On the Joint Security of Encryption and Signature, Revisited

Kenneth G. Paterson[1]*, Jacob C.N. Schuldt[2]**, Martijn Stam[3], and Susan Thomson[1]***

[1] Royal Holloway, University of London
[2] Research Center for Information Security, AIST, Japan
[3] University of Bristol

Abstract. We revisit the topic of joint security for combined public key schemes, wherein a single keypair is used for both encryption and signature primitives in a secure manner. While breaking the principle of key separation, such schemes have attractive properties and are sometimes used in practice. We give a general construction for a combined public key scheme having joint security that uses IBE as a component and that works in the standard model. We provide a more efficient direct construction, also in the standard model. We then consider the problem of how to build signcryption schemes from jointly secure combined public key schemes. We provide a construction that uses any such scheme to produce a triple of schemes – signature, encryption, and signcryption – that are jointly secure in an appropriate and strong security model.

1 Introduction

Key separation versus key reuse: The folklore principle of key separation dictates using different keys for different cryptographic operations. While this is well-motivated by real-world, security engineering concerns, there are still situations where it is desirable to use the same key for multiple operations [19]. In the context of public key cryptography, using the same keypair for both encryption and signature primitives can reduce storage requirements (for certificates as well as keys), reduce the cost of key certification and the time taken to verify certificates, and reduce the footprint of cryptographic code. These savings may be critical in embedded systems and low-end smart card applications. As a prime example, the globally-deployed EMV standard for authenticating credit and debit card transactions allows the same keypair to be reused for encryption and signatures for precisely these reasons [15].

However, this approach of reusing keys is not without its problems. For example, there is the issue that encryption and signature keypairs may have different lifetimes, or that the private keys may require different levels of protection [19]. Most importantly of all, there is the question of whether it is *secure* to use the same keypair in two (or more) different primitives – perhaps the two uses will interact with one another badly, in such a way as to undermine the security of one or both of the primitives. In the case of textbook RSA, it is obvious that using the same keypair for decryption and signing is dangerous, since the signing and decryption functions are so closely related in this case. Security issues may still arise even if some standardized padding is used prior to encryption and signing [26]. In Section 3 we will provide another example in the context of encryption and signature primitives, where the individual components are secure (according to the usual notions of security for encryption and signature) but become completely insecure as soon as they are used in combination with one another. At the protocol level, Kelsey, Schneier and Wagner [24] gave examples of protocols that are individually secure, but that interact badly when a keypair is shared between them.

The formal study of the security of key reuse was initiated by Haber and Pinkas [19]. They introduced the concept of a *combined public key scheme*. Here, an encryption scheme and signature scheme are combined: the existing algorithms to encrypt, decrypt, sign and verify are preserved, but the two key generation algorithms are modified to produce a single algorithm. This algorithm outputs two keypairs, one for the encryption scheme and one for the signature scheme, with the keypairs no longer necessarily being independent. Indeed, under certain conditions, the two keypairs may be identical, in which case the savings described above may be realised. In other cases, the keypairs are not identical but can have some shared components, leading to more modest savings. Haber and Pinkas also introduced the natural

* This author is supported by EPSRC Leadership Fellowship EP/H005455/1.
** This author is supported by a JSPS Fellowship for Young Scientists.
*** This author is supported by the EPSRC through Leadership Fellowship EP/H005455/1.

security model for combined public key schemes, where the adversary against the encryption part of the scheme is equipped with a signature oracle in addition to the usual decryption oracle, and where the adversary against the signature part of the scheme is given a decryption oracle in addition to the usual signature oracle. In this setting, we talk about the *joint security* of the combined scheme.

Setting a benchmark: As we shall see in Section 3, there is a trivial "Cartesian product" construction for a combined public key scheme with joint security. The construction uses arbitrary encryption and signature schemes as components, and the combined scheme's keypair is just a pair of vectors whose components are the public/private keys of the component schemes. Thus the Cartesian product construction merely formalises the principle of key separation. This construction, while extremely simple, provides a benchmark by which other constructions can be judged. For example, if the objective is to minimise the public key size in a combined scheme, then any construction should aim to have shorter keys than can be obtained by instantiating the Cartesian product construction with the best available encryption and signature schemes.

Re-evaluating Haber-Pinkas: In this respect, we note that, while Haber and Pinkas considered various well-known concrete schemes and conditions under which their keys could be partially shared, none of their examples having provable security in the standard model lead to *identical* keypairs for both signature and encryption. Indeed, while the approach of Haber and Pinkas can be made to work in the random oracle model by careful oracle programming and domain separation, their approach does not naturally extend to the standard model. More specifically, in their approach, to be able to simulate the signing oracle in the IND-CCA security game, the public key of the combined scheme cannot be exactly the same as the public key of the underlying encryption scheme (otherwise, successful simulation would lead to a signature forgery). This makes it hard to achieve full effective overlap between the public keys for signing and encryption. For the (standard model) schemes considered by Haber and Pinkas this results in the requirements that part of the public key be specific to the encryption scheme and that another part of it be specific to the signature scheme. Furthermore, at the time of publication of [19] only a few secure (IND-CCA2, resp. EUF-CMA) and efficient standard-model schemes were known. Consequently, no "compatible" signature and encryption schemes were identified in [19] for the standard model.

Combined schemes from trapdoor permutations: The special case of combined schemes built from trapdoor permutations was considered in [10, 27]. Here, both sets of authors considered the use of various message padding schemes in conjunction with an arbitrary trapdoor permutation to build combined public key schemes having joint security. Specifically, Coron et al. [10] considered the case of PSS-R encoding, while Komano and Ohta [27] considered the cases of OAEP+ and REACT encodings. All of the results in these two papers are in the random oracle model.

In further related, but distinct, work, Dodis et al. [14] (see also [13]) considered the use of message padding schemes and trapdoor permutations to build *signcryption* schemes. These offer the combined security properties of signature and encryption in a single cryptographic transform (as opposed to the notion of a combined public key scheme which offers these security properties separately, but with a common key generation algorithm). Dodis et al. showed, again in the random oracle model, how to build efficient, secure signcryption schemes in which each user's keypair, specifying a permutation and its trapdoor, is used for both signing and encryption purposes. They were careful to point out that the previous results of [10, 27] concerning combined public key schemes do not immediately imply similar results in the more complicated signcryption setting, nor can they be immediately applied to construct secure signcryption schemes via the generic composition methods of Dodis et al. [3].

1.1 Our Contribution

We focus on the problem of how to construct combined public key schemes which are jointly secure in the standard model, a problem for which, as we have explained above, there currently exist no fully satisfactory solutions. Naturally, for reasons of practical efficiency, we are interested in minimising the size of keys (both public and private), ciphertexts, and signatures in such schemes. The complexity of the various algorithms needed to implement the schemes will also be an important consideration.

As a warm-up, in Section 3, we give the simple Cartesian product construction, as well as a construction showing that the general problem is not vacuous (i.e. that there exist insecure combined schemes whose component schemes are secure when used in isolation).

We then present in Section 4 a construction for a combined public key scheme using an IBE scheme as a component. The trick here is to use the IBE scheme in the Naor transform and the CHK transform *simultaneously* to create a combined public key scheme that is jointly secure, under rather weak requirements on the starting IBE scheme (specifically, the IBE scheme needs to be OW-ID-CPA and IND-sID-CPA secure). This construction extends easily to the (hierarchical) identity-based setting. Instantiating this construction using standard model secure IBE schemes from the literature already yields rather efficient combined schemes. For example, using an asymmetric pairing version of Gentry's IBE scheme [18], we can achieve a combined scheme in which, at the 128-bit security level, the public key size is 1536 bits, the signature size is 768 bits and the ciphertext size is 2304 bits (plus the size of a signature and a verification key for a one-time signature scheme), with joint security being based on a q-type assumption. This is already competitive with schemes arising from the Cartesian product construction.

We then provide a more efficient direct construction for a combined scheme with joint security in Section 5. This construction is based on the signature scheme of Boneh and Boyen [6] and a KEM obtained by applying the techniques by Boyen, Mei and Waters [9] to the second IBE scheme of Boneh and Boyen in [5]. At the 128-bit security level, it enjoys public keys that consist of 1280 bits, signatures that are 768 bits and a ciphertext overhead of just 512 bits. The signatures can be shrunk at the cost of increasing the public key size.

The remainder of the paper then concerns the applications of our ideas to signcryption. We show in Section 7 that a combined public key scheme can be used to construct a signcryption scheme, using a "sign-then-encrypt" construction, that is secure in the strongest security model for signcryption (achieving insider confidentiality and insider unforgeability in the multi-user setting). For technical reasons, we need a tag-based encryption scheme for this construction, so our earlier focus is also on this extended type of encryption scheme. Instantiating this construction with our concrete combined public key scheme effectively solves the challenge implicitly laid down by Dodis *et al.* in [13], to construct an efficient standard model signcryption scheme in which a single short keypair can securely be used for both sender and receiver functions. Furthermore, we are able to show that the signcryption scheme we obtain is jointly secure (in an appropriate sense, to be made precise in Section 7) when used in combination with *both* its signature and encryption components. Thus we are able to obtain a triple of functionalities (signcryption, signature, encryption) which are jointly secure using only a single keypair.

1.2 Further Related Work

Further work on combined public key schemes in the random oracle model, for both the normal public key setting and the identity-based setting can be found in [38]. In particular, it is proved that the identity-based signature scheme of Hess [22] and Boneh and Franklin's identity-based encryption scheme [8] can be used safely together.

The topic of joint security of combined public key schemes is somewhat linked to the topic of cryptographic agility [2], which considers security when the same key (or key pair) is used simultaneously in multiple instantiations of the *same* cryptographic primitive. This contrasts with joint security, where we are concerned with security when the same key pair is used simultaneously in instantiations of *different* cryptographic primitives. The connections between these different but evidently related topics remain to be explored.

Examples of signcryption schemes making use of a single keypair are known [29–32], but these are all in the random oracle model.

2 Preliminaries

In our constructions, we will make use of a number of standard primitives, including digital signatures, (tag-based) public key encryption, identity-based encryption (IBE), a data encapsulation mechanism (DEM), and an always second-preimage resistant hash function. The definitions and security notions for these primitives are given in Appendix A. In the following, we briefly recall the properties of bilinear pairings as well as define the computational assumptions which we will make use of to prove the security of our concrete constructions.

Bilinear Pairings: Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, $\mathbb{G}_T$ be groups of prime order p. A pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies the following properties:

1. Bilinear: For all $a, b \in \mathbb{Z}, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2. Non-degenerate: $e(g_1, g_2) \neq 1$.
3. Computable: There is an efficient algorithm to compute the map e.

Note that we work exclusively in the setting of asymmetric pairings, whereas schemes are often presented in the naive setting of symmetric pairings $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. At higher security levels (128 bits and above), asymmetric pairings are far more efficient both in terms of computation and in terms of the size of group elements [17]. As a concrete example, using BN curves [4] and sextic twists, we can attain the 128-bit security level with elements of $\mathbb{G}_1$ being represented by 256 bits and elements of $\mathbb{G}_2$ needing 512 bits. By exploiting compression techniques [37], elements of $\mathbb{G}_T$ in this case can be represented using 1024 bits. For further details on parameter selection for pairings, see [16].

We will make use of the following assumptions in proving the security of our concrete constructions.

Strong Diffie-Hellman (SDH) assumption [6]: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p, respectively generated by g_1 and g_2. In the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, the q-SDH problem is stated as follows:

> Given as input a $(q + 3)$-tuple of elements
> $\left(g_1, g_1^x, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}\right) \in \mathbb{G}_1^2 \times \mathbb{G}_2^{q+1}$
> output a pair $\left(c, g_2^{1/(x+c)}\right) \in \mathbb{Z}_p \times \mathbb{G}_2$ for a freely chosen value $c \in \mathbb{Z}_p \backslash \{-x\}$.

An algorithm $\mathcal{A}$ solves the q-SDH problem in the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage ϵ if

$$\Pr\left[\mathcal{A}\left(g_1, g_1^x, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}\right) = \left(c, g_2^{1/(x+c)}\right)\right] \geq \epsilon,$$

where the probability is over the random choice of generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the random choice of $x \in \mathbb{Z}_p^*$, and the random bits consumed by $\mathcal{A}$. We say that the (t, q, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

Decisional Bilinear Diffie-Hellman Inversion (DBDHI) assumption [5]: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p, respectively generated by g_1 and g_2. In the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, the q-DBDHI problem is stated as follows:

> Given as input a $(q + 4)$-tuple of elements
> $\left(g_1, g_1^x, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}, T\right) \in \mathbb{G}_1^2 \times \mathbb{G}_2^{q+1} \times \mathbb{G}_T$
> output 0 if $T = e(g_1, g_2)^{1/x}$ or 1 if T is a random element in $\mathbb{G}_T$.

An algorithm $\mathcal{A}$ solves the q-DBDHI problem in the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage ϵ if

$$\left|\Pr\left[\mathcal{A}\left(g_1, g_1^x, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}, e(g_1, g_2)^{1/x}\right) = 0\right] - \Pr\left[\mathcal{A}\left(g_1, g_1^x, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}, T\right) = 0\right]\right| \geq \epsilon,$$

where the probability is over the random choice of generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the random choice of $x \in \mathbb{Z}_p^*$, the random choice of $T \in \mathbb{G}_T$, and the random bits consumed by $\mathcal{A}$. We say that the (t, q, ϵ)-DBDHI assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ϵ in solving the q-DBDHI problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

3 Combined Signature and Encryption Schemes

A combined signature and encryption scheme is a combination of a signature scheme and a public key encryption scheme that share a key generation algorithm and hence a keypair (pk, sk). It comprises a tuple of algorithms (KeyGen, Sign, Verify, Encrypt, Decrypt) such that (KeyGen, Sign, Verify) form a signature scheme and (KeyGen, Encrypt, Decrypt) form a PKE scheme. Since the signature and PKE schemes share a keypair the standard notions of EUF-CMA and IND-CCA security need to be extended to reflect an adversary's ability to request both signatures and decryptions under the challenge public key. When defining a security game against a component of the scheme the nature of any additional oracles depends on the required security of the other components. For example, if EUF-CMA security of the signature component of a combined signature and encryption scheme is required, then it is necessary to provide the adversary with unrestricted access to a signature oracle when proving IND-CCA security of the encryption component of the scheme. The security definitions given implicitly in [10], considering IND-CCA security of the encryption component and EUF-CMA security of the signature component, are stated formally here.

EUF-CMA security in the presence of a decryption oracle: Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a combined signature and encryption scheme. Existential unforgeability of the signature component under an adaptive chosen message attack in the presence of an additional decryption oracle is defined through the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger generates a keypair $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ and gives $\mathcal{A}$ the challenge public key pk.

Query phase: $\mathcal{A}$ requests signatures on messages m_i of its choice. The challenger responds to each signature query with a signature $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$. $\mathcal{A}$ also requests decryptions of ciphertexts c_i of its choice. The challenger responds to each decryption query with a message $m \leftarrow \mathsf{Decrypt}(sk, c_i)$ or a failure symbol $\perp$.

Forgery: $\mathcal{A}$ outputs a message signature pair (σ, m) such that m was not submitted to the signing oracle, and wins the game if $\mathsf{Verify}(pk, \sigma, m) = 1$.

The advantage of an adversary $\mathcal{A}$ is the probability it wins the above game.

A forger $\mathcal{A}$ (t, q_d, q_s, ϵ)-breaks the signature component of a combined signature and encryption scheme if $\mathcal{A}$ runs in time at most t, makes at most q_d decryption queries and q_s signature queries and has advantage at least ϵ. The signature component of a combined signature and encryption scheme is said to be (t, q_d, q_s, ϵ)-EUF-CMA secure in the presence of a decryption oracle if no forger (t, q_d, q_s, ϵ)-breaks it.

IND-CCA security in the presence of a signing oracle: Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a combined signature and encryption scheme. Indistinguishability of the encryption component under an adaptive chosen ciphertext attack in the presence of an additional signing oracle is defined through the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger generates a keypair $(pk, sk) \leftarrow \mathsf{Keyen}(1^k)$ and gives $\mathcal{A}$ the challenge public key pk.

Phase 1: $\mathcal{A}$ requests decryptions of ciphertexts c_i of its choice. The challenger responds to each decryption query with a message $m \leftarrow \mathsf{Decrypt}(sk, c_i)$ or a failure symbol $\perp$. $\mathcal{A}$ also requests signatures on messages m_i of its choice. The challenger responds to each signature query with a signature $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$.

Challenge: $\mathcal{A}$ chooses two equal length messages m_0, m_1. The challenger chooses a random bit b, computes $c^* \leftarrow \mathsf{Encrypt}(pk, m_b)$, and passes c^* to the adversary.

Phase 2: As Phase 1 but with the restriction that $\mathcal{A}$ must not request the decryption of the challenge ciphertext c^*.

Guess: $\mathcal{A}$ outputs a guess b' for b.

The advantage of $\mathcal{A}$ is $\left|\Pr[b' = b] - \frac{1}{2}\right|$.

An adversary $\mathcal{A}$ (t, q_d, q_s, ϵ)-breaks the encryption component of a combined signature and encryption scheme if $\mathcal{A}$ runs in time at most t, makes at most q_d decryption queries and q_s signature queries and has advantage at least ϵ. The encryption component of a combined signature and encryption scheme is said to be (t, q_d, q_s, ϵ)-IND-CCA secure in the presence of a signing oracle if no adversary (t, q_d, q_s, ϵ)-breaks it.

Informally, we say that a combined scheme is *jointly secure* if it is both EUF-CMA secure in the presence of a decryption oracle and IND-CCA secure in the presence of a signing oracle.

In addition to a combined public key scheme defined as above, we will consider a scheme in which the encryption component corresponds to a tag-based encryption scheme (see Appendix A.2). The corresponding security notions, which we will refer to as *EUF-CMA security in the presence of a tag-based* *decryption oracle* and *IND-tag-CCA security in the presence of a signing oracle*, are simple extensions of the above definitions in which the decryption oracle takes both a ciphertext c and a tag t as input. Furthermore, in the challenge phase of the IND-tag-CCA security in the presence of a signing oracle, the adversary $\mathcal{A}$ will submit challenge messages m_0, m_1 and a challenge tag t^*, and is not allowed to submit (c^*, t^*) to the decryption oracle in Phase 2. This type of combined public key scheme will be used in the construction of a joint signcryption, signature and encryption scheme presented in Section 7.

3.1 A Cartesian Product Construction

A trivial way of obtaining a system satisfying the above security properties is to concatenate the keys of an encryption scheme and signature scheme, then use the appropriate component of the compound key for each operation. This gives a combined signature and encryption scheme where the signature and encryption operations are essentially independent. Consequently their respective security properties are retained in the presence of the additional oracle. This simple construction sets a benchmark in terms of key size and other performance measures that any bespoke construction should best in one or more metrics.

