the security of the KEM would be broken. Therefore, we add the CL-KEM Key Verification algorithm to their definition. This algorithm later allows the storage of verified keys and their use during encryption, and makes encapsulation thus more efficient. We need the key verification algorithm to make sure that both the user generated key and the ID-based key are needed for decryption. We assume that key verification needs to be done less often than encryption, because in a usual workflow one will encrypt multiple messages to the same key. Since verification can be computationally expensive, this will make schemes more efficient. We define a certificateless KEM to consist of the following algorithms:

**CL-KEM IBE Setup**: On input $1^k$ where $k \in \mathbb{N}$ is a security parameter, it generates a master public/private key pair $(mpk, msk)$.

**CL-KEM IBE KeyDerivation**: On input $msk$ and a user identity $ID \in \{0, 1\}^n$, it generates a user partial key / ID-based private key $sk_ID$.

**CL-KEM User KeyGen**: On input $mpk$ and a user identity $ID$, it generates a user public/private key pair $(upk, usk)$.

**CL-KEM Key Verification**: On input $mpk$ and $upk$, it generates an encryption key $enc_k$ that is used for all following encapsulations and is a combination of $mpk$ and $upk$. This algorithm needs to run only if the master public key or the user public key change (which should happen less frequently than actual encapsulations take place).

**CL-KEM Encapsulation**: Takes as input $(mpk, enc_k, ID)$ (where $enc_k$ is the output of the CL-KEM Key Verification algorithm) and outputs an encapsulation key pair $(K, C) \in \mathcal{K} \times \mathcal{E}$ where $C$ is called the encapsulation of the key $K$ and $\mathcal{K}$ and $\mathcal{E}$ are the key space and the encapsulation space respectively.

**CL-KEM Decapsulation**: Takes as input $((sk_ID, usk), ID, C)$ and decapsulates $C$ to get back a key $K$, or outputs the special symbol ⊥ indicating invalid encapsulation.

Compared to the definition for an ID-based KEM (see Definition 2.31), we added the CL-KEM Key Verification algorithm. Additionally, we made a modification to the CL-KEM Decapsulation algorithm so that it now requires two private keys as input.
3.4.2 The Security Game for CL-KEM

In certificateless KEM’s (CL-KEM’s), there are (as always in certificateless cryptography) two private keys for each party. One key is the ID-based private key that is derived from the master secret key by the trusted authority, and the other private key is generated by the party on its own. The security guarantees of the KEM should hold if either key (but not both) was compromised.

To model the security guarantees of a certificateless scheme correctly, we introduce the following model that merges the requirements by Dent [Den08] and Huang & Wong [HW07b]. The adversary $\mathcal{M}$ has access to the following oracles:

**Reveal master key:** The adversary is given access to the master secret key.

**Reveal ID-based key**($\text{id}$): The adversary extracts the ID-based private key of party $\text{id}$.

**Get user public key**($\text{id}$): The adversary obtains the certificateless public key for $\text{id}$. If the certificateless key for the identity has not yet been generated, it is generated with the **CL-KEM User KeyGen** algorithm.

**Replace public key**($\text{id}$,$\text{pub}$): Party $\text{id}$’s certificateless public key is replaced with $\text{pub}$ chosen by the adversary. All communication (encryption, encapsulation) for Party $\text{id}$ will use the new public key.

**Reveal secret value**($\text{id}$): The adversary extracts the secret value that corresponds to the certificateless public key for party $\text{id}$. If the adversary issued a replace public key query for $\text{id}$ before, $\perp$ is returned.

**Decapsulate**($\text{id}$,$C$): The adversary learns the decapsulation of $C$ under $\text{id}$ or $\perp$ if $C$ is invalid or if the adversary replaced the public key of $\text{id}$.

**Decapsulate**($\text{id}$,$C$, $x$): The adversary learns the decapsulation of $C$ under $\text{id}$ using the user generated secret key $x$ which is supplied by the adversary. The special symbol $\perp$ will be returned if $C$ is invalid or if $x$ does not match the certificateless public key.

**Get challenge key encapsulation**($\text{id}^*$): The adversary requests a challenge key encapsulation and thus marks the transition from Oracles$_1$ to Oracles$_2$. 


in Experiment 3.3. The simulator returns a challenge key encapsulation as described in Experiment 3.3.

Compared to the ID-based KEM discussed in Definition 2.31, the adversary has now more oracles that he may query. These oracles correspond to the additional secrets that are defined in a certificateless KEM scheme.

We proceed to define the security experiment for a CL-KEM scheme. Since oracle access depends on the type of certificateless adversary that is interacting with the KEM scheme, we first give a broad definition, and then define oracle access for the respective adversaries.

**Experiment Challenge**\(_{\text{CL-KEM, } \mathcal{M}}^{\text{cl-kem-cca}}(k)\):

\[
\begin{align*}
(mpk, msk) & \xleftarrow{\$} \text{CL-KEM IBE Setup}(k) \\
(ID^*, \text{state}) & \xleftarrow{\$} \mathcal{M}^{\text{Oracles}_1}(\text{find, } mpk) \\
K_0^* & \xleftarrow{\$} \mathcal{K}; (C^*, K_1^*) \xleftarrow{\$} \text{CL-KEM Enc}(mpk, ID^*) \tag{3.3} \\
\gamma & \xleftarrow{\$} \{0, 1\}; K^* = K^*_\gamma \\
\gamma' & \xleftarrow{\$} \mathcal{M}^{\text{Oracles}_2}(\text{guess, } K^*, C^*, \text{state}) \\
\text{Return } \gamma &= \gamma'
\end{align*}
\]

The advantage an adversary \(\mathcal{M}\) has against a CL-KEM scheme is therefore expressed by

\[
\text{Adv}_{\mathcal{M}}^{\text{CL-KEM}}(k) = |\Pr[\text{Experiment Challenge}^{\text{cl-kem-cca}}_{\text{CL-KEM, } \mathcal{M}}(k) = 1] - 1/2|
\]

For a *Type I* adversary \(\mathcal{M}\), Oracles\(_1\) and Oracles\(_2\) mean access to all oracles listed above with the following limitations:

1. No *reveal master key* queries.
2. \(C^*\) must not be submitted to a *decapsulate* oracle under \(ID^*\).
3. Not both (*reveal secret value* OR *replace public key*) AND *reveal ID-based key* oracles may be asked for \(ID^*\).

For a *Type II* adversary \(\mathcal{M}\), Oracles\(_1\) and Oracles\(_2\) are subject to the following limitations:

1. Oracles\(_1\) and Oracles\(_2\) now includes *reveal master key* as allowed query,
2. $C^*$ must not be submitted to a decapsulate oracle under ID$^*$.

3. reveal secret value must never be asked for ID$^*$.

4. Oracles$_i$ must not include replace public key for ID$^*$.

### 3.5 Security Model for Certificateless Key Agreement Schemes

In this section, we review the requirements for certificateless key agreement schemes. In addition to the requirements for key agreement protocols that are listed in Section 2.8, certificateless protocols should have the following property, which was first discussed by Mandt and Tan [MT06]. They call the property “Resistance to known session-specific temporary information”, but they provide only an informal definition. It is not possible to provide this property in an ID-based key agreement scheme since a KGC who knows the ephemeral secrets has all inputs to the session key.

Resistence to leakage of ephemeral secrets to the KGC. If a malicious KGC learns the ephemeral secrets of any session, the KGC should not be able to compute the session key.

Now that we reviewed the requirements for certificateless key agreement protocols, we continue to develop models that try to capture these notions.

#### 3.5.1 Formal Definition of the Security Model

Formal security models for key agreement try to capture the security properties that are required of key agreement protocols, which were discussed in Section 2.8. We focus on the eCK model in our work. Based on the eCK model described in Section 2.8.2, we present the first certificateless protocol that was published with a proof of security in Section 6.1. We define the corresponding enhanced eCK model ($e^2$CK model) below.

In Section 6.2, we describe the first construction of a certificateless key agreement protocol in the standard model, which uses the KEM-KEM techniques described by Boyd et al. [BCGP08]. The protocol by Boyd et al. [BCGP08] is proven in the Canetti-Krawczyk model. We show how to prove it secure in the $e^2$CK model.
The protocols that we consider in Chapter 6 are both one-round key agreement protocols. This means that the protocol execution terminates after each party received a message from the respective other party. In a two party protocol there are then exactly two messages transmitted. The messages can be sent simultaneously by the parties during the protocol execution. This implies that in any given round, the parties do not have to wait for the messages from the other parties before sending out their messages in that particular round. Hence, in a one-round key agreement protocol no party should have to wait before initiating the protocol and sending its outgoing messages (if any).

The $e^2$CK Model for Certificateless Key Agreement

We present a strengthened version of Swanson’s [Swa08] model, which in turn is based on LaMacchia, Lauter & Mityagin’s [LLM07] extended Canetti-Krawczyk (eCK) model that we reviewed in Section 2.8.2. We discuss the changes to the respective models below.

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of parties. The protocol may be run between any two of these parties. For each party there exists an identity-based public key that can be derived from its identifier. There is a key generation centre that issues identity-based private keys to the parties through a secure channel. Additionally, the parties generate their own secret values and certificateless public keys.

The adversary is in control of the network over which protocol messages are exchanged. As discussed in Section 2.8.2, we do not use the notation for session identifiers proposed by LaMacchia et al. [LLM07]. In our notation, $\Pi_{i,j}^t$ represents the $t^{th}$ protocol session which runs at party $i$ with intended partner party $j$. Additionally, the adversary is allowed to replace certificateless public keys that are used to compute the session key. The adversary does not have to disclose the private key matching the replaced certificateless public key to the respective party.

A session $\Pi_{i,j}^t$ enters an accepted state when it computes a session key $SK_{i,j}^t$. Note that a session may terminate without ever entering into an accepted state. The information of whether a session has terminated with acceptance or without acceptance is assumed to be public. The session $\Pi_{i,j}^t$ is assigned a partner ID $pid = (ID_i, ID_j)$. The session ID $sid$ of $\Pi_{i,j}^t$ at party $i$ is the transcript of the messages exchanged with party $j$ during the session. Two sessions $\Pi_{i,j}^t$ and $\Pi_{j,i}^u$ are considered matching if they have the corresponding $pid$ (and $sid$).
The game runs in two phases. During the first phase of the game, the adversary $\mathcal{M}$ is allowed to issue the following queries in any order:

**Send($\Pi_{i,j}^t, x$)**: If the session $\Pi_{i,j}^t$ does not exist, it will be created as initiator at party $i$ if $x = \lambda$, or as a responder at party $j$ otherwise. If the participating parties have not been initiated before, the respective private and public keys are created. Upon receiving the message $x$, the next protocol step is executed. After party $i$ has sent and received the last set of messages specified by the protocol, it outputs a decision indicating accepting or rejecting the session. In the case of one-round protocols, party $i$ behaves as follows:

$x = \lambda$: Party $i$ generates an ephemeral value and responds with an outgoing message only.

$x \neq \lambda$: If party $i$ is a responder, it generates an ephemeral value for the session and responds with an outgoing message $m$ and a decision indicating acceptance or rejection of the session. If party $i$ is an initiator, it responds with a decision indicating accepting or rejecting the session.

In this work, we require $i \neq j$, i.e. a party will not run a session with itself. An example where a party would run a session with itself is a user who installs his key on multiple workstations and then runs a key agreement protocol between any two of these workstations using his key on both ends.

**Reveal master key** The adversary is given access to the master secret key.

**Session key reveal($\Pi_{i,j}^t$)**: If the session has not accepted, it returns $\bot$, otherwise it reveals the accepted session key.

**Reveal ID-based secret($i$)**: Party $i$ responds with its ID-based private key.

**Reveal secret value($i$)**: Party $i$ responds with its secret value $x_i$ that corresponds to its certificateless public key. If $i$ has been asked the replace public key query before, it responds with $\bot$.

**Replace public key($i, pub$)**: Party $i$'s certificateless public key is replaced with $pub$ chosen by the adversary. All parties communicating with party $i$ will use the new public key for all communication and computation. In the strong security model, the party
Reveal ephemeral key(\(\Pi^{t}_{i,j}\)): Party \(i\) responds with the ephemeral secret used in session \(\Pi^{t}_{i,j}\).

We can group the key reveal queries into three types: the reveal master key and reveal ID-based secret queries try to undermine the security of the ID-based part of the scheme, the reveal secret value and replace public key queries try to undermine the security of the public-key-based part of the scheme, and the reveal ephemeral key query tries to undermine the security of one particular session.

We define the state fully corrupt as a session that was asked all three types of reveal queries: the reveal master key or reveal ID-based secret, the reveal secret value or the replace public key, and the reveal ephemeral key query.

Once the adversary \(\mathcal{M}\) decides that the first phase is over, it starts the second phase by choosing a fresh session \(\Pi^{t}_{i,j}\) and issuing a \(\text{Test}(\Pi^{t}_{i,j})\) query, where the fresh session and test query are defined as follows:

\begin{definition}
A session \(\Pi^{t}_{i,j}\) is fresh if (1) \(\Pi^{t}_{i,j}\) has accepted; (2) \(\Pi^{t}_{i,j}\) is unopened (not being issued the session key reveal query); (3) the session state at neither party participating in this session is fully corrupted; (4) there is no opened session \(\Pi^{u}_{j,i}\) which has a matching conversation to \(\Pi^{t}_{i,j}\).
\end{definition}

\(\text{Test}(\Pi^{t}_{i,j})\) The input session \(\Pi^{t}_{i,j}\) must be fresh. A bit \(b \in \{0, 1\}\) is randomly chosen. If \(b = 0\), the adversary is given the session key, otherwise it randomly samples a session key from the distribution of valid session keys and returns it to the adversary.

After the \(\text{test}(\Pi^{t}_{i,j})\) query has been issued, the adversary can continue querying except that the test session \(\Pi^{t}_{i,j}\) should remain fresh. We emphasize here that partial corruption is allowed as this is a benefit of our security model. Additionally, replace public key queries may be issued to any party after the test session has been completed.

At the end of the game, the adversary outputs a guess \(\hat{b}\) for \(b\). If \(\hat{b} = b\), we say that the adversary wins. The adversary’s advantage in winning the game is defined as

\[
Adv^{\mathcal{M}}(k) = \left| \Pr[\mathcal{M} \text{ wins}] - \frac{1}{2} \right|
\]
Oracle Access for Strong and Weak Certificateless Adversaries

Certificateless encryption schemes have two notions of security — strong and weak — that we discussed in more detail in Section 3.2. We give now the definitions for strong and weak certificateless key agreement schemes. We discussed in Section 3.2 why we believe that strong security does not introduce additional vulnerabilities in key agreement protocols, as opposed to encryption schemes. In Section 6.1, we present a certificateless key agreement model in the random oracle model that we prove secure against strong certificateless adversaries.

Definition 3.6. Strong Type I Secure Key Agreement Scheme

A certificateless key agreement scheme is Strong Type I secure if every probabilistic, polynomial-time adversary $M$ has negligible advantage in winning the game described in Section 3.5.1 subject to the following constraints:

- $M$ may corrupt at most two out of three types of secrets (long-term ID-based, long term user generated, short term ephemeral) per party involved in the test session,
- $M$ is allowed to replace public keys of any party; however, this counts as the corruption of one secret,
- $M$ may not reveal the secret value of any identity for which it has replaced the certificateless public key,
- $M$ is allowed to ask session key reveal queries even for session keys computed by identities where $M$ replaced the identity's public key.
- $M$ is allowed to replace public keys of any party after the test query has been issued.

Definition 3.7. Strong Type II Secure Key Agreement Scheme

A certificateless key agreement scheme is Strong Type II secure if every probabilistic, polynomial-time adversary $M$ has negligible advantage in winning the game described in Section 3.5.1 subject to the following constraints:

- $M$ is given the master secret key $s$ at the start of the game,
- $M$ may corrupt at most one additional type of secret per party participating in the test query,

- $M$ is allowed to replace public keys of any party; however, this counts as the corruption of one secret,

- $M$ may not reveal the secret value of any identity for which it has replaced the certificateless public key,

- $M$ is allowed to ask session key reveal queries even for session keys computed by identities where it replaced the identity’s public key.

- $M$ is allowed to replace public keys of any party after the test query has been issued.

Our certificateless key agreement protocol in the standard model that we present in Section 6.2 uses a certificateless key encapsulation mechanism as a building block. Since all constructions for certificateless key encapsulation mechanisms are in the weak security model, we need to develop the weak security model for certificateless key agreement to be able to prove our protocol secure in Section 6.2.

**Definition 3.8. Weak Type I $e^2$CK-Secure Key Agreement Scheme**

A certificateless key agreement scheme is Weak Type I $e^2$CK-secure if every probabilistic, polynomial-time adversary $M$ has negligible advantage in winning the game described in Section 3.5.1 subject to the following constraints.

- $M$ may corrupt at most two long-term secrets of one party involved in the test session, and one long-term secret of the other party involved in the test session.

- $M$ is allowed to replace public keys of any party; however, this counts as the corruption of one secret.

- $M$ may not reveal the secret value of any identity for which it has replaced the certificateless public key.

- $M$ is not allowed to ask session key reveal queries for session keys computed by identities where $M$ replaced the identity’s public key.

- $M$ is allowed to replace public keys of any party after the test query has been issued.
3.5. Security Model for Certificateless Key Agreement Schemes

- $M$ is not allowed to ask session state reveal queries for sessions at identities where $M$ replaced the identity’s public key.

- $M$ is not allowed to ask session state reveal queries for the test session or the matching session to the test session.

- $M$ is not allowed to ask ephemeral key reveal queries.

Definition 3.9. Weak Type II e²CK-Secure Key Agreement Scheme

A certificateless key agreement scheme is Weak Type II e²CK-secure if every probabilistic, polynomial-time adversary $M$ has negligible advantage in winning the game described in Section 3.5.1 subject to the following constraints.

- $M$ is given the master secret key $s$ at the start of the game.

- $M$ may corrupt at most one user secret key of the parties participating in the test session.

- $M$ is allowed to replace the certificateless public key of any party; however, this counts as the corruption of a user secret key.

- $M$ may not reveal the secret value of any identity for which it has replaced the certificateless public key.

- $M$ is not allowed to ask session key reveal queries for session keys computed by identities where the identity’s public key was replaced.

- $M$ is allowed to replace public keys of any party after the test query has been issued.

- $M$ is not allowed to ask session state reveal queries for sessions at identities where $M$ replaced the identity’s public key.

- $M$ is not allowed to ask session state reveal queries for the test session or the matching session to the test session.

- $M$ is not allowed to ask ephemeral key reveal queries.
Relation to Existing Notions of Security  Compared to the strong security model discussed above, Swanson’s [Swa08] replace public key query is weaker in assuming that the party whose key was replaced continues to make its computations with its original (unreplaced) public key (and its matching private key). With respect to the weak security model, Swanson’s query is stronger, because we do not allow the adversary to ask session state reveal queries for sessions at identities where \( \mathcal{M} \) replaced the identity’s public key. So Swanson’s definition would be in between strong and weak security. Swanson does not propose any protocols that are provably secure under her definition. However, we propose two provably secure protocols that match the above definitions for strong and weak security for certificateless key agreement. It would be interesting to see if there are certificateless key agreement protocols that match Swanson’s security definition. When checking for a matching conversation, Swanson omits the certificateless public keys from the conversation transcript. This weakens the adversary compared to our model, as the adversary would not be allowed to replace public keys and try to replay the conversation with the replaced keys of the test query after the test query has been issued.

With respect to the strong security model, we would like to add that in the key agreement protocol discussed in Section 6.1, the trusted authority is not able to derive session keys for the parties involved, even though we consider strong security. This is a major difference to strongly secure encryption, where the authority is able to recover the plaintexts if it does not behave according to the specification.

With respect to LaMacchia et al. [LLM07], the main difference of our definition is that instead of having only four pieces of secret information, in certificateless protocols there are six: the ID-based secret keys, the users’ secret values, and the ephemeral private keys of both parties. We require a certificateless AKE to be secure as long as each party still holds at least one uncompromised secret.

We note that as the simulator has to answer session key reveal queries even for keys where the respective certificateless public keys have been replaced, the adversary has access to the equivalent of a strong decrypt oracle in certificateless encryption.

A strong decryption oracle in public key cryptography is able to return the plaintext for a given ciphertext (which does not necessarily mean that the plaintext has been decrypted using the correct key, as with double encryption). We
3.5. Security Model for Certificateless Key Agreement Schemes

The lines indicate what combination of secrets gives resistance against which attack type. Examples for public key schemes applicable to this diagram would be NAXOS [LLM07] and CMQV [Ust08], an example for an ID-based scheme would be the ASIACCS09 [HC09] scheme. However, a combination of these schemes would not have any security guarantees about the dashed lines in the certificateless part of the diagram.

Figure 3.1: PK-AKE + ID-AKE ≠ CL-AKE

Note that in a session key reveal query the correct key for a given session has to be revealed, which is a stronger requirement. The scheme in Section 6.1 is both Strong Type I and Strong Type II secure with respect to Dent’s definitions.

3.5.2 Why a Natural Composition of CL-AKE from ID-AKE and PK-AKE is not Possible in the e²CK Model

Considering that other certificateless protocols can be constructed by combining a public key scheme with an ID-based scheme, one would think that certificateless key agreement schemes could be constructed likewise. We do not rule this possibility out for key agreement protocols that have more than one round, but we give some strong intuition why the composition of a public key agreement scheme and an ID-based key agreement scheme run in parallel in a one-round protocol cannot achieve the security that is expected in the e²CK model from a certificateless key agreement protocol in one round.

In the e²CK security model, a session can only be fresh as long as each party still has at least one uncompromised secret. A composition of an identity-based authenticated key agreement protocol (ID-AKE) with a public-key-based authenticated key agreement protocol (PK-AKE) is shown in Figure 3.1. A
natural way to achieve such a composition consists of running the two protocols in parallel and deriving the session key of the overall composition as a publicly known function of solely the two component session keys. This composition cannot offer the desired level of security, because no security guarantees exist if party $A$ still has an uncompromised key in the PK-AKE and party $B$ still has an uncompromised key in the ID-AKE (both AKE schemes are broken at this moment). This may explain why no certificateless authenticated key agreement protocol (CL-AKE) schemes with proofs of security have been published before.

3.6 General Proof Strategy

In the following, we will do many proofs for adversaries against certificateless protocols that are “upgraded versions” from a similar ID-based protocol. All of these protocols will be easier if we do not treat $Type I$ and $Type II$ adversaries separately. Essentially, there are two strategies for dealing with an adversary:

- Embed security into the ID-based part. Then the adversary may learn the secret value (the user generated private key) or replace the certificateless public key. This is generally not applicable for $Type II$ adversaries.

- Embed security into the CL-based part. Then the adversary may learn the ID-based secret key and/or the master secret key. This is applicable for both $Type I$ and $Type II$ adversaries.

For $Type I$ adversaries that want to learn the CL-key, we can leave the proofs of the ID-based protocol unmodified and hand over the $user secret key usk_{ID}$ to the adversary. The original proof still holds in this setting; we discuss the details shortly when we present the protocols in the respective sections.

For $Type II$ adversaries and $Type I$ adversaries that want to learn the ID-based key or the master secret key, we have to consider security again. But here we have to consider security only for the part that differs from the ID-based protocol. That means, the simulator generates the master secret key and can thus derive all ID-based information and/or give it to the adversary when requested to do so. We discuss the details when we present the respective protocols.
3.7 Conclusion

This chapter explained the background assumptions for certificateless encryption. Insider and outsider adversaries for certificateless schemes were defined, and the distinction between strong and weak adversaries was made. We explained how the proofs in Chapter 4, 5 and 6 will be structured. Then we continued to define the security model for certificateless encryption, certificateless key encapsulation and certificateless key agreement. We explained why a natural combination of ID-based and public-key-based key agreement protocols does not lead to a certificateless key agreement protocol. Finally, we explained the strategy that we will use for proving some of our certificateless protocols secure.
Chapter 4

Multi-Authority IBE and Certificateless Encryption Schemes

Generic constructions for certificateless encryption were proposed by Huang and Wong at IWSEC 2007 [HW07a], by Dent, Libert and Paterson at PKC 2008 [DLP08], and by Dent in IJIS [Den08]. The constructions by Huang and Wong and by Dent are moderately efficient, the construction by Dent, Libert and Paterson is very inefficient. We propose very efficient generic constructions for IND-CCA2 secure certificateless encryption schemes in the standard model, building on IND-CPA secure hierarchical identity based encryption schemes. Our constructions can be efficiently executed in parallel, making certificateless encryption as fast as ID-based encryption on modern multi-CPU computers. Efficient IND-CCA2 secure ID-based constructions are usually realised using the constructions by Boneh, Canetti, Halevi and Katz [BCHK07]. We extend the ideas by Dodis and Katz [DK05] by formulating a new formal model for adversaries in multiple encryption schemes that gives more power to the adversary. We show that the constructions for IND-CCA2 security by Boneh et al. [BCHK07] can be proven in our new security model, improving the result by Dodis and Katz. Finally, we show how to obtain very efficient generic constructions for certificateless encryption schemes from our results. A graphical overview of the chapter is in Table 4.1.
Section 4.1

Multi Authority IBE using IND-CPA secure IBE schemes
+ IND-CCA2 security from ID-based encryption [BCHK07]

Result of Section 4.2:
Multi-key IND-CCA2 secure public key encryption

Result of Section 4.3:
+ Parallel IND-CPA secure encryption (works for many IBE schemes)
More efficient multi-key IND-CCA2 secure public key encryption

Section 4.4

Multi Authority IND-CPA HIBE schemes
+ IND-CCA2 security from ID-based encryption [BCHK07]

IND-CCA2 secure multi-authority HIBE schemes
+ Parallel encryption (Section 4.3)

Result of Section 4.4:
Efficient multi-authority IND-CCA2 secure HIBE
→ Special case of 2 authorities: certificateless encryption

Result of Section 4.5:
Efficient IND-CCA2 secure certificateless encryption
in the standard model

Table 4.1: Overview of Chapter 4

Generic constructions for certificateless encryption in the standard model have been proposed by Dent, Libert, and Paterson [DLP08] Huang and Wong [HW07a], and Dent [Den08]. All current reasonably efficient generic constructions in the standard model [HW07a,Den08] use the idea of encrypting a message both under an identity-based encryption scheme and a public-key encryption scheme to obtain a certificateless encryption scheme. Furthermore, the constructions require both the identity-based encryption scheme and the public key encryption scheme to be IND-CCA2 secure and require an additional one-time signature scheme to bind the ciphertexts together and to prove the constructions secure.

We present the most efficient generic construction for certificateless encryption in the standard model. The increased efficiency results from an application of the efficient construction for IND-CCA2 security presented by Boneh et al. [BCHK07] that uses a MAC and an encapsulation scheme. Therefore the encryption schemes used need to be only IND-CPA secure. Previous constructions [HW07a,Den08] needed IND-CCA2 secure encryption schemes and signature schemes. Dent’s [Den08] generic construction for certificateless encryption
uses two independent ciphertexts, where both ciphertexts need to encrypt a message of the size of the original message. This results in ciphertexts that are more than twice as long as a single encryption of the message, as the ciphertexts have to be signed. If the construction by Huang and Wong [HW07a] is used, then the sequential encryption also leads to ciphertext expansion. Especially with respect to IND-CCA2 secure encryption schemes, encrypting a message sequentially multiple times can require a lot of resources, as the message space in a typical public key encryption scheme is smaller than the ciphertext space, so there is always some ciphertext overhead in a single encryption. Chaining encryption schemes can thus result in a significant increase of the ciphertext overhead. Additionally, the message spaces of the “outer” encryption schemes have to be adapted to fit the message plus the ciphertext overhead from the previous schemes. This may result in even worse performance since the message space is often tied to the security parameter $k$. This means, that the security parameter $k$ may have to be increased beyond the actual security requirements, resulting in even more resource consumption.

Our second contribution is to provide schemes secure in a stronger model for so-called multiple (or multi-key) encryption than previously achieved. We expand the ideas of Dodis and Katz presented at TCC 2005 [DK05], and present a new adversarial model. In our new model, the adversary may learn secret keys or even replace public keys in addition to decryption oracle access, whereas Dodis and Katz consider only decryption oracle access for the adversary. In the proof we do not answer decryption queries for the adversary if a public key was replaced and the matching private key is not disclosed to the simulator. Incidentally, our security model is almost equivalent to the security model for weak certificateless encryption, as classified by Dent [Den08]; we will exploit this in our construction of efficient generic certificateless encryption schemes. Although Dodis and Katz [DK05] claim that the constructions that allow IND-CCA2 security from IND-CPA secure ID-based encryption schemes proposed by Boneh et al. [BCHK07] also hold if a message is encrypted under the public keys of multiple authorities, they do not give a formal proof. We prove that our constructions hold in an $n$-out-of-$n$ sharing scheme even if all but one of the authorities are corrupted by an adversary. By corruption we mean revealing a secret key or replacing a public key.

Our third contribution is a way to decrease the lengths of the ciphertexts in
an $n$-out-of-$n$ threshold encryption. We propose the new notion of parallel IND-CPA secure encryption in Section 4.3, which allows us to have an encryption under multiple public keys. In comparison to the Dodis and Katz construction, we do not use a secret sharing scheme, since this is not needed for certificateless encryption. Therefore, we don’t have to encrypt multiple messages of the size of the original message with different encryption schemes. Instead, we use the fact that most identity based encryption schemes are masking or blinding the message with a random value and transmit additional information for message recovery (the ephemeral public key). We combine the masking values to mask the original message, and append the ephemeral public keys to the encrypted message. Although the ciphertext still grows linearly with the number of encryption schemes, the ciphertext overhead is smaller than in the Dodis and Katz scheme.

Dent’s [Den08] generic construction for certificateless encryption makes use of the ideas presented by Dodis and Katz, but is only able to achieve parallelised encryption and decryption as a result of applying their work to certificateless encryption. We go one step further by showing that if we construct certificateless encryption from secure hierarchical identity-based encryption, we are able to obtain both parallelisable and more efficient certificateless encryption schemes than those which are possible using a combination of identity-based encryption and public-key encryption. Our construction is obtained by having each user generate an additional master key in the hierarchical identity-based encryption scheme, and using that key as the certificateless public key.

We prove the security of our constructions in the most generic way possible. Therefore, we align our presentation to the presentation in Boneh et al. [BCHK07]. We first show how to obtain efficient multi-key IND-CCA2 secure public key encryption (or multiple encryption, as Dodis and Katz would name it) from identity-based encryption. However, since our goal is efficient multi-key ID-based encryption, we show that the constructions for IBE’s also work for HIBE schemes in Section 4.4. This results in efficient multi-key IND-CCA2 secure (hierarchical) ID-based encryption schemes. In the last two sections, we show how to obtain efficient generic constructions for certificateless encryption schemes in the standard model from this result.
4.1 Multi-Key Encryption

We proceed to define the general notion of a multi-key encryption scheme, which is basically a public key encryption scheme with multiple public keys instead of one public key. Chen et al. [CHSS02] discuss encryption under multiple authorities as well, however they also do not consider the replacement of keys in their schemes. Furthermore, their constructions seem to be only IND-CPA secure, and they do not have an explicit security model or a formal proof of security for their constructions. Since ID-based encryption is a subclass of public key based encryption, it is useful to specify this notion also for the general case of public key encryption schemes. In the following sections, we proceed to prove the ID-based variant that was informally discussed in Dodis and Katz [DK05], although in a weaker security model. We improve the result by Dodis and Katz by allowing the adversary access to decryption oracles, private keys, and giving the adversary the power to replace public keys. Dodis and Katz considered only adversarial access to decryption oracles, which is a weaker assumption. Therefore, our results supersede the strong MCCA model of Dodis and Katz, who claim only that their construction for identity-based encryption holds in the weak MCCA model. We refer the reader to their paper for details of their security models.

In Section 4.6, we give two concrete constructions: one that is based on lattices and is the first lattice-based certificateless encryption scheme, and one that is based on bilinear pairings. The construction based on bilinear pairings is used to show how efficient instantiations of our generic construction can be.

**Definition 4.1. Multi-Key Encryption Scheme**

A multi-key encryption scheme \( \text{MK-Π} \) is a triple of PPT algorithms \((\text{Gen}, \text{Enc}, \text{Dec})\) such that:

- The randomized key generation algorithm \(\text{Gen}\) takes as input a security parameter \(1^k\) and outputs a multiple public keys \((pk_1, \ldots, pk_l)\) and corresponding secret keys \((sk_1, \ldots, sk_l)\). In the following, we refer to each of these key pairs \((pk_1, sk_1), \ldots, (pk_l, sk_l)\) as a separate encryption schemes \(Π_1, \ldots, Π_l\). The full public key of the multi-key encryption scheme is \(\text{PK} = (pk_1, \ldots, pk_l)\), and the corresponding full secret key is \(\text{SK} = (sk_1, \ldots, sk_l)\). We write \((\text{PK}, \text{SK}) \leftarrow \text{Gen}(1^k)\).

- The randomized encryption algorithm \(\text{Enc}\) takes as input the public key \(\text{PK}\) and a message \(m\) (in some implicit message space) and outputs a ciphertext
Chapter 4. Multi-Authority IBE and Certificateless Encryption Schemes

C. We write $C \leftarrow \text{Enc}(\text{PK}, m)$.

- The (possibly randomized) decryption algorithm $\text{Dec}$ takes as input a ciphertext $C$ and the secret key $\text{SK}$. It returns a message or the special symbol ⊥ which is not in the message space and indicates an invalid ciphertext. We write $m \leftarrow \text{Dec}(\text{SK}, C)$.

We require $\forall (\text{PK}, \text{SK}) \leftarrow \text{Gen}(1^k) : \text{Dec}(\text{SK}, \text{Enc}(\text{PK}, m)) = m$.

We explain the security model for multi-key encryption schemes.

Definition 4.2. Oracles for adversaries against multi-key / multi authority encryption schemes

To model the security guarantees of a multi-key / multi authority scheme correctly, we introduce the following model. An adversary $M$ against the scheme has access to the following oracles:

Reveal key($i$): The adversary is given access to the secret key of $\Pi_i$ in the multi-key / multi authority construction.

Replace public key($i, pk$): The public key of scheme $\Pi_i$ is replaced with $pk$ chosen by the adversary. All communication (encryption) will use the new public key. If the construction is IND-CCA2 secure, then all decryption queries are answered with ⊥ unless the adversary discloses the private key that matches $pk$.

In the following, whenever an adversary queries one of these oracles, we refer to this as a corruption query. For the security proofs to hold, we assume that the adversary may corrupt at most all but one encryption scheme.

When considering multi-key encryption schemes, we also consider the standard security notions of IND-CPA security and IND-CCA2 security. Since we are only interested in constructions of IND-CCA2 secure multi-key encryption schemes, we give only the security notion for IND-CCA2 secure multi-key encryption schemes.

Definition 4.3. IND-CCA2 Secure Multi Key Encryption

A multi-key encryption scheme $\text{MK-Π}$ is IND-CCA2 secure if the advantage of any PPT adversary $M$ in the following game is negligible in the security parameter $k$:
1. \((PK, SK) \leftarrow \text{Gen}(1^k)\). Adversary \(M\) is given \(PK\).

2. The adversary may corrupt at most all but one of the encryption schemes \(\Pi_i\) before the challenge ciphertext is requested. After the challenge ciphertext has been requested, the public keys of all encryption schemes may be replaced but not revealed.

3. The adversary may reveal a private key \(sk_i\) matching a replaced public key \(pk_i\) which was replaced as the result of a corruption query. The simulator checks if \(sk_i\) matches \(pk_i\) by constructing \(pk_i\) from \(sk_i\). If this is the case, then \(sk_i\) is accepted, otherwise \(sk_i\) is rejected.

4. If the adversary \(M\) did not replace any public keys of the encryption schemes, then \(M\) may make polynomially-many queries to a decryption oracle \(\text{Dec}_{SK}(\cdot)\). Otherwise, if the adversary replaced one or more public keys and disclosed all matching private keys, and all private keys were accepted by the simulator, then the adversary may make polynomially-many decryption queries. Otherwise, if one or more public keys were replaced and the matching private keys were not disclosed, then all queries to the decryption oracle are answered with \(\bot\).

5. At some point, \(M\) outputs two messages \(m_0, m_1: |m_0| = |m_1|\). A bit \(b\) is randomly chosen (we write \(b \overset\$\leftarrow \{0, 1\}\)), and the adversary is given a challenge ciphertext \(C^* = \text{Enc}(pk, m_b)\).

