Updated version of a paper published in the proceedings of the 12th ACM Conference on Computer and Communications Security—CCS 2005, Alexandria, VA, November 2005. Current version available from the IACR Cryptology ePrint Archive as Report 2005/288.

Direct Chosen Ciphertext Security from Identity-Based Techniques

Xavier Boyen * Qixiang Mei † Brent Waters ‡

June 7, 2006

Abstract

We describe a new encryption technique that is secure in the standard model against adaptive chosen ciphertext (CCA2) attacks. We base our method on two very efficient Identity-Based Encryption (IBE) schemes without random oracles due to Boneh and Boyen, and Waters.

Unlike previous CCA2-secure cryptosystems that use IBE as a black box, our approach is endogenous, very simple, and compact. It makes direct use of the underlying IBE structure, and requires no cryptographic primitive other than the IBE scheme itself. This conveys several advantages. We achieve shorter ciphertext size than the best known instantiations of the other methods, and our technique is as efficient as the Boneh and Katz method (and more so than that of Canetti, Halevi, and Katz). Further, our method operates nicely on hierarchical IBE, and since it allows the validity of ciphertexts to be checked publicly, it can be used to construct systems with non-interactive threshold decryption.

In this paper we describe two main constructions: a full encryption system based on the Waters adaptive-ID secure IBE, and a KEM based on the Boneh-Boyen selective-ID secure IBE. Both systems are shown CCA2-secure in the standard model, the latter with a tight reduction. We discuss several uses and extensions of our approach, and draw comparisons with other schemes that are provably secure in the standard model.

1 Introduction

The design of a secure encryption scheme is central to any system that strives to provide secure communication using an untrusted network. In order for a cryptographic scheme to be considered secure in an adversarial setting, it must be secure against chosen ciphertext attacks. While, there have been several efficient encryption schemes shown to be heuristically secure in the random oracle model [3], it wasn't until fairly recently that Cramer and Shoup [13] designed an encryption scheme that was both efficient and provably secure in the standard model (without random oracles) against chosen ciphertext attacks.

Somewhat surprisingly, Canetti Halevi and Katz [11] were able to show how to elegantly construct a CCA2-secure cryptosystem from any identity-based encryption (IBE) scheme secure in the selective-ID model [10]. A user's public encryption key is simply a set of IBE public parameters and the user's secret key is the corresponding IBE master key. To encrypt a message, M, for such a user, one first generates the parameters, VK, to a one-time signature scheme; next one hashes

*Voltage Inc., Palo Alto — *xb@boyen.org*

†Southwest Jiaotong University — *nupfster@gmail.com*

‡Stanford University — *bwaters@theory.stanford.edu*

the signature parameters to obtain an "identity"; then one encrypts the message to the identity calculated from the previous step; one finally signs the partial ciphertext with the one-time signature private key to get a signature σ and attaches σ as part of the ciphertext. To decrypt a message, a user first checks the validity of the signature on a ciphertext, and rejects the ciphertext if the signature is invalid. If the signature is valid, it decrypts the ciphertext from the identity determined by the one-time signature parameters, VK. Intuitively, to attack a ciphertext, C, in the chosen ciphertext model, an adversary will need to generate its own set of signature parameters to obtain a valid signature, σ', before it can construct a valid ciphertext C'. However, this will cause the receiver to attempt decryption with a different identity than the one associated with C, and by the semantic security of the IBE system this will provide no useful information for decrypting C. Boneh and Katz [9] further improve the efficiency of this scheme by using a MAC instead of a one-time signature. One interesting aspect of these schemes is that they seemingly do not fall in the characterization of previous CCA2-secure schemes given by Elkind and Sahai [16].

Both the CHK and BK techniques are generic. The efficient constructions of both methods come from instianting them with either one of the Boneh-Boyen [4] identity-based encryption schemes. A natural question is whether we can construct improved CCA2-secure encryption schemes by taking advantage of specific properties of the most efficient IBE schemes secure without random oracles [29, 4]. We answer this question in the affirmative.