Formally, let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Verify})$ be a signature scheme, and let $\mathcal{E} = (\mathcal{E}.\mathsf{KeyGen}, \mathcal{E}.\mathsf{Encrypt}, \mathcal{E}.\mathsf{Decrypt})$ be an encryption scheme. Then the Cartesian product combined signature and encryption scheme $\mathsf{CartCSE}(\mathcal{E}, \mathcal{S})$ is constructed as follows:

$\mathsf{CartCSE}(\mathcal{E}, \mathcal{S}).\mathsf{KeyGen}(1^k)$**:** Run $\mathcal{S}.\mathsf{KeyGen}(1^k)$ to get (pk_s, sk_s). Run $\mathcal{E}.\mathsf{KeyGen}(1^k)$ to get (pk_e, sk_e). Output the public key $pk = (pk_s, pk_e)$ and the private key $sk = (sk_s, sk_e)$.

$\mathsf{CartCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Sign}(sk, m)$**:** Output $\mathcal{S}.\mathsf{Sign}(sk_s, m)$.

$\mathsf{CartCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Verify}(pk, \sigma, m)$**:** Output $\mathcal{S}.\mathsf{Verify}(pk_s, \sigma, m)$.

$\mathsf{CartCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Encrypt}(pk, m)$**:** Output $\mathcal{E}.\mathsf{Encrypt}(pk_e, m)$.

$\mathsf{CartCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Decrypt}(sk, c)$**:** Output $\mathcal{E}.\mathsf{Decrypt}(sk_e, c)$.

We omit the straightforward proof that this scheme is jointly secure if $\mathcal{S}$ is EUF-CMA secure and $\mathcal{E}$ is IND-CCA secure.

3.2 An Insecure CSE Scheme whose Components are Secure

To show that the definitions are not trivially satisfied, we give a pathological example to show that a PKE scheme and a signature scheme that are individually secure may not be secure when used in combination. Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Verify})$ be an EUF-CMA secure signature scheme, and let $\mathcal{E} = (\mathcal{E}.\mathsf{KeyGen}, \mathcal{E}.\mathsf{Encrypt}, \mathcal{E}.\mathsf{Decrypt})$ be an IND-CCA secure encryption scheme. A combined signature and encryption scheme $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S})$ can be constructed as follows.

$\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{KeyGen}(1^k)$**:** Run $\mathcal{S}.\mathsf{KeyGen}(1^k)$ to get (pk_s, sk_s). Run $\mathcal{E}.\mathsf{KeyGen}(1^k)$ to get (pk_e, sk_e). Output the public key $pk = (pk_s, pk_e)$ and the private key $sk = (sk_s, sk_e)$.

$\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Sign}(sk, m)$**:** Compute $\sigma' = \mathcal{S}.\mathsf{Sign}(sk_s, m)$. Output $\sigma = \sigma' || sk_e$.

$\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Verify}(pk, \sigma, m)$**:** Parse σ as $\sigma' || sk_e$. Run $\mathcal{S}.\mathsf{Verify}(pk_s, \sigma', m)$ and output the result.

$\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Encrypt}(pk, m)$**:** Output $c = \mathcal{E}.\mathsf{Encrypt}(pk_e, m)$.

$\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Decrypt}(sk, c)$**:** Run $\mathcal{E}.\mathsf{Decrypt}(sk_e, c)$. If this decryption is successful, output the decrypted message. Otherwise (if $\perp$ was returned), output sk_s.

From the security of the base schemes it is easy to see that the signature scheme given by the algorithms $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{KeyGen}$, $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Sign}$, $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Verify}$ is EUF-CMA secure, and the PKE scheme with algorithms $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{KeyGen}$, $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Encrypt}$, $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S}).\mathsf{Decrypt}$ is IND-CCA secure. However when key generation is shared a single signature reveals the PKE scheme's private key, and the decryption of a badly formed ciphertext reveals the private key of the signature scheme. Thus $\mathsf{BadCSE}(\mathcal{E}, \mathcal{S})$ is totally insecure, even though its component schemes are secure.

```
$\mathsf{CSE}(\mathcal{I}).\mathsf{KeyGen}(1^k)$:
    $(mpk, msk) \leftarrow \mathcal{I}.\mathsf{Setup}(1^k)$
    $(pk, sk) = (mpk, msk)$
    return $(pk, sk)$

$\mathsf{CSE}(\mathcal{I}).\mathsf{Sign}(sk, m)$:
    $ID = 0||m$
    $\sigma \leftarrow \mathcal{I}.\mathsf{Extract}(sk, ID)$
    return $\sigma$

$\mathsf{CSE}(\mathcal{I}).\mathsf{Verify}(pk, \sigma, m)$:
    $ID = 0||m$
    $x \leftarrow_R \mathcal{M}$
    $c \leftarrow \mathcal{I}.\mathsf{Encrypt}(pk, ID, x)$
    if $\mathcal{I}.\mathsf{Decrypt}(pk, \sigma, c) = x$
    then return $1$
    else return $0$

$\mathsf{CSE}(\mathcal{I}).\mathsf{Encrypt}(pk, t, m)$:
    $(vk, sk') \leftarrow \mathcal{OT}.\mathsf{KeyGen}$
    $ID = 1||vk$
    $c' \leftarrow \mathcal{I}.\mathsf{Encrypt}(pk, ID, m)$
    $\sigma \leftarrow \mathcal{OT}.\mathsf{Sign}(sk', c'||t)$
    return $(vk, \sigma, c')$

$\mathsf{CSE}(\mathcal{I}).\mathsf{Decrypt}(sk, t, c)$:
    Parse $c$ as $(vk, \sigma, c')$
    if $\mathcal{OT}.\mathsf{Verify}(vk, \sigma, c'||t) = 1$
    then $ID = 1||vk$
        $sk_{ID} \leftarrow \mathcal{I}.\mathsf{Extract}(sk, ID)$
        return $\mathcal{I}.\mathsf{Decrypt}(pk, sk_{ID}, c')$
    else return $\perp$
```

Fig. 1. Generic construction from IBE

4 A Generic Construction from IBE

We show how to build a combined signature and encryption scheme from an IBE scheme $\mathcal{I}$ with algorithms $\mathcal{I}.\mathsf{Setup}$, $\mathcal{I}.\mathsf{Extract}$, $\mathcal{I}.\mathsf{Encrypt}$, $\mathcal{I}.\mathsf{Decrypt}$. We make use of a one time strongly secure signature scheme $\mathcal{OT}$ with algorithms $\mathcal{OT}.\mathsf{KeyGen}$, $\mathcal{OT}.\mathsf{Sign}(sk, m)$, $\mathcal{OT}.\mathsf{Verify}(pk, \sigma, m)$. The construction is particularly simple: the signature scheme component is constructed through the Naor transform [8] and the PKE scheme component through a tag-based version of the CHK transform [7]. Since in the Naor construction signatures are just private keys from the IBE scheme, and these private keys can be used to decrypt ciphertexts in the PKE scheme resulting from the CHK transform, we use a bit prefix in the identity space to provide domain separation between the signatures and private keys.

We assume $\mathcal{I}$ has message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and identity space $\{0,1\}^{n+1}$, and that $\mathcal{OT}$ has public key space $\{0,1\}^n$. Then the signature scheme component of $\mathsf{CSE}(\mathcal{I})$ has message space $\{0,1\}^n$ but can be extended to messages of arbitrary length through the use of a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^n$. The PKE component of $\mathsf{CSE}(\mathcal{I})$ has message space $\mathcal{M}$. The algorithms of $\mathsf{CSE}(\mathcal{I})$ are shown in Figure 1.

Theorem 1 *Let $\mathcal{I}$ be a (t', q, ϵ)-OW-ID-CPA secure IBE scheme. Then the signature component of $\mathsf{CSE}(\mathcal{I})$ is (t, q_d, q_s, ϵ)-EUF-CMA secure in the presence of a tag-based decryption oracle provided that*

$$q_s + q_d \leq q \qquad \text{and} \qquad t \leq t' - q_d(T_v + T_d) - T_d,$$

where T_v is the maximum time for a verification in $\mathcal{OT}$ and T_d is the maximum time for a decryption in $\mathcal{I}$.

Proof of Theorem 1. Suppose there exists a forger $\mathcal{F}$ that (t, q_d, q_s, ϵ) breaks the EUF-CMA security of the signature component of $\mathsf{CSE}(\mathcal{I})$ in the presence of a decryption oracle. We construct an algorithm $\mathcal{A}$ that interacts with the forger $\mathcal{F}$ to (t', q, ϵ)-OW-ID-CPA break the IBE scheme $\mathcal{I}$.

Setup: $\mathcal{A}$ is given a master public key mpk which it gives to $\mathcal{F}$ as the public key.

Signing queries: In response to a request for a signature on message m, $\mathcal{A}$ queries its extraction oracle for the identity $ID = 0||m$ to obtain sk_{ID} which it returns to $\mathcal{F}$ as the signature.

Decryption queries: In response to a decryption query for a ciphertext $c = (vk, \sigma, c')$ with tag t, $\mathcal{A}$ verifies that σ is a valid signature on $c'||t$ with verification key vk. If it is not a valid signature, $\mathcal{A}$

returns $\perp$. If the signature is valid, $\mathcal{A}$ queries its extraction oracle for the identity $ID = 1||vk$ to obtain sk_{ID} which it uses to decrypt c', returning the output of the decryption operation as the result of the decryption query.

Forgery: Eventually $\mathcal{F}$ will return a forgery (σ^*, m^*) on a message m^* for which a signing query was not made. At this point $\mathcal{A}$ outputs $ID^* = 0||m^*$ as the target identity. This is a valid choice; since a signing query was not made for message m^* an extraction query was not made for $ID = 0||m^*$.

Challenge: $\mathcal{A}$ receives a ciphertext c^*, which is the encryption of a random message m for identity ID^*. If σ^* is a valid signature for message m^* then σ^* is a valid decryption key for identity ID^*. This allows $\mathcal{A}$ to decrypt c^* using $sk_{ID^*} = \sigma^*$ to retrieve the message m which it subsequently outputs.

$\mathcal{A}$ succeeds precisely when $\mathcal{F}$ succeeds, so if $\mathcal{F}$ outputs a valid forgery with probability ϵ in time t then algorithm $\mathcal{A}$ succeeds in time at most $t + q_d(T_v + T_d) + T_d$ with the same probability ϵ.

Theorem 2 *Let $\mathcal{I}$ be an (t_i, q_i, ϵ_i)-IND-sID-CPA secure IBE scheme and let $\mathcal{OT}$ be a (t_s, ϵ_s)-strongly unforgeable one time signature scheme. Then the encryption component of* $\mathsf{CSE}(\mathcal{I})$ *is (t, q_d, q_s, ϵ)-IND-tag-CCA secure in the presence of a signing oracle provided that*

$$\epsilon > \frac{1}{2}\epsilon_s + \epsilon_i, \quad q_s + q_d < q_i, \quad \textit{and} \quad t < t_i - T_{kg} - T_{sig} - q_d(T_v + T_d),$$

where T_{kg}, T_{sig} and T_v are the maximum times for key generation, signing and verifying respectively in $\mathcal{OT}$, and T_d is the maximum decryption time in $\mathcal{I}$.

Proof of Theorem 2. The proof follows closely that of Theorem 1 in [7]. Let $\mathcal{D}$ be an adversary against the IND-CCA security of the encryption component of $\mathsf{CSE}(\mathcal{I})$ in the presence of a signing oracle running in time at most t and making at most q_s signature queries and q_d decryption queries. We use $\mathcal{D}$ to build an IND-sID-CPA adversary $\mathcal{B}$ against $\mathcal{I}$ as follows.

Setup: $\mathcal{B}$ runs $\mathcal{OT}.\mathsf{KeyGen}$ to obtain a keypair (vk^*, sk^*) then submits $ID^* = 1||vk^*$ as the target identity. $\mathcal{B}$ is then given master public key mpk which it gives to $\mathcal{D}$ as the challenge public key.

Decryption queries: We partition the decryption queries into three possible cases and show how $\mathcal{B}$ responds to each case. Suppose the query is for ciphertext (vk, σ, c') with tag t, and let $\mathcal{OT}.\mathsf{Verify}(vk, \sigma, c'||t) = validity$.

Case 1: $vk = vk^*$
If $validity = 0$ then $\mathcal{B}$ responds to the decryption query with $\perp$. If $validity = 1$ then a forgery has been made against $\mathcal{OT}$, call this event Forge. If Forge occurs, $\mathcal{B}$ aborts and outputs a random bit b'.

Case 2: $vk \neq vk^*$ and $validity = 0$
$\mathcal{B}$ responds to the decryption query with $\perp$.

Case 3: $vk \neq vk^*$ and $validity = 1$
$\mathcal{B}$ queries the extraction oracle for identity $ID = 1||vk$ to obtain sk_{ID}, then uses sk_{ID} to decrypt c', responding to the decryption query with the output of the decryption operation.

Signature queries: In response to a signature query for message m, $\mathcal{B}$ queries its extraction oracle for identity $ID = 0||m$ to obtain sk_{ID} which it returns as the signature.

Challenge: Eventually $\mathcal{D}$ will output a pair of messages m_0, m_1 and a tag t^*. $\mathcal{B}$ forwards these messages and receives a challenge ciphertext c^*. $\mathcal{B}$ calls $\mathcal{OT}.\mathsf{Sign}(sk^*, c^*||t^*)$ to obtain σ^* and sends $C = (vk^*, \sigma^*, c^*)$ to $\mathcal{D}$. $\mathcal{D}$ may make more signature and decryption queries under the restriction that it must not submit to the decryption oracle its challenge ciphertext C with tag t^*. $\mathcal{D}$ then submits a guess b' which $\mathcal{B}$ outputs as its guess.

$\mathcal{B}$ represents a legal strategy for attacking $\mathcal{I}$, in particular $\mathcal{B}$ never requests the private key corresponding to the target identity ID^*. Provided Forge does not occur, $\mathcal{B}$ provides a perfect simulation for $\mathcal{D}$ so $\mathcal{B}$ succeeds with the same probability as $\mathcal{D}$. If Forge does occur then $\mathcal{B}$ outputs a random bit and succeeds with probability $\frac{1}{2}$. Letting $\Pr^{\mathcal{B}}_{\mathrm{IBE}}[\mathsf{Succ}]$ denote the probability of $\mathcal{B}$ outputting the correct bit in the IBE security game and $\Pr^{\mathcal{D}}_{\mathrm{PKE}}[\mathsf{Succ}]$ denote the probability of $\mathcal{D}$ outputting the correct bit in the PKE security game, it can be seen that

$$\left|\Pr^{\mathcal{D}}_{\mathrm{PKE}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \frac{1}{2}\Pr^{\mathcal{D}}_{\mathrm{PKE}}[\mathsf{Forge}] - \frac{1}{2}\right| = \left|\Pr^{\mathcal{B}}_{\mathrm{IBE}}[\mathsf{Succ}] - \frac{1}{2}\right|.$$

Since $\mathcal{I}$ is an (t_i, q_i, ϵ_i)-IND-sID-CPA secure IBE scheme, $\left|\Pr^{\mathcal{B}}_{\text{IBE}}[\mathsf{Succ}] - \frac{1}{2}\right| < \epsilon_i$. The event Forge represents a signature forgery against $\mathcal{OT}$, so $\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}] < \epsilon_s$. It follows that

$$\begin{aligned}
\epsilon &= \left|\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Succ}] - \frac{1}{2}\right| \\
&= \left|\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Succ} \wedge \mathsf{Forge}] + \Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] - \frac{1}{2}\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}] + \frac{1}{2}\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}] - \frac{1}{2}\right| \\
&\leq \left|\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Succ} \wedge \mathsf{Forge}] - \frac{1}{2}\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}]\right| + \left|\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \frac{1}{2}\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}] - \frac{1}{2}\right| \\
&\leq \frac{1}{2}\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}] + \left|\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \frac{1}{2}\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}] - \frac{1}{2}\right| \\
&= \frac{1}{2}\Pr^{\mathcal{D}}_{\text{PKE}}[\mathsf{Forge}] + \left|\Pr^{\mathcal{B}}_{\text{IBE}}[\mathsf{Succ}] - \frac{1}{2}\right| \\
&\leq \frac{1}{2}\epsilon_s + \epsilon_i.
\end{aligned}$$

The running time of $\mathcal{B}$ is at most $t + T_{kg} + q_d(T_v + T_d) + T_{sig}$, and it asks at most $q_s + q_d$ private key extraction queries, so the theorem holds. □

IBE schemes meeting the standard model security requirements include those of Gentry [18] and Waters [39]. The latter results in a large public key ($n+3$ group elements), though this could be reduced in practice by generating most of the elements from a seed in a pseudo-random manner. We focus on the instantiation of our construction using Gentry's scheme. This scheme was originally presented in the setting of symmetric pairings. When we translate it to the asymmetric setting (see Appendix C for details) and apply our construction at the 128-bit security level using BN curves with sextic twists, we obtain a combined public key scheme in which the public key consists of two elements of $\mathbb{G}_1$ and two elements of $\mathbb{G}_2$, giving a public key size of 1536 bits. Ciphertexts encrypt elements of $\mathbb{G}_T$ and consist of an element of $\mathbb{G}_1$, two elements of $\mathbb{G}_T$, and a verification key and signature from $\mathcal{OT}$, so are 2304 bits plus the bit length of a verification key and signature in $\mathcal{OT}$. Signatures consist of an element of $\mathbb{Z}_p$ and an element of $\mathbb{G}_2$, so are 768 bits in size. Here we assume that descriptions of groups and pairings are domain parameters that are omitted from our key size calculations. The security of this scheme depends on an assumption closely related to the decisional q-augmented bilinear Diffie-Hellman exponent assumption.

This construction could be improved further using the Boneh-Katz [7] alternative to the CHK transform. We omit the details in favour of our next scheme.

5 A More Efficient Construction

The following scheme is based on the signature scheme by Boneh and Boyen [6] and a KEM obtained by applying the techniques by Boyen, Mei and Waters [9] to the second IBE scheme by Boneh and Boyen in [5]. The schemes make use of a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where the groups are of order p, and the KEM furthermore makes use of an always second-preimage resistant (aSec-secure) hash function $\mathsf{H} : \mathbb{G}_1 \to \{0,1\}^{n-1}$ where $2^n < p$. To obtain a full encryption scheme, the KEM is combined with a DEM, and we assume for simplicity that the key space of the DEM is $\mathcal{K} = \mathbb{G}_T$. Where a binary string is treated as a member of $\mathbb{Z}_p$ it is implicitly converted in the natural manner. The signature scheme supports messages in $\{0,1\}^{n-1}$, but can be extended to support message in $\{0,1\}^*$ by using a collision resistant hash function, while the encryption scheme supports messages of arbitrary length due to the use of a DEM. Note that to minimize the public key size and ciphertext overhead in the scheme, the elements of the public key are placed in the group $\mathbb{G}_1$. However, this implies that signatures contain an element of group $\mathbb{G}_2$, having larger bit representations of elements.