6. \(M\) may continue to query its decryption oracle \(\text{Dec}_{sk}(\cdot)\), except that it may not request the decryption of \(C^*\).

7. Finally, \(M\) outputs a guess \(b'\) for \(b\).

We say that the adversary \(M\) succeeds if \(b' = b\) and denote the probability of this event by \(P_{M,\Pi, \Pi}^{\text{IND-CCA2 MK}}[\text{Succ}]\). The adversary's advantage is defined as \(\text{Adv}_{M,\Pi, \Pi}^{\text{IND-CCA2 MK}}(k) = \left| P_{M,\Pi, \Pi}^{\text{IND-CCA2 MK}}[\text{Succ}] - \frac{1}{2} \right|\).

We describe now how to obtain multi-key encryption schemes from identity-based encryption schemes.
4.2 CCA Security from ID-Based Encryption Reconsidered for Multi-Key Encryption Schemes

Boneh, Canetti, Halevi and Katz [BCHK07] construct IND-CCA2 security from ID-based encryption in two ways. The first way, which is shown in Figure 4.1, results in an IND-CCA2 secure public key encryption scheme. The construction uses a selective identity IND-CPA secure IBE (see Definition 2.24) and a strong one-time signature scheme (see Definition 2.15), and combines them to obtain an IND-CCA2 secure public key encryption scheme. The second construction uses a strong one-time MAC (see Definition 2.17) and an encapsulation scheme (see Definition 2.18) to obtain a similar construction. Boneh et al. [BCHK07] also show how to construct an ID-based scheme using a similar construction, but with a HIBE instead of an IBE.

We show that both constructions given by Boneh, Canetti, Halevi and Katz [BCHK07] hold in a generic multi-authority setting, where the user chooses a set of authorities and relies on the fact that at least one of them is honest. Our only restriction is that the simulator does not have to answer decryption queries if the adversary replaces a public key and does not disclose the matching private
key (this corresponds to the “weak” certificateless security model discussed in Chapter 3, which Dent [Den08] classifies as the “most realistic”).

For the construction, we run $l$ ID-based schemes simultaneously. This results in one IND-CCA2 secure multi-key encryption scheme. In the certificateless case discussed in Section 4.5, the first ID-based scheme represents the ID-based scheme that is under control of the trusted authority, which must be hierarchical, so that the resulting construction is still ID-based. The other ID-based scheme will be set up by the user, and does not have to be hierarchical, as it is only used to obtain a certificateless IND-CCA2 secure scheme in the standard model. Both ID-based schemes need to be only IND-CPA secure, IND-CCA2 security is then obtained by the constructions discussed below.

It would seem intuitive to use the proof discussed by Boneh et al. [BCHK07] as a black box and just show how to do a direct reduction from the multi-authority case. This is not possible because the construction by Boneh et al. uses an IND-CPA secure identity-based encryption scheme and a strong one-time signature scheme or a MAC and an encapsulation scheme together to obtain an IND-CCA2 secure encryption scheme. The proof that Boneh et al. give relies on the fact that the IND-CPA secure encryption is done under an “identity” that is the verification key of either the signature scheme or the MAC (encapsulated by the encapsulation scheme). If we wanted to use this construction as a black box, then we would have to pass the messages that we obtain from the adversary to the Boneh et al. construction. In return, we would obtain an IND-CCA2 secure ciphertext. However, if we now encrypted only the part of the ciphertext that contains the message with IND-CPA secure encryption schemes, then the ciphertext would become invalid (because any modification of the bits of the ciphertext will result in an invalid ciphertext). A way around this would be to encrypt the complete ciphertext each time with a full Boneh et al. construction, which would be much less efficient than the scheme that we propose here. Another way around it would be to specify that decryption first decrypts all the IND-CPA secure schemes and finally decrypts the IND-CCA2 secure scheme. However, this would specify exactly which of the trusted authorities is “really” trusted, and discriminate against the other authorities. So we choose to encrypt under the verification key for all authorities, and apply the signature/MAC only after all encryptions are performed. But then we cannot treat the Boneh et al. construction as a blackbox, and have to prove anew that the construction is still
valid. This is what we do below.

### 4.2.1 Signature-Based CCA Security from Multi-Authority IBE

Given $l$ SID-IND-CPA secure ID-based schemes $\Pi_1 = (\text{IBE Setup}_1, \text{IBE KeyDer}_1, \text{IBE Enc}_1, \text{IBE Dec}_1), \ldots, \Pi_l = (\text{IBE Setup}_l, \text{IBE KeyDer}_l, \text{IBE Enc}_l, \text{IBE Dec}_l)$ (see Definition 2.23) that support identities of length $n$, and a strong one-time signature scheme $\text{Sig} = (\mathcal{G}, \text{Sign}, \text{Ver})$ (see Definition 2.15) with verification keys of length $n$, we construct a multi-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ (see Definition 4.1) secure against adaptive chosen-ciphertext attacks. Since the resulting scheme is just a “normal” public key encryption scheme, it seems somewhat futile to construct a public key encryption scheme that uses multiple keys. However, later we will use the scheme in a hierarchical ID-based setting, where each ID is a root node for at least one level of hierarchy below. This will then result in a multi-authority ID-based encryption scheme, and will enable us to give a construction of an IND-CCA2 secure certificateless encryption scheme in the standard model. In the certificateless setting, each ID will generate a certificateless key (that is, publish an additional, possibly non-hierarchical IBE scheme; or just generate a new master key pair for the current scheme and publish that as the certificateless public key).

We now proceed to describe the multi-key encryption scheme, which is shown in Figure 4.2.

**Gen** On input $1^k$, the key generation algorithm $\text{Gen}$ runs $\text{IBE Setup}_1(1^k)$ to obtain the public key $\text{PK}_1 = (\text{param}_1, \text{mpk}_1)$ and the matching secret key $msk_1$, to $\text{PK}_l = (\text{param}_l, \text{mpk}_l)$ and $msk_l$.

The public key is $\text{PK} = (\text{param}_1, \text{mpk}_1, \ldots, \text{param}_l, \text{mpk}_l)$ and the secret key is $\text{SK} = (msk_1, \ldots, msk_l)$.

**Enc** To encrypt a message $m$, the sender runs $\mathcal{G}(1^k)$ to obtain a verification key $vk$ and a signing key $sk$ from the strong one-time signature scheme $\text{Sig}$. The sender then computes $C_1 \leftarrow \text{IBE Enc}_1(\text{param}_1, \text{mpk}_1, vk, m)$ up to $C_l \leftarrow \text{IBE Enc}_l(\text{param}_l, \text{mpk}_l, vk, C_{l-1})$ and computes $\sigma = \text{Sign}_sk(C_l)$. The final ciphertext is $C = (vk, C_l, \sigma)$.

**Dec** To decrypt a ciphertext $C = (vk, C_l, \sigma)$ using the secret key $\text{SK} = (msk_1,$
4.2. CCA Security from ID-Based Encryption Reconsidered for Multi-Key Encryption Schemes

Not ID-based.
- Shorter than adding a signature every time.

Figure 4.2: An IND-CCA2 secure multi-key encryption scheme based on the CHK construction.

..., msk_l), the receiver first checks whether \( \text{Ver}_{vk}(C_l, \sigma) \neq 1 \). If not, the receiver simply outputs \( \bot \). Otherwise, the receiver computes \( d_{vk,IBE_i} \leftarrow IBE \text{KeyDer}_i(\text{param}_1, msk_1, vk) \) to \( d_{vk,IBE_i} \leftarrow IBE \text{KeyDer}_i(\text{param}_l, msk_l, vk) \) and computes \( C_{l-1} \leftarrow IBE \text{Dec}_i(\text{param}_l, d_{vk,IBE_i}, vk, C_l) \) to \( m \leftarrow IBE \text{Dec}_i(\text{param}_1, d_{vk,IBE_i}, vk, C_1) \).

The algorithms for MK-II satisfy correctness.

We allow an adversary \( \mathcal{M} \) against the scheme to partially corrupt the user. A corruption is to learn the master secret key or to replace the public key of a scheme. The adversary is allowed to corrupt all but one encryption scheme. We claim that the scheme is IND-CCA2 secure even if \( \mathcal{M} \) may corrupt all but one encryption scheme. However, as we prove security in the “weak” certificateless model discussed in Section 3.2, decryption queries are not answered if a public key is replaced and the matching private key is not disclosed.

To state our claim, we rephrase [BCHK07, Theorem 1] for the certificateless case.

**Theorem 4.1.** If IBE-Π_1 to IBE-Π_L are identity-based encryption schemes that are selective-ID secure against chosen-plaintext attacks (SID-IND-CPA-IBE) and
Sig is a strong one-time signature scheme, then MK-II is a multi-key encryption scheme secure against chosen-ciphertext attacks.

Overview of the Proof

The Boneh et al. [BCHK07] proof is divided into two parts, one for each component that the scheme consists of. In the first part, an adversary against the multi-key encryption scheme is used to forge a signature with respect to the one-time signature scheme Sig. Due to the security of the signature scheme, the probability of this event happening is negligible. In the second part, an adversary against II is used to get an advantage in the IND-CPA security game against a selective-identity secure identity-based encryption scheme. Due to the SID-IND-CPA security of the IBE scheme, the probability of this event happening is also negligible.

Our construction uses \( l \) SID-IND-CPA secure ID-based encryption schemes IBE-II_1 to IBE-II_l, and a one-time signature scheme Sig. An adversary against our system II can corrupt all but one of the IBE schemes. We want to reason that the adversary can either attack the one-time signature scheme Sig or one of the IBE schemes. The proof of security for the attack against the one-time signature scheme Sig is exactly the same as in Boneh et al. [BCHK07, Section 4, Claim 1]. To see that the adversary can also be used against a SID-IND-CPA secure IBE scheme, we introduce a decisional phase before the game starts. The simulator randomly decides which of the IBE schemes may not be corrupted by the adversary before the simulation starts. All IBE schemes that can be corrupted are set up by the simulator so that all information can be disclosed if necessary. The other IBE scheme, say IBE-II_x, is the scheme that shall be attacked. We now have to deal with two caveats. If the adversary decides to corrupt the IBE scheme that cannot be corrupted (IBE-II_x), then the simulator aborts the game and starts again. This reduces the success probability of the simulator by the factor \((l - 1)/l\). The second caveat is the challenge ciphertext. The challenge ciphertext may be requested after the adversary replaced the public key of the corruptable schemes. To request the challenge ciphertext, the adversary submits two messages \( m_0, m_1 : |m_0| = |m_1| \). The simulator encrypts both of them using the encryption algorithm under identity \( vk^* \), until an encryption with IBE-II_x is needed. The simulator then forwards the output of IBE-II_{x-1} for \( m_0 \) and \( m_1 \) (that is, \( C^*_{x-1}(vk^*, m_0) \) and \( C^*_{x-1}(vk^*, m_1) \)) to the simulator for IBE-II_x, and gets
4.2. CCA Security from ID-Based Encryption Reconsidered for Multi-Key Encryption Schemes

a ciphertext $C^*_x$ under identity $vk^*$ back. Then, the simulator continues to encrypt $C^*_x$ with the current public keys of $\text{IBE-Π}_{x+1}$ to $\text{IBE-Π}_l$ to obtain $C^*_l$ and signs it using $vk^*$ to obtain the final challenge ciphertext $C^* = \langle vk^*, C^*_l, \sigma^* \rangle$ and forwards $C^*$ to the adversary against $\Pi$. Clearly, any decision that a successful adversary makes on $C^*$ can be used to get an advantage on $\text{IBE-Π}_x$. Thus, the proof of security in the Boneh et al. [BCHK07, Section 4, Claim 2] holds as well. Notice that the simulator is not required to answer decryption queries if the public key of any IBE has been replaced and has not been disclosed to the simulator, thus all decryption queries that must be answered can be answered successfully.

Detailed Proof

This subsection is a rewrite of the proof by Boneh et al. [BCHK07] for the above scheme. The presentation is very similar to the presentation in Boneh et al. [BCHK07].

Proof. Assume we are given a ppt adversary $M$ attacking $\Pi$ in an adaptive chosen-ciphertext attack. Say a ciphertext $\langle vk, C, \sigma \rangle$ is valid if $\text{Ver}_{vk}(C, \sigma) = 1$. Let $\langle vk^*, C^*, \sigma^* \rangle$ denote the challenge ciphertext received by $M$ during a particular run of the experiment, and let $\text{Forge}$ denote the event that $M$ submits a valid ciphertext $\langle vk^*, C, \sigma \rangle$ to the decryption oracle with with $C \neq C^*$ or $\sigma \neq \sigma^*$ or both. We may assume that $vk^*$ is chosen at the outset of the experiment, so this event is well-defined even before $M$ is given the challenge ciphertext. By definition of the IND-CCA experiment, $M$ is disallowed from submitting the challenge ciphertext to the decryption oracle once the challenge ciphertext is given to $M$. We prove the following claims:

Claim 1. $\text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Forge}]$ is negligible.

Claim 2. $\left| \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Succ} \land \text{Forge}] + \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Forge}] - \frac{1}{2} \right|$ is negligible.

The rationale why these two claims imply Theorem 4.1 is:

\[
\left| \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Succ}] - \frac{1}{2} \right| \\
\leq \left| \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Succ} \land \text{Forge}] - \frac{1}{2} \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Forge}] \right| + \left| \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Succ} \land \text{Forge}] + \frac{1}{2} \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Forge}] - \frac{1}{2} \right| \\
\leq \frac{1}{2} \text{Pr}_{M,\Pi}[^{\text{Forge}] + \left| \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Succ} \land \text{Forge}] + \frac{1}{2} \text{Pr}_{M,\Pi}[^{\text{PKE}}\text{Forge}] - \frac{1}{2} \right| .
\]
We continue to prove Claim 1.

**Proof.** We construct a ppt forger $\mathcal{F}$ who forges a signature with respect to a signature scheme $\text{Sig}$ (see Definition 2.14) with probability exactly $\Pr^{\text{PKE}}_{\mathcal{M},\Pi}[\text{Forge}]$. Security of $\text{Sig}$ implies the claim.

$\mathcal{F}$ is defined as follows: given input $1^k$ and a verification key $vk^*$ (output by $G$), $\mathcal{F}$ first runs the IBE Setup algorithms to obtain PK and SK, and then runs $\mathcal{M}(1^k, \text{PK})$. Note that $\mathcal{F}$ can answer any valid decryption queries of $\mathcal{M}$. If $\mathcal{M}$ happens to submit a valid ciphertext $\langle vk^*, C, \sigma \rangle$ to its decryption oracle before requesting the challenge ciphertext, then $\mathcal{F}$ simply outputs the forgery $(C, \sigma)$ and stops. Otherwise, when $\mathcal{M}$ outputs the two messages $m_0$ and $m_1$, $\mathcal{F}$ proceeds as follows: it chooses a random bit $b$, computes $C_1^* = \text{IBE Enc}_1(\text{param}_1, \text{mpk}_1, vk^*, m_b)$ to $C_l^* = \text{IBE Enc}_l(\text{param}_l, \text{mpk}_l, vk^*, C_{l-1}^*)$, and obtains (from its signing oracle) a signature $\sigma^*$ on $C_l^*$. Finally, $\mathcal{F}$ hands the challenge ciphertext $\langle vk^*, C, \sigma \rangle$ to $\mathcal{M}$. If $\mathcal{M}$ submits a valid ciphertext $\langle C, \sigma \rangle$ to its decryption oracle, then we must have $\langle C, \sigma \rangle \neq \langle C_1^*, \sigma^* \rangle$. In this case $\mathcal{F}$ simply outputs $(C, \sigma)$ as its forgery. It is easy to see that $\mathcal{F}$’s success probability (in the sense of Definition 2.14) is exactly $\Pr^{\text{PKE}}_{\mathcal{M},\Pi}[\text{Forge}]$. \qed

We now prove Claim 2.

**Proof.** We use $\mathcal{M}$ to construct a ppt adversary $\mathcal{M}'$ which attacks one of the IBE schemes in the construction (in the sense of Definition 2.24). Define the adversary $\mathcal{M}'$ as follows.

1. $\mathcal{M}'$ enumerates the IBE schemes in the construction and randomly selects one to attack (let’s call it $\text{IBE-}\Pi_x$). $\mathcal{M}'$ constructs the other IBE schemes with the respective IBE Setup algorithms for security parameter $1^k$.

2. $\mathcal{M}'$ runs $G(1^k)$ to generate $(vk^*, sk^*)$, and outputs the “target” identity $\text{ID}^* = vk^*$ for $\text{IBE-}\Pi_x$.

3. $\mathcal{M}'$ is given the master public key for $\text{IBE-}\Pi_x$. $\mathcal{M}'$ uses it together with the output of the other IBE Setup algorithms to construct PK. It then runs $\mathcal{M}(1^k, \text{PK})$.

4. When $\mathcal{M}$ makes a corruption query, adversary $\mathcal{M}'$ proceeds as follows:

   (a) If $\mathcal{M}$ wants to learn the master secret key of $\text{IBE-}\Pi_x$, then $\mathcal{M}'$ aborts the game and returns a random guess.
4.2. CCA Security from ID-Based Encryption Reconsidered for Multi-Key Encryption Schemes

(b) If $\mathcal{M}$ wants to replace the master public key of IBE-$\Pi_x$, then $\mathcal{M}'$ aborts the game and returns a random guess.

(c) If $\mathcal{M}$ wants to learn any other the master secret key, then $\mathcal{M}'$ can disclose it. ($\mathcal{M}'$ is in control of the other IBE)

(d) If $\mathcal{M}$ wants to replace any other master public key, then $\mathcal{M}'$ will not answer any decryption queries any more, unless $\mathcal{M}$ discloses the matching private key to $\mathcal{M}'$.

5. When $\mathcal{M}$ makes a decryption oracle query $D((vk, C, \sigma))$, simulator $\mathcal{M}'$ proceeds as follows:

(a) If any master public key was replaced, and there was no matching private key submitted by $\mathcal{M}$, then $\mathcal{M}'$ returns $\bot$.

(b) If $vk = vk^*$ then $\mathcal{M}'$ checks whether $\text{Verify}_{vk^*}(C, \sigma) = 1$. If so, then $\mathcal{M}'$ aborts and outputs a random bit. Otherwise, $\mathcal{M}'$ responds with $\bot$. Note that if $\text{Verify}_{vk^*}(C, \sigma) = 1$, then $\mathcal{M}$ produced a valid forgery for Sig which is negligible due to the security of Sig.

(c) If $vk \neq vk^*$ and $\text{Verify}_{vk}(C, \sigma) = 0$ then $\mathcal{M}'$ responds with $\bot$.

(d) If $vk \neq vk^*$ and $\text{Verify}_{vk}(C, \sigma) = 1$ then $\mathcal{M}'$ makes an oracle query IBE KeyDer($vk$) to IBE-$\Pi_x$ to obtain the decryption key for $vk$ (which is the attacked IBE), and derives the decryption key for $vk$ from the other IBE’s. It then computes $m \leftarrow \text{Dec}_{sk_{vk}}(vk, C)$ and responds with $m$.

6. At some point, $\mathcal{M}$ outputs two equal-length messages $m_0, m_1 : |m_0| = |m_1|$. The simulator $\mathcal{M}'$ encrypts both of them using the encryption algorithm under identity $vk^*$, until an encryption with IBE-$\Pi_x$ is needed. The simulator $\mathcal{M}'$ then forwards the output of IBE-$\Pi_{x-1}$ for $m_0$ and $m_1$ (that is, $C^*_{x-1, vk^*, m_0}$ and $C^*_{x-1, vk^*, m_1}$) to the simulator for IBE-$\Pi_x$, and gets a ciphertext $C^*_x$ under identity $vk^*$ back. Then, the simulator continues to encrypt $C^*_x$ with the current public keys of IBE-$\Pi_{x+1}$ to IBE-$\Pi_l$ under $vk^*$ to obtain $C^*_l$ and signs it using $vk^*$ to obtain the final challenge ciphertext $C^* = (vk^*, C^*_l, \sigma^*)$ and forwards $C^*$ to the adversary against $\Pi$.

7. $\mathcal{M}$ may continue to make decryption oracle queries and/or key corruption queries, these are answered by $\mathcal{M}'$ as before.
8. Finally $\mathcal{M}$ outputs a guess $b'$, the same guess is output by $\mathcal{M}'$.

Note that $\mathcal{M}'$ represents a legal adversarial strategy for attacking $\text{IBE-}\Pi_\times$; in particular, $\mathcal{M}'$ never requests the secret key corresponding to the “target” identity $vk^*$. Furthermore, $\mathcal{M}'$ provides a perfect simulation for $\mathcal{M}$ until the event $\text{Forge}$ occurs, or it has to abort the game due to invalid requests by $\mathcal{M}$ (which happens only with probability $(l-1)/l$ in each game). It is thus easy to see that

$$\left| \text{IBE}_{\mathcal{M}'}\text{IBE-}\Pi_x[\text{Succ}] - \frac{1}{2} \right| \geq \frac{1}{2} \left| \frac{l-1}{l} \text{PKE}_{\mathcal{M}'}[\text{Succ} \land \text{Forge}] + \frac{1}{2} \text{PKE}_{\mathcal{M}}[\text{Forge}] - \frac{1}{2} \right|$$

$$= \frac{1}{2} \left| \frac{2(l-1)}{l} \text{PKE}_{\mathcal{M}'}[\text{Succ} \land \text{Forge}] + \text{PKE}_{\mathcal{M}}[\text{Forge}] \right|,$$

and the left-hand side of the above is negligible by the assumed security of $\text{IBE-}\Pi_x$.

This concludes the proof of the theorem.

### 4.2.2 More Efficient CCA Security

The second construction given in Boneh et al. [BCHK07, Section 5] is also applicable to our construction for SID-IND-CCA certificateless encryption scheme (CLE) schemes. The construction described by Boneh et al. [BCHK07] uses an SID-IND-CPA IBE scheme $\text{IBE-}\Pi = (\text{IBE Setup}, \text{IBE KeyDer}, \text{IBE Enc}, \text{IBE Dec})$, a secure encapsulation scheme $(\text{Init}, S, R)$ (see Definition 2.18) and a message authentication code $(\text{Mac}, \text{Vrfy})$ (see Definition 2.16). The multi-key encryption scheme $\text{MK-}\Pi$ is constructed as follows:

**Gen** On input $1^k$, the key generation algorithm $\text{Gen}$ runs $\text{IBE Setup}_1(1^k)$ to $\text{IBE Setup}_l(1^k)$ to obtain $\text{PK-IBE}_1 = (\text{param}_1, \text{mpk}_1)$ to $\text{PK-IBE}_l = (\text{param}_l, \text{mpk}_l)$ and the matching secret keys $\text{msk}_1$ to $\text{msk}_l$. The public key $\text{pub}$ for the encapsulation scheme is generated by $\text{pub} \leftarrow \text{Init}(1^k)$.

The public key is $\text{PK} = (\text{param}_1, \text{mpk}_1, \ldots, \text{param}_l, \text{mpk}_l, \text{pub})$ and the secret key is $\text{SK} = (\text{msk}_1, \ldots, \text{msk}_l)$.

**Enc** To encrypt a message $m$, the sender encapsulates a random value by running $(r, \text{com}, \text{dec}) \leftarrow S(1^k, \text{pub})$. The message is then padded with $\text{dec}$
4.2. CCA Security from ID-Based Encryption Reconsidered for Multi-Key Encryption Schemes

1. To obtain \( m \circ \text{dec} \). The sender parses \((\text{param}_1, \text{mpk}_1, \ldots, \text{param}_l, \text{mpk}_l) \leftarrow \text{PK}\). The sender then computes \( C_1 \leftarrow \text{IBE Enc}_1(\text{param}_1, \text{mpk}_1, m \circ \text{dec}) \) to \( C_l \leftarrow \text{IBE Enc}_l(\text{param}_l, \text{mpk}_l, C_{l-1}) \) and computes \( \text{tag} = \text{Mac}(r, C_l) \). The final ciphertext is \( C = \langle \text{com}, C_l, \text{tag} \rangle \).

2. \text{Dec} To decrypt a ciphertext \( C = \langle \text{com}, C_l, \text{tag} \rangle \) using the secret key \( \text{SK} = (\text{msk}_1, \ldots, \text{msk}_l) \) the receiver first derives the identity-based secret keys corresponding to the “identity” \( \text{com} \): the receiver computes \( d_{\text{com}, \text{IBE}_1} \leftarrow \text{IBE KeyDer}_1(\text{param}_1, \text{msk}_1, \text{com}) \) to \( d_{\text{com}, \text{IBE}_l} \leftarrow \text{IBE KeyDer}_l(\text{param}_l, \text{msk}_l, \text{com}) \). Then, the receiver uses these keys to decrypt \( C_l \): \( C_{l-1} = \text{IBE Dec}_l(d_{\text{com}, \text{IBE}_1}, \text{com}, C_l) \). If the decryption fails, then the receiver outputs \( \bot \). Next, the receiver runs \( R(\text{pub}, \text{com}, \text{dec}) \) to obtain a string \( r \); if \( r \neq \bot \) and \( \text{Verify}_r(C_l, \text{tag}) = 1 \) then the receiver outputs \( m \). Otherwise, the receiver outputs \( \bot \).

3. **Theorem 4.2.** If \( \text{IBE-Pi}_1 \) to \( \text{IBE-Pi}_l \) are SID-IND-CPA secure identity-based encryption schemes, the encapsulation scheme is secure and \( (\text{Mac}, \text{Verify}) \) is a strong one-time message authentication code, then \( \text{MK-Pi} \) is an IND-CCA2 secure multi-key encryption scheme.

The proof of the construction follows from the proof by Boneh et al. [BCHK07]. As in the proof for the signature-based variant, we introduce a decisional stage ahead of the game: the simulator randomly determines which of the ID-based schemes will be corrupted by the adversary. All other schemes will be set up so that all relevant information can be given to the adversary. Since each ciphertext is encrypted relative to a particular “identity” \( \text{com} \) for that ciphertext, all ciphertexts can easily be decrypted (except for the challenge ciphertext).

**Proof.** Let \( \mathcal{M} \) be a ppt adversary attacking \( \text{MK-Pi} \) in an adaptive chosen-ciphertext attack. Let \( \langle \text{com}^*, C^*, \text{tag}^* \rangle \) denote the challenge ciphertext received by \( \mathcal{M} \). We will show that (1) \( \mathcal{M} \) submits to its decryption oracle a valid ciphertext \( \langle \text{com}^*, C, \text{tag} \rangle \) (with \( \langle C, \text{tag} \rangle \neq \langle C^*, \text{tag}^* \rangle \)) only with negligible probability; and (2) assuming that the previous event does not occur, the decryption queries made by \( \mathcal{M} \) do not help \( \mathcal{M} \) to “learn” the underlying plaintext. The second statement is relatively easy to prove based on the security of one \( \text{IBE-Pi} \) scheme; the first, however, is more challenging to prove since the validity of a ciphertext cannot be determined without knowledge of \( \text{msk} \). Because of this we structure the proof...
as a sequence of games to make it easier to follow. We let \( \Pr_i[-] \) denote the probability of a particular event occurring in Game \( i \).

**Game 0.** This is the original game in which \( \mathcal{M} \) attacks MK-II in an IND-CCA2 attack. Let \( r^*, \text{com}^*, \text{dec}^* \) denote the values that are used in computing the challenge ciphertext, and notice that these values can be generated even before the experiment with \( \mathcal{M} \) starts. We are interested in upper-bounding \( |\Pr_0[\text{Succ}] - \frac{1}{2}| \), where \( \text{Succ} \) denotes the event that \( \mathcal{M} \)'s output bit \( b' \) is identical to the bit \( b \) used in constructing the challenge ciphertext.

**Game 1.** The experiment is modified as follows: on input of a ciphertext of the form \( \langle \text{com}^*, C, \text{tag}^* \rangle \), the decryption oracle simply outputs \( \bot \). Let \( \text{Valid} \) denote the event that \( \mathcal{M} \) submits a ciphertext \( \langle \text{com}^*, C, \text{tag} \rangle \) to its decryption oracle which is valid, and note that

\[
|\Pr_1[\text{Succ}] - \Pr_0[\text{Succ}]| \leq \Pr_0[\text{Valid}] = \Pr_1[\text{Valid}].
\]

The above holds since Games 0 and 1 are identical until \( \text{Valid} \) occurs.

Let \( \text{NoBind} \) denote the event that \( \mathcal{M} \) at some point submits a ciphertext \( \langle \text{com}^*, C, \text{tag} \rangle \) to its decryption oracle such that: (1) \( C \) decrypts to \( m \circ \text{dec} \) (using the secret keys derived from the respective IBE schemes) and (2) \( R(\text{pub}, \text{com}, \text{dec}) = r \) with \( r \not\in \{r^*, \bot\} \). Let \( \text{Forge} \) denote the event that \( \mathcal{M} \) at some point submits a ciphertext \( \langle \text{com}^*, C, \text{tag} \rangle \) to its decryption oracle such that \( \text{Verify}_{r^*}(C, \text{tag}) = 1 \).

We clearly have \( \Pr_1[\text{Valid}] \leq \Pr_1[\text{NoBind}] + \Pr_1[\text{Forge}] \).

One can show that \( \Pr_1[\text{NoBind}] \) is negligible assuming the binding property of the encapsulation scheme (see Definition 2.19). Formally, consider an adversary \( \mathcal{B} \) acting as follows: given input \( (1^k, \text{pub}, \text{com}^*, \text{dec}^*) \), adversary \( \mathcal{B} \) generates \( (\text{PK}, \text{SK}) \) by running \( \text{Setup}(1^k) \), and then runs \( \mathcal{M} \) on inputs \( 1^k \) and \( (\text{PK}, \text{pub}) \). Whenever \( \mathcal{M} \) makes a query to its decryption oracle, \( \mathcal{B} \) can respond to this query as required by Game 1; specifically, \( \mathcal{M} \) simply responds with \( \bot \) to a decryption query of the form \( \langle \text{com}^*, C, \text{tag} \rangle \), and responds to other queries using \( \text{SK} \). When \( \mathcal{M} \) submits its two messages \( m_0, m_1 \), adversary \( \mathcal{B} \) simply chooses \( b \in \{0, 1\} \) at random and encrypts \( m_b \) in the expected way to generate a completely valid challenge ciphertext \( \langle \text{com}^*, C^*, \text{tag}^* \rangle \) (note that this is even possible when \( \mathcal{M} \) requested to replace IBE keys). \( \mathcal{B} \) can easily do this because it has \( \text{dec}^* \) and can compute \( r^* \). At the end of the experiment, \( \mathcal{B} \) can decrypt every valid decryption query of the form \( \langle \text{com}^*, C, \text{tag} \rangle \) that \( \mathcal{M} \) made to see whether \( \text{NoBind} \) occurred and, if so, learn the value \( \text{dec} \) such that \( R(\text{pub}, \text{com}^*, \text{dec}) \not\in \{\bot, r^*\} \). Notice that
\( \mathcal{M} \) is forbidden to make decryption queries once it replaced one of the IBE’s public keys and does not disclose the matching private key, so it is guaranteed that \( \mathcal{B} \) can decrypt all queries that are made to its decryption oracle. If \( \mathcal{B} \) finds a \( \text{dec} \) such that \( R(\text{pub}, \text{com}^*, \text{dec}) \not\in \{\bot, r^*\} \), this violates the binding property of the encapsulation scheme, and thus \( \Pr_1[\text{NoBind}] \) must be negligible.

**Game 2.** We modify the way the challenge ciphertext is computed. Specifically, when \( \mathcal{M} \) submits its two messages \( m_0, m_1 \) we now compute \( C^* \leftarrow \text{Enc}_{PK}(\text{com}, 0| m_0 | \circ 0^n) \) followed by \( \text{tag} \leftarrow \text{Mac}_{r^*}(C^*) \). The challenge ciphertext is \( \langle \text{com}^*, C^*, \text{tag}^* \rangle \). A random bit \( b \) is still chosen, but only to define event \( \text{Succ} \). Since the challenge ciphertext is independent of \( b \), it follows that \( \Pr_2[\text{Succ}] = \frac{1}{2} \).

We claim that \( | \Pr_2[\text{Succ}] - \Pr_1[\text{Succ}] | \) is negligible. To see this, consider the following adversary \( \mathcal{M}' \) attacking one of the SID-IND-CPA secure IBE schemes.

- Algorithm \( \mathcal{M}' \) first runs \( \text{Init}(1^k) \) to generate \( \text{pub} \) and then runs \( \text{S}(1^k, \text{pub}) \) to obtain \( (r^*, \text{com}^*, \text{dec}^*) \). It randomly chooses which IBE scheme to attack, let us name it IBE-\( \Pi_x \). The other IBE schemes are set up by \( \mathcal{M}' \); it uses the respective IBE Setup\( (1^k) \) algorithms to generate \( \text{param}_1, \text{mpk}_1, \text{msk}_1, \ldots, \text{param}_{l-1}, \text{mpk}_{l-1}, \text{msk}_{l-1} \). \( \mathcal{M}' \) outputs \( \text{com}^* \) as the target identity for IBE-\( \Pi_x \) and is then given \( \text{param}_x, \text{mpk}_x \). It then inserts IBE-\( \Pi_x \) at a random position in the list of IBE schemes and constructs \( \text{PK} \) and runs \( \mathcal{M} \) on input \( (1^k, \text{PK}, \text{pub}) \).

- When \( \mathcal{M} \) makes a corruption query, adversary \( \mathcal{M}' \) proceeds as follows:
  1. If \( \mathcal{M} \) wants to learn the master secret key of IBE-\( \Pi_x \), then \( \mathcal{M}' \) aborts the game and returns a random guess.
  2. If \( \mathcal{M} \) wants to replace the master public key of IBE-\( \Pi_x \), then \( \mathcal{M}' \) aborts the game and returns a random guess.
  3. If \( \mathcal{M} \) wants to learn any other master secret key, then \( \mathcal{M}' \) can disclose it. (\( \mathcal{M}' \) is in control of the other IBE)
  4. If \( \mathcal{M} \) wants to replace any other master public key, then \( \mathcal{M}' \) will not answer any decryption queries unless \( \mathcal{M} \) discloses the matching private key.

- Decryption queries of \( \mathcal{M} \) are then answered in the natural way:
  - If any master public key was replaced, then \( \mathcal{M}' \) returns \( \bot \) unless the matching private key is disclosed to \( \mathcal{M}' \).
Queries of the form $\langle \text{com}^*, C, \text{tag} \rangle$ are answered with $\bot$.

Queries of the form $\langle \text{com}, C, \text{tag} \rangle$ with $\text{com} \neq \text{com}^*$ are answered by first querying $\text{IBE KeyDer}_x(\text{com})$ to obtain $d_{\text{IBE}_x, \text{com}}$, then computing $d_{\text{IBE}, \text{com}} \leftarrow \text{IBE KeyDer}_1(msk_1, \text{com})$ to $d_{\text{IBE}_{l-1}, \text{com}} \leftarrow \text{IBE KeyDer}_{l-1}(msk_{l-1}, \text{com})$, and then decrypting in the usual way.

- Eventually, $\mathcal{M}$ submits two equal-length messages $m_0, m_1$. $\mathcal{M}'$ selects a bit $b$ at random and proceeds as follows. The simulator encrypts $m_b \circ \text{dec}^*$ and $0^{\|m_0\|} \circ 0^n$ using the encryption algorithm under “identity” $\text{com}^*$, until an encryption with $\text{IBE-}\Pi_x$ is needed. The simulator then forwards the output of $\text{IBE-}\Pi_{x-1}$ for $m_b \circ \text{dec}^*$ and $0^{\|m_0\|} \circ 0^n$ (that is, $C_{x-1, \text{com}^*, m_b \circ \text{dec}^*}$ and $C_{x-1, \text{com}^*, 0^{\|m_0\|} \circ 0^n}$) to the simulator for $\text{IBE-}\Pi_x$, and gets a ciphertext $C^*_x$ under identity $\text{com}^*$ back. Then, the simulator continues to encrypt $C^*_x$ with the current public keys of $\text{IBE-}\Pi_x + 1$ to $\text{IBE-}\Pi_l$ under $\text{com}^*$ to obtain $C^*_l$. $\mathcal{M}'$ then computes $\text{tag}^* \leftarrow \text{Mac}_{r^*}(C^*_l)$ and returns $C^* = \langle \text{com}^*, C^*_l, \text{tag}^* \rangle$ to $\mathcal{M}$.

- Further decryption queries of $\mathcal{M}$ are answered as above.

- Finally, $\mathcal{M}$ outputs a bit $b'$. If $b = b'$, then $\mathcal{M}'$ outputs 0; otherwise, $\mathcal{M}'$ outputs 1.

