UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Fakultät für Elektrotechnik, Informatik und Mathematik

Diplomarbeit

Construction and Applications of Identity-Based Encryption without Pairings*

Jonas Schrieb
E-Mail: `jonas@upb.de`

Paderborn, den 10. November 2008

vorgelegt bei
Prof. Dr. Johannes Blömer
Prof. Dr. Friedhelm Meyer auf der Heide

*The title of this thesis unfortunately does not fit its content. A more meaningful title would have been "Efficient Chosen-Ciphertext Security from Selective-ID Secure Identity-Based Encryption". The reason is that the topic of this thesis has shifted during my work. Whereas chosen-ciphertext security originally was meant to be the application of one specific IBE scheme it is now the main topic.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Paderborn, den 10. November 2008

Jonas Schrieb

Contents

1 Introduction

In this diploma thesis, two constructions are investigated that transform weakly secure identity-based encryption schemes into strongly secure public key encryption schemes. The following sections briefly explain those two concepts of encryption, the meaning of weak and strong security, and give an intuition for the transformation. Finally, an overview of related work and this thesis' topics is given.

1.1 Public Key and Identity-Based Encryption

The concept of *Public Key Encryption* (PKE), suggested by Diffie and Hellman [DH76][1], started a revolution in cryptography. It enabled Alice and Bob (two persons well known in cryptography, who want to talk confidently over insecure channels) to encrypt their communication without ever having met before. The scheme is depicted in Figure 1.1.

To be able to receive confidential messages, Bob has to generate a pair of keys PK_B, SK_B (algorithm: KeyGen). PK_B is posted to a public directory, but SK_B is kept secret. If Alice (or anyone else) wants to communicate with Bob, she retrieves his public key PK_B from the directory and uses it to encrypt the messages (algorithm: Encrypt). Only Bob can decrypt the ciphertext with help of his secret key SK_B (algorithm: Decrypt). For secure message transfer in the other direction, Alice's key pair is used, which she has generated independently.

So far, there remains one important problem: If Eve can fool Alice to use her public key PK_E instead of Bob's, she can decrypt all messages destined to Bob. As a simple solution, Bob could hand over his public key personally, but this would remove the most important property of PKE: Alice and Bob shall not have to meet personally prior to secure communication. Certificates are a much better counter measure against this attack. This is not part of PKE but belongs to the *Public Key Infrastructure* [KL07, §12.3]. It is sketched in the following to motivate identity-based encryption.

A trusted third party (TTP) stands surety for the validity of public keys. After key generation, Bob personally contacts the TTP, gives a proof of his identity (e. g., by showing his passport), and hands in his public key. The TTP then issues a certificate crt_B that connects Bob's identity (e. g., his full name or his email address) with his public key PK_B. In order to validate certificates, Alice needs the TTP's public parameters. The distribution of those is still vulnerable to the attack

[1] In fact, work on this field had been done earlier by British researchers at the CESG, but was classified as "confidential" for a long time [Ell87].

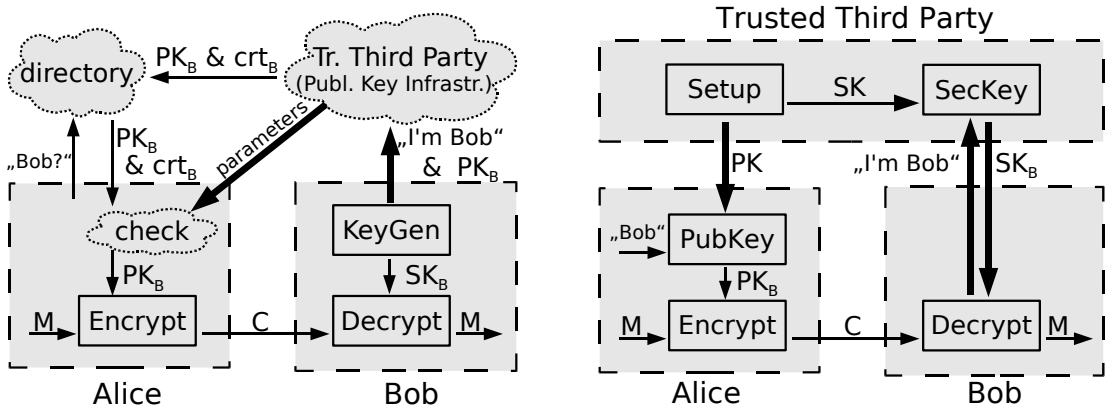

Figure 1.1: Public Key Encryption (left) and Identity-Based Encryption (right). The clouds represent parts that do not belong to the scheme but are helpful to understand the similarities and differences of PKE and IBE. Thick lines denote the necessity of personal contact.

by Eve, as described for the exchange of public keys above, so Alice has to fetch them personally from the TTP.

In total, since certificates cannot completely avoid the need of personal contact, all interaction with the TTP remains expensive. But for each user, this is necessary only once in order to get the TTP's parameters and a certificate for the own public key. After this initialization steps, the users only have to contact a directory and retrieve some public keys plus according certificates. This may be done over untrusted connections and only needs local validity checks.

Several years after invention of PKE, Shamir [Sha84] wanted to get rid of key directories and certificates. He came up with a new idea that he called *Identity-Based Encryption* (IBE). After initially having received some parameters, Alice should be able to derive Bob's public key PK_B from his identity only by local computations. This removes the need for directories to store PK_B and for certificates to link Bob's identity with PK_B. The full scheme can be seen in Figure 1.1.

In contrast to PKE, where the TTP is "only" an add-on to simplify public key distribution, the TTP plays a vital role for the definition of IBE. Initially, it sets up the system by generating a *master key pair* PK, SK (algorithm: Setup). When Bob joins the system, the TTP verifies his identify (e. g., by checking his passport), derives his personal secret key SK_B from his identity and the master secret key SK (algorithm: SecKey), and finally returns SK_B to him. When Alice joins the system, she is given the TTP's public master key PK. As for PKE, all interaction with the TTP has to be done via personal contact. After these initial steps, Alice can compute PK_B simply by using PK and Bob's identity (algorithm: PubKey). Encryption and decryption stay the same (algorithms: Encrypt and Decrypt).

IBE has the advantage of easy key distribution and low run-time costs—the TTP may even be removed after every user has gotten his secret key. It fits very well

to the needs of wireless networks [KKA03] or multiplayer games [DHMR07], since no delay occurs on determining the public key for new communications partners. The possibility to put more information into the "identity string" (e. g., not only the email address, but also an expiry date) opens its use to several interesting applications as in [MBH03] or [BF01, §1.1].

So, why should someone still be interested in PKE schemes? If Eve corrupts the TTP, she can compute all secret keys and thus read all confidential messages. Alice and Bob will not even be able to notice that. For PKE on the other hand, the worst thing Eve can do with a corrupted TTP is to certify her own public key as the one of Bob. From that moment on, she can read all messages sent from Alice to Bob, but not the older ones. Furthermore, eventually Alice will be suspicious if Bob cannot decrypt the messages she sends to him. In total, IBE needs a higher level of trust in the TTP's integrity, which is not given in every situation. For this reason, both concepts coexist legitimately.

1.2 Security Against Weak and Strong Attackers

One important part of security definitions for encryption schemes is a precise model for the power of an attacker (for an overview see [BDPR98]). There are three very common scenarios. All have in common that Eve intercepts a ciphertext $C^\star$ that has been sent from Alice to Bob. Eve wants to gain information on the underlying message. The very basic form of security under *chosen plaintext attacks (CPA)* models situations where an attacker Eve listens to the communication between Alice and Bob, but does not interfere. It is assumed that Eve knows the algorithms and public keys in use, but nothing else. So she can encrypt plaintexts of her choice and, for example, compare them to what Alice sends to Bob.

For security under *non-adaptive chosen ciphertext attacks (CCA_1)*, Eve may prepare her attack using an oracle which decrypts any ciphertexts of her choice. It may only be used until Eve intercepts the ciphertext $C^\star$. This models a situation where Eve has access to Bob's decryption device while he is out for lunch (and thus also known as "lunchtime attack"). After Bob has returned, Eve tries to decrypt his incoming messages, but she cannot access the decryption device any longer.

The strongest type of security is against *(adaptive) chosen ciphertext attacks (CCA_2)*, where Eve may access the decryption oracle even after intercepting $C^\star$. The only restriction is that she may not ask for the decryption of the the intercepted ciphertext $C^\star$, as this would make the attack trivial. On first sight, securing against such a strong attacker seems to be a very hypothetical problem without much impact on practical uses of cryptography. But for several reasons (some are described in the next chapter), CCA_2 security has gained a lot of attention, today.

The construction of CCA_2 secure PKE schemes is a very active field in cryptography. There are several approaches to gain CCA_2 security from weaker primitives such as trapdoor permutations [Sho02] or CPA secure PKE schemes [FO99, NY90]. The first two are very efficient, but their security proofs rely on the random or-

acle model [BR93]. For practical implementations, where random functions are replaced by hash functions, this provides only heuristic (yet widely tolerated) security arguments. The third is highly inefficient due to its use of non-interactive zero-knowledge techniques [BFM88].

There is another proposal on how to obtain CCA_2 secure PKE schemes from algebraic constructs with particular properties [CS02]. It has a rigorous security proof and makes a huge step towards practicability. The ciphertext is "only" four times as long as the message, which is a good improvement, but still unsatisfactory in practice. Therefore, it is an interesting task to find even more efficient CCA_2 secure PKE schemes. Interestingly, this is possible if CPA secure IBE (instead of PKE) is used as the basic building block.

1.3 From Weak IBE to Strong PKE

IBE has one essential property that allows this strong transformation to PKE. It is a kind of "independence" between the key pairs of different users. Even, if Eve spies out the secret keys of many other users or gets the keys by collaboration, this shall not help her with messages encrypted for Bob. If the secret keys may not help, then particularly decryptions of arbitrary messages under those secret keys cannot help. This is so important that it should be noted down:

Fact: Decryptions of ciphertexts using $\mathbf{SK_{ID}} \neq \mathbf{SK_{ID^\star}}$, do not give any information on a message encrypted with $\mathbf{PK_{ID^\star}}$.

To understand the basic idea of the following transformation, it is advisable to completely forget about the structure of IBE as described in the sections before. Instead, the master key pair PK, SK is given a new interpretation, which is very handy for the use in the following PKE construction. Consider the table in Figure 1.2, where each row consists of an arbitrary string (the former identity) and a corresponding key pair. The middle and right columns are essentially determined by PK and SK (the former master key pair), so these two keys can be seen as a very compact representation for a large set of public and secret keys. In the following, it is best to think of the Setup algorithm as a generator for a large table of key pairs. The PubKey and SecKey algorithms can be interpreted as table lookups for specific public and secret keys, respectively.

This table can then be used to create a CCA_1 secure PKE scheme as follows. Bob uses the Setup algorithm to generate the table of key pairs (represented by PK and SK) and publishes the column of public keys (represented by PK). If Alice wants to send a message M, she selects one of those public keys by randomly choosing a bitstring $ID^\star$ and encrypts M under $PK_{ID^\star}$ to obtain $C^\star$. Since Bob needs to know which secret key is the right one for decryption, Alice sends as ciphertext the tuple $\langle ID^\star, C^\star \rangle$. Now, Bob can determine $SK_{ID^\star}$ and decrypt $C^\star$.

How does this give CCA_1 security? Recall that the attacker Eve may use the decryption oracle only before she intercepts the ciphertext. Furthermore, in this

ID	$\mathsf{PubKey}(PK, ID)$	$\mathsf{SecKey}(SK, ID)$
"Bob"	PK_B	SK_B
"Eve"	PK_E	SK_E
"banana"	PK_{banana}	SK_{banana}
"fgvztgm"	$\vdots$	$\vdots$

Figure 1.2: In IBE, a master key pair "PK, SK" can be interpreted as a very compact representation of a large table of corresponding public and secret keys.

new PKE, the intercepted ciphertext is a tuple $\langle ID^\star, C^\star \rangle$. This means that Eve can query the decryption for several ciphertexts of the form $\langle ID, C \rangle$, but since she does not know $ID^\star$ at that time and the table is large, $ID \neq ID^\star$ holds for every query with high probability. Hence, all decryptions are done with secret keys $SK_{ID} \neq SK_{ID^\star}$, and thus by the above fact are useless for Eve. Without a useful decryption oracle, she cannot do better than in a chosen plaintext attack, which is covered by CPA security of the IBE.

Paradigm 1: To rule out the decryption oracle *before* interception of $\langle \mathbf{ID}^\star, \mathbf{C}^\star \rangle$, the selection of $\mathbf{ID}^\star$ should be unpredictable to an attacker.

In contrast, for CCA_2 security, Eve may still use the decryption oracle after intercepting $\langle ID^\star, C^\star \rangle$. Therefore, she could ask for the decryption of $\langle ID^\star, C \rangle$, with $C \neq C^\star$. This would give her the decryption of a ciphertext using the secret key $SK_{ID^\star}$. Such a decryption will not automatically help her—in fact, Eve has to choose C in a clever way to benefit from its decryption—, but it renders all arguments invalid that are based on the above stated fact. Several efficient approaches have been proposed, that circumvent these problems and obtain CCA_2 security. They all follow the same idea:

Paradigm 2: To rule out the decryption oracle *after* interception of $\langle \mathbf{ID}^\star, \mathbf{C}^\star \rangle$, it should be infeasible (or impossible) for an attacker to find a modification $\langle \mathbf{ID}^\star, \mathbf{C} \rangle$, where $\mathbf{C}$ is *valid* under $\mathbf{PK}_{\mathbf{ID}^\star}$.

As a result, no ciphertext submitted to the decryption oracle can help Eve: Either holds $ID \neq ID^\star$ (allowing use of the above fact) or $ID = ID^\star$ but the ciphertext is probably invalid and hence the oracle only returns "ciphertext invalid".

1.4 Related Work

For this thesis, basically five constructions (presented in three papers) are important. They are depicted in Figure 1.3. All have in common that they turn CPA secure IBE into CCA_2 secure PKE somehow using the above mentioned paradigms. They essentially differ in their tradeoff between generality and efficiency.

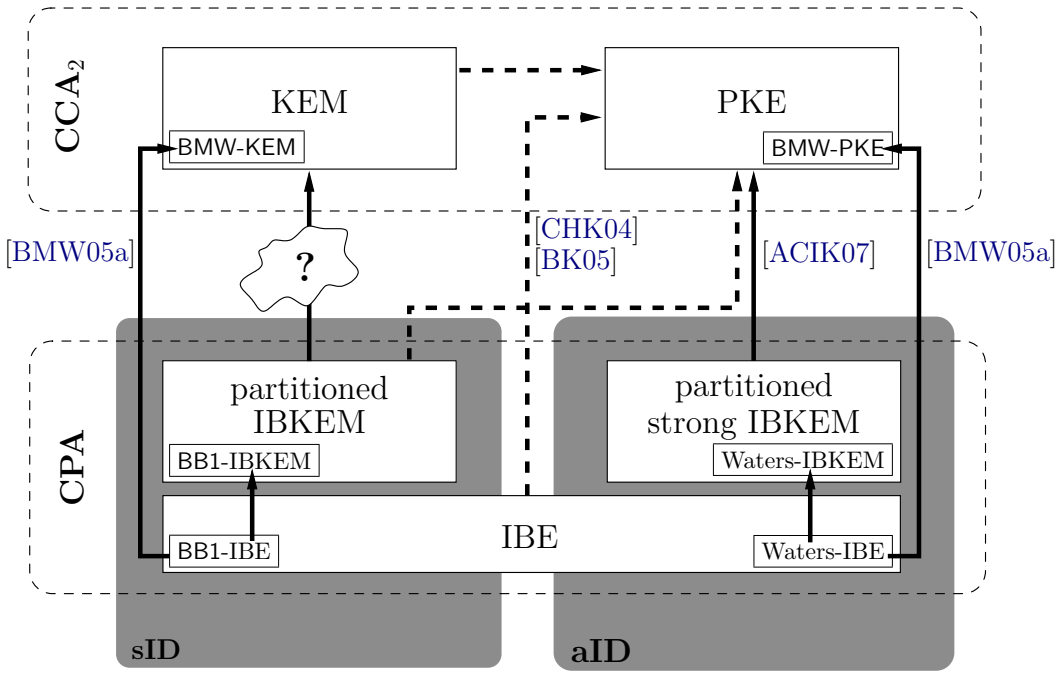

Figure 1.3: A schematic depiction of related work. Solid lines denote efficient transformations without efficiency loss. Dashed lines indicate transformations, that introduce computational overhead or enlarge ciphertexts. The question mark roughly points out the open problem that is tackled in this paper.
The standard security variant for IBE is "aID" and the weaker form is "sID". "Partitioned" and "strong" denote the additional structural properties and the strengthened variant of CPA security from [ACIK07].

The first construction is due to Canetti et al. [CHK04]. The ideas presented in the previous section originate from their work. The CHK transformation uses a signature scheme to link $ID^\star$ and $C^\star$. Any modification as described in the second paradigm implies a forgery of signatures and thus is infeasible. This construction is generic, i. e., any CPA secure IBE can be used as basic building block. The use of the signature scheme introduces a computational overhead and enlarges ciphertexts. Boneh and Katz [BK05] replace the signatures by more efficient means to improve the efficiency. The BK transformation is not considered here, because it is similar in spirit to the CHK transformation but technically much more complicated.

The second approach due to Boyen et al. [BMW05a], investigates another way to implement the above paradigms. They directly define CCA_2 secure schemes that are reminiscent of the original IBE, but do not result from a generic transformation rule. Specific properties of the "underlying" IBE are exploited to avoid any overhead. The ciphertext size of the new PKE is the same as in the original IBE and there is no need to use any additional cryptographic primitives, like signatures as above. In a nutshell, generality is sacrificed for efficiency.

These ideas are applied to two concrete schemes: The first, Waters-IBE [Wat05]

is turned into BMW-PKE. The second, BB1-IBE[2] [BB04], has weaker security properties than Waters-IBE and is transformed into a *key encapsulation mechanism*: BMW-KEM. KEM is a restricted form of PKE, only capable of encrypting randomly generated one-time keys instead of arbitrary messages chosen by the sender. These one-time keys may then be used with techniques from secret key encryption to obtain a full-fledged *hybrid* PKE. Consequently, a KEM is not a real restriction compared to a PKE. In fact, for efficiency reasons hybrid PKE schemes are preferred to plain PKE schemes in practice. The same considerations are true for *identity-based KEM* (IBKEM), which is a restricted form of IBE. All in all, it is perfectly reasonable to do research on KEM (IBKEM) instead of PKE (IBE).

The third paper is closely related to the second. Abe et al. [ACIK07] analyze the construction of BMW-PKE from Waters-IBE and extract the properties that have been exploited. The results are additional structural requirements, a slightly strengthened variant of CPA security, and a construction rule that is applicable to any IBKEM with these additional properties. Their transformation has the full efficiency of the "Waters-IBE-to-BMW-PKE" construction, but broadens its applicability to other IBKEM.

The additional structural requirements are satisfied by many IBKEM and are easy to verify by only looking at the algorithms. The strengthened security variant, on the other hand, might be not so widespread or is more elaborate to check, because one has to get into the security proof. As a compromise, the authors give a second transformation, which only requires the structural properties but not the strengthened security variant. Therefore, it sacrifices a little bit of efficiency by introduction of a chameleon hash function[3].

1.5 Contribution and Organization of this Thesis

Applying the transformation of Abe et al. to Waters-IBE gives BMW-PKE. Thus, it may be seen as a good explanation for the general principle that underlies the construction of BMW-PKE. However, the authors cannot tell how BMW-KEM results from BB1-IBE, i. e., they do not give a transformation that turns IBKEMs with "properties like BMW-KEM" into CCA_2 secure KEMs. Actually, they expect that such a transformation would require "a stronger (and less natural) security requirement" and proving satisfaction of this requirement to be "not easier than providing a direct proof for the transformed KEM" [ACIK07, §5.4]. In other words, they think that it is not possible to find some "useful" construction that explains the general principle behind BMW-KEM.

In this thesis, their conjecture will be refuted. The "BB1-IBE-to-BMW-KEM" transformation will be generalized in a similar (and practical) way, as Abe et al. do it for "Waters-IBE-to-BMW-PKE". Furthermore, its relationship with the CHK

[2] Boneh and Boyen define two IBE. BB1-IBE denotes the first one, described in §4 of their paper.

[3] This second transformation has also been found (and published earlier) by Zhang [Zha07]. As he does not consider the construction of BMW-PKE, his paper is less of interest for this thesis.

transformation will be investigated. It will turn out that both have a common core, which may be interpreted as the formalization of paradigm 1 and 2.

In Chapter 2, all necessary definitions are introduced. In Chapter 3, the above mentioned related work is presented in more detail. In Chapter 4, first the relationship between BB1-IBE and BMW-KEM is reviewed intuitively, and then formalized. The chapter ends with a brief overview on the applicability of both constructions. In Chapter 5, the CHK transformation is reviewed in this new framework, which reveals its relationship to BMW-KEM. Finally, the main achievements are subsumed in Chapter 6, and interesting open problems are proposed for future work.

2 Basic Definitions and Concepts

Modern cryptography distinguishes from classic cryptography by a much more precise conception of what security is and how to convince others that a scheme might be secure. Often, the formal process can be divided into the following steps.

1. Define the goals of a scheme, i. e., its algorithmic interface and correctness properties to tell whether an algorithm correctly implements this interface.
2. Provide a precise definition for the circumstances under that a scheme is considered to be secure. In particular, define the abilities of a possible attacker.
3. State a hardness assumption on which the security relies. This can either be the assumption that some number-theoretic problem is hard to solve, or that a secure implementation for some underlying cryptographic scheme exists.
4. Reduce the number-theoretic problem or the security of an underlying cryptographic scheme to the breakage of a concrete implementation. This proves the concrete implementation to be secure under the condition that the hardness assumption holds.

The introduction already has given an informal description of the algorithms for PKE and IBE and briefly mentioned KEM and IBKEM. In Section 2.1, all those encryption schemes will be specified formally. In Section 2.2, several security definitions will be given for each of these schemes that vary in the power of the attacker. In Section 2.3, the relations between different encryption schemes and their differently strong security definitions are briefly stated. Section 2.4 introduces two more cryptographic primitives that are essential for the upcoming transformations: Hash functions and signatures. The algebraic construct of bilinear groups together with some related problems that are widely believed to be computationally hard are presented in Section 2.5. Bilinear groups are the base for many IBE or IBKEM schemes, including BB1-IBE and Waters-IBE. Finally, the widespread proof techniques of reductionist proofs and game playing are briefly explained in Section 2.6.

Notation: An algorithm is probabilistic polynomial time (PPT) if it uses random bits in addition to its input and has a running time that is polynomially bounded in the bit-length of its input. $poly(\lambda)$ denotes the set of all polynomials in variable λ. In several definitions, a PPT algorithm is equipped with a security parameter $\lambda \in \mathbb{N}$ written as unary number 1^λ. The purpose is to give the algorithm an input of bit-length λ (instead of $\log_2(\lambda)$ if λ was encoded as a binary number), which implies

a time bound in $poly(\lambda)$ and such, for example, the ability of producing a key pair of bit-length λ. A PPT algorithm $\mathcal{A}$ can be given access to several (not necessarily polynomially bounded) oracles $O_1, O_2, \ldots$, which are denoted as $\mathcal{A}^{O_1,O_2,\cdots}$. Each oracle query counts as one time-step.

In probabilistic algorithms or random experiments an operator "$\leftarrow$" is extensively used that has two meanings. If X is a finite set, then $x \leftarrow X$ means picking x uniformly at random from X. Otherwise, $x \leftarrow \mathcal{A}(y, z, \ldots)$ means feeding the (possibly randomized) algorithm $\mathcal{A}$ with inputs $y, z, \ldots$ and assigning the output to x. $Pr_{Exp}[E]$ is the probability that event E occurs in experiment Exp, where the probability is over all sampled values and the internal randomness of all involved algorithms. Sometimes the following abbreviations are used: $Pr[Exp = win]$, which denotes the probability for the designated return value of the experiment to become "win", or $Pr[\text{description of an experiment} : E]$, where the (short) experiment is directly defined between "[" and ":".

In the following, advantage functions (depending on the security parameter λ) will measure the success of an attacker in such an experiment. A function $f : \mathbb{N} \to [0, 1]$ is called negligible if it decreases faster than any inverse of a polynomial, i. e., $\forall p \in poly(\lambda) : \exists \Lambda \in \mathbb{N} : \forall \lambda > \Lambda : f(\lambda) < 1/p(\lambda)$. Furthermore, $negl(\lambda)$ is the set of all negligible functions. The sum of two negligible functions and the product of a polynomial and negligible function are again negligible.

2.1 Encryption Schemes

In this section, the algorithms for public key encryption (PKE) and the concept of hybrid encryption (KEM, DEM) are formally defined. Then, identity-based encryption (IBKEM, IBE) plus a not yet mentioned cross between public key and identity-based encryption called *tag-based encryption* (TBKEM, TBE) are presented.

2.1.1 Public Key Encryption (PKE)

A public key encryption scheme PKE is defined by three PPT algorithms for key generation, encryption, and decryption [KL07, §10.2]. Exchange and certification of public keys (§1.1 on p.1) is beyond the scope of this definition.

KeyGen$(1^\lambda) \rightsquigarrow PK, SK$**;** The key generator takes a security parameter $\lambda \in \mathbb{N}$ and computes a public/secret key pair PK, SK.

Encrypt$(PK, M) \rightsquigarrow C$**;** The encryption algorithm takes a message M and encrypts it to a ciphertext C under the public key PK. The set of *valid ciphertexts under* PK is denoted as $\mathcal{C}(PK)$ and contains all ciphertexts that might be returned by Encrypt on input PK and some message M.

Decrypt$(SK, C) \rightsquigarrow M$**;** The decryption algorithm recovers the message M from a ciphertext C with help of the secret key SK.

In the definition only SK is given as input for Decrypt, although PK might also be needed. To avoid clutter, throughout all definitions only the strongest key is explicitly stated as input and the weaker keys are implicitly assumed to be available, too. I. e., for PKE, feeding Decrypt with SK implicitly includes PK.

PKE is *correct* if for all key pairs PK, SK and all messages M it holds that:

$$\mathsf{Decrypt}(SK, \mathsf{Encrypt}(PK, M)) = M$$

Correctness only affects valid ciphertexts. If Decrypt is started on an invalid $C \notin \mathcal{C}(PK)$, it may return an arbitrary message or a special *rejection symbol* $\perp$.

2.1.2 Hybrid Encryption (KEM, DEM)

Apart from the drawback of involved key exchange, there are two big advantages with the older technique of secret key encryption (SKE) compared to PKE: Flexible message spaces and efficiency. Many SKE constructions can encrypt messages of arbitrary length and structure, whereas in PKE this often is restricted due to the encoding of messages as elements of the underlying algebraic groups. Furthermore, SKE is very fast and has almost no ciphertext expansion, i. e., message and ciphertext have essentially the same length.

To have the advantages of both worlds, it has become common practice to use the following procedure to encrypt a message M under a public key PK: First, generate a short random key K and encrypt it to C_K using PKE with the public key PK. Then, encrypt the longer message M to C_M using SKE with the one-time key K. Finally send both ciphertexts $\langle C_K, C_M \rangle$ to the recipient, who can first obtain K from C_K using his secret key SK corresponding to PK and then recover M from C_M using K as key for the SKE scheme. By now, the definitions of PKE and SKE have been further optimised towards this application. The result is a Key Encapsulation Mechanism (KEM) as replacement for PKE, a Data Encryption Mechanism (DEM) in place of SKE and a key derivation function (KDF) that translates the output of KEM into a usable key for DEM [CS04].