$\mathsf{KeyGen}(1^k)$: Choose random generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and random integers $x, y \in \mathbb{Z}_p^*$, and compute $X = g_1^x$ and $Y = g_1^y$. The public key is (g_1, g_2, X, Y) and the private key is (x, y).

$\mathsf{Sign}(sk, m)$: To sign a message $m \in \{0,1\}^{n-1}$ first prepend a zero to m to give $m' = 0||m \in \{0,1\}^n$. Choose random $r \in \mathbb{Z}_p$. If $x+ry+m' \equiv 0 \bmod p$ then select another $r \in \mathbb{Z}_p$. Compute $\sigma = g_2^{\frac{1}{x+m'+yr}} \in \mathbb{G}_2$. The signature is $(\sigma, r) \in \mathbb{G}_2 \times \mathbb{Z}_p$.

$\mathsf{Verify}(pk, \sigma, m)$: If $e(X \cdot g_1^{m'} \cdot Y^r, \sigma) = e(g_1, g_2)$, where $m' = 0||m$, then return 1, otherwise return 0.

$\mathsf{Encrypt}(pk, m)$: To encrypt a message $m \in \{0,1\}^*$, choose random $s \in \mathbb{Z}_p^*$ and compute $c_1 = Y^s$ and $h = \mathsf{H}(c_1)$. Prepend a 1 to h to give $h' = 1||h \in \{0,1\}^n$, and compute $c_2 = X^s \cdot g_1^{s \cdot h'}$. Lastly, compute the key $K = e(g_1, g_2)^s \in \mathbb{G}_T$ and encrypt the message m using the DEM i.e. $c_3 = \mathsf{DEnc}(K, m)$. The ciphertext is $c = (c_1, c_2, c_3)$.

$\mathsf{Decrypt}(sk, c)$: To decrypt a ciphertext $c = (c_1, c_2, c_3)$, first compute $h = \mathsf{H}(c_1)$ and prepend a 1 to h to get $h' = 1||h$. If $c_1^{(x+h')/y} \neq c_2$, output $\perp$. Otherwise, compute the key $K = e(c_1, g_2^{1/y}) \in \mathbb{G}_T$, and output the message $m = \mathsf{DDec}(K, c_3)$.

We note that the computational cost of encryption and signature verification can be reduced by adding the redundant element $v = e(g_1, g_2)$ to the public key, but that this will significantly increase the public key size.

Theorem 3 *Suppose the (t', q, ϵ')-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then the above combined public key scheme is (t, q_d, q_s, ϵ)-EUF-CMA secure in the presence of a decryption oracle given that*

$$q_s \leq q, \quad \epsilon \geq 2\epsilon' + q_s/p \approx 2\epsilon' \quad \text{and} \quad t \leq t' - \Theta(q_d T_p + (q_d + q^2) T_e),$$

where T_p is the maximum time for evaluating a pairing and T_e is the maximum time for computing an exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{Z}_p$.

Proof. See Appendix B.

Theorem 4 *Suppose that the hash function H is (t_h, ϵ_h)-aSec secure, that the $(t_{dhi}, q_{dhi}, \epsilon_{dhi})$-DBDHI assumption holds in the groups $\mathbb{G}_1, \mathbb{G}_2$, and that the DEM is $(t_{dem}, q_{dem}, \epsilon_{dem})$-IND-CCA secure. Then the combined public key scheme above is $(t, q_d, q_s, \epsilon,)$-IND-CCA secure in the presence of a signing oracle given that*

$$q_s \leq q_{dhi}, \quad q_d \leq q_{dem}, \quad \epsilon \geq \epsilon_h + \epsilon_{dhi} + \epsilon_{dem} + q_s/p, \quad \text{and} \quad t \leq t_{min} - \Theta(q_d T_p + (q_{dhi} + q_d) T_e),$$

where $t_{min} = \min(t_h, t_{dhi}, t_{dem})$, T_p is the maximum time for evaluating a pairing, and T_e is the maximum time for computing an exponentiation in $\mathbb{G}_1$,$\mathbb{G}_2$.

Proof. See Appendix B.

The above scheme provides public keys consisting of three group elements of $\mathbb{G}_1$ and one group element of $\mathbb{G}_2$. If the scheme is instantiated using BN curves with sextic twists mentioned above, this translates into a public key size of 1280 bits for a 128 bit security level. Furthermore, assuming that the DEM is redundancy-free (which can be achieved if the DEM is a strong pseudorandom permutation [35]), the total ciphertext overhead is just two group elements of $\mathbb{G}_1$ which translates into 512 bits. Signatures consist of a single group element of $\mathbb{G}_2$ and an element of $\mathbb{Z}_p$, and will be 768 bits. Again, we assume that descriptions of groups and pairings are ignored in these calculations.

5.1 A Tag-based Extension

Unlike the generic construction presented in Section 4, we do not obtain a tag-based encryption component directly from the above construction of a combined public key scheme. However, to allow the scheme to be used to instantiate our combined signcryption, signature and encryption scheme presented in Section 7, we now show how to extend the scheme to support tag-based encryption.

The extension is based on the construction of a Tag-KEM from an ordinary KEM and a MAC by Abe, Gennaro and Kurosawa [1]. For this purpose, we furthermore need a key derivation function $\mathsf{KDF} : \mathbb{G}_T \to \mathcal{K}_d \times \mathcal{K}_m$ where $\mathcal{K}_d$ is the keyspace of the used DEM and $\mathcal{K}_m$ is the keyspace of the used MAC. The idea is to derive both a DEM and a MAC key from the original key used for the DEM in the above construction, and then use the MAC key to authenticate the tag. More specifically, the encryption component of the above scheme is modified as follows:

$\mathsf{Encrypt}(pk, t, m)$: To encrypt a message $m \in \{0,1\}^*$ under the tag t, choose random $s \in \mathbb{Z}_p^*$ and compute $c_1 = Y^s$ and $h = \mathsf{H}(c_1)$. Prepend a 1 to h to give $h' = 1||h \in \{0,1\}^n$, and compute $c_2 = X^s \cdot g_1^{s \cdot h'}$. Lastly, compute $K = e(g_1, g_2)^s$ and the keys $(K_d, K_m) = \mathsf{KDF}(K)$, encrypt the message m using the DEM, $c_3 = \mathsf{DEnc}(K_d, m)$, and compute a MAC of the tag t, $\tau = \mathsf{MAC}(K_m, t)$. The ciphertext is $c = (c_1, c_2, c_3, \tau)$.

$\mathsf{Decrypt}(sk, t, c)$: To decrypt a ciphertext $c = (c_1, c_2, c_3, \tau)$ under tag t, first compute $h = \mathsf{H}(c_1)$ and prepend a 1 to h to get $h' = 1||h$. If $c_1^{(x+h')/y} \neq c_2$, output $\perp$. Otherwise, compute $K = e(c_1, g_2^{1/y})$ and the keys $(K_d, K_m) = \mathsf{KDF}(K)$. If $\mathsf{MVer}(K_m, t, \tau) = 0$, output $\perp$, and otherwise output the message $m = \mathsf{DDec}(K, c_3)$.

It is relatively easy to confirm that the unforgeability of the combined scheme will not be affected by the above changes and that the proof of Theorem 3 is still valid for the extended scheme when the obvious changes are made to the proof. The confidentiality of the extended scheme is guaranteed by the following theorem.

Theorem 5 *Suppose that the hash function* H *is* (t_h, ϵ_h)*-aSec secure, that the* $(t_{dhi}, q_{dhi}, \epsilon_{dhi})$*-DBDHI assumption holds in* $\mathbb{G}_1, \mathbb{G}_2$*, that the key derivation function* KDF *is* $(t_{kdf}, \epsilon_{kdf})$*-secure, that the DEM is* $(t_{dem}, q_{dem}, \epsilon_{dem})$*-IND-CCA secure, and that the MAC is* $(t_{mac}, \epsilon_{mac})$*-sUF-OT secure. Then the combined tag-based public key scheme above is* (t, q_d, q_s, ϵ)*-IND-tag-CCA secure in the presence of a signing oracle given that*

$$q_s \leq q_{dhi}, \quad q_d \leq q_{dem}, \quad \epsilon \geq \epsilon_h + 2\epsilon_{dhi} + 2\epsilon_{kdf} + \epsilon_{dem} + q_d\epsilon_{mac} + (q_d + 2q_s)/p, \quad \textit{and}$$
$$t \leq t_{min} - \Theta(q_d T_p + (q_{dhi} + q_d)T_e),$$

where $t_{min} = \min(t_h, t_{dhi}, t_{kdf}, t_{dem}, t_{mac})$*,* T_p *is the maximum time for evaluating a pairing, and* T_e *is the maximum time for computing an exponentiation in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*.*

Proof. See Appendix B.

The tag-based extension will have the same public key and signature size as the original scheme, but the ciphertext overhead will be increased with the size of a MAC, which for a 128 bit security level, means an extra 128 bits. Hence, the total ciphertext overhead will be 640 bits.

6 Comparison of Schemes

In this section, we provide a comparison of the schemes arising from our IBE-based construction, our more efficient construction in Section 5 and the Cartesian product construction. In our comparison we will limit ourselves to other discrete-log/pairing-based schemes since provably secure (standard model) lattice-based schemes with short public keys are still unavailable and factoring-based schemes do not scale very well (for 128-bit security, the modulus would need to be > 3000 bits which is not competitive).We will include group generators in public key size calculations as the required number depends on the scheme, but we allow sharing of generators between signature and encryption component in Cartesian product instantiations to improve these constructions. Note that it is possible to reduce the private key of any scheme to a single short random seed by making the following simple modification to the scheme: to generate a public/private keypair, pick a random seed, generate the randomness required by the key generation algorithm by applying a pseudorandom generator to the seed, and generate the public/private keypair using this randomness, but store only the seed as the private key. Whenever the original private key is needed, re-compute this by applying the pseudorandom generator to the seed and re-run the key generation algorithm with the resulting randomness. This observation essentially makes the difference in private key sizes irrelevant, and we will not include this aspect in our comparison. We consider several instantiations of the Cartesian product construction with standard model secure encryption and signature schemes and give the results in Figure 2.

We will focus on Cartesian product instantiations using the scheme by Boneh and Boyen [6] as a signature component. This scheme is among the most efficient signature schemes and additionally has a short public key. To reduce the public key size even further, we can remove the redundant element $v = e(g_1, g_2)$ and place as many elements as possible in the group $\mathbb{G}_1$ of the pairing. The latter implies

Signature Scheme	PKE Scheme	Public Key Size	Signature Size	Ciphertext Overhead
BB [6]	BB [5] + BMW [9]	1792	768	512
BB [6]	KD [28]	2048	768	640
BB [6]	Kiltz [25]	1792	768	512
CSE(Gentry)		1536	768	$1280 + \lvert vk_{\mathcal{OT}}\rvert + \lvert\sigma_{\mathcal{OT}}\rvert$
Scheme from Sec. 5		1280	768	512

Fig. 2. Comparison of schemes at the 128-bit security level.

that signatures will be elements of $\mathbb{G}_2 \times \mathbb{Z}_p$ which results in an increase in signature size. However, since the Cartesian product constructions should compete with the combined public key schemes in terms of public key size, this tradeoff is desirable. While other signature schemes could be considered, we were not able to find a scheme providing shorter public keys without a significant disadvantage elsewhere. For instance, hash-based signature schemes give extremely short public keys (the hash function description plus the root digest), but result in signatures with length logarithmic in the number of messages to be signed. The signature scheme by Hofheinz and Kiltz [23] has shorter signatures than the Boneh-Boyen scheme and a public key consisting of a few group elements plus a hash key, but here the hash key will be long to achieve provable programmability.

For the encryption component, a relevant option is a DEM combined with the KEM obtained by applying the techniques by Boyen, Mei and Waters [9] to the second IBE scheme of Boneh and Boyen in [5], which also forms the basis of our concrete scheme. Combined with the Boneh-Boyen signature scheme, and assuming the group generators in the two schemes are shared, this yields a very efficient instantiation of the Cartesian product construction in which public keys consist of five group elements of $\mathbb{G}_1$, one group element of $\mathbb{G}_2$ (and a key defining a target collision resistant hash function). This is larger by two elements of $\mathbb{G}_1$ than the public key in our concrete construction from Section 5, which translates to a difference of 512 bits. Note that signature size, ciphertext overhead and computation costs are the same for the Cartesian product scheme and our construction.

Another encryption scheme to consider is that of Kurosawa and Desmedt [28]. Instantiating the Cartesian product construction with the Kurosawa-Desmedt scheme and the Boneh-Boyen signature scheme yields a scheme with a public key consisting of six elements of $\mathbb{G}_1$, one element of $\mathbb{G}_2$ (and a key defining a target collision resistant hash), assuming that the Kurosawa-Desmedt scheme is implemented in $\mathbb{G}_1$. Hence, the public key will be larger by three group elements of $\mathbb{G}_1$ compared to our concrete construction, which equates to a difference of 768 bits at the 128-bit security level. Signature size and signing and verification costs will be the same as in our construction, whereas the ciphertext overhead will be slightly larger (an extra 128 bits) due to the requirement that the symmetric encryption scheme used in the Kurosawa-Desmedt scheme is authenticated. However, decryption costs will be lower since no pairing computations are required.

Lastly, the encryption scheme of Kiltz [25] might be considered. Again, combining this with the Boneh-Boyen signature scheme, and assuming group generators are shared, will yield a Cartesian product scheme with public keys consisting of five elements of $\mathbb{G}_1$ and one element of $\mathbb{G}_2$. This is two group elements of $\mathbb{G}_1$ larger than the public key of our concrete construction, which equates to an increase of 512 bits at the 128-bit security level. Signature size and ciphertext overhead will be the same while decryption in the Cartesian product scheme will be more efficient, since no pairing computations are required.

In summary, our concrete construction of a combined public key scheme admits shorter public keys than any instantiation of the Cartesian product construction of Section 3.1 with known standard model secure encryption and signature schemes, and furthermore enjoys compact ciphertexts and signatures.

7 Signcryption

A signcryption scheme combines the functionality and security properties of signatures and encryption, and allows users to obtain message confidentiality and origin authentication through one operation. However, with the exception of a few random oracle model schemes [29–32], most signcryption schemes define separate key generation algorithms for senders and receivers, or essentially define a public/private keypair to consist of the concatenation of separate sender and receiver keypairs. Hence, a user playing the role of both sender and receiver will have to generate the equivalent of two keypairs. Extending the ideas of a combined public key scheme, we show how to construct a signcryption scheme which enables a user to use a single keypair for both sender and receiver roles, and which furthermore allows efficient instantiations with short public keys in the standard model.

We define a signcryption scheme to consist of the following algorithms:

$\mathsf{Setup}(1^k)$: This algorithm returns the common parameters of the scheme, cp.

$\mathsf{KeyGen}(cp)$: This algorithm returns a keypair (pk, sk) which can be used to both send and receive messages. To clarify which role a keypair is used in, we attach the subscript r when used as a receiver keypair, i.e. (pk_r, sk_r), and attach the subscript s when used as a sender keypair, i.e. (pk_s, sk_s).

$\mathsf{Signcrypt}(cp, sk_s, pk_r, m)$: This algorithm returns a signcryptext of message m from a sender with private key sk_s to a receiver with public key pk_r.

$\mathsf{Unsigncrypt}(cp, pk_s, sk_r, c)$: This algorithm returns either the message m from a sender with public key pk_s to a receiver with private key sk_r, or an error symbol $\perp$.

Note that any signcryption scheme can be redefined to use a single key generation algorithm as defined above. More specifically, a signcryption scheme using separate key generation algorithms for senders and receivers, $\mathsf{KeyGen}_s(cp)$ and $\mathsf{KeyGen}_r(cp)$, can be redefined to use a single key generation algorithm which simply runs $(pk_s, sk_s) \leftarrow \mathsf{KeyGen}_s(cp)$ and $(pk_r, sk_r) \leftarrow \mathsf{KeyGen}_r(cp)$, and returns the public key $pk = pk_s||pk_r$ and private key $sk = sk_s||sk_r$. When using (pk, sk) as either a sender or receiver keypair, only the relevant part of the keys are used. This trivial construction can be used as a benchmark when judging the public key size and other performance measures of concrete signcryption schemes using a single keypair for both sender and receiver roles.

7.1 Combined Signcryption, Signature and Encryption Scheme

While efficient signcryption schemes using a single short keypair for both sender and receiver roles are interesting in their own right, we will consider the more extended primitive which additionally allows users to use their keypair as part of an ordinary signature and encryption scheme, i.e. we consider a scheme implementing the functionality of signcryption, signature and encryption using a single keypair. This type of scheme consists of algorithms (Setup, KeyGen, Signcrypt, Unsigncrypt, Encrypt, Decrypt, Sign, Verify) such that (Setup, KeyGen, Signcrypt, Unsigncrypt) form a signcryption scheme as defined above, and (Setup, KeyGen, Encrypt, Decrypt) and (Setup, KeyGen, Sign, Verify) form an encryption and signature scheme in which the key generation is divided into a two-step process consisting of parameter generation using Setup and actual key generation using KeyGen.

As in the case of a combined public key scheme, a combined signcryption, signature and encryption scheme which is jointly secure (as defined in the following section) can trivially be constructed by concatenating the public keys and private keys of independent signcryption, signature and encryption schemes. However, as above, we focus on schemes which are more efficient (in terms of public key size and other measures) than this type of trivial construction. Note also that while an encryption or signature scheme can be constructed from a signcryption scheme by attaching an honestly generated "dummy" keypair to the public parameters and using this in combination with the signcrypt and unsigncrypt algorithms, this construction will not be jointly secure if a single keypair is used for the signcryption, signature and encryption components.

7.2 Security Model

A number of security models capturing different levels of security have been proposed for signcryption (e.g. see [33] for an overview). The main differences between these models concern whether or not the adversary is considered to be an insider with the knowledge of the secret key material of the challenge sender and challenge receiver in the definition of confidentiality and unforgeability, respectively, and to what extent the adversary is allowed to maliciously generate the keys of the users in the system. We will focus on the strongest security model of these which captures the notions of insider confidentiality and insider unforgeability in the multi-user setting. However, unlike the case of signcryption schemes using separate keypairs for the sender and receiver roles, we must give an adversary access to a signcryption and unsigncryption oracle in both the confidentiality and unforgeability definitions, since the challenge private key can be used to both send and receive messages. Furthermore, since we are considering a scheme which additionally implements the functionality of signature and encryption, we must also give the adversary access to signing and decryption oracles. In the following, we will formally define the security of the signcryption component of the type of combined scheme we are considering.

MU-IND-iCCA security in the presence of additional oracles: Multi-user indistinguishability against an insider chosen ciphertext attack (MU-IND-iCCA) in the presence of additional oracles captures the property that an adversary cannot distinguish between the signcryption of two different messages for a challenge receiver, even though all other keys in the system are maliciously generated, and the adversary is given access to signcryption, unsigncryption, signing and decryption oracles. Formally, this security notion is defined through the following game between a challenger and an adversary:

Setup: The challenger runs $\mathsf{Setup}(1^k)$ to get the common parameters cp, then runs $\mathsf{KeyGen}(cp)$ to generate a keypair (pk^*, sk^*) and gives pk^* to the adversary.