In this paper we first show how to build a direct CCA2-secure public key cryptosystem from the Waters identity-based encryption cryptosystem [29]. We construct a CCA2-secure cryptosystem in which ciphertexts consist of just three group elements with no attached signatures or MACs. The basic idea behind our scheme is as follows. As in the CHK method, the public key of a user will correspond to the public parameters of an IBE scheme. To encrypt a message, M, the encryption algorithm first creates the first two elements of the ciphertext, which in the Waters scheme are *independent* of the identity; next these two elements are hashed to determine a "one-time identity"; finally the ciphertext is completed by constructing the third element to form an encryption to the identity determined from the previous step. That is, the identity that we encrypt to is actually determined by the first two elements of the ciphertext itself. In this manner a "well formed" ciphertext is self-contained in that we do not need any auxiliary signatures or MACs.

We get our leverage from two properties of the Waters scheme [29]. The first is that since we work in groups with efficiently computable bilinear maps, we can use the bilinear map to check that the third element is formed correctly, and thus that the ciphertext is well formed (this is only necessary for the simulation, as the decryption algorithm can do the check more efficiently). Secondly, we take advantage of the semantic security of the IBE system in the full adaptive-identity security model (as opposed to the weaker selective-ID model). When proving security of our scheme, the simulator will not know until the challenge phase which "identity" the challenge ciphertext will be for, since the challenge identity depends partially on the adversary's input. Since the identity is not determined until well after setup, we need to base our scheme on an adaptive-ID secure IBE scheme.

Perhaps more surprisingly, we also show that our technique can be used to build a Key Encapsulation Mechanism (KEM) with full CCA2 security based only on the scheme of Boneh and Boyen,[1] which is only selective-ID secure (in its basic configuration). Since in a KEM there is no message to encrypt, in a chosen ciphertext attack the challenge ciphertext can be at once properly distributed and independent of any adversarial input. Therefore, the challenge ciphertext and the associated

[1]Although two distinct efficient IBE constructions are given in [4]; in this paper "the Boneh-Boyen scheme" refers by default to their first scheme, i.e., the (H)IBE scheme based on the Bilinear Diffie-Hellman assumption [4, §4].

identity can be chosen before setup when running a security simulation, as in the selective-ID model. Besides simplicity, the main benefit of this construction is that we get a tight security reduction from an already very reasonable underlying complexity assumption. The fact that we get a KEM (as opposed to a complete cryptosystem) is practically irrelevant since public-key encryption is almost exclusively used to encrypt random session keys in practical applications.

The two CCA2-secure systems we describe have advantages over both the CHK [11] and BK [9] generic constructions.

First, our ciphertexts are short, consisting of just three group elements (or two for the KEM), with no attached signature or MAC. For comparison, a ciphertext in the CHK scheme will need to have attached a one time signature and public key. Typically, fast one time signatures schemes [22] will have long signatures lengths and thus blow up the ciphertext size. Alternatively, as pointed out by Boneh et al. [7], we could base one-signature schemes off "full-blown" signature schemes that use number-theoretic constructions. However, such signatures take longer to both create and verify, making the CHK method less efficient in both the encryption and decryption stage than ours.

By contrast, the construction of BK avoids to a large extent the previous drawbacks by replacing the signature with a MAC, and is much faster since the time to compute a MAC is insignificant compared to the IBE operations; and indeed, the performance of the BK scheme is roughly the same as ours (though BK still requires three to five times as many random bits, most of which are used in the MAC construction).

The main drawback of using a MAC in the BK system is that its verification requires knowledge of the private key, whereas in our construction the ciphertext validity test may be done with the public key. This distinction is crucial for the construction of threshold systems (where the private key is shared amongst decryption servers, each of which can only perform a partial decryption of a given ciphertext). Public key-only ciphertext verification allows the threshold decryption servers to operate without interaction, which greatly simplifies the system. Boneh, Boyen, and Halevi [6] recently described a generic and efficient non-interactive CCA2 threshold system without random oracles, based on the CHK transformation. Using our technique we are able to construct an even more efficient (albeit non-generic) fully non-interactive threshold KEM with CCA2 security in the standard model, by specializing the method of [6].

Finally, if we apply our technique to the last level of the depth-$(\ell+1)$ hierarchical version of the Waters or Boneh-Boyen IBE scheme, we immediately obtain a depth-ℓ HIBE with intrinsic CCA2 security.

In summary, our schemes enjoy the efficiency of the BK scheme, can be used in threshold CCA2-secure systems like CHK, and have shorter ciphertexts than both.

1.1 Related Work

We restrict our comparisons to encryption systems that are CCA2-secure [25] in the standard model. There are several efficient schemes that can be shown to be secure in the random oracle [3] model, however, we can only make heuristic arguments for the security of these schemes.