Key Encapsulation Mechanism: PKE is designed to encrypt arbitrary messages of the sender's choice, but for hybrid encryption, this is too powerful. There is no need to choose the session key first and then, in a separate step, encrypt it. It would be perfectly ok if the encryption algorithm outputs both: A random looking session key K and its encryption C. This is exactly, what a KEM does. Surprisingly, in many cases this can be done more efficiently than PKE.

A key encapsulation mechanism KEM is a collection of three PPT algorithms for key pair generation, session key encapsulation and restoration.

KeyGen$(1^\lambda) \rightsquigarrow PK, SK$**;** The key generator takes a security parameter $\lambda \in \mathbb{N}$ and computes a public/secret key pair PK, SK.

Encaps$(PK) \rightsquigarrow K, C$**;** The encapsulation algorithm generates a session key K and encapsulates it in a ciphertext C under the public key PK. The *session key space* and the set of *valid ciphertexts* are denoted as $\mathcal{K}(PK)$ and $\mathcal{C}(PK)$. They contain all session keys and ciphertexts, respectively that might be returned by Encaps on input PK. $\mathcal{K}(PK)$ has to be superpolynomially large in λ.

Decaps$(SK, C) \rightsquigarrow K$**;** The decapsulation algorithm recovers the session key K from a ciphertext C with help of the secret key SK.

KEM is *correct* if for all key pairs PK, SK it holds that

$$\mathsf{Decaps}(SK, C) = K \quad \text{if} \quad K, C \leftarrow \mathsf{Encaps}(PK)$$

Again, this correctness property only affects valid ciphertexts. For an invalid ciphertext, Decaps may output arbitrary session keys or the rejection symbol $\perp$.

Data Encryption Mechanism: From the algorithmic point of view, a DEM is like an SKE, except that there is no need for an algorithm to generate the secret key. The real difference will be in the security definition. A Data Encryption Mechanism (DEM) is specified by two *deterministic* polynomial time algorithms and a function $\ell \in poly(\lambda)$ that determines the size of session keys for a security parameter $\lambda \in \mathbb{Z}$.

Encrypt$(K, M) \rightsquigarrow C$**;** Encrypts a message M to a ciphertext C with help of the session key $K \in \{0,1\}^{\ell(\lambda)}$.

Decrypt$(K, C) \rightsquigarrow M$**;** Recovers the message M from a ciphertext C with help of the session key $K \in \{0,1\}^{\ell(\lambda)}$.

A DEM is *correct* if for all $K \in \{0,1\}^{\ell(\lambda)}$ and all messages M it holds that

$$\mathsf{Decrypt}(K, \mathsf{Encrypt}(K, M)) = M$$

2.1.3 Identity-Based Encryption (IBKEM, IBE)

As outlined in the introduction (§1.1 on p.1), PKE and IBE (or KEM and IBKEM) only differ in key generation and distribution. In the identity-based setting, the KeyGen algorithm is replaced by three algorithms for master key pair generation, computing the public key, and computing the secret key for an identity [BF01, BFMLS05]. Similar to PKE, identification of the user at the TTP and secure transport of the secret key to the user are not part of this definition and have to be handled by other means. An identity-based key encapsulation mechanism IBKEM is defined by the following five PPT algorithms:

Setup$(1^\lambda) \rightsquigarrow PK, SK$**;** The setup algorithm takes a security parameter $\lambda \in \mathbb{N}$ and computes a master public/secret key pair PK, SK.

PubKey$(PK, ID) \rightsquigarrow PK_{ID}$**; (deterministic)** The public key generator computes the public key PK_{ID} for identity $ID \in \{0,1\}^*$ using master public key PK.

SecKey$(SK, ID) \rightsquigarrow SK_{ID}$**;** The secret key generator computes a secret key SK_{ID} for identity $ID \in \{0,1\}^*$ using the master secret key SK.

Encaps$(PK_{ID}) \rightsquigarrow K, C$**;** The encapsulation algorithm generates a session key K and encapsulates it in a ciphertext C under the public key PK_{ID}. The *session key space* and the set of *valid ciphertexts* are denoted as $\mathcal{K}(PK_{ID})$ and $\mathcal{C}(PK_{ID})$ and defined analogously to that of KEM.

Decaps$(SK_{ID}, C) \rightsquigarrow K$**;** The decapsulation algorithm recovers the session key K from a ciphertext C with help of the secret key SK_{ID}.

As for PKE, for every algorithm only the strongest key is listed. All weaker keys are implicitly assumed included, too. This means that, e. g., Decaps also gets PK_{ID} and PK. IBKEM is *correct* if for all $ID \in \{0,1\}^*$ and all key pairs PK_{ID}, SK_{ID}:

$$\mathsf{Decaps}(SK_{ID}, C) = K \quad \text{if} \quad K, C \leftarrow \mathsf{Encaps}(PK_{ID})$$

The notion of identity-based encryption (IBE) is related in the same way to IBKEM as KEM to PKE, i. e., IBE.Encrypt additionally takes a message M, but does not output a session key K and IBE.Decrypt restores M instead of K.

2.1.4 Tag-Based Encryption (TBKEM, TBE)

Tag-based encryption (key encapsulation) is a mixture of PKE and IBE (KEM and IBKEM) [MRY04][1]. As in the public key setting, key pairs are generated by every user on his own. But similar to the identities, binary strings called tags are used as additional input for encryption and decryption. TBKEM is likewise suitable for the ideas sketched in the introduction (§1.3 on p.4), although with identity-based techniques the basic idea is more intuitively comprehensible. Furthermore, basing the constructions on TBKEM gives more generality, as IBKEM naturally gives TBKEM, but TBKEM is strictly weaker. It can be built from primitives that are currently not known to give IBKEM [Kil06]. A tag-based key encapsulation mechanism TBKEM is defined by three PPT algorithms for key pair generation, encryption and decryption.

KeyGen$(1^\lambda) \rightsquigarrow PK, SK$**;** The key generator takes a security parameter $\lambda \in \mathbb{N}$ and computes a public/secret key pair PK, SK.

Encaps$(PK, tag) \rightsquigarrow K, C$**;** The encapsulation algorithm generates a session key K and encapsulates it in a ciphertext C under the public key PK and $tag \in \{0,1\}^*$. The *session key space* and the set of *valid ciphertexts* are denoted as $\mathcal{K}(PK, tag)$ and $\mathcal{C}(PK, tag)$ and defined analogously to that of KEM.

[1] [MRY04] define tag-based *encryption*. To my best knowledge, tag-based *KEM* has not been formalized yet, but the definition given here is the straight forward merger of TBE and KEM. It must not be confused with Tag-KEM [AGKS05], which is a TBKEM with special properties.

Decaps$(SK, tag, C) \rightsquigarrow K$**;** The decapsulation algorithm recovers the session key K from a ciphertext C with help of the secret key SK and the $tag \in \{0,1\}^*$.

TBKEM is *correct* if for all key pairs PK, SK and tag it holds that:

$$\mathsf{Decaps}(SK, tag, C) = K \quad \text{if} \quad K, C \leftarrow \mathsf{Encaps}(PK, tag)$$

The notion of tag-based encryption (TBE) is related in the same way to TBKEM as KEM to PKE, i. e., TBE.Encrypt additionally takes a message M, but does not output a session key K and TBE.Decrypt restores M instead of K.

2.2 Security Definitions

For encryption schemes, security definitions consist of two parts: A *security goal* that describes the type of attack a scheme should resist, and an *attack model* that defines the power of the attacker. For both, there are several flavors that can be freely combined. These are the most common security goals:

Semantic Security (SS): Given a ciphertext, no attacker should be able to compute any "meaningful information" about the underlying message.

Non-Malleability (NM): Given a ciphertext, no attacker should be able to compute another ciphertext such that the underlying messages are "meaningfully related".

Indistinguishable Encryptions (IND): Given a ciphertext that is the encryption for one randomly selected of two equal-size messages, the attacker should not be able to tell which message has been encrypted.

The first one models the confidentiality that an encryption scheme should provide and is clearly desirable. The second models some kind of tamper-resistance. Consider a tendering where several companies send their encrypted prices to the ordering party. In a malleable cryptosystem, it could be possible to intercept the ciphertext of a competitor and modify it to a slightly lower price. The "meaningful relation" could be "$M' = M - 1$", for example. The usefulness of the third one is not so obvious. Luckily, it can be shown to be equivalent with semantic security and in the strongest attack model even with non-malleability. Since its simple formalization makes it handy for proofs, it is the security goal of choice in this paper (and most others).

The attack models define how much information is given to the attacker:

Chosen Plaintext Attack (CPA): The attacker can obtain encryptions for messages of his choice. In the public key setting, this is accomplished by giving the public key as input.

Non-Adaptive Chosen Ciphertext Attack (CCA_1): In addition to the public key, the attacker can obtain decryptions for ciphertexts of his choice. Therefore, he is given access to a decryption oracle that on input C outputs $\mathsf{Decrypt}(SK, C)$. He may use it only to prepare his attack before seeing the ciphertext.

(Adaptive) Chosen Ciphertext Attack (CCA_2): The attacker may use the decryption oracle during the whole attack. To rule out trivial attacks, the attacker may not request the decryption for the ciphertext he is given.

The complete security definitions are then obtained by combing both tokens, e. g., the strongest one is IND-CCA_2. One might wonder why this strong notion is desirable, or whether it might be too strong. There are at least three good reasons to try to achieve it:

1. There is a practical attack on SSL that uses a weak form of a chosen ciphertext attack against RSA. IND-CCA_2 is a sufficient condition for resistance against this attack, the others are not.

2. IND-CCA_2 implies *both* SS-CCA_2 and NM-CCA_2, whereas for the weaker attack models it only gives SS-$*$ [BDPR98, GM84, ACG+06]. Therefore, a single security proof can provide both, confidentiality and tamper-resistance.

3. Tackling difficult problems should be in the nature of every researcher ☺.

In the following, the formal security definitions for indistinguishability are given. If not otherwise stated, they can be found in the same literature as cited in the previous section. Security is defined by a random experiment called $\mathrm{Exp}^{\mathsf{Scheme},\mathcal{A}}_{\text{IND-ATK}}(\lambda)$, where ATK is a token representing the attack model and Scheme is one of PKE, KEM, IBKEM, ... The outcome of the experiment is either *win* or *lose*. $\mathcal{A}$'s quality is measured in its *advantage* compared to a trivial attacker that only guesses without looking at its input and such wins with probability 1/2.

$$\mathrm{Adv}^{\mathsf{Scheme},\mathcal{A}}_{\text{IND-ATK}}(\lambda) := \left| Pr[\mathrm{Exp}^{\mathsf{Scheme},\mathcal{A}}_{\text{IND-ATK}}(\lambda) = win] - \frac{1}{2} \right|$$

Definition 2.1. *A Scheme is* ϵ-IND-ATK secure *if $\epsilon \in negl(\lambda)$ and it holds for* any *PPT algorithm $\mathcal{A}$ that*

$$Adv^{\mathsf{Scheme},\mathcal{A}}_{IND\text{-}ATK}(\lambda) \leq \epsilon(\lambda)$$

In all definitions, $\mathcal{A}$ may be run in several phases. For indistinguishable encryptions of PKE, there are two phases: Phase 1 before $\mathcal{A}$ is shown the ciphertext and phase 2 afterwards. Then, $\mathcal{A}$ is split into several subalgorithms, e. g., $A = \langle \mathcal{A}_1, \mathcal{A}_2 \rangle$. It is implicitly assumed that $\mathcal{A}_2$ has access to all variables used by $\mathcal{A}_1$. In particular, external input and query answers are available to $\mathcal{A}$ from the moment they are provided to $\mathcal{A}$, until the end of the experiment. All computations of $\mathrm{Exp}^{\mathsf{Scheme},\mathcal{A}}_{\text{IND-ATK}}(\lambda)$ that are not part of $\mathcal{A}$ are said to be executed by the *environment*.

Definition 2.2. *For PKE, the experiment of* indistinguishable encryptions under chosen ciphertext attacks (IND-CCA$_2$) *is defined as follows.*

$$
\begin{aligned}
&Exp^{\mathsf{PKE},\mathcal{A}}_{\text{IND-CCA}_2}(\lambda)\text{:}\\
&\quad PK, SK \leftarrow \mathsf{KeyGen}(1^\lambda)\\
&\quad M_0, M_1 \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(PK)\\
&\quad b \leftarrow \{0,1\}\\
&\quad C^\star \leftarrow \mathsf{Encrypt}(PK, M_b)\\
&\quad b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(C^\star)\\
&\quad \textit{if } b = b', \textit{ then output win}
\end{aligned}
$$

M_0, M_1 have to be of equal length. On query $\mathsf{Decrypt}$-oracle(C)—where C may be an arbitrary (valid or invalid) ciphertext—the decrypt oracle returns $M \leftarrow \mathsf{Decrypt}(SK, C)$. In phase 2, $\mathcal{A}$ may not ask for the decryption of $C^\star$.

The other two experiments are exactly the same except that for IND-CCA$_1$, only $\mathcal{A}_1$ may access the $\mathsf{Decrypt}$ oracle, and for IND-CPA, neither $\mathcal{A}_1$ nor $\mathcal{A}_2$ may.

This exactly captures the intuitive definitions above: In phase 1, $\mathcal{A}$ may prepare its attack given the ability to encrypt messages of his choice using PK and to decrypt ciphertexts of his choice via the oracle. It may request any C – it might even request the not yet known $C^\star$ by accident. Eventually, $\mathcal{A}$ outputs two messages. One of the two messages is randomly chosen, encrypted and the *challenge ciphertext* $C^\star$ is given back to $\mathcal{A}$. Then in phase 2, $\mathcal{A}$ tries to find out whether M_0 or M_1 has been encrypted (i. e., $b = 0$ or $b = 1$), still having the power of encrypting messages and decrypting ciphertexts $\neq C^\star$. Finally, $\mathcal{A}$ outputs a *guess* b' for the value of b. $\mathcal{A}$ wins if this guess is correct.

In case that $\mathcal{A}$ violates the restrictions on message length and oracle queries, the experiment is *aborted*. This means that the execution of $\mathcal{A}$ is stopped and the outcome of the experiment is set such that it doesn't contribute to $\mathcal{A}$'s advantage, i. e., $\mathcal{A}$'s "guess" is chosen randomly ($b' \leftarrow \{0,1\}$).

For KEM, the above stated intuition for indistinguishability does not make any sense. It is replaced by another that is similar in spirit: Given a ciphertext and a session key, the attacker should not be able to tell whether the key corresponds to the ciphertext or is random. This definition is tailored for proving the security of hybrid encryptions.

Definition 2.3. *For KEM, the experiment of* indistinguishable encapsulation under chosen ciphertext attacks (IND-CCA$_2$) *is defined as follows.*

$$
\begin{aligned}
&Exp^{\mathsf{KEM},\mathcal{A}}_{\text{IND-CCA}_2}(\lambda)\text{:}\\
&\quad PK, SK \leftarrow \mathsf{KeyGen}(1^\lambda)\\
&\qquad\qquad \mathcal{A}_1^{\mathsf{Decaps}}(PK)\\
&\quad b \leftarrow \{0,1\}\\
&\quad K_0^\star, C^\star \leftarrow \mathsf{Encaps}(PK)\\
&\quad K_1^\star \leftarrow \mathcal{K}(PK)\\
&\quad b' \leftarrow \mathcal{A}_2^{\mathsf{Decaps}}(K_b^\star, C^\star)\\
&\quad \textit{if } b = b', \textit{ then output win}
\end{aligned}
$$

On query Decaps*-oracle*(C)*, where* C *may be an arbitrary (valid or invalid) ciphertext, the decapsulation oracle returns* $M \leftarrow \mathsf{Decaps}(SK, C)$*. In phase 2,* $\mathcal{A}$ *may not ask for the decapsulation of* $C^\star$*.*

For IND-CCA$_1$*, phase 2 is replaced by* $\mathcal{A}_2(K_b^\star, C^\star)$*, and for IND-CPA both, phases are merged to* $\mathcal{A}(PK, K_b^\star, C^\star)$ *that takes place at the end of the experiment.*

Phase 1 might look a little bit strange, since $\mathcal{A}_1$ has no output. But remember that $\mathcal{A}_1$ also has some "hidden output", since by convention $\mathcal{A}_2$ knows all the values computed by $\mathcal{A}_1$. Since $\mathcal{A}_1$ does not have any restriction on Decaps queries (opposed to $\mathcal{A}_2$ that may not query $C^\star$), it really might help $\mathcal{A}_2$ and is not useless. This is different for IND-CPA attacks, where, without oracle access, $\mathcal{A}_1$ would not have any more power than $\mathcal{A}_2$, so both can be merged.

Now, let's turn to the security of IBKEM. With KEM, every user individually generates his own key pair, and such, all these key pairs are independent of each other. For IBKEM, on the other hand, all key pairs depend on the master key pair of the TTP. To rule out the possibility that knowledge of some secret key helps attacking another one, the security definition has to be adjusted. A secret key for identity $ID^\star$ shall stay secure even if the attacker gets knowledge of some secret keys for other identities (recall the "Fact" mentioned in §1.3 on p.4). This is modeled by a SecKey oracle returning the secret key SK_{ID} on input ID. Queries for secret keys of $ID^\star$ itself have to be denied in order to rule out trivial attacks.

The identity $ID^\star$ may be chosen by the attacker. The attack models are extended by two variants that determine the attacker's power regarding the selection of $ID^\star$:

Selective Identity Attack (sID): The attacker has to select $ID^\star$ without prior to seeing any other value of the experiment. This model is due to [CHK03].

Adaptive Identity Attack (aID): The attacker may use the master public key and all oracles to choose $ID^\star$. This model is due to [BF01].

These identity attack models can be additionally combined with the other attack models, ranging from IND-sID-CPA to IND-aID-CCA$_2$. In the following, both experiments are defined for IBKEM. The experiments for IBE are related to it in the same way as those for PKE to those for KEM. IND-$*$ID-CPA security is the most interesting for this paper and is therefore defined here.

Definition 2.4. *For* IBKEM*, the experiment of* indistinguishable encapsulation under chosen plaintext attacks (IND-$*$ID-CPA) *is defined as follows.*

$Exp^{\mathsf{IBKEM},\mathcal{A}}_{\text{IND-sID-CPA}}(\lambda)$:
$PK, SK \leftarrow \mathsf{Setup}(1^\lambda)$
$ID^\star \leftarrow \mathcal{A}_1(1^\lambda)$
$b \leftarrow \{0,1\}$
$PK_{ID^\star} \leftarrow \mathsf{PubKey}(PK, ID^\star)$
$K_0^\star, C^\star \leftarrow \mathsf{Encaps}(PK_{ID^\star})$
$K_1^\star \leftarrow \mathcal{K}(PK_{ID^\star})$
$b' \leftarrow \mathcal{A}_2^{\mathsf{SecKey}}(PK, K_b^\star, C^\star)$
if $b = b'$, *then output win*

$Exp^{\mathsf{IBKEM},\mathcal{A}}_{\text{IND-aID-CPA}}(\lambda)$:
$PK, SK \leftarrow \mathsf{Setup}(1^\lambda)$
$ID^\star \leftarrow \mathcal{A}_1^{\mathsf{SecKey}}(PK)$
$b \leftarrow \{0,1\}$
$PK_{ID^\star} \leftarrow \mathsf{PubKey}(PK, ID^\star)$
$K_0^\star, C^\star \leftarrow \mathsf{Encaps}(PK_{ID^\star})$
$K_1^\star \leftarrow \mathcal{K}(PK_{ID^\star})$
$b' \leftarrow \mathcal{A}_2^{\mathsf{SecKey}}(K_b^\star, C^\star)$
if $b = b'$, *then output win*

On query SecKey*-oracle*(ID), *the oracle uses the* SecKey *algorithm to compute* SK_{ID} *and returns that key. The queried* ID *may not be* $ID^\star$. *In the adaptive identity setting, this restriction is retroactive for phase 1, i. e.,* $\mathcal{A}_1$ *may not output an* $ID^\star$ *that has been part of a* SecKey *query before.*

One might wonder why for IND-sID-CPA, $\mathcal{A}$ is given no chance to access a SecKey-oracle before getting the challenge ciphertext. The reason is that this gives $\mathcal{A}$ no additional advantage, similar as in IND-CPA for KEM. Therefore it is dropped. In the CCA experiments, $\mathcal{A}_1$ (which selects $ID^\star$ in IND-sID-CPA) is renamed to $\mathcal{A}_0$ and a new $\mathcal{A}_1$ is added that may access the SecKey- and Decaps-oracle.

As TBKEM is a hybrid of KEM and IBKEM, it is not surprising that this is also true for the security definitions. Everything being related to keys is similar to the KEM definitions, and everything with tags is reminiscent of the IBKEM definitions. As for IBKEM the attacker may choose a $tag^\star$ it would like to attack. There are several new attack models related to this $tag^\star$:

Selective/Adaptive Tag Attack (sTag/aTag): This is analogous to sID and aID for IBKEM. The selective notion is due to [Kil06] the adaptive due to [MRY04].

Weak Chosen Ciphertext Attack (wCCA$_1$/wCCA$_2$): This is similar to CCA$_{1/2}$, except that the Decaps oracle is replaced by wDecaps. This weaker oracle is restricted to queries where $tag^\star$ is not involved. These weakened attack models are due to [MRY04].

Again all the tag and ciphertext attack models can be freely combined, giving eight different notions of security. The IND-$*$Tag-wCCA$_2$ security is the most interesting for this paper and is therefore defined in the following.

Definition 2.5. *For* TBKEM*, the experiment of* indistinguishable encapsulation under selective/adaptive tag and weak chosen ciphertext attacks (IND-$*$Tag-wCCA$_2$) *is defined as follows.*

$Exp^{\mathsf{TBKEM},\mathcal{A}}_{\text{IND-sTag-wCCA}_2}(\lambda)$:
$PK, SK \leftarrow \mathsf{Setup}(1^\lambda)$
$tag^\star \leftarrow \mathcal{A}_1(1^\lambda)$
$b \leftarrow \{0,1\}$
$K_0^\star, C^\star \leftarrow \mathsf{Encaps}(PK, tag^\star)$
$K_1^\star \leftarrow \mathcal{K}(PK, tag^\star)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{wDecaps}}(PK, K_b^\star, C^\star)$
if $b = b'$, *then output win*

$Exp^{\mathsf{TBKEM},\mathcal{A}}_{\text{IND-aTag-wCCA}_2}(\lambda)$:
$PK, SK \leftarrow \mathsf{Setup}(1^\lambda)$
$tag^\star \leftarrow \mathcal{A}_1^{\mathsf{wDecaps}}(PK)$
$b \leftarrow \{0,1\}$
$K_0^\star, C^\star \leftarrow \mathsf{Encaps}(PK, tag^\star)$
$K_1^\star \leftarrow \mathcal{K}(PK, tag^\star)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{wDecaps}}(K_b^\star, C^\star)$
if $b = b'$, *then output win*

On query wDecaps*-oracle*(tag, C), *for* $tag \neq tag^\star$ *and an arbitrary (valid or invalid) ciphertext* C, *the decapsulation oracle returns* $M \leftarrow \mathsf{Decaps}(SK, tag, C)$. *Similar to the restriction on the* SecKey *oracle for IBKEM, this rule is retroactive for* $\mathcal{A}_1$ *in the adaptive tag attack, i. e.,* $\mathcal{A}_1$ *may not output a* $tag^\star$ *that has been part of a* wDecaps *query before.*

Last but not least, the security for DEM has to be defined. It is structurally similar to that of PKE above, but optimized towards being the minimal preconditions for the security of hybrid encryption.

Definition 2.6. *For* DEM, *the experiment of* indistinguishable encryption under chosen ciphertext attacks (IND-CCA$_2$) *is defined as follows.*

$Exp^{\mathsf{DEM},\mathcal{A}}_{\text{IND-CCA}_2}(\lambda)$:
$K \leftarrow \{0,1\}^{\ell(\lambda)}$
$M_0, M_1 \leftarrow \mathcal{A}_1(1^\lambda)$
$b \leftarrow \{0,1\}$
$C^\star \leftarrow \mathsf{Encrypt}(K, M_b)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(C^\star)$
if $b = b'$, *then output win*

M_0, M_1 *have to be of equal length. On query* Decrypt*-oracle*(C), *where* C *may be an arbitrary (valid or invalid) ciphertext* $\neq C^\star$, *the decrypt oracle returns* $M \leftarrow \mathsf{Decrypt}(K, C)$.

Contrary to PKE, there is no public key that could be given to the attacker. This withdraws $\mathcal{A}$ the possibility to get encryptions for messages of its choice. Furthermore, the Decrypt oracle may not be accessed in phase 1. This simply stems from the fact that this is not needed in the security proof for hybrid encryption.

2.3 Relations Between the Encryption Schemes

Figure 2.1 shows several relations between the different encryption schemes and the most important attack models (the security goal is always ciphertext indistinguishability). The constructions are sketched below:

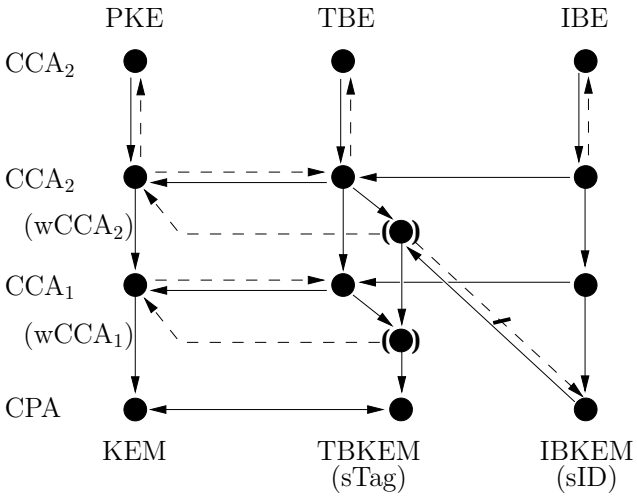

Figure 2.1: Relations between the different schemes. Solid lines are direct implications or trivial constructions without efficiency loss. Dashed lines indicate transformations that expand ciphertext length and loosen the security bound. The cancelled line denotes an impossibility result.

PKE|TBE|IBE to KEM|TBKEM|IBKEM: If the message space of PKE is large enough, a "message" can be chosen randomly and used as session key.

$\mathsf{KEM.Encaps}(PK)$:
$K \leftarrow MsgSpace$
$C \leftarrow \mathsf{PKE.Encrypt}(PK, K)$

$\mathsf{KEM.Decaps}(SK, C)$:
$K \leftarrow \mathsf{PKE.Decrypt}(SK, C)$

If the message space is too small, multiple messages can be chosen and the concatenation thereof forms the session key. The construction is analogous to obtain TBKEM from TBE or IBKEM from IBE.

KEM|TBKEM|IBKEM to PKE|TBE|IBE: For KEM-to-PKE this is shown in [CS04]. One needs an IND-CCA$_2$ secure DEM and a suitable key derivation function KDF. As a simple, but very restrictive example for a KDF, one may think of a function that maps a uniform distribution in the session key space $\mathcal{K}(\lambda)$ of KEM to an "almost uniform" distribution in $\{0,1\}^{\ell(\lambda)}$.

$\mathsf{PKE.Encrypt}(PK, M)$:
$K, C_K \leftarrow \mathsf{KEM.Encaps}(PK)$
$\overline{K} \leftarrow \mathsf{KDF}(K)$
$C_M \leftarrow \mathsf{DEM.Encrypt}(\overline{K}, M)$
$C \leftarrow \langle C_K, C_M \rangle$

$\mathsf{PKE.Decrypt}(SK, C)$:
$\langle C_K, C_M \rangle \leftarrow C$
$K \leftarrow \mathsf{KEM.Decaps}(PK, C_K)$
$\overline{K} \leftarrow \mathsf{KDF}(K)$
$C_M \leftarrow \mathsf{DEM.Decrypt}(\overline{K}, C_M)$

The construction is similar to obtain TBE from TBKEM or IBE from IBKEM.