Phase 1: The adversary is given access to four oracles

- a signcryption oracle which, given a message m and a public receiver key pk_r, returns the signcryptext $c = \mathsf{Signcrypt}(cp, sk^*, pk_r, m)$,
- an unsigncryption oracle which, given a signcryptext c and a public sender key pk_s, returns the output of $\mathsf{Unsigncrypt}(cp, pk_s, sk^*, c)$,
- a signing oracle which, given a message m, returns the signature $\sigma = \mathsf{Sign}(sk^*, m)$, and
- a decryption oracle which, given a ciphertext c, returns the output of $\mathsf{Dec}(sk^*, c)$

Challenge: The adversary outputs a keypair (pk'_s, sk'_s) and a pair of equal length messages m_0, m_1. The challenger chooses a random bit b and computes a signcryptext $c^* = \mathsf{Signcrypt}(cp, sk'_s, pk^*, m_b)$ from the adversary's chosen user pk'_s to the challenger user pk^*, then gives c^* to the adversary.

Phase 2: As phase 1, with the restriction that the adversary may not submit the challenge signcryptext and sender (c^*, pk'_s) to the unsigncryption oracle. The adversary may however submit signcryptexts $c \neq c^*$ from its chosen challenge user pk'_s, and the challenge signcryptext c^* from other users $pk_s \neq pk'_s$. Furthermore, the adversary may use the signing and decryption oracles in an unrestricted manner.

Guess: The adversary makes a guess b' for b and wins if $b' = b$.

The adversary's advantage is $\left|\Pr[b' = b] - \frac{1}{2}\right|$.

We say that a signcryption scheme is $(t, q_{sc}, q_{usc}, q_s, q_d, \epsilon)$-MU-IND-iCCA secure in the presence of additional oracles if there exists no adversary running in time at most t and asking at most q_{sc}, q_{usc}, q_s, and q_d signcryption, unsigncryption, signature and decryption queries, respectively, and which has advantage at least ϵ in the above game.

MU-EUF-iCMA security in the presence of additional oracles: Multi-user existential unforgeability against an insider chosen message attack (MU-EUF-iCMA) in the presence of additional oracles captures the property that an adversary cannot create a valid signcryptext from a challenge sender which contains a new message, even if all other keys in the system are maliciously generated and the adversary is given access to signcryption, unsigncryption, signing and decryption oracles. Formally, this security notion is defined through the following game between an adversary and a challenger:

Setup: The challenger runs Setup to get the common parameters cp, then runs $\mathsf{KeyGen}(cp)$ to generate a keypair (pk^*, sk^*) and gives pk^* to the adversary.

Query phase: The adversary is given access to the four oracles defined in the MU-IND-iCCA game.

Forgery: The adversary outputs a keypair (pk'_r, sk'_r) and a signcryptext c^* and wins if $\mathsf{Unsigncrypt}(cp, pk^*, sk'_r, c^*)$ is a valid message m and the adversary never made a query (m, pk'_r) to its signcryption oracle.

The adversary's advantage is defined by $\Pr[\mathcal{A} \text{ wins}]$.

We say that a signcryption scheme is $(t, q_{sc}, q_{usc}, q_s, q_d, \epsilon)$-MU-EUF-iCCA secure in the presence of additional oracles if there exists no adversary running in time at most t and asking at most q_{sc}, q_{usc}, q_s, and q_d signcryption, unsigncryption, signature and decryption queries, respectively, and which has advantage at least ϵ in the above game.

```
$\mathsf{CSESC}(\mathcal{C}).\mathsf{KeyGen}(1^k)$:
        return $\mathcal{C}.\mathsf{KeyGen}(1^k)$

$\mathsf{CSESC}(\mathcal{C}).\mathsf{Sign}(sk, m)$:
        return $\mathcal{C}.\mathsf{Sign}(sk, 0||m)$

$\mathsf{CSESC}(\mathcal{C}).\mathsf{Verify}(pk, m, \sigma)$:
        return $\mathcal{C}.\mathsf{Verify}(pk, 0||m, \sigma)$

$\mathsf{CSESC}(\mathcal{C}).\mathsf{Enc}(pk, m)$:
        $tag = 0$
        return $\mathcal{C}.\mathsf{TEnc}(pk, tag, m)$

$\mathsf{CSESC}(\mathcal{C}).\mathsf{Dec}(sk, c)$:
        $tag = 0$
        return $\mathcal{C}.\mathsf{TDec}(sk, tag, c)$

$\mathsf{CSESC}(\mathcal{C}).\mathsf{Signcrypt}(cp, sk_s, pk_r, m)$:
        $\sigma \leftarrow \mathcal{C}.\mathsf{Sign}(sk_s, 1||pk_r||m)$
        $tag = 1||pk_s$
        $c \leftarrow \mathcal{C}.\mathsf{TEnc}(pk_r, tag, m||\sigma)$
        return $c$

$\mathsf{CSESC}(\mathcal{C}).\mathsf{Unsigncrypt}(cp, pk_s, sk_r, c)$:
        $tag = 1||pk_s$
        $m' \leftarrow \mathcal{C}.\mathsf{TDec}(sk_r, tag, c)$
        if $m' = \perp$
            return $\perp$
        $m||\sigma = m'$
        if $1 \leftarrow \mathcal{C}.\mathsf{Verify}(pk_s, 1||pk_r||m, \sigma)$
            return $m$
        return $\perp$
```

Fig. 3. Combined signcryption, signature and encryption scheme $\mathsf{CSESC}(\mathcal{C})$

The security of the encryption and signature components is defined like the security of the corresponding components of a combined public key scheme (see Section 3), except that the adversary will additionally have access to a signcryption and unsigncryption oracle defined as in the MU-IND-iCCA game above. We omit the straightforward definitions of these models.

7.3 Construction Based on a Combined Public Key Scheme

We will now show how a (tag-based) combined public key scheme can be used to construct a combined signcryption, signature and encryption scheme. Our construction is based on the "sign then tag-based encrypt" (StTE) construction of [33]. The tag-based property of the underlying combined public key scheme serves a dual purpose: firstly, using the public sender key as a tag in the construction of the signcryption component allows the signcryptext to be bound to a specific sender/receiver keypair which is required to achieve security in the multi-user setting, and secondly, separation between the signcryption and the encryption components of the scheme is ensured by using different tags for the construction of these.

Given a tag-based combined public key scheme $\mathcal{C} = (\mathcal{C}.\mathsf{KeyGen}, \mathcal{C}.\mathsf{Sign}, \mathcal{C}.\mathsf{Verify}, \mathcal{C}.\mathsf{TEnc}, \mathcal{C}.\mathsf{TDec})$, the combined signcryption, signature and encryption scheme $\mathsf{CSESC}(\mathcal{C})$ is defined as shown in Figure 3. The next four theorems establish the joint security of $\mathsf{CSESC}(\mathcal{C})$.

Theorem 6 *Let $\mathcal{C}$ be a joint signature and tag-based encryption scheme, the signature component of which is $(t, q_d, q_s\epsilon)$-EUF-CMA secure in the presence of a decryption oracle. Then the signcryption component of* $\mathsf{CSESC}(\mathcal{C})$ *is $(t', q'_{sc}, q'_{usc}, q'_s, q'_d, \epsilon)$-MU-EUF-iCMA secure in the presence of additional oracles, where $q_s = q'_{sc} + q'_s$, $q_d = q'_{usc} + q'_d$, $t = t' + O(q'_{sc}T_e + q'_{usc}T_v)$, and T_e and T_v are the maximum time for computing an encryption and verifying a signature in $\mathcal{C}$.*

Proof of Theorem 6. Suppose there exists an adversary $\mathcal{A}$ breaking the unforgeability in the presence of additional oracles property of the signcryption component of $\mathsf{CSESC}(\mathcal{C})$. We construct an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ to break the EUF-CMA security in the presence of additional oracles of the signature component of $\mathcal{C}$.

Setup: $\mathcal{B}$ is given public key pk^* which it passes to $\mathcal{A}$.
Queries: $\mathcal{B}$ responds to $\mathcal{A}$'s queries as follows:

– On signcryption query (pk_r, m), $\mathcal{B}$ submits $1||pk_r||m$ to its signing oracle and is given the signature σ. $\mathcal{B}$ sets $tag = 1||pk^*$ and computes $c \leftarrow \mathcal{C}.\mathsf{TEnc}(pk_r, tag, m||\sigma)$, then returns c to $\mathcal{A}$.

– On unsigncryption query (pk_s, c), $\mathcal{B}$ sets $tag = 1||pk_s$ then submits (tag, c) to its decryption oracle and is given the response m'. If $m' = \perp$ then $\mathcal{B}$ returns $\perp$, otherwise it parses m' as $m||\sigma$ and checks $\mathcal{C}.\mathsf{Verify}(pk_s, 1||pk^*||m, \sigma)$. If the signature verifies it returns m to $\mathcal{A}$, otherwise it returns $\perp$.
– On signing query m, $\mathcal{B}$ submits $0||m$ to its signing oracle and returns the signature to $\mathcal{A}$.
– On decryption query (tag, c), $\mathcal{B}$ submits $(0||tag, c)$ to its decryption oracle and returns result to $\mathcal{A}$.

Forgery: $\mathcal{A}$ outputs a keypair (pk'_r, sk'_r) and a signcryptext c^*. $\mathcal{B}$ sets $tag = 1||pk^*$ and computes $m' \leftarrow \mathcal{C}.\mathsf{TDec}(sk'_r, tag, c^*)$. If $\mathcal{A}$'s forgery is valid then $m' = m||\sigma$ and $\mathcal{C}.\mathsf{Verify}(pk^*, 1||pk'_r||m, \sigma) = 1$. In addition, $\mathcal{A}$ never made a query (pk'_r, m) to its signcryption oracle, so $\mathcal{B}$ never queried $1||pk'_r||m$ to its signing oracle. $\mathcal{B}$ outputs message $1||pk'_r||m$ and signature σ and succeeds with the same probability as $\mathcal{A}$.

□

Theorem 7 *Let $\mathcal{C}$ be a joint signature and tag-based encryption scheme, the encryption component of which is (t, q_d, q_s, ϵ)-IND-tag-CCA secure in the presence of additional oracles. Then the signcryption component of* $\mathsf{CSESC}(\mathcal{C})$ *is $(t', q'_{sc}, q'_{usc}, q'_d, q'_s, \epsilon)$-MU-EUF-iCMA secure in the presence of additional oracles, where $q_s = q'_{sc} + q'_s$, $q_d = q'_{usc} + q'_d$, $t = t' + O(q'_{sc}T_e + q'_{usc}T_v)$, and T_e and T_v are the maximum time for computing an encryption and verifying a signature in $\mathcal{C}$.*

Proof of Theorem 7. Suppose there exists an adversary $\mathcal{A}$ breaking the confidentiality in the presence of additional oracles property of the signcryption component of $\mathsf{CSESC}(\mathcal{C})$. We construct an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ to break the joint IND-tag-CCA security in the presence of additional oracles of the encryption component of $\mathcal{C}$.

Setup: $\mathcal{B}$ is given public key pk^* which it passes to $\mathcal{A}$.
Phase 1: $\mathcal{B}$ handles $\mathcal{A}$'s queries as in the proof of Theorem 6.
Challenge: $\mathcal{A}$ outputs a keypair (pk'_s, sk'_s) and a pair of equal length messages m_0, m_1. $\mathcal{B}$ submits $1||pk^*||m_0$ and $1||pk^*||m_1$ to its signing oracle to get signatures σ_0 and σ_1, then outputs challenge messages $m_0||\sigma_0, m_1||\sigma_1$ and tag $1||pk'_s$. $\mathcal{B}$ receives in return the challenge ciphertext c^* which it passes to $\mathcal{A}$.
Phase 2: $\mathcal{B}$ handles queries as in phase 1. $\mathcal{A}$ is not permitted to make the query (pk'_s, c^*) to its unsigncryption oracle, so $\mathcal{B}$ does not submit c^* with tag $1||pk'_s$ to its decryption oracle.
Guess: $\mathcal{A}$ outputs a guess b' which $\mathcal{B}$ outputs as its guess, winning with the same probability as $\mathcal{A}$.

□

Theorem 8 *Let $\mathcal{C}$ be a joint signature and tag-based encryption scheme, the signature component of which is (t, q_d, q_s, ϵ)-EUF-CMA secure in the presence of a decryption oracle. Then the signature component of* $\mathsf{CSESC}(\mathcal{C})$ *is $(t', q'_{sc}, q'_{usc}, q'_s, q'_d, \epsilon)$-EUF-CMA secure in the presence of additional oracles, where $q_s = q'_{sc} + q'_s$, $q_d = q'_{usc} + q'_d$, $t = t' + O(q'_{sc}T_e + q'_{usc}T_v)$, and T_e and T_v are the maximum time for computing an encryption and verifying a signature in $\mathcal{C}$.*

Proof of Theorem 8. Suppose there exists an adversary $\mathcal{A}$ breaking the EUF-CMA security in the presence of additional oracles of the signature component of $\mathsf{CSESC}(\mathcal{C})$. We construct an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ to break the EUF-CMA security in the presence of additional oracles of the signature component of $\mathcal{C}$.

Setup: $\mathcal{B}$ is given public key pk^* which it passes to $\mathcal{A}$.
Queries: $\mathcal{B}$ handles $\mathcal{A}$'s queries as in the proof of Theorem 6.
Forgery: $\mathcal{A}$ outputs a message m^* and a signature σ^*. If $\mathcal{A}$'s forgery is valid then $\mathcal{C}.Verify(pk, 0||m^*, \sigma^*) = 1$ and $\mathcal{A}$ never made a query m^* to its signing oracle, so $\mathcal{B}$ never queried $0||m^*$ to its signing oracle. $\mathcal{B}$ outputs message $0||m^*$ and signature σ^* and succeeds with the same probability as $\mathcal{A}$.

□

Theorem 9 *Let $\mathcal{C}$ be a joint signature and tag-based encryption scheme, the encryption component of which is (t, q_d, q_s, ϵ)-IND-tag-CCA secure in the presence of a signing oracle. Then the encryption component of* $\mathsf{CSESC}(\mathcal{C})$ *is $(t', q'_{sc}, q'_{usc}, q'_s, q'_d, \epsilon)$-IND-CCA secure in the presence of additional oracles, where $q_s = q'_{sc} + q'_s$, $q_d = q'_{usc} + q'_d$, $t = t' + O(q'_{sc}T_e + q'_{usc}T_v)$, and T_e and T_v are the maximum time for computing an encryption and verifying a signature in $\mathcal{C}$.*

Proof of Theorem 9. Suppose there exists an adversary $\mathcal{A}$ breaking the IND-tag-CCA security in the presence of additional oracles of the encryption component of $\mathsf{CSESC}(\mathcal{C})$. We construct an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ to break the IND-tag-CCA security in the presence of additional oracles of the encryption component of $\mathcal{C}$.

Setup: $\mathcal{B}$ is given public key pk^* which it passes to $\mathcal{A}$.
Phase 1: $\mathcal{B}$ handles $\mathcal{A}$'s queries as in the proof of Theorem 6.
Challenge: $\mathcal{A}$ outputs a pair of equal length messages m_0, m_1 and a tag t^*. $\mathcal{B}$ outputs challenge messages m_0, m_1 and tag $0||t^*$. $\mathcal{B}$ receives in return the challenge ciphertext c^* which it passes to $\mathcal{A}$.
Phase 2: $\mathcal{B}$ handles queries as in phase 1. $\mathcal{A}$ is not permitted to make the query (t^*, c^*) to its decryption oracle, so $\mathcal{B}$ does not submit c^* with tag $0||t^*$ to its decryption oracle.
Guess: $\mathcal{A}$ outputs a guess b' which $\mathcal{B}$ outputs as its guess, winning with the same probability as $\mathcal{A}$.
□

The above combined signcryption, signature and encryption scheme can be instantiated by any combined signature and tag-based encryption scheme. In particular, we can use the tag-based extension of the concrete scheme presented in Section 5. Note that in this case, the group generators g_1, g_2 can be shared among all users and can hence be included in the public parameters. This yields public keys consisting of only two group elements of $\mathbb{G}_1$. Hence, when instantiated with BN curves, public keys will have a size of 512 bits for a 128 bit security level. Ciphertext overhead and signature size will remain unchanged from the underlying tag-based extension i.e. 640 and 768 bits, respectively. The signcryptext overhead corresponds to the sum of the ciphertext overhead and signature size, and will hence be 1408 bits.

8 Conclusions and Future Research

We have revisited the topic of joint security for combined public key schemes, focussing on the construction of schemes in the standard model, an issue not fully addressed in prior work. We gave a general construction for combined public key schemes from weakly secure IBE, as well as a more efficient concrete construction based on pairings. Using BN curves, these can be efficiently instantiated at high security levels and have performance that is competitive with the best schemes arising from the Cartesian product construction. Our results fill the gap left open in the original work of Haber and Pinkas [19], of constructing standard-model-secure combined public key schemes in which the signature and encryption components share an identical keypair. An interesting open problem is to construct efficient combined public key schemes in the standard model not using pairings. For example, is it possible to obtain joint security in the discrete log or in the RSA setting, in the standard model?

We also considered the construction of signcryption schemes from combined public key schemes, giving a method to produce a triple of schemes (signature, encryption, and signcryption) that are jointly secure in an appropriate and strong security model, from any jointly secure tag-based combined public key scheme. This leads to efficient standard model signcryption schemes in which a single short keypair can be used for both sender and receiver functions.

Our work points the way to an interesting new research area in cryptography, which closely relates to and generalises the topic of cryptographic agility [2]. The general question can be posed as follows: under what conditions is it safe to use the same key (or key pair) across multiple instantiations of the *same* or *different* cryptographic primitives?