Note that $\mathcal{M}'$ is a valid adversary. When the encryption query of $\mathcal{M}'$ is answered with an encryption of $m_b \circ \text{dec}^*$, then the view of $\mathcal{M}$ is exactly as in Game 1; on the other hand, when the encryption query of $\mathcal{M}'$ is answered with $0^{\|m_0\|} \circ 0^n$, then the view of $\mathcal{M}$ is exactly as in Game 2. Note, that the probability $\mathcal{M}'$ aborting the game with $\mathcal{M}$ is at most $\frac{l-1}{l}$, depending on the corruption queries of $\mathcal{M}$. Thus,

$$\text{Adv}_{\mathcal{M}', \text{IBE-}\Pi_x}^{\text{IBE}}(k) = \frac{l-1}{l} \left| \frac{1}{2} \text{Pr}_1[\text{Succ}] + \frac{1}{2} \text{Pr}_2[\text{Succ}] - \frac{1}{2} \right|$$

$$= \frac{l-1}{2l} \left| \text{Pr}_1[\text{Succ}] - \text{Pr}_2[\text{Succ}] \right|$$

The security of $\text{IBE-}\Pi_x$ implies that $\text{Adv}_{\mathcal{M}', \text{IBE-}\Pi_x}^{\text{IBE}}$ is negligible, implying that $|\text{Pr}_2[\text{Succ}] - \text{Pr}_1[\text{Succ}]|$ is negligible. An exactly analogous argument shows that $|\text{Pr}_2[\text{Forge}] - \text{Pr}_1[\text{Forge}]|$ is negligible as well. The only difference in the forging argument is that $\mathcal{M}'$ runs $\mathcal{M}$ to completion and checks whether $\mathcal{M}$ made any decryption query of the form $\langle \text{com}^*, C, \text{tag} \rangle$ for which $\text{Verify}_{r^*}(C, \text{tag}) = 1$. This
4.2. CCA Security from ID-Based Encryption Reconsidered for Multi-Key
Encryption Schemes

is possible for $\mathcal{M}'$ because $\mathcal{M}$ can issue these queries only while $\mathcal{M}'$ knows all private keys: then $\mathcal{M}'$ can always decrypt. If $\mathcal{M}'$ finds such a query, then it outputs 1, else it outputs 0.

**Game 3.** The components $c^*$ and $C^*$ of the challenge ciphertext are computed as in Game 2, however, the component $\mathbf{tag}^*$ is now computed by choosing a random key $r \in \{0, 1\}^k$ and setting $\mathbf{tag}^* = \text{Mac}_r(C^*)$ (that is, $\mathbf{tag}^*$ is now completely independent of $c^*$ and $\mathbf{dec}^*$, and a decryption of this ciphertext with $\text{Dec}$ would result in $\bot$). Event $\text{Forge}$ in this game is defined as before, but using the key $r$; that is $\text{Forge}$ is now the event that $\mathcal{M}$ makes a decryption query of the form $\langle c^*, C, \mathbf{tag} \rangle$ for which $\text{Verify}_r(C, \mathbf{tag}) = 1$.

We claim that $|\Pr_3[\text{Forge}] - \Pr_2[\text{Forge}]|$ is negligible. To see this, consider the following algorithm $B$ attacking the hiding property of the encapsulation scheme (that is, $B$ has to determine if a given $\hat{r}$ is valid for a given $c$ without knowing $\text{dec}$):

- $B$ is given input $1^k$ and $(\text{pub}, c^*, \hat{r})$. It then runs $\text{IBE Setup}_1(1^k)$ to $\text{IBE Setup}_l(1^k)$ to generate $(\text{PK}, \text{SK})$, and then runs $\mathcal{M}$ on $(1^k, \text{PK}, \text{pub})$.
- Corruption queries of $\mathcal{M}$ are answered in the natural way.
- Decryption queries of $\mathcal{M}$ are answered in the natural way.
- Eventually, $\mathcal{M}$ submits two messages $m_0, m_1$. $B$ computes $C_1^* \leftarrow \text{IBE Enc}_1(\mathbf{param}_1, \mathbf{mpk}_1, c^*, 0^{\text{mod} \cdot 0^n})$ to $C_l^* \leftarrow \text{IBE Enc}_l(\mathbf{param}_l, \mathbf{mpk}_l, \mathbf{com}^*, C_{l-1})$, computes $\mathbf{tag}^* = \text{Mac}_r(C_l^*)$, and returns the challenge ciphertext $C^* = \langle c^*, C_l^*, \mathbf{tag}^* \rangle$ to $\mathcal{M}$.
- Further decryption / corruption queries of $\mathcal{M}$ are answered as above.
- When $\mathcal{M}$ halts, $B$ checks whether $\mathcal{M}$ has made any decryption query of the form $\langle c^*, C, \mathbf{tag} \rangle$ for which $\text{Verify}_r(C, \mathbf{tag}) = 1$. If so, $B$ outputs 1; otherwise, it outputs 0.

Now, if $\hat{r}$ is such that $(\hat{r}, c^*, \mathbf{dec}^*)$ was output by $S(1^k, pub)$, then the view of $\mathcal{M}$ is exactly as in Game 2 and so $B$ outputs 1 with probability $\Pr_2[\text{Forge}]$. On the other hand, if $\hat{r}$ is chosen at random independently of $c^*$, then the view of $\mathcal{M}$ is exactly as in Game 3 and so $B$ outputs 1 with probability $\Pr_3[\text{Forge}]$. The hiding property of the encapsulation scheme thus implies that $|\Pr_3[\text{Forge}] - \Pr_2[\text{Forge}]|$ is negligible.
We now have to show that $\Pr_3[\text{Forge}]$ is negligible. This follows from the security of the message authentication code. Let $q = q(k)$ be an upper bound on the number of decryption oracle queries made by $\mathcal{M}$. Consider the following forging algorithm $\mathcal{F}$. First, $\mathcal{F}$ chooses a random index $j \leftarrow \{1, \ldots, q\}$. Next, $\mathcal{F}$ begins simulating Game 3 for $\mathcal{M}$ as above. If the $j$th decryption query $\langle \text{com}_j, \text{C}_j, \text{tag}_j \rangle$ occurs before $\mathcal{M}$ makes its encryption query, then $\mathcal{F}$ simply outputs $\langle \text{C}_j, \text{tag}_j \rangle$ as its forgery for the Mac and halts. Otherwise, in response to the challenge encryption query $(m_0, m_1)$ of $\mathcal{M}$, forger $\mathcal{F}$ computes $(r^*, \text{com}^*, \text{dec}^*) \leftarrow \text{S}(1^k, \text{pub})$ and encrypts $C^*_i \leftarrow \text{IBE Enc}_1(\text{param}, \text{mpk}, \text{com}^*, 0|\text{m}_0| \circ 0^n)$ to $C^*_l \leftarrow \text{IBE Enc}_l(\text{param}, \text{mpk}, \text{com}^* \text{C}_{l-1})$. $\mathcal{F}$ then submits $C^*_i$ to its Mac oracle and receives in turn $\text{tag}^*$. $\mathcal{F}$ then returns $C^* = \langle \text{com}^*, C^*_i, \text{tag}^* \rangle$ to $\mathcal{M}$ as the challenge ciphertext and continues running $\mathcal{M}$ until $\mathcal{M}$ submits its $j$th decryption query $\langle \text{com}_j, \text{C}_j, \text{tag}_j \rangle$. At this point, $\mathcal{F}$ outputs $\langle \text{C}_j, \text{tag}_j \rangle$ as its Mac forgery and halts.

Considering Game 3, it is easy to see that the success probability of $\mathcal{F}$ in outputting a valid forgery is at least $\Pr_3[\text{Forge}]\frac{q}{l}$. Since (Mac, Verify) is a strong one-time message authentication code and $ql$ is polynomial, this shows that $\Pr_3[\text{Forge}]$ is negligible.

Putting everything together, we have

\[
\left| \Pr_0[\text{Succ}] - \frac{1}{2} \right| \leq \left| \Pr_0[\text{Succ}] - \Pr_1[\text{Succ}] \right| + \left| \Pr_1[\text{Succ}] - \frac{1}{2} \right| \\
\leq \Pr_1[\text{NoBind}] + \Pr_1[\text{Forge}] + \left| \Pr_1[\text{Succ}] - \Pr_2[\text{Succ}] \right| \\
+ \left| \Pr_2[\text{Succ}] - \frac{1}{2} \right| \\
= \Pr_1[\text{NoBind}] + \Pr_1[\text{Forge}] + \left| \Pr_1[\text{Succ}] - \Pr_2[\text{Succ}] \right| \\
+ \left| \Pr_2[\text{Succ}] - \frac{1}{2} \right| \\
\leq \Pr_1[\text{NoBind}] + \Pr_3[\text{Forge}] + \left| \Pr_2[\text{Forge}] - \Pr_3[\text{Forge}] \right| \\
+ \left| \Pr_1[\text{Forge}] - \Pr_2[\text{Forge}] \right| + \left| \Pr_1[\text{Succ}] - \Pr_2[\text{Succ}] \right| \\
and all terms in the final equation are negligible.

\[ \square \]

### 4.3 Parallel IND-CPA Encryption

To make the constructions in the previous section even more efficient, we note that the encryption does not necessarily have to be performed sequentially / serially. However, we defined the constructions there using sequential encryption, because then the constructions are applicable to all IBE schemes. In this section,
we propose a way to do IND-CPA secure encryption in parallel. This offers two advantages: first, on modern multiprocessor computers, both encryption and decryption can be performed in parallel, and second, the resulting ciphertext is shorter. The shorter ciphertext results from the fact that the outer encryption schemes have to encrypt a ciphertext (and not a plaintext message), which is usually longer than the plaintext message.

We continue now to give the definition of an IND-CPA secure encryption scheme that masks messages.

**Definition 4.4. Masking IND-CPA Secure Encryption Scheme**

The encryption algorithm of a masking IND-CPA secure (MSK-IND-CPA secure) encryption scheme operates in the following way. The message $m \in M$ is enciphered with a one-time mask $\text{mask}$ that is generated from the public key $pk$ using an ephemeral value $e$ by the mask creation function $f$: $\text{mask} = f(pk, e)$. Then a recovery value $r$ generated from the ephemeral value $e$ by the one-way function $i$ by computing $r = i(e)$, and is sent along with the enciphered message to form the ciphertext $C$. We write

$$C = \langle m \otimes \text{mask}, r \rangle = \langle m \otimes f(pk, e), i(e) \rangle = \langle C_0, r \rangle$$

where $\otimes$ denotes the enciphering of the message by combining the message with the mask as specified in the description of the IND-CPA secure encryption scheme. Waters [Wat09] calls masking “blinding”.

We note that the decryption algorithm must follow the inverse pattern to recover the message: it first parses $C$ as $\langle C_0, r \rangle$ and then computes mask from $r$ using the secret key $sk$ and the recovery function $s$: $\text{mask} = s(sk, r)$, and then uses the inverse operation of $\otimes$ (we label it $\oslash$) to recover $m$: $m = C_0 \oslash \text{mask}$.

Consider the El-Gamal encryption scheme as an example. The message space is a group $G$ of prime order $p$ with a generator $g$. A public key is a random $y \in G$, and an ephemeral value is $e \xleftarrow{\$} \mathbb{Z}_p$. To form a ciphertext, the one-time mask is constructed by $\text{mask} = f(y, e) = ye$. The recovery value is created by $r = i(e) = g^e$. The message $m \in G$ is then encrypted by sending

$$C = \langle m \otimes \text{mask}, r \rangle = \langle m \cdot ye, g^e \rangle$$

Given the private key $sk = \log_y g \in \mathbb{Z}_p$, decryption is then done by parsing $C$ as $\langle C_0, r \rangle$ and computing $m = C_0 \oslash s(sk, r) = C_0 / rzsk$. 
Other examples of masking IND-CPA secure encryption schemes are:

1. The Canetti-Halevi-Katz SID-IND-CPA secure HIBE [CHK03]
2. The Boneh-Boyen SID-IND-CPA secure IBE [BB04a]
3. The Boneh-Boyen IND-CPA secure IBE [BB04b]
4. The Heng-Kurosawa IBE [HK04]
5. The Boneh-Boyen-Goh IND-CPA secure HIBE [BBG05]
6. Waters’ IBE [Wat05]
7. Waters’ IBE and HIBE [Wat09]

We note that schemes that are MSK-IND-CPA secure and use the same enciphering operation $\otimes$ and have the same message space support parallel multi-key encryption and decryption. The idea behind the construction is to show that a successful adversary against the multi-key encryption scheme can be used to construct a successful adversary against one particular instance of a MSK-IND-CPA secure encryption scheme. Thus we show that the multi-key encryption scheme is secure as long as at least one key is not compromised by the adversary. We will now formalize this idea.

**Definition 4.5. MK-MSK-IND-CPA secure encryption scheme**

A multi-key masking IND-CPA secure (MK-MSK-IND-CPA secure) encryption scheme formed from $l$ MSK-IND-CPA secure encryption schemes $\Pi_1, \ldots, \Pi_l$ that use the same enciphering operation $\otimes$ and have the same message space is a tuple of probabilistic polynomial time (PPT) algorithms $(\text{MK-MSK Setup}, \text{MK-MSK Enc}, \text{MK-MSK Dec})$ such that:

- **MK-MSK Setup** On input $1^k$ where $k \in \mathbb{N}$ is a security parameter, it fixes an ordering of the schemes $\Pi_1, \ldots, \Pi_l$ and executes $\Pi_1\text{Setup}(1^k), \ldots, \Pi_l\text{Setup}(1^k)$ to generate $(pk_1, sk_1), \ldots, (pk_l, sk_l)$. It then sets the public key to $PK = (pk_1, \ldots, pk_l)$ and the private key to $SK = (sk_1, \ldots, sk_l)$.

- **MK-MSK Enc** parses $PK$ to obtain $(pk_1, \ldots, pk_l) \leftarrow PK$. It then generates $l$ ephemeral secrets $(e_1, \ldots, e_l)$ and computes $l$ masking values $\text{mask}_1, \ldots, \text{mask}_l = f(pk_1, e_1), \ldots f(pk_l, e_l)$ and $l$ recovery values $r_1, \ldots, r_l =$
The message $m$ is then encrypted in the ciphertext $C$ by computing

$$C = (m \otimes \text{mask}_1 \otimes \ldots \otimes \text{mask}_l, r_1, \ldots, r_l) = (m \bigotimes_{j=1}^l \text{mask}_j, r_1, \ldots, r_l)$$

We note that this increases the size of the ciphertext only linearly. We also note that $\text{mask}_1$ to $\text{mask}_l$ and $r_1$ to $r_l$ can be computed in parallel as they are independent of each other.

• MK-MSK Dec parses $SK$ to obtain $(sk_1, \ldots, sk_l) \leftarrow SK$ and parses $C$ as $(C_0, r_1, \ldots, r_l) \leftarrow C$. It then computes $\text{mask} = \text{mask}_1 \otimes \ldots \otimes \text{mask}_l = s(sk_1, r_1) \otimes \ldots \otimes s(sk_l, r_l)$ and recovers $m$ by computing $m = C_0 \otimes \text{mask}$.

The security model for an adversary against a multi-key MSK-IND-CPA secure encryption scheme is as follows:

4.3.1 Security of MK-MSK-IND-CPA Secure Encryption Schemes

To model the security guarantees of a MK-MSK-IND-CPA secure encryption scheme correctly, we introduce the following model. An adversary $M$ against a MK-MSK-IND-CPA secure encryption scheme has access to the following oracles:

Reveal key($i$): The adversary is given access to the secret key of $\Pi_i$ in the MK-MSK-IND-CPA construction

Replace public key($i, pk$): The public key of scheme $\Pi_i$ is replaced with $pk$ chosen by the adversary. All communication (encryption) will use the new public key. Decryption queries are answered with $\bot$ unless the adversary discloses the private key that matches $pk$.

Get challenge encryption($m_0, m_1$): The adversary supplies two equal length messages $m_0, m_1$ and requests a challenge key encryption. The simulator returns a challenge ciphertext as described in Experiment 4.1 below.

Definition 4.6. MK-MSK-IND-CPA secure encryption

The security game against an adversary $M$ for a MK-MSK-IND-CPA secure encryption scheme is as follows:
encryption scheme \( \Pi \) is associated with the following experiment.

**Experiment Challenge** \( \text{Challenge}_{\Pi, M}^{mk-msk-ind-cpa}(k) : \)

\[
\begin{align*}
(PK, SK) & \overset{\$}{\leftarrow} \text{MK-MSK Setup}(1^k) \\
(m_0, m_1, \text{state}) & \overset{\$}{\leftarrow} M^{\text{Oracles}}(\text{find}, PK) \\
b & \overset{\$}{\leftarrow} \{0, 1\} \\
C^* & \overset{\$}{\leftarrow} \text{MK-MSK Enc}(PK, m_b) \\
b' & \overset{\$}{\leftarrow} M^{\text{Oracles}}(\text{guess}, C^*, \text{state}) \\
\text{Return} & \ b == b'
\end{align*}
\tag{4.1}
\]

The advantage an adversary \( M \) has against a MK-MSK-IND-CPA scheme is therefore expressed by

\[
\text{Adv}_{M}^{mk-msk-ind-cpa}(k) = \left| \text{Pr} \left[ \text{Experiment Challenge}_{\Pi, M}^{mk-msk-ind-cpa}(k) \right] - \frac{1}{2} \right|
\]

*Oracles* refers to the two oracle queries reveal key and replace public key listed above with the restriction that the adversary must not compromise at least one of the schemes \( \Pi_i \) that are composed to obtain MK-MSK-\( \Pi \).

**Theorem 4.3.** A successful adversary \( M \) against a MK-MSK-IND-CPA secure encryption scheme can be used to construct an adversary \( M' \) against one instance of a MSK-IND-CPA secure encryption scheme.

*Proof.* \( M' \) chooses the MSK-IND-CPA secure scheme \( \Pi_i \) to attack at random from the schemes used in the MK-MSK-IND-CPA secure scheme, and obtains \((pk_i)\) from its challenger. It then sets up all other MSK-IND-CPA secure schemes on its own, constructs PK and forwards PK to \( M \). \( M' \) is able to answer all corruption queries (reveal private key / replace public key) except for scheme \( \Pi_i \). If a corruption query of \( M \) targets \( \Pi_i \), then \( M' \) aborts the game and outputs a random bit. Note that due to IND-CPA security, \( M' \) does not have to answer any decryption queries. Once it receives the messages \( m_0, m_1 \) from \( M \), it forwards them to its challenger and receives a ciphertext \( C^* = \langle C_0^*, r_i^* \rangle \) back. \( M' \) then generates \( l \) ephemeral secrets, and uses them to compute \( l \) masks and \( l \) recovery values. It then replaces the recovery value \( r_i \) with \( r_i^* \) and discards mask \( i \). From the remaining masks it computes \( C^{*'} = \langle C_0^* \bigotimes_{j=1}^{l-1} \text{mask}_j, r_1, \ldots, r_l \rangle \) and returns \( C^{*'} \) to \( M \). Once \( M \) outputs its guess \( b' \) for \( b \), \( M' \) outputs the same guess. We
see easily that
\[
\Pr_{\mathcal{M}',\text{MSK-II}}^{\text{ind-}\text{cpa}}[\text{Succ}] \geq \frac{1}{t} \Pr_{\mathcal{M},\text{MK-MSK-II}}^{\text{ind-}\text{cpa}}[\text{Succ}].
\]

Thus, if the success probability of an adversary against \( \Pi_i \) is negligible, then the success probability of an adversary against MK-MSK-II is also negligible. \( \square \)

Since IBE schemes are a subclass of public key encryption schemes, we note that if the SID-IND-CPA secure IBE schemes used in Section 4.2 can be used as masking SID-IND-CPA secure IBE schemes, then the construction can be updated to use an MK-MSK-SID-IND-CPA secure IBE encryption as described above. This will make the construction more efficient in most cases, because encryption and decryption computations can be done in parallel, and the public keys of the schemes can be shorter in most cases. It can be verified that both constructions for IND-CCA2 security from Section 4.2 hold also with an multi-key masking selective-identity IND-CPA secure (MK-MSK-SID-IND-CPA secure) IBE. The intuition behind this is that masking selective-identity IND-CPA secure (MSK-SID-IND-CPA secure) IBE schemes are still SID-IND-CPA secure, and thus the proof must still hold. We give an explicit proof for the HIBE case in Theorem 4.4 in the next section. This proof can be adopted for the standard IBE case as well.

### 4.4 CCA Security for multi-authority HIBE Schemes

The above construction yields only IND-CCA2 secure public key encryption schemes, but not actual identity-based encryption schemes. To use the above constructions as identity-based schemes, we need to use hierarchical IBE schemes and perform the CCA security construction only in the level of the hierarchy “below” the user, as described by Boneh et al. [BCHK07].

In reality, when using multi-authority IBE encryption, a user may be at different levels of the hierarchy for different authorities. This does not pose a problem, as the padding technique introduced later will make sure that the “identity” used for verification purposes (\( vk \) in the CHK construction and \( \text{com} \) in the BK construction) is uniquely distinguishable from the path that leads to the identity. Figure 4.3 shows our result. We use this section to illustrate how the proof of the masking schemes presented in Section 4.3 works, and give the
Chapter 4. Multi-Authority IBE and Certificateless Encryption Schemes

Figure 4.3: An ID-based IND-CCA2 secure multi-key encryption scheme

- Full ID-based functionality, short ciphertexts.

proof using the strong one-time signature construction. A similar proof can be
given for the MAC-based construction from Section 4.2.2.

We want to combine HIBE schemes, of possibly different depths, in a multi-
key HIBE scheme. For a given HIBE scheme we write $l_x$-$\text{HIBE}_x$ to indicate
that HIBE $x$ has maximal depth $l_x$. When we combine multiple HIBE schemes
into one encryption scheme, we get a $\vec{l}$-HIBE scheme as the result where $\vec{l} =
(l_1, \ldots, l_b)$ is a vector of depths. To construct an $\vec{l}$-HIBE, we require $b$ HIBE
schemes, that support ID’s of length $n + 1$ in each HIBE level.

For the one-time signature scheme construction, we also require a signature
scheme $\text{Sig} = G, \text{Sign}, \text{Ver}$. A verification key in the signature scheme has length
$n$. An identity $\text{ID} = \vec{\text{ID}}_1, \ldots, \vec{\text{ID}}_b$ in the scheme consists of $b$ ID-vectors $\vec{\text{ID}}_i =
\vec{v}_i = (v_{i,1}, \ldots, v_{i,L}) \in \{0, 1\}^n$. For the construction, an identity is encoded in
4.4. CCA Security for Multi-Authority HIBE Schemes

the following way.

\[ \text{EncodeAll}(ID) : \]

\[ \text{for } i \in (1 \ldots b); \text{do} \]

\[ \text{Encode}(\vec{id}_i) \]

\[ \text{done} \]

where \( \text{Encode}(\vec{v}_i) = (0v_{1,1}, \ldots, 0v_{1,L_1}) \in (\{0,1\}^{n+1})^L \). That is, \( \text{Encode} \) prepends a 0 to every element of \( \vec{v}_i \). This means that an identity \( ID \) in the multi-authority IBE scheme is mapped to the padded and longer identity \( \hat{ID} = \text{EncodeAll}(ID) \) in the respective HIBE schemes. The secret key \( \overrightarrow{sk_{ID}} \) for an identity \( ID \) is the secret keys for the mapped identity \( \hat{ID} \): \( \overrightarrow{sk_{ID}} = (sk_{\text{Encode}(\vec{id}_1)}, \ldots, sk_{\text{Encode}(\vec{id}_b)}) \).

To encrypt a message \( m \) to \( ID \), the sender generates a one-time signature key pair \((vk, sk)\) and then encrypts \( m \) to \( \hat{ID} \circ 1vk = ((0v_{1,1}, \ldots, 0v_{1,L_1}, 1vk), \ldots, (0v_{b,1}, \ldots, 0v_{b,L_b}, 1vk)) \) using the respective HIBE schemes. The resulting ciphertext \( C \) is then signed using \( sk \): \( \sigma = \text{Sign}(C, sk) \), and \( \langle vk, C, \sigma \rangle \) is sent to the recipient. The padding ensures that the decryption queries asked by the adversary cannot target children of the target identity vector (which would be a legal query in a HIBE scheme). Furthermore, it enables a node in the construction to both have children below and receive CCA secure ciphertexts. So the final multi-authority HIBE scheme \( \Pi \) is constructed as follows.

**Setup** On input \( 1^k \), the \( \text{Setup} \) algorithm runs for each HIBE \( i \) \((\text{param}_i, mpk_i, msk_i) \leftarrow \text{Setup}_i(1^k) \). The public key is \( PK = (\text{param}_1, mpk_1, \ldots, \text{param}_b, mpk_b) \) and the secret key is \( SK = (msk_1, \ldots, msk_b) \). Note that this algorithm is in practice run distributed, that is, each authority \( i \) runs the setup algorithm for their HIBE scheme and publishes \( \text{param}_i, mpk_i \).

**Key derivation** To derive the secret key for a child node \( r \), the parent node \( ID \) runs for each HIBE \( sk_{\overrightarrow{id}_{Dor}} \leftarrow \text{KeyDer}_i(\text{Encode}(\vec{id}_i), \text{Encode}(r)) \) (here \( \text{Encode}(r) \) considers \( r \) to be a one-element vector) and returns \( \overrightarrow{sk_{ID_{Dor}}} = (sk_{\vec{id}_{1,or}}, \ldots, sk_{\vec{id}_{b,or}}) \).

**Encryption** \( \text{Enc}(ID, m) \) first runs \( \text{Gen}(1^k) \) to obtain \((vk, sk)\). It then encrypts to \( \hat{ID} \circ (1vk) = ((0v_{1,1}, \ldots, 0v_{1,L_1}, 1vk), \ldots, (0v_{b,1}, \ldots, 0v_{b,L_b}, 1vk)) \) by computing \( C = (m \otimes mask_1 \otimes \ldots \otimes mask_b, r_1, \ldots, r_b) \), and \( \sigma = \text{Sign}(C, sk) \). The final ciphertext is \( \langle vk, C, \sigma \rangle \).
Decryption \text{Dec}(\text{ID}, \langle \nu k, C, \sigma \rangle)$ first checks if $\text{Verify}_\nu k(C, \sigma) = 1$, if not it outputs $\bot$. Else it derives the private keys for $\text{Encode}(\text{ID}) \circ (1 \nu k)$ from the respective HIBE schemes by running $sk_{\text{ID}_1 \circ (1 \nu k)} \leftarrow \text{KeyDer}_i(\text{Encode}(\text{ID}_i) \circ (1 \nu k))$ to $sk_{\text{ID}_b \circ (1 \nu k)} \leftarrow \text{KeyDer}_b(\text{Encode}(\text{ID}_b) \circ (1 \nu k))$ and then decrypting by computing $\text{mask}_1 = s(sk_{\text{ID}_1 \circ (1 \nu k)}, r_1)$ to $\text{mask}_b = s(sk_{\text{ID}_b \circ (1 \nu k)}, r_b)$ and decrypting by $m = C_0 \odot (\bigotimes_{j=1}^b \text{mask}_j)$.

An analogous construction can be given using the MAC-based construction from Section 4.2.2. We now state the main theorem for this Section.

**Theorem 4.4.** If the $b$ HIBE schemes $(l_1+1)\text{HIBE}_1$ to $(l_b+1)\text{HIBE}_b$ are selective-ID secure against chosen-plaintext attacks and $\text{Sig}$ is a strong one-time signature scheme, then the multi-key hierarchical IBE scheme $\Pi$ is selective-ID secure against chosen-ciphertext attacks.

The proof in [BCHK07, Section 6] can be transformed for multi-authority hierarchical IBE schemes as well. The transformation uses the same as techniques as in Section 4.2. We now give the full proof.

As in Section 4.2.1, we have to show that $\frac{1}{2} \left| \frac{2(b-1)}{b} \text{Pr}_{\mathcal{M},\Pi}^{\text{HIBE}}[\text{Succ} \wedge \text{Forge}] + \text{Pr}^{\text{HIBE}}_{\mathcal{M},\Pi}[\text{Forge}] \right|$ and $\text{Pr}[\text{Forge}]$ are negligible for the signature-based construction. The technique used in the proof below can also be used to prove the construction from Section 4.2.2 secure.

**Proof.** The adversary against the signature scheme works exactly as in Section 4.2.1, as the simulator is again able to construct all IBE schemes and disclose all keys / decrypt all messages. Since forging the signature is independent of the contents of the ciphertext, this can be verified easily even without being able to decrypt.

The adversary $\mathcal{M}'$ against one of the HIBE schemes is constructed as follows. Given a PPT adversary $\mathcal{M}$ that has an advantage against the construction, we construct an adversary $\mathcal{M}'$ that has an advantage against the CPA security of the masking IND-CPA secure encryption schemes. $\mathcal{M}'$ is defined as follows.

1. $\mathcal{M}'(1^k)$ runs $\mathcal{M}(1^k)$ who outputs ID-vectors $\text{ID} = (\text{ID}_1, \ldots, \text{ID}_b)$. $\mathcal{M}'$ runs $G(1^k)$ to generate $(\nu k^*, sk^*)$, randomly selects $\text{ID}_i \leftarrow \{\text{ID}_1, \ldots, \text{ID}_b\}$ and outputs the target ID vector $V^* = \text{Encode}(\text{ID}_i) \circ (1 \nu k^*)$ for HIBE $i$ to its challenger.
2. In return, $M'$ is given the public key and the public parameters $(\text{param}_i, mpk_i)$ for HIBE $i$. $M'$ sets up the other HIBE schemes by running $\text{Setup}_1, \ldots, \text{Setup}_b$, discarding the output of $\text{Setup}_i$ and replacing it with the key and parameters for HIBE $i$ obtained from its challenger. $M'$ then forwards the public parameters of the scheme to $M$.

3. Unless $M$ wants to corrupt (replace master public key / reveal master secret key) HIBE $i$, $M'$ can simulate all queries. When $M$ requests the secret keys for ID-vectors $ID = ID_1, \ldots, ID_b$, then $M'$ executes $\text{EncodeAll}(ID)$, requests the secret key for ID-vector $\text{Encode}(ID_i)$ and derives the secret keys or the other encoded ID’s from the HIBE parameters under its control. Note that since ID must not be a prefix of the target ID-vectors ID$^*$ of $M$, it follows that $\text{Encode}(ID_i)$ is not a prefix of V$^*$. If $M$ replaced the public key of any HIBE and did not disclose the matching private key, then $M'$ is unable to derive the secret key for that particular ID-vector and returns $\bot$ in the respective position of the private key. If $M$ wants to corrupt HIBE $i$, then $M'$ aborts and outputs a random guess. The probability of $M$ corrupting HIBE $i$ is at most $(b - 1)/b$.

4. When $M$ makes a decryption query $\text{Dec}(\text{ID}, \langle vk, C, \sigma \rangle)$ (where $\text{ID} = ID_1, \ldots, ID_b$), adversary $M'$ proceeds as follows.

   - If $M$ replaced one of the master public keys and did not disclose the matching private key, then $M'$ returns $\bot$.
   - If $ID_i = ID^*_i$ and $vk = vk^*$, then $M'$ checks whether $\text{Verify}(C, \sigma) = 1$. If so, then $M'$ aborts and outputs a random bit. Notice that this constitutes a valid forgery under $vk^*$, so $M$ is an adversary against the signature scheme in this case. If $\text{Verify}(C, \sigma) \neq 1$, then $M'$ returns $\bot$.
   - If $ID_i \neq ID^*_i$ or if $ID_i = ID^*_i$ and $vk \neq vk^*$, then $M$ computes $\text{EncodeAll}(ID)$ and derives the secret key for $\text{Encode}(ID_i) \circ (1vk)$ from HIBE $i$. Note that $\text{Encode}(ID_i) \circ (1vk)$ cannot be a prefix of V$^*$, so $M'$ is able to obtain the required secret key. $M'$ computes the remaining secret keys for $\text{Encode}(ID_1) \circ (1vk)$ to $\text{Encode}(ID_b) \circ (1vk)$ from the HIBE schemes under its control. It decrypts the submitted ciphertext and returns the result to $M$. 


5. When $\mathcal{M}$ outputs its two messages $m_0, m_1$, $\mathcal{M}'$ outputs these messages to HIBE $i$ as well. In return, $\mathcal{M}'$ receives a challenge ciphertext $C_i^* = \langle C_0^*, r_i^* \rangle$. $\mathcal{M}'$ generates $b$ ephemeral keys $e_i^*, \ldots, e_b^*$ and uses them to generate $b$ masks $\text{mask}_1^*, \ldots, \text{mask}_b^*$ and likewise recovery values $r_1^*, \ldots, r_b^*$. It discards mask $\text{mask}_i$ and recovery value $r_i$, so it still has $b - 1$ masks. $\mathcal{M}'$ sets $C^* = \langle C_0^* \otimes_{j=1,j\neq i}^b, \text{mask}_j^*, r_1^*, \ldots, r_i^*, \ldots, r_b^* \rangle$ and computes $\sigma^* \leftarrow \text{Sign}_{sk^*}(C^*)$. Note that $\mathcal{M}'$ can compute these values even if $\mathcal{M}$ replaced the public keys of some HIBE schemes and did not disclose the matching private keys. It then returns $\langle vk^*, C^*, \sigma^* \rangle$ to $\mathcal{M}$.

6. Any subsequent queries of $\mathcal{M}$ are answered as before.

7. Finally, $\mathcal{M}$ outputs a guess $b'$, the same guess is output by $\mathcal{M}'$.

Note that $\mathcal{M}'$ represents a legal adversarial strategy for attacking HIBE-$\Pi_i$; in particular, $\mathcal{M}'$ never requests the secret key corresponding to the “target” identity $\text{Encode}(\text{ID}_{i}^*) \circ (1vk^*)$. Furthermore, $\mathcal{M}'$ provides a perfect simulation for $\mathcal{M}$ until the event $\text{Forge}$ occurs, or it has to abort the game due to invalid requests by $\mathcal{M}$ (which happens only with probability $1/b$ in each game). It is thus easy to see that

$$\left| \Pr_{\mathcal{M}', \text{HIBE-}\Pi_i}^{\text{HIBE}} [\text{Succ}] - \frac{1}{2} \right| \geq \frac{b - 1}{b} \Pr_{\mathcal{M}, \Pi}^{\text{HIBE}} [\text{Succ} \land \text{Forge}] + \frac{1}{2} \Pr_{\mathcal{M}, \Pi}^{\text{HIBE}} [\text{Forge}] - \frac{1}{2}$$

$$= \frac{1}{2} \left| 2(b - 1) \Pr_{\mathcal{M}, \Pi}^{\text{HIBE}} [\text{Succ} \land \text{Forge}] + \Pr_{\mathcal{M}, \Pi}^{\text{HIBE}} [\text{Forge}] \right|,$$

and the left-hand side of the above is negligible by the assumed security of HIBE-$\Pi_i$.

4.5 Generic Construction of Certificateless Encryption Schemes

In this section we answer the question whether CCA security from identity-based encryption as described by Boneh, Canetti, Halevi and Katz [CHK04, BK05, BCHK07] also holds in the certificateless case affirmatively. This result make the construction of certificateless schemes in the standard model much easier and much more efficient. In this section, we provide a more generic conversion that
enables us to use these techniques to obtain IND-CCA2 security in a combination of IBE schemes. We then use the special case of one HIBE scheme and one IBE scheme run in parallel to construct efficient certificateless schemes in Section 4.6.

Based on the above generic constructions, we propose a new way to construct certificateless encryption schemes in the standard model, which is shown in Figure 4.4. Our construction uses one IND-CPA secure HIBE scheme, and one IND-CPA secure IBE scheme, and a strong one-time signature scheme or a MAC and an encapsulation scheme. We also use the masking construction introduced above to obtain maximum efficiency. The resulting construction will be more efficient than the Huang and Wong [HW07a] construction because the underlying encryption schemes do not have to be IND-CCA2 secure, and we use masking instead of double encryption. This makes the decryption process on modern multi-processor computers much faster, because both the encryption and the decryption can be performed in parallel. In the Huang and Wong [HW07a] construction, both encryption and decryption must be done sequentially, and can thus not directly benefit from multi-processor computers.

The idea behind the conversion is quite simple: Each user generates an additional IBE scheme, and the message is then encrypted under the HIBE scheme of the trusted authority and the IBE scheme that the user controls. Since most
IBE schemes prove only IND-CPA security and obtain IND-CCA2 security by the constructions described by Boneh et al. [BCHK07], it is now easy to “upgrade” existing schemes to the certificateless case. We can guarantee IND-CCA2 security with the constructions that are proven in the previous section.