TBKEM to KEM: Leaving out the *tag* during all computations gives a KEM.

KEM to TBKEM: This transformation is based on a simple idea given in [Kil06]. To construct a TBE from a PKE, simply concatenate the message M and the tag.

$\mathsf{TBE.Encrypt}(PK, tag, M)$:		$\mathsf{TBE.Decrypt}(SK, tag, C)$:	
$\overline{M}$	$\leftarrow M \| tag$	$M \| tag'$	$\leftarrow \mathsf{PKE.Decrypt}(SK, C)$
C	$\leftarrow \mathsf{PKE.Encrypt}(PK, \overline{M})$	check	$tag \stackrel{?}{=} tag'$

To obtain TBKEM from KEM one can use the following sequence of transformations: KEM $\rightsquigarrow$ PKE $\rightsquigarrow$ TBE $\rightsquigarrow$ TBKEM. Clearly, this construction is only of theoretical interest. In the formal security game for TBKEM, the tag's purpose is only to control the oracle access. Thus, for CPA security, KEM and TBKEM are equivalent, as there is no oracle access.

Weak TBKEM to KEM: Recall the "fact" that has been essential for (intuitive) security of the transformations from CPA secure IBE to CCA_2 secure PKE (§1.3 on p.4): Decapsulations using $SK_{ID} \neq SK_{ID^\star}$ shall not help with a ciphertext encrypted with $PK_{ID^\star}$. An analogous fact is implied by the $\mathsf{wDecaps}$ oracle for $\text{wCCA}_{1/2}$ secure TBKEM: decapsulations using $tag \neq tag^\star$ will not help with a ciphertext encrypted using $tag^\star$. Therefore, the constructions as sketched in the introduction (§1.4 on p.5) will also obtain $\text{IND-CCA}_{1/2}$ secure KEM from $\text{IND-sTag-wCCA}_{1/2}$. As these transformations are the main topic of this thesis, they are omitted here and will be studied in Chapter 4 and Chapter 5.

IBKEM to TBKEM: This transformation is simple again: the tag is used as ID.

$\mathsf{TBKEM.Encaps}(PK, tag)$:		$\mathsf{TBKEM.Decaps}(SK, tag, C)$:	
ID	$\leftarrow tag$	ID	$\leftarrow tag$
PK_{ID}	$\leftarrow \mathsf{IBKEM.PubKey}(PK, ID)$	SK_{ID}	$\leftarrow \mathsf{IBKEM.SecKey}(SK, ID)$
K, C	$\leftarrow \mathsf{IBKEM.Encaps}(PK_{ID})$	K	$\leftarrow \mathsf{IBKEM.Decaps}(SK_{ID}, C)$

The "rise" of CPA to wCCA_2 security stems from the fact that a $\mathsf{IBKEM.SecKey}$ oracle can be used to decapsulate messages for any $ID \neq ID^\star$ and thus allow the simulation of the $\mathsf{wDecaps}$ oracle.

PKE|KEM|TBE|TBKEM do *not* give IBKEM: Boneh et al. [BPR+08] show that IND-CCA_2 secure PKE cannot give IBE. It is easy to see in the diagram that PKE, KEM, TBE and TBKEM of any security level is equivalent to or weaker than IND-wCCA_2 secure TBKEM, particularly IND-CCA_2 secure PKE is equivalent. This justifies the placement of the cancelled arrow and the title of this paragraph.

aTag/aID and sTag/sID: The diagram looks the same for aTag/aID security. It is clear that aTag/aID implies sTag/sID security. On the other hand, even $\text{IND-sTag/sID-CCA}_2$ does not necessarily give IND-aTag/aID-CPA security [Gal06][2].

[2] This seems to contradict to another result due to Boneh and Boyen which roughly says that any IND-sID-$*$ secure IBE already has IND-aID-$*$ security, but with a bad security bound [BB04, §7]. In fact, their results are incomparable due to subtle differences in the notions of *asymptotic security* (used in [Gal06] and this thesis) and *concrete security* (used in [BB04]).

2.4 Other Cryptographic Primitives

In this section, two more cryptographic schemes are presented. They are not directly related to encryption but needed for several constructions in later chapters. With help of *target collision resistant hash functions*, arbitrary tags/identities from $\{0,1\}^*$ can be mapped to elements of some finite group in a secure way. Those group elements can be used in the encryption schemes. *Strongly unforgeable one-time signature schemes* are crucial for the CHK transformation.

2.4.1 Target Collision Resistant Hash Functions

A family of *keyed hash functions* H [KL07, §4.6] is a collection of functions that map a binary string $x \in \{0,1\}^*$ and a hash key $k \in \{0,1\}^{\ell(\lambda)}$ of length $\ell \in poly(\lambda)$ into elements from finite groups $\mathbb{G}_\lambda$:

$$H = \left\{ H_\lambda : \{0,1\}^* \times \{0,1\}^{\ell(\lambda)} \to \mathbb{G}_\lambda \right\}_{\lambda \in \mathbb{N}}$$

Usually, one drops λ and writes $H_k(x)$ instead of $H(x,k)$ for any fixed k. A standard security demand for hash functions used in cryptography is collision resistance, which roughly means that given a random k, it should be computationally infeasible to find $x, x^\star$ with $H_k(x) = H_k(x^\star)$. Here, the weaker but cheaper to implement *target collision resistance* (also known as *universal one-wayness*) [NY89] suffices, where the attacker has to commit to $x^\star$ before knowing the hash key.

Definition 2.7. *For a family of keyed hash functions H, the experiment of* target collision resistance (TCR) *is defined as follows.*

$Exp^{H,\mathcal{A}}_{TCR}(\lambda)$*:*
$x^\star \leftarrow \mathcal{A}_1(1^\lambda)$
$k \leftarrow \{0,1\}^{\ell(\lambda)}$
$x \leftarrow \mathcal{A}_2(k)$
if $H_k(x) = H_k(x^\star)$ *and*
$x \neq x^\star$*, then output win*

Similar to above, the quality of $\mathcal{A}$ is measured relative to an algorithm that only guesses without looking at k. Since this algorithm is supposed to win the experiment with probability close to 0 (instead of 1/2 as before), the advantage of $\mathcal{A}$ is defined a little bit different this time:

$$\mathrm{Adv}^{H,\mathcal{A}}_{\mathrm{TCR}}(\lambda) := Pr[\mathrm{Exp}^{H,\mathcal{A}}_{\mathrm{TCR}}(\lambda) = win]$$

H is ϵ-TCR secure if $\epsilon \in negl(\lambda)$ and for every PPT algorithm $\mathcal{A}$ it holds that

$$\mathrm{Adv}^{H,\mathcal{A}}_{\mathrm{TCR}}(\lambda) \leq \epsilon(\lambda)$$

2.4.2 Strongly Existentially Unforgeable One-Time Signatures

A signature scheme Sig is defined by three PPT algorithms for key pair generation, digital signing and signature verification [KL07, §12.2].

KeyGen$(1^\lambda) \rightsquigarrow VK, SigK$**;** The key generator takes a security parameter $\lambda \in \mathbb{N}$ and computes a public verification key VK and a secret signing key $SigK$.

Sign$(SigK, M) \rightsquigarrow sig$**;** The signing algorithm takes a message M and generates a digital signature sig with help of the signing key $SigK$.

Verify$(VK, M, sig) \rightsquigarrow true/false$**;** The verification algorithm checks the signature sig using verification key VK and outputs $true$ if it is valid for message M.

The scheme is *correct* if for all key pairs $VK, SigK$ and messages M it holds that:

$$\mathsf{Verify}(VK, M, \mathsf{Sign}(SigK, M)) = true$$

The scheme is secure if, given VK, no efficient attacker is able to produce a pair M, sig, such that sig is a valid signature of M under VK, even if he gets the signature $sig^\star$ for one message $M^\star$ of his choice. This is formalized as follows:

Definition 2.8. *For a signature scheme Sig, the experiment of* strongly existentially unforgeability under one-time chosen message attacks (SEU-OT) *is defined as:*

$$\begin{aligned} &Exp^{\mathsf{Sig},\mathcal{A}}_{SEU\text{-}OT}(\lambda): \\ &\quad VK, SigK \leftarrow \mathsf{KeyGen}(1^\lambda) \\ &\quad M, sig \leftarrow \mathcal{A}^{\mathsf{Sign}}(VK) \\ &\quad \textit{if } \mathsf{Verify}(VK, M, sig) = true, \textit{ then output win} \end{aligned}$$

$\mathcal{A}$ may query the Sign oracle for at most one message $M^\star$, which is answered with $sig^\star \leftarrow \mathsf{Sign}(SigK, M^\star)$. If $\mathcal{A}$ makes this query, then its output $\langle M, sig\rangle$ has to be $\neq \langle M^\star, sig^\star\rangle$.

The restriction on $\mathcal{A}$'s output only demands that $M \neq M^\star$ *or* $sig \neq sig^\star$, in particular, $\mathcal{A}$ is allowed to output a new signature sig for the same message $M^\star$. This is unlike other variants of unforgeability, where $M^\star$ may not be reused [ADR02].

Sig is ϵ-SEU-OT secure if $\epsilon \in negl(\lambda)$ and for all PPT algorithms $\mathcal{A}$ it holds that

$$Pr[\mathrm{Exp}^{\mathsf{Sig},\mathcal{A}}_{\mathrm{SEU\text{-}OT}}(\lambda) = win] \;=:\; \mathrm{Adv}^{\mathsf{Sig},\mathcal{A}}_{\mathrm{SEU\text{-}OT}}(\lambda) \;\leq\; \epsilon(\lambda)$$

2.5 Bilinear Groups

Bilinear groups provide a rich algebraic structure that fits very well the needs for many cryptographic constructions. They are used for (but not limited to) many identity-based encryption schemes. Two constructions presented in Chapter 3 are based on bilinear groups. Boneh and Franklin [BF01] construct bilinear groups from elliptic curves over finite fields and a modification of the Weil- or Tate-pairing.

Definition 2.9. *A cyclic group $\mathbb{G}$ of prime order p is called* bilinear *if there exists another cyclic group $\mathbb{G}_T$ of order p together with a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:*

- *e is bilinear, i. e., for any $x, y, z \in \mathbb{G}$ it holds that*

$$e(x \cdot y, z) = e(x, z) \cdot e(y, z) \qquad \text{and} \qquad e(x, y \cdot z) = e(x, y) \cdot e(x, z)$$

- *e is not degenerate, i. e., if g generates $\mathbb{G}$ then $e(g, g)$ generates $\mathbb{G}_T$*
- *the operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the map e are efficiently computable*

As convention, g will always denote a generator in $\mathbb{G}$.

The following fact will be used in several calculations with bilinear groups:

$$\forall a, b \in \mathbb{Z} : \; e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$$

When doing cryptography in cyclic finite groups, there are many common problems that security may be based on. They are roughly defined as follows [KL07, §7.3]:

Discrete logarithm (DLOG): Given $g, x \in \mathbb{G}$, compute $a \in \mathbb{Z}_{|\mathbb{G}|}$, such that $x = g^a$.

Computational Diffie-Hellman (CDH): Given $g, x, y \in \mathbb{G}$, compute $t \in \mathbb{G}$ such that $\langle g, x, y, t\rangle$ is a Diffie-Hellman tuple, i. e., of the form $\langle g, g^a, g^b, g^{ab}\rangle$ for some $a, b \in \mathbb{Z}_{|\mathbb{G}|}$.

Decisional Diffie-Hellman (DDH): Given $g, x, y, t \in \mathbb{G}$, decide whether $\langle g, x, y, t\rangle$ is a Diffie-Hellman tuple.

The latter two problems can be transferred to bilinear groups [BF01]. A direct translation by replacing $t \in \mathbb{G}$ with $T \in \mathbb{G}_T$ and g^{ab} with $e(g, g)^{ab}$ would clearly fail due to the bilinearity of e. Therefore, a third element $z = g^c$ comes into play.

Bilinear computational Diffie-Hellman (BCDH): Given $g, x, y, z \in \mathbb{G}$, compute $T \in \mathbb{G}_T$ such that $\langle g, x, y, z, T\rangle$ is a bilinear Diffie-Hellman tuple, i. e., of the form $\langle g, g^a, g^b, g^c, e(g, g)^{abc}\rangle$ for some $a, b, c \in \mathbb{Z}_{|\mathbb{G}|}$.

Bilinear decisional Diffie-Hellman (BDDH): Given $g, x, y, z \in \mathbb{G}$ and $T \in \mathbb{G}_T$, decide whether $\langle g, x, y, z, T\rangle$ is a bilinear Diffie-Hellman tuple.

The assumption that the BDDH problem is hard is used for the security proof of several encryption schemes, so it will be formalized now. Let BGGen be an PPT algorithm that on input 1^λ outputs a tuple $\langle p, \mathbb{G}, \mathbb{G}_T, e, g\rangle$ where $\mathbb{G}$ is a bilinear group of order p, e the bilinear map into $\mathbb{G}_T$ and g a generator of $\mathbb{G}$.

Definition 2.10. *For the bilinear group generator* BGGen*, the* bilinear decision Diffie-Hellman experiment (BDDH) *is defined as:*

$$
\begin{aligned}
&Exp^{\mathsf{BGGen},\mathcal{A}}_{BDDH}(\lambda)\text{:}\\
&\quad p,\mathbb{G},\mathbb{G}_T,e,g &&\leftarrow \mathsf{BGGen}(1^\lambda)\\
&\quad a,b,c &&\leftarrow \mathbb{Z}_p\\
&\quad x,y,z &&\leftarrow g^a,g^b,g^c\\
&\quad T_0 &&\leftarrow e(g,g)^{abc}\\
&\quad T_1 &&\leftarrow \mathbb{G}_T\\
&\quad \hat{b} &&\leftarrow \{0,1\}\\
&\quad b' &&\leftarrow \mathcal{A}(p,\mathbb{G},\mathbb{G}_T,e,g,x,y,z,T_{\hat{b}})\\
&\quad \textit{if } \hat{b}=b',\textit{ then output win}
\end{aligned}
$$

The BDDH is ϵ-hard in groups generated by BGGen if $\epsilon \in negl(\lambda)$ and for every PPT algorithm $\mathcal{A}$ it holds that

$$Pr[\mathrm{Exp}^{\mathsf{BGGen},\mathcal{A}}_{\mathrm{BDDH}}(\lambda)=win] =: \mathrm{Adv}^{\mathsf{BGGen},\mathcal{A}}_{\mathrm{BDDH}}(\lambda) \quad \leq \quad \epsilon(\lambda)$$

2.6 Reductionist Proofs and Game Playing Technique

Proofs by reduction are a very common tool in cryptography. In the following the basic method and the more sophisticated game playing technique are described in a quite abstract fashion. Two extensive examples, exactly employing the following descriptions, are given in Section 3.1 and Section 3.2. Consider the case that the implementation of a concrete scheme (e. g., an IBKEM) has to be proven secure in some sense (e. g., IND-sID-CPA), under the assumption that some number-theoretic problem (e. g., the BDDH problem) is hard. The basic organization of such a proof often is as described in the following.

Let $\mathcal{A}$ be an arbitrary attacker, who tries to break IBKEM in the IND-sID-CPA experiment. Nothing may be assumed about $\mathcal{A}$ except that it conforms to the rules of the IND-sID-CPA experiment. The task is to construct an algorithm $\mathcal{B}$ that somehow uses $\mathcal{A}$ to solve the BDDH problem. $\mathcal{B}$ is given some input values according to the BDDH experiment and has to simulate the environment of $\mathcal{A}$ as defined in the IND-sID-CPA experiment, i. e., set up a master key pair, compute a challenge ciphertext, and answer all queries to the SecKey oracle. $\mathcal{B}$'s goal is to compute all input values for $\mathcal{A}$ from its own input in such a way that $\mathcal{A}$ does not notice the difference (i. e., all values given to $\mathcal{A}$ have exactly the probability distribution as defined in the random experiment and the concrete IBKEM), and $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins. This implies:

$$
\begin{aligned}
\mathrm{Adv}^{\mathsf{IBKEM},\mathcal{A}}_{\text{IND-sID-CPA}}(\lambda) &= \left| Pr[\mathrm{Exp}^{\mathsf{IBKEM},\mathcal{A}}_{\text{IND-sID-CPA}}(\lambda)=win] - \frac{1}{2}\right|\\
&= \left| Pr[\mathrm{Exp}^{\mathsf{BGGen},\mathcal{B}}_{\mathrm{BDDH}}(\lambda)=win] - \frac{1}{2}\right| = \mathrm{Adv}^{\mathsf{BGGen},\mathcal{B}}_{\mathrm{BDDH}}(\lambda)
\end{aligned}
$$

If the BDDH problem is hard by assumption, then there is a bound $\epsilon \in negl(\lambda)$ for the advantage of every algorithm trying to solve it (including $\mathcal{B}$). This gives:

$$\mathrm{Adv}_{\mathrm{BDDH}}^{\mathsf{BGGen},\mathcal{B}}(\lambda) \leq \epsilon(\lambda)$$

Combining both (in-)equalities gives ϵ as upper bound for $\mathcal{A}$'s advantage. Since $\mathcal{A}$ has been chosen arbitrary, the advantage of any attacker of IBKEM has to be less or equal ϵ and thus IBKEM is ϵ-secure.

For more complex proofs, this simple reduction technique does not suffice (or more precisely, would make the proof hard to follow). If this is the case, it may sometimes be helpful to organize the proof in a sequence of games. Assume this time that IBKEM is a scheme which is composed by another IND-sID-CPA secure $\overline{\mathsf{IBKEM}}$ and a TCR-secure hash function H. In many cases, one of the underlying schemes can be considered the main primitive (here $\overline{\mathsf{IBKEM}}$) and the other(s) are auxiliary primitives (here H). Note that for a security proof, somehow two reductions have to be combined, as IBKEM's security relies on *both* underlying primitives.

One solution is to define a sequence of random experiments *Game 1*... *Game n* where *Game 1* equals the original experiment $\mathrm{Exp}_{\text{IND-sID-CPA}}^{\mathsf{IBKEM},\mathcal{A}}(\lambda)$ and *Game i+1* results from a small modification of *Game i*. As the goal of these modifications, finally $\overline{\mathcal{B}}$ (trying to break the main primitive) should be able to easily simulate *Game n* for $\mathcal{A}$ as in the basic reduction technique described above. Furthermore, the difference of $\mathcal{A}$'s winning probability between two consecutive games should be very small, i. e., one should be able to prove

$$\Big|Pr[\textit{Game } i\,^{\mathcal{A}}(\lambda) = \textit{win}] - Pr[\textit{Game } i{+}1\,^{\mathcal{A}}(\lambda) = \textit{win}]\Big| \leq \epsilon_i(\lambda)$$

for suitable $\epsilon_i \in negl(\lambda)$. Often, this can be proven by reducing the differences to security of the auxiliary primitives. Then, $\mathcal{A}$'s advantage can be upper bounded using a telescoping sum and the equations above.

In the concrete example with IBKEM, H, and $\overline{\mathsf{IBKEM}}$, there would be a "sequence" of two games *Game 1* $= \mathrm{Exp}_{\text{IND-sID-CPA}}^{\mathsf{IBKEM},\mathcal{A}}(\lambda)$ and *Game 2*. The difference between *Game 1* and *2* could be bounded by the security of H, and $\mathcal{A}$'s advantage in *Game 2* could be bounded by the security of $\overline{\mathsf{IBKEM}}$:

$$\begin{aligned}
\mathrm{Adv}_{\text{IND-sID-CPA}}^{\mathsf{IBKEM},\mathcal{A}}(\lambda) &= \left|Pr[\mathrm{Exp}_{\text{IND-sID-CPA}}^{\mathsf{IBKEM},\mathcal{A}}(\lambda) = \textit{win}] - \frac{1}{2}\right| \\
&= \left|Pr_{\textit{Game 1}}[\mathcal{A}\ \textit{wins}] - \frac{1}{2}\right| \\
&= \left|Pr_{\textit{Game 1}}[\mathcal{A}\ \textit{wins}] - \underbrace{Pr_{\textit{Game 2}}[\mathcal{A}\ \textit{wins}] + Pr_{\textit{Game 2}}[\mathcal{A}\ \textit{wins}]}_{=\,0} - \frac{1}{2}\right| \\
&\leq \left|Pr_{\textit{Game 1}}[\mathcal{A}\ \textit{wins}] - Pr_{\textit{Game 2}}[\mathcal{A}\ \textit{wins}]\right| + \left|Pr_{\textit{Game 2}}[\mathcal{A}\ \textit{wins}] - \frac{1}{2}\right| \\
&\leq \mathrm{Adv}_{\mathrm{TCR}}^{H,\mathcal{B}}(\lambda) + \mathrm{Adv}_{\text{IND-sID-CPA}}^{\overline{\mathsf{IBKEM}},\overline{\mathcal{B}}}(\lambda) \leq \epsilon_1(\lambda) + \epsilon(\lambda)
\end{aligned}$$

Consequently, for any attacker $\mathcal{A}$ of IBKEM, the advantage of breaking the scheme can be bounded by the sum of two negligible functions. This again is a negligible function and such gives a suitable bound for the security of IBKEM.

A simple but important tool for bounding $|Pr_{Game\ i}[\mathcal{A}\ wins] - Pr_{Game\ i+1}[\mathcal{A}\ wins]|$ is the analysis based on failure events F. Intuitively, if two games are "identical until F occurs", then the difference of $\mathcal{A}$'s winning probability can be bounded by $Pr[F]$. More formally, this can be done with a framework of Bellare and Rogaway [BR06], where games are considered that are "identical until *bad* is set". As here only a simple subset of this framework is used, it is introduced by example: *Game 1* and *Game 2* are "identical until F occurs", whereas *Game 1'* and *Game 2'* are their equivalents, rewritten to be "identical until bad is set".

Game 1:	*Game 1'*:	*Game 2'*:	*Game 2*:
	if F	if F	if F
	$bad \leftarrow true$	$bad \leftarrow true$	do that
do this	do this	do that	
	else	else	else
	do this	do this	do this

Lemma 2.11 (Fundamental Lemma, [BR06])**.** *If Game i and Game i+1 are "identical until bad is set", then*

$$\left|Pr_{Game\ i}[\mathcal{A}\ wins] - Pr_{Game\ i+1}[\mathcal{A}\ wins]\right| \quad \leq \quad Pr_{Game\ i}[bad\ is\ set]$$

Furthermore, bad is set with the same probability in both games

$$Pr_{Game\ i}[bad\ is\ set] = Pr_{Game\ i+1}[bad\ is\ set]$$

If further on, $Pr_{Game\ i}[\mathcal{A}\ wins \mid bad\ is\ set] = \frac{1}{2}$ *or* $Pr_{Game\ i+1}[\mathcal{A}\ wins \mid bad\ is\ set] = \frac{1}{2}$ *holds, then the above can be refined to*[3]*:*

$$\left|Pr_{Game\ i}[\mathcal{A}\ wins] - Pr_{Game\ i+1}[\mathcal{A}\ wins]\right| \quad \leq \quad \frac{1}{2} \cdot Pr_{Game\ i}[bad\ is\ set]$$

With one exception, in the following proofs this lemma will be applied to games that are "only" identical until some failure event F occurs. Their rewriting to games that are identical until *bad* is set is trivial in all cases and thus omitted.

[3] This does not occur in the original lemma, but can be obtained by a simple modification.

3 Review of Related Work

This chapter deals with all schemes that are necessary to understand the following work. These are: A restricted and a full IBKEM version of BB1-IBE (Section 3.1, Section 3.2), BMW-KEM (Section 3.3), the CHK transformation (Section 3.4), and finally a very brief summary of Waters-IBKEM, BMW-PKE and the related ACIK transformation (Section 3.5)

3.1 Hash-Free BB1-IBKEM

In the original paper, BMW-KEM is said to be based on BB1-IBE, but according to the authors' description of the analogy between both schemes [BMW05a, §4.2], BMW-KEM may be interpreted as being based on an IBKEM version of BB1-IBE. For this reason, only the IBKEM (§2.1.3 on p.12) version is presented here.

In this section, a restricted version of BB1-IBKEM is defined where identities may be chosen only from $\mathbb{Z}_p$. In the next section, this will be extended to arbitrary identities by hashing these from $\{0,1\}^*$ into $\mathbb{Z}_p$.

Construction 3.1. *The hash-free version hfBB1-IBKEM is defined from a bilinear group generated by BGGen (Def. 2.9 on p.24) as follows:*

$\mathsf{Setup}(1^\lambda)$	$\langle p, \mathbb{G}, \mathbb{G}_T, e, g\rangle$	$\leftarrow \mathsf{BGGen}(1^\lambda)$
	a, b, c	$\leftarrow \mathbb{Z}_p$
	g_1, g_2, g_3	$\leftarrow g^a, g^b, g^c$
	Z	$\leftarrow e(g_1, g_2)$
	PK	$\leftarrow \langle p, \mathbb{G}, \mathbb{G}_T, e, g, g_1, g_2, g_3, Z\rangle$
	SK	$\leftarrow g_2^a$
$\mathsf{PubKey}(PK, ID \in \mathbb{Z}_p)$	PK_{ID}	$\leftarrow g_1^{ID} \cdot g_3$
$\mathsf{SecKey}(SK, ID \in \mathbb{Z}_p)$	r	$\leftarrow \mathbb{Z}_p$
	SK_{ID}	$\leftarrow \langle g^r, SK \cdot PK_{ID}^r\rangle$
$\mathsf{Encaps}(PK_{ID})$	s	$\leftarrow \mathbb{Z}_p$
	C	$\leftarrow \langle g^s, PK_{ID}^s\rangle$
	K	$\leftarrow Z^s$
$\mathsf{Decaps}(SK_{ID}, C)$	$\langle S_1, S_2\rangle$	$\leftarrow SK_{ID}$
	$\langle C_1, C_2\rangle$	$\leftarrow C$
	K	$\leftarrow e(S_2, C_1)/e(C_2, S_1)$

The correctness can easily be checked using the bilinearity of e:

$$\frac{e(S_2, C_1)}{e(C_2, S_1)} = \frac{e(SK \cdot PK_{ID}^r, g^s)}{e(PK_{ID}^s, g^r)} = \frac{e(SK, g^s) \cdot e(PK_{ID}, g)^{rs}}{e(PK_{ID}, g)^{rs}} = e(g, g)^{abs} = Z^s$$

The proof of IND-sID-CPA security (Def. 2.4 on p.17) is based on the BDDH problem (Def. 2.10 on p.25). It is straightforward, as it exactly follows the "basic reduction technique" (§2.6 on p.25).

Theorem 3.2 (analogously to [BB04, Theorem 4.1], but as IBKEM version)**.** *If the BDDH problem is ϵ-hard in the groups generated by* BGGen*, then:*

*hf*BB1*-IBKEM is ϵ-IND-sID-CPA secure.*

Proof. Let $\mathcal{A}$ be an arbitrary IND-sID-CPA attacker of hfBB1-IBKEM. An algorithm $\mathcal{B}$ has to be constructed that uses $\mathcal{A}$ as help for solving the BDDH problem. The goal is that $\mathcal{B}$ gives a perfect simulation of the IND-sID-CPA experiment for $\mathcal{A}$ such that $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins.

On the left side below, the definition of the IND-sID-CPA experiment is recapitulated, using the concrete algorithms of hfBB1-IBKEM. The right side shows the simulation by $\mathcal{B}$, which runs itself in the BDDH experiment.