References

1. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-kem/dem: A new framework for hybrid encryption and a new analysis of kurosawa-desmedt kem. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005)
2. Acar, T., Belenkiy, M., Bellare, M., Cash, D.: Cryptographic agility and its relation to circular encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 403–422. Springer, Heidelberg (2010)
3. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: EUROCRYPT 2002. pp. 83–107 (2002)

4. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S.E. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
5. Boneh, D., Boyen, X.: Efficient selective identity-based encryption without random oracles. Journal of Cryptology To appear, available from `http://www.springerlink.com/content/n6363233lk4q4h11/`
6. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)
7. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. 36(5), 1301–1328 (2007)
8. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)
9. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM Conference on Computer and Communications Security. pp. 320–329. ACM (2005)
10. Coron, J.S., Joye, M., Naccache, D., Paillier, P.: Universal padding schemes for RSA. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 226–241. Springer, Heidelberg (2002)
11. Cramer, R., Hofheinz, D., Kiltz, E.: Chosen-ciphertext secure encryption from hard algebraic set systems. Cryptology ePrint Archive, Report 2009/142 (2009)
12. Cui, Y., Fujisaki, E., Hanaoka, G., Imai, H., Zhang, R.: Formal security treatments for IBE-to-signature transformation: Relations among security notions. IEICE Transactions 92-A(1), 53–66 (2009)
13. Dodis, Y., Freedman, M.J., Jarecki, S., Walfish, S.: Optimal signcryption from any trapdoor permutation. Cryptology ePrint Archive, Report 2004/020 (2004), `http://eprint.iacr.org/`
14. Dodis, Y., Freedman, M.J., Jarecki, S., Walfish, S.: Versatile padding schemes for joint signature and encryption. In: ACM Conference on Computer and Communications Security. pp. 344–353. ACM (2004)
15. EMV Specifications, Version 4.2, Books 1–4 (June 2008), `http://www.emvco.com/`
16. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology 23(2), 224–280 (2010)
17. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
18. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
19. Haber, S., Pinkas, B.: Securely combining public-key cryptosystems. In: ACM Conference on Computer and Communications Security. pp. 215–224 (2001)
20. Hanaoka, G., Kurosawa, K.: Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In: ASIACRYPT. pp. 308–325 (2008)
21. Hanaoka, G., Kurosawa, K.: Between hashed dh and computational dh: Compact encryption from weaker assumption. IEICE Transactions 93-A(11), 1994–2006 (2010)
22. Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
23. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
24. Kelsey, J., Schneier, B., Wagner, D.: Protocol interactions and the chosen protocol attack. In: Christianson, B., Crispo, B., Lomas, T.M.A., Roe, M. (eds.) Security Protocols Workshop. LNCS, vol. 1361, pp. 91–104. Springer, Heidelberg (1997)
25. Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
26. Klíma, V., Rosa, T.: Further results and considerations on side channel attacks on RSA. In: Jr., B.S.K., Çetin Kaya Koç, Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 244–259. Springer, Heidelberg (2003)
27. Komano, Y., Ohta, K.: Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 366–382. Springer, Heidelberg (2003)
28. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
29. Li, C.K., Yang, G., Wong, D.S., Deng, X., Chow, S.S.M.: An efficient signcryption scheme with key privacy. In: Lopez, J., Samarati, P., Ferrer, J.L. (eds.) EuroPKI 2007. LNCS, vol. 4582, pp. 78–93. Springer, Heidelberg (2007)
30. Libert, B., Quisquater, J.J.: Efficient signcryption with key privacy from gap Diffie-Hellman groups. In: Bao, F., Deng, R.H., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 187–200. Springer, Heidelberg (2004)
31. Libert, B., Quisquater, J.J.: Improved signcryption from q-Diffie-Hellman problems. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 220–234. Springer, Heidelberg (2005)
32. Malone-Lee, J.: Signcryption with non-interactive non-repudiation. Des. Codes Cryptography 37(1), 81–109 (2005)
33. Matsuda, T., Matsuura, K., Schuldt, J.C.N.: Efficient constructions of signcryption schemes and signcryption composability. In: Roy, B.K., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 321–342. Springer, Heidelberg (2009)

34. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC. pp. 33–43 (1989)
35. Phan, D.H., Pointcheval, D.: About the security of ciphers (semantic security and pseudo-random permutations). In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 182–197. Springer, Heidelberg (2005)
36. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: FSE. pp. 371–388 (2004)
37. Rubin, K., Silverberg, A.: Compression in finite fields and torus-based cryptography. SIAM J. Comput. 37(5), 1401–1428 (2008)
38. Vasco, M.I.G., Hess, F., Steinwandt, R.: Combined (identity-based) public key schemes. Cryptology ePrint Archive, Report 2008/466 (2008), `http://eprint.iacr.org/`
39. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A Definitions and Security Notions of Standard Primitives

A.1 Signature Schemes

A digital signature scheme consists of three algorithms:

$\mathsf{KeyGen}(1^k)$**:** A probabilistic key generation algorithm that takes as input a security parameter and outputs a pair (pk, sk) where pk is the public key and sk the private key.

$\mathsf{Sign}(sk, m)$**:** A signing algorithm, often probabilistic, that takes a message m from some message space $\mathcal{M}$ and a private key sk, and outputs a signature σ from some signature space $\mathcal{S}$.

$\mathsf{Verify}(pk, \sigma, m)$**:** A verification algorithm, which may be probabilistic, that takes a message signature pair (σ, m) and a public key pk and outputs 1 or 0.

A signature scheme must be correct, that is $\mathsf{Verify}(pk, \mathsf{Sign}(sk, m), m) = 1$ should hold for all messages m in the message space and all public-private keypairs (pk, sk) generated by KeyGen. If (σ, m) is such that $\mathsf{Verify}(pk, \sigma, m) = 1$, then σ is said to be a valid signature for the message m under the public key pk.

The standard notion of security for signature schemes is that of existential unforgeability under an adaptive chosen message attack (EUF-CMA). It is defined using the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger generates a keypair $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$ and gives $\mathcal{A}$ the challenge public key pk.

Queries: $\mathcal{A}$ requests signatures on messages m_i of its choice. The challenger responds to each query with a signature $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$. These requests may depend not only on the public key pk but also on previous queries and the signatures obtained in response.

Forgery: Eventually $\mathcal{A}$ outputs a message signature pair (σ, m) and wins the game if m is not one of the queried messages m_i and $\mathsf{Verify}(pk, \sigma, m) = 1$.

The advantage of an adversary $\mathcal{A}$ is the probability that it wins the above game.

An adversary $\mathcal{A}(t, q_S, \epsilon)$-breaks a signature scheme if $\mathcal{A}$ runs in time at most t, makes at most q_S signature queries and has advantage at least ϵ. A signature scheme is (t, q_S, ϵ)-EUF-CMA secure if no adversary (t, q_S, ϵ)-breaks it.

Another notion of security is strong existential unforgeability (sEUF-CMA). This captures the idea that an adversary cannot generate a new signature on a message it already holds a signature for. It is formalised using the same game as above, but now the adversary wins if its forgery σ is valid and its output is not one of (σ_i, m_i).

A signature scheme is one-time strongly unforgeable if it is $(t, 1, \epsilon)$-strongly unforgeable under adaptive chosen message attacks.

A.2 Tag-based Encryption

A tag-based encryption (TBE) scheme is a public key encryption (PKE) scheme where the encryption and decryption operations require an additional binary string, the tag. Such a scheme consists of three algorithms:

$\mathsf{KeyGen}(1^k)$**:** A probabilistic key generation algorithm that takes as input a security parameter and outputs a pair (pk, sk) where pk is the public key and sk the private key.

$\mathsf{Encrypt}(pk, t, m)$**:** A probabilistic encryption algorithm that takes a public key pk, a tag t and a message m from some message space $\mathcal{M}$, and outputs a ciphertext c from some ciphertext space $\mathcal{C}$.

$\mathsf{Decrypt}(sk, t, c)$**:** A decryption algorithm, occasionally probabilistic, that takes as input a private key sk, a tag t and a ciphertext c, and outputs a message m or a failure symbol $\perp$.

A TBE scheme should be correct, that is $\mathsf{Decrypt}(sk, t, \mathsf{Encrypt}(pk, t, m)) = m$ should hold for all messages m in the message space and all public-private keypairs (pk, sk) generated by KeyGen. The standard security notion of a tag-based encryption scheme is called IND-tag-CCA security and is captured by the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger runs $\mathsf{KeyGen}(1^k)$ to produce a public key pk and a private key sk. The adversary $\mathcal{A}$ is given pk.

Phase 1: $\mathcal{A}$ has repeated access to a decryption oracle which it may query on any ciphertext and tag of its choosing.

Challenge: $\mathcal{A}$ chooses two equal length messages m_0, m_1 and a tag t^*. The challenger chooses a random bit b, computes $c^* \leftarrow \mathsf{Encrypt}(pk, t^*, m_b)$, and passes c^* to the adversary.

Phase 2: $\mathcal{A}$ again has repeated access to a decryption oracle, but with the restriction that it may not submit the challenge ciphertext c^* with the challenge tag t^*. The adversary may however submit the challenge ciphertext with a different tag $t \neq t^*$, or a ciphertext $c \neq c^*$ with the challenge tag t^*.

Guess: $\mathcal{A}$ outputs a guess b' for bit b.

The adversary's advantage is $\left|\Pr[b' = b] - \frac{1}{2}\right|$. An adversary $\mathcal{A}$ (t, q_d, ϵ)-breaks a TBE scheme if $\mathcal{A}$ runs in time at most t, makes at most q_d decryption queries and has advantage at least ϵ. A TBE scheme is (t, q_d, ϵ)-IND-tag-CCA secure if no adversary (t, q_d, ϵ)-breaks it.

A.3 Identity-based Encryption

An identity-based encryption (IBE) scheme has the following algorithms:

$\mathsf{Setup}(1^k)$**:** A probabilistic setup algorithm that takes as input the security parameter and outputs the master private key msk and the master public key mpk.

$\mathsf{Extract}(msk, ID)$**:** A private key extraction algorithm, often probabilistic, that takes the master private key msk and an identity ID, and outputs a corresponding private key sk_{ID}.

$\mathsf{Encrypt}(mpk, ID, m)$**:** A probabilistic encryption algorithm that takes the master public key mpk, and identity ID and a message m from some message space $\mathcal{M}$, and outputs a ciphertext c from some ciphertext space $\mathcal{C}$.

$\mathsf{Decrypt}(mpk, sk_{ID}, c)$**:** A decryption algorithm, occasionally probabilistic, that takes the master public key mpk, a private key sk_{ID} and a ciphertext c, and out puts a message m or a failure symbol $\perp$

An IBE scheme should be correct, that is $\mathsf{Decrypt}(mpk, sk_{ID}, \mathsf{Encrypt}(mpk, ID, m)) = m$ should hold for all messages m in the message space whenever sk_{ID} is a private key output by $\mathsf{Extract}$ on input ID. We recall the standard definitions of correctness, selective-ID (IND-sID-CPA) security and one-wayness (OW-ID-CPA) for IBE.[4] Selective-ID (IND-sID-CPA) security of an IBE scheme is defined through the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The adversary $\mathcal{A}$ chooses a challenge identity ID^*. The challenger runs $\mathsf{Setup}(1^k)$ to produce a public key mpk and a private key msk. $\mathcal{A}$ is given mpk.

[4] Note from [12] that in general achieving IND-ID-CPA security does not imply one-wayness, however for an IBE scheme with a sufficiently large message space the implication holds. The IBE schemes considered here all satisfy this property.

Phase 1: $\mathcal{A}$ has repeated access to an extraction oracle which it may query on any identity of its choosing except the challenge identity ID^*.

Challenge: $\mathcal{A}$ outputs two equal length messages m_0, m_1. The challenger chooses a random bit b, computes $c^* \leftarrow \mathsf{Encrypt}(mpk, ID^*, m_b)$, and passes c^* to the adversary.

Phase 2: $\mathcal{A}$ again has repeated access to an extraction oracle which it may query on any identity of its choosing except the challenge identity ID^*.

Guess: $\mathcal{A}$ outputs a guess b' for bit b.

The adversary's advantage is $\left|\Pr[b' = b] - \frac{1}{2}\right|$. An adversary $\mathcal{A}$ (t, q_e, ϵ)-breaks an IBE scheme if $\mathcal{A}$ runs in time at most t, makes at most q_e exrtaction queries and has advantage at least ϵ. An IBE scheme is (t, q_e, ϵ)-IND-sID-CPA secure if no adversary (t, q_e, ϵ)-breaks it.

One-wayness of an IBE scheme (OW-ID-CPA) is defined through the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger runs $\mathsf{Setup}(1^k)$ to produce a master public key mpk and a master private key msk. The adversary $\mathcal{A}$ is given mpk.

Phase 1: $\mathcal{A}$ has repeated access to a private key extraction oracle which it may query on any identity of its choosing.

Challenge: $\mathcal{A}$ outputs an identity ID^* for which it has not made an extraction query. The challenger chooses a random message m, computes $c^* \leftarrow \mathsf{Encrypt}(mpk, ID^*, m)$, and passes c^* to the adversary.

Phase 2: $\mathcal{A}$ again has repeated access to an extraction oracle, but with the restriction that it may not submit the challenge identity ID^*.

Guess: $\mathcal{A}$ outputs a guess m' for message m.

The adversary's advantage is $Pr[m' = m]$. An adversary $\mathcal{A}$ (t, q_e, ϵ)-breaks an IBE scheme if $\mathcal{A}$ runs in time at most t, makes at most q_e extraction queries and has advantage at least ϵ. An IBE scheme is (t, q_e, ϵ)-OW-ID-CPA secure if no adversary (t, q_e, ϵ)-breaks it.

A.4 Data Encapsulation Mechanism

A data encapsulation mechanism (DEM) is given by a keyspace $\mathcal{K}$ and the following two algorithms:

$\mathsf{DEnc}(K, m)$**:** A deterministic encryption algorithm that takes as input a key $K \in \mathcal{K}$ and a message m, and returns a ciphertext c.

$\mathsf{DDec}(K, c)$**:** A deterministic decryption algorithm that takes as input a key $K \in \mathcal{K}$ and a ciphertext c, and returns a message m or an error symbol $\perp$.

It is required that for all $K \in \mathcal{K}$ and all messages m, $\mathsf{DDec}(K, \mathsf{DEnc}(K, m)) = m$.

Indistinguishability against a chosen ciphertext attack (IND-CCA) for a DEM is defined through the following game between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger chooses a key $K \in \mathcal{K}$ uniformly at random.

Phase 1: $\mathcal{A}$ can adaptively submit decryption queries c which the challenger responds to by returning the output of $\mathsf{DDec}(K, c)$.

Challenge: $\mathcal{A}$ outputs two messages m_0, m_1 of equal length. The challenger flips a fair coin $b \leftarrow \{0, 1\}$, and returns the challenge ciphertext $c^* = \mathsf{DEnc}(K, m_b)$.

Phase 2: As Phase 1, but with the restriction that $\mathcal{A}$ cannot submit c^* to the decryption oracle.

Guess: Eventually, $\mathcal{A}$ outputs a bit b'.

The advantage of $\mathcal{A}$ is $|\Pr[b = b'] - 1/2|$.

A DEM is said to be (t, q, ϵ)-IND-CCA secure if there exists no adversary $\mathcal{A}$ which runs in time at most t, makes at most q decryption queries, and has advantage at least ϵ.

A.5 Always Second-Preimage Resistant Hash Functions

The construction presented in Section 5 requires the used hash function H to be *always second-preimage resistant* (aSec secure). This type of collision resistance, which was named by Rogaway and Shrimpton [36], informally guarantees that an adversary given a randomly chosen message m from the domain of the hash function, cannot compute a second message m' that collides with m when applying the hash function[5]. More specifically, we define aSec security as follows:

Let $\mathsf{H} : \mathcal{D} \to \mathcal{R}$ be a hash function with domain $\mathcal{D}$ and range $\mathcal{R}$. We say that H is (t, ϵ)-aSec secure if there exists no adversary $\mathcal{A}$ which runs in time at most t, and for which

$$\Pr[m \leftarrow \mathcal{D}; m' \leftarrow \mathcal{A}(m) : m \neq m' \wedge \mathsf{H}(m) = \mathsf{H}(m')] \geq \epsilon.$$

Note that aSec security is different from the notion of *universal one-way hash function family* [34] which allows the adversary to choose the message m, but requires the collision to be found for a randomly chosen key k defining the hash function (k will only be available to the adversary after m has been chosen). In the literature, aSec-secure hash functions have also been used under the somewhat ambiguous name *target collision resistance hash functions* (e.g. in [11, 20, 21]).

A.6 Message Authentication Code

A message authentication code (MAC) is given by a keyspace $\mathcal{K}$ and the following algorithms:

$\mathsf{MAC}(K, m)$**:** A deterministic algorithm which on input a key $K \in \mathcal{K}$ and a message m, returns a message authentication tag τ.

$\mathsf{MVer}(K, m, \tau)$**:** A deterministic verification algorithm which on input a key $K \in \mathcal{K}$, a message m, and a message authentication tag τ, returns either 0 or 1.

It is required that for all $K \in \mathcal{K}$ and all messages m that $\mathsf{MVer}(K, m, \mathsf{MAC}(K, m)) = 1$.

Strong unforgeability against a one-time attack (sUF-OT) for a MAC is defined through the following game between a challenger and an adversary $\mathcal{A}$:

Setup: The challenger chooses a key $K \in \mathcal{K}$ uniformly at random.

Query: $\mathcal{A}$ is allowed to make a single query on a message m, and the challenger responds by returning $\tau = \mathsf{MAC}(K, m)$.

Forgery: $\mathcal{A}$ returns a message/tag pair (m^*, τ^*).

The advantage of $\mathcal{A}$ in the above game is $\Pr[\mathsf{MVer}(K, m^*, \tau^*) = 1 \wedge (m^*, \tau^*) \neq (m, \tau)]$

We say that a MAC is (t, ϵ)-sUF-OT secure if there exists no adversary $\mathcal{A}$ which runs in time at most t and has advantage at least ϵ in the above game.

A.7 Key Derivation Function

We define a key derivation function (KDF) for a set of keyspaces $(\mathcal{K}, \mathcal{K}_d, \mathcal{K}_m)$ as a function $\mathsf{KDF} : \mathcal{K} \to \mathcal{K}_d \times \mathcal{K}_m$, and require that the output of a KDF is indistinguishable from a uniformly chosen keypair in $\mathcal{K}_d \times \mathcal{K}_m$ if the input is chosen uniformly in $\mathcal{K}$. More specifically, we define the advantage of an adversary against a KDF to be

$$|\Pr[K \leftarrow \mathcal{K}; (K_d, K_m) = \mathsf{KDF}(K) : \mathcal{A}(K_d, K_m) = 1] - \Pr[(K_d, K_m) \leftarrow \mathcal{K}_d \times \mathcal{K}_d : \mathcal{A}(K_d, K_m) = 1]|.$$

A KDF is said to be (t, ϵ)-secure if there exists no adversary $\mathcal{A}$ which runs in time at most t and has advantage at least ϵ against the KDF.

In our concrete construction, we will make use of a KDF to generate the keys of a DEM with keyspace $\mathcal{K}_d$ and a MAC with keyspace $\mathcal{K}_m$.

[5] Note that [36] defines aSec security for a family of hash functions indexed by a key, but requires the described collision property to hold for *all* keys. Like [11, 20, 21], we will take a slightly simpler approach and define aSec security for a single hash function.

B Proofs of Security of More Efficient Combined Scheme Construction

Proof of Theorem 3: We follow the proof of Lemma 10 in [6].