In the certificateless case, there are two private keys per user (the ID-based private key $d_{ID}$ and the user-generated private key $usk_{ID}$, we discussed the details in Section 3.3). Additionally, there are two security models for outsider (Type I) and insider (= Key Generation Centre, Type II) adversaries. With respect to the challenge identity $ID^\ast$, outsider (Type I) adversaries are allowed to either corrupt the ID-based private key $d_{ID^\ast}$ or corrupt the user private key $usk_{ID^\ast}$ or replace the user public key $upk_{ID^\ast}$. They may however make at most one of these queries. Insider adversaries (Type II) already know the master private key $msk$ and are thus not allowed to corrupt the user private key $usk_{ID^\ast}$ or replace the certificateless public key $upk_{ID^\ast}$ of the challenge identity. The fact that the adversary may corrupt private keys and even replace public keys of the challenge identity seems to give the adversary tremendous power.

For certificateless encryption, there are currently three published schemes in the standard model. The first scheme is a generic way to construct an IND-CCA2 secure certificateless encryption scheme from any generic IND-CPA certificateless encryption scheme by Dent, Libert and Paterson [DLP08] using non-interactive zero knowledge proofs and double encryption. The resulting scheme is in the strong certificateless security model, and is very inefficient. The inefficiency results from the fact that non-interactive zero knowledge proofs in the standard model are computationally expensive and very long, and that encrypting a plaintext multiple times does not add to efficiency either. A concrete and efficient strongly secure scheme was proposed by the same authors in the same paper, which is the second scheme in the standard model. The third scheme is a generic composition of an IND-CCA2 secure ID-based scheme with an IND-CCA2 secure public key encryption scheme and a strong one-time signature scheme by Huang and Wong [HW07a]. The Huang and Wong scheme is in the weak certificateless security model and is not susceptible to the malicious KGC attack described by Au et al. [AMC+07], that we discussed in more detail in Section 3.2. Our constructions are also not susceptible to this kind of attack, as they are in the weak security model for certificateless encryption. If they were susceptible, then a HIBE scheme where the adversary replaced the public key would be sufficient to
gain an advantage in the game described in the proof of Theorem 4.4. However, replacing a public key does not help the adversary in winning the game.

We now give a generic construction of a hierarchical certificateless encryption scheme in the standard model. Concrete examples are discussed in Section 4.6 below. As discussed in Definition 3.3, a hierarchical certificateless encryption scheme HCLE consists of the PPT algorithms (CL-HIBE Setup, CL-HIBE KeyDer, HCLE User KeyGen, HCLE Enc, HCLE Dec). We proceed to describe the respective algorithms in detail for our construction.

HCL-IBE Setup(1^k) The trusted authority sets up an l + 1 level masking HIBE scheme, and reserves the last level of the scheme for verification keys. That is, valid identities have at most depth l. The HIBE setup algorithm outputs (param, mpk, msk). The authority sets up a secure encapsulation scheme (Init, S, R) and runs Init(1^k) to obtain pub. The authority publishes the security parameters for strong one-time MAC schemes (MAC, Verify) that are acceptable to use during encryption. The authority’s public key is PK_TA = (param, mpk, pub), the authority’s secret key is msk.

HCL-IBE KeyDer(ID) To derive a key for an identity ID, the authority computes d_ID = HIBE KeyDer(Encode(ID)) and returns d_ID. The definition of Encode is given in Section 4.4.

HCLE User KeyGen Each user selects a masking IBE that has the same message space as the HIBE published by the trusted authority. The user generates (param, upk, usk) ← IBE Setup(1^k), and sets PK-user = (param, upk) and publishes (PK_TA, PK-user) as his certificateless public key. Alternatively, the user may just generate a new master public key for the (u + 1)st level of the HIBE scheme set up by the trusted authority, where u is the user’s level in the HIBE scheme. This will most likely result in shorter public keys but potentially longer ciphertexts.

HCLE Enc(PK_TA, PK-user, ID, m) To encrypt a message to the user with identity ID in the scheme of the trusted authority, encapsulate a random value r by running (r, com, dec) ← S(1^k, pub). Then encrypt m ⊕ dec to Encode(ID)⊙(1com) under PK_TA to obtain C_TA = ⟨C_0, r_TA⟩. Generate a mask mask_user and a recovery value r_user from PK_user and set C = ⟨C_0 ⊗ mask_user, r_TA, r_user⟩. Compute tag ← Mac_ruser(C). The final ciphertext is ⟨com, C, tag⟩.
HCLE Dec$(sk, ID, \langle \text{com}, C, \text{tag} \rangle)$ To decrypt the ciphertext $\langle \text{com}, C, \text{tag} \rangle$ given the secret key $sk = \langle d_{ID}, usk_{ID} \rangle$ matching ID do the following. Derive the private key for ID $d_{TA} \leftarrow \text{HIBE KeyDer}(d_{ID}, \text{Encode}(ID) \circ (1\text{com}))$ using $d_{ID}$, and the private key for $d_{user} \leftarrow \text{IBE KeyDer}(usk_{ID}, 1\text{com})$ using $usk_{ID}$. Then compute $\text{mask}_{user}$ using $d_{user}$ and $r_{user}$, and recover $C'_0 = C_0 \odot \text{mask}_{user}$. Obtain $m \circ \text{dec}$ by computing $m \circ \text{dec} \leftarrow \text{HIBE Dec}((C'_0, r_{TA}), d_{TA})$. Reconstruct $r$ by computing $r \leftarrow R(\text{pub}, \text{com}, \text{dec})$. If $r \neq \perp$ and $\text{Verify}_r(C, \text{tag}) = 1$, then output $m$, else output $\perp$.

A similar construction is also possible for the signature-based algorithm discussed in Section 4.2.1.

**Theorem 4.5.** If the $(l+1)$-HIBE scheme is SID-IND-CPA secure, the user generated IBE is SID-IND-CPA secure, $(\text{Init}, S, R)$ is a secure encapsulation scheme, and $(\text{Mac}, \text{Verify})$ is a strong one-time message authentication code, then HCLE is a selective identity IND-CCA2 secure hierarchical certificateless encryption scheme.

**Proof.** Given an adversary HCLE-$M$ against SID-IND-CCA2 HCLE schemes as described in Definition 3.4, we show how to build an adversary MK-HIBE-$M$ against our IND-CCA2 secure MK-HIBE construction consisting of one $l + 1$-level HIBE and one 1-level HIBE (which is effectively an IBE), a strong one-time MAC and a secure encapsulation scheme.

By Theorem 4.2 and Theorem 4.4, this construction is IND-CCA2 secure. We have to make only two small changes to Theorem 4.2. The first change allows the creation of additional IBE schemes after the Setup phase of the experiment by the challenger. The second change is that decryptions can be requested to any subset of encryption schemes registered in the construction; i.e. one can specify a vector of encryption schemes that shall be used for decryption. It is evident that these changes does not affect the security of the encryption scheme for the following reasons.

- The encryption schemes are initially created under control of the challenger, therefore they will have reasonable security parameters.
- Since the challenger is able to decrypt ciphertexts under these additional IBE schemes (because the challenger created them), all security guarantees still hold.
• Reveal key and replace public key queries that an adversary may use as specified in Definition 4.3 can be answered in the usual way.

• Since any adversary has to commit to the target identity at the start of the experiment, the challenge ciphertext will be encrypted under the target identity and not under any of the IBE schemes that were created after the Setup phase of the experiment. Therefore, the additional IBE schemes that are registered have no effect on the challenge ciphertext.

We now describe now we simulate a selective-ID IND-CCA2 secure encryption scheme to an adversary HCLE-\mathcal{M} that runs against selective-ID IND-CCA2 secure certificateless schemes. At the start of the experiment, HCLE-\mathcal{M} outputs a target identity vector \( \vec{v}^* \). MK-HIBE-\mathcal{M} forwards \( \text{ID}^* = (\vec{v}^*, \{\}) \) (that is, \( \vec{v}^* \) and an empty vector) to the challenger B who will run the multi-key HIBE scheme. In return, MK-HIBE-\mathcal{M} is given a public key \( \text{PK}^* = (\text{param}^*_\text{HIBE}, \text{mpk}^*_\text{HIBE}, \text{param}^*_\text{IBE}, \text{mpk}^*_\text{IBE}) \), the description of a secure encapsulation scheme \( (\text{Init}, S, R) \) and a public key for the encapsulation scheme \( \text{pub} \), and a setup algorithm for acceptable strong one-time MAC schemes that can be used with the construction. MK-HIBE-\mathcal{M} sets \( \text{mpk} = (\text{param}^*_\text{HIBE}, \text{mpk}^*_\text{HIBE}) \) and \( \text{upk}_{\vec{v}^*} = (\text{param}^*_\text{IBE}, \text{mpk}^*_\text{IBE}) \). MK-HIBE-\mathcal{M} forwards \( \text{mpk} \) and \( \text{upk}_{\vec{v}^*} \), the description of the secure encapsulation scheme and \( \text{pub} \), and the security parameters that are acceptable for encryption with a strong one-time MAC to HCLE-\mathcal{M}. Additionally, MK-HIBE-\mathcal{M} sets up a table of identity vectors \( \vec{v} \) and corresponding public keys \( \text{mpk}_{\vec{v}} \) and private keys \( \text{msk}_{\vec{v}} \) (in the following referred to as the “table of identity vectors”). It records \( \vec{v}^* \) and \( \text{upk}_{\vec{v}^*} \) in the table. It then answers HCLE-\mathcal{M}’s queries as follows.

Reveal master key: MK-HIBE-\mathcal{M} asks the “reveal key” query for the HIBE scheme. Since MK-HIBE-\mathcal{M} may corrupt either the key for the HIBE scheme or the key for the IBE scheme, this query is only allowed if the “reveal user secret key(\( \vec{v}^* \))” was not asked before, which matches the descriptions of Type I and Type II IND-CCA2 HCLE adversaries from Definition 3.4.

Reveal ID-based key(\( \vec{v} \)): MK-HIBE-\mathcal{M} relays these queries to its challenger B, as they are allowed in an MK-HIBE scheme. Notice that this query is allowed in the MK-HIBE even for \( \vec{v}^* \) if “reveal user secret key(\( \vec{v}^* \))” is not asked.
Get user public key($\vec{v}$): If $\vec{v}$ is not in the table of identity vectors, then MK-HIBE-$\mathcal{M}$ requests the creation of an additional IBE scheme as discussed above. In return, it receives $(\text{param,mpk})$, records them as $\vec{v}$, $(\text{param}_{\vec{v}}, \text{mpk}_{\vec{v}})$ in the table of identity vectors, and returns $(\text{param}_{\vec{v}}, \text{mpk}_{\vec{v}})$ to HCLE-$\mathcal{M}$. Otherwise, it just returns the matching entry from the table.

Replace public key($\vec{v}$): MK-HIBE-$\mathcal{M}$ relays this query to its challenger. Notice, that according to Definition 3.4, in the case of $\vec{v} = \vec{v}^*$, “reveal ID-based key($\vec{v}^*$)” / “reveal master key” and “reveal user secret key($\vec{v}^*$)” cannot be asked by HCLE-$\mathcal{M}$ in one game. MK-HIBE-$\mathcal{M}$ changes the entries in its table of identity vectors accordingly.

Reveal user secret key($\vec{v}$): If $\vec{v}$ is in the table of identity vectors, then MK-HIBE-$\mathcal{M}$ forwards the query to its challenger returns the answer of its challenger. Otherwise, an entry is generated via the “get user public key” query above, and then the query is forwarded to MK-HIBE-$\mathcal{M}$’s challenger and the answer is returned. Since a certificateless adversary is not allowed to ask both “reveal ID-based key” and “reveal master key” / “reveal ID-based secret key” for the target identity, these queries can always be answered.

Decrypt($\vec{v}, C$): MK-HIBE-$\mathcal{M}$ looks up the IBE matching $\vec{v}$ in its table of identities, and then requests a decryption under that IBE and the identity $\vec{v}$ in the initially registered HIBE and returns the result to HCLE-$\mathcal{M}$. This is possible because MK-HIBE-$\mathcal{M}$ can request decryptions under a vector of encryption schemes, not necessarily only the initial combination of schemes.

Decrypt($\vec{v}, C, usk_{\vec{v}}$): MK-HIBE-$\mathcal{M}$ looks up the IBE matching $\vec{v}$ in its table of identities, submits $usk_{\vec{v}}$ as the private key corresponding to the public key for that IBE to its challenger, and then returns the result of “decrypt($\vec{v}, C$)”.

Get challenge encryption($\vec{v}^*, m_0, m_1$): This query is forwarded to $\mathcal{B}$, and the answer relayed to HCLE-$\mathcal{M}$.

Since MK-HIBE-$\mathcal{M}$ is able to answer all queries of HCLE-$\mathcal{M}$, the simulation is perfect, and we can say that our construction fulfills the requirements of a selective-identity IND-CCA2 secure hierarchical certificateless encryption scheme.
Theorem 4.3 shows that masking encryption schemes do not change the security of the construction. We use therefore masking encryption schemes as they are computationally more efficient and provide shorter ciphertexts.

We note that our construction is more powerful than a HCLE, because an HCLE would only support one trusted authority. Using our construction and the modification above, we are able to construct certificateless encryption schemes that support multiple trusted authorities and do not require any collaboration of the authorities as it would be required if the master key was split by a secret sharing scheme or a trusted teller.

## 4.6 Concrete Certificateless Encryption Schemes

We describe the phases of our certificateless encryption schemes in this section. Each scheme consists of the 5 algorithms CL-IBE Setup, CL-IBE KeyDer, CLE User KeyGen, CLE Enc, and CLE Dec).

### 4.6.1 Lattice-Based CL-PKE Algorithms

This scheme is a conversion from the HIBE in the standard model recently published by Peikert [Pei09a]. Peikert’s scheme is again based on a KEM mechanism. We first describe an alternative KEM mechanism where there are two public keys used for encryption (which later correspond to the IBE master key and the user key in the CL-HIBE). One of the public keys can be corrupted and the KEM is still CPA secure.

The KEM in Peikert [Pei09a] is parametrized by the modulus $q$, the lattice dimension $m$, the key length $l$, and a bound $\tilde{L}$ that determines the error distribution $\chi$ used for encapsulation. The scheme is based on the Learning With Error (LWE) problem. The KEM also uses an algorithm Invert described in [Pei09b, Section 4.2.1], which recovers $\vec{s} \in \mathbb{Z}_q^n$ given $\vec{A}$ and a suitably short basis $\vec{S}$ of $\Lambda^\perp(\vec{A})$ which is generated by TrapGen as described by Alwen and Peikert [AP09]. The algorithms are stated in Figure 4.5.

**Correctness and Security**

**Lemma 4.1.** Let $q \geq 2L\sqrt{m}$ be prime and $\chi = \Psi_{\alpha}$ for $1/\alpha \geq \tilde{L} \cdot \omega(\sqrt{\log n})$. Then Decaps is correct with overwhelming probability over the randomness of
Chapter 4. Multi-Authority IBE and Certificateless Encryption Schemes

Gen($k$):
\[(\vec{A}, \vec{S}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{m \times m} \leftarrow \text{TrapGen}(\vec{L})\]
\[(\vec{A}', \vec{S}') \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{m \times m} \leftarrow \text{TrapGen}(\vec{L})\]
\[\vec{U} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n \times l}\]
\[pk \leftarrow (\vec{A}, \vec{A}', \vec{U})\]
\[sk \leftarrow (\vec{S}, \vec{S}', \vec{A}, \vec{A}')\]

Return(pk, sk)

Encap($pk$):
\[(\vec{A}, \vec{A}', \vec{U}) \leftarrow pk\]
\[\bar{k} \stackrel{\$}{\leftarrow} \{0, 1\}^l\]
\[(\bar{s}, \bar{s}') \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n\]
\[(\bar{x}, \bar{x}') \stackrel{\$}{\leftarrow} \chi^n\]
\[\bar{b} = \vec{A}' \bar{s} + \bar{x}\]
\[\bar{y} = \vec{A}' \bar{s}' + \bar{x}'\]
\[\bar{p} = \vec{U}' \bar{s} + \vec{U}' \bar{s}' + [q/2] \bar{k} + \bar{y}\]
\[C = (\bar{b}, \bar{b}', \bar{p})\]

Return($\bar{k}, C$)

Decap($C, sk$):
\[(\vec{S}, \vec{S}', \vec{A}, \vec{A}') \leftarrow sk\]
\[(\bar{b}, \bar{b}', \bar{p}) \leftarrow C\]
\[\bar{s} = \text{Invert}(b, S, A)\]
\[\bar{s}' = \text{Invert}(b', S', A')\]
\[\bar{k} = \bar{p} - \vec{U}' \bar{s} - \vec{U}' \bar{s}' = (k_1, \ldots, k_l)^l\]
\[\bar{k}_i = \begin{cases} 1 & \text{if } \frac{q}{4} \leq k_i \text{ mod } q < \frac{3q}{4} \\ 0 & \text{if } -\frac{q}{4} \leq k_i \text{ mod } q < \frac{q}{4} \end{cases}\]

Return($\bar{k}$)

Figure 4.5: Our lattice-based two-key KEM.

**Encaps.** Moreover, there exists a PPT oracle algorithm (a reduction) $S$ attacking the LWE$_{q,\chi}$ problem such that, for any adversary $M$ mounting an IND-CPA attack on KEM,

\[\text{Adv}_{\text{LWE}_{q,\chi}}(S^M) \geq \frac{1}{2} \text{Adv}^{\text{IND-CPA}}_{\text{KEM}}(M) - \text{negl}(n)\]

**Correctness.** For a distribution $\chi$ on $\mathbb{Z}_q$, define $\chi^k$ as the distribution obtained by summing together $k$ samples from $\chi$, where addition is performed in $\mathbb{Z}_q$.

Let $\delta > 0$. Assume that for any $w \in 0, 1, \ldots, m$, $\chi^k$ satisfies that

\[\Pr_{e \sim \chi^k} \left[ |e| < \left[ \frac{p}{2} \right]/2 \right] > 1 - \delta \tag{4.2}\]

where $\delta \sim 2^{-k}$ for security parameter $k$.

Consider first an encryption of a bit 0 for a vector $\bar{k}$ of length 1. In this case, $U$ is a vector in $\mathbb{Z}_q$ with length $n$. The encryption is given by $\bar{b} = \vec{A} \bar{s} + \bar{x}$, $\bar{b}' = $
Thus, bounded by \( \nu \) replace \( M \) that challenge encapsulation. Without loss of generality, for the following we assume and repeats the process. During the game, the adversary \( M \) query correctly with probability 1. Thus, \( Z \) described by Alwen and Peikert [AP09]. Thus, \( B \)TrapGen and constructs the respective lattice and its basis with the algorithm \( B \) for \( \nu \) supply by \( M \) and sets \( S \). The same argument holds for an encryption of \( k \rightarrow 0 \) of length \( l \), we have that \( k \rightarrow 0 \) can be recovered with probability \( (1 - \delta)l \), which is non-negligible if \( \delta \) is negligible and \( l \) is large. (It is easy to see that \( \lim_{x \to \infty} \frac{1}{1 - \delta} \) converges towards 1 considering \( \lim_{x \to \infty} a(x)^{1-x} = 1 \) for \( \lim_{x \to \infty} a(x) = 1 \). Here \( a(x) = 1 - 2^{-x} \), thus the difference for larger key sizes is negligible. Actually, for \( x = 60 \) (thus with very low security of 60 bits), the difference between the functions is below \( 5.07 \cdot 10^{-15\%} \).

**Security.** Given an adversary \( \mathcal{M} \) that can distinguish an encryption of a randomly drawn key \( k \rightarrow 0 \) from an encryption of a random key \( k \rightarrow 0 \) we build a distinguisher \( B \) for \( \text{LWE}_{\mathcal{X}, \mathcal{X}} \).

Let \( \text{Adv}_{\text{KEM}, \text{IND-CPA}}(\mathcal{M}) \) be the probability that \( \mathcal{M} \) distinguishes an encryption of \( k \rightarrow 0 \) from an encryption of \( k \rightarrow 0 \) under a random public key \( (A, A', U) \), where \( k \rightarrow 0 \) is chosen by the simulator and given to \( \mathcal{M} \) together with the challenge encapsulation.

During the game, the adversary \( \mathcal{M} \) may choose to corrupt either \( A \) and thus learn \( S \) or choose to corrupt \( A \) and learn \( S \) or choose to replace one of \( A \) or \( A \) with a public key of its own. Before the game starts, the simulator \( B \) for \( \mathcal{M} \) randomly guesses the secret key that \( \mathcal{M} \) may learn during the game and constructs the respective lattice and its basis with the algorithm \( \text{TrapGen} \) described by Alwen and Peikert [AP09]. Thus \( B \) is able to answer the corrupt query correctly with probability 1/2. If \( B \)'s guess was wrong, \( B \) aborts the game and repeats the process. During the game, the adversary \( \mathcal{M} \) will request a challenge encapsulation. Without loss of generality, for the following we assume that \( \mathcal{M} \) will choose to learn \( S \) and thus corrupt \( A \) or alternatively choose to replace \( A \) with a lattice of its own.

Given an oracle \( \mathcal{O} \) that outputs a distribution \( R \) with elements \( (\mathcal{X}, d) \in \mathbb{Z}_q^r \times \mathbb{Z}_q \) which are either from \( \overrightarrow{A} \rightarrow \mathcal{X} \) for fixed \( \overrightarrow{s} \) or randomly drawn from \( \mathbb{Z}_q^r \times \mathbb{Z}_q \), \( B \) constructs \( \overrightarrow{A}, \overrightarrow{S} \) with \( \text{TrapGen} \), draws \( l \) samples \( (\mathcal{X}, d) \) for \( i = 1, \ldots, l \) from \( R \) and sets its public key to \( (A, A', U) \). During the game, \( \mathcal{M} \) may request to replace \( \overrightarrow{A} \). \( B \) then replaces its current \( \overrightarrow{A} \) with the one supplied by \( \mathcal{M} \) and sets \( S = \bot \). When \( \mathcal{M} \) requests the challenge encapsulation,
\[ B \text{ randomly selects } k \overset{\$}{\leftarrow}\{0,1\}^t, s \overset{\$}{\leftarrow}\mathbb{Z}_q^t \text{ and } x \overset{\$}{\leftarrow}\chi^n. \]

\[ B \text{ computes } b' = A' t s' + x' \text{ and } u = U t s'. \]

\[ B \text{ constructs the vector } p \text{ of the challenge ciphertext as follows: } \]

\[
\begin{cases} 
\vec{k}_i = 0 : \vec{p}_i = d_i + u_i \\
\vec{k}_i = 1 : \vec{p}_i = d_i + u_i + \lfloor q/2 \rfloor
\end{cases}
\]

Then \( B \) outputs \((b, b', p)\) as the challenge encapsulation and \( k \) as the challenge key to the adversary \( M \). Once \( M \) outputs its guess for \( k \), \( B \) outputs the same guess for \( R \). We argue that \((b, b', p)\) is a valid encapsulation of \( k \) and that the advantage of \( M \) is also the advantage of \( B \) in distinguishing the distributions if \( B \) does not have to abort during the game. To see this, notice that \( p \) encrypts a random key.

The term \( \text{negl}(n) \) in Lemma 4.1 comes from the fact that in the real world, the scheme is not set up with \( A \overset{\$}{\leftarrow}\mathbb{Z}_n^X \times \mathbb{Z}_q^X \) but with \( A \overset{\$}{\leftarrow}\text{TrapGen}(\tilde{L}) \). Because \( A \overset{\$}{\leftarrow}\text{TrapGen}(\tilde{L}) \) has a \( \text{negl}(n) \) deviation from \( A \overset{\$}{\leftarrow}\mathbb{Z}_n^X \times \mathbb{Z}_q^X \), we have to add this error to the success probability of the simulator. \( \square \)

**From Two-Key KEM to HCLE**

We extend the binary tree encryption (BTE) scheme from Peikert [Pei09a] to the certificateless case and show that the scheme is secure even if the adversary learns either the HIBE master secret key or alternatively corrupts the certificateless key of the user or replaces the certificateless key of the user. In the proof, we will use the proof strategy discussed in Section 3.6. The algorithms of our CL-HIBE scheme are described in Figure 4.6. Besides the algorithms that are also used in the two-key KEM, the underlying HIBE construction by Peikert uses the algorithms \text{ExtBasis}(\vec{S}, \vec{A}' = \vec{A}||\vec{A}) \) described in [Pei09a, Section 3.3] and \text{RandBasis}(\vec{S}, \vec{s}) \) described in [Pei09a, Section 3.4]. Additionally, the scheme uses the same parameters as the BTE scheme from Peikert, i.e. it has dimensions \( m_1, m_2 = O(n \lg q) \). Additionally, for an identity at depth \( 0 \leq i \leq d \) the parameters are:

- The dimension of the lattice used for encryption to the identity is \( m_1 + (i + 1)m_2 \).
- The upper bound on the the Gram-Schmidt lengths of its secret short basis \( \tilde{L}_i \).
4.6. Concrete Certificateless Encryption Schemes

- For \( i \geq 1 \), the Gaussian parameter \( s_i \) used to generate that secret basis, where \( \forall j < i : s_i \geq \tilde{L}_j\omega(\sqrt{\log n}) \)

As the KEM itself is not suitable to encrypt a message \( m \), we use the output of the KEM as input to an IND-CPA secure DEM. Thus we get an IND-CPA secure encryption scheme as shown by Herranz, Hofheinz and Kiltz [HHK06].

Proof of Security for the Lattice-Based CL-HIBE

Theorem 4.6 (Security of Lattice CL-HIBE). There exists a ppt algorithm \( B \) attacking KEM (instantiated with dimension \( m = m_1 + (d + 1)m_2 \) and \( q, \chi \) as in BTE) such that, for any adversary \( M \) mounting an attack on BTE,

\[
\text{Adv}_{\text{KEM}}(B^M) \geq \frac{1}{2}\text{Adv}_{\text{BTE}}^{\text{id-ind-cca}}(M) - \text{negl}(n)
\]

The proof by Peikert [Pei09a] uses the principles of “undirected growth” and “controlled growth” explained in [Pei09a, Section 3] to relate the probability of an adversary against the HIBE construction (HIBE-\( M \)) to a probability of an adversary against the KEM construction (KEM-\( M \)). KEM-\( M \) receives the KEM public key \((\vec{A}, \vec{U})\), an encapsulation \((\vec{b}, \vec{p})\) and a key \( k \in \{0, 1\}^l \). Since Peikert’s HIBE scheme is only selective identity secure, the HIBE-\( M \) has to output a target identity vector \( \vec{v}^* \) before it receives the system parameters and the public keys. KEM-\( M \) sets the public key of identity vector \( \vec{v}^* \) to be the KEM’s public key, and then uses the controlled growth technique to construct lattices from suitable parts of \( \vec{A} \) so that a short basis for the respective lattice is known. Therefore, KEM-\( M \) can simulate all decryption and private key queries for HIBE-\( M \), except for the target identity. KEM-\( M \) then uses \((\vec{b}, \vec{p})\) as the challenge ciphertext and thus wins its game whenever HIBE-\( M \) wins its game. We refer to Peikert [Pei09a] for the details of the ID-based proof. In the following, we list the changes that are needed for the proof in the certificateless case following the principle explained in Section 3.6.

For Type I adversaries that want to learn the CL-key, we use the proof from Peikert [Pei09a] unmodified and hand over the user secret value usk\( \text{id} \) to the adversary. The original proof does still hold in this setting. Alternatively, if the adversary chooses to replace the certificateless public key, the challenge ciphertext can still be computed in the KEM/DEM framework, and the security still relies on the ID-based secret key used for encapsulation.
Chapter 4. Multi-Authority IBE and Certificateless Encryption Schemes

CL-HIBE Setup\((d, k)\):
\((\vec{A}_0, \vec{S}_0) \in (\mathbb{Z}_q^{n \times (m_1 + m_2)}, \mathbb{Z}_q^{n \times n}) \leftarrow \text{TrapGen}(\vec{L}_0)\)
for \((j = 1, \ldots, d)\); do
\((\vec{A}_0^j, \vec{A}_1^j) \leftarrow \mathbb{Z}_q^{n \times m_2}\)
\((\vec{A}_0^j, \vec{A}_1^j) \leftarrow \mathbb{Z}_q^{n \times m_2}\)
done
\(\vec{U} \leftarrow \mathbb{Z}_q^{n \times 1}\)
\(\text{mpk} = \vec{A}_0\)
\(\text{param} = (\vec{A}_0^0, \vec{A}_1^0, \ldots, \vec{A}_0^d, \vec{A}_1^d, \vec{U}, d, \vec{L}_0, \ldots, \vec{L}_d)\)
\(\text{msk} = \vec{S}_0\)
\(\text{Return}(\text{param, mpk, msk})\)

CL-HIBE KeyDer\((\text{param}, d, \text{ID}, \text{ID}') = \text{ID}'||\text{ID})\)
if \(\|\text{ID}'\| > d\) Return(⊥)
\(\vec{t} = \vec{t} = \text{ID}\)
\(s_{\vec{t}} \geq \vec{L}_t \omega(\sqrt{\log n})\)
\(\vec{S}_{\text{ID}'} \leftarrow \text{RandBasis}(\text{ExtBasis}(\vec{S}_{\text{ID}'}, \vec{A}_{\text{ID}'}, s_{\vec{t}}))\)
\(\text{Return}(\vec{S}_{\text{ID}'})\)

CLE Enc\((\text{param, mpk, upk}_{\text{ID}}, \text{ID}, m)\)
\(\vec{A}_0 \leftarrow \text{mpk}; \langle \vec{A}_{\text{upk}_{\text{ID}}}, \vec{A}_{\text{upk}_{\text{ID}}, 1}, \vec{A}_{\text{upk}_{\text{ID}, 1}} \rangle \leftarrow \text{upk}_{\text{ID}}\)
\((\vec{A}_0^0, \vec{A}_1^0, \ldots, \vec{A}_0^d, \vec{A}_1^d, \vec{U}, d) \leftarrow \text{param}\)
\(\vec{A}_{\text{TA}} = \vec{A}_0 || \vec{A}_1^0 || \ldots || \vec{A}_1^d\)
\(\vec{A}_{\text{user}} = \vec{A}_{\text{upk}_{\text{ID}}} || \vec{A}_{\text{upk}_{\text{ID}, 1}}\)
\((\kappa, \sigma) \leftarrow \text{KEM.Encaps}(\vec{A}_{\text{TA}}, \vec{A}_{\text{user}}, U)\)
\(e \leftarrow \text{DEM.Encaps}(\kappa, \sigma)\)
\(C = (\sigma, e)\)
\(\text{Return}(C)\)

CLE User KeyGen\((\text{param, mpk})\)
\(\langle \vec{A}, \vec{S} \rangle \leftarrow \text{TrapGen}(\vec{L}_0)\)
\((\vec{A}_0^0 \leftarrow \mathbb{Z}_q^{n \times m_2})\)
\((\vec{A}_1^0 \leftarrow \mathbb{Z}_q^{n \times m_2})\)
\(\text{upk}_{\text{ID}} = \langle \vec{A}, \vec{A}_0^0, \vec{A}_1^0 \rangle\)
\(\text{usk}_{\text{ID}} = \vec{S}\)
\(\text{Return(\text{upk}_{\text{ID}}, \text{usk}_{\text{ID}})}\)

CLE Dec\((\text{param, (d, usk}_{\text{ID}}), \text{ID}, C)\)
\(\vec{S}_{\text{ID}} \leftarrow d\)
\(\vec{S}_{\text{ID}}' \leftarrow \text{usk}_{\text{ID}}\)
\((\sigma, e) \leftarrow C\)
\(\kappa \leftarrow \text{KEM.Decaps}(\vec{S}_{\text{ID}}, \vec{S}_{\text{ID}}', \sigma)\)
\(m \leftarrow \text{DEM.Dec}(\kappa, e)\)
\(\text{Return}(m)\)

Figure 4.6: Our lattice-based CL-HIBE
We have shown that the two-key KEM construction is CPA secure if either of the keys that are used as input is corrupted or replaced by the adversary. Thus the KEM is also CPA secure in the certificateless setting if the master private key is corrupted by the adversary or if the adversary wishes to learn the ID-based private key $d^*_{\text{ID}}$. In this case, the simulator embeds the challenge into the user generated public key for $\text{ID}^*$ and constructs the ID-based protocol “by the book”. Since the simulator knows the target identity that the adversary will attack in advance but has to guess the corruption strategy of the adversary, we get a security reduction of $\frac{1}{2}$ with respect the ID-based case. In practice this means that we have to set up the scheme such that the keys output by the KEM are 1 bit longer and the remaining security parameters are adapted accordingly.

Chosen ciphertext security (CCA) follows from the constructions in Section 4.2. The encryption algorithm in the construction is already modified so that the encryption is performed to two HIBE schemes: one under the control of the trusted authority, and one under control of the user ($\overrightarrow{A}_{\text{TA}}$ and $\overrightarrow{A}_{\text{user}}$ in the encryption algorithm). To use the CCA construction from Section 4.2, the user just has to derive the correct key for the verification key and then decrypt with it (by using the HIBE KeyGen algorithm for either $\text{com}$ or $\text{vk}$ with $\text{usk}_{\text{ID}}$ and its $\overrightarrow{A}_{\text{upk}_{\text{ID}},1}^0, A_{\text{upk}_{\text{ID}},1}^1$). We already “cut” the user’s HIBE scheme down to one level, and that level is used only for CCA security, so that there is as little as possible overhead. We note that all of the other variations of the encryption scheme given in Peikert [Pei09a, Section 5.2.1] are also applicable for our construction.

### 4.6.2 Pairing-Based HCLE

We review the selective ID secure construction from [BBG05, Section 2], and add the algorithms that are necessary to get an IND-CCA2 secure certificateless encryption scheme according to the conversion given in Section 4.5. That is, we use a 2-level HIBE at the trusted authority to obtain an IND-CCA2 secure IBE scheme and a 1-level HIBE at the user to obtain a CCA2 secure encryption scheme. Both schemes will use the construction given in Section 4.2.2 to become IND-CCA2 secure.