The experiment as expected by $\mathcal{A}$:

$$
\begin{array}{ll}
\langle p, \mathbb{G}, \mathbb{G}_T, e\rangle & \leftarrow \mathsf{BGGen}(1^\lambda) \\
a, b, c & \leftarrow \mathbb{Z}_p \\
g_1, g_2, g_3 & \leftarrow g^a, g^b, g^c \\
Z & \leftarrow e(g_1, g_2) \\
PK & \leftarrow \langle p, \mathbb{G}, \mathbb{G}_T, e, \\
 & \qquad g_1, g_2, g_3, Z\rangle \\
SK & \leftarrow g_2^a \\
 & \\
ID^\star & \leftarrow \mathcal{A}_1(1^\lambda) \\
 & \\
\hat{b} & \leftarrow \{0,1\} \\
PK_{ID^\star} & \leftarrow g_1^{ID^\star} \cdot g_3 \\
s & \leftarrow \mathbb{Z}_p \\
C^\star & \leftarrow \langle g^s, PK_{ID}^s\rangle \\
K_0^\star & \leftarrow Z^s \quad (= e(g,g)^{abs}) \\
K_1^\star & \leftarrow \mathbb{G}_T \\
 & \\
b' & \leftarrow \mathcal{A}_2^{\mathsf{SecKey}}(PK, K_{\hat{b}}^\star, C^\star)
\end{array}
$$

The SecKey oracle (for $ID \neq ID^\star$):

$$
\begin{array}{ll}
r & \leftarrow \mathbb{Z}_p \\
SK_{ID} & \leftarrow \langle g^r, SK \cdot PK_{ID}^r\rangle
\end{array}
$$

The experiment as simulated by $\mathcal{B}$:

$\mathcal{B}$'s environment computes:

$$
\begin{array}{ll}
\langle p, \mathbb{G}, \mathbb{G}_T, e\rangle & \leftarrow \mathsf{BGGen}(1^\lambda) \\
a, b, s & \leftarrow \mathbb{Z}_p \\
x, y, z & \leftarrow g^a, g^b, g^s \\
T_0 & \leftarrow e(g,g)^{abs} \\
T_1 & \leftarrow \mathbb{G}_T \\
\hat{b} & \leftarrow \{0,1\}
\end{array}
$$

$\mathcal{B}$ on input $\langle p, \mathbb{G}, \mathbb{G}_T, e, x, y, z, T_{\hat{b}}\rangle$:

$$
\begin{array}{ll}
ID^\star & \leftarrow \mathcal{A}_1(1^\lambda) \\
 & \\
g_1, g_2 & \leftarrow x, y \\
Z & \leftarrow e(g_1, g_2) \\
c' & \leftarrow \mathbb{Z}_p \\
g_3 & \leftarrow g^{c'} \cdot g_1^{-ID^\star} \\
PK & \leftarrow \langle p, \mathbb{G}, \mathbb{G}_T, e, \\
 & \qquad g_1, g_2, g_3, Z\rangle \\
C^\star & \leftarrow \langle z, z^{c'}\rangle \\
K_{\hat{b}}^\star & \leftarrow T_{\hat{b}} \\
b' & \leftarrow \mathcal{A}_2^{\mathsf{SecKey}}(PK, K_{\hat{b}}^\star, C^\star) \\
\text{return} & b' \text{ as own guess for BDDH}
\end{array}
$$

The SecKey oracle as simulated by $\mathcal{B}$:

$$
\begin{array}{ll}
r' & \leftarrow \mathbb{Z}_p \\
SK_{ID} & \leftarrow \langle g^{r'} \cdot g_2^{\frac{-1}{ID-ID^\star}}, PK_{ID}^{r'} \cdot g_2^{\frac{-c'}{ID-ID^\star}}\rangle
\end{array}
$$

($\mathcal{B}$ never divides by 0, as $ID \neq ID^\star$)

The high-level strategy of $\mathcal{B}$ is as follows. The values from $\mathcal{B}$'s environment are used in such a way that $T_0 = K_0^\star$ and $T_1 = K_1^\star$, and thus, $\mathcal{B}$ can solve the BDDH problem if and only if $\mathcal{A}$ distinguishes honestly computed keys from random. Unfortunately, by doing this, $\mathcal{B}$ neither knows the master secret key SK $(=g^{ab})$ nor the randomness s used in the challenge ciphertext. Nevertheless, $\mathcal{B}$ must be able to compute $SK \cdot PK_{ID}^r$ for the SecKey oracle and $PK_{ID^\star}^s$ for the challenge ciphertext. The crucial idea that enables $\mathcal{B}$ to simulate these steps is the special choice of g_3 based on $ID^\star$. It is important to note that although g_3 is based computationally on $ID^\star$, both are stochastically independent.

The formal argument now is as follows. First of all, it has to be shown that $\mathcal{B}$ gives a perfect simulation, i. e., all values given to $\mathcal{A}$ by $\mathcal{B}$ have the probability distribution as defined in the original experiment:

- It is easy to see that most values—namely $p, \mathbb{G}, \mathbb{G}_T, e, a, b, g_1\ (=x), g_2\ (=y)$, $Z, \hat{b}, s,$ the first part of $C^\star\ (=z), K_0^\star\ (=T_0), K_1^\star\ (=T_1)$ and $K_{\hat{b}}^\star\ (=T_{\hat{b}})$—are computed as expected by comparing the left and right side above. The only difference is the order of their computation and their initial naming, as some of them are computed by $\mathcal{B}$'s environment instead of $\mathcal{B}$ itself.

- g_3 is expected to be computed as g^c for c uniform in $\mathbb{Z}_p$.
  Instead, $\mathcal{B}$ computes it as $g^{c'} \cdot g_1^{-ID^\star}$, which equals g^c for $c := c' - a \cdot ID^\star$. If c' is uniform in $\mathbb{Z}_p$ then c is, too, no matter what a or $ID^\star$ is. As result, g_3 is distributed correctly, although $\mathcal{B}$ does not know the corresponding c.

- The second part of $C^\star$ is expected to be $PK_{ID^\star}^s = \left(g_1^{ID^\star} \cdot g_3\right)^s$. But this is exactly what $\mathcal{B}$ computes, because $z^{c'} = (g^s)^{c'} = \left(g^{c'}\right)^s = \Big(\underbrace{(g^{c'} \cdot g_1^{-ID^\star})}_{=g_3} \cdot g_1^{ID^\star}\Big)^s$.

- The argument for the first part of a secret key SK_{ID} is the same as for g_3, leading to $g^{r'} \cdot g_2^{\frac{-1}{ID-ID^\star}} = g^r$ for $r := r' - 1/(ID - ID^\star)$. By easy (but bulky) computation one can evaluate that the second part is correct, too, because $PK_{ID}^{r'} \cdot g_2^{\frac{-c'}{ID-ID^\star}} = SK \cdot PK_{ID}^r$.

Then, as all input values of $\mathcal{A}$ have the same probability distribution in the simulation and in the original experiment, it can be concluded that $\mathcal{A}$'s output b' must be distributed the same in both experiments, too. Furthermore, $\hat{b}$ is in both cases computed in the same way. In total, $b = \hat{b}$ must hold with the same probability:

$$Pr_{\mathrm{Exp}^{\text{hfBB1-IBKEM},\mathcal{A}}_{\text{IND-sID-CPA}}(\lambda)}\left[\hat{b} = b'\right] = Pr_{\text{simulation by } \mathcal{B}}\left[\hat{b} = b'\right] = Pr_{\mathrm{Exp}^{\mathsf{BGGen},\mathcal{B}}_{\text{BDDH}}(\lambda)}\left[\hat{b} = b'\right]$$

As described in Section 2.6 (p.25), this gives an upper bound for the advantage of every attacker $\mathcal{A}$ for hfBB1-IBKEM and finishes the proof.

$$\begin{aligned}\mathrm{Adv}^{\text{hfBB1-IBKEM},\mathcal{A}}_{\text{IND-sID-CPA}}(\lambda) &= \left|Pr[\mathrm{Exp}^{\text{hfBB1-IBKEM},\mathcal{A}}_{\text{IND-sID-CPA}}(\lambda) = win] - \frac{1}{2}\right| \\ &= \left|Pr[\mathrm{Exp}^{\mathsf{BGGen},\mathcal{B}}_{\text{BDDH}}(\lambda) = win] - \frac{1}{2}\right| = \mathrm{Adv}^{\mathsf{BGGen},\mathcal{B}}_{\text{BDDH}}(\lambda) \leq \epsilon(\lambda)\end{aligned}$$

□

As this was the first proof, it was given quite in detail. In the following proofs, only the implementation of $\mathcal{B}$ will be given—not its embedding in its own attack game or the concrete experiment that $\mathcal{A}$ expects. It will be much easier to see that $\mathcal{B}$'s simulation fits $\mathcal{A}$'s expectation and often subsumed by a sentence like "$\mathcal{B}$ perfectly simulates ... as expected by $\mathcal{A}$ and consequently wins if and only if $\mathcal{A}$ wins".

The ciphertexts in hfBB1-IBKEM can be characterized as follows:

Fact 3.3. *Let PK_{ID} a valid public key, $C = \langle C_1, C_2 \rangle$ be an arbitrary ciphertext and $C_1 = g^s$ (recall: g generates $\mathbb{G}$). Then the following statements are equivalent:*

1. *C is valid for PK_{ID}, i. e., $C \in \mathcal{C}(PK_{ID})$*
2. $C_2 = PK_{ID}^{s'}$ *and* $s' = s$ *(if PK_{ID} is a generator)*
   $C_2 = g^{s'}$ *and* $s' = 0$ *(if not, i. e., $PK_{ID} = 1$ as $\mathbb{G}$ has prime order)*
3. $C_2 = C_1^{a \cdot ID + c}$ *(recall: $g_1 = g^a$, $g_3 = g^c$ and $PK_{ID} = g_1^{ID} \cdot g_3$)*
4. $e(C_2, g) = e(C_1, PK_{ID})$

3.2 Full BB1-IBKEM

Now, hfBB1-IBKEM can be turned into full BB1-IBKEM that supports arbitrary identities from $\{0,1\}^*$ (§2.1.3 on p.12). The construction is quite easy and only requires the application of a suitable hash function (§2.4.1 on p.22).

Construction 3.4. *The transformation $\mathsf{IBKEM} := arbID(\overline{\mathsf{IBKEM}}, H)$ turns any $\overline{\mathsf{IBKEM}}$ with a restricted identity space $IDSpace$ and a keyed hash function $H_k : \{0,1\}^* \to IDSpace$ into an IBKEM with arbitrary identities. It is defined as follows:*

$\mathsf{IBKEM.Setup}(1^\lambda)$	$\overline{PK}, SK$	$\leftarrow \overline{\mathsf{IBKEM.Setup}}(1^\lambda)$
	k	$\leftarrow \{0,1\}^{\ell(\lambda)}$
	PK	$\leftarrow \langle \overline{PK}, k \rangle$
$\mathsf{IBKEM.PubKey}(PK, ID)$	$\overline{ID}$	$\leftarrow H_k(ID)$
	PK_{ID}	$\leftarrow \overline{\mathsf{IBKEM.PubKey}}(\overline{PK}, \overline{ID})$
$\mathsf{IBKEM.SecKey}(SK, ID)$	$\overline{ID}$	$\leftarrow H_k(ID)$
	SK_{ID}	$\leftarrow \overline{\mathsf{IBKEM.SecKey}}(SK, \overline{ID})$

Encaps and Decaps stay unchanged.

The proof of IND-sID-CPA security (Def. 2.4 on p.17) will be a good example for the game playing technique (second half of §2.6 on p.26). As this is the first use of the "fundamental lemma" (Lem. 2.11 on p.27), things will be done in great detail.

Theorem 3.5 ([BB04, Theorem 8.1])**.** *If $\overline{\mathsf{IBKEM}}$ is ϵ-IND-sID-ATK secure (for $ATK \in \{CPA, CCA_{1/2}\}$) and H is $\hat{\epsilon}$-TCR secure (Def. 2.7 on p.22), then:*

$$\mathsf{IBKEM} := arbID(\overline{\mathsf{IBKEM}}, H) \text{ is } (\epsilon + \hat{\epsilon}/2)\text{-IND-sID-ATK secure.}$$

Proof. The proof is only given for IND-sID-CPA security. It can easily be adapted for the other two cases of IND-sID-CCA$_1$ and IND-sID-CCA$_2$ security.

The security of IBKEM is heavily based on $\overline{\mathsf{IBKEM}}$. Indeed, an attacker $\mathcal{A}$ of IBKEM can be turned almost directly into an attacker $\overline{\mathcal{B}}$ of $\overline{\mathsf{IBKEM}}$. A problem will occur only if $\mathcal{A}$ requests a secret key for some ID with $H_k(ID) = H_k(ID^\star)$, but this can be avoided due to the TCR-security of H. The proof is based on the following "sequence" of two games.

Game 1 This is the IND-sID-CPA experiment for IBKEM.

Game 2 This is the same as *Game 1*, except if $\mathcal{A}$ queries the IBKEM.SecKey oracle for an $ID \neq ID^\star$ with $H_k(ID) = H_k(ID^\star)$. Then the experiment will be aborted, i. e., $\mathcal{A}$ is stopped and its guess is chosen randomly ($b' \leftarrow \{0,1\}$).

Proof Overview The change from *Game 1* to *2* can be bounded by an algorithm $\mathcal{B}$ attacking the security of the auxiliary primitive H. This results in:

$$|Pr^{\mathcal{A}}_{Game\ 1}[b = b'] - Pr^{\mathcal{A}}_{Game\ 2}[b = b']| \quad \leq \quad \mathrm{Adv}^{H,\mathcal{B}}_{\mathrm{TCR}}(\lambda)/2 \quad \leq \quad \hat{\epsilon}(\lambda)/2$$

Game 2 can be perfectly simulated by attacker $\overline{\mathcal{B}}$ of the main primitive $\overline{\mathsf{IBKEM}}$:

$$|Pr^{\mathcal{A}}_{Game\ 2}[b = b'] - 1/2| \quad = \quad \mathrm{Adv}^{\overline{\mathsf{IBKEM}},\overline{\mathcal{B}}}_{\text{IND-sID-CPA}}(\lambda) \quad \leq \quad \epsilon(\lambda)$$

Combining both equations bounds $\mathcal{A}$'s advantage (as described in §2.6 on p.26):

$$\mathrm{Adv}^{\mathsf{IBKEM},\mathcal{A}}_{\text{IND-sID-CPA}}(\lambda) \quad \leq \quad \epsilon(\lambda) + \hat{\epsilon}(\lambda)/2$$

***Game 1* to *Game 2*:** The analysis is based on failure events, a technique introduced together with the fundamental lemma. The failure event is $F :=$ "$\mathcal{A}$ makes a query to the IBKEM.SecKey oracle for an ID with $H_k(ID) = H_k(ID^\star)$". In order to derive the desired bound on the difference of *Game 1* and *Game 2* the following three facts have to be proven.

- *Game 1* and *2* can be rewritten to games that are "identical until *bad* is set" and where *bad* is set if and only if F occurs.
- If F occurs, then $\mathcal{A}$ wins *Game 2* with probability 1/2.
- An algorithm $\mathcal{B}$ can be constructed that breaks the TCR-security of H if and only if F occurs in *Game 2*.

The first step gives the applicability of the lemma. The second step allows to even use the slightly sharper bound at the end of that lemma. This results in:

$$|Pr^{\mathcal{A}}_{Game\ 1}[b = b'] - Pr^{\mathcal{A}}_{Game\ 2}[b = b']| \quad \leq \quad Pr_{Game\ 2}[F]/2$$

The third step bounds

$$Pr_{Game\ 2}[F] \quad = \quad Pr_{\mathrm{Exp}^{H,\mathcal{B}}_{\mathrm{TCR}}(\lambda)}[\mathcal{B}\ wins] \quad = \quad \mathrm{Adv}^{H,\mathcal{B}}_{\mathrm{TCR}}(\lambda) \quad \leq \quad \hat{\epsilon}(\lambda)$$

Let's begin with the first step: As both games differ only in the implementation of the SecKey oracle, only these parts are rewritten:

$\mathsf{SecKey}(ID)$ oracle in *Game 1*:	$\mathsf{SecKey}(ID)$ oracle in *Game 2*:
if $ID \neq ID^\star \wedge H_k(ID) = H_k(ID^\star)$	if $ID \neq ID^\star \wedge H_k(ID) = H_k(ID^\star)$
$\quad bad \leftarrow true$	$\quad bad \leftarrow true$
$\quad$ proceed	$\quad$ abort
if $ID = ID^\star$	if $ID = ID^\star$
$\quad$ abort	$\quad$ abort
$\overline{ID} \leftarrow H_k(ID)$	$\overline{ID} \leftarrow H_k(ID)$
$SK_{ID} \leftarrow \overline{\mathsf{IBKEM}}.\mathsf{SecKey}(SK, \overline{ID})$	$SK_{ID} \leftarrow \overline{\mathsf{IBKEM}}.\mathsf{SecKey}(SK, \overline{ID})$

Both oracles are equivalent to those as defined in *Game 1* and *Game 2*. The last four lines give an oracle as defined in the IND-sID-CPA experiment, and thus as in *Game 1*: For any $ID \neq ID^\star$, it is expected to output a secret key computed as defined in the $arbID$ transformation (Con. 3.4 on p.32). The first three lines have no impact on the left side. On the right side, they implement the newly introduced abort in *Game 2*. It is obvious that *bad* is set to true if and only if F occurs. This finishes the first of the above stated steps.

Now assume that F occurs in *Game 2*. Then, by definition of *Game 2*, the experiment is aborted, i. e., $\mathcal{A}$ is stopped and its guess is chosen as $b' \leftarrow \{0,1\}$. Therefore, it holds with probability $1/2$ that $b = b'$, i. e., $Pr_{Game\ 2}[\mathcal{A}\ wins \mid F] = 1/2$. This allows the application of the third part of the fundamental lemma.

As last step, an algorithm $\mathcal{B}$ has to be created that simulates *Game 2* for $\mathcal{A}$ and attacks H in the TCR experiment. Whenever F occurs during that simulation, $\mathcal{B}$ has to break H, i. e., find an x that collides in H with a previously selected target $x^\star$. $\mathcal{B}$ is defined as follows:

$\mathcal{B}_1(1^\lambda)$:
- run $\mathcal{A}_1(1^\lambda)$ to obtain $ID^\star$
- output $x^\star \leftarrow ID^\star$

$\mathcal{B}_2(k)$:
- $\overline{PK}, SK \leftarrow \overline{\mathsf{IBKEM}}.\mathsf{Setup}(1^\lambda)$
- $PK \leftarrow \langle \overline{PK}, k \rangle$
- compute $b, PK_{ID^\star}, K_0^\star, C^\star, K_1^\star$ as expected
- start $\mathcal{A}_2^{\mathsf{IBKEM.SecKey}}(PK)$
- if the simulation has not been aborted by $\mathcal{B}$ during the IBKEM.SecKey oracle, then output $x \leftarrow ID^\star$

IBKEM.SecKey-oracle(ID) for $\mathcal{A}$:

- if $ID \neq ID^\star \wedge H_k(ID) = H_k(ID^\star)$ $\leftarrow F$ occurs
  then abort and output $x \leftarrow ID$
- if $ID = ID^\star$
  then abort and output $x \leftarrow ID^\star$
- else compute and return SK_{ID} as expected

In fact, $\mathcal{B}$ does all computations of *Game 2* on its own, except choosing the hash key k. After $\mathcal{A}_1$ has selected $ID^\star$, $\mathcal{B}$ outputs this identity as target $x^\star$ for attacking the hash function H. In return, it is given a hash key k which it uses to complete PK. In this way, $\mathcal{B}$ has generated a valid public key PK *as well as the secret key* SK and thus is able to give a perfect simulation, including the SecKey oracle. If F occurs, then $\mathcal{B}$ outputs an x with $x = ID \neq ID^\star = x^\star$ and $H_k(x) = H_k(ID) = H_k(ID^\star) = H_k(x^\star)$, i. e., it wins in the TCR experiment for H. On the other hand, if F does not occur, then $\mathcal{B}$ eventually outputs x with $x = ID^\star = x^\star$ and definitely does not win. In total, $\mathcal{B}$ wins if and only if F occurs. This ends the proof of the third step above and finishes the bounding of differences between *Game 1* and *2*.

***Game 2* to $\overline{\text{IBKEM}}$:** Now, an IND-sID-CPA attacker $\overline{\mathcal{B}}$ of $\overline{\text{IBKEM}}$ can easily simulate *Game 2* for $\mathcal{A}$. It delegates all $\overline{\text{IBKEM}}$ related computations to its environment and only adds the hash function. $\mathcal{B}$ is defined as follows:

$\overline{\mathcal{B}}_1(1^\lambda)$**:**
- start $\mathcal{A}_1(1^\lambda)$ to obtain the attacked identity $ID^\star$
- choose a hash key $k \leftarrow \{0,1\}^{\ell(\lambda)}$
- output as own attacked identity $\overline{ID^\star} \leftarrow H_k(ID^\star)$

$\overline{\mathcal{B}}_2^{\overline{\text{IBKEM}}.\text{SecKey}}(\overline{PK}, K_b^\star, C^\star)$:
- set $PK \leftarrow \langle \overline{PK}, k \rangle$
- start $\mathcal{A}_2^{\text{IBKEM.SecKey}}(PK, K_b^\star, C^\star)$
- use $\mathcal{A}_2$'s output b' as own guess

IBKEM.SecKey-oracle(ID) for $\mathcal{A}_2$:
- if $H_k(ID) = H_k(ID^\star)$ abort, else
- $\overline{ID} \leftarrow H_k(ID)$ $\quad (\neq H_k(ID^\star) = \overline{ID^\star})$
- $SK_{ID} \leftarrow \overline{\text{IBKEM}}.\text{SecKey-oracle}(\overline{ID})$

Due to $\mathcal{B}$'s choice $\overline{ID^\star} \leftarrow H_k(ID^\star)$, the challenge ciphertext is computed (by $\mathcal{B}$'s environment) with respect to the public key $\overline{\text{IBKEM}}.\text{PubKey}(\overline{PK}, \overline{ID^\star})$. This is equivalent to IBKEM.PubKey$(PK, ID^\star)$. In the same way, all answers of the IBKEM.SecKey-oracle fulfill $\mathcal{A}$'s expectations. $\mathcal{B}$'s $\overline{\text{IBKEM}}$.SecKey oracle queries are always permitted, as $\overline{ID} \neq \overline{ID^\star}$ always holds. Consequently, $\mathcal{B}$ gives a perfect simulation and wins exactly if $\mathcal{A}$ wins. □

The steps to show the applicability of the fundamental lemma (the biggest part of "*Game 1* to *Game 2*") have been quite technical but not very difficult or enlightening. As a consequence, they will be left out in the following proofs. Instead, only the probability for F to occur will be analyzed.

This construction (Con. 3.4 on p.32), applied to the hash-free variant of BB1-IBKEM, gives the full version. Let H be a hash function from $\{0,1\}^*$ to $\mathbb{Z}_p$, then:

$$\text{BB1-IBKEM} := arbID(\text{hfBB1-IBKEM}, H)$$

Corollary 3.6. *If the BDDH problem is ϵ-hard in the groups generated by BGGen (Def. 2.10 on p.25) and H is $\hat{\epsilon}$-TCR secure (Def. 2.7 on p.22), then:*

BB1-IBKEM is $(\epsilon + \hat{\epsilon}/2)$-IND-sID-CPA secure.

3.3 BMW-KEM

BMW-KEM is a direct construction of an IND-CCA$_2$ secure KEM (§2.1.2 on p.11 and Def. 2.3 on p.16). In fact, in can be interpreted as a very efficient application of paradigm 1 and 2 from the introduction (§1.3 on p.5) to BB1-IBKEM.

Construction 3.7. *BMW-KEM is defined as follows:*

KeyGen(1^λ)	*exactly as Setup in BB1-IBKEM but*	
	SK	$\leftarrow \langle g_2^a, a, c\rangle$
Encaps(PK)	s	$\leftarrow \mathbb{Z}_p$
	C	$\leftarrow \langle C_1, C_2\rangle = \langle g^s, (g_1^{H_k(C_1)} \cdot g_3)^s\rangle$
	K	$\leftarrow Z^s$
Decaps(SK, C)	$\langle \overline{SK}, a, c\rangle$	$\leftarrow SK$
	$\langle C_1, C_2\rangle$	$\leftarrow C$
	check	$C_1^{a\cdot H_k(C_1)+c} \stackrel{?}{=} C_2$
	if not	*then return* $K \leftarrow \perp$
	else	*return* $K \leftarrow e(\overline{SK}, C_1)$

In comparison to BB1-IBKEM, the encapsulation algorithm is subject to a subtle but important change. Instead of getting an PK_{ID} as external input, the first part of the ciphertext is used to compute "PK_{C_1}" as base for C_2. The decapsulation algorithm is modified a little bit more, but this will turn out as a comparatively unimportant change. Note that there is no overhead in BMW-KEM compared to BB1-IBKEM. Ciphertext size is the same and the Decaps algorithm is even more efficient in BMW-KEM. The exact relationship between both schemes and the realization of the paradigms will be studied in Chapter 4.

Note: This version of BMW-KEM maps C_1 into $\mathbb{Z}_p$ using a keyed hash function H_k. The main version of [BMW05a] improves this by replacing H_k with a (non-keyed) function $H\colon \mathbb{G} \to \mathbb{Z}_p$, which is "almost injective". It is not considered here.

Theorem 3.8 (almost[1] as [BMW05a, Theorem 4.1])**.** *If the BDDH problem is ϵ-hard in the groups generated by* BGGen *and H is $\hat{\epsilon}$-TCR secure, then follows:*

$$\textsf{BMW-KEM} \text{ is a } (\epsilon + \hat{\epsilon}/2 + q_{\textsf{Decaps}}/2p)\text{-IND-CCA}_2 \text{ secure KEM,}$$

where $q_{\textsf{Decaps}}$ is the number of oracle queries and p is the size of the bilinear group.

3.4 The CHK Transformation

The CHK transformation is a generic transformation from any IND-sID-CPA secure IBE and any strongly existentially unforgeable one-time signature to IND-CCA$_2$ secure PKE. Here, an IBKEM-to-KEM version is presented (for all definitions see §2.1 on p.10 and §2.4.2 on p.23).

Construction 3.9. *The CHK(-KEM) transformation* $\textsf{KEM} := CHK(\overline{\textsf{IBKEM}}, \textsf{Sig})$ *for an arbitrary* $\overline{\textsf{IBKEM}}$ *and a signature scheme* Sig *is defined as follows:*

$\textsf{KEM.KeyGen}(1^\lambda)$	PK, SK	$\leftarrow \overline{\textsf{IBKEM}}.\textsf{Setup}(1^\lambda)$
$\textsf{KEM.Encaps}(PK)$	$VK, SigK$	$\leftarrow \textsf{Sig.KeyGen}(1^\lambda)$
	ID	$\leftarrow VK$
	PK_{ID}	$\leftarrow \overline{\textsf{IBKEM}}.\textsf{PubKey}(PK, ID)$
	$K, \overline{C}$	$\leftarrow \overline{\textsf{IBKEM}}.\textsf{Encaps}(PK_{ID})$
	sig	$\leftarrow \textsf{Sig.Sign}(SigK, \overline{C})$
	C	$\leftarrow \langle ID, \overline{C}, sig \rangle$
	return	K, C
$\textsf{KEM.Decaps}(SK, C)$	$\langle ID, \overline{C}, sig \rangle$	$\leftarrow C$
	VK	$\leftarrow ID$
	check	$\textsf{Sig.Verify}(VK, \overline{C}, sig) \stackrel{?}{=} true$
	if not	*then return* $K \leftarrow \perp$
	SK_{ID}	$\leftarrow \overline{\textsf{IBKEM}}.\textsf{SecKey}(SK, ID)$
	K	$\leftarrow \overline{\textsf{IBKEM}}.\textsf{Decaps}(SK_{ID}, \overline{C})$

For each key encapsulation, a fresh verification key VK is chosen and used as ID. The session key K is encapsulated under this ID, and the resulting ciphertext $\overline{C}$ is signed with the corresponding signing key $SigK$. For decapsulation, the signature is checked first, and then K is restored with the appropriate secret key. The computational overhead of KEM compared to $\overline{\textsf{IBKEM}}$ is given by a signing key generation and a signature computation for encapsulation, and a signature verification for decapsulation. The ciphertext is enlarged by a verification key and a signature.