Firstly, we recall the *basic* Boneh-Boyen signature scheme defined in groups $(\mathbb{G}_1, \mathbb{G}_2)$ equipped with a paring $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, but which has signatures in $\mathbb{G}_2$. This scheme has public keys of the form $(g_1, g_2, u = g_1^x, v = e(g_1, g_2))$ where g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Signatures are of the form $\sigma = g_2^{1/(x+m)}$ except if $m = -x$ in which case the signature on m is defined to be $\sigma = 1$. Lastly, verification of a signature $\sigma \neq 1$ consists of checking that $e(u \cdot g_1^m, \sigma) = v$, and verification of $\sigma = 1$ consists of checking that $ug_1^m = 1$.

Lemma 9 of [6] implies that if the (t', q, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the above basic Boneh-Boyen signature scheme is (t, q_s, ϵ)-secure against existential forgery under a weak chosen message attack (EUF-wCMA) provided that

$$q_s \leq q, \qquad \text{and} \qquad t \leq t' - \Theta(q^2 T),$$

where T is the maximum time for an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{Z}_p$.

Given an sEUF-CMA in the presence of a decryption oracle adversary $\mathcal{A}$ against the combined scheme, we will construct an algorithm $\mathcal{B}$ that breaks the EUF-wCMA security of the above basic Boneh-Boyen signature scheme. More specifically, we will distinguish between two types of adversaries $\mathcal{A}_1$, $\mathcal{A}_2$, and for each type, we will define a separate breaking algorithm $\mathcal{B}_1$, $\mathcal{B}_2$. The two types of adversaries are defined as follows: Let $m_1, \ldots, m_{q_s}$ be the messages $\mathcal{A}$ adaptively queries to his signing oracle, and let (σ_i, r_i) be the signature returned in response to the ith query. Furthermore, let $w_i = 0||m_i + yr_i$ for each i, and let (m_*, σ_*, r_*) be the forgery output by $\mathcal{A}$. We say that $\mathcal{A}$ is a *type 1* adversary if

– $\mathcal{A}$ makes a decryption query on a ciphertext $c = (c_1, c_2, c_3)$ such that $1||h = -x$, where $h = \mathsf{H}(c_1)$, or
– $\mathcal{A}$ makes a signature query on a message m such that $0||m = -x$, or
– $\mathcal{A}$ outputs a forgery on a message m^* such that $0||m_* + yr_* \notin \{w_1, \ldots, w_{q_s}\}$

On the other hand, we say that $\mathcal{A}$ is a *type 2* adversary if

– $\mathcal{A}$ never makes a decryption query on a ciphertext $c = (c_1, c_2, c_3)$ such that $1||h = -x$ where $h = \mathsf{H}(c_1)$, and
– $\mathcal{A}$ never makes a signature query on a message m such that $0||m = -x$, and
– $\mathcal{A}$ outputs a forgery on a message m^* such that $0||m_* + yr_* \in \{w_1, \ldots, w_{q_s}\}$

It should be clear that a successful adversary against the combined scheme will either play the role of a type 1 or a type 2 adversary. We proceed by describing the breaking algorithm $\mathcal{B}_1$, $\mathcal{B}_2$ for each type of adversary.

Type 1 adversary. The algorithm $\mathcal{B}_1$ interacting with a type 1 adversary $\mathcal{A}_1$ is defined as follows:

Setup: Initially, $\mathcal{B}_1$ selects and outputs random $w_1, \ldots, w_q \in \mathbb{Z}_p$. $\mathcal{B}_1$ then receives a public key (g_1, g_2, u, v) and signatures $\sigma_1, \ldots, \sigma_{q_s}$. Let $u = g_1^x$ where $x \in \mathbb{Z}_p$. If $\sigma_i = 1$ for any i, $\mathcal{B}_1$ would have learned the private key $x = -w_i$ corresponding to (g_1, g_2, u, v), and will trivially be able to construct a valid forgery of the basic Boneh-Boyen signature scheme. Otherwise, we have that w_i is uniformly distributed in $\mathbb{Z}_p \setminus \{-x\}$ and that $e(g_1^{w_i} u, \sigma_i) = v = e(g_1, g_2)$ for all $i = 1, \ldots, q$. Lastly, $\mathcal{B}_1$ selects random $y \in \mathbb{Z}_p$, sets $X = u$ and $Y = g_1^y$, and passes the public key (g_1, g_2, X, Y) to $\mathcal{A}_1$.

Signing queries: In $\mathcal{A}_1$'s ith signature query on message $m_i \in \{0, 1\}^{n-1}$, $\mathcal{B}_1$ prepends a 0 to m_i to get $m'_i = 0||m_i$. Since $\mathcal{A}_1$ is a type 1 adversary, it might occur that $m'_i = -x$ which $\mathcal{B}_1$ can detect by checking that $g_1^{-m_i} = u$. In this case, $\mathcal{B}_1$ has learned the private key corresponding to (g_1, g_2, u, v), and no further interaction with $\mathcal{A}_1$ is needed since $\mathcal{B}_1$ can forge a signature on any message of his choice. Otherwise, $\mathcal{B}_1$ sets $r_i = (w_i - m'_i)/y$, and returns the signature (σ_i, r_i) to $\mathcal{A}_1$. Note that (σ_i, r_i) is a valid signature on m_i since

$$e(X \cdot g_1^{m'_i} \cdot Y^{r_i}, \sigma_i) = e(u \cdot g_1^{m'_i + r_i y}, \sigma_i) = e(u \cdot g_1^{w_i}, \sigma_i) = v = e(g_1, g_2),$$

and that r_i is uniformly distributed in $\mathbb{Z}_p \setminus \{-\frac{x+m'_i}{y}\}$ since w_i is uniformly distributed in $\mathbb{Z}_p \setminus \{-x\}$.

Decryption queries: Given a ciphertext $c = (c_1, c_3, c_3)$, $\mathcal{B}_1$ first computes $h = \mathsf{H}(c_1)$, and prepends a 1 to h to get $h' = 1||h$. Then $\mathcal{B}_1$ checks if the equation

$$e(c_2 c_1^{y^{-1}(w_1 - h')}, \sigma_1) = e(c_1^{y^{-1}}, g_2) \tag{1}$$

holds. If this is the case, then, due to the properties of the pairing and that $\sigma_1 = g_2^{1/(x+w_1)}$, we must have that

$$\left(c_2 c_1^{y^{-1}(w_1 - h')}\right)^{1/(x+w_1)} = c_1^{y^{-1}}$$

which implies that $c_2 = c_1^{(x+h')y^{-1}}$ i.e. c_1, c_2 must be of the form $c_1 = Y^s$, $c_2 = X^s g_2^{s \cdot h'}$ for some $s \in \mathbb{Z}_p$. Hence, $\mathcal{B}_1$ computes the key

$$K = e(c_1^{y^{-1}}, g_2) = e(g_1^s, g_2) = e(g_1, g_2)^s$$

and returns the message $m = \mathsf{DDec}(K, c_3)$. Otherwise, if equation (1) does not hold, it must be the case that $c_2 \neq c_1^{(x+h')y^{-1}}$, and $\mathcal{B}_1$ returns $\perp$ to $\mathcal{A}_1$.

Forgery: Eventually, $\mathcal{A}_1$ outputs a forgery (σ_*, r_*) on a message m_*. If $\mathcal{A}_1$ is successful, then $(m_*, \sigma_*, r_*) \neq (m_i, \sigma_i, r_i)$ for all i, and

$$z = e(g_1, g_2) = e(X g_1^{0||m_*} Y^{r_*}, \sigma_*) = e(u \cdot g_1^{0||m_* + y r_*}, \sigma_*).$$

Since $\mathcal{A}_1$ is a type 1 adversary, it must furthermore be the case that $0||m_* + yr_* \notin \{w_1, \ldots, w_{q_s}\}$. Hence, $\mathcal{B}_1$ outputs σ_* as a forgery on the message $0||m_* + yr_*$.

From the above description, it should be clear that if $\mathcal{A}_1$ is successful with probability ϵ and has running time t, then $\mathcal{B}_1$ will succeed with probability ϵ in time $t + \Theta(q_d T_p + (q_s + q_d) T_e)$ where T_p is the maximum time for evaluating a pairing, and T_e is the maximum time for computing an exponentiation in $\mathbb{G}_2$.

Type 2 adversary. The algorithm $\mathcal{B}_2$ which interacts with a type 2 adversary $\mathcal{A}_2$ is defined as follows:

Setup: Initially, $\mathcal{B}_2$ selects and outputs random $w_1, \ldots, w_{q_s} \in \mathbb{Z}_p^*$ (note that $\mathcal{B}_2$ selects $w_i \neq 0$ for all i), and receives a public key (g_1, g_2, u, v) and signatures $\sigma_1, \ldots, \sigma_{q_s}$. Let $u = g_1^y$. Like $\mathcal{B}_1$, $\mathcal{B}_2$ checks that $\sigma_i \neq 1$ for all i. If this is not the case, there must be an i such that $w_i = -y$, and $\mathcal{B}_2$ will trivially be able to construct a valid signature $\sigma = g_2^{1/(y+m)}$ on any message m of its choice. $\mathcal{B}_2$ then picks random $x \in \mathbb{Z}_p^*$, sets $X = g_1^x$ and $Y = u$, and passes the public key (g_1, g_2, X, Y) to $\mathcal{A}_2$.

Signing queries: In $\mathcal{A}_2$'s ith signing query on the message m_i, $\mathcal{B}_2$ sets $m_i' = 0||m_i$ and computes $r_i = (x + m_i')/w_i$. Note that $r_i \neq 0$ since a type 2 adversary will only submit messages $m_i' \neq -x$. $\mathcal{B}_2$ then stores the tuple (m_i, r_i) for later use, and returns the signature $(\sigma_i^{1/r_i}, r_i)$. Note that this is a valid signature since

$$e(X g_1^{m_i'} Y^{r_i}, \sigma_i^{1/r_i}) = e(g_1^{x+m_i'} u^{r_i}, \sigma_i^{1/r_i}) = e(g_1^{(x+m)/r_i} u, \sigma_i) = e(g_1^{w_i} u, \sigma_i) = v = e(g_1, g_2)$$

and that r_i is uniform in $\mathbb{Z}_p^* \setminus \{-\frac{x+m}{y}\}$ since w_i is uniform in $\mathbb{Z}_p^* \setminus \{-y\}$. However, since r_i would be distributed uniformly in $\mathbb{Z}_p \setminus \{-\frac{x+m}{y}\}$ in a real interaction, there is a statistical distance of $1/p$ from the correct distribution. Hence, $\mathcal{B}_2$'s simulation of all signature queries will at most have a statistical distance of q_s/p from the correct distribution.

Decryption queries: Given a ciphertext $c = (c_1, c_2, c_3)$, $\mathcal{B}_2$ computes $h = \mathsf{H}(c_1)$, prepends a 1 to h to get $h' = 1||h$. Note that since $\mathcal{A}_2$ is a type 2 adversary, $h' \neq -x$. $\mathcal{B}_2$ then checks that the equation

$$e(c_1^{(x+h')} c_2^{w_1}, \sigma_1) = e(c_2, g_2) \tag{2}$$

holds. If this is the case, then, due to the properties of the pairing and that $\sigma_1 = g_2^{1/(y+w_1)}$, we must have

$$\left(c_1^{(x+h')} c_2^{w_1}\right)^{1/(y+w_1)} = c_2$$

which implies that $c_2 = c_1^{(x+h')/y}$ and that c_1, c_2 are of the form $c_1 = Y^s$, $c_2 = X^s g_1^{h's}$ for some $s \in \mathbb{Z}_p$. In this case, $\mathcal{B}_2$ computes the key

$$K = e(c_2^{(x+h')^{-1}}, g_2) = e(g_1^s, g_2) = e(g_1, g_2)^s$$

and returns the message $m = \mathsf{DDec}(K, c_3)$. Otherwise, if equation (2) does not hold, then $c_2 \neq c_1^{(x+h')/y}$, and $\mathcal{B}_2$ returns $\perp$ to $\mathcal{A}_2$.

Forgery: Eventually, $\mathcal{A}_2$ returns a forgery (σ_*, r_*) on a message m_*. If $\mathcal{A}_2$ is successful, we must have that $(m_*, \sigma_*, r_*) \neq (m_i, \sigma_i, r_i)$ for all i which implies that $(m_*, r_*) \neq (m_i, r_i)$ for all i. Furthermore, since $\mathcal{A}_2$ is a type 2 adversary, we must have that $0||m_* + yr_* = 0||m_i + yr_i$ for some i. Hence, $\mathcal{B}_2$ can recover $y = (0||m_* - 0||m_i)/(r_i - r_*)$ which is the private key corresponding to the given public key (g_1, g_2, u, v), and $\mathcal{B}_2$ will therefore be able to construct a forgery on any message of its choice.

From the above description it should be clear that if $\mathcal{A}_2$ succeeds with probability ϵ, then $\mathcal{B}_2$ succeeds with probability at least $\epsilon - q_s/p$, and if $\mathcal{A}_2$ runs in time t, then $\mathcal{B}_2$ runs in time $t + \Theta(q_d T_p + (q_s + q_d)T_e)$ where T_p is the maximum time for evaluating a pairing and T_e is the maximum time of computing an exponentiation in $\mathbb{G}_1$ and $\mathbb{G}_2$.

To obtain an algorithm $\mathcal{B}$ which breaks the basic Boneh-Boyen signature scheme using an arbitrary adversary $\mathcal{A}$ whose type is unknown, it is sufficient to let $\mathcal{B}$ run either $\mathcal{B}_1$ or $\mathcal{B}_2$ with equal probability. Assuming $\mathcal{A}$ breaks the (t, q_d, q_s, ϵ)-security of the combined scheme, this will yield an algorithm $\mathcal{B}$ with running time $t + \Theta(q_d T_p + (q_s + q_d)T_e)$ and success probability at least $\frac{1}{2}\min(\epsilon, \epsilon - q_s/p) = (\epsilon - q_s/p)/2$. Combining this with the above mentioned result of Lemma 9 in [6] yields the bounds given in the theorem.

Proof of Theorem 4: To prove the theorem, we consider an adversary $\mathcal{A}$ against the IND-CCA security of the combined public key scheme, and the following sequence of games.

Game$_0$**:** This is the original IND-CCA in the presence of a signing oracle experiment.

Game$_1$**:** In this game, the DEM key used to encrypt the challenge messages m_b and the corresponding part of the challenge ciphertext is generated before interacting with $\mathcal{A}$. More specifically, after generating a public/private key pair $(pk, sk) = ((g_1, g_2, X, Y), (x, y))$, a random $s \in \mathbb{Z}_p$ is picked and the values $c_1^* = Y^s$, $h_* = \mathsf{H}(c_1^*)$, $h'_* = 1||h_*$, $c_2^* = X^s g_1^{h'_* \cdot s}$, $K^* = e(g_1, g_2)^s$ are computed. Then pk is passed to $\mathcal{A}$, and when $\mathcal{A}$ submits (m_0, m_1), the challenge ciphertext is set to $c^* = (c_1^*, c_2^*, c_3^*)$ where $c_3^* = \mathsf{DEnc}(K^*, m_b)$ and b is the random bit chosen in the experiment.

Game$_2$**:** In this game, if $\mathcal{A}$ submits a decryption query $c = (c_1, c_2, c_3)$ to the decryption oracle where $c_1 \neq c_1^*$ but $h_* = \mathsf{H}(c_1)$, the decryption oracle returns $\perp$ to $\mathcal{A}$.

Game$_3$**:** In this game, the construction of the private key is changed as follows. Firstly, random values $\alpha, a, s \leftarrow \mathbb{Z}_p$ are picked, and the private and public key components y and Y are assigned values $y \leftarrow \alpha$ and $Y \leftarrow g_1^y$. Then, the challenge ciphertext components $c_1^* = Y^s$, $h_* = \mathsf{H}(c_1^*)$, and $h'_* = 1||h_*$ are computed, as well as the value $b \leftarrow h'_*/a$. Lastly, the remaining public and private key components x and X are assigned values $x \leftarrow -a(\alpha + b)$ and $X \leftarrow g_1^x$. Furthermore, when responding to the signature queries by $\mathcal{A}$, the randomness r will be picked uniformly at random from $\mathbb{Z}_p \setminus \{a\}$ instead of $\mathbb{Z}_p$.

Game$_4$**:** In the last game, the challenge key K^* is replaced by a randomly chosen key $K' \in \mathbb{G}_T$. Furthermore, if $\mathcal{A}$ submits a decryption query of the form (c_1^*, c_2^*, c_3) where $c_3 \neq c_3^*$, the decryption oracle will decrypt c_3 using K'.

Let E_i denote the event that $\mathcal{A}$ correctly guesses the challenge bit b in *Game*$_i$. The advantage of $\mathcal{A}$ against the scheme can be expressed as follows:

$$|\Pr[E_0] - 1/2| \leq \sum_{i=0}^{3} |\Pr[E_i] - \Pr[E_{i+1}]| + |\Pr[E_4] - 1/2|$$

Since the view of $\mathcal{A}$ is identical in *Game*$_0$ and *Game*$_1$, it follows that $\Pr[E_0] = \Pr[E_1]$. To complete the proof, we show the following claims.

Claim. $|\Pr[E_1] - \Pr[E_2]| < \epsilon_h$

Proof. Note that the games $Game_1$ and $Game_2$ are identical unless $\mathcal{A}$ makes a decryption query such that $c_1 \neq c_1^*$ but $\mathsf{H}(c_1) = \mathsf{H}(c_1^*)$. Hence, it is sufficient to show that if $\mathcal{A}$ makes such a query, it is possible to break the aSec security of H. For this purpose, we construct the algorithm $\mathcal{B}$ which uses $\mathcal{A}$ as a building block. $\mathcal{B}$ is defined as follows:

Initially, $\mathcal{B}$ is given a random element $z \in \mathbb{G}_1$, and the goal of $\mathcal{B}$ is to produce an element $z' \in \mathbb{G}_1$ such that $\mathsf{H}(z) = \mathsf{H}(z')$. Firstly, to generate a public/private key pair, $\mathcal{B}$ picks random generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and random integers $x, y \in \mathbb{Z}_p$, and sets $pk = (g_1, g_2, g_1^x, g_2^y)$ and $sk = (x, y)$. Then, to generate challenge ciphertext components c_1^*, c_2^*, K^*, $\mathcal{B}$ sets $c_1^* = z$, $h_* = \mathsf{H}(c_1^*)$, $h'_* = 1||h_*$, $c_2^* = z^{(x+h'_*)/y}$, and $K^* = e(z^{1/y}, g_2)$. Note that since z is a random element of $\mathbb{G}_1$, c_1^*, c_2^*, K^* are distributed exactly as in $Game_1$ and $Game_2$. Finally, $\mathcal{B}$ runs $\mathcal{A}$ with input pk.