Naor and Yung [24] described a scheme provably secure against lunch-time attacks. These techniques were later extended by Dolev, Dwork, and Naor [15] and Sahai [26] to protect against an adaptive adversary in a chosen ciphertext attack. None of the above methods, however, yields a scheme to be efficient enough to be of practical use.

Cramer and Shoup [13] developed the first practical CCA2-secure scheme that was provably secure in the standard model. Later, Cramer and Shoup [14] generalized their techniques by constructing CCA2-secure schemes from “projective hash functions”. Shoup [27] showed how to make an efficient hybrid scheme by using the original Cramer-Shoup system as a KEM. Kurosawa and Desmedt [21] further demonstrated an even more efficient CCA2-secure hybrid system by using a KEM that was not necessarily CCA2-secure; Abe et al. [1] recently generalized their construction.

Canetti, Halevi, and Katz [11] describe a new paradigm for constructing CCA2-secure schemes from selective-ID secure identity-based encryption systems. Boneh and Katz [9] later improved upon the efficiency of this result. Both of these methods are generic in that they can be applied to any selective-ID secure cryptosystem, whereas our method is particular to the Waters [29] adaptive-ID secure identity-based encryption scheme. For concreteness when comparing the performance of the schemes we consider their construction applied to the Boneh and Boyen [4] IBE scheme.

1.2 Organization

In Section 2 we give a few preliminaries necessary for our constructions. We describe our fully secure encryption system in Section 3, and reason about its security. In Section 4 we describe an alternative key encapsulation scheme with tight security. In Section 5 we mention a few extensions of practical interest to both constructions. Then, in Section 6, we focus on the qualitative properties of our schemes, and draw detailed comparisons with related work in the literature. Finally, we state our conclusions in Section 7.

2 Preliminaries

We briefly review the notions of chosen ciphertext security for encryption and key encapsulation. We also define bilinear groups and pairings, and state our complexity assumption.

2.1 Secure Encryption

A public key encryption system consists of three (randomized) algorithms that are modeled as follows.

KeyGen(λ)**:** Takes as input a security parameter $\lambda \in \mathbb{Z}^+$. It outputs a public/private key pair.

Encrypt(PK, M)**:** Takes as input a public key PK and a message M. It outputs a ciphertext.

Decrypt(SK, C)**:** Takes as input a private key SK and a ciphertext C. It outputs a plaintext message or the special symbol $\perp$.

The strongest and commonly accepted notion of security for a public key encryption system is that of indistinguishability against an adaptive chosen ciphertext attack. This notion, denoted IND-CCA2, is defined using the following game between a challenger and an adversary $\mathcal{A}$. Both are given the security parameter $\lambda \in \mathbb{Z}^+$ as input.

Setup. The challenger runs $KeyGen(\lambda)$ to obtain a random instance of public and private key pair $(\mathsf{PK}, \mathsf{SK})$. It gives the public key PK to the adversary.

Query phase 1. The adversary adaptively issues decryption queries C where $C \in \{0,1\}^*$. The challenger responds with $\mathit{Decrypt}(\mathsf{SK}, C)$.

Challenge. The adversary outputs two (equal length) messages M_0, M_1. The challenger picks a random $b \in \{0,1\}$ and sets $C^* = \mathit{Encrypt}(\mathsf{PK}, M_b)$. It gives C^* to the adversary.

Query phase 2. The adversary continues to issue decryption queries C as in phase 1, with the added constraint that $C \neq C^*$. The challenger responds with $\mathit{Decrypt}(\mathsf{SK}, C)$.

Guess. Algorithm $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

The above is commonly known as the IND-CCA2 game. We define the advantage of $\mathcal{A}$ in this game as $\mathsf{AdvCCA}_{\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$. An encryption system is (t, q, ϵ)-IND-CCA2 secure if there is no randomized algorithm $\mathcal{A}$ that runs in time t, makes at most q decryption queries, and has advantage at least ϵ in the IND-CCA2 game.

2.2 Key Encapsulation

A Key Encapsulation Mechanism (KEM) is a cryptographic primitive whose purpose is to securely convey a random session key to the recipient. Unlike interactive key exchange protocols such as Diffie-Hellman, the session key is entirely determined by the random bits used by the sender. Unlike with ordinary encryption as above, the session key is not a message that can be chosen by the sender. Formally a KEM is modeled by three algorithms:

KeyGen(λ)**:** Takes as input a security parameter $\lambda \in \mathbb{Z}^+$. It outputs a public/private key pair.