Paradigm 1 from the introduction (§1.3 on p.5) is not realized exactly in the proposed way. ID is chosen randomly from all valid verification keys VK of Sig

[1] The $\hat{\epsilon}/2$ in the security bound does not occur in [BMW05a] as the authors prove the version where H is "almost injective". If they also had proven the version with a hash function, the security bound would be exactly as above.

instead of arbitrary binary strings. This subset is still big enough to make $ID = VK$ unpredictable for an attacker and thus render the Decaps oracle in phase 1 useless.

The reason for paradigm 2 to be fulfilled is roughly as follows. Assume that an attacker is given a ciphertext $\langle ID^\star, \overline{C}^\star, sig^\star \rangle$, and he manages to submit a valid ciphertext $\langle ID^\star, \overline{C}, sig \rangle$ to the Decaps oracle. This would include a signature sig for the "message" $\overline{C}$ that is valid under the verification key $ID^\star = VK^\star$. Such a forgery can easily be converted into an SEU-OT attacker for Sig (Def. 2.8 on p.23) and thus is infeasible. Consequently, the Decaps oracle in phase 2 is useless, too.

Together with the IND-sID-CPA security of $\overline{\mathsf{IBKEM}}$ (Def. 2.4 on p.17) this gives IND-CCA$_2$ security for KEM (Def. 2.3 on p.16):

Theorem 3.10 (analogously to [CHK04, Thm. 1] but as IBKEM-to-KEM version)**.** *Let $\overline{\mathsf{IBKEM}}$ be ϵ-IND-sID-CPA secure and Sig be $\hat{\epsilon}$-SEU-OT secure, then follows:*

$$CHK(\overline{\mathsf{IBKEM}}, \mathsf{Sig}) \text{ is a } (\epsilon + \hat{\epsilon})\text{-IND-CCA}_2 \text{ secure } \mathsf{KEM}.$$

This will not be proven here. Instead, the analysis of the relationship between the CHK transformation and BMW-KEM in Chapter 5 yields a similar theorem that implies a weakened version of the above as corollary. A modification to produce exactly the above theorem as corollary will also be briefly suggested.

3.5 BMW-PKE and the ACIK Transformation

Here, an IBKEM version of Waters-IBE and the related BMW-PKE are sketched briefly and then the two ACIK transformations are defined. This section shall enable the reader to compare the work of [ACIK07] with the ideas presented in later chapters. Therefore, it is not given with much explanations. The reader might want to skim over this section for now and return after reading Chapter 4.

Waters-IBKEM and BMW-PKE: The IBKEM version of Waters-IBE is closely related to hfBB1-IBKEM (Con. 3.1 on p.29). The main difference is the computation of the users' public keys. BB1-IBKEM assumes $ID \in \mathbb{Z}_p$ and computes $PK_{ID} \leftarrow g_1^{ID} \cdot g_3$. Waters-IBKEM uses an $ID \in \{0,1\}^n$ (where n is arbitrary but fixed), parses this as $(I_1, \ldots, I_n)$ and computes:

$$PK_{ID} \leftarrow \Big(\prod_{I_j=1} h_j \Big) \cdot g_3$$

where $h_1, \ldots, h_n \leftarrow \mathbb{G}$ are additional elements in PK chosen during Setup. The other algorithms SecKey, Encaps and Decaps work as in hfBB1-IBKEM. This special form of computing PK_{ID} is also known as the Waters hash of an identity. It makes the crucial difference that allows to prove security under adaptive-identity attacks.

Theorem 3.11 (analogously to [Wat05, Theorem 1], but as IBKEM version)**.** *If the BDDH problem is ϵ-hard in the groups generated by BGGen, then:*

$$\text{Waters-IBKEM is } \big(32 \cdot (n+1) \cdot q_{\mathsf{SecKey}} \cdot \epsilon\big)\text{-IND-aID-CPA secure.}$$

where q_{SecKey} is the number of the attacker's oracle queries.

The relationship between BMW-PKE and BMW-KEM is quite similar, although the differences are a little bit bigger. In BMW-KEM a random key and ciphertext are encapsulated as $K \leftarrow Z^s$, $C_1 \leftarrow g^s$ and $C_2 \leftarrow (g_1^{H_k(ID)} \cdot g_3)^s$ for a randomly chosen $s \in \mathbb{Z}_p$. In BMW-PKE on the other hand, a message $M \in \mathbb{G}_T$ is encrypted as:

$$C_M \leftarrow M \cdot Z^s, \qquad C_1 \leftarrow g^s, \qquad (I_1, \ldots, I_n) \leftarrow H_k(C_M \,||\, C_1), \qquad C_2 \leftarrow \Big(\prod_{I_j=1} h_j \Big)^s$$

I. e., Z^s is not used as session key K but to hide M. Furthermore in C_2 the Waters hash is used and applied to both C_M and C_1. The decapsulation can be done in an efficient way similar to BMW-KEM by using the discrete logarithms of all h_i. The security relation between BMW-PKE and Waters-IBKEM is as between BMW-KEM and BB1-IBKEM:

Theorem 3.12 (almost[2] as [BMW05a, Theorem 3.1])**.** *If the BDDH problem is ϵ-hard in the groups generated by BGGen and H is $\hat{\epsilon}$-TCR secure, then follows:*

$$\textit{BMW-PKE is a } \big(32 \cdot (n+1) \cdot q_{\textit{Decrypt}} \cdot \epsilon + \hat{\epsilon}/2\big)\textit{-IND-CCA}_2 \textit{ secure KEM,}$$

where $q_{\textit{Decaps}}$ is the number of oracle queries and p is the size of the bilinear group.

The ACIK transformations: As noted in the introduction Abe et al. [ACIK07] have extracted the specific properties of Waters-IBKEM that allow its transformation into BMW-PKE. The results are the following definitions and transformations:

Definition 3.13 ([ACIK07, Definition 2])**.** *An IBKEM (§2.1.3 on p.12) is said to be* ACIK-partitioned *if (for all possible public/secret keys and identities) it satisfies the following three properties:*

- Independence property of K*: The encapsulated key K does not depend on ID.*
- Unique split property of C*: The ciphertext C can be split into two parts $C = \langle C_1, C_2 \rangle$ such that the first part C_1 does not depend on ID. Furthermore, the first part C_1 and ID together uniquely determine the second part C_2.*
- Perfect \$-rejection property*: For every invalid $C = \langle C_1, C_2 \rangle \notin \mathcal{C}(PK_{ID})$ it is required that IBKEM.Decaps(IBKEM.SecKey$(SK, ID), \langle C_1, C_2 \rangle)$ outputs a random $K \in \mathcal{K}(PK)$.*

These partition properties imply that IBKEM.Encaps can be split into two parts, $\mathsf{IBKEM.Encaps}_1$, and $\mathsf{IBKEM.Encaps}_2$ that create $\langle K, C_1 \rangle$ and C_2, respectively.

$\mathsf{Encaps}_1(PK) \rightsquigarrow K, C_1, \sigma$; which outputs the session key K, first part of the ciphertext C_1 and some important values σ from its internal state. E. g., σ could be all randomness produced by the internal coin tosses. It is meant only as input for the second stage and has to be deleted afterwards.

$\mathsf{Encaps}_2(PK_{ID}, \sigma) \rightsquigarrow C_2$; (deterministic) computes the second ciphertext part C_2.

[2] As for BMW-KEM the $\hat{\epsilon}/2$ in the security bound does not occur in [BMW05a] as the authors prove the version where H is "almost injective".

Furthermore, they define a strengthened version of IND-aID-CPA security. Compared to the standard version of this security definition (Def. 2.4 on p.17), the main difference is that $K_b^\star$ and $C_1^\star$ can be computed independently of $ID^\star$ and are given to the attacker already at the beginning of phase 1.

Definition 3.14 ([ACIK07, §5.3])**.** *For an ACIK-partitioned* IBKEM*, the* strong IND-aID-CPA *experiment is defined as follows:*

$$
\begin{aligned}
&Exp^{\mathsf{IBKEM},\mathcal{A}}_{\text{strong-IND-aID-CPA}}(\lambda)\text{:}\\
&\quad PK, SK \leftarrow \mathsf{Setup}(1^\lambda)\\
&\quad K_0^\star, C_1^\star, \sigma^\star \leftarrow \mathsf{Encaps}_1(PK)\\
&\quad K_1^\star \leftarrow \mathcal{K}(PK)\\
&\quad b \leftarrow \{0,1\}\\
&\quad ID^\star \leftarrow \mathcal{A}_1^{\mathsf{SecKey}}(PK, K_b^\star, C_1^\star)\\
&\quad PK_{ID^\star} \leftarrow \mathsf{PubKey}(PK, ID^\star)\\
&\quad C_2^\star \leftarrow \mathsf{Encaps}_2(PK_{ID^\star}, \sigma^\star)\\
&\quad b' \leftarrow \mathcal{A}_2^{\mathsf{SecKey}}(C_2^\star)\\
&\quad \textit{if } b = b', \textit{ then output win}
\end{aligned}
$$

On query SecKey*-oracle*(ID)*, the oracle uses the* SecKey *algorithm to compute* SK_{ID} *and returns that key. The queried* ID *may not be* $ID^\star$*. Furthermore,* $\mathcal{A}_1$ *may not output an* $ID^\star$ *that has been part of a* SecKey *query before.*

Now all preconditions for the first[3] ACIK transformation are met. In fact, the authors give a construction that turns a partitioned IBKEM (§2.1.3 on p.12) into a *Tag-KEM* [AGKS05]. This is a special case of a TBKEM (§2.1.4 on p.13) which allows transformations into PKE (§2.1.1 on p.10) that are more efficient than KEM-to-PKE transformations (§2.3 on p.19). To avoid the need of yet another definition, here a simplified version of their transformation is defined which directly gives PKE.

Construction 3.15 ([ACIK07, §5.3])**.** *The first ACIK transformation* $\mathsf{PKE} := ACIK_1(\overline{\mathsf{IBKEM}})$ *for an ACIK-partitioned* $\overline{\mathsf{TBKEM}}$ *is defined as follows:*

$\mathsf{PKE.KeyGen}(1^\lambda)$	PK, SK	$\leftarrow \overline{\mathsf{IBKEM}}.\mathsf{Setup}(1^\lambda);$
$\mathsf{PKE.Encrypt}(PK, M)$	$\overline{K}, \overline{C_1}, \overline{\sigma}$	$\leftarrow \overline{\mathsf{IBKEM}}.\mathsf{Encaps}_1(PK)$
	$\overline{C_M}$	$\leftarrow M \oplus \overline{K}$
	$\overline{ID}$	$\leftarrow \overline{C_M} \mid\mid \overline{C_1}$
	$PK_{\overline{ID}}$	$\leftarrow \mathsf{PubKey}(PK, \overline{ID})$
	$\overline{C_2}$	$\leftarrow \overline{\mathsf{IBKEM}}.\mathsf{Encaps}_2(PK_{\overline{ID}}, \overline{\sigma})$
	C	$\leftarrow \langle \overline{C_M}, \overline{C_1}, \overline{C_2} \rangle$
$\mathsf{PKE.Decaps}(SK, C)$	$\langle \overline{C_M}, \overline{C_1}, \overline{C_2} \rangle$	$\leftarrow C$
	$\overline{ID}$	$\leftarrow \overline{C_M} \mid\mid \overline{C_1}$
	$SK_{\overline{ID}}$	$\leftarrow \mathsf{SecKey}(SK, \overline{ID})$
	K	$\leftarrow \overline{\mathsf{IBKEM}}.\mathsf{Decaps}(SK_{\overline{ID}}, \langle \overline{C_1}, \overline{C_2} \rangle)$
	M	$\leftarrow \overline{C_M} \oplus \overline{K}$

[3] In fact, the authors define this construction as second in their paper, but it better fits the presentation in this thesis, to do it the other way round.

If the above criteria of ACIK-partitioned IBKEM and strong security are met, then this transformation results in IND-CCA$_2$ secure (Def. 2.2 on p.16) PKE.

Theorem 3.16 ([ACIK07, Theorem 7])**.** *If* $\overline{\mathsf{IBKEM}}$ *is ACIK-partitioned and* ϵ*-strong-IND-sID-CPA secure, then:*

$$\mathsf{PKE} := ACIK_1(\overline{\mathsf{IBKEM}}) \text{ is } \epsilon\text{-IND-CCA}_2 \text{ secure.}$$

This essentially reflects the transformation from Waters-IBKEM to BMW-PKE, Waters-IBKEM fits all preconditions and the outcome of $ACIK_1$(Waters-IBKEM) is basically BMW-PKE.

The second ACIK transformation is less demanding, as it does not need strong IND-aID-CPA security but can be applied to any (*non-strong*) IND-*sID*-CPA secure IBKEM. The downside is that it introduces an additional computational overhead and enlarges ciphertexts by using a *chameleon hash function*.

A chameleon hash function is roughly a hash function that allows someone to obtain collisions if a secret trapdoor t is known, but remains collision resistant for anyone else. It consists of three algorithms CHM.KeyGen, which generates a hash key k and a trapdoor t, a randomized hash algorithm CHM.H, that takes input x and randomness r and produces a hash value y, and finally a trapdoor algorithm CHM.Trap, which on input t, x, r, x' finds different randomness r' such that $\mathsf{H}(x, r) = \mathsf{H}(x', r')$. A chameleon hash function is computationally more demanding than a target collision resistant hash function (§2.4.1 on p.22) but much more efficient than a strongly existentially unforgeable signature scheme (§2.4.2 on p.23). For more details see [ACIK07, §2.6].

With this chameleon hash function the second ACIK transformation can be defined. Again, the construction is presented in a simplified version resulting in PKE.

Construction 3.17 ([ACIK07, §4])**.** *The second ACIK transformation* $\mathsf{PKE} := ACIK_2(\overline{\mathsf{IBKEM}}, \overline{\mathsf{CHM}})$ *for an ACIK-partitioned* $\overline{\mathsf{TBKEM}}$ *and a chameleon hash function* $\overline{\mathsf{CHM}}$ *is defined as follows:*

$$
\begin{array}{lll}
\mathsf{PKE.KeyGen}(1^\lambda) & \overline{PK}, \overline{SK} & \leftarrow \overline{\mathsf{IBKEM.Setup}}(1^\lambda); \\
 & k, t & \leftarrow \overline{\mathsf{CHM.KeyGen}}(1^\lambda); \\
 & PK, SK & \leftarrow \langle \overline{PK}, k \rangle, \overline{SK} \\
\mathsf{PKE.Encrypt}(PK, M) & \overline{K}, \overline{C_1}, \overline{\sigma} & \leftarrow \overline{\mathsf{IBKEM.Encaps}_1}(PK) \\
 & \overline{C_M} & \leftarrow M \oplus \overline{K} \\
 & \overline{r} & \leftarrow Rand \\
 & \overline{ID} & \leftarrow \mathsf{CHM.H}(\overline{C_M} \mid\mid \overline{C_1}, \overline{r}) \\
 & PK_{\overline{ID}} & \leftarrow \mathsf{PubKey}(PK, \overline{ID}) \\
 & \overline{C_2} & \leftarrow \overline{\mathsf{IBKEM.Encaps}_2}(PK_{\overline{ID}}, \overline{\sigma}) \\
 & C & \leftarrow \langle \overline{C_M}, \overline{C_1}, \overline{C_2}, \overline{r} \rangle \\
\mathsf{PKE.Decaps}(SK, C) & \langle \overline{C_M}, \overline{C_1}, \overline{C_2}, \overline{r} \rangle & \leftarrow C \\
 & \overline{ID} & \leftarrow \mathsf{CHM.H}(\overline{C_M} \mid\mid \overline{C_1}, \overline{r}) \\
 & SK_{\overline{ID}} & \leftarrow \mathsf{SecKey}(SK, \overline{ID}) \\
 & K & \leftarrow \overline{\mathsf{IBKEM.Decaps}}(SK_{\overline{ID}}, \langle \overline{C_1}, \overline{C_2} \rangle) \\
 & M & \leftarrow \overline{C_M} \oplus \overline{K}
\end{array}
$$

Compared to the first ACIK transformation ciphertexts are enlarged by the size of $\overline{r}$. Furthermore, for encryption and decryption an additional $\overline{\mathsf{CHM}}.\mathsf{H}$ computation has to be done. Note, that the trapdoor t and the $\overline{\mathsf{CHM}}.\mathsf{Trap}$ algorithm is never used in the scheme. It is only needed for the security proof.

Theorem 3.18 ([ACIK07, Theorem 3])**.** *If $\overline{\mathsf{IBKEM}}$ is ACIK-partitioned and ϵ-IND-sID-CPA secure and $\overline{\mathsf{CHM}}$ is $\hat{\epsilon}$-RPC-secure (this form of security has not been defined in this paper), then:*

$$\mathsf{PKE} := ACIK_2(\overline{\mathsf{IBKEM}}, \overline{\mathsf{CHM}}) \text{ is } (\epsilon + \hat{\epsilon})\text{-IND-CCA}_2 \text{ secure.}$$

For proving security of both ACIK transformations it is essential that either a strong form of *adaptive-identity security* is present ($ACIK_1$) or the missing security is compensated by a *chameleon hash function* ($ACIK_2$). Abe et al. [ACIK07, §5.4] do not expect that one can do without both (and get KEM instead of PKE):

> "Similar to [BMW05a, Kil06] one could try to directly build a key-encapsulation mechanism [CS04] (KEM) out of an sID partitioned IBKEM using a target-collision resistant hash function H as follows. To encapsulate, first compute key and the first part of the ciphertext as $\langle K, C_1, \sigma \rangle \leftarrow \mathsf{IBKEM.Encaps}_1(PK)$. Then, the second part of the ciphertext is computed as $C_2 \leftarrow \mathsf{IBKEM.Encaps}_2(PK_{ID}, \sigma)$, where $ID \leftarrow H(C_1)$ is used to tie the two ciphertexts together. Whereas syntactically this is correct, without assuming further algebraic structure it seems hard to relate security of the KEM to the security of the IBKEM. This is since a simulator for the KEM security experiment interacting with the sID IBKEM challenger has to commit to a target identity before seeing the public key. But in the scheme the target identity depends on the first part of the target ciphertext and hence a stronger (and less natural) security requirement to the IBKEM scheme is needed for a general security reduction. In general, proving that the IBKEM satisfies such a stronger security property seems not easier than providing a direct proof for the transformed KEM."

(The algorithmic parts have been slightly adjusted to fit the notation used here.)

4 Efficient CCA_2 Security from Special Tag-/Identity-Based KEM

Boyen et al. [BMW05a] give two direct constructions of CCA_2 secure public-key schemes that are reminiscent of CPA secure identity-based schemes. The IND-CCA_2 secure BMW-PKE is obtained from the IND-aID-CPA secure Waters-IBKEM and the IND-CCA_2 secure BMW-KEM is derived from the IND-sID-CPA secure BB1-IBKEM. The distinguishing feature of Waters-IBKEM and BB1-IBKEM is that the former has aID security whereas the latter only has sID security.

Abe et al. [ACIK07] define *partitioned* IBKEMs and give two constructions of IND-CCA_2 secure PKE that either use a *strong* version of IND-aID-CPA security or need additional cryptographic primitives: *Chameleon hash functions* which are more demanding than standard hash functions. These transformations give a good explanation of the relationship between Waters-IBKEM and BMW-PKE.

In this chapter, analogue transformations are sought that explain the relationship between BB1-IBKEM and BMW-KEM. The transformations should give CCA_2 secure KEM instead of PKE, but therefore neither need aID security nor need additional strong cryptographic primitives—but a standard hash function is ok.

In Section 4.1, the construction of BMW-KEM is examined in an intuitive way. This will result in several properties of BB1-IBKEM that allow this construction. These properties together with a transformation rule will be formalized in Section 4.2 as the "Core Transformation". This is the analogue to the first ACIK transformation. Analogue to the second ACIK transformation, in Section 4.3, the need of the stronger security variant will be avoided by introducing the "Hash-Based Transformation". Finally, in Section 4.4, the applicability of both transformations as well as their similarities and differences to the ACIK transformations will be reviewed.

4.1 Inspiration from BMW-KEM

Before starting with abstract definitions, it is advisable to have a closer look at BB1-IBKEM and BMW-KEM and to work out the basic ideas. In Figure 4.1, the IND-sID-CPA secure (Def. 2.4 on p.17) BB1-IBKEM and the IND-CCA_2 secure (Def. 2.3 on p.16) BMW-KEM are recapitulated. As already mentioned before, both differ slightly in the Encaps algorithm and a little bit more in the Decaps algorithm. What changes are relevant for the security gain?

As an intermediate step a TBKEM (§2.1.4 on p.13) version of BB1-IBKEM is given. BB1-TBKEM may be seen as a hybrid of BB1-IBKEM and BMW-KEM. It leaves the encapsulation essentially unchanged, but incorporates already the changes in decapsulation. The Decaps algorithm first checks whether C is a valid ciphertext with respect to tag (compare equation 3 of Fact 3.3 on p.32), and then restores K directly using SK. This takes a cheap exponentiation and only one expensive computation of the bilinear map e. It is more efficient than the generic IBKEM-to-TBKEM transformation (§2.3 on p.19), which would interpret tag as identity, derive a secret key SK_{ID} for that identity from SK, and restore K using SK_{ID}. BB1-TBKEM obtains IND-sTag-wCCA$_2$ security (Def. 2.5 on p.18):

Theorem 4.1. *If the BDDH problem is ϵ-hard in the groups generated by BGGen (Def. 2.10 on p.25) and the H is $\hat{\epsilon}$-TCR secure (Def. 2.7 on p.22), then:*

$$\textit{BB1-TBKEM is } (\epsilon + \hat{\epsilon}/2)\textit{-IND-sTag-wCCA}_2 \textit{ secure.}$$

Proof Sketch. The proof is basically the same as for BB1-IBKEM (Thm. 3.2 on p.30 and Cor. 3.6 on p.36), except that the attacker is given a wDecaps oracle, which decapsulates ciphertexts for any $tag \neq tag^\star$, instead of a SecKey oracle, which returns the secret key for any $ID \neq ID^\star$.

As in the proof for hfBB1-IBKEM (Thm. 3.2 on p.30), the simulator $\mathcal{B}$ postpones the generation of PK until after $\mathcal{A}$ has chosen $tag^\star$. Thus, it can base the choice of g_3 on $tag^\star$ in a way that enables it to correctly construct the challenge ciphertext and to simulate the wDecaps oracle. For oracle simulation, valid ciphertexts can be answered using techniques of the SecKey oracle from the security proof of hfBB1-IBKEM and invalid ciphertexts can be detected via equation 4 of Fact 3.3 (p.32) and is answered with $\perp$. □

Although the naming of IND-sID-CPA versus IND-sTag-wCCA$_2$ might suggest it, there is no improvement of security from step BB1-IBKEM to BB1-TBKEM. The SecKey oracle gives the attacker the same power as the wDecaps oracle (in fact even more), as the secret keys can be used by the attacker to decapsulate ciphertexts for any $ID \neq ID^\star$.

Consequently, the subtle change of Encaps between BB1-TBKEM and BMW-KEM must be the key part of the transformation. In the introduction (§1.3 on p.4) some intuition for creating IND-CCA$_2$ secure PKE from IND-sID-CPA secure IBE has been given. Indeed, BMW-KEM may be seen as a somewhat modified implementation of the paradigms given there (on p.5). They are repeated in the following (but formulated in the TBKEM setting).

- Essential fact: Decapsulations using $tag' \neq tag$ will not help with a ciphertext encapsulated using tag. (This is modelled by the wDecaps oracle.)
- Basic idea: The sender uses a different tag for every encapsulation. The tag has to be prepended to the ciphertext, i. e., send $\langle tag, C \rangle$.

common $\mathsf{Setup}/\mathsf{KeyGen}(1^\lambda)$

$\langle p, \mathbb{G}, \mathbb{G}_T, e, g\rangle \leftarrow \mathsf{BGGen}(1^\lambda)$
$a, b, c \leftarrow \mathbb{Z}_p$
$g_1, g_2, g_3 \leftarrow g^a, g^b, g^c$

$Z \leftarrow e(g_1, g_2)$
$k \leftarrow \{0,1\}^{\ell(\lambda)}$
$PK \leftarrow \langle p, \mathbb{G}, \mathbb{G}_T, e, g, g_1, g_2, g_3, Z, k\rangle$
$SK \leftarrow g_2^a$ resp. $SK \leftarrow \langle g_2^a, a, c\rangle$

(1) $\mathsf{PubKey}(PK, ID)$

$PK_{ID} \leftarrow g_1^{H_k(ID)} \cdot g_3$

$\mathsf{Encaps}(PK_{ID})$

$s \leftarrow \mathbb{Z}_p$
$C \leftarrow \langle g^s, PK_{ID}^s\rangle$
$= \langle g^s, (g_1^{H_k(ID)} \cdot g_3)^s\rangle$
$K \leftarrow Z^s$

$\mathsf{SecKey}(SK, ID)$

$r \leftarrow \mathbb{Z}_p$
$SK_{ID} \leftarrow \langle g^r, SK \cdot PK_{ID}^r\rangle$

$\mathsf{Decaps}(SK_{ID}, C)$

$\langle S_1, S_2\rangle \leftarrow SK_{ID}$
$\langle C_1, C_2\rangle \leftarrow C$
$K \leftarrow e(S_2, C_1)/e(C_2, S_1)$

(2) $\mathsf{Encaps}(PK, tag)$

$s \leftarrow \mathbb{Z}_p$
$C \leftarrow \left\langle g^s, (g_1^{H_k(tag)} \cdot g_3)^s\right\rangle$
$K \leftarrow Z^s$

$\mathsf{Decaps}(SK, tag, C)$

$\langle \overline{SK}, a, c\rangle \leftarrow SK$
$\langle C_1, C_2\rangle \leftarrow C$
check $C_1^{a\cdot H_k(tag)+c} \stackrel{?}{=} C_2$
if true $K \leftarrow e(\overline{SK}, C_1)$, else $K \leftarrow \perp$

(3) $\mathsf{Encaps}(PK)$

$s \leftarrow \mathbb{Z}_p$
$C \leftarrow \langle C_1, C_2\rangle$
$= \left\langle g^s, (g_1^{H_k(C_1)} \cdot g_3)^s\right\rangle$
$K \leftarrow Z^s$

$\mathsf{Decaps}(SK, C)$

$\langle \overline{SK}, a, c\rangle \leftarrow SK$
$\langle C_1, C_2\rangle \leftarrow C$
check $C_1^{a\cdot H_k(C_1)+c} \stackrel{?}{=} C_2$
if true $K \leftarrow e(\overline{SK}, C_1)$, else $K \leftarrow \perp$

Figure 4.1: A comparison of (1) BB1-IBKEM and (3) BMW-KEM plus an intermediate step (2) BB1-TBKEM. The step (1) → (2) explains the increased efficiency of BMW-KEM and the step (2) → (3) gives the gain in security.