**Converting the Selective ID-Based HIBE from [BBG05]**

Bonel, Boyen and Goh construct the master public key by picking a generator $g$ of $G$ of order $p$, then selecting $g_2 \leftarrow G$, $\alpha \leftarrow Z_p$, and letting $g_1 = g^\alpha$. The public
Chapter 4. Multi-Authority IBE and Certificateless Encryption Schemes

**CLE Setup**(1\textsuperscript{st})

\[
\begin{align*}
\text{G}, \text{G}_T, p & \leftarrow (1^k) \\
g & \in \text{G} \text{ generator of } \text{G} \\
\alpha & \sim \mathbb{Z}_p \\
(g_2, g_3, h_1, h_2) & \sim \mathbb{G}_T^k \\
E & = e(g^\alpha, g_2) \\
\text{mpk} & = \langle \text{E}, g_3, h_1, h_2 \rangle \\
\text{msk} & = g_2^\alpha \\
\text{H} & \sim (1^k) \\
\text{MAC} & \sim (1^k) \\
\text{param} & = \langle \text{G}, \text{G}_T, p, g, \text{pub}, \text{MAC}, H \rangle \\
\text{Return}(\text{param}, \text{mpk}, \text{msk})
\end{align*}
\]

**CLE Enc**(param, mpk, upk\textsubscript{ID}, ID, m)

\[
\begin{align*}
\langle \text{E}, g_3, h_1, h_2 \rangle & \leftarrow \text{mpk} \\
\langle h, t \rangle & \leftarrow \text{pub} \\
\langle \text{E}_{\text{upk}}, \text{g}_\text{ID}, \text{g}_\text{upk}, h, t \rangle & \leftarrow \text{upk}\text{ID} \\
\langle s, s' \rangle & \sim \mathbb{Z}_p^2 \\
\text{dec} & \sim \{0, 1\}^{3k} \\
C & = \left\langle E^s \cdot E_{\text{upk}}^{s'}, (m \circ \text{dec}), g^s, (h_1^s \cdot h_2^{H(\text{dec})} \cdot g_3), g_\text{ID}^s \cdot (h_{\text{upk}}^{H(\text{dec})} \cdot g_\text{upk})^{s'} \right\rangle \\
r & \leftarrow h(\text{dec}), \text{com} & \leftarrow H_{\text{ID}}(\text{dec}), \text{tag} & \leftarrow \text{MAC}_x(C) \\
\text{Return}(\langle \text{com}, \text{C}, \text{tag} \rangle)
\end{align*}
\]

**CLE Dec**(param, mpk, upk\textsubscript{ID}, (d\textsubscript{ID}, usk\textsubscript{ID}), ID, (com, C, tag))

\[
\begin{align*}
\langle a_0, \text{TA}, a_1, \text{TA}, b_2 \rangle & \leftarrow d_\text{ID} \\
\langle a_0, \text{com}, a_1, \text{com} \rangle & \leftarrow \text{CL-IBE KeyDer}(\text{param, upk}\text{ID}, \text{usk}\text{ID}, \text{com}) \\
\langle a_0, \text{ID}, a_0, \text{ID}, \text{com} \rangle & \leftarrow \text{CL-IBE KeyDer}(\text{param, mpk, d}\text{ID}, \text{com}) \\
\langle A, B, C, D, E \rangle & \leftarrow C \\
m & \circ \text{dec} = \text{A} \cdot \frac{e(a_0, \text{com}, D)^{e(a_0, \text{com}, E)} \cdot e(B, \text{com}, \text{C}^x, \text{com})}{e(B, \text{com}, \text{C}^x, \text{com})} \\
\langle \text{G}, \text{G}_T, p, g, \text{pub}, \text{MAC}, H \rangle & \leftarrow \text{param} \\
\langle h, t \rangle & \leftarrow \text{pub} \\
\langle \text{E}_{\text{upk}}, \text{g}_\text{ID}, \text{g}_\text{upk}, h, t \rangle & \leftarrow \text{upk}\text{ID} \\
r & \leftarrow R(h, t \oplus t'), \text{com, dec} \\
\text{if Verify}(C, \text{tag}); \text{then} \\
\text{Return}(m) \\
\text{fi} \\
\text{Return}(\bot)
\end{align*}
\]

Figure 4.7: The CLE version of the HIBE in [BBG05].
4.6. Concrete Certificateless Encryption Schemes

key is \( g_1, g_2 \), the private key is \( g_2^\alpha \). During the encryption process, \( e(g_1, g_2) \) is computed. They note that this could alternatively be precomputed and included in the system parameters, and we prefer that approach. Thus the master public key is \( mpk = e(g_1, g_2) \in G_T \), the master secret key is \( msk = g_2^\alpha \in G \), and the system parameters are \( \text{param} = (g_3, h_1, \ldots, h_l) \in G^{l+1} \), which are chosen randomly.

To obtain an IND-CCA2 secure certificateless encryption scheme, we need a two-level HIBE scheme at the trusted authority, which means that \( l = 2 \). The only modification that we introduce into the scheme are the \text{User KeyGen} algorithm to generate certificateless keys and modified \text{CLE Enc} and \text{CLE Dec} algorithms as explained in Section 4.5. The algorithms are listed in Figure 4.7. Note that we modify the setup algorithm slightly, because we include the parameters for the MAC-based CCA construction. We use the encapsulation scheme proposed in [BCHK07, Section 7.2] and is listed in Section 2.5, which is essentially a hash function. In the \text{User KeyGen} algorithm, the user sets up his/her own IBE, and since every 1-level HIBE is trivially an IBE, he/she uses the BBG HIBE with \( l = 1 \). The user picks a random \( usk \overset{\$}{\leftarrow} Z_p \) and random \( (g_{ID}, g_{upk}, h_{upk}) \in G^2 \).

The user public key is \( upk = \langle e(g, g_{ID})^{usk}, g_{upk}, h_{upk} \rangle \). The user private key is \( usk \).

Correctness of encryption and decryption of the derived certificateless scheme is trivially shown. To see that the scheme fulfills the notion of SID-IND-CPA-CLE, notice that we use the masking HIBE approach as described in Section 4.3. We claim that a simulator \( B \) who uses an efficient adversary \( M \) against the SID-IND-CPA-CLE derived from the Boneh, Boyen, and Goh IBE [BBG05] has an advantage against the \( l\text{-wBDHP} \) problem. See the proof by Boneh, Boyen and Goh [BBG05] for a proof in the ID-based case. Following the proof strategy explained in Section 3.6, we only list the required changes for the certificateless case here.

If the adversary wants to learn the certificateless private key, or replace the certificateless key, then the proof of the IBE scheme does still hold, because even with a replaced certificateless public key for \( ID^* \) we are able to generate a valid challenge ciphertext

\[
C^* = (M_b \cdot T \cdot e(y_1, h^\gamma) \cdot upk^{s'}, h, h^\delta + \sum_{i=1}^l I_i^* \cdot \gamma_i, g^{s'}, (h_{upk}^{2T} g_{upk})^{s'}). 
\]

Notice, that as described in Section 4.3, this is a valid re-randomization of the \( C^* \) presented in [BBG05], thus if \( T = e(g, h)^{a+1} \) then \( C^* \) is a valid encryption of
Otherwise, it is independent of $M_b$. Thus, the proof given in [BBG05] carries through to the certificateless case if the adversary wants to corrupt/replace the certificateless key. However, if the adversary wishes to learn the master secret key $msk$, then we set up the scheme “by the book” and set the user public key to $upk = e(g^n, g^b)$. Notice, that we are now able to give the master secret key to the adversary and therefore are also able to give $d_{ID}^u$ to a Type I adversary. The challenge is now embedded as

$$C^* = (M_b \cdot mpk^a \cdot T, g^a, (h_1^{ID_1} \cdots h_k^{ID_k} \cdot g_3)^a, h, h^{\frac{1}{k} + I_{1^k}})$$

which is again a valid ciphertext if $T = e(g, h)^{\alpha l_{1^k}}$.

Because the simulator $B$ has to decide in advance which strategy the adversary $M$ may take, the security reduction suffers by a factor of $1/2$ which can usually be fixed by generating system parameters that use $1^{k+1}$ instead of $1^k$ during the setup phase. CCA security follows again from the constructions in Section 4.2. Therefore, the advantage of an adversary against our pairing-based HCLE construction is

$$\text{Adv}_{l-wBDHI}^*(B^M) \geq \frac{1}{2} \text{Adv}_{HCLE}^{\text{sid-ind-cpa}}(M).$$

### 4.7 Efficiency Considerations

There are currently three more or less generic constructions for certificateless encryption in the standard model. The only efficient generic constructions are the constructions by Huang and Wong [HW07a], and by Dent [Den08]. Both efficient constructions require a message to be encrypted by an IND-CCA2 secure IBE scheme and an IND-CCA2 secure PKE scheme. The resulting encryption has to be signed using a strong one-time signature scheme. In comparison, we require only two IND-CPA secure IBE schemes, a hash (the efficient encapsulation mechanism presented in Section 2.5), and a MAC for our construction. To summarize: we save two times IND-CCA2 security in the encryption schemes, our ciphertexts are shorter, and we use a hash and a MAC as opposed to a strong one-time signature scheme for message integrity.

The other inefficient generic construction is by Dent, Libert and Paterson [DLP08]. It uses an IND-CPA secure certificateless scheme to start with and does not explain how to obtain one in the standard model from an identity-
4.7. Efficiency Considerations

<table>
<thead>
<tr>
<th>Scheme</th>
<th>KeyGen</th>
<th>Enc</th>
<th>Dec</th>
<th>Keysize</th>
<th>Ciphertext overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#pairings + #\text{multi,regular, fixed-base-exp}</td>
<td>pk overheard</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pairing-based</td>
<td>0 + [4,1,0]</td>
<td>3 + [1, 0, 0]</td>
<td>3 + [1, 0, 0]</td>
<td>7 + 2</td>
<td>ID</td>
</tr>
<tr>
<td>[DLP08] (2 CPU)</td>
<td>0 + [4,1,0]</td>
<td>2 + [1, 0, 0]</td>
<td>2 + [1, 0, 0]</td>
<td>7 + 2</td>
<td>ID</td>
</tr>
<tr>
<td>Ours (2 CPU)</td>
<td>2 + [1,3,0]</td>
<td>0 + [2, 3, 0]</td>
<td>4 + [2,2,0]</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Ours (2 CPU)</td>
<td>2 + [1,3,0]</td>
<td>0 + [1, 2, 0]</td>
<td>2 + [1,1,0]</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4.2: Efficiency comparison of our generic construction with existing schemes.

(2 CPU) means parallel execution on 2 CPU’s wherever possible. Keysize and ciphertext overhead are measured in group elements.

Based encryption scheme. Then it uses two additional public key encryption schemes and a non-interactive zero knowledge proof. Since non-interactive zero knowledge proofs are not very efficient, the authors do not even bother to give an example scheme using their generic construction. Due to the fact that the message has to be encrypted three times, and the inefficiency of non-interactive zero knowledge proofs, we expect that our generic construction will be vastly more efficient than their generic construction. They do however give an efficient construction for a certificateless encryption scheme in the standard model based on bilinear pairings, which has not much in common with the generic construction that they propose. In Section 4.6, we constructed efficient schemes based on our construction. Table 4.7 gives an overview of the comparison of our schemes with the two generic constructions by Huang and Wong [HW07a] and Dent [Den08], and the concrete construction by Dent et al. [DLP08]. A detailed explanation for the table is below.

### 4.7.1 Lattice-Based CLE

Current lattice-based encryption schemes have a ciphertext expansion factor of $O(q)$ meaning that one plaintext bit is mapped to approximately $q$ bits in the ciphertext. The Huang and Wong construction [HW07a] uses sequential double encryption of an IND-CCA2 secure IBE and an IND-CCA2 secure PKE ($\text{Enc}_{\text{PKE}}(\text{Enc}_{\text{IBE}}(m)))$ and a one-time signature scheme to obtain CCA2 secure
certificateless encryption in the standard model. Lattice-based IBE encryption will map one bit of plaintext to approximately $|q|$ bits (where $|q|$ is the bitlength of a prime $q$) of ciphertext. If this is followed by lattice-based public key encryption, the resulting ciphertext will be expanded in the order of $O(|q|^2)$ bits per plaintext bit. Therefore, if we were to construct the lattice-based CLE using the Huang and Wong construction [HW07a], relying only on lattice-based cryptography, then the encryption would result in a length of $O(|q|^2)$ instead of $O(|q|)$ with our construction, due to the expansion factor of current lattice-based constructions. The ciphertext expansion is already prohibitively enough, but additionally, the encryption algorithm would also be $|q|$ times slower because each of the bits of the first encryption has to be encrypted individually. This makes the Huang and Wong construction unsuitable for lattice-based cryptography. We summarized our findings in Table 4.7.

### 4.7.2 Pairing-Based CLE

The only direct construction to compare our generically obtained modified certificateless Boneh Boyen Goh [BBG05] scheme from Appendix 4.6.2 to is the direct construction by Dent et al. [DLP08]. We note that our construction does not need any pairings during encryption, as opposed to three pairings in the Dent et al. construction. On the other hand, to obtain a CCA2 secure construction, we need a MAC (which is a symmetric algorithm and thus very efficient) and a secure encapsulation scheme (which is equivalent to evaluating a hash function, see [BCHK07, Section 7.2]). Our CCA2 secure decryption uses four pairing computations, which is one pairing more than the Dent et al. construction. The ciphertexts we obtain are longer than the Dent et al. construction. However, our scheme is not susceptible to the Au et al. attack as opposed to the Dent et al. construction. Since our construction uses two encryption schemes in parallel, one under control of the trusted authority and one under control of the user. Therefore the authority cannot make the encryption scheme under control of the user insecure by choosing its parameters in whichever way, because it is an independent encryption scheme. For an overview see Table 4.7.

Huang and Wong [HW07a] do not give an example construction and have a very vague efficiency discussion in which they claim that their scheme can also be instantiated with a MAC. However, in their proof they rely on the functionality of a strong one-time signature scheme, so this claim is not proven. We discussed
4.8 Conclusion

We presented a framework to obtain efficient certificateless encryption schemes in the standard model from identity-based encryption schemes. We extended the ideas presented by Dodis and Katz so that the adversary is given more power than in their constructions, and showed how to prove our new ideas secure. Using our new paradigm of parallel IND-CPA secure encryption, we are able to obtain shorter ciphertexts than the ciphertexts that are generated by the Dodis and Katz construction. We were able to show that the constructions for IND-CCA2 security from identity-based encryption by Boneh et al. [BCHK07] work both for multi-authority IBE and HIBE schemes, and are also valid when used with parallel IND-CPA encryption. As a result, we are able to give a general technique to convert an identity-based encryption scheme into a certificateless encryption scheme, but there are other interesting application scenarios as well. In the certificateless case, the conversion has a tight security reduction and leaves the proof for the IBE scheme intact (in fact, it just uses two IBE schemes in parallel). We give the first construction for a lattice-based certificateless encryption scheme, and also present an efficient conversion using pairing-based schemes. This shows that the conversion technique is independent of the way the IBE scheme is constructed. The resulting schemes are among the most efficient certificateless encryption schemes in the standard model today, and benefit from modern multi-CPU computers.
Chapter 4. Multi-Authority IBE and Certificateless Encryption Schemes
Chapter 5

Certificateless Key Encapsulation Scheme

As described in Section 2.7, a key encapsulation scheme (KEM) allows the transport of a random key used for symmetric encryption from a sender to a receiver. In this chapter, we propose a new certificateless key encapsulation mechanism in the standard model that is very efficient. Our scheme uses only 1 additional exponentiation during decapsulation compared to the IBE-KEM that it is based on and two pairings for key verification before encapsulation is done. For certificateless schemes, we propose to distinguish between the key verification process, and the encapsulation. In practice, a public key changes less often than encapsulated messages are exchanged. Thus, the efficiency of the scheme can be increased if keys are verified once for multiple encryption. This requires storing the verified key securely.

In the literature, there are only two certificateless key encapsulation schemes. One, in the random oracle model, is by Bentahar et al. [BFMLS08]. Since our construction is in the standard model, this construction is not directly comparable. The only standard model secure construction is by Huang and Wong [HW07b] and is a generic construction. We show in Section 5.2 that our construction is more efficient than the Huang and Wong construction. Efficient certificateless KEM schemes are interesting in two aspects. First, when used with the KEM-DEM framework proposed by Shoup, standard model secure certificateless KEM schemes allow very efficient certificateless encryption in the standard
model. Secondly, the key agreement scheme in Section 6.2 uses a certificateless KEM as a building block. If the key agreement scheme is instanciated with the KEM proposed here, we get a very efficient certificateless key exchange protocol in the standard model.

5.1 The CL-KEM Scheme

The security definitions for certificateless key encapsulation schemes were given in Section 3.4. The scheme we propose in this section uses the definition given there. In this section, we give the description of our certificateless key encapsulation mechanism. As specified in Section 3.4.1, the protocol consists of six phases: setup, identity-based key derivation, user key generation, key verification, key encapsulation, and key decapsulation. Our encapsulation scheme extends the identity-based scheme proposed by Kiltz and Galindo [KG06b] to the certificateless case. The algorithms setup, and identity-based key derivation are exactly the same as in Kiltz and Galindo’s KEM [KG06b]. We add the algorithms for user key generation and key verification to the scheme and modify the key encapsulation and key decapsulation algorithms so that they are suitable for certificateless encryption. In Section 5.3 we prove the construction given below secure. We will use bilinear pairings (see Definition 2.2) and Waters’ hash (see Definition 2.13) in the scheme.

5.1.1 CL-KEM Algorithms

Setup

On input of the security parameter $k$, the key generation centre picks suitable bilinear pairing parameters $(e(\cdot, \cdot), p, G, G_T, g)$, a target collision resistant hash function TCR, and uses $\text{HGen}(G)$ to obtain a suitable Waters’ hash function. The KGC also publishes system parameters $(u_1, u_2, z) \in G$. See Algorithm CL-KEM IBE Setup in Figure 5.1 for details.

Identity-Based Key Derivation

To generate an ID-based key for an identity $ID \in \{0,1\}^n$, the key generation centre follows the Algorithm CL-KEM IBE KeyDerivation in Figure 5.1.
5.1. The CL-KEM Scheme

CL-KEM IBE Setup($k$):

\[ u_1, u_2 \leftarrow \mathbb{G}^* \]
\[ a, b \leftarrow \mathbb{Z}_p^*, z \leftarrow e(g^a, g^b) \]
\[ H \leftarrow H_{\text{Gen}}(G) \]
\[ mpk \leftarrow (u_1, u_2, g^a, g^b, z, H) \]
\[ msk \leftarrow \alpha = g^{ab} \]
\[ \text{Return}(mpk, msk) \]

CL-KEM IBE KeyDer($msk, ID$):

\[ s \leftarrow \mathbb{Z}_p^* \]
\[ sk_{ID} \leftarrow (\alpha \cdot H(ID)^s, g^s) \]
\[ \text{Return}(sk_{ID}) \]

CL-KEM User Keygen($mpk, ID$):

\[ (u_1, u_2, g^a, g^b, z, H) \leftarrow mpk \]
\[ \beta_{ID}, \gamma_{ID} \leftarrow \mathbb{Z}_p^* \]
\[ upk \leftarrow (g^{\beta_{ID}}, g^{\gamma_{ID}}, g^{\beta_{ID}a}, g^{\gamma_{ID}b}) \]
\[ usk \leftarrow \beta_{ID} \cdot \gamma_{ID} \]
\[ \text{Return}(upk, usk) \]

CL-KEM KeyVer($mpk, upk$):

\[ (g^{\alpha_{ID}}, g^{\gamma_{ID}}, g^{\beta_{ID}a}, g^{\gamma_{ID}b}) \leftarrow upk \]
\[ (u_1, u_2, g^a, g^b, z, H) \leftarrow mpk \]
\[ \text{Check if } e(g^a, g^{\alpha_{ID}}) \overset{?}{=} e(g, g^{\alpha_{ID}a}) \text{ and } e(g^b, g^{\gamma_{ID}}) \overset{?}{=} e(g, g^{\gamma_{ID}b}) \]
\[ \text{TRUE : } enc_k \leftarrow e(g^{\alpha_{ID}}, g^{\beta_{ID}a}) \]
\[ \text{FALSE : } enc_k \leftarrow G_T \]
\[ \text{Return}(enc_k) \]

CL-KEM Enc($mpk, upk, ID, M$):

\[ enc_k \leftarrow \text{CL-KEM KeyVer}(mpk, upk) \]
\[ (u_1, u_2, g^a, g^b, z, H) \leftarrow mpk \]
\[ r \leftarrow \mathbb{Z}_p^* \]
\[ c_1 \leftarrow g^r \]
\[ c_2 \leftarrow H(ID)^r, t \leftarrow \text{TCR}(c_1) \]
\[ c_3 \leftarrow (u_1 \cdot u_2)^r \]
\[ K \leftarrow enc_k^r \]
\[ C \leftarrow (c_1, c_2, c_3) \]
\[ \text{Return}(K, C) \]

CL-KEM Dec($sk_{ID}, usk, C$):

\[ c_1, c_2, c_3 \leftarrow C \]
\[ d_1, d_2 \leftarrow sk_{ID} \]
\[ r_1, r_2 \leftarrow \mathbb{Z}_p^* \]
\[ t \leftarrow \text{TCR}(c_1) \]
\[ K \leftarrow \left( \frac{e(c_1, d_1 \cdot (u_1 u_2)^r_1 \cdot H(ID)^r_2)}{e(c_2, d_2 \cdot g^r_2) e(g^r_1, c_3)} \right)^{usk} \]
\[ \text{Return}(K) \]

Figure 5.1: Our CCA secure CL-KEM.
Chapter 5. Certificateless Key Encapsulation Scheme

User Key Generation
To obtain a certificateless KEM, we introduce the new algorithm user key generation into the Kiltz-Galindo KEM. The user generates a certificateless key pair from the system parameters as outlined by Algorithm CL-KEM User Keygen in Figure 5.1. After key generation, the user publishes upk and keeps usk private.

Certificateless Key Verification
We add the new key verification algorithm to the certificateless KEM, labeled CL-KEM KeyVer in Figure 5.1. This algorithm makes sure that the master public key is included in the key that is used for encryption. If the master public key is not part of the encryption, key decapsulation will result in a key that is different from the key generated during encapsulation. The output of the key verification algorithm is the encapsulation base key enc_k, that is then used for key encapsulation. Normally, this algorithm is run as a subroutine of the encapsulation algorithm. To make encapsulation more efficient, we suggest to store the output of this algorithm together with the ID and the user public key upk securely, and then retrieve it from the store whenever possible. The reason for this optimisation is that we expect public keys to change less frequently than encaptulations to that public key occur.

Certificateless Key Encapsulation
We modify the Kiltz-Galindo encapsulation mechanism by using the output of the key verification algorithm enc_k instead of the master public key z for encryption. Thus we get a very efficient encapsulation mechanism, outlined by Algorithm CL-KEM Enc in Figure 5.1. The key K is used for encryption in a data encapsulation mechanism (DEM), C is the certificateless encapsulation of K.

Certificateless Key Decapsulation
Decapsulation is also very efficient as it needs only one additional exponentiation over the Kiltz-Galindo KEM decapsulation algorithm. The Algorithm CL-KEM Dec in Figure 5.1 describes the decapsulation. The randomness \( r_1, r_2 \) is used to check the integrity of the ciphertext.

This concludes the description of the certificateless KEM construction.
5.2. Efficiency Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>KeyGen</th>
<th>Enc</th>
<th>Dec</th>
<th>Keysize</th>
<th>Ciphertext</th>
<th>pk</th>
<th>overhead</th>
<th>pk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#pairings + #[multi,regular, fixed-base]-exp</td>
<td></td>
<td></td>
<td>pk</td>
<td>overhead</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IB-KEM [KG06b]</td>
<td>0 + [0,2,0] 0 + [1,3,1] 3 + [1,0,2] n+4 3l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ PKE [KD04]</td>
<td>0 + [0,4,0] 0 + [0,4,0] 0 + [0,2,0] 4 2l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= CL-KEM [HW07b]</td>
<td>0 + [0,6,0] 0 + [1,7,1] 3 + [1,2,2] n+8 5l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>0 + [0,4,2] 0 + [1,3,1] 3 + [1,1,2] n+4 3l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We instantiate the Huang & Wong [HW07b] scheme with the most efficient CCA2 secure PKE scheme by Kurosawa & Desmedt [KD04] and the most efficient CCA2 secure ID-based KEM by Kiltz & Galindo [KG06b] and compare it to our direct construction from the Kiltz & Galindo KEM. Keysize and ciphertext overhead are given in group elements.

Table 5.1: Comparison of the Huang-Wong scheme with our scheme

5.2 Efficiency Comparison

When compared to the only other CL-KEM in the standard model by Huang and Wong [HW07b] instantiated with the most efficient schemes available today, we note that both key generation and encapsulation are more efficient, with encapsulation being almost twice as efficient. We save one exponentiation during decapsulation, key size is smaller and ciphertext size is approximately halved. On the other hand, we introduce the new key verification algorithm, which adds 5 pairings but needs to be executed only infrequently. Furthermore, this algorithm can be run as a regular background task and can do batch checking on the certificateless keys of communication partners. The pairings needed by the key verification algorithm can be seen as two fixed-base pairings and three regular pairings. Fixed-base pairings are potentially faster to compute than regular pairings. The regular pairings are $e(g^a, g^β)$, $e(g^b, g^γ)$ and $e(g^{αβ}, g)$, and the fixed base pairings are $e(g^{βν}, g)$ (keeping $g$ fixed) and $e(g^{αβ}, g^{βν})$ (keeping $g^{αβ}$ fixed), since both can reuse the precomputation steps from $e(g^{αβ}, g)$. We suspect that speedups that are available for fixed point multiplication in elliptic curve cryptography will carry over to the pairings case, using a memory trade-off. Hankerson, Menezes and Vanstone [HMV03, Chapter 3.3.2] give detailed algorithms in the elliptic curve case. For a detailed comparison of our scheme with the CL-KEM construction in the standard model by Huang and Wong [HW07b] see Table 5.1.

We now proceed to prove the security of the proposed certificateless KEM construction.
5.3 Proof of Security for the CL-KEM

**Theorem 5.1.** Assume TCR is a target collision resistant hash function. Under the decisional Bilinear Diffie-Hellman assumption relative to the generator $g$, the CL-KEM from Section 5.1 is secure against chosen ciphertext attacks.

As already described in Section 3.6, proving the protocol is easier if we do not treat Type I and Type II adversaries separately. Essentially, there are two strategies for dealing with an adversary:

- Embed the challenge into the ID-based part. Then the adversary may learn the secret value or replace the certificateless public key. This is generally not applicable for Type II adversaries.

- Embed the challenge into the CL-based part. Then the adversary may learn the ID-based secret key. This is applicable for both Type I and Type II adversaries.

For Type I adversaries that want to learn the CL-key, we use the proof from Kiltz and Galindo [KG06b] unmodified and hand over the user secret value $x_{ID}$ to the adversary. The original proof still holds in this setting.

For Type II adversaries and Type I adversaries that want to learn the ID-based key, we have to modify the proof. The simulator $B$ gets the dBDH challenge $(g, g^a, g^b, g^c, T)$ from its challenger. Given that the adversary $\mathcal{M}$ has an
advantage in the CL-KEM game, $B$ uses the adversary $M$ to get an advantage in solving the dBDH challenge. This strategy simplifies proving the security of the scheme: a well known proof in the ID-based setting is expanded only with what is necessary for the certificateless setting. As it turns out, the proof for the CL-part of the scheme is easier to understand as it does not have to deal with artificial aborts because the public key that is attacked can be determined before the game starts, and constructed accordingly. A graphical overview of the proof is in Figure 5.2.

We rewrite the proof by Kiltz and Galindo to get a proof for the CL-KEM scheme for Type II adversaries. As in Kiltz and Galindo’s paper, the main idea is again that the simulator knows a back door for the hash function $H$. Knowing the back door for $H$ allows the simulator to let $H$ “vanish” for the target identity. To achieve this, we have to embed the challenge slightly differently from the original proof by Kiltz and Galindo [KG06b]. We also use a game based approach. The simulator $B$ starts with knowing the discrete logarithms of $g^a, g^b, g^c$ and “forgets” the discrete logarithms during modifications of the game.

**Game 0.** (Forget $b$) The simulator $B$ picks $(a, b, c) \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, computes $g^c$ and $t^* = \text{TCR}(g^c)$ and additionally picks $d, \delta \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. The **CL-KEM IBE Setup** algorithm is modified as follows:

\[
\text{CL-KEM IBE Setup}(k) : \\
g, u_1 = g^a, u_2 = (g^a)^{-t^*} g^d, \alpha = g^b; z \leftarrow e(g, \alpha) = e(g, g^b) \\
H \overset{\$}{\leftarrow} \text{HGen}(G) \\
mpk \leftarrow (u_1, u_2, g^{1/\delta}, g^{b\delta}, z, H); msk \leftarrow \alpha \\
\text{Return}(mpk, msk)
\] (5.1)

We assume that the adversary $M$ makes no more than $q_0$ queries for distinct identities. One of these identities will be used to create the challenge ciphertext. We enumerate these queries. The simulator $B$ guesses the index of the target identity $ID^*$ that the adversary will use in the test query by selecting $q^* \overset{\$}{\leftarrow} \mathbb{Z}_{q_0}$. We also assume that the adversary does not make more than $q$ decapsulation queries. $B$ sets the target identity’s certificateless public key to $e(g^a, g^b) = z^a$. Both the KGC public key and the master secret key $\alpha = g^b$ can be given to the adversary at the start of the game.
FIND PHASE. During its execution, $\mathcal{M}$ makes a number of reveal master key, reveal ID-based key, reveal secret value, replace public key, and decapsulate requests. The simulator deals with the adversary’s queries in the following way:

Get master key: $B$ returns $\alpha$.

Get user public key($\text{ID}$): If these requests target an identity that has not been initialized before, there are two possibilities: If it is the $q^*$th distinct query, the simulator picks $\epsilon \leftarrow \mathbb{Z}_p$ and returns $(g^{a\epsilon}, g^{1/\epsilon}, g^{a\epsilon/\delta}, g^{b\delta/\epsilon})$. Otherwise, the simulator generates a new certificateless key on the fly as specified by CL-KEM User Keygen, publishes the ID’s certificateless public key in the directory of certificateless public keys and records the certificateless private key $usk = \beta_{\text{ID}} \cdot \gamma_{\text{ID}}$ along with the ID in a table (later referred to as the table of certificateless private keys).

Replace user public key($\text{ID}, upk_{\text{ID}}$): The simulator inserts the new certificateless public key $upk'_{\text{ID}}$ into the table of certificateless public keys and inserts $\perp$ into the table of certificateless private keys at position $\text{ID}$.

Reveal ID-based key($\text{ID}$): (only Type I) As the simulator knows $\alpha = g^b$ these queries can always be answered throughout the game for Type I adversaries. For Type II adversaries, $\alpha$ can be passed to the adversary at the start of the game. Then it is not necessary to provide this functionality to the adversary (the adversary may compute the keys on its own).

Decapsulation($C, \text{ID}$): The simulator returns the decapsulation of $C$ under ID query using the entry from the table of certificateless private keys or $\perp$ if the certificateless public key was replaced by the adversary or $C$ is an invalid encapsulation.

Decapsulation($C, \text{ID}, x$): The simulator returns the decapsulation of $C$ under ID query using $x$ as the user secret value or $\perp$ if $C$ is an invalid encryption.

Eventually, the adversary returns a target identity $\text{ID}^*$. The simulator chooses a random key $K_0^*$ and runs the encapsulation algorithm to create a key $K_1^*$ together with the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$. The challenge ciphertext is computed as

$$c_1^* \leftarrow g^c, t^* \leftarrow \text{TCR}(g^c), c_2^* = H(\text{ID}^*)^c, c_3^* = (u_1^* u_2)^c$$
Then, the simulator chooses a random bit $b'$ and the challenge ciphertext $C^*$ is returned together with the key $K^* = K^*_b$ to the adversary.

**Guess Phase.** The adversary continues to query the oracles provided by the simulator under the condition that he may not request a decapsulation of $C^*$ under $\text{ID}^*$ and may not request the *user secret value* $x_{\text{ID}^*}$. Finally, the adversary returns a bit $b''$. If $b'' = b'$ then the simulator returns 1, else it returns 0. This completes the description of the simulator. Let $X_i$ denote the event that the adversary $\mathcal{M}$ wins game $i$. Since the parameters used in the modified CL-KEM IBE Setup($k$) algorithm are distributed randomly, the system parameters are indistinguishable for any adversary. Furthermore, the values used in the certificateless public key returned in the $q^*$th distinct “get user public key” query are also randomly distributed and are therefore also not distinguishable from a normal key by any adversary. Finally, the challenge ciphertext is constructed in a way that it is a valid ciphertext under identity $\text{ID}^*$ and is therefore also not distinguishable by any adversary. Thus we have for the advantage of the adversary against the CL-KEM scheme: $\text{Adv}_{\text{CL-KEM}, \mathcal{M}}^{\text{cl-kem-cca}} = |\Pr[X_0] - 1/2|.$

**Game 1.** (Eliminate hash collisions) The simulator fixes $c_1^* = g^c$ and $t^* = \text{TCR}(g^c)$ at the start of the game and aborts if a decapsulation query is made for any ciphertext $C = (c_1, c_2, c_3)$ for which $\text{TCR}(c_1) = t^*$ and $c_1 \neq c_1^*$. Otherwise, Game 0 and Game 1 are identical. This event happens only with negligible probability as otherwise $\mathcal{M}$ could be used as an efficient adversary against $\text{TCR}$. Thus we have

$$|\Pr[X_1] - \Pr[X_0]| \leq \text{Adv}_{\text{TCR}, \mathcal{M}}^{\text{hash-TCR}}(k)$$

**Game 2.** (Change of hash keys) The game continues as in Game 1 except that the simulator changes the way the hash keys $\vec{h} = (h_0, h_1, \ldots, h_n)$ are generated. Set $m = 2q$ (where $q$ is the upper bound on the decapsulation queries) and randomly choose

\begin{align*}
x_0, x_1, \ldots, x_n &\leftarrow \{0, \ldots, p - 1\}; \quad y_0', y_1', \ldots, y_n' \leftarrow \{0, \ldots, m - 1\} \\
k &\leftarrow \{0, \ldots, n\}
\end{align*}

and set $y_0 \leftarrow p - km + y_0'$.

$\mathcal{B}$ redefines the public hash keys $\vec{h} = \{h_0, \ldots, h_n\}$ as $h_i = g^{x_i} u_i^{y_i} = g^{x_i}(g^a)^{y_i}$ for $0 \leq i \leq n$. Thus, the public hash function $H$ evaluated at identity $\text{ID} \in \mathcal{S}$
\{0, 1\}^n is given by
\[
H(ID) = h_0 \prod_{i=1}^{n} h_i^{ID_i} = g^{x(ID)} u_1^{y(ID)} = g^{x(ID)} (g^a)^{y(ID)}
\]
with \(x(ID) = x_0 + \sum_{i=1}^{n} ID_i x_i\) and \(y(ID) = y_0 + \sum_{i=1}^{n} ID_i y_i\) (where \(x()\) and \(y()\) are only known to the simulator). As this does not change the distribution of the hash keys, the probability of success for the adversary does not change:
\[
\Pr[X_2] = \Pr[X_1].
\]

**Game 3. (Abort for wrong challenge identity)** The simulation proceeds as in Game 2. Once the simulator is being asked the *challenge ciphertext* query, it checks that \(ID^*\) is the \(q^*\)th distinct identity and aborts otherwise. The simulator also aborts if \(y(ID^*) \neq 0\).

As we do not need to change the key derivation oracle during the sequence of games (as Kiltz and Galindo do), we can simplify the proof significantly. We especially do not have to deal with artificial aborts, as the abort probability for the simulator can be estimated directly using results from Kiltz and Galindo [KG06b, Section A.2]. From Equation 5.2 we have that

\[
y(ID^*) = 0 = p - km + y'_0 + \sum_{i=1}^{n} ID_i^* y_i
\]

and from the distribution of the \(y_i\) we get that

\[
0 \leq y'_0 + \sum_{i=1}^{n} ID_i^* y_i < (n + 1)m.
\]

Thus if \(y(ID^*) = 0 \mod m\), then there is a unique \(0 \leq k < n + 1\) such that \(y(ID^*) = 0\) over the integers. Since \(k\) is uniformly and independently distributed over the integers, we get:

\[
\Pr[y(ID^*) = 0] = \Pr[y(ID^*) = 0 \mod p] \geq \Pr[y(ID^*) = 0 \mod m]/(n + 1).
\]

Thus for a fixed \(k\) and \(b \in \mathbb{Z}_m\) we have that \(\Pr[y(ID) = b \mod m] = 1/m\). So we
conclude with
\[ \Pr[y(\text{ID}^*) = 0] \geq \frac{1}{n + 1} \Pr[y(\text{ID}^*) = 0 \mod m] = \frac{1}{n + 1} \cdot \frac{1}{m} = \frac{1}{m(n + 1)}. \]

Thus, the probability that Game 3 succeeds is given by the probability that \( y(\text{ID}^*) = 0 \) and that \( \text{ID}^* \) is the \( q^* \)th distinct identity. As there are at most \( q_0 \) distinct ID queries by the adversary we have
\[ \Pr[X_3] \geq \Pr[X_2]/(q_0m(n + 1)). \]

**Game 4.** (Change of decapsulation oracle / Forget \( a \)) The simulator knows all user secret keys except for those the adversary replaced with a replace certificateless public key request. Regarding decapsulation queries, the simulator does not have to answer requests for identities that were issued a replace certificateless public key query unless the adversary supplies the user secret key matching the replaced certificateless public key. As the simulator can derive ID-based private keys from the master parameters, answering decapsulation queries for all identities except \( \text{ID}^* \) is easy, as all secret information to do this is readily available using the standard CL-KEM Dec algorithm as described in Figure 5.1.