If $\mathcal{A}$ submits a decryption query $c = (c_1, c_2, c_3)$, $\mathcal{B}$ checks whether $\mathsf{H}(c_1) = \mathsf{H}(c_1^*)$ and $c_1 \neq c_1^*$. If this is the case, $\mathcal{B}$ outputs $z' = c_1$ and halts. Otherwise, $\mathcal{B}$ returns $\mathsf{Decrypt}(sk, c)$ to $\mathcal{A}$. Likewise, if $\mathcal{A}$ submits a signature query m, $\mathcal{B}$ returns $\mathsf{Sign}(sk, m)$. At some point, $\mathcal{A}$ submits challenge messages m_0, m_1. $\mathcal{B}$ completes the challenge ciphertext by picking random $b \leftarrow \{0,1\}$ and setting $c_3^* = \mathsf{DEnc}(K^*, m_b)$, and then returns $c^* = (c_1^*, c_2^*, c_3^*)$ to $\mathcal{A}$. Note that c^* will be distributed as in $Game_1$ and $Game_2$. Queries made by $\mathcal{A}$ after the challenge phase are answered as above. If $\mathcal{A}$ terminates with output a bit b', $\mathcal{B}$ aborts.

It should be clear from the above description that $\mathcal{B}$ succeeds whenever $\mathcal{A}$ makes a decryption query of the described form. Hence, the claim follows. □

Claim. $|\Pr[E_2] - \Pr[E_3]| < q_s/p$

Proof. Note that the public key components X, Y computed in $Game_3$ are independent of the challenge ciphertext components, and are distributed as random elements in $\mathbb{G}_1$. Hence, the only difference between $Game_2$ and $Game_3$ is that the randomness used in the construction of signatures is picked from $\mathbb{Z}_p \setminus \{a\}$ instead of $\mathbb{Z}_p$, where a is a random element of $\mathbb{Z}_p$. This implies that a signature created in $Game_3$ is distributed with a statistical difference of $1/p$ from the distribution of signatures in $Game_2$. Since $\mathcal{A}$ asks for q_s signatures, the signatures created in $Game_3$ will be jointly distributed with a statistical difference of at most q_s/p from the signatures constructed in $Game_2$. Hence, the claim follows. □

Claim. $|\Pr[E_3] - \Pr[E_4]| < \epsilon_{dhi}$

Proof. Let $\mathcal{A}$ be an adversary which correctly guesses the challenge bit with a different probability in $Game_2$ and $Game_3$, and let $\epsilon_{\mathcal{A}} = |\Pr[E_3] - \Pr[E_4]|$. Using $\mathcal{A}$, we will construct an algorithm $\mathcal{B}$ which solves the q-DBDHI problem with advantage $\epsilon_{\mathcal{A}}$.

Initially, $\mathcal{B}$ is given a description of a pairing $\mathcal{G} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p)$ and an instance of the q-DBDHI problem $(g_1, g_1^\alpha, g_2, g_2^\alpha, g_2^{\alpha^2}, \ldots, g_2^{\alpha^q}, , T) \in \mathbb{G}_1^2 \times \mathbb{G}_2^{q+1} \times \mathbb{G}_T$. The goal of $\mathcal{B}$ is to output a bit β which indicates whether $T = e(g_1, g_2)^{1/\alpha}$ or T is a random element of $\mathbb{G}_T$. We construct $\mathcal{B}$ as follows:

Setup: Firstly, $\mathcal{B}$ constructs a generator g'_2 of $\mathbb{G}_2$ for which it knows a set of q values of the form $(w_i, (g'_2)^{1/(w_i+\alpha)})$. This is done as follows: $\mathcal{B}$ picks random $w_1, \ldots, w_q \in \mathbb{Z}_p^*$, defines the polynomial $f(z) = \prod_{i=1}^q (z + w_i)$, and expands $f(z) = \sum_{i=0}^q c_i z^i$ to obtain the coefficients $c_0, \ldots, c_q \in \mathbb{Z}_p$. Then $\mathcal{B}$ computes the generator

$$g'_2 = \prod_{i=1}^{q} \left(g_2^{\alpha^i}\right)^{c_i} = g_2^{f(\alpha)} \in \mathbb{G}_2.$$

We assume that $g'_2 \neq 1$ (if this is not the case, we must have that $w_i = -\alpha$ for some i, and $\mathcal{B}$ would be able to solve the q-DBDHI problem without interacting with $\mathcal{A}$). Observe that $\mathcal{B}$ can compute $(g'_2)^{1/(w_i+\alpha)}$ for any of the above chosen w_i by expanding the polynomial $f_i(z) = f(z)/(z + w_i) = \sum_{i=0}^{q-1} d_i z^i$ and computing $g_2^{f_i(\alpha)} = \prod_{i=0}^{q-1} (g_2^{\alpha^i})^{d_i} = (g'_2)^{1/(w_i+\alpha)}$.
$\mathcal{B}$ proceeds by picking random $a, l \in \mathbb{Z}_p$ and computing challenge ciphertext components $c_1^* = g_1^l$, $h^* = \mathsf{H}(c_1^*)$, $h'_* = 1||h^*$, and $c_2^* = g_1^{-al}$, as well as the value $b = h'_*/a$. Furthermore, $\mathcal{B}$ computes the challenge key $K^* = T^{lc_0} \prod_{i=0}^{q-1} e(g_1, g_2^{\alpha^i})^{lc_{i+1}}$.
Lastly, $\mathcal{B}$ computes the public key components $X = (g_1^\alpha)^{-a} g_1^{-ab}$ and $Y = g_1^\alpha$. This will implicit define the private key as $(x, y) = (-a(\alpha + b), \alpha)$. Define $s = l/\alpha$. Observe that

$$c_1^* = g_1^l = Y^{l/\alpha} = Y^s$$
$$c_2^* = g_1^{-al} = g_1^{-a\alpha(l/\alpha)} = g_1^{(x+ab)(l/\alpha)} = X^s g_1^{s \cdot h'_*}$$

Furthermore, if $T = e(g_1, g_2)^{1/\alpha}$, then $K^* = e(g_1, g_2)^{lf(\alpha)/\alpha} = e(g_1, g'_2)^{l/\alpha} = e(g_1, g'_2)^s$. On the other hand, if T is random in $\mathbb{G}_T$, then so is K^*.
Lastly, $\mathcal{B}$ passes the public key $pk = (g_1, g'_2, X, Y)$ to $\mathcal{A}$.

Signing queries: To respond to $\mathcal{A}$'s ith signing query m_i, $\mathcal{B}$ retrieves the ith pair $((g'_2)^{1/(\alpha+w_i)}, w_i)$ generated in the setup phase. Let $h_i = (g'_2)^{1/(\alpha+w_i)}$. Then $\mathcal{B}$ sets $m'_i = 0||m_i$, computes $r = a + \frac{m'_i - ab}{w_i}$ and returns the signature $\sigma = (h_i^{1/(r-a)}, r)$. Note that $r - a \neq 0$ since, due to a different prefix, $m'_i \neq h'_* = ab$. Furthermore, observe that σ is a valid signature since

$$h_i^{1/(r-a)} = (g'_2)^{1/(r-a)(\alpha+w_i)} = (g'_2)^{1/(r\alpha+m'_i-ab-a\alpha)} = (g'_2)^{1/(ry+m'_i+x)},$$

and that r is distributed uniformly at random among the elements in $\mathbb{Z}_p$ for which $m'_i + x + ry \neq 0$ and $r \neq a$, since w_i is uniformly distributed in $\mathbb{Z}_p \setminus \{0, -\alpha\}$.

Decryption queries: If $\mathcal{A}$ submits a decryption query $c = (c_1, c_2, c_3)$, $\mathcal{B}$ first computes $h = \mathsf{H}(c_1)$ and sets $h' = 1||h$. If either
- $c_1 \neq c_1^*$ but $h = h^*$, or
- $e(c_1, \widehat{X} g_2^{h'}) \neq e(c_2, \widehat{Y})$ where $\widehat{X} = (g_2^\alpha)^{-a} g_2^{-ab} = g_2^x$ and $\widehat{Y} = g_2^\alpha = g_2^y$ (note that this implies that $c_2 \neq c_1^{(x+h')/y}$).

$\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Furthermore, if $(c_1, c_2) = (c_1^*, c_2^*)$, then $\mathcal{B}$ returns $m = \mathsf{DDec}(K^*, c_3)$ to $\mathcal{A}$.
Otherwise, we must have that c_1, c_2 are of the form $c_1 = Y^s$, $c_2 = X^s g_1^{h' \cdot s}$ for some $s \in \mathbb{Z}_p$ and that $h' \neq h'_*$. Note that $c_2 c_1^a = X^s g_1^{s \cdot h'} Y^{sa} = g_1^{s(x+h'+ay)} = g_1^{s(-a(\alpha+b)+h'+a\alpha)} = g_1^{s(h'-h'_*)}$. Hence, $\mathcal{B}$ computes the key

$$K = e((c_2 c_1^a)^{(h'-h'_*)^{-1}}, g'_2) = e(g_1^s, g'_2) = e(g_1, g'_2)^s,$$

and returns the message $m = \mathsf{DDec}(K, c_3)$.

Challenge: At some point, $\mathcal{A}$ submits challenge messages (m_0, m_1) of equal length. $\mathcal{B}$ responds by picking a random bit $b \in \{0, 1\}$, and returning the challenge ciphertext $c^* = (c_1^*, c_2^*, c_3^*)$ where c_1^*, c_2^* were computed in the setup phase, and $c_3^* = \mathsf{DEnc}(K^*, m_b)$. After receiving the challenge ciphertext c^*, $\mathcal{A}$ can submit additional signing and decryption queries, which $\mathcal{B}$ answers as above. Eventually, $\mathcal{A}$ outputs a bit b'. If $b' = b$, $\mathcal{B}$ returns $\beta = 1$ indicating that $T = e(g_1, g_2)^{1/\alpha}$. Otherwise, $\mathcal{B}$ returns $\beta = 0$.

Let *Real* denote the event that $T = e(g_1, g_2)^{1/\alpha}$ and let *Random* denote the event that T is random in $\mathbb{G}_T$. Note that if *Real* occurs, then $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of $Game_3$ whereas if *Random* occurs, then $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of $Game_4$. The advantage of $\mathcal{B}$ in solving the q-DBDHI problem is given by

$$\begin{aligned} |\Pr[\beta = 1|Real] - \Pr[\beta = 1|Random]| &= |\Pr[b' = b|Real] - \Pr[b' = b|Random]| \\ &= |\Pr[E_3] - \Pr[E_4]| \\ &= \epsilon_{\mathcal{A}} \end{aligned}$$

Hence, the claim follows. □

Claim. $|\Pr[E_4] - 1/2| < \epsilon_{dem}$

Proof. Assume that an adversary with advantage $\epsilon_{\mathcal{A}}$ in $Game_4$ exists i.e. $\epsilon_{\mathcal{A}} = |\Pr[E_4] - 1/2|$. Using $\mathcal{A}$, we will construct an algorithm $\mathcal{B}$ with advantage $\epsilon_{\mathcal{A}}$ against the DEM. $\mathcal{B}$ is constructed as follows:

Setup: $\mathcal{B}$ generates the public and private key (pk, sk) and the challenge ciphertext components c_1^*, c_2^* as specified in $Game_4$. The random key K' will correspond to the key of the DEM $\mathcal{B}$ is attacking. $\mathcal{B}$ runs $\mathcal{A}$ with input pk.

Signing queries: Given a message m from $\mathcal{A}$, $\mathcal{B}$ simply computes a signature σ on m by using sk, but follows the restriction on the choice of randomness required in $Game_4$.

Decryption queries: Upon a query (c_1, c_2, c_3) from $\mathcal{A}$, $\mathcal{B}$ responds as follows:
- If $c_1 \neq c_1^*$ but $\mathsf{H}(c_1) = \mathsf{H}(c_1^*)$, $\mathcal{B}$ returns $\perp$ as required in $Game_4$.
- Otherwise, if $(c_1, c_2) \neq (c_1^*, c_2^*)$, $\mathcal{B}$ uses $sk = (x, y)$ to check if $c_2 = c_1^{(x+h')/y}$ where $h' = 1||h$ and $h = \mathsf{H}(c_1)$, and if this is not the case, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ computes $K = e(c_1, g_2^{1/y})$ and returns $m = \mathsf{DDec}(K, c_3)$.

– Lastly, if $(c_1, c_2) = (c_1^*, c_2^*)$, $\mathcal{B}$ submits c_3 to his decryption oracle and returns the obtained message m to $\mathcal{A}$.

Challenge: Eventually, $\mathcal{A}$ outputs challenge messages m_0, m_1 of equal length. $\mathcal{B}$ submits m_0, m_1 as his own challenge messages to obtain c_3^*, and returns (c_1^*, c_2^*, c_3^*) to $\mathcal{A}$. After receiving the challenge, $\mathcal{A}$ can ask additional queries which $\mathcal{B}$ responds to as above. Finally, $\mathcal{A}$ will output a bit b which $\mathcal{B}$ forwards as his own response to the DEM challenge.

From the above description it should be clear that $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation of $Game_4$ and that $\mathcal{B}$ will succeed in guessing the challenge bit whenever $\mathcal{A}$ does. Hence, the claim follows. □

Combining the above claims yields the bound on the success probability of an IND-CCA in the presence of a decryption oracle adversary against the scheme. □

Proof of Theorem 5: Since large parts of this proof overlap with the proof of Theorem 4, we will only sketch the modifications required to obtain a proof of the extended scheme. However, for clarity, we provide the full description of the sequence of games used to obtain the proof:

Game$_0$**:** This is the original IND-tag-CCA in the presence of a signing oracle experiment.

Game$_1$**:** Like in $Game_1$ of Theorem 4, we precompute the challenge ciphertext components c_1^*, c_2^* and the challenge keys K_d^*, K_m^* before interacting with $\mathcal{A}$ i.e. after generating the public/private key pair $(pk, sk) = ((g_1, g_2, X, Y), (x, y))$, a random $s \in \mathbb{Z}_p$ is picked and the values $c_1^* = Y^s$, $h_* = \mathsf{H}(c_1^*)$, $h'_* = 1||h_*$, $c_2^* = X^s g_1^{h'_* \cdot s}$, $K^* = e(g_1, g_2)^s$ and $(K_d^*, K_m^*) = \mathsf{KDF}(K^*)$. Then $\mathcal{A}$ is run on input pk. When $\mathcal{A}$ submits the challenge messages m_0, m_1 and the challenge tag t^*, the components $c_3^* = \mathsf{DEnc}(K_d^*, m_b)$ and $c_4^* = \mathsf{MAC}(K_m^*, t^*)$ are computed, where b is the challenge bit chosen in the experiment, and the challenge ciphertext $c^* = (c_1^*, c_2^*, c_3^*, \tau^*)$ is returned to $\mathcal{A}$.

Game$_2$**:** In this game, if $\mathcal{A}$ submits a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and tag t such that $c_1 \neq c_1^*$ but $\mathsf{H}(c_1) = \mathsf{H}(c_1^*)$, the decryption oracle returns $\perp$ to $\mathcal{A}$.

Game$_3$**:** In this game, if $\mathcal{A}$ submits a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and tag t in Phase 1 such that $c_1 = c_1^*$, the decryption oracle returns $\perp$ to $\mathcal{A}$.

Game$_4$**:** In this game, if $\mathcal{A}$ submits a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and a tag t in Phase 2 such that $(c_1, c_2) = (c_1^*, c_2^*)$ but $(\tau, t) \neq (\tau^*, t^*)$, the decryption oracle returns $\perp$ to $\mathcal{A}$.

Game$_5$**:** Like in $Game_3$ of Theorem 4, the construction of the private key is changed as follows. Firstly, random values $\alpha, a, s \leftarrow \mathbb{Z}_p$ are picked, and the private and public key components y and Y are assigned values $y \leftarrow \alpha$ and $Y \leftarrow g_1^y$. Then, the challenge ciphertext components $c_1^* = Y^s$, $h_* = \mathsf{H}(c_1^*)$, and $h'_* = 1||h_*$ are computed, as well as the value $b \leftarrow h'_*/a$. Lastly, the remaining public and private key components x and X are assigned values $x \leftarrow -a(\alpha + b)$ and $X \leftarrow g_1^x$. Furthermore, when responding to the signature queries by $\mathcal{A}$, the randomness r will be picked uniformly at random from $\mathbb{Z}_p \setminus \{a\}$ instead of $\mathbb{Z}_p$.

Game$_6$**:** In this game, the value K^* used as input to KDF when generating the challenge keys (K_d^*, K_m^*), is replaced by a random value in $\mathbb{G}_T$. Furthermore, if $\mathcal{A}$ submits a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and a tag t such that $(c_1, c_2, \tau, t) = (c_1^*, c_2^*, \tau^*, t^*)$, the DEM key derived from the randomly chosen K^* is used to decrypt c_3 i.e. the decryption oracle computes $(K_d^*, K_m^*) = \mathsf{KDF}(K^*)$ and returns the output of $\mathsf{DDec}(K_d^*, c_3)$.

Game$_7$**:** In this game, the keys $(K_d^*, K_m^*) = \mathsf{KDF}(K^*)$ are replaced by keys chosen uniformly at random in $\mathcal{K}_d \times \mathcal{K}_m$. Furthermore, if $\mathcal{A}$ submits a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and a tag t such that $(c_1, c_2, \tau, t) = (c_1^*, c_2^*, \tau^*, t^*)$, the randomly chosen DEM key K_d^* is used directly to decrypt c_3 i.e. the decryption oracle just returns the output of $\mathsf{DDec}(K_d^*, c_3)$.

Let E_i denote the probability that the adversary $\mathcal{A}$ succeeds in guessing the challenge bit in $Game_i$. Furthermore, let F_i be the event that, in Phase 2 of $Game_i$, $\mathcal{A}$ submits a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and tag t such $(c_1, c_2) = (c_1^*, c_2^*)$, $(\tau, t) \neq (\tau^*, t^*)$ and $\mathsf{MVer}(K_m^*, t, \tau) = 1$. The advantage of the adversary is given by

$$|\Pr[E_0] - 1/2| \leq \sum_{i=0}^{6} |\Pr[E_i] - \Pr[E_{i+1}] + |\Pr[E_7] - 1/2|$$

Note that $Game_3$ and $Game_4$ are identical unless $\mathcal{A}$ submits a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and tag t such that $(c_1, c_2) = (c_1^*, c_2^*)$, $(\tau, t) \neq (\tau^*, t)$ and $\mathsf{Dec}(sk, c, t) \neq \perp$. Since $\mathsf{Dec}(sk, c, t) \neq \perp$ requires that $\mathsf{MVer}(K_m^*, t, \tau) = 1$, we must have

$$|\Pr[E_3] - \Pr[E_4]| \leq \Pr[F_4] \leq |\Pr[F_4] - \Pr[F_5]| + |\Pr[F_5] - \Pr[F_6]| + |\Pr[F_6] - \Pr[F_7]| + \Pr[F_7]$$

To complete the proof, we show the following claims.

Claim. $\Pr[E_0] = \Pr[E_1]$, $|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_h$, $|\Pr[E_4] - \Pr[E_5]| \leq q_s/p$, $|\Pr[F_4] - \Pr[F_5]| \leq q_s/p$, $|\Pr[E_5] - \Pr[E_6]| \leq \epsilon_{dhi}$, and $|\Pr[E_7] - 1/2| \leq \epsilon_{dem}$

These claims can be shown using almost identical arguments to the ones used for the corresponding claims in the proof of Theorem 4.