- Paradigm 1: To rule out the decapsulation oracle *before* interception of $\langle tag, C\rangle$, the selection of tag should be unpredictable to an attacker.
- Paradigm 2: To rule out the decapsulation oracle *after* interception of $\langle tag, C\rangle$, it should be infeasible (or impossible) for an attacker to find a modification $\langle tag, C'\rangle$, where C' is *valid* under PK and tag.

In BMW-KEM, the randomly chosen tag in the computation of C_2 is replaced by the first ciphertext part C_1. This is only possible, as in BB1-TBKEM *the ciphertext is dividable into two parts, where the first one does not depend on the tag*. Replacing tag by C_1 can be interpreted as a modification of the basic idea as follows. Instead of choosing tag separately and prepending it to C, it is encoded as part of C. If C_1 is interpreted as this encoding, it is written as "tag" in the following, e. g., $C = \langle C_1, C_2\rangle = \langle$"$tag$", $C_2\rangle$.

With paradigm 1, "tag" has to be unpredictable for the attacker. This is fulfilled, as C_1 ="tag" is uniformly distributed in $\mathbb{G}$ and $|\mathbb{G}| = p$ is large. The requirement for BB1-TBKEM is that *the choice of the first ciphertext part should be unpredictable.*

Paradigm 2 is satisfied by the following observation. In BMW-KEM, C_2 is uniquely determined by PK and C_1. This means that for fixed PK and C_1, there is only a single C_2 such that $C = \langle C_1, C_2\rangle$ is a valid ciphertext. Consequently, it is impossible for an attacker given $\langle$"tag",$C_2\rangle = \langle C_1, C_2\rangle$ to find a C_2' such that $\langle$"tag",$C_2'\rangle = \langle C_1, C_2'\rangle$ is valid. The structural requirement for BB1-TBKEM which permits this argument is that *the second part of the ciphertext is uniquely determined by the first one, the public key and the tag.* The above observations plus a fourth more technical property of BB1-TBKEM will make up the definition of *partitioned* TBKEM.

Similar to the work of [ACIK07], BB1-TBKEM also fulfills a *strong* version of IND-sTag-wCCA$_2$ security, which is essential for the security proof of BMW-KEM. In Theorem 4.1 (which originates in Thm. 3.2 on p.30), the high-level strategy for the simulator $\mathcal{B}$ was as follows: First, it lets the attacker $\mathcal{A}_1$ choose $tag^\star$, and then it determines PK, $C^\star = \langle C_1^\star, C_2^\star\rangle$ and finally $K_b^\star$. This order seemed necessary, as parts of PK—namely g_3—were chosen dependently on $tag^\star$. Looking more carefully at the proof, one can see that there is much more freedom in the order of computation. Particularly, $\mathcal{B}$ does not base the computation of $C_1^\star$ on $tag^\star$—neither directly nor indirectly—, and hence it can be determined before $\mathcal{A}$ chooses $tag^\star$.

This is very important in order to use the proof techniques from BB1-TBKEM for BMW-KEM. Assume that in the proof of BB1-TBKEM, $\mathcal{B}$ would have to know $tag^\star$ in order to compute $C_1^\star$. This would result in a cyclic dependency when carrying over the proof to BMW-KEM: To compute $C_1^\star$ the value of "$tag^\star$" would be needed—but "$tag^\star$" is nothing else than $C_1^\star$. As result, it is crucial for the proof of BMW-KEM that during the proof of IND-sTag-weakCCA$_2$ security for BB1-TBKEM, *the simulator $\mathcal{B}$ may determine $C_1^\star$ before $tag^\star$ is chosen by $\mathcal{A}_1$, and consequently the computation of $C_1^\star$ is not based on $tag^\star$.*

All observations from this section may be summarized as follows. IND-CCA$_2$ secure BMW-KEM is constructed from IND-sID-CPA secure BB1-IBKEM in two steps. First, BB1-TBKEM is obtained from BB1-IBKEM in a non-generic way. It optimizes the efficiency of decapsulation but does not have any impact on security. Then, a seemingly simple modification of the encapsulation algorithm—which may be seen as very efficient implementation of the basic idea from the introduction—raises security to IND-CCA$_2$. Five properties of BB1-TBKEM, combined as *partitioned* TBKEM and *strong* IND-sTag-wCCA$_2$ security, are essential to render this transformation possible.

4.2 The Core Transformation

In this section, the common core of all following transformations is formalized. The result is a novel construction rule that explains how BMW-KEM is obtained

from BB1-TBKEM (Figure 4.1 on p.45). First, *partitioned TBKEMs* and their *transformation into KEMs* will be defined. Then, the notion of *strong* IND-sTag-wCCA$_2$ security will be defined, and finally *IND-CCA$_2$ security* will be proven for KEMs that are based on strong IND-sTag-wCCA$_2$ secure partitioned TBKEMs. As a side effect, this gives a very cheap way to obtain IND-CCA$_1$ security.

Definition 4.2. *A TBKEM (§2.1.4 on p.13) is* partitioned *if it additionally satisfies the following properties:*

- Split of ciphertext*: The ciphertext C can be split into two parts, $C = \langle C_1, C_2 \rangle$, where the first part C_1 only depends on the public key PK and not the tag. This allows to rewrite the* Encaps *algorithm as follows:*

  Encaps$_1(PK) \rightsquigarrow C_1, \sigma$***;*** *which outputs the first part of the ciphertext C_1 and some important values σ from its internal state. E. g., σ could be all randomness produced by the internal coin tosses. It is meant only as input for the second stage and has to be deleted afterwards.*

  Encaps$_2(PK, tag, \sigma) \rightsquigarrow K, C_2$***; (deterministic)*** *which returns session key K and the second ciphertext part C_2.*

- Unpredictability of C_1*: There is a negligible bound $\epsilon_1(\lambda)$ such that*

$$\max_{PK} \max_{C_1} \quad Pr[C_1^\star \leftarrow \textsf{Encaps}_1(PK) : C_1^\star = C_1] \quad =: \quad \epsilon_1(\lambda) \in negl(\lambda)$$

- Uniqueness of C_2*: The second part C_2 is uniquely determined by the public key PK, the tag, and the first part C_1.*

- Simulatable rejection*: There is an efficient algorithm* Reject(PK, tag, C) *that has the same output distribution as* Decaps(SK, tag, C) *for any invalid ciphertext $C \notin \mathcal{C}(PK, tag)$. There is no requirement for the output distribution of* Reject *on any input with $C \in \mathcal{C}(PK, tag)$!*

The first three properties have been identified in the section before. The last one is needed in the security proof where the "decapsulation" of some invalid ciphertexts has to be simulated without knowing the secret key.

The uniqueness property implies that there exists a function from PK, tag, C_1 to C_2. So one might wonder why Encaps$_2$ is given σ instead of C_1 as input. Examining this question on the concrete example BB1-TBKEM gives a first intuition: Encaps$_1$ chooses a random s and computes C_1 as g^s. Now Encaps$_2$ shall compute $K \leftarrow e(g_1, g_2)^s$ and $C_2 \leftarrow (g_1^{tag} g_3)^s$. Doing this given only g, g_1, g_2, g_3, tag and g^s would require solving[1] the BCDH problem in $\mathbb{G}_T$ and the CDH problem in $\mathbb{G}$ (§2.5 on p.23), but with s as additional input this is easy.

[1] The reduction is as follows (sketch): given a CDH instance $\langle g, x = g^s, y = g^t \rangle$ one would choose random g_1, set $g_3 \leftarrow y \cdot g_1^{-tag}$ and $C_1 \leftarrow x$. Having an algorithm that computes C_2 from these values would solve the CDH problem, since $C_2 = (g_1^{tag} g_3)^s = y^s = g^{st}$. Similar reasoning works for K and the BCDH problem.

Following a more abstract argument, computing K, C_2 from PK, tag, C_1 generally should be a computational hard problem. First assume that it was computationally feasible to compute K from these values. This would clearly break security, since an eavesdropper knowing PK, tag, C_1 could then directly compute K. Now assume that C_2 could be efficiently computed from PK, tag, C_1. This would not necessarily break security, but then C_2 would at least be a redundant value, since the receiver could compute it on his own.

Due to the split property, a KEM (§2.1.2 on p.11) construction can be defined.

Construction 4.3. *The core transformation* $\mathsf{KEM} := Core(\overline{\mathsf{TBKEM}})$ *for a partitioned* $\overline{\mathsf{TBKEM}}$ *is defined as follows:*

$$
\begin{array}{lll}
\mathsf{KEM.KeyGen}(1^\lambda) & PK, SK & \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{KeyGen}(1^\lambda); \\
\mathsf{KEM.Encaps}(PK) & \overline{C_1}, \overline{\sigma} & \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Encaps}_1(PK) \\
 & \overline{tag} & \leftarrow \overline{C_1} \\
 & K, \overline{C_2} & \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Encaps}_2(PK, \overline{tag}, \overline{\sigma}) \\
 & C & \leftarrow \langle \overline{C_1}, \overline{C_2} \rangle \\
\mathsf{KEM.Decaps}(SK, C) & \langle \overline{C_1}, \overline{C_2} \rangle & \leftarrow C \\
 & \overline{tag} & \leftarrow \overline{C_1} \\
 & K & \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Decaps}(SK, \overline{tag}, C)
\end{array}
$$

As seen in the section before, to make a security proof for KEM out of the proof for $\overline{\mathsf{TBKEM}}$, the latter one must have a special structure: The simulator has to be able to determine $C_1^\star$ before $tag^\star$ is chosen by $\mathcal{A}_1$. This is enforced by modifying the IND-sTag-wCCA$_2$ experiment (Def. 2.5 on p.18) such that the simulator is required to give $C_1^\star$ as input for $\mathcal{A}_1$.

Definition 4.4. *For a partitioned TBKEM, the* strong IND-sTag-wCCA$_2$ *experiment is defined as follows:*

$$
\begin{array}{l}
Exp^{\mathsf{TBKEM},\mathcal{A}}_{\textit{strong IND-sTag-wCCA}_2}(\lambda)\textit{:} \\
\quad PK, SK \leftarrow \mathsf{KeyGen}(1^\lambda) \\
\quad C_1^\star, \sigma^\star \leftarrow \mathsf{Encaps}_1(PK) \\
\quad tag^\star \leftarrow \mathcal{A}_1(C_1^\star) \\
\quad b \leftarrow \{0,1\} \\
\quad K_0^\star, C_2^\star \leftarrow \mathsf{Encaps}_2(PK, tag^\star, \sigma^\star) \\
\quad K_1^\star \leftarrow \mathcal{K}(PK, tag^\star) \\
\quad b' \leftarrow \mathcal{A}_2^{\mathsf{wDecaps}}(PK, K_b^\star, C_2^\star) \\
\quad \textit{if } b = b', \textit{ then output win}
\end{array}
$$

On query $\mathsf{wDecaps}$*-oracle*(tag, C)*, for* $tag \neq tag^\star$ *and an arbitrary (valid or invalid) ciphertext* C*, the decapsulation oracle returns* $M \leftarrow \mathsf{Decaps}(SK, tag, C)$*.*

Note that the only difference to the standard IND-sTag-wCCA$_2$ experiment is the input value for $\mathcal{A}_1$. Given an IND-sTag-wCCA$_2$ TBKEM, most likely there is no need for a dedicated proof of strong security. Instead, the proof of IND-sTag-wCCA$_2$ security can be analyzed for whether the simulator may determine $C_1^\star$

without knowing $tag^\star$. Clearly, for BB1-TBKEM this is the case, as $\mathcal{B}$ simply uses its own input value z as $C_1^\star$ (see Thm. 4.1 on p.44 and Thm. 3.2 on p.30). Now, all prerequisites for the proof of IND-CCA$_2$ security (Def. 2.3 on p.16) for $Core(\overline{\mathsf{TBKEM}})$ are met.

Theorem 4.5. *If $\overline{\mathsf{TBKEM}}$ (partitioned) is ϵ-strong-IND-sTag-wCCA$_2$ secure, then:*

$$\mathsf{KEM} := Core(\overline{\mathsf{TBKEM}}) \text{ is } (\epsilon + q_{\mathsf{Decaps}} \cdot \epsilon_1/2)\text{-IND-CCA}_2 \text{ secure,}$$

where ϵ_1 is the upper bound for the C_1-unpredictability Definition 4.2, and where q_{Decaps} is the number of the attacker's oracle queries.

Proof. Let $\mathcal{A}$ be an attacker of KEM in the IND-CCA$_2$ experiment. It has to be shown that its advantage is smaller than $\epsilon + q_{\mathsf{Decaps}} \cdot \epsilon_1/2$. To bound this value, this IND-CCA$_2$ experiment is modified slightly in a sequence of three games, such that in the last game, $\mathcal{A}$ can be turned into an attacker $\mathcal{B}$ of $\overline{\mathsf{TBKEM}}$ (Definition 4.4). The idea behind these games follows the two paradigms identified in the introduction and restated in the section before (§4.1 on p.44).

Game 1 This is the IND-CCA$_2$ game for KEM.

Game 2 The execution of $\mathcal{A}_1$ is delayed a bit and its KEM.Decaps oracle aborts[2] if a ciphertext $\langle \overline{C_1^\star}, \ldots \rangle$ is queried, i. e., that starts as the challenge ciphertext.

Game 3 At the KEM.Decaps oracle in phase 2, any query for a ciphertext $\langle \overline{C_1^\star}, \overline{C_2} \rangle$ with $\overline{C_1} \neq \overline{C_1^\star}$, i. e., starting as but being different from the challenge, is rejected by setting $\overline{tag} \leftarrow \overline{C_1^\star}$ and then returning $\overline{\mathsf{TBKEM}}.\mathsf{Reject}(PK, \overline{tag}, C)$.

$\mathrm{Exp}^{\mathsf{KEM},\mathcal{A}}_{Game\ 2/3}(\lambda)$:

PK, SK	$\leftarrow \mathsf{KEM.KeyGen}(1^\lambda)$	
	~~$\mathcal{A}_1^{\mathsf{KEM.Decaps}}(PK)$~~	
b	$\leftarrow \{0,1\}$	
$K_0^\star, C^\star$	$\leftarrow \mathsf{KEM.Encaps}(PK)$	(recall that $C^\star = \langle \overline{C_1^\star}, \overline{C_2^\star} \rangle$)
$K_1^\star$	$\leftarrow \mathcal{K}(\lambda)$	
	$\mathcal{A}_1^{\mathsf{KEM.Decaps}}(PK)$	$\left(\boxed{\text{abort } \langle \overline{C_1^\star}, \ldots \rangle}_{Game\ 2}\right)$
b'	$\leftarrow \mathcal{A}_2^{\mathsf{KEM.Decaps}}(K_b^\star, C^\star)$	$\left(\text{abort } \langle \overline{C_1^\star}, \overline{C_2^\star} \rangle, \boxed{\text{reject } \langle \overline{C_1^\star}, \overline{C_2} \rangle}_{Game\ 3}\right)$

if $b = b'$, then output *win*

Proof Overview: The attacker $\mathcal{B}$ of $\overline{\mathsf{TBKEM}}$ has to simulate a full KEM.Decaps oracle for $\mathcal{A}$. This will be no problem for queries where the decapsulation of a ciphertext $\langle \overline{C_1}, \overline{C_2} \rangle$ with $\overline{C_1} \neq \overline{C_1^\star}$ (recall the convention from the previous section to think of "tag" for $\overline{C_1}$, then the inequality reads as "$\overline{tag}$"≠"$\overline{tag^\star}$"). $\mathcal{B}$ can do this easily with the help of the $\overline{\mathsf{TBKEM}}.\mathsf{wDecaps}$ oracle. The other queries, i. e., where

[2] As before "abort" means, that the execution of $\mathcal{A}$ is stopped and $b' \leftarrow \{0,1\}$ is chosen randomly.

$\overline{C_1} = \overline{C_1^\star}$, are called *problematic*, as $\mathcal{B}$ may not use its $\overline{\mathsf{TBKEM}}.\mathsf{wDecaps}$ oracle in these cases. Those problematic queries can be correctly answered by $\mathcal{B}$ in

- neither phase of *Game 1*,
- only phase 1 of *Game 2*,
- both phases of *Game 3*.

↓ paradigm 1

↓ paradigm 2

The change in *Game 2* reflects the exploitation of paradigm 1. As $\mathcal{B}$ may abort on any problematic query, it is able to simulate the full (modified) $\mathsf{KEM}.\mathsf{Decaps}$ oracle in phase 1. As $\mathcal{A}$ does not know $\overline{C_1^\star}$ at that time, it may ask such a problematic query only by accident. This can be bounded using the unpredictability of $\overline{C_1^\star}$ as:

$$|Pr^{\mathcal{A}}_{Game\ 1}[b = b'] - Pr^{\mathcal{A}}_{Game\ 2}[b = b']| \leq q_{\mathsf{Decaps}}(\lambda) \cdot \epsilon_1(\lambda)/2$$

Game 3 mainly takes advantage of the uniqueness of $\overline{C_2^\star}$ (reflecting paradigm 2). $\mathcal{B}$ may either abort or reject on any problematic query, which solves its problems with simulating the $\mathsf{KEM}.\mathsf{Decaps}$ oracle in phase 2. The differences from *Game 2* to *Game 3* will turn out as purely "cosmetic" and are not at all recognized by $\mathcal{A}$:

$$|Pr^{\mathcal{A}}_{Game\ 2}[b = b'] - Pr^{\mathcal{A}}_{Game\ 3}[b = b']| = 0$$

Finally, *Game 3* can be perfectly simulated by attacker $\mathcal{B}$ of $\overline{\mathsf{TBKEM}}$, leading to:

$$|Pr^{\mathcal{A}}_{Game\ 3}[b = b'] - 1/2| = \mathrm{Adv}^{\overline{\mathsf{TBKEM}},\mathcal{B}}_{\text{strong IND-sTag-wCCA}_2}(\lambda) \leq \epsilon(\lambda)$$

Collecting all these equations bounds $\mathcal{A}$'s advantage (as explained in §2.6 on p.25).

***Game 1* to *Game 2*:** The shift of $\mathcal{A}_1$ makes no difference at all to the result of the experiment, but for the abort condition to be well-defined it is necessary that $C^\star$ is determined prior to the execution of $\mathcal{A}_1$. The other change, the newly introduced abort, makes a real difference. But with the help of the fundamental lemma (Lem. 2.11 on p.27) and a failure event F it will be shown that this difference is small. Recall from Theorem 3.5 (p.32) how the applicability of that lemma can be proven. This will not be redone here. Instead, only a suitable failure event F is defined and $Pr^{\mathcal{A}}_{Game\ 1}[F]$ is bounded, which gives:

$$|Pr^{\mathcal{A}}_{Game\ 1}[b = b'] - Pr^{\mathcal{A}}_{Game\ 2}[b = b']| \leq Pr^{\mathcal{A}}_{Game\ 1}[F]/2$$

Let the challenge ciphertext be written as $C^\star = \langle \overline{C_1^\star}, \overline{C_2^\star} \rangle$. Let Q_1 be the first halves of all ciphertexts queried at the $\mathsf{KEM}.\mathsf{Decaps}$ oracle by $\mathcal{A}_1$. Both games are the same unless $\mathcal{A}_1$ makes a problematic query, i. e., the failure event F is "$C_1^\star \in Q_1$". The probability for this event to occur in *Game 1* will be bounded as follows.

Consider a conditioned probability space that fixes all randomness in *Game 1* except that one used for computing the challenge ciphertext $C^\star$. In particular, $C^\star = \langle \overline{C_1^\star}, \overline{C_2^\star} \rangle$ may still vary, but PK, SK, the randomness in the Decaps oracle,

and the randomness in $\mathcal{A}$ are fixed. The important question is whether Q_1 is fixed, too. Assuming this for a short while allows the following bound for every conditioned space (and so for the whole *Game 1*):

$$\begin{aligned} Pr[F] &= Pr[\overline{C_1^\star} \in Q_1] \\ &= \sum_{\overline{C_1} \in Q_1} Pr[\overline{C_1^\star} = \overline{C_1}] \qquad \Big(\swarrow \ \overline{C_1^\star} \text{ is determined by } \mathsf{Encaps}_1(PK) \Big) \\ &= \sum_{\overline{C_1} \in Q_1} \underbrace{Pr[\overline{C_1^\star} \leftarrow \mathsf{Encaps}_1(PK) \ : \ \overline{C_1^\star} = \overline{C_1}]}_{\leq \epsilon_1(\lambda) \quad \text{see Definition 4.2 on p.47}} \\ &\leq |Q_1| \cdot \epsilon_{cp}(\lambda) \\ &\leq q_{\mathsf{Decaps}}(\lambda) \cdot \epsilon_{cp}(\lambda) \end{aligned}$$

It still has to be shown that Q_1 is fixed in all conditioned spaces. The first Decaps query C of $\mathcal{A}_1$ is a function of PK and $\mathcal{A}$'s randomness. Since both are fixed, C will always be the same over all possibilities of $C^\star$. The answer K of the Decaps oracle is a function of SK, the randomness of the oracle and C. Since all are fixed, the same holds for K. The next query of $\mathcal{A}_1$ is a function of PK, $\mathcal{A}$'s randomness and the first answer K and is thus fixed, too. By induction, the set of all ciphertexts queried by $\mathcal{A}_1$ is fixed and thus Q_1 has to be fixed, too.

In the following, a reasoning in this manner will be subsumed as "$\mathcal{A}_1$'s view is independent of $C^\star$ and so is Q_1". It is important to note the subtleties of such an argument. For *Game 2* it would be *wrong*! $\mathcal{A}$'s view is not independent of $C^\star$. The reason is that the answer of the modified Decaps oracle also takes $C^\star$ into account in order to decide whether it should return the key or abort. Nevertheless, in *Game 1* this argument is valid and the above bound holds.

***Game 2* to *Game 3*:** In both games, the only valid ciphertext starting with $\overline{C_1^\star}$ is $\langle \overline{C_1^\star}, \overline{C_2^\star} \rangle$ due to the unique split property and the fact that in KEM $\overline{C_1^\star}$ is used as "tag". So $\overline{\mathsf{TBKEM}}$.Reject perfectly simulates decapsulation for all other (invalid) ciphertexts starting with $\overline{C_1^\star}$ and nothing changes from $\mathcal{A}$'s point of view.

***Game 3* to $\overline{\mathsf{TBKEM}}$:** As promised, this game can easily be simulated by $\mathcal{B}$, which is an attacker in the strong IND-sTag-wCCA$_2$ experiment for $\overline{\mathsf{TBKEM}}$ (Def. 4.4 on p.48). $\mathcal{B}$ works as follows:

$\mathcal{B}_1(\overline{C_1^\star})$**:** output $\overline{tag^\star} \leftarrow \overline{C_1^\star}$ for the own attack on $\overline{\mathsf{TBKEM}}$

$\mathcal{B}_2^{\overline{\mathsf{TBKEM}}.\mathsf{wDecaps}}(PK, K_b^\star, \overline{C_2^\star})$**:**

- set $C^\star \leftarrow \langle \overline{C_1^\star}, \overline{C_2^\star} \rangle$
- first run $\mathcal{A}_1^{\mathsf{KEM.Decaps}}(PK)$
- and then $\mathcal{A}_2^{\mathsf{KEM.Decaps}}(K_b^\star, C^\star)$
- use $\mathcal{A}_2$'s output b' as own guess

$\overline{\mathsf{KEM}}$.Decaps-oracle(C) **for** $\mathcal{A}_{1/2}$**:**

- parse $\langle \overline{C_1}, \overline{C_2} \rangle \leftarrow C$
- set $\overline{tag} \leftarrow \overline{C_1}$
- if $\overline{C_1} = \overline{C_1^\star} \wedge \overline{C_2} = \overline{C_2^\star}$: abort simulation and output random b'
- if $\overline{C_1} = \overline{C_1^\star} \wedge \overline{C_2} \neq \overline{C_2^\star}$:
  - phase 1: abort simulation and output random b'
  - phase 2: return $K \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Reject}(PK, \overline{tag}, C)$
- if $\overline{C_1} \neq \overline{C_1^\star}$: return $K \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{wDecaps\text{-}oracle}(\overline{tag}, C)$

$\mathcal{B}$ perfectly simulates *Game 3* for $\mathcal{A}$. The challenge $K_b^\star$ and $C^\star$ is distributed as expected, where the intermediate values $\overline{C_1^\star}, \overline{C_2^\star}, K_0^\star, K_1^\star, b$ are computed by $\mathcal{B}$'s environment, and $\overline{tag^\star}$ is chosen appropriately by $\mathcal{B}$ itself. The modified $\mathsf{KEM}.\mathsf{Decaps}$-oracle works as expected, too. The first two cases reflect the modifications introduced by *Game 2* and 3, and in the third case, the answer is the correct decapsulation of the ciphertext. $\mathcal{B}$ uses its $\overline{\mathsf{TBKEM}}.\mathsf{wDecaps}$-oracle only with $\overline{tag} = \overline{C_1} \neq \overline{C_1^\star} = \overline{tag^\star}$, and therefore, the oracle never aborts. $\mathcal{B}$ wins exactly if $\mathcal{A}$ does. □

Remark: The proof above also gives IND-CCA$_1$ security (Def. 2.3 on p.16) for $Core(\overline{\mathsf{TBKEM}})$ from weaker preconditions. Recall that an IND-CCA$_1$ experiment is the same as an IND-CCA$_2$ experiment, except that the $\mathsf{KEM}.\mathsf{Decaps}$ oracle may be used only in phase 1. Hence, for proving IND-CCA$_1$ security, *Game 3* in the proof is not necessary. The uniqueness of C_2 as well as the simulatable rejections from the definition of partitioned TBKEM (Def. 4.2 on p.47) have only been needed for the transition from *Game 2* to *Game 3*. Consequently, IND-CCA$_1$ security can be obtained without these properties:

Corollary 4.6. *If $\overline{\mathsf{TBKEM}}$ has a split of ciphertexts, unpredictability of C_1 (as in Def. 4.2 on p.47), and is ϵ-strong-IND-sTag-wCCA$_1$ secure (analoguesly to strong IND-sTag-wCCA$_2$ security; Def. 4.4 on p.48), then follows:*

$$\mathsf{KEM} := Core(\overline{\mathsf{TBKEM}}) \text{ is } (\epsilon + q_{\mathsf{Decaps}} \cdot \epsilon_1/2)\text{-IND-CCA}_1 \text{ secure,}$$

where ϵ_1 is the upper bound for the C_1-unpredictability Definition 4.2, and where q_{Decaps} is the number of the attacker's oracle queries.

Any TBKEM can easily be transformed into a $\overline{\mathsf{TBKEM}}$ with these two properties:

$\overline{\mathsf{TBKEM}}.\mathsf{Encaps}(PK, tag)$:		$\overline{\mathsf{TBKEM}}.\mathsf{Decaps}(SK, tag, C)$:	
C_1	$\leftarrow \{0,1\}^\lambda$	$\langle C_1, C_2 \rangle$	$\leftarrow C$
K, C_2	$\leftarrow \mathsf{TBKEM}.\mathsf{Encaps}(PK, tag)$	K	$\leftarrow \mathsf{TBKEM}.\mathsf{Decaps}(SK, tag, C_2)$

Clearly, this fulfills the split property, as C_1 does not depend on tag. Furthermore, C_1 is unpredictable, as any possible value occurs with probability $2^{-\lambda} \in negl(\lambda)$. Nonetheless, this looks very strange, as C_1 is not used at all during Decaps. Setting $\mathsf{KEM} := Core(\overline{\mathsf{TBKEM}})$ gives C_1 a meaningful interpretation:

$\mathsf{KEM.Encaps}(PK)$:
$C_1 \leftarrow \{0,1\}^\lambda$
$tag \leftarrow C_1$
$K, C_2 \leftarrow \mathsf{TBKEM.Encaps}(PK, tag)$

$\mathsf{KEM.Decaps}(SK, tag, C)$:
$\langle C_1, C_2\rangle \leftarrow C$
$tag \leftarrow C_1$
$K \leftarrow \mathsf{TBKEM.Decaps}(SK, tag, C_2)$

I. e., in order to create an IND-CCA$_1$ secure KEM from an IND-sTag-wCCA$_1$ secure TBKEM, simply do the following: Choose a random tag for every new encapsulation, compute a ciphertext C under this tag, and send $\langle tag, C\rangle$ to the receiver. Probably, this is, what Canetti et al. [CHK04, Acknowledgments] have meant with their statement: "our techniques imply a conversion from weak IBE to 'lunchtime' CCA1 security with essentially no overhead".