The simulator established in Game 3 that \( y(\text{ID}^*) = 0 \). This enables the simulator to answer decapsulation queries for \( \text{ID}^* \) in the following way: instead of answering the decapsulation as in CL-KEM Dec in Figure 5.1, the simulator computes the decapsulations for \( \text{ID}^* \) as follows: with \( u_1 = g^a, u_2 = (g^a)^{-t^*}g^d \) and \( c_1 = g^* \) we have
\[ c_3 = (u_1^tu_2)^r = ((g^a)^t g^{-t^*}g^d)^r = ((g^{a(t-t^*)}g^d)^r) = (c_1^t)^{t-t^*} \cdot c_1^d. \]

To decapsulate the correct key \( K \), we would like to compute \( e(g^a, g^b)^r \). Thus knowing \( g^b \) and computing \( c_1^a = (g^r)^a = g^{ra} \) will allow us to compute \( K \) by computing \( e(g^a, g^b) = e(g, g)^{rab} \):
\[ \left( c_3/c_1^d \right)^{t-t^*} = \left( (c_1^t)^{t-t^*} \cdot c_1^d / c_1^d \right)^{t-t^*} = (c_1^a)^{t-t^*} / c_1^d = c_1^a = g^{ra}. \]

As \( K = e(g^a, g^b)^r = e(g, g)^{abr}, \) knowing \( t = TCR(g^r) \) we can recompute \( K \) with
\[ K = e \left( g^b, \left( c_3/c_1^d \right)^{t-t^*} \right) = e(g^b, g^{ar}) = e(g, g)^{abr}. \]
Chapter 5. Certificateless Key Encapsulation Scheme

As this behaviour does not alter the adversary’s view of the game we have

$$\Pr[X_4] = \Pr[X_3].$$

**Game 5.** (Modify the challenge / Forget c) The simulator changes its answer to the get challenge key encapsulation query. Game 3 established that $$y(\ID^*) = 0 \mod p$$, thus the simulator can compute the challenge ciphertext $$C^* = (c^*_1, c^*_2, c^*_3)$$ as

$$c^*_1 = g^e, c^*_2 = (g^e)^x(\ID^*), c^*_3 = (g^e)^d, K = T$$

where $$g^e$$ and $$T$$ are given by the challenger before the game starts. Now the answer of the adversary to the challenge ciphertext is directly related to the challenge, and thus the simulator has an advantage in solving the dBDH challenge if the adversary has an advantage in winning the game:

$$\text{Adv}^{cl-kem-cca}_{CL-KEM,M} = \left| \Pr[X_0] - \frac{1}{2} \right| \leq \frac{1}{q_0m(n+1)} \text{Adv}^{dBDH}_M(k) + \text{Adv}^{\text{hash-tcr}}_{\text{TCR},M}(k) - \frac{1}{2}.$$ 

**5.4 Conclusion**

We have shown how to construct an efficient CL-KEM scheme from an existing ID-based KEM scheme in the standard model. Our construction requires only two standard and two fixed-base exponentiations during the construction of the certificateless key and one additional exponentiation during the decapsulation compared to the original ID-based KEM scheme. We introduced the additional key verification algorithm which uses 5 pairings, which may be reduced to two fixed-base pairings and one regular pairing. However, key verification needs to be done only infrequently compared to encryption and decryption. In Section 5.2 we showed that the proposed scheme is more efficient than any generic construction that has been published before. By modifying the ID-based KEM scheme by Kiltz and Galindo [KG06b] which is one of the most efficient ID-based KEM schemes in the standard model, we obtain the most efficient CL-KEM scheme in the standard model today. Proving the scheme secure is relatively straightforward since we can rely on the proof published by Kiltz and Galindo and need to consider only the differences needed in the certificateless case.
Chapter 6

Certificateless Key Agreement Schemes

Key agreement enables two parties to securely agree on a common key over a potentially insecure channel. There are many identity-based key agreement schemes with a proof of security, however ID-based cryptography has the inherent drawback that the trusted authority is able to do key escrow. One can argue that if ID-based key agreement uses ephemeral keys in addition to the user’s static keys, then the trusted authority will not be able to decrypt. However ephemeral keys do not provide long-term security as Krawczyk [Kra05b, Kra05a] points out. Since ephemeral keys are usually generated in advance and not stored in especially protected memory, they may leak to the adversary (in this case the trusted authority). Therefore, ID-based key agreement does not fully preserve the user’s privacy, even with ephemeral keys.

A valid solution is certificateless key agreement with a security model that matches the capabilities of a potential adversary closely. The first certificateless key agreement scheme was proposed by Al-Riyami and Paterson [ARP03] as a side note to their famous paper on the first certificateless encryption scheme. However, the scheme was published without a security model for certificateless key agreement and also without a proof of security. Other certificateless key agreement schemes were published after that. None of the published schemes has a security model for certificateless key agreement. Some of the schemes did not have a proof of security either [Zh05, WCW06, MT06, HKK07, WCB08, XWSX08],
but argued only heuristically why they would be secure. Other schemes [WZ07, LWZ08, WZ08] did have a proof of security in the generic Canetti-Krawczyk model for key agreement (discussed in Section 2.8.1) but did not consider the abilities that a certificateless adversary has, and thus the proofs were incomplete. A certificateless adversary may replace certificateless public keys or corrupt some of the long term keys of the parties that participate in the test query. Somehow the authors of the proofs set in the generic CK model overlooked these additional capabilities of a certificateless adversary.

Swanson [Swa08, SJ09] published a model for certificateless key agreement and analysed four of the certificateless key agreement schemes [Zh05, WCW06, MT06, XWSX08] in her model. She showed generic attacks in her model for certificateless key agreement that break the notions of security claimed by the respective authors.

In this chapter, we propose two new certificateless key agreement schemes which are based on the security models for certificateless key agreement developed in Section 3.5. Our e\textsuperscript{2}CK security model is an extension of the model proposed by Swanson that is slightly stronger and thus gives more power to the adversary. The schemes that we present in this chapter fall in two categories. One scheme is in the random oracle model and is in the strong certificateless security model (see Section 3.2). The other scheme is in the standard model and considers the weak certificateless security model, where queries of the adversary are not answered if a public key has been replaced. This scheme uses (as a subroutine) a certificateless KEM, for example the certificateless KEM that we discussed in the previous chapter.

6.1 Strongly Secure Key Agreement in the Random Oracle Model

In this section we propose a new certificateless key agreement scheme in the random oracle model that can withstand all of the attacks that Swanson [Swa08, SJ09] published. The scheme discussed in this section is in the strong certificateless security model and is thus able to answer key derivation queries even if the adversary replaced some of the certificateless public keys in the schemes and did not disclose the matching private keys to the simulator. To answer these queries, we make use of the twin bilinear Diffie-Hellman oracles as discussed in Theorems
6.1. Strongly Secure Key Agreement in the Random Oracle Model

2.1, 2.2 and 2.3 in Section 2.2.7. The security of the scheme is based on the computational bilinear Diffie-Hellman assumption (see Definition 2.7), and on the computational Diffie-Hellman assumption (see Definition 2.5). We note that the computational bilinear Diffie-Hellman assumption is a stronger assumption than the computational Diffie-Hellman assumption meaning that if an adversary has an advantage against the computational Diffie-Hellman assumption, then that adversary also has an advantage against the computational bilinear Diffie-Hellman assumption.

We proceed to describe the algorithms that the key agreement protocol consists of and then prove the protocol secure.

6.1.1 The Key Agreement Algorithms

We describe the phases of our certificateless authenticated key exchange protocol in this section. A graphical representation of the message exchange and the key computation of our protocol is shown in Figure 6.1. Our protocol consists of three phases: setup, message exchange and key computation. We also briefly address the efficiency of the proposed protocol.

Setup

- The KGC publishes a generator \( g \in G \) and an admissible bilinear pairing \( e : G \times G \rightarrow G_T \) (see Definition 2.2).

Suitable pairing groups for this protocol would be Type 1 and Type 4 pairings (see Chen, Cheng & Smart [CCS07] for a discussion). Asymmetric pairings are not possible because we use the non-interactive ID-based key agreement of Sakai, Ohgishi and Kasahara (SOK) [SOK00] as part of our protocol. This requires hashing to both \( G_1 \) and \( G_2 \). The SOK protocol has been proven by Dupont and Enge [DE02] using gap assumptions. As an added benefit of our proof, we show how to prove the SOK protocol secure under the weaker computational bilinear Diffie-Hellman assumption using the twin bilinear Diffie-Hellman trapdoor [CKS08] in Section 6.1.2, Strategy 9.

- The KGC picks a random \( s \in \mathbb{Z}_p \) as master secret key and sets its public key to \( g^s \).

The KGC selects three cryptographic hash functions

\[ H_1 : \{0, 1\}^* \rightarrow G \]
\[ H_2 : \{0, 1\}^* \times \{0, 1\}^* \times G^8 \times G_T^6 \rightarrow \{0, 1\}^n \text{ for some integer } n > 0 \]
\[ H_3 : G \rightarrow G \]

\( H_2 \) is the key derivation function for our scheme.

Each party participating in the key agreement protocol additionally computes a private key and a matching certificateless public key:

- Each user \( U \) generates a secret value \( x_U \leftarrow \mathbb{Z}_p \) and a public key \( g^{x_U} \in G \)
- Each user \( U \) gets an ID-based private key \( \{H_1(\text{ID}_U)^s, H_3(H_1(\text{ID}_U))^s\} \in G^2 \) from the key generation centre.

**Message Exchange**

To establish a common key, user \( A \) generates the ephemeral secret \( r_A \leftarrow \mathbb{Z}_p \) and user \( B \) generates the ephemeral secret \( r_B \leftarrow \mathbb{Z}_p \). They exchange the following messages:

\[ A \rightarrow B : E_A = (g^{r_A}, g^{x_A}) \quad B \rightarrow A : E_B = (g^{r_B}, g^{x_B}) \]

We note that the certificateless public keys can be stripped from the messages if they are published in a public online directory. This will save bandwidth, but at the same time may make the scheme more vulnerable to the equivalent of denial of decryption attacks (discussed in Section 3.1.1) in certificateless encryption: an adversary may manipulate the entries of the directory more easily than the message exchange between two parties.

As we propose a one-round protocol, our protocol achieves only implicit authentication. Krawczyk [Kra05a, Section 8] shows that explicit authentication is possible with three messages. To achieve explicit authentication, this protocol can be patched in the same way that HMQV is patched to HMQV-C, by having the responder party compute the session key \( SK \) first, and then sending the message \( \text{Mac}_{SK}(0) \) in addition to the regular messages that are sent. The initiator party then sends an additional message \( \text{Mac}_{SK}(1) \) after it computed the session key to the responder party. Both parties verify the MAC’s of the prescribed
messages before accepting the session. These are the only changes required for explicit authentication.

In the following we require implicitly that each party always checks subgroup membership for all elements of messages that are exchanged in the protocol to defend against small subgroup attacks [LL97].

Key Computation

To compute the certificateless session key, each user computes

\[ K_A = e(H_1(ID_A), g^s)^{x_A} e(H_3(ID_A)\langle H_1(ID_A)\rangle^s, g^{x_B}) \]
\[ = e(H_1(ID_B), g)^{r_A} e(H_1(ID_A), g)^{r_B s} \]
\[ = e(H_1(ID_A), g^s)^{r_B} e(H_1(ID_B)\langle H_3(ID_A)\rangle^s, g^{x_A}) \]
\[ = K_B = K \]

\[ K'_A = e(H_3(ID_B), g^s)^{r_A} e(H_3(ID_A)\langle H_1(ID_A)\rangle^s, g^{x_B}) \]
\[ = e(H_3(ID_B), g)^{r_A} \cdot e(H_3(ID_A)\langle H_1(ID_A)\rangle^s, g^{r_B s}) \]
\[ = e(H_3(ID_A)\langle H_1(ID_A)\rangle^s, g^{r_A}) \cdot e(H_3(ID_A)\langle H_1(ID_A)\rangle^s, g^{x_B}) \]
\[ = K'_B = K' \]
The session key is then computed as 

\[ SK = H_2(A, B, E_A, E_B, g^r_{AB}, g^{r_{AB}}, g^{r_{AB}}, K, K', L, L', N, N') \]

In Section 6.1.2 the simulator \( B \) uses the adversary \( M \) to solve either the computational Diffie-Hellman (CDH) or the computational bilinear Diffie-Hellman (CBDH) problem. \( K, L, \) and \( N \) are used in the proof to embed the input to the CBDH challenge into the test session. Each of these values is necessary to defend against one possible attack strategy of the adversary \( M \). \( K \) is the product of two encapsulated Boneh-Franklin session keys, \( L \) is similar but with certificateless long-term keys. \( N \) is the non-interactive ID-based key agreement scheme proposed by [SOK00]. \( K', L', \) and \( N' \) are needed to answer reveal queries of the adversary \( M \) consistently. To answer reveal queries, the simulator \( B \) makes use of the twin bilinear Diffie-Hellman problem as introduced by Cash, Kiltz and Shoup [CKS08] and our two additional theorems presented in Section 2.2.7. The twin bilinear Diffie-Hellman “backdoor” is embedded in \( K', L' \) and \( N' \).
6.1. Strongly Secure Key Agreement in the Random Oracle Model

Efficiency Considerations

Although the protocol is one round, the computational overhead imposed on the parties is rather high: each party has to compute 5 exponentiations in $G$ and 10 pairings. We would like to note that we need the $H_3$ hash function in the proof for full computational bilinear Diffie-Hellman security. If the gap bilinear Diffie-Hellman assumption is used (see Okamoto and Pointcheval [OP01] and Kudla and Paterson [KP05] for gap assumptions), the $H_3$ hash function can be omitted which saves 2 hash queries and reduces the complexity of the protocol to 3 exponentiations in $G$ and 5 pairing computations (as $K'$, $L'$, and $N'$ do not have to be computed). If there are multiple runs of the protocol between the same users (e.g. for rekeying in VPN’s), then the complexity can be reduced by caching $g^{x^A x_B}$, $L$, $L'$, $N$, and $N'$ in secure memory which then reduces the complexity for successive runs to 4 exponentiations and 4 pairing computations (or 2 exponentiations and 2 pairing computations if the gap bilinear Diffie-Hellman assumption is used). It may be possible to do better in terms of computational efficiency. However, we aim to provide a strong model for certificateless key agreement and to show that schemes corresponding to the model exist.

In the proof below, we use Theorems 2.1, 2.2, and 2.3 as decisional oracles to be able to answer the $H_2$ queries of the adversary consistently (and to determine when the adversary submits the solution to a hard problem to the $H_2$ oracle). We continue then by embedding a hard problem in each of the uncorrupted secrets that are available in the respective strategies.

6.1.2 Security Proof for the Certificateless Key Agreement Scheme

We will prove that the certificateless key agreement scheme is a secure key agreement scheme in the random oracle model under the computational bilinear Diffie-Hellman (CBDH) assumption (see Definition 2.7) and the computational Diffie-Hellman (CDH) assumption (see Definition 2.5). The proof of security is in the $e^2$CK model that we discussed in Section 3.5.1.

In the security proof that follows, we do not differentiate between Type I and Type II adversaries but treat them together. If the adversary was split to be either Strong Type I or Strong Type II, then a Strong Type II adversary would
be applicable only for the Strategies 1, 2, 3, and 4 that we explain in Section “Possible strategies for the simulator” on page 157. Being able to distinguish between Type I and Type II adversaries would thus increase the probability of success for the simulator.

To relate the advantage of an adversary against our protocol to the CBDH and CDH assumptions, we use a classical reduction approach. We assume that an adversary $\mathcal{M}$ has an advantage in winning the game outlined in Section 3.5.1. Additionally, the adversary $\mathcal{M}$ may query the random oracles $H_1, H_2$, and $H_3$. In the following, the simulator $\mathcal{B}$ is interested to use the adversary $\mathcal{M}$ to turn $\mathcal{M}$’s advantage in distinguishing a random session key from the correct session key into an advantage to solve either the computational Diffie-Hellman problem or the computational bilinear Diffie-Hellman problem. Let $q_0$ be the maximum number of sessions that any one party may have. We assume that the adversary $\mathcal{M}$ makes at most $q_1$ distinct $H_1$ queries. The adversary may make any number of $H_2$ queries or $H_3$ queries. At the end of the game, $\mathcal{M}$ outputs its guess $\hat{b} \in \{0, 1\}$ for $b$. Let $\text{Adv}_\mathcal{M}^{\Pi}(k)$ be the advantage that the adversary $\mathcal{M}$ has against the protocol, i.e. the event that $\hat{b} = b$ and $\mathcal{M}$ wins the game.

**Theorem 6.1.** If there exists an efficient PPT adversary that has an advantage against our certificateless key agreement scheme $\text{Adv}_\mathcal{M}^{\Pi}(k)$, then the simulator $\mathcal{B}$ can use this adversary to solve either the computational Diffie-Hellman or the computational bilinear Diffie-Hellman problem. We show that the success probability of any adversary is limited by

$$\text{Adv}_\mathcal{M}^{\Pi}(k) \leq 9q_0q_1^2 \max(\text{Adv}_\mathcal{B}^{\text{CDH}}(k), \text{Adv}_\mathcal{B}^{\text{CBDH}}(k))$$

where $\text{Adv}_\mathcal{B}^{\text{CDH}}(k)$ is the advantage that the simulator gets in solving the computational Diffie-Hellman problem given security parameter $k$ using the adversary and $\text{Adv}_\mathcal{B}^{\text{CBDH}}(k)$ is the advantage that the simulator gets in solving the computational bilinear Diffie-Hellman problem given security parameter $k$ using the adversary.

We note that the CBDH problem is strictly weaker than the CDH problem. Thus, an adversary that is able to solve the CDH problem will also be able to solve the CBDH problem. We differentiate between these two problems because security against a Type II adversary is based solely on the CDH problem, whereas security against a Type I adversary is based on both the CDH problem and the
6.1. Strongly Secure Key Agreement in the Random Oracle Model

CBDH problem.

Possible Strategies for the Simulator

We use the notation $\Pi_{i,j}^t$ to identify a session, where $i$ and $j$ are the parties participating in the session and $t$ is the $t$th execution of a session at party $j$. In the paper where LaMaccia et al. [LLM07] propose the eCK model, they suggest to label a session by $\text{sid} = (\text{role}, \text{ID}_I, \text{ID}_J^*, \text{comm}_1, \ldots, \text{comm}_n)$ where $\text{comm}_x$ is the communication between the parties. However, these session identifiers are not constant over time (because they change after each exchanged message) and are therefore also not easily enumerated. Therefore we decided to use the notation introduced above.

Before the game starts, the simulator $B$ tries to guess the test session. To this end, $B$ randomly selects two indexes $I, J \in \{1, \ldots, q_1\}$: $I \neq J$ that represent the $I^{th}$ and the $J^{th}$ distinct query to the $H_1$ oracle. The probability that $B$ chooses $I$ and $J$ correctly is (as there are at most $q_1$ entries in $H_1$)

\[
\frac{1}{q_1(q_1 - 1)} > \frac{1}{q_1^2}
\]

Let $q_0$ be the upper limit of sessions that any one party will run during the execution of the protocol in the security game. $B$ chooses $T \in \{1, \ldots, q_0\}$ and thus determines the test oracle $\Pi_{i,j}^T$, which is correct with probability larger than $\frac{1}{q_0 q_1}$. If $B$ did not guess the test session correctly, $B$ aborts the game.

In order to use the adversary $M$ to gain an advantage in computing the CBDH or the CDH challenge, the simulator $B$ will guess the parts of the key in the session corresponding to the test query that the adversary may not learn. Depending on the chosen strategy, $B$ aborts the game whenever $M$’s queries target one of the forbidden elements. Otherwise, the game proceeds as usual. There are nine choices for $B$ (see also Table 6.1):

1. The adversary may neither reveal the secret value of $\text{ID}_I$ nor of $\text{ID}_J$.

2. The adversary may neither reveal the ephemeral private key of $\text{ID}_I$ nor of $\text{ID}_J$.

3. The adversary may neither reveal the ephemeral private key of $\text{ID}_J$ nor the secret value of $\text{ID}_I$. 
Strategy 1 - 4 are related to the computational Diffie-Hellman problem, Strategies 5 - 9 are related to the computational bilinear Diffie-Hellman problem. In the proof, the problem is always embedded in the values that the adversary may not corrupt or replace.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>1</th>
<th>2</th>
<th>3/4(mirr.)</th>
<th>5/6(mirr.)</th>
<th>7/8(mirr.)</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value at party p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$H_1(ID_p)^*$</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>$H_3(H_1(ID_p))^*$</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>$x_p / g^{x_p}$</td>
<td>c/r</td>
<td>c/r</td>
<td>c/r</td>
<td>c/r</td>
<td>c/r</td>
<td>c/r</td>
</tr>
<tr>
<td>$r_p$</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>Embedding in</td>
<td>$g^{r_1 x_2}$</td>
<td>$g^{r_2 x_1}$</td>
<td>$g^{r_1 x_2} \text{ or } g^{r_2 x_1}$</td>
<td>K</td>
<td>L</td>
<td>N</td>
</tr>
<tr>
<td>Problem type</td>
<td>CDH</td>
<td>CDH</td>
<td>CDH</td>
<td>CBDH</td>
<td>CBDH</td>
<td>CBDH</td>
</tr>
</tbody>
</table>

$c = \text{corrupt/reveal}, \ r = \text{replace}, \ \text{mirr.} = \text{swap columns I and J}$

Table 6.1: Possible corrupt queries sorted by strategy

4. The adversary may neither reveal the ephemeral private key of $ID_I$ nor the secret value of $ID_J$.

5. The adversary may neither reveal the secret value of $ID_J$ nor replace the public key of $ID_J$ and may also not reveal the ID-based private key of $ID_I$.

6. The adversary may neither reveal the secret value of $ID_I$ nor replace the public key of $ID_I$ and may also not reveal the ID-based private key of $ID_J$.

7. The adversary may neither reveal the ephemeral private key of $ID_J$ nor the ID-based private key of $ID_I$.

8. The adversary may neither reveal the ephemeral private key of $ID_I$ nor the ID-based private key of $ID_J$.

9. The adversary may neither reveal the ID-based private key of $ID_I$ nor of $ID_J$.

$B$ selects one of these strategies uniformly at random for the remainder of the proof and aborts if this is not the strategy chosen by the adversary. As there are nine strategies, the probability that $B$ does not abort the game after $B$ selected the strategy and the test session beforehand is now larger than $\frac{1}{9^{q_0 q_1}}$. The adversary may learn the key generation centre’s master secret only in Strategy 1,2,3, and 4. Furthermore, $B$ replaces the $H_2$ oracle by a table which records input/output pairs. If a query is made that matches one of the previous inputs, the corresponding output is returned, otherwise, a value from the respective output domain is chosen at random, the new input/output pair is added to the
6.1. Strongly Secure Key Agreement in the Random Oracle Model

Instead of choosing $H_1(\text{ID}_i)$ at random from $G$, $B$ chooses $l_i \in \mathbb{Z}_p$ at random, records it, and sets $H_1(\text{ID}_i)$ to $g^{l_i}$. For Strategy 5, 7 and 9, the $I^{th}$ entry is set to $H_1(\text{ID}_I) = g^b$; for Strategy 6 and 8, the $J^{th}$ entry is set to $H_1(\text{ID}_J) = g^c$. For Strategy 9 the $J^{th}$ entry is set to $H_1(\text{ID}_J) = g^c$. $g^b$ and $g^c$ are taken from the inputs to the BDH challenge. As $g^b$ and $g^c$ are random in $G$, this modification is indistinguishable for any adversary. The table above shows the $H_1$ oracle for Strategy 9 as an example. In Strategy 1 to 4, there are no special entries for the target identities, so all entries will look like entry 1.

Table 6.2: Modified $H_1$ oracle

<table>
<thead>
<tr>
<th>ID</th>
<th>$H_1(\text{ID})$</th>
<th>$l \xrightarrow{\text{Z}_p}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>$g^{l_i}$</td>
<td>$l_i$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ID</td>
<td>$g^b$</td>
<td>$\perp$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ID</td>
<td>$g^c$</td>
<td>$\perp$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

list and the value is returned. The $H_1$ and $H_3$ oracle operate as explained in Table 6.2 and Table 6.3 respectively.

**Behaviour of the Simulator Based on the Chosen Strategy**

To solve the computational Diffie-Hellman problem using $\mathcal{M}$, $B$ is given the values $(g^a, g^b)$ and $B$’s task is to compute $g^{ab}$. To solve this problem, $B$ uses the $H_2$ oracle. The bilinear pairing is used for consistency checks.

To solve the computational bilinear Diffie-Hellman problem using $\mathcal{M}$, $B$ is given the values $(g^a, g^b, g^c)$ and $B$’s task is to compute $e(g, g)^{abc}$. To solve this problem, $B$ uses the $H_2$ and the $H_1$ oracle. The $H_3$ oracle is used for consistency checks and operates as in Table 6.3.

The session key $SK$ is generated by querying $H_2$ on $(\text{ID}_I, \text{ID}_j, g^{r_i}, g^{x_i}, g^{r_j}, g^{r_jx_i}, g^{x_i}, g^{x_j}, g^{x_jx_i}, K, K', L, L', N, N')$ where

$$K = e(H_1(\text{ID}_j), g)^{r_i} \cdot e(H_1(\text{ID}_I), g)^{r_j},$$

$$L = e(H_1(\text{ID}_j), g)^{sx_i} \cdot e(H_1(\text{ID}_I), g)^{sx_j},$$

$$N = e(H_1(\text{ID}_I), H_1(\text{ID}_j))^s$$

Depending on the chosen strategy, $B$ embeds the challenge in the test query...
Instead of choosing $H_3(g_i)$ for $g_i \in G$ at random from $G$, $B$ chooses $y_i \in Z_p$ at random, records it, and sets $H_3(g_i)$ to $g^{y_i}$. For Strategy 5, 6, 7, 8 and 9, the oracle is patched before the game starts by setting $H_3(g_b) = g^{y_{tbdh_1}/g^{z_b}}$. For Strategy 9, the oracle is additionally patched before the game starts with $H_3(g_c) = g^{y_{tbdh_2}/g^{z_c}}$. $g_b$ and $g_c$ are taken from the inputs to the BDH challenge. As the pre-patched values are completely re-randomized, this modification is indistinguishable for any adversary. The table above shows the $H_3$ oracle for Strategy 9 as an example. In Strategy 1 to 4 this table does not have any special entries with $z$ values.

Table 6.3: Modified $H_3$ oracle suitable for twin bilinear Diffie-Hellman and answers the test query as specified in Section 3.5.1.

### Patching the $H_2$ Oracle

$B$ has to maintain consistency between the $H_2$ oracle and session key reveal queries, as $B$ will not be able to compute all data necessary to query the $H_2$ oracle for a valid session key in some instances (e.g. if certificateless public keys have been replaced by the adversary). If $B$ has been asked on the $H_2$ oracle first and is then later asked a matching session key reveal query, $B$ is always able to answer these requests correctly (it uses its decisional oracles that are explained in the proofs for respective strategies on page 162). However, if $B$ is asked a session key reveal query for which no matching $H_2$ query exists yet, $B$ proceeds as follows: $B$ inserts all available data and all data that $B$ is able to compute (see also the section on session key reveal queries on page 161) into the $H_2$ oracle but may have to leave some fields (like $K$ and $K'$ or $L$ and $L'$ or $N$ and $N'$) empty. $B$ chooses a random value from $H_2$’s output domain as the session key and records that value together with the incomplete $H_2$ query data. For the following $H_2$ queries, $B$ first checks if one of the incomplete entries of the $H_2$ oracle matches $M$’s query data by using the respective decisional oracle(s). If that is the case, $B$ records the complete information submitted by $M$ and returns the $H_2$ entry. $B$ additionally fills up all long term values that it can determine (even if it is
not able to fill a $H_2$ entry completely). If $B$ finds no matching entry, $B$ simply generates a new $H_2$ entry as usual.

**Handling a Session Key Reveal Query for Sessions $Π^s_{i,j}$ where Party $i$ and $j$ are not Participating in the Test Query**

Without loss of generality, we assume that $i$ is the initiator of the session. Given party $i$ that has incoming message $(g^{r_i}M_j,g^{r_M_j})$ (where $M_j$ indicates that the values may be adversarial controlled) and that thus accepts, the simulator knows at least the identity-based private keys and the ephemeral private key of party $i$, i.e. the simulator knows $H_1(ID_i)^s$, $H_3(H_1(ID_i))^s$, $r_i$. The adversary may have replaced the certificateless public key of party $i$ with $g^{r_iM_i}$. To obtain a session key, party $i$ has to query the $H_2$ oracle with the session data (as explained in Section 6.1.1) on the following elements:

$$SK = H_2(ID_i,ID_j,g^{r_i},g^{r_M_i},g^{r_M_j},g^{r_R_i},g^{r_R_M_j},g^{r_{iM_j}},g^{r_{M_i},r_{M_j}},K,K',L,L',N,N')$$

Besides the public values $ID_i,ID_j,g^{r_i},g^{r_M_i},g^{r_M_j},g^{r_R_i},g^{r_R_M_j}$ that are part of the $H_2$ query, the simulator acting as party $i$ is able to compute the following values knowing its (possibly corrupted) private information $H_1(ID_i)^s$, $H_3(H_1(ID_i))^s$, $r_i$:

- $g^{r_{M_j}}$ trivially, by computing $(g^{r_{M_j}})^{r_i}$
- $g^{r_{iM_j}}$ by computing $(g^{r_{M_j}})^{r_i}$
- $K$ due to the patched $H_1$ oracle (see Table 6.2), the simulator knows $\log_g H_1(ID_i) = l_i$ and $\log_g H_1(ID_j) = l_j$. Thus $K$ can be computed as

$$K = e(H_1(ID_j),g^s)^{r_i} = e((g^s)^{l_i},g^{r_{M_j}})$$

- $K'$ just like for $K$, the simulator knows $\log_g H_3(H_1(ID_i)) = y_i$ and $\log_g H_3(H_1(ID_j)) = y_j$ (see Table 6.3). Thus $K'$ can be computed as

$$K' = e(H_3(H_1(ID_j)),g^s)^{r_i} = e((g^s)^{y_i},g^{r_{M_j}})$$

- $L$ Knowing $l_i$ and $l_j$ from the $H_1$ oracle computing $L$ is easy:

$$L = e((g^{r_{M_j}})^{l_j},g^s) = e((g^s)^{l_i},g^{r_{M_j}})$$
$L'$ can be computed similarly, just like $K'$ above.

- $N$ and $N'$ are easy as the ID-based private keys are known.

The only missing values are $g^{x_1, x_2}$ and $g^{x_1, r_2}$ which cannot be computed by the simulator. However, as we point out in the proof for Strategy 1 below, the simulator is still able to answer session state reveal and $H_2$ queries consistently: If the simulator is asked a $H_2$ query first and then later asked a matching session state reveal query, the simulator can identify the corresponding $H_2$ entry by checking for all entries if $e(g^{x_1}, g^{x_2}) = e(g^{x_1, r_2}, g)$ and if $e(g^{x_1, r_2}, g) = e(g^{x_1}, g)$. If the simulator is asked a session state reveal query, but there is no matching $H_2$ entry, the simulator can create a new random value from the output domain of $H_2$ and assign it to the incomplete entry. The simulator checks the subsequent queries of the adversary to the $H_2$ oracle and is able to answer the queries correctly by using the pairing as above.

In the following, we will split the simulator’s behaviour based on the strategy chosen as explained on page 157. Additionally, we omit the indices $i, j$ with respect to key computations for specific sessions to increase readability. Usually it is evident for which particular session the computations are needed. For the proof we assume that the adversary $M$ does not get an advantage in outputting its guess $\hat{b}$ for $b$ unless $M$ queries the $H_2$ oracle on the session key.

**Proofs for Strategy 1 to 9**

**Strategy 1** The allowed corrupt queries for the adversary are listed in Table 6.1. The simulator $B$ wants to use the adversary $M$ to solve the computational Diffie-Hellman problem. The input for $B$ is $(g^a, g^b) \in G^2$ and $B$’s goal is to compute $g^{ab}$. To this end, $B$ sets the certificateless public key of $ID_I$ to $g^a$ and the certificateless public key of $ID_J$ to $g^b$. $B$ uses the pairing to check whether the queries of the adversary to the $H_2$ oracle are valid: by computing $e(g^a, g^b) = e(g^{ab}, g)$, $B$ is able to identify valid queries. As soon as $B$ finds such a query, $B$ aborts the game and returns $g^{ab}$ as solution of the CDH challenge.

The probability that $B$ is able to find a solution to the CDH challenge is

$$\text{Adv}_{B}^{CDH}(k) \geq \frac{\text{Adv}_{M}^{\Pi}(k)}{q_0 q_1^2}$$

$B$ is able to compute all other elements $(g^{x_1, x_2}, K, K', L, L', N, N')$ that are
necessary for \( H_2 \) queries as the respective private values are under \( B \)'s control. If \( \mathcal{M} \) is a Type II adversary as explained in Section 3.5.1, \( B \) gives \( s \) to \( \mathcal{M} \) at the start of the game. We note that as \( B \) knows \( s \), \( B \) is able to generate ID-based private keys for any identity; thus the game does not have to be changed for Type II adversaries. We note that \( \mathcal{M} \) is allowed to replace the certificateless public key of \( \text{ID}_I \) and/or \( \text{ID}_J \) after the test query has been issued.

If \( \mathcal{M} \) replaces the certificateless public keys of other identities and asks reveal queries, \( B \) first uses the pairing to check for matching queries to the \( H_2 \) oracle. If no matching query is found, \( B \) first generates a random value \( v \) of the output domain of \( H_2 \), inserts the available session data together with \( v \) into the \( H_2 \) table as described in Section 6.1.2 (i.e. everything including the certificateless public keys; except \( g^{rx_i} \) which \( B \) cannot compute) and returns \( v \). If \( B \) is then later asked \( H_2 \) queries containing the correct \( g^{rx_J} \) and the certificateless keys \( g^{x_i} \) and \( g^{x_j} \), \( B \) is able to tell so by using the pairing computation and completes the entries in the \( H_2 \) table wherever possible.

**Strategy 2**  
The allowed corrupt queries for the adversary are listed in Table 6.1. The simulator \( B \) wants to use the adversary \( \mathcal{M} \) to solve the computational Diffie-Hellman problem. The input for \( B \) is \( (g^a, g^b) \in G^2 \) and \( B \)'s goal is to compute \( g^{ab} \). To this end, \( B \) sets the ephemeral key of \( \text{ID}_I \) to \( g^a \) and the ephemeral key of \( \text{ID}_J \) to \( g^b \) in the test query. \( B \) uses the pairing to check whether the queries of the adversary to the \( H_2 \) oracle are valid: by computing \( e(g^a, g^b) = e(g^{ab}, g) \), \( B \) is able to identify valid queries. As soon as \( B \) find such a query, \( B \) aborts the game and returns \( g^{ab} \) as solution of the CDH challenge.

The probability that \( B \) is able to find a solution to the CDH challenge is

\[
\text{Adv}_{\mathcal{B}}^{\text{CDH}}(k) \geq \frac{\text{Adv}_{\mathcal{M}}^{\Pi}(k)}{9q_0q_1^2}
\]

As \( \mathcal{M} \) is allowed to replace the certificateless public keys of any identity, \( B \) uses the technique described in Strategy 1 to decide how to answer reveal queries and \( H_2 \) queries.

**Strategy 3 and 4**  
For Strategy 3, we want to embed the CDH challenge in \( g^{rx_J} \), because the input to other values used in the key derivation function can be corrupted by the adversary. Here, \( B \) selects the master private key \( s \leftarrow Z_p \). \( B \) is able to provide ID-based secret keys for all identities, as \( B \) is in possession
of the master secret key. Furthermore, $B$ sets the certificateless public key of $\text{ID}_I$ to $g^{x_I} = g^a$ and the ephemeral public key of party $\text{ID}_J$ to $g^{r_{IJ}} = g^b$ in session $\Pi_{IJ}^T$. If the adversary is a Type II adversary as described in Section 3.5.1, then $B$ gives $s$ to $M$ at the start of the game.

Similar to Strategy 1 and 2, $B$ checks the $H_2$ queries for entries where

$$e(g, g^{r_{IJ}}) \equiv e(g^a, g^b)$$

As soon as $B$ finds such an entry, $B$ aborts the game and returns $g^{r_{IJ}}$ as solution to the CDH challenge. The probability that this happens is lower bounded by

$$\text{Adv}^{CDH}_B(k) \geq \frac{\text{Adv}^{H}_M(k)}{9q_0q_1^2}$$

$B$ uses the techniques described in Strategy 1 to deal with replaced certificateless keys of identities other than $\text{ID}_I$. We note that $M$ is allowed to replace the certificateless public key of $\text{ID}_I$ after the test query has been issued.

We note that as Strategy 4 is symmetric to Strategy 3, its probability of success is equal to the probability of success for Strategy 3. Only $\text{ID}_I$ and $\text{ID}_J$ are exchanged and the computational BDH challenge is embedded in $g^{r_{IJ}}$ instead of $g^{r_{IJ}}$.