Claim. $|\Pr[E_2] - \Pr[E_3]| \leq q_d/p$

Proof. Observe that $Game_2$ and $Game_3$ will be identical unless $\mathcal{A}$ submits a decryption query in Phase 1 such that $c_1 = c_1^*$. However, since c_1^* is perfectly hidden from $\mathcal{A}$ before the challenge phase, this will happen with probability at most $1/p$ for a single query. Since $\mathcal{A}$ makes at most q_d queries, the probability that $c_1 = c_1^*$ in any of $\mathcal{A}$'s queries in Phase 1 is upper bounded by q_d/p. □

Claim. $|\Pr[F_5] - \Pr[F_6]| \leq \epsilon_{dhi}$

Proof. The proof of this claim is only slightly different from the proof of $|\Pr[E_5] - \Pr[E_6]| \leq \epsilon_{dhi}$ since we are interested in bounding the probability of the adversary forging a MAC instead of correctly guessing the challenge bit. More specifically, given a DBDHI instance and an adversary $\mathcal{A}$, we create a simulator $\mathcal{B}$ identical to that used for showing $|\Pr[E_4] - \Pr[E_5]| \leq \epsilon_{dhi}$ which in turn is almost identical to the simulator used in the proof of the corresponding claim in Theorem 4. However, instead of using the bit b' output by $\mathcal{A}$, $\mathcal{B}$ checks all decryption queries made by $\mathcal{A}$, and if any of the queries satisfy $(c_1, c_2) = (c_1^*, c_2^*)$, $(\tau, t) \neq (\tau^*, t^*)$, and $\mathsf{MVer}(K_m^*, t, \tau) = 1$, $\mathcal{B}$ outputs 1. Otherwise $\mathcal{B}$ outputs 0.

Depending on whether the value T given in the DBDHI problem corresponds to $e(g_1, g_2)^{1/\alpha}$ or a random value, $\mathcal{B}$ will either simulate $Game_5$ or $Game_6$ for $\mathcal{A}$. Furthermore, note that $\mathcal{B}$ outputs 1 only if the event F_i occurs, $i = 5, 6$. Hence, if $|\Pr[F_5] - \Pr[F_6]| = \epsilon_{\mathcal{A}}$, then $\mathcal{B}$ will solve the DBDHI problem with probability $\epsilon_{\mathcal{A}}$. □

Claim. $|\Pr[E_6] - \Pr[E_7]| \leq \epsilon_{kdf}$ and $|\Pr[F_6] - \Pr[F_7]| \leq \epsilon_{kdf}$

Proof. The only difference between $Game_6$ and $Game_7$ is that in $Game_6$, the keys K_d^*, K_m^* are derived from KDF whereas they are picked uniformly at random from the appropriate keyspaces in $Game_7$. Furthermore, since the input to KDF is a key K^* chosen uniformly at random, the claim follows from a simple reduction to the security of KDF. □

Claim. $\Pr[F_7] \leq q_d \epsilon_{mac}$

Proof. Let $\mathcal{A}$ be an adversary making q_d decryption queries and let $\epsilon_{\mathcal{A}} = \Pr[F_7]$. Using $\mathcal{A}$, we will construct an algorithm $\mathcal{B}$ that breaks the unforgeability of the MAC with probability $\epsilon_{\mathcal{A}}/q_d$. $\mathcal{B}$ is constructed as follows:

Setup $\mathcal{B}$ computes the public and private keys (pk, sk) and the challenge ciphertext components c_1^*, c_2^* as dictated in $Game_7$, and lastly pick a random DEM key $K_d^* \in \mathcal{K}_d$. The MAC key K_m^* will correspond to the key of $\mathcal{B}$'s challenger. Then $\mathcal{B}$ passes pk to $\mathcal{A}$.

Phase 1 Given a signature query m, $\mathcal{B}$ simply constructs a signature σ on m using sk, but follows the restriction on the used randomness which is required in $Game_7$. Given a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and a tag t, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ if $c_1 = c_1^*$ or $c_1 \neq c_1^*$ but $\mathsf{H}(c_1) = \mathsf{H}(c_1^*)$ as required in $Game_1$. Otherwise, $\mathcal{B}$ simply returns the output of $\mathsf{Dec}(sk, c, t)$. Note that since all queries for which $c_1 = c_1^*$ is rejected in Phase 1, $\mathcal{B}$ will not be required to use K_m^* to verify any MAC.

Challenge At some point, $\mathcal{A}$ submits challenge messages m_0, m_1 and a challenge tag t^*. $\mathcal{B}$ flips a bit $b \leftarrow \{0, 1\}$, computes $c_3^* = \mathsf{DEnc}(K_d^*, m_b)$, submits t^* to his MAC oracle to obtain τ^*, and finally returns $c^* = (c_1^*, c_2^*, c_3^*, \tau^*)$ to $\mathcal{A}$.

Phase 2 $\mathcal{B}$ responds to signature queries as in Phase 1. Given a decryption query consisting of $c = (c_1, c_2, c_3, \tau)$ and a tag t, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ if $c_1 \neq c_1^*$ but $\mathsf{H}(c_1) = \mathsf{H}(c_1^*)$ or $(c_1, c_2) = (c_1^*, c_2^*)$ but $(\tau, t) \neq (\tau^*, t^*)$ as required in $Game_7$. If $(c_1, c_2, \tau, t) = (c_1^*, c_2^*, \tau^*, t^*)$, $\mathcal{B}$ returns the output of $\mathsf{DDec}(K_d^*, c_3)$, and otherwise return the output of $\mathsf{Dec}(sk, c, t)$. At some point, $\mathcal{A}$ outputs a bit b' which $\mathcal{B}$ discards. $\mathcal{B}$ then picks a random decryption query submitted by $\mathcal{A}$, and returns (τ, t) as his forgery.

From the above description, it should be clear that $\mathcal{B}$ provides a perfect simulation of $Game_7$ to $\mathcal{A}$. Furthermore, if F_7 occurs, at least one of $\mathcal{A}$'s decryption queries must have the property that $(c_1, c_2) = (c_1^*, c_2^*)$, $(\tau, t) \neq (\tau^*, t^*)$, and $\mathsf{MVer}(K_m^*, t, \tau) = 1$. With probability at least $1/q_d$, such query will be picked and returned by $\mathcal{B}$. Hence, $\mathcal{B}$ succeeds in breaking the unforgeability of the MAC with probability $\epsilon_{\mathcal{A}}/q_d$, and thus the claim follows. $\square$

Combining the above claims yields the bound on the success probability of an IND-tag-CCA in the presence of a signing oracle adversary against the scheme. $\square$

C Asymmetric Gentry IBE

The IBE scheme of Gentry [18] can be extended naturally to the setting of asymmetric pairings $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ in the following way.

Setup: The PKG picks random generators $X \in \mathbb{G}_1, Y, H \in \mathbb{G}_2$ and random $\alpha \in \mathbb{Z}_p$. It sets $A = X^\alpha \in \mathbb{G}_1$. The public mpk and private msk are given by

$$mpk = (X, A, H, Y) \qquad msk = \alpha.$$

Extract: To generate a private key for identity $ID \in \mathbb{Z}_p$, the PKG generates random $r_{ID} \in \mathbb{Z}_p$, and outputs the private key

$$d_{ID} = (r_{\mathsf{ID}}, H_{\mathsf{ID}}), \text{ where } H_{ID} = \left(HY^{-r_{ID}}\right)^{1/(\alpha - ID)}.$$

If $ID = \alpha$, the PKG aborts. We require that the PKG always use the same random value r_{ID} for ID. This can be accomplished, for example, using a PRF or an internal log to ensure consistency.

Encrypt: To encrypt $m \in \mathbb{G}_T$ using identity $ID \in \mathbb{Z}_p$, the sender generates random $s \in \mathbb{Z}_p$ and sends the ciphertext

$$C = \left(A^s \cdot X^{-s \cdot ID}, e(X, Y)^s, m \cdot e(X, H)^{-s}\right) \in \mathbb{G}_1 \times \mathbb{G}_T \times \mathbb{G}_T.$$

Decrypt: To decrypt ciphertext $C = (U, V, W)$ with ID, the recipient outputs

$$m = W \cdot e(U, H_{ID}) \cdot V^{r_{ID}}$$

Correctness: Assuming the ciphertext is well-formed for ID:

$$e(U, H_{ID}) \cdot V^{r_{ID}} = e(X^{s(\alpha - ID)}, H^{1/(\alpha - ID)} \cdot Y^{-r_{ID}/(\alpha - ID)}) \cdot e(X, Y)^{s r_{ID}} = e(X, H)^s,$$

as required.

The security of this scheme depends on a problem closely related to the truncated decision q-augmented bilinear Diffie-Hellman exponent problem (q-ABDHE). An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the problem if

$$\Big| \Pr[\mathcal{B}(T, T^{\alpha^{q+2}}, X, X^\alpha, Y, Y^\alpha, \dots, Y^{\alpha^q}, e(T, Y^{\alpha^{q+1}})) = 0] - \Pr[\mathcal{B}(T, T^{\alpha^{q+2}}, X, X^\alpha, Y, Y^\alpha, \dots, Y^{\alpha^q}, Z) = 0] \Big| \geq \epsilon$$

where the probability is over the random choice of generators T, X in $\mathbb{G}_1$, Y in $\mathbb{G}_2$, the random choice of α in $\mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$.

Assumption 1 *No t-time algorithm has advantage greater than ϵ in solving the above problem.*

Theorem 10 *Let $q = q_{ID} + 1$. Assume Assumption 1 holds for $(\mathbb{G}_1, \mathbb{G}_2)$. Then, the above IBE scheme is (t', q_{ID}, ϵ')-IND-ID-CPA secure for $t' = t - \mathcal{O}(t_{exp} \cdot q^2)$ and $\epsilon' = \epsilon + 1/p$, where t_{exp} is the time required to exponentiate in $\mathcal{G}$.*

Proof. The proof closely follows that of Theorem 1 in [18], with appropriate modifications for the asymmetric setting. Let $\mathcal{A}$ be an adversary that (t', q_{ID}, ϵ')-breaks the IND-ID-CPA security of the IBE scheme described above. We construct and algorithm, $\mathcal{B}$, that solves the above problem, as follows. $\mathcal{B}$ takes as input a random challenge of the form $(T, T^{\alpha^{q+2}}, X, X^{\alpha}, Y, Y^{\alpha}, \ldots, Y^{\alpha^q}, Z)$, where Z is either $e(T, Y^{\alpha^{q+1}})$ or a random element of $\mathcal{G}_T$. Algorithm $\mathcal{B}$ proceeds as follows.

Setup: $\mathcal{B}$ generates a random polynomial $f(x) \in \mathbb{Z}_p$ of degree q. It sets $H = Y^{f(\alpha)}$, computing H from $(Y, Y^{\alpha}, \ldots, Y^{\alpha^q})$. It sends the public key (X, A, H, Y) to $\mathcal{A}$. Since Y, α, and $f(x)$ are chosen uniformly at random, H is uniformly random and this public key has a distribution identical to that in the actual construction.

Phase 1: $\mathcal{A}$ makes key generation queries. $\mathcal{B}$ responds to a query on $ID \in \mathbb{Z}_p$ as follows. If $ID = \alpha$, $\mathcal{B}$ uses α to solve the problem immediately. Else, let $F_{ID}(x)$ denote the $(q-1)$-degree polynomial $(f(x) - f(ID))/(x - ID)$. $\mathcal{B}$ sets the private key (r_{ID}, H_{ID}) to be $(f(ID), Y^{F_{ID}(\alpha)})$. This is a valid private key for ID, since $Y^{F_{ID}(\alpha)} = Y^{(f(\alpha)-f(ID))/(\alpha-ID)} = \left(HY^{-f(ID)}\right)^{1/(\alpha-ID)}$, as required.

Challenge: $\mathcal{A}$ outputs identity ID_* and messages M_0, M_1. Again, if $\alpha \in \{M_0, M_1\}$, $\mathcal{B}$ uses α to solve the problem immediately. Else, $\mathcal{B}$ generates a bit $b \in \{0,1\}$, and computes a private key (r_{ID_*}, H_{ID_*}) for ID_* as in Phase 1. Let $f_2(x) = x^{q+2}$ and let $F_{2,ID_*}(x) = (f_2(x) - f_2(ID_*))/(x - ID_*)$, which is a polynomial of degree $q+1$. $\mathcal{B}$ sets

$$U = T^{(f_2(\alpha)-f_2(ID_*))}, \qquad V = Z \cdot e\left(T, \prod_{i=0}^{q} Y^{F_{2,ID_*,i}\alpha^i}\right) \qquad W = M_b / e\left(U, H_{ID_*}\right) \cdot V^{r_{ID_*}},$$

where $F_{2,ID_*,i}$ is the coefficient of x^i in $F_{2,ID_*}(x)$. It sends (U, V, W) to $\mathcal{A}$ as the challenge ciphertext. Let $s = (\log_X T)F_{2,ID_*}(\alpha)$. If $Z = e(T, Y^{\alpha^{q+1}})$, then

$$\begin{aligned}
U &= T^{(f_2(\alpha)-f_2(ID_*))} \\
&= T^{((\alpha-ID_*)(f_2(\alpha)-f_2(ID_*))/(\alpha-ID_*)} \\
&= T^{(\alpha-ID_*)F_{2,ID_*}(\alpha)} \\
&= X^{(\log_X T)F_{2,ID_*}(\alpha)(\alpha-ID_*)} \\
&= X^{s(\alpha-ID_*)},
\end{aligned}$$

$$\begin{aligned}
V &= e(T, Y^{\alpha^{q+1}}) \cdot e(T, Y^{F_{2,ID_*}(\alpha)-\alpha^{q+1}}) \\
&= e(T, Y^{\alpha^{q+1}+F_{2,ID_*}(\alpha)-\alpha^{q+1}}) \\
&= e(T, Y^{F_{2,ID_*}(\alpha)}) \\
&= e(X^{\log_X T}, Y^{F_{2,ID_*}(\alpha)}) \\
&= e(X, Y)^{(\log_X T)F_{2,ID_*}(\alpha)} \\
&= e(X, Y)^s,
\end{aligned}$$

and

$$\begin{aligned}
M_b/W &= e(U, H_{ID_*})V^{r_{ID_*}} \\
&= e(X^{s(\alpha-ID_*)}, Y^{F_{ID_*}(\alpha)})e(X, Y)^{sf(ID_*)} \\
&= e(X^{s(\alpha-ID_*)}, Y^{(f_2(\alpha)-f_2(ID_*))/(\alpha-ID_*)})e(X, Y)^{sf(ID_*)} \\
&= e(X, Y)^{s(f(\alpha)-f(ID_*)+f(ID_*))} \\
&= e(X, Y^{f(\alpha)})^s \\
&= e(X, H)^s;
\end{aligned}$$

thus (U,V,W) is a valid ciphertext for (ID_*, M_b) under randomness s. Since $\log_X T$ is uniformly random, s is uniformly random, and so (U,V,W) is a valid, appropriately distributed challenge to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ makes key generation queries, and $\mathcal{B}$ responds as in Phase 1.

Guess: Finally, the adversary outputs a guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{B}$ outputs 0 (indicating that $Z = e(T, Y^{\alpha^{q+1}})$); otherwise, it outputs 1.

Perfect simulation: When $Z = e(T, Y^{\alpha^{q+1}})$, the public key and challenge ciphertext issued by $\mathcal{B}$ are identically distributed to those of the actual construction. It still remains to show that the private keys are appropriately distributed. Let $\mathcal{I}$ be a set consisting of α, ID_*, and the identities for which $\mathcal{A}$ requested private keys; observe that $|\mathcal{I}| \leq q+1$. Since $f(x)$ is a uniformly random polynomial of degree q, the values $\{f(a) : a \in \mathcal{I}\}$ are uniformly independent from $\mathcal{A}$'s view, so the private keys issued by $\mathcal{B}$ are appropriately distributed.

Probability analysis: If $Z = e(T, Y^{\alpha^{q+1}})$, then the simulation is perfect, and $\mathcal{A}$ will guess the bit b correctly with probability $1/2 + \epsilon'$. Else, Z is uniformly random, and thus (U,V) is a uniformly random an independent element of $(\mathbb{G}_1, \mathbb{G}_T)$. In this case the inequality $V \neq e(U,Y)^{1/(\alpha - ID_*)}$ holds with probability $1 - 1/p$. When this inequality holds, the value of $e(U, H_{ID_*})V^{r_{ID_*}} = e(U, (HY^{-r_{ID_*}})^{1/(\alpha - ID_*)})V^{r_{ID_*}} = e(U,H)^{1/(\alpha - ID_*)}(V/e(U,Y)^{1/(\alpha - ID_*)})^{r_{ID_*}}$ is uniformly random and independent from $\mathcal{A}$'s view (except for the value W), since r_{ID_*} is uniformly random and independent from $\mathcal{A}$'s view (except for the value W). Thus, W is uniformly random and independent, and (U,V,W) gives no information regarding the bit b.

Assuming no queried identity equals α (which would only increase $\mathcal{A}$'s probability of success), we see that $\left|\Pr[(T, T^{\alpha^{q+2}}, X, X^{\alpha}, Y, Y^{\alpha}, \ldots, Y^{\alpha^q}, Z) = 0] - 1/2\right| \leq 1/p$ when Z is a random element of $\mathbb{G}_T$. However, we have that $\left|\Pr[(T, T^{\alpha^{q+2}}, X, X^{\alpha}, Y, Y^{\alpha}, \ldots, Y^{\alpha^q}, Z) = 0] - 1/2\right| \geq \epsilon'$ when $Z = e(T, Y^{\alpha^{q+1}})$. Thus, for uniformly random T, X, Y, α and Z, we have that

$$\Big|\Pr[\mathcal{B}(T, T^{\alpha^{q+2}}, X, X^{\alpha}, Y, Y^{\alpha}, \ldots, Y^{\alpha^q}, e(T, Y^{\alpha^{q+1}})) = 0] \\ - \Pr[\mathcal{B}(T, T^{\alpha^{q+2}}, X, X^{\alpha}, Y, Y^{\alpha}, \ldots, Y^{\alpha^q}, Z) = 0]\Big| \geq \epsilon' - 1/p.$$

Time-complexity: $\mathcal{B}$'s overhead is dominated by computing $Y^{F_{ID}(\alpha)}$ in response to $\mathcal{A}$'s key generation query on ID, where $F_{ID}(x)$ is a polynomial of degree $q-1$. Each such computation requires $\mathcal{O}(q)$ exponentiations in $\mathbb{G}_2$. Since $\mathcal{A}$ makes at most $q-1$ such queries, $t = t' + \mathcal{O}(q^2)$.

This concludes the proof of Theorem 10. □