4.3 The Hash-Based Transformation

As seen in the last section, the partition property (Def. 4.2 on p.47) together with *strong* IND-sTag-wCCA$_2$ security (Def. 4.4 on p.48) allows for very efficient transformations from TBKEM to IND-CCA$_2$ secure KEM. Unfortunately, for validation of the stronger security notion, the security proof has to be examined carefully. Even if a scheme has this property, it might not be as easy to prove this as it is for BB1-TBKEM. Interestingly, there exists a transformation from normal to strong IND-sTag-wCCA$_2$ security introducing as overhead only a target collision resistant hash function and one XOR computation.

The IND-sTag-wCCA$_2$ experiment (Def. 2.5 on p.18) guarantees security if the attacker chooses the $tag^\star$ at the beginning of the experiment. It cannot depend on any other values $\mathcal{A}$ will see later. In the strong IND-sTag-wCCA$_2$ experiment this independence is discarded: Knowing $C_1^\star$ could help an attacker $\mathcal{A}$ of TBKEM in selecting an "easy" $tag^\star$. To cope with this, the following transformation implements a "shuffle" of the tags. In the security simulate this will mix up the tags after $\mathcal{A}$ has selected $tag^\star$. So, in fact, a random tag is attacked that is independent of $C_1^\star$.

Construction 4.7. *The hash-based transformation* $\mathsf{TBKEM} := HB(\overline{\mathsf{TBKEM}}, H)$ *for a partitioned* $\overline{\mathsf{TBKEM}}$ *and a hash function* $H_k : \{0,1\}^* \to \{0,1\}^{\hat{\ell}(\lambda)}$ *(§2.4.1 on p.22) is defined as follows:*

$\mathsf{TBKEM.KeyGen}(1^\lambda)$	$\overline{PK}, \overline{SK}$	$\leftarrow \overline{\mathsf{TBKEM}}.\mathsf{KeyGen}(1^\lambda)$
	k	$\leftarrow \{0,1\}^{\ell(\lambda)}$
	h	$\leftarrow \{0,1\}^{\hat{\ell}(\lambda)}$
	PK, SK	$\leftarrow \langle \overline{PK}, k, h\rangle, \overline{SK}$
$\mathsf{TBKEM.Encaps}_1(PK)$	C_1, σ	$\leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Encaps}_1(\overline{PK})$
$\mathsf{TBKEM.Encaps}_2(PK, tag, \sigma)$	$\overline{tag}$	$\leftarrow H_k(tag) \oplus h$
	K, C_2	$\leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Encaps}_2(\overline{PK}, \overline{tag}, \sigma)$
$\mathsf{TBKEM.Decaps}(SK, tag, C)$	$\overline{tag}$	$\leftarrow H_k(tag) \oplus h$
	K	$\leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Decaps}(\overline{SK}, \overline{tag}, C)$

This lifts the standard to strong IND-sTag-wCCA$_2$ security (Def. 2.5 on p.18, Def. 4.4 on p.48) by using the TCR-security (Def. 2.7 on p.22) of H.

Theorem 4.8. *If* $\overline{\mathsf{TBKEM}}$ *is* ϵ*-IND-sTag-wCCA*$_2$ *and* H *is* $\hat{\epsilon}$*-TCR secure, then:*

$$\mathsf{TBKEM} := HB(\overline{\mathsf{TBKEM}}, H) \text{ is } (\epsilon + \hat{\epsilon}/2)\text{-strong-IND-sTag-wCCA}_2 \text{ secure.}$$

Proof. The proof is quite simple: Let $\mathcal{A}$ be an attacker of TBKEM in the *strong* IND-sTag-wCCA$_2$ experiment. It has to be shown that its advantage is smaller than $\epsilon = \bar{\epsilon} + \hat{\epsilon}/2$. Therefore, an algorithm $\mathcal{B}$ is constructed to attack a randomly chosen $\overline{tag}^\star$ in the underlying $\overline{\mathsf{TBKEM}}$. After $\mathcal{A}$ has chosen its target $tag^\star$, $\mathcal{B}$ can use h to translate $tag^\star$ (in TBKEM) into $\overline{tag}^\star$ (in $\overline{\mathsf{TBKEM}}$). One intermediate step is needed to get rid of collisions in H_k.

Game 1 This is the strong IND-sTag-wCCA$_2$ experiment for TBKEM.

Game 2 Any query of $\mathcal{A}$ to the TBKEM.wDecaps oracle for a tag with $H_k(tag) = H_k(tag^\star)$ will abort the experiment.

***Game 1* to *Game 2*:** This is completely analogous to the proof of Theorem 3.5 (p.32), where collisions in a hash function had to be removed. The result is:

$$|Pr^{\mathcal{A}}_{Game\ 1}[b = b'] - Pr^{\mathcal{A}}_{Game\ 2}[b = b']| \quad \leq \quad \mathrm{Adv}^{H,\mathcal{B}}_{\mathrm{TCR}}(\lambda)/2 \quad \leq \quad \hat{\epsilon}(\lambda)/2$$

***Game 2* to $\overline{\mathsf{TBKEM}}$:** This game can be easily simulated by an IND-sTag-wCCA$_2$ attacker $\mathcal{B}$ for $\overline{\mathsf{TBKEM}}$. As attacker of the "normal" (non-strong) security, $\mathcal{B}_1$ first has to choose a target $\overline{tag}^\star$ for $\overline{\mathsf{TBKEM}}$. Then $\mathcal{B}_2$ gets the public key $\overline{PK}$ and the challenge key and ciphertext. On the other hand, $\mathcal{A}_1$ may use input $C_1^\star$ to select a target $tag^\star$ for TBKEM. $\mathcal{B}$ works as follows:

$\mathcal{B}_1(1^\lambda)$**:** output $\overline{tag}^\star \leftarrow \{0,1\}^{\hat{\ell}(\lambda)}$ for the own attack on $\overline{\mathsf{TBKEM}}$

$\mathcal{B}_2^{\overline{\mathsf{TBKEM}}.\mathsf{wDecaps}}(\overline{PK}, K_b^\star, C^\star)$**:**

- parse $\langle C_1^\star, C_2^\star \rangle \leftarrow C^\star$
- run $\mathcal{A}_1(C_1^\star)$ to determine $tag^\star$
- choose a hash key $k \leftarrow \{0,1\}^{\ell(\lambda)}$
- set $h \leftarrow H_k(tag^\star) \oplus \overline{tag}^\star$
- and $PK \leftarrow \langle \overline{PK}, k, h \rangle$
- start $\mathcal{A}_2^{\mathsf{TBKEM.Decaps}}(PK, K_b^\star, C_2^\star)$
- use $\mathcal{A}_2$'s output b' as own guess

TBKEM.wDecaps-oracle(tag, C) **for** $\mathcal{A}_2$**:**

- if $H_k(tag) = H_k(tag^\star)$ abort, else
- $\overline{tag} \leftarrow H_k(tag) \oplus h \qquad \left(\neq H_k(tag^\star) \oplus h = \overline{tag}^\star \right)$
- $K \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Decaps\text{-}oracle}(\overline{tag}, C)$

h is chosen as expected, since it is uniform in $\{0,1\}^{\ell(\lambda)}$ and independent from any other value of PK and $tag^\star$. The challenge is computed (by $\mathcal{B}$'s environment) as $\overline{\mathsf{TBKEM}}.\mathsf{Encaps}(\overline{PK}, \overline{tag^\star})$. This is equivalent to $\mathsf{TBKEM}.\mathsf{Encaps}(\overline{PK}, tag^\star)$, as $\overline{tag^\star} = H_k(tag^\star) \oplus h$. In the same way, all answers of the $\mathsf{TBKEM}.\mathsf{wDecaps}$-oracle fulfill $\mathcal{A}$'s expectations. So $\mathcal{B}$ gives a perfect simulation and wins exactly if $\mathcal{A}$ wins:

$$|Pr^{\mathcal{A}}_{Game\ 2}[b = b'] - 1/2| \quad = \quad \mathrm{Adv}^{\overline{\mathsf{TBKEM}},\mathcal{B}}_{\text{IND-sTag-wCCA}_2}(\lambda) \quad \leq \quad \epsilon(\lambda)$$

Adding both equations gives $\epsilon + \hat{\epsilon}/2$ as bound for $\mathrm{Adv}^{\mathsf{TBKEM},\mathcal{A}}_{\text{strong IND-sTag-wCCA}_2}(\lambda)$. □

4.4 Review of both Transformations

In this section, first the application of the core and hash-based transformation to BB1-TBKEM is presented. Then, the general applicability of these transformations to other schemes is briefly discussed. Finally, its applicability is compared to that of the ACIK transformations.

Applying the Core Transformation to BB1-TBKEM:
The core transformation (Con. 4.3 on p.48) results from analysis of the relation between BMW-KEM and BB1-TBKEM (Figure 4.1 on p.45). As a minimal requirement, it should be applicable to BB1-TBKEM and result exactly in BMW-KEM. The first thing to check is whether BB1-TBKEM is partitioned (Def. 4.2 on p.47).

- The *split of ciphertext* is as follows:
  PK_1 consists of $\langle p, \mathbb{G}, g\rangle$
  $\mathbf{Encaps}_1(PK_1)$; $s \leftarrow \mathbb{Z}_p$; $\sigma \leftarrow s$; $C_1 \leftarrow g^s$
  $\mathbf{Encaps}_2(PK, tag, \sigma)$; $s \leftarrow \sigma$; $C_2 \leftarrow \left(g_1^{H_k(tag)} \cdot g_3\right)^s$
- The *unpredictability of* C_1 stems from the fact that C_1 is uniformly distributed in $\mathbb{G}$, since g is a generator of $\mathbb{G}$ and $|\mathbb{G}| = p$. This gives $\epsilon_1(\lambda) = \frac{1}{p}$. Note, that p is part of the output from $\mathsf{BGGen}(1^\lambda)$ and (although not shown here) grows exponentially in λ. This gives that $\epsilon_1 \in negl(\lambda)$.
- The *uniqueness of* C_2 is easy, too: Since g is a generator and $C_1 = g^s$, g and C_1 uniquely determine s. As result, C_2 is determined uniquely by the following sequence of mappings: $PK, tag, C_1 \mapsto PK, tag, s \mapsto \left(g_1^{H_k(tag)} \cdot g_3\right)^s = C_2$.
- As any invalid ciphertext fails the validity check in the Decaps algorithm ofBB1-TBKEM (compare Fact 3.3 on p.32), the output then is always $\perp$. This makes the Reject algorithm trivial: It simply returns $\perp$ on any input.

Next, it has to be verified that BB1-TBKEM is not only standard but *strong* IND-sID-wCCA$_2$ secure (Def. 4.4 on p.48). As described before, this can be done by looking at the proof of Theorem 4.1 (p.44) which is based on Theorem 3.2 (p.30).

One has to verify whether the simulator $\mathcal{B}$ is able to give attacker $\mathcal{A}$ the first part of the challenge ciphertext $C_1^\star$ at the beginning of phase 1 instead of the beginning of phase 2. This is clearly possible as $\mathcal{B}$ simply passes its input z as $C_1^\star$ to $\mathcal{A}$. Consequently, BB1-TBKEM is strong IND-sTag-wCCA$_2$ secure.

The above considerations mean that the core transformation is applicable to BB1-TBKEM. It is easy to see that $Core$(BB1-TBKEM) = BMW-KEM. Even the resulting security bound for BMW-KEM is as in Theorem 3.8 (p.37).

Applying the Hash-Based Transformation to "non-strong" BB1-TBKEM: What happens if someone is not willing to examine if a scheme's proof of IND-sTag-wCCA$_2$ security already suffices to prove the *strong* variant? Or if a scheme simply does not satisfy this stronger variant?

In this case, the hash-based transformation (Con. 4.7 on p.53) can be applied to lift the standard to strong IND-sTag-wCCA$_2$ security first, which then allows the application of the core transformation. Fortunately, the overhead is very small. $Core(HB$(BB1-TBKEM)) is quite similar to BMW-KEM. The public key would be extended by a second hash key $\hat{k}$ and the perturbation $h \in \{0,1\}^{\ell(\lambda)}$. The ciphertexts would look like

$$C = \langle C_1, C_2 \rangle = \left\langle g^s, \left(g_1^{H_k\left(H_{\hat{k}}(tag) \,\oplus\, h\right)} \cdot g_3 \right)^s \right\rangle$$

and encapsulation/decapsulation would only involve one additional hash and XOR computation introduced by the hash-based transformation.

But there is an interesting observation that allows to optimize this a little bit. The hash-based transformation may also be applied to any $\overline{\text{TBKEM}}$ with a restricted $TagSpace \neq \{0,1\}^*$. The only constraint is that it must have a group structure, e. g., $TagSpace = (\mathbb{Z}_p, +)$. Then, H has to be a hash function mapping into $\mathbb{Z}_p$ instead of $\{0,1\}^{\hat{\ell}(\lambda)}$, and the perturbation h has to be chosen from $\mathbb{Z}_p$. For encapsulation and decapsulation, $\overline{tag} \leftarrow H_k(tag) \oplus h \in \{0,1\}^{\hat{\ell}(\lambda)}$ is replaced by $\overline{tag} \leftarrow H_k(tag) + h \in \mathbb{Z}_p$.

There are many schemes which, like BB1-TBKEM, are defined with a restricted $TagSpace$ first and then extended to arbitrary tags (similar to Con. 3.4 on p.32). This allows to apply the hash-based transformation and the core transformation directly to those restricted versions. For example, $Core(HB$(hfBB1-TBKEM)) does look even more similar to BMW-KEM than $Core(HB$(BB1-TBKEM))—still without requiring the hash-free version of BB1-TBKEM to be *strong* IND-sTag-wCCA$_2$ secure. A ciphertext would look like

$$C = \langle C_1, C_2 \rangle = \left\langle g^s, \left(g_1^{H_k(tag) \,+\, h} \cdot g_3 \right)^s \right\rangle$$

This differs from BMW-KEM only in one addition in $\mathbb{Z}_p$. Particularly, there is no need for a second hash-function and no weakening of the security bound!

Applicability to other Schemes: The partition property is quite easy to check and fulfilled by many schemes, e. g., [BF01, BB04, Wat05, BBG05, BW06, Kil06, Kil07]. The same holds for the *strong* IND-sID-CPA security, although this is a little more elaborate to verify—but not much! In the security proofs of almost all these schemes, the simulator $\mathcal{B}$ simply passes one of its own input values as C_1, like in the proof of BB1-TBKEM. Furthermore, all are defined first with restricted identity spaces $\mathbb{Z}_p$ or $\{0,1\}^n$ and afterwards extended to $\{0,1\}^*$. Consequently, even if someone does not accept their strong security, they can be efficiently transformed by using the optimized version of the hash-based transformation as shown in the previous paragraph. Three of these schemes are particularly interesting, as they point out the advantage of basing the definitions on TBKEM instead of IBKEM:

Kiltz [Kil06] constructs an IND-sTag-wCCA$_2$ secure TBE from algebraic primitives which are currently not known to give IBE. He furthermore constructs an IND-CCA$_2$ secure KEM out of it using the same technique as for BMW-KEM. In particular, again the security of the KEM has to be proven from scratch. The relation between a TBKEM version of his TBE and the KEM can be perfectly explained with the core transformation. This would not have been possible if the core transformation had been defined only for IBKEM instead of TBKEM.

Another example is the IND-CCA$_2$ secure KEM in [Kil07]. Using the core transformation, the construction can be "reverse engineered". The result is an IND-sTag-wCCA$_2$ secure TBKEM that may be seen of a generalization of BB1-TBKEM. Thus, the original KEM can be interpreted as the result of applying the core transformation to the reverse engineered TBKEM. On the other hand, it does not seem possible to interpret it as originating from an IBKEM.

Boneh and Boyen also have defined a second IBE [BB04, §5], called BB2-IBE. The encryption algorithm is the same as for BB1-IBE but decryption is more efficient. In the original version of [BMW05a, §4.4], the authors claim in a footnote that their techniques to obtain BMW-KEM from BB1-IBE cannot be transferred to obtain some KEM from BB2-IBE. Later in an updated version [BMW05b, §4.4], they revoke this claim and note that the resulting KEM is surprisingly similar to BMW-KEM. One benefit of the core transformation is that it explicitly points out the requirements a scheme has to fulfill. Verifying these properties is less error-prone than comparing the scheme to BB1-IBE and deciding whether "somehow" the techniques to obtain BMW-KEM can be transferred. A second benefit of the core transformation is that it can explain why the resulting KEMs for BB1-IBE and BB2-IBE are so similar. In fact, the cause is not the core transformation—the TBKEM versions of both schemes are already almost identical. This observation would not have been possible if the core transformation was based on IBKEM.

There exist schemes that are not partitioned [Gen06, BGH07, Coc01, SK03]. Although the core transformation applied to the first two schemes gives IND-CCA$_2$ security, this cannot be concluded from the above theorems, as they do not have simulatable rejections. The third scheme is very inefficient and thus not of interest anyway. Interestingly, the fourth scheme is the only that even does not have splittable ciphertexts, i. e., all parts of the ciphertext depend on ID (or tag in the

TBKEM version). This can easily be "repaired" but makes the result less efficient. Furthermore, all schemes but the first are only secure in the random oracle model. If on is willing to accept the heuristic security arguments of the random oracle model, there are much more efficient constructions.

Applicability Compared to the ACIK Transformations: Last but not least, the work of [ACIK07] (see §3.5 on p.38) shall be compared to the work in this thesis.

One very obvious difference is the use of IBKEM vs. TBKEM. The latter approach followed in this thesis is a little bit more general: It covers the IBKEM setting due to the generic IBKEM-to-TBKEM transformation (§2.3 on p.19) but broadens the applicability as described in the paragraph before. Nevertheless, the approach of using TBKEM is not novel as it has already been suggested by Abe et al. [ACIK07, §7.3]. In the following comparison, it is best to think of IND-sID-CPA secure IBKEM and IND-sTag-wCCA$_2$ secure TBKEM as the same things.

The definitions of ACIK-partitioned IBKEM (Def. 3.13 on p.39) and partitioned TBKEM (Def. 4.2 on p.47) are almost the same. The independence property of K in ACIK-partitionedness is needed to avoid cyclic dependencies between $\overline{C_M} \leftarrow M \oplus K$ and $\overline{ID}$ in Construction 3.15 (p.40). For constructing a KEM this is not necessary, and thus it is not part of the partitionedness property defined here. On the other hand, it is a not very restrictive property, as it is fulfilled by all the above mentioned schemes except [BF01, Coc01, BGH07].

The rejection properties of both definitions are very similar, too. In fact, the simulatable rejection property is more general than perfect \$-rejection, but using this more general version has already been remarked in [ACIK07].

The most notably difference between both definitions is the absence of the C_1 unpredictability in ACIK-partitioned IBKEM. In fact, this is due to a slight flaw in the security proofs of the ACIK transformations. By correctly guessing the challenge ciphertext $C^\star$ and asking for its decryption during phase 1, an attacker can break their proof of security for both ACIK transformations. One possibility to circumvent this flaw is adding the C_1 unpredictability to the definition of ACIK-partitioned IBKEM.

All in all, both definitions of partitionedness are quite similar. In particular, for every ACIK-partitioned IBKEM, the result of the generic IBKEM-to-TBKEM transformation (§2.3 on p.19) is a partitioned TBKEM. At least, if one assumes that the C_1 unpredictability is added to the definition of ACIK-partitioned IBKEM.

Strong IND-sTag-wCCA$_2$ security (Def. 4.4 on p.48) is related to *non-strong* IND-sTag-wCCA$_2$ security (Def. 2.3 on p.16) in an analogue way as *strong* IND-aID-CPA security (Def. 3.14 on p.40) is related to *non-strong* IND-aID-CPA security (Def. 2.4 on p.17). The main similarity of the strong versions that both give $C_1^\star$ to $\mathcal{A}$ already in phase 1 instead of phase 2. The main difference is that ACIK version needs adaptive-identity security, whereas in this thesis selective-tag security suffices. The ACIK version also gives $K_b^\star$ to $\mathcal{A}$ already in phase 1, but as for the independence property of K this is not needed here.

One can conclude that the definition of strong security given here is less demanding than the ACIK version, as strong IND-aID-CPA security implies strong IND-sTag-wCCA$_2$ security (via the generic IBKEM-to-TBKEM transformation, as above). Furthermore, this definition really is useful, as there are several schemes that are strong IND-sTag-wCCA$_2$ secure but do not have aID/aTag security. The effort someone has to put into the verification whether a proof of standard security already gives the strong version is comparable in the adaptive and selective setting.

All in all, the stronger version of security used in this thesis is neither "less natural" than the ACIK version, nor "not easier than providing a direct proof for the transformed KEM" (compare the quote on p.42). If the stronger needs of the first ACIK transformation (Construction 3.15) are fulfilled, then it is preferable to use this transformation and obtain very efficient PKE. In cases where this is not given but the preconditions of the core transformation (Construction 4.3) are still satisfied, the work of this thesis still allows very efficient transformations to KEM.

5 CCA_2 Security from Generic Tag-/Identity-Based KEM

In the introduction it was claimed that both the CHK transformation and the construction of BMW-KEM are based on the same paradigms. The core transformation may be seen as the formalization of those paradigms used for BMW-KEM. Thus, it is a self-evident idea to investigate the relationship between the CHK transformation and the core transformation. It turns out that the CHK transformation may be seen as a composition of the core transformation and a second transformation if the definition of partitioned TBKEM is slightly generalized. Section 5.1 presents the basic construction and names problems that occur with the old definition. In Section 5.2, the necessary generalizations are made, and the security of the core transformation under those new circumstances is proven. Finally, in Section 5.3 the construction from the first section is proven to fulfill the definition of the second section.

5.1 The Signature-Based Transformation (Idea)

To accomplish the above goal, first (a TBKEM version of) the CHK transformation (Con. 3.9 on p.37) is broken into two parts. The first one is the core transformation (Con. 4.3 on p.48) and the remaining part has to lift an arbitrary TBKEM (§2.1.4 on p.13) into a partitioned TBKEM (Def. 4.2 on p.47) such that both combined (syntactically) give the CHK transformation. This results in the following:

Construction 5.1. *The signature-based transformation* $\mathsf{TBKEM} := SB(\overline{\mathsf{TBKEM}})$ *for an arbitrary* $\overline{\mathsf{TBKEM}}$ *and a signature scheme* $\overline{\mathsf{Sig}}$ *is defined as follows:*

$\mathsf{TBKEM.KeyGen}(1^\lambda)$	PK, SK	$\leftarrow \overline{\mathsf{TBKEM}}.\mathsf{KeyGen}(1^\lambda)$
$\mathsf{TBKEM.Encaps}_1(PK)$	$VK, SigK$	$\leftarrow \overline{\mathsf{Sig}}.\mathsf{KeyGen}(1^\lambda)$
	C_1, σ	$\leftarrow VK, SigK$
$\mathsf{TBKEM.Encaps}_2(PK, tag, \sigma)$	$SigK$	$\leftarrow \sigma$
	$K, \overline{C}$	$\leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Encaps}(PK, tag)$
	sig	$\leftarrow \overline{\mathsf{Sig}}.\mathsf{Sign}(SigK, \overline{C})$
	C_2	$\leftarrow \langle \overline{C}, sig \rangle$
$\mathsf{TBKEM.Decaps}(SK, tag, C)$	$\langle VK, \overline{C}, sig \rangle$	$\leftarrow C$
	check	$\overline{\mathsf{Sig}}.\mathsf{Verify}(VK, \overline{C}, sig) \stackrel{?}{=} true$
	if not	*return* $K \leftarrow \perp$
	K	$\leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Decaps}(SK, tag, \overline{C})$

The TBKEM version of the CHK transformation turns arbitrary IND-sTag-wCCA$_2$ secure (Def. 2.5 on p.18) $\overline{\mathsf{TBKEM}}$ into IND-CCA$_2$ secure (Def. 2.3 on p.16) KEM. Furthermore, it can be syntactically split into the core transformation and the signature-based transformation. Therefore, it is reasonable to hope that the signature-based transformation lifts arbitrary IND-sTag-wCCA$_2$ secure $\overline{\mathsf{TBKEM}}$ to s *strong* IND-sTag-wCCA$_2$ secure *partitioned* TBKEM (Def. 4.2 on p.47 and Def. 4.4 on p.48), i. e., that fulfills the prerequisites of the core transformation.

Unfortunately, at first glance, the signature-based construction does not seem to fit the definition for partitioned TBKEM at all. There are problems with every part of the definition, except the (quite trivial) split property:

- It is unclear if $C_1 = VK$ is unpredictable for arbitrary signature schemes.
- $C_2 = \langle \overline{C}, sig \rangle$ does not have the uniqueness property, as due to the randomness in $\overline{\mathsf{TBKEM}}.\mathsf{Encaps}$ and $\overline{\mathsf{Sig}}.\mathsf{Sign}$ $\mathsf{TBKEM}.\mathsf{Encaps}_2$ is not deterministic. As result, for fixed PK, tag, C_1 there might be many different valid C_2.
- It might not be possible to simulate the rejection of every invalid ciphertext. Given a ciphertext $C = \langle VK, \overline{C}, sig \rangle$ with a correct signature but an invalid $\overline{C}$, it is impossible to simulate the outcome of $\mathsf{TBKEM}.\mathsf{Decaps}$, unless $\overline{\mathsf{TBKEM}}$ has simulatable rejections, too.
- It might not even be possible to recognize every invalid ciphertext. This problem holds for the same class of invalid ciphertexts as above, unless $\overline{\mathsf{TBKEM}}$ has public verifiability, i. e., it is easy to tell whether $\overline{C}$ is valid or not.

The first point is the easiest. Indeed, it is easy to show that in a secure signature scheme there cannot be a single VK that occurs with significant probability. For the other problems, a closer look at the security proof of the core transformation (Thm. 4.5 on p.49) shows that the properties are not needed in their full entirety. The observations below will result in a slight generalization of the definition, i. e., any scheme fulfilling the old definition will also comply with the new one.

The uniqueness of C_2 has been used in the proof step from *Game 2* to *Game 3*. It ensures that it is *impossible* for an attacker to modify the challenge ciphertext $C^\star = \langle \overline{C_1^\star}, \overline{C_2^\star} \rangle$ into some different *valid* ciphertext $C = \langle \overline{C_1^\star}, \overline{C_2} \rangle$ and ask for the decapsulation of C. For the above construction, this is not impossible but at least *computationally infeasible*. The security of the signature makes it hard to modify $C^\star = \langle VK^\star, \overline{C^\star}, sig^\star \rangle$ into a *pseudo-valid* ciphertext $C = \langle VK^\star, \overline{C}, sig \rangle$, where pseudo-valid means that it is not necessarily valid but at least has a correct signature. Clearly, this *integrity of* C_2 (instead of uniqueness) already suffices.