**Strategy 5 and 6** The allowed corrupt queries for the adversary applicable to Strategy 5 are listed in Table 6.1. The BDH challenge can only be embedded in $L_2$ if Strategy 5 is chosen, because the input to all other values used in the key derivation function can be corrupted by the adversary. To accomplish this, the simulator $B$ sets the master public key to $g^a$ and implements the $H_1$ oracle as described in Table 6.2, thus $H_1(\text{ID}_I) = g^b$. $B$ patches the $H_3$ oracle as described in Table 6.3, thus $H_3(H_1(\text{ID}_I)) = H_3(g^b) = g^{3w_{1a} + 1}/g^z$. $B$ can still generate private keys for all identities except $\text{ID}_I$ by computing $H(\text{ID}_i)^s = (g^a)^{t_i}$ and $H_3(H_1(\text{ID}_i))^s = (g^a)^{y_i}$. Additionally, $B$ sets the certificateless public key of $\text{ID}_J$ to $g^c$.

A problem for $B$ arises when the adversary asks session key reveal queries for other sessions than the test session that include $\text{ID}_I$ and $\text{ID}_J$, or for sessions that include $\text{ID}_J$ and another party for which the adversary issued a replace public key query. Whenever $B$ is asked a reveal query, $B$ first checks if the key derivation function $H_2$ was asked with a matching session string involving both $\text{ID}_I$ and $\text{ID}_J$. 
As $B$ is unable to compute $L$, $B$ uses the twin bilinear Diffie-Hellman trapdoor (see Theorem 2.1) to check if $M$ submitted a valid query, i.e. if the query should be answered with a record from $H_2$ (if such a record exists). The simulator extracts the discrete logarithm for $ID_J$’s private keys, $l_J$ and $y_J$, from the $H_1$ and $H_3$ oracle respectively ($H_3(H_1(ID_J)) = H_3(g^{l_J}) = g^{y_J}$ and $B$ is able to extract both $l_J$ and $y_J$). Then, $B$ extracts $L$ and $L'$ from each entry that matches the session for which the reveal query is being asked, computes $L_1 = e((g^a)^{l_J}, g^{x_I}), L_1' = e((g^a)^{y_J}, g^{x_I})$ and checks if

\[
\left( \frac{L}{L_1} \right)^z \cdot \frac{L'}{L_1'} = \left( \frac{e(H_1(ID_J), g)^{x_I} \cdot e(H_1(ID_J), g)^{x_J}}{e(g^{lJa}, g^{x_I})} \cdot e(H_3(H_1(ID_J)), g)^{x_I} \cdot e(H_3(H_1(ID_J)), g)^{x_J}}{e(g^{yJa}, g^{x_I})} \right)^z \\
= \left( \frac{e(g^{lJ}, g^{a})^{x_I} \cdot e(g^b, g)^{ac}}{e(g^{lJa}, g^{x_I})} \cdot e(g^{yJ}, g)^{x_I} \cdot e(g^{yJa}, g^{x_I}, g)^{ac}}{e(g^{yJa}, g^{x_I})} \right)^z \\
= e(g^b, g)^{acz_I} \cdot e(g^{yJa}, g^{x_I})^{ac} \\
= e(g, g)^{z_Iabc} \cdot e(g, g)^{yJa} \cdot e(g, g)^{yJa} = e(g, g)^{yJa}^{ac} \\
= e(g^a, g^c)^{yJa}
\]

As soon as $M$ submits such an entry to the $H_2$ oracle, $B$ aborts the game and returns

\[
\frac{L}{L_1} = \frac{e(H_1(ID_J), g)^{x_I} \cdot e(H_1(ID_J), g)^{x_J}}{e(g^{lJa}, g^{x_I})} = \frac{e(g^{lJ}, g^{a})^{x_I} \cdot e(g^b, g)^{ac}}{e(g^{lJa}, g^{x_I})}
\]

as solution to the BDH challenge.

$B$ uses the same strategy for reveal queries to sessions of $ID_I$ where the adversary replaced the certificateless public key of $ID_J$, except that $B$ does not abort the game if a matching $H_2$ query is found but returns the correct $H_2$ value. If no matching $H_2$ query is found, $B$ proceeds as described in Section “Patching the $H_2$ oracle” on page 160. If the adversary replaces the certificateless public key of $ID_I$, $B$ additionally uses the technique described in Strategy 1. We note that $M$ is allowed to replace the certificateless public key of $ID_J$ after the test query has been issued.
Chapter 6. Certificateless Key Agreement Schemes

The probability that $B$ is able to find a solution to the CBDH challenge is

$$
\text{Adv}_{B}^{CBDH}(k) \geq \frac{\text{Adv}_{M}^{\Pi}(k)}{g_{q_{0}q_{1}^{2}}}
$$

Strategy 6 is symmetric to Strategy 5, so it has the same probability (only $\text{ID}_I$ and $\text{ID}_J$ are exchanged). The BDH challenge is embedded in $L_1$ instead of $L_2$.

**Strategy 7 and 8** The BDH challenge can only be embedded in $K_2$, because the input to all other values used in the key derivation function can be corrupted by the adversary. Using this strategy, the simulator sets the master public key $g^s$ to $g^a$ (notice that $B$ does not know $s$). $B$ changes the mode of operation of the $H_1$ oracle so that $H_1(\text{ID}_I) = g^b$. $B$ patches the $H_3$ oracle as described in Table 6.3, thus $H_3(H_1(\text{ID}_I)) = H_3(g^b) = g^{\text{tbdh} / y^b}$. $B$ can still generate private keys for all identities except $\text{ID}_I$ by computing $H(\text{ID}_i) = (g^a)^{l_i}$ and $H_3(H_1(\text{ID}_i)) = (g^a)^{y_i}$. As queries for $\text{ID}_I$’s private keys were ruled out by the selected strategy, this does not affect the overall success probability. Additionally, $B$ sets the ephemeral public key of party $J \neq I$ that participates in the $T^{th}$ oracle $\Pi_{T,I,J}$ to $g^c$.

If the adversary has an advantage in this strategy, then $M$ needs to query the $H_2$ oracle on the session key. To distinguish this entry from other $H_2$ queries, $B$ re-computes $K_1 = e(g^a, g)^{y_{T,I}}$ and similarly the $K'_1 = e(g^a, g)^{y_{T,I}}$. Then, $B$ searches in the table of the $H_2$ oracle for an entry where

$$
\left( \frac{K}{K_1} \right)^z \cdot \frac{K'}{K'_1} = e(g^a, g^c)^{y_{tbdh}}
$$

$B$ aborts the game as soon as such an entry is submitted to the $H_2$ oracle and returns $K/K_1$ as solution to the computational bilinear Diffie-Hellman challenge. The probability that this happens is lower bounded by

$$
\text{Adv}_{B}^{CBDH}(k) \geq \frac{\text{Adv}_{M}^{\Pi}(k)}{g_{q_{0}q_{1}^{2}}}
$$

A problem for $B$ occurs if $M$ replaces certificateless public keys. As $B$ knows the ID-based private keys for all identities except $\text{ID}_I$, $B$ can compute $K, K', L, L', N$ and $N'$ for any session except for sessions involving $\text{ID}_I$. $B$ may be unable to compute $g^{x_ix_j}$ if $M$ replaced both $g^{x_i}$ and $g^{x_j}$ but can use the pairing as described
in Strategy 1. For reveal queries involving $\text{ID}_I$ and replaced certificateless public keys, $\mathcal{B}$ uses the $H_3$ oracle as described in Strategy 5.

Strategy 8 is symmetric to Strategy 7, so it has the same probability (only $\text{ID}_I$ and $\text{ID}_J$ are exchanged). The BDH challenge is embedded in $K_1$ instead of $K_2$.

**Strategy 9**  
The BDH challenge will be embedded in $N$. To accomplish this, the simulator sets the master secret key to $g^a$, $H_1(\text{ID}_I) = g^b$, and $H_1(\text{ID}_J) = g^c$. Additionally, the $H_3$ oracle (see Table 6.3) is modified before the game starts so that $H_3(H_1(\text{ID}_I)) = H_3(g^b) = g^{\text{tbh}_1}/g^{z_1}$ and $H_3(H_1(\text{ID}_J)) = H_3(g^c) = g^{\text{tbh}_2}/g^{z_2}$.

A problem for $\mathcal{B}$ arises when the adversary asks session key reveal queries for other sessions than the test session that include $\text{ID}_I$ and $\text{ID}_J$, or for sessions where the adversary $\mathcal{M}$ replaces the certificateless public keys of any of the target identities. In these cases the simulator is unable to compute neither $N$ nor $L$. Whenever $\mathcal{B}$ is asked a session key reveal query, $\mathcal{B}$ first checks if $H_2$ was asked with a matching session string involving both $\text{ID}_I$ and $\text{ID}_J$. As $\mathcal{B}$ is generally unable to compute either $L$ or $N$, $\mathcal{B}$ uses the trapdoor as explained in Theorem 2.3 for $N$ and Theorem 2.2 for $L$ to check if $\mathcal{M}$ submitted a valid query, i.e. if the query should be answered with a record from $H_2$ (if such a record exists). To this end, $\mathcal{B}$ extracts $L$, $L'$, $N$ and $N'$ from each entry that matches the session for which the reveal query is being asked, and checks if

$$\frac{N'}{N^2} = \frac{e(g^{a}, g)^{\text{tbh}_1} e(g^{c}, g^a)^{\text{tbh}_2}}{e(g^{a}, g)^{\text{tbh}_1} e(g^{c}, g^a)^{\text{tbh}_2}}$$
and if \( L^2 L' = e(g^a, g^{x_1})^{y_{bdh2}} e(g^a, g^{x_2})^{y_{bdh1}} \). To verify this, note that

\[
N = e(g, g)^{abc} \\
N' = e(g, g)^{a(y_{bdh1} - b) y_{bdh2} - cz} \\
= e(g, g)^{a(y_{bdh1} y_{bdh2} - y_{bdh1} cz - b z y_{bdh2} + b c z^2)} \\
= e(g, g)^{abc z^2 + a y_{bdh1} y_{bdh2} - y_{bdh1} z a c - y_{bdh2} z a b} \\
\Rightarrow N^{-2} \cdot N' = (e(g, g)^{abc})^{-2} \cdot e(g, g)^{abc z^2 + a y_{bdh1} y_{bdh2} - y_{bdh1} z a c - y_{bdh2} z a b} \\
= e(g, g)^{a y_{bdh1} y_{bdh2} - y_{bdh1} z a c - y_{bdh2} z a b} \\
\frac{e(g^a, g^b)^{y_{bdh1} y_{bdh2}}}{e(g^a, g^b)^{y_{bdh1} z} \cdot e(g^a, g^b)^{y_{bdh2} z}} \\
L = e(H_1(\{ID\}), g)^{xz_1} e(H_1(\{ID\}), g)^{xz_2} \\
L' = e(H_1(\{ID\}), g)^{xz_1} e(H_1(\{ID\}), g)^{xz_2} \\
= e(g^c, g)^{ax_1} e(g^b, g)^{ax_2} = e(g, g)^{x_1 ac} e(g, g)^{x_2 ab} \\
L' = e(g, g)^{ax_1 y_{bdh1} - ax_1 z c} e(g, g)^{ax_1 y_{bdh2} - ax_1 z b} \\
\Rightarrow L^2 \cdot L' = (e(g, g)^{x_1 ac} e(g, g)^{x_1 ab})^z \cdot e(g, g)^{ax_1 y_{bdh2} - ax_1 z c} e(g, g)^{ax_1 y_{bdh1} - ax_1 z b} \\
= e(g, g)^{x_1 ac} e(g, g)^{x_1 ab} \cdot \frac{e(g, g)^{ax_1 y_{bdh2}} e(g, g)^{ax_1 y_{bdh1}}}{e(g, g)^{ax_1 z c} \cdot e(g, g)^{ax_1 z b}} \\
= e(g, g)^{x_1 ac - x_1 z a c} e(g, g)^{x_1 ab - x_1 z a b} \cdot e(g, g)^{ax_1 y_{bdh1} y_{bdh2} e(g^a, g^{x_1})^{y_{bdh1}} e(g^a, g^{x_1})^{y_{bdh2}}.}

If no matching record exists, \( B \) patches the \( H_2 \) oracle as explained in Section “Patching the \( H_2 \) oracle” on page 160. As soon as \( M \) submits such an entry to the \( H_2 \) oracle, \( B \) aborts the game and returns \( N \) as solution to the BDH challenge. The probability that this happens is lower bounded by

\[ \text{Adv}^C_{BDH}(k) \geq \frac{\text{Adv}^H_{M}(k)}{q_{y_{bdh}}} \]

\( B \) is able to distinguish between \( H_2 \) queries that have correct session data and \( H_2 \) queries that have invalid session data and is thus able to operate the \( H_2 \) oracle consistently. \( B \) may have to use the techniques explained in Strategy 1
and Strategy 5 to operate the $H_2$ oracle.

Theorem 6.1 follows from the above strategies.

\[ \square \]

6.1.3 Summary

We give the first construction that is secure the strongest security model for certificateless key agreement, the $e^2$CK model. We prove our strongly secure one round certificateless key agreement scheme secure in the random oracle model, if the computational bilinear Diffie-Hellman and the computational Diffie-Hellman assumptions hold. This enables us to answer three open questions that Swanson posed in her Master’s thesis [Swa08, Chapter 7]. The first question, whether it is even possible to construct a certificateless key agreement scheme that meets the $e^2$CK model, can be answered positively. Our protocol is compatible with existing identity-based key infrastructures and can thus be deployed easily. It is furthermore a natural complement to certificateless encryption, which brings us to Swanson’s second question: we show that a practical protocol for CL-AKE exists, although it is computationally expensive. We also show how the computational cost can be reduced if we use gap assumptions. We prove our scheme to be more secure than ID-based schemes, in the sense that the trusted authority can be more actively trying to learn secrets. To answer Swanson’s third question, whether the flexibility of certificateless schemes is worth the increased likeliness of vulnerabilities, we note that the ability of the adversary to replace public keys does not necessarily have to introduce vulnerabilities. CL-AKE schemes therefore combine user flexibility with enhanced privacy.

In the next section we give a computationally more efficient one round protocol for certificateless key agreement in the standard model that is proven secure with respect to the DDH problem and the problem used in the certificateless KEM that is run as a sub-protocol in the key agreement scheme.

6.2 Certificateless Key Agreement in the Standard Model

The strongly secure scheme in the previous section answered many open questions for certificateless key agreement schemes and showed that certificateless key agreement schemes are practical. The proof of security for the scheme is in the
random oracle model and it would be interesting to see if standard model secure schemes exist. In this section, we will answer this question positively and present a certificateless key agreement scheme in the standard model that is secure in the \(e^2\)CK model described in Section 3.5.1.

To obtain such a key agreement scheme, we modify the generic construction for key agreement by Boyd et al. [BCGP08]. The construction uses two instances of a key encapsulation mechanism to let the parties agree on a common key. To use the construction as a certificateless key agreement scheme, we modify the construction in two ways. First, we use a certificateless key encapsulation mechanism as discussed in Chapter 5 in the key agreement phase. Our second modification is the extension of the construction to the \(e^2\)CK model for certificateless key agreement. To do this, we employ a modified version of the “NAXOS-trick”: we use a pseudorandom permutation to derive the randomness used during key agreement from the ephemeral keys and the private keys of each party.

Consider a key encapsulation mechanism (KEM) as described in Definition 2.29. To make the IND-CCA game not trivial to win for the adversary, a KEM must use one or multiple sources of randomness in the encapsulation process. We write \((C, K) \leftarrow \text{KEM Encap}(pk, r)\) where \(r\) is the randomness used during the encapsulation. In the case of the KEM discussed in Chapter 5, \(r\) is a random element from \(Z_p^*\). To be able to prove the scheme in the \(e^2\)CK model, we have to be able to reveal the randomness used in the protocol run. Therefore, we specify the random value used in the key encapsulation algorithm in the algorithm description.

### 6.2.1 An \(e^2\)CK Secure Certificateless AKE in the Standard Model

We recall the generic AKE protocol constructions from KEM schemes by Boyd et al. [BCGP08] from ACISP’08. Boyd et al. give two constructions which they prove secure in the CK model. The first scheme does not offer weak perfect forward secrecy (wPFS) as described in Section 3.5 but offers key compromise impersonation (KCI) resistance as described in Section 3.5. The second scheme adds an additional Diffie-Hellman to the first protocol and then achieves both weak perfect forward secrecy and KCI resistance. Boyd et al. [BCGP08] prove these protocols secure in the Canetti-Krawczyk model for authenticated key agreement.
6.2. Certificateless Key Agreement in the Standard Model

Figure 6.2: Our e²CK secure certificateless key agreement protocol in the standard model.

even if the long term secret of one party is known to the adversary.

We extend the second protocol to construct a weakly e²CK-secure certificateless key agreement scheme in the standard model, by replacing the KEM scheme in the Boyd et al. [BCGP08] construction with a certificateless KEM (CL-KEM), for example the KEM scheme that formed the basis of the previous chapter. The only requirement on the scheme is that the randomness used during encapsulation can be a separate input to the encapsulation algorithm. We also show how to prove a variation of their protocol secure in the e²CK model. The resulting scheme is shown in Figure 6.2. We explain the protocol now in more detail.

The protocol in Figure 6.2, running between ID’s A and B consists of the following steps. We focus on A’s protocol run.

- Each party obtains an ID-based private key \( d_{ID} \) from the trusted authority and generates a certificateless private key \( x_{ID} \) on its own. The party publishes the resulting certificateless public key \( upk_{ID} \). Additionally, each party generates a long-term private key \( k_{ID} \) for the random permutation function \( H \). \( k_{ID} \) is split into two parts \( k_{ID_d} \) and \( k_{ID_{cl}} \) such that \( k_{ID} = k_{ID_d} \oplus k_{ID_{cl}} \). \( k_{ID_d} \) is stored with the ID-based secret key \( d_{ID} \) and \( k_{ID_{cl}} \) is stored with the certificateless private key \( x_{ID} \). A now has the secrets \( (d_A, k_{IA_d}) \) and \( (x_A, k_{IA_{cl}}) \). A’s certificateless public key is \( upk_A \). B’s certificateless public key is \( upk_B \). The system parameters include a string \( \kappa \) that is used for the randomness extraction function.
Chapter 6. Certificateless Key Agreement Schemes

• At the start of the protocol run, each party generates ephemeral secrets $e_{ID} \leftarrow \{0, 1\}^n$ and $y_{ID} \leftarrow Z_p$. The randomness that is used in the KEM is derived by computing $r_{ID} = H_{k_{ID}}(e_{ID}) = H_{k_{ID}} \oplus_{k_{IDCL}}(e_{ID})$, and the Diffie-Hellman key exchange value is computed as $Y_{ID} = g^{y_{ID}}$. A thus obtains the ephemeral secrets $e_A, y_A$ and later uses $r_A$ as input to the CL-KEM.

• Then, each party generates an encapsulation under a certificateless KEM using the public key of the partner party $ID_{\text{partner}}$ and the randomness $r_{ID}$ obtained in the previous step: $(K'_{ID},C_{ID}) \leftarrow \text{CL-KEM}(mpk, ID_{\text{partner}}, upk_{ID\text{partner}}, r_{ID})$. ID and $C_{ID}$ are sent to the partner ID. Thus A computes $(K'_A, C_A) \leftarrow \text{CL-KEM}(mpk, B, upk_B, r_A)$ and sends $(A, C_A, Y_A)$ to B.

• Upon receiving $(B, C_B, Y_B)$ from B, A uses $d_A, x_A$ to decapsulate $C_B$: $K'_B \leftarrow \text{CL-KEM Dec}(mpk, d_A, x_A, C_B)$.

• A computes $K''_B = \text{Exct}_\kappa(K'_B)$, $K''_A = \text{Exct}_\kappa(K'_A)$ and $K''_{AB} = \text{Exct}_\kappa(Y_B^{y_A})$ and sets $s = A \circ C_A \circ Y_A \circ B \circ C_B \circ Y_B$. Here, $\text{Exct}_\kappa(\cdot): K \to U_1$ is a randomly chosen strong $(m, \epsilon)$-randomness extractor as in Definition 2.10 for appropriate $m$ and $\epsilon$. $\kappa \leftarrow \{0, 1\}^d$ is a system-wide randomly generated string associated with $\text{Exct}$.

• Finally, the session key $K_A$ is computed by $K_A = \text{Expd}_{K''_A}(s) \oplus \text{Expd}_{K''_B}(s) \oplus \text{Expd}_{K''_{AB}}$ is derived by using the pseudorandom function family $\{\text{Expd}_\kappa(\cdot)\}_{\kappa \in U_1}: \{0, 1\}^g \to U_2$ as defined in Definition 2.11.

6.2.2 Proving the Protocol Secure

In Section 6.2.3 we use a series of games to prove the protocol secure. Compared to the proof by Boyd et al. [BCGP08], the proof is restructured to be easier to follow and to include the necessary changes for the certificateless model and the eCK model. We give a short overview of the changes that are required to prove the scheme secure against a certificateless adversary here. Recall that the weak certificateless adversary may not ask session key reveal queries for sessions at a party where the adversary replaced that party’s certificateless public key nor may these queries be asked for the test session.

We give some intuition how we modify the proof for the Boyd et al. [BCGP08] protocol to make it work in the eCK model. In a CL-KEM setting, there are only two secrets per party considered, as a KEM is a “receive only” protocol. The
ephemeral secret used to construct the message is never disclosed to the adversary in a KEM protocol. The security of the KEM holds in the certificateless case if a party has at least one uncompromised secret, i.e. an uncompromised ID-based key or an uncompromised user secret key. Now consider the case where a AKE protocol uses two instances of a CL-KEM. Let us assume that the test session runs between party $A$ with ID-based private key $(d_A, k_{A_{ID}})$ and user secret key $(x_A, k_{A_{CL}})$ and party $B$ with ID-based private key $(d_B, k_{B_{ID}})$ and user secret key $(x_B, k_{B_{CL}})$. There are now essentially four cases to distinguish:

1. A weak Type I $e^2$CK-adversary $M$ that corrupts both long-term secrets $(d_A, k_{A_{ID}})$ and $(x_A, k_{A_{CL}})$ of party $A$ and one long-term secret of party $B$. In this case the security of the KEM at party $B$ guarantees the security of the protocol. Note that in this case, the adversary is not allowed to learn the ephemeral secret of party $A$. Therefore, the randomness used in the KEM message from $A$ to $B$ is not known to the adversary, and the adversary is not able to decrypt this KEM message as it does not know both of $B$’s private keys. Therefore, the KEM at party $B$ guarantees the security in this case. This is applicable to strategy 4 and 8 in Table 6.4.

2. A weak Type I $e^2$CK-adversary $M$ that corrupts only one long-term secret of party $A$ but both long-term secrets $(d_B, k_{B_{ID}}), (x_B, k_{B_{CL}})$ of party $B$. In this case the security of the KEM at party $A$ guarantees the security of the protocol, using the same reasoning as in the previous case. This is applicable to strategy 3 and 7 in Table 6.4.

3. The case where the $e^2$CK-adversary corrupts all long-term keys $(d_A, k_{A_{ID}}), (x_A, k_{A_{CL}}), (d_B, k_{B_{ID}}), (x_B, k_{B_{CL}})$ of the respective parties. In this case the additional Diffie-Hellman allows us to prove the security of the protocol. This is applicable to strategy 2 in Table 6.4.

4. The case where the $e^2$CK-adversary corrupts all ephemeral keys, but only one long-term key per party. In this case, the splitting of the key $k_{ID} = k_{ID_{ID}} \oplus k_{ID_{CL}}$ to the random permutation function $H$ allows us to base the security of the protocol on the security KEM at either party. Since the adversary is unable to reconstruct $k_{ID}$ given only $k_{ID_{ID}}$ or $k_{ID_{CL}}$, the security of the random permutation function guarantees that the input to the KEM encapsulation function is indistinguishable from random, even if the
respective ephemeral secrets are known. Then, because the adversary cannot decrypt the KEM messages of either party, it cannot learn either key encapsulated in the KEM. Therefore, the KEM guarantees the security of the protocol. This is applicable to strategy 1, 5, 6, and 9 in Table 6.4.

6.2.3 Proof of Security in the e²CK Model

We use a game-hopping proof similar to the proof in Boyd et al. [BCGP08]. However, since we want to prove security in the e²CK model, we restructure the proof and switch to the notation that we also used in the proof of the previous section.

For our protocol we propose the following theorem similar to Boyd et al. [BCGP08]:

**Theorem 6.2.** Let $\mathcal{M}$ be any adversary against our protocol. Then the advantage of $\mathcal{M}$ against the security of our protocol is:

$$\text{Adv}_{\text{sk}}^{M}(k) \leq 2^2q^2\left(2p + \text{Adv}_{F,C}^{p-\text{rand}}(k) + \text{Adv}_{KEM,B}^{\text{CL-KEM-CCA}}(k),
\text{Adv}_{G,B}^{\text{DDH}}(k) + \epsilon + \text{Adv}_{F,C}^{p-\text{rand}}(k)\right)$$

where $\mathcal{B}$ is the simulator for the key agreement protocol that $\mathcal{M}$ interacts with. $\text{Adv}_{G,B}^{\text{DDH}}(k)$ is the advantage that $\mathcal{B}$ has against the decisional Diffie-Hellman problem discussed in Definition 2.6 as a result of interacting with $\mathcal{M}$, $\text{Adv}_{F,B}^{p-\text{rand}}(k)$ is the advantage that $\mathcal{B}$ has against the pseudorandom function family discussed in Definition 2.11, $\epsilon$ is the advantage that $\mathcal{M}$ has against the strong randomness extractor discussed in Definition 2.10, and $\text{Adv}_{KEM,B}^{\text{CL-KEM-CCA}}(k)$ is the advantage that $\mathcal{B}$ has against the certificateless KEM as discussed in Section 3.4.2.

In the proof, the adversary is allowed to corrupt all but one secret per party. That means that each party holds at least one uncorrupted/unrevealed secret. If this was not the case, the adversary would be able to trivially break the protocol. All possible reveal queries are shown in Table 6.4.

The proof is split into three parts. In the first part we consider the security of the KEM as a basis for the security of the key agreement scheme. As discussed in Section 6.2.2, we show that the security of the KEM suffices for strategies 1,
6.2. Certificateless Key Agreement in the Standard Model

<table>
<thead>
<tr>
<th>Strategy</th>
<th>1</th>
<th>2</th>
<th>3/4 (mirr.)</th>
<th>5/6 (mirr.)</th>
<th>7/8 (mirr.)</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secret / Party</td>
<td>A B</td>
<td>A B</td>
<td>A B</td>
<td>A B</td>
<td>A B</td>
<td>A B</td>
</tr>
<tr>
<td>ID-based ((a_{ID}, k_{ID}))</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Certificateless ((x_{ID}, k_{CL}))</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Ephemeral ((e_{ID}, y_{ID}))</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

\(x = \text{secret that must not be revealed, mirr. = swap columns A and B for this strategy.}\)

Table 6.4: Possible corrupt queries sorted by strategy

3/4, 5/6, 7/8, and 9. In the second part, we show that the additional Diffie-Hellman suffices for strategy 2. In the last part, we show that the adversary cannot learn anything about the session key as long as either the KEM is secure or the Diffie-Hellman key exchange is secure. This allows us then to conclude the proof.

The eCK model proposed by LaMacchia et al. [LLM07] considers sessions to be enumerated with session identifiers \(\text{sid} = (\text{role, ID, ID}^*, \text{comm}_1, \ldots, \text{comm}_n)\) where \(\text{comm}_i\) refers to the transcript of the communication between \(\text{ID}\) and \(\text{ID}^*\). Since these session identifiers do not allow sessions to be enumerated easily, we revert to BR-style session identifiers. We assume that there is a set of parties \(P\) between which the protocol runs, with \(|P| = l\). Furthermore, let \(q\) be the maximum number of sessions that any one party may have. Then a session is uniquely identified by \(\Pi_{ID_i, ID_j}^{t_i, t_j}\), where \(t_i, t_j \leq q\), and \(ID_i, ID_j \in P\). Note that there may not be a matching session activated at \(ID_j\). In the following, we need to have an upper limit on the number of sessions that can be run. Apparently, there are \(l\) possibilities for the first party \(ID_i\) and then \((l - 1)\) possibilities for the second party \(ID_j\). Since each party can run at most \(q\) sessions, we have that there are then at most \(l(l - 1)q^2 < l^2q^2\) possibilities to select the test session.

We consider the following series of games to prove Theorem 6.2.

**Game 0.** This game is the same as a real interaction with the protocol. A random bit \(b\) is chosen, and when \(b = 0\), the real key is returned by \(B\) in answer to \(M\)'s test session query; otherwise a random key from \(U_2\) (the range of the pseudorandom function family) is returned.

**Game 1.** This game is the same as the previous one, except that before the adversary begins, two random values \(A, B \in P\) are chosen, and a random \(T_A, T_B \in \{1, \ldots, q\}\) are chosen by \(B\). We call the session \(\Pi_{A, B}^{T_A, T_B}\) the test session. If the test session is not guessed correctly, the protocol halts.
At this stage, we split the proof into two avenues. In the first avenue discussed in Game 2 to Game 4, we build the security of the key agreement protocol on the fact that at least one of the KEMs is still secure. That means that this avenue is applicable for strategies 1, 3, 4, 5, 6, 7, 8, and 9 shown in Table 6.4.

In the other avenue, the security of the key agreement protocol is based on the hardness to solve the decisional Diffie-Hellman problem. This avenue is discussed in Games 1 and 5 below.

In the following games (Game 6 and Game 7) we ensure that the adversary is unable to compute the correct session key if security holds in any one of these two avenues, which cover all possible strategies to attack the key agreement scheme.

**Game 2.** This game is the same as Game 1, except that if two different sessions output exactly the same KEM message and have the same intended partner, the protocol halts. This game guarantees that the ciphertext output by the KEM in the test session is unique.

**Game 3.** The party whose long-term secrets must not be fully revealed is randomly chosen by the simulator, let’s assume in the following that this is party A. The randomness used in the KEM message from B to A scheme is not derived from the ephemeral value $e_{B,T_B}$ and the secret keys $(d_B, k_{B_B}), (x_B, k_{B_{Cl}})$, but independent randomness is used instead. So we replace $r^*_{B} \leftarrow H_{k_B}(e_{B,T_B})$ with a random $r^*_{B}$. Therefore, the KEM can be treated as a black box in the following games, even if the adversary chooses to reveal the ephemeral secrets. The adversary’s probability of detecting this change is limited by the security of the pseudorandom function family $H$.

**Game 4.** Instead of using the key encapsulated in the KEM message from B to A in the test session, a random key $K^*_B$ is used. Assuming that at least one of the long term secrets of either party is not revealed by the adversary, the adversary’s probability of detecting this change is limited by the security of the KEM. In the previous game, we established that the randomness used in the KEM is not related to the ephemeral secrets, so the adversary has no increased advantage in detecting this even when the adversary learns the ephemeral secret of party B.

The rest of the Game 4 is the same as Game 3. If $b = 1$, a random key from $U_2$ is returned. Otherwise, $K^*_B$ is used in the computation of the test session as described above.
This is the end of the first avenue. In Game 6 we continue to analyze the success probability of the adversary for the remaining steps in the protocol.

The following game establishes the security of the key agreement protocol based on the hardness of the DDH problem. This game is only used in the case that all long term keys of both parties are revealed by the adversary, so it is only relevant for strategy 2 from Table 6.4.

**Game 5.** This game is the same as Game 1, except that the key \( K_{A,B}^{r*} \) is replaced by a random value. The probability that the adversary detects this change is limited by \( \mathcal{M} \)'s advantage against the DDH problem.

The rest of the Game 5 is the same as Game 1. If \( b = 1 \), a random key from \( U_2 \) is returned. Otherwise, \( K_{ID_A, ID_B}^{r*} \) is then used in the computation of the test session.

This is the end of the second avenue. In Game 6 we continue to analyze the success probability of the adversary for the remaining steps in the protocol.

In the final two games, we show that the security of the key agreement scheme holds as long as either the KEM is secure or the Diffie-Hellman is correct. To show this, we replace the output of the extract function and then the output of the expand function with random values. After the last game, the key is completely random, so \( b \) is independent of the session key. Therefore, the adversary cannot have any advantage in distinguishing the session key from a random key. Finally, we sum the advantages of the adversary in distinguishing the session key up and can thus show that the probability of the adversary in breaking the protocol is negligible if the adversary has no advantage against the components that we use in the protocol.

**Game 6.** This game is the same as the previous one, except that a random value \( K_{rs} \) \( \xleftarrow{\$} U_1 \) is chosen and the use of \( \text{Exct}(K_{rs}) \) is replaced with \( K_{rs}^{r*} \).

**Game 7.** This game is the same as the previous one, except that whenever the value \( \text{Expd}_{K_{rs}}(s') \) for any \( s' \) would be used in generating keys, a random value from \( U_2 \) is used instead (the same random value is used for the same value of \( s' \); a different random value is chosen for different \( s' \)).
We use the following notation.

\[ \sigma_i = \text{The event that } M \text{ guesses the value of } b \text{ correctly in Game } i \]  

(6.1)

\[ \tau_i = \left| 2 \Pr[\sigma_i] - 1 \right| = \text{Advantage of } M \text{ in Game } i \]  

(6.2)

We observe that if Games \( i \) and \( i + 1 \) are identical when event \( E \) does not occur, and if there is a probability of \( \frac{1}{2} \) that \( B \) is correct in Game \( i + 1 \) when \( E \) does occur, then we have

\[
\Pr[\sigma_{i+1}] = \Pr[\sigma_{i+1}|E]\Pr[E] + \Pr[\sigma_{i+1}|\neg E]\Pr[\neg E]
\]

(6.3)

\[
= \frac{1}{2}\Pr[E] + \Pr[\sigma_i|\neg E]\Pr[\neg E]
\]

\[
\Pr[\sigma_i] - \Pr[\sigma_{i+1}] = \Pr[\sigma_i|E]\Pr[E] - \frac{1}{2}\Pr[E]
\]

(6.4)

\[
\left| \Pr[\sigma_i] - \Pr[\sigma_{i+1}] \right| \leq \frac{1}{2}\Pr[E]
\]

(6.5)

Consider an event \( E \) that may occur during \( B \)'s execution such that \( E \) is detectable by the simulator, \( E \) is independent of \( \sigma_i \), Game \( i \) and Game \( i + 1 \) are identical unless \( E \) occurs, and \( \Pr[\sigma_{i+1}|E] = \frac{1}{2} \). Then we have:

\[
\Pr[\sigma_{i+1}] = \Pr[\sigma_{i+1}|E]\Pr[E] + \Pr[\sigma_{i+1}|\neg E]\Pr[\neg E]
\]

(6.6)

\[
= \frac{1}{2}\Pr[E] + \Pr[\sigma_i|\neg E]\Pr[\neg E]
\]

\[
= \frac{1}{2}(1 - \Pr[\neg E]) + \Pr[\sigma_i]\Pr[\neg E]
\]

\[
= \frac{1}{2} + \Pr[\neg E]\left(\Pr[\sigma_i] - \frac{1}{2}\right)
\]

Hence, \( \tau_{i+1} = 2 \left| \Pr[\sigma_{i+1}] - \frac{1}{2} \right| \)

(6.6)

\[
= 2 \left| \Pr[\neg E]\left(\Pr[\sigma_i] - \frac{1}{2}\right) \right|
\]

(6.7)

\[
\overset{(6.2)}{=} \Pr[\neg E]\tau_i.
\]
Analysis of Game 1:

In Game 1, the probability of the protocol halting due to an incorrect choice of $\Pi_{A,B}^{T_A,T_B}$ is $1 - \frac{1}{l^2q^2}$. Therefore we have

$$\tau_1 = \frac{1}{l^2q^2} \tau_0 \Rightarrow l^2q^2 \tau_1 = \tau_0. \quad (6.8)$$

Analysis of Game 2:

Let $\frac{1}{p}$ be the maximum probability that $C_1 = C_2$ where $(C_1, K_1) \overset{\$}{\leftarrow} \text{enc}(pk, ID)$ and $(C_2, K_2) \overset{\$}{\leftarrow} \text{enc}(pk, ID)$ for any identity ID. Let $p_{\text{sameMsg}}$ be the probability of two or more sessions output the same KEM ciphertext. Since there are two KEM messages exchanged per session run, we have

$$1 - p_{\text{sameMsg}} > 1 - \frac{(l^2q^2)^2}{p}.$$

Then by Equation (6.5) we have that

$$|\Pr[\sigma_1] - \Pr[\sigma_2]| < \frac{(l^2q^2)^2}{2p} \quad (6.9)$$

when $\frac{1}{2} > \frac{l^2q^2-1}{p} > 0$.