The simulatable rejection property has been used in *Game 3* in order to simulate the decapsulation of all ciphertexts that have been recognized as invalid due to the uniqueness of C_2. If this uniqueness property is relaxed to integrity, it suffices if only non-pseudo-valid ciphertexts (which are a subset of all non-valid ciphertexts) are rejectable. In the above example, non-pseudo-valid ciphertexts are easily recognized by checking the signature and rejected by outputting $\perp$.

5.2 The Generalized Core Transformation

All the observations of the previous section lead to the following generalization of the definition for partitioned TBKEM:

Definition 5.2 (replaces Def. 4.2 on p.47)**.** *A TBKEM (§2.1.4 on p.13) is* partitioned *if it has the following properties:*

- Split of ciphertext*: The ciphertext C can be split into two parts, $C = \langle C_1, C_2 \rangle$, where the first part C_1 only depends on the public key PK and not the tag. Again, this allows to split* Encaps *into two algorithms* Encaps_1 *and* Encaps_2*.*
- Unpredictability of C_1*: There is a negligible bound $\epsilon_1(\lambda)$ such that*

$$\max_{PK} \max_{C_1} \quad Pr[C_1^\star \leftarrow \mathsf{Encaps}_1(PK) : C_1^\star = C_1] \quad =: \quad \epsilon_1(\lambda) \in negl(\lambda)$$

- Pseudo-valid ciphertexts: *For all fixed $PK, tag^\star$, there is a denoted set of pseudo-valid ciphertexts $pv(PK, tag^\star) \supseteq \mathcal{C}(PK, tag^\star)$. Also, an efficient algorithm $\mathsf{InPV}(PK, tag^\star, C_1^\star, C_2^\star, C_2)$ exists that decides if $\langle C_1^\star, C_2 \rangle \in pv(PK, tag^\star)$, given that $\langle C_1^\star, C_2^\star \rangle \in pv(PK, tag^\star)$.*
- Integrity of C_2*: There is a negligible $\epsilon_2(\lambda)$ that bounds for any $\mathcal{A}$ the advantage $Adv^{\mathsf{TBKEM},\mathcal{A}}_{C_2\text{-}INT}(\lambda) := Pr[Exp^{\mathsf{TBKEM},\mathcal{A}}_{C_2\text{-}INT}(\lambda) = win]$ in the following experiment:*

$$\begin{array}{l} Exp^{\mathsf{TBKEM},\mathcal{A}}_{C_2\text{-}INT}(\lambda)\text{:} \\ \quad PK, SK \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \quad C_1^\star, \sigma^\star \leftarrow \mathsf{Encaps}_1(PK) \\ \quad tag^\star \leftarrow \mathcal{A}_1(C_1^\star) \\ \quad K^\star, C_2^\star \leftarrow \mathsf{Encaps}_2(PK, tag^\star, \sigma^\star) \\ \quad C_2 \leftarrow \mathcal{A}_2^{\mathsf{wDecaps}}(PK, K^\star, C_2^\star) \\ \quad \textit{if } \langle C_1^\star, C_2 \rangle \in pv(PK, tag^\star) \\ \qquad \textit{and } C_2 \neq C_2^\star \textit{ then output win} \end{array}$$

- Simulatable rejection*: There is an efficient algorithm $\mathsf{Reject}(PK, tag, C)$, that has the same output distribution as $\mathsf{Decaps}(SK, tag, C)$ for any ciphertext $C \notin pv(PK, tag)$. There is no requirement for the output distribution of* Reject *on any input with $C \in pv(PK, tag)$!*

Clearly, the old definition is a special case of the new one. To see this, one has to set $pv(PK, tag) := \mathcal{C}(PK, tag)$. The uniqueness property implies for any fixed $PK, tag^\star, C_1^\star$ that there is only one $C_2^\star$ such that $\langle C_1^\star, C_2^\star \rangle$ is valid, i. e., an element of $\mathcal{C}(PK, tag^\star) = pv(PK, tag^\star)$. Thus, InPV can be implemented by testing if $C_2 = C_2^\star$ holds. Furthermore, it is impossible to win the C_2-INT experiment, since there is no other correct solution $C_2 \neq C_2^\star$. This gives $\epsilon_2 := 0$.

The next task is to show IND-CCA$_2$ security (Def. 2.3 on p.16) of the core transformation (Con. 4.3 on p.48) under these generalized circumstances.

Theorem 5.3. *If* $\overline{\mathsf{TBKEM}}$ *is partitioned (in the new sense of Definition 5.2) and* ϵ*-strong-IND-sTag-wCCA*$_2$ *secure (Def. 4.4 on p.48), then:*

$$\mathsf{KEM} := Core(\overline{\mathsf{TBKEM}}) \text{ is } (\epsilon + q_{\mathsf{Decaps}} \cdot \epsilon_1/2 + \epsilon_2/2)\text{-IND-CCA}_2 \text{ secure,}$$

where $\overline{\epsilon}_1, \overline{\epsilon}_2$ *are the upper bounds for the* C_1 *unpredictability and for the* C_2 *integrity in Definition 5.2, and where* q_{Decaps} *is the number of* $\mathcal{A}$*'s oracle queries.*

Proof. The proof is basically the same as for Theorem 4.5 (p.49), except that the modification of the KEM.Decaps oracle in *Game 3* is slightly generalized, and the difference between *Game 2* and *Game 3* (the newly introduced abort below) is bounded with help of the C_2 integrity.

***Game 1* and *Game 2*:** as in Theorem 4.5.

***Game 3*:** The modified KEM.Decaps oracle for phase 2 works on input C as follows:

- parse $\langle \overline{C_1^\star}, \overline{C_2^\star} \rangle \leftarrow C^\star$ ($C^\star$ is already determined in phase 2)
- parse $\langle \overline{C_1}, \overline{C_2} \rangle \leftarrow C$
- set $\overline{tag} \leftarrow \overline{C_1}$
- if $\overline{C_1} = \overline{C_1^\star} \wedge \overline{C_2} = \overline{C_2^\star}$:
  abort the experiment
- if $\overline{C_1} = \overline{C_1^\star} \wedge \overline{C_2} \neq \overline{C_2^\star} \wedge \mathsf{InPV}(PK, tag^\star, \overline{C_1^\star}, \overline{C_2^\star}, \overline{C_2})$:
  abort the experiment
  (This case did not exist in *Game 3* of Theorem 4.5, as a ciphertext with $\overline{C_1} = \overline{C_1^\star}$ but $\overline{C_2} \neq \overline{C_2^\star}$ was invalid, there.)
- if $\overline{C_1} = \overline{C_1^\star} \wedge \neg\mathsf{InPV}(PK, \overline{tag}, \overline{C_1^\star}, \overline{C_2^\star}, \overline{C_2})$:
  return $K \leftarrow \mathsf{Reject}(PK, \overline{tag}, \langle \overline{C_1}, \overline{C_2} \rangle)$
- if $\overline{C_1} \neq \overline{C_1^\star}$:
  return $K \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Decaps}(SK, \overline{tag}, C)$

Game 2 and *Game 3* differ if a pseudo-valid ciphertext $\langle \overline{C_1^\star}, \overline{C_2} \rangle \neq \langle \overline{C_1^\star}, \overline{C_2^\star} \rangle$ is submitted to the Decaps oracle. Let this event be denoted as failure event F. $Pr_{Game\ 3}[F]$ is bounded by the success probability of an algorithm $\mathcal{B}$ that attacks $\overline{\mathsf{TBKEM}}$ in the C_2-INT experiment (see Definition 5.2) and simulates *Game 3* for $\mathcal{A}$ as follows:

$\mathcal{B}_1(PK_1, C_1^\star)$**:** output $tag^\star \leftarrow C_1^\star$

$\mathcal{B}_2^{\overline{\mathsf{TBKEM}}.\mathsf{wDecaps}}(PK, K^\star, C_2^\star)$**:**

- $b \leftarrow \{0,1\}$; $K_0^\star \leftarrow K^\star$; $K_1^\star \leftarrow \mathcal{K}(PK, tag^\star)$; $C^\star \leftarrow \langle C_1^\star, C_2^\star \rangle$
- start $\mathcal{A}^{\mathsf{KEM.Decaps}}(PK, K_b^\star, C^\star)$
- if $\mathcal{B}_2$ has not stopped during KEM.Decaps-oracle, then it outputs $C_2 \leftarrow C_2^\star$ as (invalid) solution in its own experiment.

modified KEM.Decaps-oracle(C) **for** $\mathcal{A}$**:**

- simulate this as defined in *Game 3* using the $\overline{\mathsf{TBKEM}}$.Decaps-oracle
- if F happens (i. e., the second "if" in *Game 3*)
  - prematurely stop the execution of $\mathcal{A}$
  - output $\overline{C_2}$ as (correct) solution for $\mathcal{B}$'s own experiment

Clearly, $\mathcal{B}$ wins exactly if F happens. This ends the proof sketch. □

5.3 The Signature-Based Transformation (Proof)

Now, a proof of partitionedness (Def. 5.2 on p.63) and strong IND-sTag-wCCA$_2$ security (Def. 4.4 on p.48) for the signature-based transformation (Con. 5.1 on p.61) finishes all steps to conclude the security of the (TBKEM version of) the CHK transformation (Con. 3.9 on p.37) as corollary.

Theorem 5.4. *If* $\overline{\mathsf{TBKEM}}$ *is* ϵ*-IND-sTag-wCCA*$_2$ *secure (Def. 2.5 on p.18) and* $\overline{\mathsf{Sig}}$ *is* $\hat{\epsilon}$*-SEU-OT secure (Def. 2.8 on p.23), then it follows that:*

$$\mathsf{TBKEM} := SB(\overline{\mathsf{TBKEM}}, \overline{\mathsf{Sig}}) \text{ is partitioned with } \epsilon_1 := \sqrt{\hat{\epsilon}} \text{ and } \epsilon_2 := \hat{\epsilon}$$
$$\text{and furthermore is } (\epsilon + \hat{\epsilon}/2)\text{-strong-IND-sTag-wCCA}_2 \text{ secure.}$$

where ϵ_1, ϵ_2 *are the bounds for* C_1 *unpredictability and* C_2 *integrity.*

Proof. The split of the ciphertext is already given in the construction. The set of pseudo-valid ciphertexts is defined as: $\langle VK, \overline{C}, sig\rangle \in pv(PK, tag)$ if and only if $\mathsf{Verify}(VK, \overline{C}, sig) = true$. This makes the InPV and Reject algorithms trivial: The former is the Verify algorithm and the latter always outputs $\perp$.

Note that $C_1 = VK$ and $\mathsf{Encaps}_1 = \overline{\mathsf{Sig}}.\mathsf{KeyGen}$. Because of this, the unpredictable C_1 property simply claims that there is no verification key which is output with significant probability. Assume that there was a $VK^\star$ that is output with non-negligible probability ϵ_1. Then consider the following attacker $\mathcal{A}$: On input VK it generates a new key pair $\overline{VK}, \overline{SigK}$ and tests whether $VK = \overline{VK}$. In that case, the attacker forges a valid signature sig for an arbitrary message M using $\overline{SigK}$. This gives

$$\mathrm{Adv}^{\overline{\mathsf{Sig}},\mathcal{A}}_{\text{SEU-OT}}(\lambda) = Pr[VK = \overline{VK}] > Pr[VK = VK^\star \wedge \overline{VK} = VK^\star] = (\epsilon_1(\lambda))^2$$

The last step holds since $\mathcal{A}$ and its environment compute $\overline{VK}$ and VK independently. By security of $\overline{\mathsf{Sig}}$ it must hold that $\mathrm{Adv}^{\overline{\mathsf{Sig}},\mathcal{A}}_{\text{SEU-OT}}(\lambda) \leq \hat{\epsilon}(\lambda)$ and therefore

$$\epsilon_1(\lambda) \leq \sqrt{\hat{\epsilon}(\lambda)}$$

The C_2 integrity can easily be bounded by the security of $\overline{\mathsf{Sig}}$. Note that $C_2 = \langle \overline{C}, sig\rangle$ is in terms of $\overline{\mathsf{Sig}}$ a pair of message and valid signature. Let $\mathcal{A}$ be an attacker in the C_2-INT experiment. An attacker $\mathcal{B}$ that uses $\mathcal{A}$ for an attack on the SEU-OT security on $\overline{\mathsf{Sig}}$ works as follows.

$\mathcal{B}^{\overline{\mathsf{Sig}}.\mathsf{Sign}}(VK)$

- set $C_1^\star \leftarrow VK$
- start $\mathcal{A}_1(PK_1, C_1^\star)$ to obtain $tag^\star$
- compute $PK, SK \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{KeyGen}(1^\lambda)$
- compute $K^\star, \overline{C^\star} \leftarrow \overline{\mathsf{TBKEM}}.\mathsf{Encaps}(PK)$
- query $sig^\star \leftarrow \overline{\mathsf{Sig}}.\mathsf{Sign}\text{-oracle}(\overline{C^\star})$
- set $C_2^\star \leftarrow \langle \overline{C^\star}, sig^\star \rangle$
- start $\mathcal{A}_2^{\mathsf{TBKEM}.\mathsf{wDecaps}}(PK, K^\star, C_2^\star)$
- answer the $\mathsf{TBKEM}.\mathsf{wDecaps}$ oracle as expected using SK
- eventually $\mathcal{A}_2$ outputs $C_2 = \langle \overline{C}, sig \rangle$—forward this value as own output

If $\mathcal{A}$ wins, then $\langle C_1^\star, C_2 \rangle = \langle VK, \overline{C}, sig \rangle$ must be pseudo-valid, i. e., the signature is valid, and $\neq \langle VK, \overline{C^\star}, sig^\star \rangle$. Consequently, $\langle \overline{C}, sig \rangle$ is a pair of message and valid signature under the verification key VK and $\neq \langle \overline{C^\star}, sig^\star \rangle$. This means that $\mathcal{B}$ wins its own attack. By security of $\overline{\mathsf{Sig}}$ it holds that

$$\mathrm{Adv}^{\mathsf{TBKEM},\mathcal{A}}_{C_2\text{-INT}}(\lambda) = \mathrm{Adv}^{\overline{\mathsf{Sig}},\mathcal{B}}_{\text{SEU-OT}}(\lambda) \leq \hat{\epsilon}(\lambda) = \epsilon_2(\lambda)$$

Finally, the strong IND-sTag-wCCA$_2$ security of TBKEM has to be shown. This is easy. For any attacker $\mathcal{A}$ of TBKEM, an attacker $\mathcal{B}$ of the (non-strong) IND-sTag-wCCA$_2$ security of $\overline{\mathsf{TBKEM}}$ can be constructed that lets most of the work be done by its own environment and only adds the signature parts. $\mathcal{B}$ works as follows:

$\mathcal{B}_1(1^\lambda)$**:**

- generate $VK, SigK \leftarrow \overline{\mathsf{Sig}}.\mathsf{KeyGen}(1^\lambda)$
- set $C_1^\star \leftarrow VK$
- start $\mathcal{A}_1(C_1^\star)$ to obtain $tag^\star$ and output this value

$\mathcal{B}_2^{\overline{\mathsf{TBKEM}}.\mathsf{wDecaps}}(PK, \mathcal{K}_b^\star, \overline{C^\star})$**:**

- compute $sig^\star \leftarrow \overline{\mathsf{Sig}}.\mathsf{Sign}(SigK, \overline{C^\star})$
- set $C_2^\star \leftarrow \langle \overline{C^\star}, sig^\star \rangle$
- start $\mathcal{A}_2^{\mathsf{TBKEM}.\mathsf{wDecaps}}(PK, K_b^\star, C_2^\star)$
  (to answer $\mathcal{A}$'s queries, $\mathcal{B}$ checks signatures and then uses its own oracle)
- output as guess whatever b' is output by $\mathcal{A}_2$

Clearly, $\mathcal{B}$ gives a perfect simulation and wins, if $\mathcal{A}$ wins. This gives

$$\mathrm{Adv}^{\mathsf{TBKEM},\mathcal{A}}_{\text{strong IND-sTag-wCCA}_2}(\lambda) = \mathrm{Adv}^{\overline{\mathsf{TBKEM}},\mathcal{B}}_{\text{IND-sTag-wCCA}_2}(\lambda) \leq \overline{\epsilon}(\lambda) = \epsilon(\lambda)$$

□

Composing both theorems gives:

Corollary 5.5. *For $\epsilon, \hat{\epsilon}$ as before it holds that:*

The (TBKEM version of) the CHK transformation is $(\epsilon + \hat{\epsilon}/2 + q_{\mathsf{Decaps}} \cdot \sqrt{\hat{\epsilon}}/2)$-IND-CCA$_2$ secure.

This is theoretically satisfying, but looser than possible (compare Thm. 3.10 on p.38). The term $q_{\mathsf{Decaps}} \cdot \sqrt{\hat{\epsilon}}$ can be replaced by $\hat{\epsilon}$, although requiring yet another change in the definition. The reason for this is that the C_1 unpredictability demands more than needed (but having the advantage of being very simple). Two changes in this definition will give the desired results: The statistical bound is replaced by a computational bound and furthermore, only the unpredictability of pseudo-valid (instead of arbitrary) ciphertexts beginning with C_1 is claimed.

Definition 5.6 (replaces the C_1-unpredictability in Def. 5.2 on p.63)**.** *The experiment of C_1-unpredictability (C_1-UNP) is defined as follows:*

$$
\begin{array}{l}
Exp^{\mathsf{TBKEM},\mathcal{A}}_{C_1\text{-}UNP}(\lambda)\text{:} \\
\quad PK, SK \leftarrow \mathsf{KeyGen}(1^\lambda) \\
\quad C_1^\star, \sigma^\star \leftarrow \mathsf{Encaps}_1(PK) \\
\quad tag^\star \leftarrow \mathcal{A}_0(C_1^\star) \\
\qquad\qquad \boxed{\mathcal{A}_1 \text{ may not access } \mathcal{A}_0\text{'s variables!}} \\
\qquad\qquad \leftarrow \mathcal{A}_1^{\mathsf{wDecaps},\mathsf{InPV}}(PK) \\
\quad \text{if the } \mathsf{InPV}\text{-oracle eventually has} \\
\qquad \text{returned true, then output win}
\end{array}
$$

The wDecaps *oracle is as before. The* InPV *on input C_1, C_2, tag works as follows:*

- *if $C_1 = C_1^\star$ and $tag = tag^\star$, then it returns whether $\langle C_1, C_2 \rangle \in pv(PK, tag)$*
- *else it returns "don't know"*

The advantage of $\mathcal{A}$ is defined as $Adv^{\mathsf{TBKEM},\mathcal{A}}_{C_1\text{-}UNP}(\lambda) := Pr[Exp^{\mathsf{TBKEM},\mathcal{A}}_{C_1\text{-}UNP}(\lambda) = win]$.

Note that $\mathcal{A}$ wins if the InPV oracle has returned *true*. This only happens if a pseudo-valid ciphertext beginning with $C_1^\star$ is submitted – i. e., $\mathcal{A}$ has "predicted" such a ciphertext. The restriction that $\mathcal{A}_1$ may not access the variables of $\mathcal{A}_0$ makes this definition of unpredictability "compatible" with the previous. With this even more generalized definition, one can prove something similar to Theorem 5.3 above, replacing $q_{\mathsf{Decaps}} \cdot \epsilon_1$ by the security bound of $\hat{\epsilon}_1 := \mathrm{Adv}^{\mathsf{TBKEM},\mathcal{A}}_{C_1\text{-UNP}}(\lambda)$. For the signature-based construction (Con. 5.1 on p.61) follows $\hat{\epsilon}_1 \leq \hat{\epsilon}$, where $\hat{\epsilon}$ bounds the security of $\overline{\mathsf{Sig}}$. Combining this new unpredictability bound with the modified theorem for the security of the signature-based construction this implies a much tighter bound of $\epsilon + \hat{\epsilon}$ for Corollary 5.5.

6 Conclusion and Future Work

This chapter briefly concludes the main results of this thesis, points out its most important contributions, and finally gives some pointers to possible future work.

6.1 Conclusion

The main goal of this thesis was to analyse the relationship between the IND-CCA$_2$ secure BMW-KEM and the IND-sID-CPA secure BB1-IBE in order to find out the specific properties of the latter that allow construction of the former. It has turned out that the key idea shows up most clearly when instead comparing BMW-KEM to an IND-sTag-wCCA$_2$ secure TBKEM version of BB1-IBE. As any IBE can be converted into such a TBKEM, this is no loss of generality.

The results of this investigation are additional structural requirements (*partitioned* TBKEM) and a strengthened version of IND-sTag-wCCA$_2$ security (*strong*-IND-sTag-wCCA$_2$) that admit the following two transformations:

The Core Transformation allows to turn any TBKEM that is *partitioned* and has *strong* IND-sTag-wCCA$_2$ security into an IND-CCA$_2$ secure KEM. The resulting KEM has the same efficiency as the original TBKEM. As expected, BMW-KEM results from BB1-TBKEM by applying this transformation.

The Hash-Based Transformation lifts any TBKEM from the standard to strong security variant. In combination with the core transformation, any *partitioned* TBKEM that has (*non-strong*) IND-sTag-wCCA$_2$ security can be turned into an IND-CCA$_2$ secure KEM. This introduces a very small computational overhead in form of a hash function and an XOR computation, but does not enlarge ciphertexts. In many cases, one can even omit the hash function.

The preconditions of both transformations are quite easy to verify and are satisfied by many schemes. Thus, the conjecture of [ACIK07] that such a transformation would not exist (or be "useless" due to strong requirements) has proven to be wrong.

The definition of partitioned TBKEM and the core transformation is quite similar to the work of [ACIK07], and thus their novelty might be questioned. But the following things, in my opinion, may be seen as significant contributions:

- the definition of strong IND-sTag-wCCA$_2$ security and moreover the observation that this gentle strengthening of the security is fulfilled by many schemes
- the security proof for the core transformation; in particular as the proof gives IND-CCA$_1$ security from weaker preconditions as a corollary

- the hash-based transformation as fallback solution if someone does not accept the reasonableness of strong security (maybe the most important contribution)

Besides these two transformations that mostly investigate BMW-KEM, a third transformation has been defined. This third transformation, in fact, is not a novel construction, but stems from the observation that a slight generalization of the definitions allows to view the CHK transformation as a composition of the core transformation and the following:

The Signature-Based Transformation uses the techniques of the CHK transformation to turn any (*non-strong*) IND-sTag-wCCA$_2$ secure (*non-partitioned*) TBKEM into one that satisfies these additional properties. As expected, its combination with the core transformation gives a TBKEM version of the CHK transformation.

This last transformation justifies the claim that BMW-KEM and the CHK transformation use the same techniques to obtain IND-CCA$_2$ security. Indeed, the core transformation may be seen as a formalization of the common underlying paradigm.

6.2 Future Work

Several questions arose during research for this thesis that seem to be worth a deeper investigation. Some of these are stated briefly in the following:

Hierarchical IBE (HIBE): In an ℓ-HIBE, users are arranged in a tree-like hierarchy of depth ℓ. Identities are vectors of up to ℓ binary strings. The techniques presented here can be generalized to convert a CPA secure $(\ell+1)$-HIBE (or ℓ-Tag-HIBE) into a CCA$_2$ secure ℓ-HIBE. In particular, it suffices if the $(\ell+1)$-th level (resp. the tag) is only sID/sTag secure. This can be used to explain the very efficient Kiltz-Galindo-IBKEM [KG06] or the compact version of Kiltz-Vahlis-IBE [KV08, §4.3.3] as an IBE (= 1-HIBE) that is constructed from a special 2-HIBE.

Observations on the Work of [ACIK07]: All transformations given in this thesis have the core transformation in common. In contrast, the two transformations of Abe et al. [ACIK07] are defined independently of each other. In the identity-based setting it seems unlikely that their chameleon-hash-based construction can be decomposed into their "core" transformation plus a new third transformation. In the tag-based setting however, this is (at least syntactically) possible. The open question is whether there exists a suitable generalization for partitioned TBKEM, similar to that which allowed the signature-based transformation. Generally, it would be nice to find more similarities between their work and mine.

Similar Security Notions: Variants of the selective-identity attack model have been suggested [CS06]. It would be interesting to investigate whether the above transformations can be transferred to this notion of security.

Bibliography

[ACG+06] Nuttapong Attrapadung, Yang Cui, David Galindo, Goichiro Hanaoka, Ichiro Hasuo, Hideki Imai, Kanta Matsuura, Peng Yang, and Rui Zhang. Relations among notions of security for identity based encryption schemes. In José R. Correa, Alejandro Hevia, and Marcos A. Kiwi, editors, *LATIN*, volume 3887 of *Lecture Notes in Computer Science*, pages 130–141. Springer, 2006.

[ACIK07] Masayuki Abe, Yang Cui, Hideki Imai, and Eike Kiltz. Efficient hybrid encryption from ID-based encryption. Technical Report 23, Cryptology ePrint Archive, 2007.

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[AGKS05] Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 128–146. Springer, 2005.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112. ACM, 1988.

[BFMLS05] K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart. Generic constructions of identity-based and certificateless KEMs. Technical Report 58, Cryptology ePrint Archive, 2005.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657. IEEE Computer Society, 2007.

[BK05] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.

[BMW05a] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 320–329. ACM, 2005.

[BMW05b] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. Technical Report 288, Cryptology ePrint Archive, 2005. Updated version of [BMW05a].

[BPR+08] Dan Boneh, Periklis Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. To appear in FOCS, 2008.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.

[CS04] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2004.

[CS06] Sanjit Chatterjee and Palash Sarkar. Generalization of the selective-id security model for hibe protocols. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 241–256. Springer, 2006.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, Nov 1976.

[DHMR07] Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. Cryptographic techniques for mobile ad-hoc networks. *Computer Networks*, 51(18):4938–4950, 2007.

[Ell87] James H. Ellis. The history of non-secret encryption. Technical report, Communication-Electronics Security Group (CESG), 1987. Kept confidentially until 1997: http://www.cesg.gov.uk/site/publications/media/ellis.pdf.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999.

[Gal06] David Galindo. A separation between selective and full-identity security notions for identity-based encryption. In Marina L. Gavrilova, Osvaldo Gervasi, Vipin Kumar, Chih Jeng Kenneth Tan, David Taniar, Antonio Laganà, Youngsong Mun, and Hyunseung Choo, editors,

ICCSA (3), volume 3982 of *Lecture Notes in Computer Science*, pages 318–326. Springer, 2006.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2006.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[KG06] Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2006.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, 2006.

[Kil07] Eike Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2007.

[KKA03] Aram Khalili, Jonathan Katz, and William A. Arbaugh. Toward secure key distribution in truly ad-hoc networks. In *SAINT Workshops*, pages 342–346. IEEE Computer Society, 2003.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[KV08] Eike Kiltz and Yevgeniy Vahlis. CCA2 secure IBE: Standard model efficiency through authenticated symmetric encryption. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 221–238. Springer, 2008.

[MBH03] Marco Casassa Mont, Pete Bramhall, and Keith Harrison. A flexible role-based secure messaging service: Exploiting IBE technology for privacy in health care. In *DEXA Workshops*, pages 432–437. IEEE Computer Society, 2003.

[MRY04] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 171–190. Springer, 2004.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43. ACM, 1989.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. ACM, 1990.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[Sho02] Victor Shoup. OAEP reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

[SK03] Ryuichi Sakai and Masao Kasahara. ID based cryptosystems with pairing on elliptic curve. Technical Report 54, Cryptology ePrint Archive, 2003.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[Zha07] Rui Zhang. Tweaking TBE/IBE to PKE transforms with chameleon hash functions. In Jonathan Katz and Moti Yung, editors, *ACNS*, volume 4521 of *Lecture Notes in Computer Science*, pages 323–339. Springer, 2007.