This can be used to bound $\tau_1$ as follows:

$$\tau_1 = |2\Pr[\sigma_1] - 1| \quad (6.10)$$

$$\leq 2 \left( |\Pr[\sigma_1] - \Pr[\sigma_2]| + \left| \Pr[\sigma_2] - \frac{1}{2} \right| \right) \quad (6.11)$$

$$\leq \frac{(l^2q^2)^2}{p} + \tau_2. \quad (6.12)$$

From Game 1 we have that $l^2q^2 \tau_1 = \tau_0$. Therefore we can relate the success probability in this game to the actual game by

$$\tau_0 \leq \frac{(l^2q^2)^2}{p} + l^2q^2 \tau_2. \quad (6.13)$$
Analysis of Game 3

At the start of the game, $B$ randomly selects either of $A$ or $B$ as the identity whose long-term secrets must not be fully compromised in the test session. In the following, we assume that $A$ is selected. If $B$’s guess is wrong, then $B$ aborts the game and tries again.

In all sessions except for the test session, the adversary $M$ against the protocol is allowed to corrupt all secrets involved ($(e_{ID,II}, y_{ID,II}), (x_{ID}, k_{ID_{CL}}), (d_{ID}, k_{ID_{ID}})$). In the test session however, the adversary may at most corrupt two out of the three secrets per party involved. We use this fact to our advantage in using independent randomness in this game.

For the cases where $M$ wants to learn at most one of the ephemeral secrets of the test session, we can limit $M$’s advantage by security of the pseudorandom function family $\{\text{Expd}_K(\cdot)\}_{K \in U_1}$ (see Definition 2.11). $B$ is constructed such that when it uses the real randomness $(e_{B,T_B}, y_{B,T_B})$ and the private key $k_B = k_{B_{ID}} \oplus k_{B_{CL}}$ to derive the randomness $r_{B,T_B} = H_{k_B}(e_{B,T_B}) = \text{Expd}_{k_B}(e_{B,T_B})$ used in the KEM queries, then the view of $M$ is the same as in Game 2, but if $B$ uses independent randomness in the KEM, then the view of $M$ is the same as in Game 3. Then, by the security of the pseudorandom function, we can claim that these games are indistinguishable.

The randomness $r$ is generated by letting $r = \text{Expd}_{k_B}(e_{B,T_B}) = \text{Expd}_{k_{B_{ID}} \oplus k_{B_{CL}}}(e_{B,T_B})$. Recall that when $M$ wants to learn the ephemeral secret $(e_{B,T_B}, y_{B,T_B})$ used in the test session, then $M$ may at most corrupt one other secret (either $(d_{B}, k_{B_{ID}})$ or $(x_{B}, k_{B_{CL}})$) at the same party. Since the key $k_B$ for the pseudorandom function is determined by the XOR of the two random subkeys $k_{B_{ID}}$ and $k_{B_{CL}}$, the adversary does not learn any information about $k_B$ by corrupting only one of the private secrets (see also Shannon [Sha49]).

**Lemma 6.1.** Given $(e_{B,T_B}, y_{B,T_B})$, and one of the long-term secrets (either $(d_{B}, k_{B_{ID}})$ or $(x_{B}, k_{B_{CL}})$), and the output of the KEM $C_{ID}^r$, there exists no efficient adversary that can decide whether there exists an output $r = H_{k_B}(e_{B,T_B})$ such that $r$ is the randomness used in $C_{ID}$ or if the randomness used in the KEM is independent of $e_{B,T_B}$.

**Proof.** To begin with, $B$ is given access to an oracle that given inputs $e \in \{0, 1\}^n$ returns outputs $r$ so that either $r$ is drawn randomly or that $r$ is derived from $e$ using a permutation $f_k(e)$ with unknown key $k$. $B$’s task is to decide whether $r$ is random or derived via a permutation, in analogy to Definition 2.11.
6.2. Certificateless Key Agreement in the Standard Model

\( B \) runs as described in Game 2, except that whenever a value \( H_k(e) \) for party \( B \) is required, \( B \) feeds \( e \) into the oracle and uses the output \( r \) of the oracle as the randomness for the KEM. The two games differ only in the way the randomness for the KEM is derived. \( B \) always computes the session key of the test session using the \( r \) obtained from its oracle. When \( M \) outputs it guess of the bit \( b \), \( M \) outputs that its oracle is a member of the given function family if \( M \) is correct, and \( B \) outputs that its oracle is a truly random function otherwise. Since \( B \) never has to disclose \( k_B \) fully to \( M \), the oracle can be safely substituted. The probability that \( B \) is correct is \( \frac{1}{2} (\Pr[\sigma_2] + 1 - \Pr[\sigma_3]) \).

By the security of the random function family we have:

\[
\text{Adv}_{F,B}^{b-rand}(k) \geq |2 \Pr[A \text{ correct}] - 1| = |\Pr[\sigma_2] - \Pr[\sigma_3]|. \tag{6.14}
\]

For the cases where \( M \) wants to learn both long term keys of party \( B \) and thus gains knowledge of \( k_B \), we argue that \( B \) has no advantage in determining if the randomness used in the KEM is derived from \( e_{B,T_B} \) via \( k_B \) or if independent randomness is used. This stems from the fact \( e_{B,T_B} \) is randomly chosen and never revealed, so all \( e_{B,T_B} \) are equally likely. Even if \( M \) recovers the randomness used in the KEM, it is thus not possible to judge whether it was derived correctly. Thus we argue that there exists no efficient adversary that can determine whether the randomness \( r \) is derived from \( e_{B,T_B} \) or if independent randomness is used in the KEM. Therefore we have in this case

\[ \tau_2 = \tau_3. \tag{6.15} \]

Summarizing this game, we have

\[
\tau_2 = |2 \Pr[\sigma_2] - 1| \\
\leq |2 \Pr[\sigma_2] - 2 \Pr[\sigma_3]| + |2 \Pr[\sigma_3] - 1| \\
\leq 2 \text{Adv}_{F,F}^{b-rand}(k) + \tau_3, \tag{6.16}
\]

since (6.15) does not make any contribution to the success probability of this game. This allows us to treat the KEM as a black box in the following game.
Analysis of Game 4:

We want to replace the key that is encapsulated under $\text{ID}_A$’s KEM with a random key. We claim that this change is indistinguishable unless $\mathcal{M}$ has an advantage against the security of the KEM scheme.

Recall that we established in Game 3, that party $A$ participates at the test session and that not both of $A$’s long term keys may be revealed by $\mathcal{M}$. We show how to use $\mathcal{B}$ as an adversary against the security of the CL-KEM, using $\mathcal{M}$. $\mathcal{B}$ is constructed such that when it receives the real key for the CL-KEM scheme, the view of $\mathcal{M}$ is the same as in Game 3, but if $\mathcal{B}$ receives a random CL-KEM key, the view of $\mathcal{M}$ is the same as in Game 4. Then, by the security of the CL-KEM scheme, we can claim these games are indistinguishable. This game is essential for the security of all strategies except strategy 2.

To begin, $\mathcal{B}$ is given the master public key $pk$. $\mathcal{B}$ passes this value as well as the description of $\text{Exct}()$, its key $\kappa$ and $\{\text{Ex}d_{K}(\cdot)\}_{K \in U_1}$ to $\mathcal{M}$. Recall that $\mathcal{B}$ has access to the corresponding oracles $\mathcal{O}_{\text{KeyDer}}(\cdot)$ and $\mathcal{O}_{\text{dec}}(\cdot, \cdot)$ for the KEM. By the security of the certificateless KEM, $\mathcal{B}$ is also allowed to replace user public keys if $\mathcal{M}$ desires to do so.

$\mathcal{B}$ runs as described in Game 3, except that when the test session is activated, $\mathcal{B}$ outputs $A$ as the identity for which it wants to receive the KEM challenge ciphertext. In return, $\mathcal{B}$ receives a ciphertext $C^*$ for $A$ and a key $K'^*_A$, which may be the decryption of $C^*$ or may be a random CL-KEM key, each with equal probability. $\mathcal{B}$ then uses $C^*$ as KEM message that is sent from $B$ to $A$ in session $\Pi_{A,T_A}^{T_{A,T_B}}$, modifies the calculation of keys so that $K'^*_A$ is used in place of $\text{dec}(pk, \text{KeyDer}(pk, \alpha, A), C^*)$, and uses $K'^*_A$ instead of $K'_A$ to find the answer to the test session query when $b = 0$.

All legitimate queries made by $\mathcal{B}$ can still be answered by $\mathcal{A}$ using its oracles in as follows.

- A corrupt query on some identity $\text{ID}_X \neq A$ may be answered with $\mathcal{O}_{\text{KeyDer}}(\text{ID}_X)$.
- All keys of the partner of the test session $B$ can be revealed. This query can be answered with $\mathcal{O}_{\text{KeyDer}}(B)$ for both secrets, as these queries are allowed in a certificateless KEM.
- A partial corrupt query on $A$ is also allowed by the security of the CL-KEM, and can be answered by $\mathcal{O}_{\text{KeyDer}}(A)$. A full corrupt query is not
allowed, but we discuss $B$’s strategy in the case of a full corrupt query in Game 5.

• All keys used in all sessions except the test session can be computed correctly. Since we proved in Game 3 that the adversary cannot determine which randomness is used in the KEM, the simulation of the KEM by $B$ is perfect.

• Any message $C_X \neq C^*$ to any party (including $A$) with identity $ID_X$ may be decrypted using $O_{dec}(ID_X, C_X)$ to generate keys for reveal session key queries and the test query. We established in Game 1 that $C^*$ is a unique message.

When $M$ halts and outputs its bit $b'$, $B$ halts and outputs $1 - b'$. The probability that $B$ is correct is $\Pr[\sigma_3]$ when $K^*_A$ is the real key for the CL-KEM message, and $1 - \Pr[\sigma_4]$ when $K^*_A$ is not the key for the CL-KEM message. We can then find that:

$$\text{Adv}_{CL-KEM-CCA}^{KEM, B}(k) = (6.17)$$

$$\left| 2\left(\frac{1}{2}(\Pr[\sigma_3] + 1 - \Pr[\sigma_4])\right) - 1 \right| = \Pr[\sigma_3] - \Pr[\sigma_4]. \quad (6.18)$$

Therefore we have that

$$\tau_3 \leq |2 \Pr[\sigma_3] - 1| \quad (6.20)$$

$$\tau_3 \leq |2 \Pr[\sigma_3] - 2 \Pr[\sigma_4]| + |2 \Pr[\sigma_4] - 1| \quad (6.21)$$

We deal now with the case, where the adversary chooses to reveal all long-term secrets of the parties involved in the test session. Note however that the adversary is restricted to being passive during the protocol run corresponding to the test session – a consequence of only being able to achieve weak forward secrecy in one round. As we will see below this allows us to inject a challenge Decisional Diffie-Hellman triplet into the test session.
Analysis of Game 5

In this game we want to show that the adversary has negligible probability of distinguishing whether $Y_A^y$ and $Y_B^z$ respectively are randomly chosen or computed correctly from the key exchange protocol, given that $\mathcal{M}$ may not corrupt any ephemeral secrets. To show this, we construct adversary $\mathcal{B}$ against the DDH problem, using $\mathcal{M}$. $\mathcal{B}$ is constructed such that provided $\mathcal{B}$ does not have to abort the protocol as specified in Game 1, then if $\mathcal{B}$’s input is from $\mathcal{D}_H$, the view of $\mathcal{M}$ is the same as in Game 1, but if $\mathcal{B}$’s input is from $\mathcal{R}_F$, the view of $\mathcal{M}$ is the same as in Game 5. Then, by Definition 2.6, we can claim these games are indistinguishable.

To begin, $\mathcal{B}$ generates all the protocol parameters and passes the public parameters to $\mathcal{M}$.

Let $(g^a, g^b, h)$ be $\mathcal{B}$’s challenge DDH inputs (with $h$ either $g^c$ for $c \leftarrow Z_p^*$ or $g^{ab}$). When the session $\Pi_{A,B}^{T_A,T_B}$ is activated, $\mathcal{B}$ uses its inputs $g^a$ and $g^b$ instead of the values $Y_A$ and $Y_B$ when it generates the outputs of these sessions. Apart from this change, all session inputs and outputs are generated according to the protocol specification.

When the test session query is made, $\mathcal{B}$ uses $\text{Exct}(h)$ in place of $K''_{AB}$ when calculating the real test session key.

$\mathcal{B}$ is able to answer all queries except for the ephemeral secrets used in the test session correctly since it knows all of the system parameters. $\mathcal{B}$ outputs 1 if $\mathcal{M}$ is correct, and 0 otherwise.

The probability that $\mathcal{B}$ will not have to abort the protocol as described in Game 1 is $\frac{1}{r_q}$. Hence, by the game hopping technique by Dent [Den06] we have that:

$$\tau_0 \leq 2f^2q\text{Adv}^{\text{DDH}}_{G,R}(k) + \tau_5$$  \hspace{1cm} (6.23)

Analysis of Game 6

At this point, we have that either the output of Game 4 or Game 5 is indistinguishable from random for any PPT adversary, depending on the reveal queries that the adversary uses. Let us call this combined result “Game 4 or 5”. Let us further assume that the output of “Game 4 or 5” is used in the randomness extraction function $K''^*$, where $K''^*$ is either $K''_{AB}$ (in Game 5) or $K''_A$ or $K''_B$ (in Game 4). We now consider an adversary, $\mathcal{B}$, against the security of the random-
ness extraction function. This adversary interacts with $\mathcal{M}$ in such a manner that it is the same as when $\mathcal{M}$ interacts with either “Game 4 or 5” or 6. $\mathcal{B}$ receives a key $\kappa$ for the randomness extraction function and a value $R_1$ such that either $R_1 = \text{Exct}(X)$ for some $X \in \mathcal{K}$ or $R_1 \in \mathcal{U}_1$. $\mathcal{B}$ sets $\kappa$ to be the public parameter used to key the randomness extraction function, and chooses the other public parameters according to the protocol. $\mathcal{B}$ runs as described for Game 6, except that $\mathcal{B}$ uses $R_1$ in place of $K''$. When $\mathcal{M}$ outputs its guess of the bit $b$, $\mathcal{B}$ outputs that $R_1 = \text{Exct}(X)$ for some $X$ if $\mathcal{M}$ is correct, and $\mathcal{B}$ outputs that $R_1 \in \mathcal{U}_1$ otherwise. The probability that $\mathcal{B}$ is correct is $\frac{1}{2} (\Pr[\sigma_{4 \text{ or } 5}] + 1 - \Pr[\sigma_6])$. By the security of the randomness extraction function (see Definition 2.10), we have:

$$\epsilon \geq |2 \Pr[D \text{ correct}] - 1|$$
$$= |\Pr[\sigma_{4 \text{ or } 5}] - \Pr[\sigma_6]|$$

$$\tau_{4 \text{ or } 5} = |2 \Pr[\sigma_{4 \text{ or } 5}] - 1|$$
$$\leq |2 \Pr[\sigma_{4 \text{ or } 5}] - 2 \Pr[\sigma_6]| + |2 \Pr[\sigma_6] - 1|$$
$$\leq 2\epsilon + \tau_6$$

### Analysis of Game 7

Next, we show how to build an adversary $\mathcal{B}$ against the randomness expansion (or pseudorandom) function family $\{\text{Expd}_K(\cdot)\}_{K \in \mathcal{U}_1}$ using $\mathcal{M}$. We define $\mathcal{B}$ to run a copy of $\mathcal{M}$, and to interact with $\mathcal{M}$ in such a manner that it is the same as when $\mathcal{M}$ interacts with either Game 6 or 7. $\mathcal{B}$ receives the definition of the function family $\{\text{Expd}_K(\cdot)\}_{K \in \mathcal{U}_1}$, and an oracle $\mathcal{O}(\cdot)$ which is either $\text{Expd}_K(\cdot)$ for some value of $K$ unknown to $\mathcal{B}$ or a truly random function. $\mathcal{B}$ runs a copy of the protocol for $\mathcal{M}$ in the same way as described for Game 6, except that whenever the value $\text{Exd}_{K''}(s')$ for any $s'$ would be used in generating keys, $\mathcal{B}$ uses the value $\mathcal{O}(s')$ instead. When $\mathcal{M}$ outputs its guess of the bit $b$, $\mathcal{B}$ outputs that its oracle is a member of the given function family if $\mathcal{M}$ is correct, and $\mathcal{B}$ outputs that its oracle is a truly random function otherwise. The probability that $\mathcal{B}$ is correct is $\frac{1}{2} (\Pr[\sigma_6] + 1 - \Pr[\sigma_7])$. By the security of the randomness expansion function we have:

$$\text{Adv}_{\mathcal{F},\mathcal{C}}^{\text{p-rand}}(k) \geq |2 \Pr[\mathcal{B} \text{ correct}] - 1|$$
$$= |\Pr[\sigma_6] - \Pr[\sigma_7]|$$
\[ \tau_6 = |2 \Pr[\sigma_6] - 1| \quad (6.27) \]
\[ \leq |2 \Pr[\sigma_6] - 2 \Pr[\sigma_7]| + |2 \Pr[\sigma_7] - 1| \quad (6.28) \]
\[ \leq 2 \text{Adv}^{\text{p-rand}}_{\mathcal{C}, \mathcal{C}}(k) + \tau_7 \quad (6.29) \]

In Game 7, let us denote the key returned in the test session query with \( R_1 \oplus \text{Expd}_{K_B^{s'}}(s') \oplus \text{Expd}_{K_{A,B}^{s'}}(s') \) when \( b = 0 \), and \( R_2 \) when \( b = 1 \), where \( R_1 \) and \( R_2 \) are chosen uniformly at random from \( U_2 \). We assume here that the security of the protocol is guaranteed by the KEM at \( A \), however, \( R_1 \) could be injected instead of \( \text{Expd}_{K_B^{s'}}(s') \) or \( \text{Expd}_{K_{A,B}^{s'}}(s') \) as well if the respective secrets are left intact by the adversary. Now, \( R_2 \) is chosen independently of all other values in the protocol, so \( \mathcal{M} \) can gain no information about \( R_2 \) directly; \( \mathcal{M} \) can only gain information about \( R_2 \) by determining whether \( b = 0 \) or \( b = 1 \). Furthermore, when \( b = 0 \), unless \( \mathcal{M} \) can gain some information about \( R_1 \), the response to the test session query also looks random and is therefore indistinguishable from the case when \( b = 1 \).

To gain information about \( R_1 \) from a source other than the test session query response, \( \mathcal{M} \) must obtain the key of a session that has also used \( R_1 \) in the generation of its key. Now, if \( R_1 \) is used in the generation of a session’s key, then that session must have had the same session identifier, and hence exchanged the same messages as the test session. Therefore, the session is either owned by \( A \) with intended partner \( B \) and received \( C^* \) as part of its input or is owned by \( B \) with intended partner \( A \) and had \( C^* \) as part of its output (both variants are for Game 4), or is shared by \( A \) and \( B \) and has \( g^a \) and \( g^b \) as part of its output (Game 5). However, such a session owned by either \( A \) or \( B \) or both of them will match the test session. Therefore, at least one of the secrets of both \( A \) and \( B \) must not be revealed. Hence, \( \mathcal{M} \) can gain no information about \( R_1 \), and so \( \mathcal{M} \) can gain no information about \( b \) in Game 7, and therefore

\[ \tau_7 = 0 \]

**Combining Results**

The proof starts with Game 0 and Game 1. Game 1 establishes in Equation (6.8) that \( l^2 q^2 \tau_1 = \tau_0 \). After that, there are two possible avenues for the proof, either through Game 2, 3, 4, (avenue 1); or through Game 5 instead (avenue 2). Both avenues end in the same “end games”, Game 6, and Game 7.
We now proceed to compute the advantage that an adversary for avenue 1 has. Equation (6.13) covers Game 0 to 2, and shows that \( \tau_0 \leq \frac{l^2q^2}{p} + l^2q^2 \tau_2 = l^2q^2 \left( \frac{l^2q^2}{p} + \tau_2 \right) \). Game 3 established in Equation (6.16) that \( \tau_2 \leq 2 \text{Adv}^{p-rand}_{F,C}(k) + \tau_3 \). In Game 4 we showed in Equation (6.22) that \( \tau_3 \leq 2 \text{Adv}^{CL-KEM-CCA}_{KEM,B}(k) + \tau_4 \). Putting these equations together we obtain

\[
\tau_0 \leq l^2q^2 \left( \frac{l^2q^2}{p} + 2 \text{Adv}^{p-rand}_{F,C}(k) + 2 \text{Adv}^{CL-KEM-CCA}_{KEM,B}(k) + \tau_4 \right) \quad (6.30)
\]

Avenue 2 results in \( \tau_0 \leq 2l^2q \text{Adv}^{DDH}_{G,B}(k) + \tau_5 \)

So from here we have for any adversary that

\[
\tau_0 \leq 2l^2q \left( \max \left( \frac{l^2q^2}{2p} + \text{Adv}^{p-rand}_{F,C}(k) + \text{Adv}^{CL-KEM-CCA}_{KEM,B}(k) + \tau_4, \text{Adv}^{DDH}_{G,B}(k) + \tau_5 \right) \right) \quad (6.31)
\]

Game 6 continues at this point by showing that \( \tau_4 \) or 5 \( \leq 2 \epsilon + \tau_6 \) and Game 7 concludes by showing that \( \tau_6 \leq 2 \text{Adv}^{p-rand}_{F,C}(k) + \tau_7 \). Using these two terms in Equation (6.31), we find that

\[
\tau_0 \leq 2l^2q \left( \max \left( \frac{l^2q^2}{2p} + \text{Adv}^{p-rand}_{F,C}(k) + \text{Adv}^{CL-KEM-CCA}_{KEM,B}(k), \text{Adv}^{DDH}_{G,B}(k) \right) + \epsilon + \text{Adv}^{p-rand}_{F,C}(k) \right) \quad (6.32)
\]

This concludes the proof of the protocol.

### 6.2.4 Summary

We give the first construction of an efficient certificateless key agreement scheme proven secure in the standard model. We modify the KEM-KEM construction by Boyd et al. [BCGP08] from ACISP 2008 to construct our scheme. Boyd et al. [BCGP08] prove their scheme secure in the Canetti-Krawczyk model for key agreement. We show that the new KEM-KEM construction in the certificateless setting can be proven secure in the e\(^2\)CK model for certificateless key agreement. From our result it is evident that the Boyd et al. [BCGP08] protocol can be proven in the eCK model as well using similar techniques. We present the first
Chapter 6. Certificateless Key Agreement Schemes

<table>
<thead>
<tr>
<th>AKE Random Oracle Model (Section 6.1)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>10 pairing</td>
<td>5 exp</td>
</tr>
<tr>
<td>gap</td>
<td>5 pairing</td>
<td>3 exp</td>
</tr>
<tr>
<td>caching</td>
<td>4 pairing</td>
<td>4 exp</td>
</tr>
<tr>
<td>caching + gap</td>
<td>2 pairing</td>
<td>2 exp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AKE Standard Model (Section 6.2), KEM: Chapter 5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>8 pairing</td>
<td>8 exp</td>
</tr>
<tr>
<td>caching</td>
<td>3 pairing</td>
<td>8 exp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AKE Standard Model (Section 6.2), KEM: [HW07b]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>3 pairing</td>
<td>13 exp</td>
</tr>
</tbody>
</table>

Table 6.5: Efficiency comparison of our CL-AKE schemes

...provably secure key agreement scheme in the e²CK model in the certificateless setting.

6.3 Conclusion

In this chapter, we present the first two certificateless key exchange protocols with a proof of security that actually consider the security requirements for certificateless key agreement which were outlined in Chapter 3. The first protocol is in the random oracle model and allows the adversary to ask reveal session key queries even if the adversary replaced the certificateless public keys of the parties participating in the session that is targeted by the reveal session key query. Since the proof for the scheme is in the e²CK model which we introduced in Section 3.5.1, it also allows an adversary to corrupt parties that participate in the test session partially. As long as each party holds at least one uncompromised secret, the protocol is secure. The second protocol that we discussed in this chapter is in the standard model and is based on the KEM-KEM construction by Boyd et al. [BCGP08], which we modify so that we can also prove it in the e²CK model for certificateless key agreement. With these two protocols we show that there exist certificateless key agreement schemes with reasonable security guarantees, both in the random oracle model and in the standard model.

It is interesting to compare the efficiency of the standard model protocol with the efficiency of the random oracle model protocol, as shown in Table 6.5. With all possible optimisations, the ROM protocol is barely more efficient than the standard model protocol (when considering multiple runs between the same
parties). Without any optimisations, the ROM protocol is even less efficient than the standard model protocol. When considering other protocols, it is usually so that random oracle model protocols are much more efficient than standard model protocols. However, our random oracle model protocol is proven in the strong certificateless security model, whereas the standard model protocol is proven in the weak certificateless security model. To get an indication whether the loss of efficiency is related to the different security models, it would be interesting to compare our random oracle model protocol with a strongly secure standard model protocol. However, since there currently are no strongly secure standard model certificateless AKE protocols, this leaves us with an open question that requires further research.
Chapter 7

Conclusion and Future Work

In this chapter, we summarize the results of this thesis, discuss an application of our work, and give directions for possible future work. First, we give an overview of the contributions of the individual chapters. Then, we discuss an application of our work in a mobile health solution, based on the key agreement scheme presented in Section 6.1. Finally, an outlook on future work lists some open problems that may warrant future research.

7.1 Contributions

This thesis discusses new cryptographic constructions in certificateless cryptography. The individual contributions made are as follows.

- In Chapter 2, we set the stage for the thesis by introducing the theoretical background that is used throughout the thesis. The main contributions of this chapter are the twin bilinear trapdoor theorems in Section 2.2.7. We also recapitulate the standard notions for encryption, key encapsulation, and key agreement, both in terms of security and expected functionality.

- In Chapter 3, we review the security models associated with certificateless cryptography in more depth. In particular, we focus on certificateless encryption, certificateless key encapsulation, and certificateless key agreement. We introduce new models for certificateless key agreement and selective-ID secure hierarchical certificateless encryption and extend exist-
ing models for certificateless key encapsulation. We also develop a proof strategy that we use for proofs of certificateless schemes in the thesis.

- In Chapter 4 the idea of multi-authority identity-based encryption originally presented by Dodis and Katz is proven secure for the first time in the strongest security model today. We show that existing ID-based encryption schemes can be used to build multi-authority encryption which is secure as long as at least one of the authorities is not compromised. This is similar to an $n$-out-of-$n$ group encryption scheme without thresholds. We also show how a class of IND-CPA secure schemes can be combined using parallel/masking/blinding encryption. This speeds up decryption and encryption on multiprocessor systems and helps against ciphertext expansion. We show that the techniques to obtain IND-CCA2 security from identity-based encryption described by Boneh et al. [BCHK07] are also applicable to the case of multi-authority encryption. All of these constructions are in the standard model. We then show how efficient certificateless hierarchical encryption schemes can be constructed as a special case of two-authority HIBE, building on the fact that the user simply may become his own authority. We conclude the chapter by showing two examples for certificateless encryption in the standard model obtained from the previous ideas. As a result we obtain the first lattice-based certificateless encryption scheme and a very efficient certificateless encryption scheme based on bilinear pairings.

- Chapter 5 proposes a new and efficient certificateless key encapsulation mechanism in the standard model. Our construction is an extension of the ID-based KEM by Kiltz and Galindo [KG06b], and we use the proof technique discussed in Chapter 3 to prove only the certificateless case. The resulting scheme is more efficient than a scheme obtained from the generic construction in the standard model by Huang and Wong [HW07b].

- In this chapter we give the first constructions for certificateless key agreement with a proof of security. Our first construction is in the random oracle model and uses the new e$^2$CK model for certificateless key agreement that we developed in Chapter 3 from Swanson’s model. This construction is interesting because it has very strong security guarantees: the simulator is able to answer key derivation queries even if the adversary replaces certifi-
cateless public keys and does not disclose the matching private keys. We prove the construction secure in the random oracle model as long as each party has at least one uncompromised secret.

Our second construction for certificateless key agreement is in the standard model. We show that a construction similar to the KEM-KEM construction by Boyd et al. [BCGP08] that uses certificateless key encapsulation schemes can be proven secure in the e\textsuperscript{2}CK model. An interesting observation is that although this key agreement scheme is proven secure in the standard model, it is computationally more efficient than the scheme proven in the random oracle model (ROM). We suspect that this is a result of the fact that the standard model scheme is only weakly secure, that is, the simulator does not have to answer session key reveal queries if the adversary replaces a certificateless public key.

Summarizing the contributions we can say that we propose new security models for certificateless schemes and give new constructions for certificateless encryption, key encapsulation and key agreement, both in the random oracle model and in the standard model. Our constructions and security models for key agreement schemes are the first provably secure constructions in certificateless cryptography, and are therefore a milestone in the field. Our construction for certificateless key encapsulation is one of the most efficient constructions in the standard model today. Our generic construction for certificateless encryption allows to obtain SID-IND-CCA2 secure certificateless encryption schemes from any IBE scheme that is SID-IND-CPA secure, and results in generic constructions for certificateless encryption. Compared to the most efficient constructions for certificateless encryption in the standard model today [DLP08, HW07a, Den08], our generic constructions have comparable efficiency and are in some cases even more efficient.

### 7.2 Applications

The key agreement protocol discussed in Section 6.1 has been implemented in the Brasilian mobile health solution Borboleta. Borboleta is developed at the University of São Paulo, in the Institute for Mathematics and Statistics. This is the first deployment of certificateless cryptography in a nation-wide application, and shows that certificateless cryptography is mature enough for everyday use. Goya,
Chapter 7. Conclusion and Future Work

Okida, and Terada [GOT10] describe the implementation and optimisations of our protocol in more detail.

7.3 Future Work

With the generic constructions for encryption schemes developed in Chapter 4, it is easy to construct efficient certificateless schemes in the standard model from existing or new ID-based encryption schemes. These constructions are in the standard model and are more efficient than previous generic constructions for certificateless encryption. In fact, they are almost as efficient as the ID-based schemes themselves, because the additional computational overhead needed for the certificateless part can be performed parallel to the computations needed for the ID-based part, if the ID-based scheme is a masking IND-CPA secure scheme. As most computer systems today are multi-processor systems, the additional computational overhead for certificateless encryption (which can be computed simultaneously with the ID-based encryption) will then not be noticeable for the user. From here, it would be interesting to obtain new constructions that have virtually no computational overhead, even on single processor systems.

As a by-product of the lattice-based certificateless encryption scheme discussed in Section 4.6.1, we obtained a lattice-based certificateless KEM in the standard model. It should be fairly evident that the multi-authority encryption paradigm from Chapter 4 may also be transferred to certificateless KEMs, however a proof of security in the generic case needs to be written. We consider a proof for the generic case of combining two ID-based KEMs to obtain a certificateless KEM a possible avenue for future work.

Regarding key agreement schemes, it would be interesting to obtain certificateless key agreement in the standard model that is secure in the strong certificateless model, because it would allow us to estimate if the computational overhead in our random oracle model construction from Section 6.1 can be attributed to the strong certificateless model.

It seems as if the attack by Au et al. [AMC+07] is not yet fully understood. Its implications on key agreement and key encapsulation mechanisms should be studied and it would be interesting to see if there is a generic way to attack strongly secure certificateless encryption schemes in the standard model based on Au et al.’s observations. Furthermore, efficient certificateless schemes prefer-
ably in the standard model that are resistant to denial of decryption attacks as discussed in Section 3.1.1 would be an interesting avenue for future research.
Bibliography


[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS.


[CCMLS05] Liqun Chen, Zhaohui Cheng, John Malone-Lee, and Nigel P. Smart. An Efficient ID-KEM Based On The Sakai-Kasahara Key Con-
http://eprint.iacr.org/2005/224. 5


[CD07] Rafael Castro and Ricardo Dahab. Two Notes on the Security of Certificateless Signatures. In Susilo et al. [SLM07], pages 85–102. 8


http://eprint.iacr.org/2009/351. 21


[CPHL07] Kyu Young Choi, Jong Hwan Park, Jung Yeon Hwang, and Dong Hoon Lee. Efficient Certificateless Signature Schemes. In Katz and Yung [KY07], pages 443–458. 8


[CRR08] Sherman S. M. Chow, Volker Roth, and Eleanor G. Rieffel. General Certificateless Encryption and Timed-Release Encryption. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors,


BIBLIOGRAPHY


[GLHC07] Zheng Gong, Yu Long, Xuan Hong, and Kefei Chen. Two Certificateless Aggregate Signatures From Bilinear Maps. In Feng and Gao [FG07], pages 188–193. 8


[GS05] M. Choudary Gorantla and Ashutosh Saxena. An Efficient Certificateless Signature Scheme. In Hao et al. [HLW+05], pages 110–116. 8


In 2009 International Conference on Computational Intelligence and Security, CIS 2009, Beijing, China, 11-14 December 2009, Volume 2 - Workshop Papers [DBL09a], pages 208–212. 9


[HLH08] Zhenjie Huang, Xuanzhi Lin, and Rufen Huang. Certificateless Concurrent Signature Scheme. In Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008, Zhang Jia Jie, Hunan, China, November 18-21, 2008 [DBL08b], pages 2102–2107. 8


[HMS+07] Xinyi Huang, Yi Mu, Willy Susilo, Duncan S. Wong, and Wei Wu. Certificateless Signature Revisited. In Pieprzyk et al. [PGD07], pages 308–322. 8


[KG06a] Eike Kiltz and David Galindo. Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation Without Random Oracles. In Batten and Safavi-Naini [BSN06], pages 336–347. 5


[LDLK09] Junzuo Lai, Robert H. Deng, Shengli Liu, and Weidong Kou. RSA-Based Certificateless Public Key Encryption. In Bao et al. [BLW09], pages 24–34. 8

[LHW07] Rongbo Lu, Dake He, and Changji Wang. Cryptanalysis and Improvement of a Certificateless Proxy Signature Scheme from Bilinear Pairings. In Feng and Gao [FG07], pages 285–290. 8


[LLM07] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger Security ofAuthenticated Key Exchange. In Susilo et al. [SLM07], pages 1–16. xv, 50, 52, 74, 80, 81, 157, 175


[LST09] Fagen Li, Masaaki Shirase, and Tsuyoshi Takagi. Certificateless Hybrid Signcryption. In Bao et al. [BLW09], pages 112–123. 8


[PK07] Je Hong Park and Bo Gyeong Kang. Security Analysis of the Certificateless Signature Scheme Proposed at SecUbiq 2006. In Denko et al. [DSL+07], pages 686–691. 8

[PS06] Kenneth G. Paterson and Jacob C. N. Schuldt. Efficient Identity-Based Signatures Secure in the Standard Model. In Batten and Safavi-Naini [BSN06], pages 207–222. 6


[SZB07] Yinxia Sun, Futai Zhang, and Joonsang Baek. Strongly Secure Certificateless Public Key Encryption Without Pairing. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang, and Chaoping Xing,


Lichen Wang, Zhenfu Cao, Xiangxue Li, and Haifeng Qian. Certificateless Threshold Signature Schemes. In Hao et al. [HLW+05], pages 104–109.


[WHY08] Xu An Wang, Xinyi Huang, and Xiaoyuan Yang. Further Observations on Certificateless Public Key Encryption. In Yung et al. [YLL09], pages 217–239. 8, 58


[XLZ08] Zhong Xu, Xue Liu, Guoqing Zhang, Wenbo He, Guanzhong Dai, and Weihuan Shu. A Certificateless Signature Scheme for Mobile

[XQL08a] Hu Xiong, Zhiguang Qin, and Fagen Li. An Improved Certificateless Signature Scheme Secure in the Standard Model. Fundamenta Informaticae, 88(1-2):193–206, 2008. 8


[YCD08] Piyi Yang, Zhenfu Cao, and Xiaolei Dong. Chosen Ciphertext Secure Certificateless Threshold Encryption in the Standard Model. In Yung et al. [YLL09], pages 201–216. 8


[YHX09] Bo Yang, Zhengming Hu, and Zibi Xiao. Efficient Certificateless Strong Designated Verifier Signature Scheme. In 2009 International


[YLWS09] Xiaoyuan Yang, Zhongyin Liang, Ping Wei, and Junwei Shen. A Provably Secure Certificateless Blind Signature Scheme. In Proceedings of the Fifth International Conference on Information As-


[ZZW07] Lei Zhang, Futai Zhang, and Wei Wu. A Provably Secure Ring Signature Scheme in Certificateless Cryptography. In Susilo et al. [SLM07], pages 103–121.