Encapsulate(PK)**:** Takes as input a public key PK. It outputs a ciphertext and a session key.

Decapsulate(SK, C)**:** Takes as input a private key SK and a ciphertext C. It outputs a session key or the special symbol $\perp$.

The notion of adaptive chosen ciphertext security for key encapsulation is similar to that for encryption, except that there are no challenge messages to encrypt. Instead, in the challenge phase the challenger flips a coin $b \in \{0,1\}$, and the adversary is given a ciphertext C^* and a string K^*, which will be the session key encapsulated by the ciphertext if $b = 1$, or a random string if $b = 0$. The adversary makes adaptive decapsulation queries (except on C^*, once revealed), and eventually outputs a guess b' for b.

We refer to this interaction as the KEM-CCA2 game, and define the advantage of $\mathcal{A}$ as $\mathsf{AdvCCA}_{\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$. A key encapsulation system is (t, q, ϵ)-KEM-CCA2 secure if there is no randomized algorithm $\mathcal{A}$ that runs in time t, makes at most q decapsulation queries, and has advantage at least ϵ in the KEM-CCA2 game.

2.3 Asymmetric Bilinear Groups and Maps

Our constructions make use of bilinear pairings. For the sake of generality, we shall describe them in the *asymmetric* bilinear group framework, which provides for two, possibly distinct, isomorphic groups $\mathbb{G}$ and $\hat{\mathbb{G}}$, between which a bilinear map is defined. Pairing-based encryption systems have traditionally been described in the simpler *symmetric* bilinear setting where these groups are equal, although there are significant practical benefits to consider the more general case (e.g., a broader choice of elliptic curve implementations, more compact ciphertexts, etc.). The security of our systems relies on the familiar Bilinear Diffie-Hellman assumption, which we restate in the asymmetric setting.

Let $\mathbb{G}$ and $\hat{\mathbb{G}}$ be a pair of (possibly distinct) cyclic groups of large prime order p, related by some homomorphism $\phi : \hat{\mathbb{G}} \to \mathbb{G}$. Let $g \in \mathbb{G}^*$ and $h \in \hat{\mathbb{G}}^*$ be generators of $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively, such that $\phi(h) = g$. Let $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ be a function that maps pairs of elements in $(\mathbb{G}, \hat{\mathbb{G}})$ to elements of some group $\mathbb{G}_T$, where $\mathbb{G}_T$ has order p (and is distinct from $\mathbb{G}$ and $\hat{\mathbb{G}}$). Further suppose that:

– the maps e, ϕ, and the respective group operations in $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_T$ (written multiplicatively), are all efficiently computable;
– the map e is non-degenerate, in the sense that $e(g, h) \neq 1$;
– the map e is bilinear, i.e., $\forall u \in \mathbb{G}, \forall v \in \hat{\mathbb{G}}, \forall a, b \in \mathbb{Z}, e(u^a, v^b) = e(u, v)^{ab}$.

Then we say that $(\mathbb{G}, \hat{\mathbb{G}})$ is a bilinear group pair, and that e is a bilinear map or pairing in $(\mathbb{G}, \hat{\mathbb{G}})$.

We emphasize that, in our formulation, the homomorphism ϕ is only used in the abstract definitions, and not in the actual constructions or even the security reductions. The 'asymmetry' refers to the non-interchangeability of the arguments of the bilinear map e.

2.4 Bilinear Diffie-Hellman Assumption

The Bilinear Diffie-Hellman (BDH) problem was first proposed in the symmetric setting in [20, 8], and later generalized to the asymmetric setting in the full version of [4]. The generalization proposed by Boneh and Boyen differs from other proposals in that it does not require the homomorphism ϕ to be efficiently computable, which gives a weaker assumption.

Thus, following Boneh and Boyen, we consider the BDH problem stated for fixed $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e$, as follows:

Given $(g, g^a, g^c, h, h^a, h^b) \in \mathbb{G}^3 \times \hat{\mathbb{G}}^3$ for random $a, b, c \in \mathbb{Z}_p$, $h \in \hat{\mathbb{G}}^*$, output $e(g, h)^{abc} \in \mathbb{G}_T$.

Notice that ϕ and the elements g^b and h^c are omitted from the statement.

Accordingly, we say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the (computational) BDH problem in $(\mathbb{G}, \hat{\mathbb{G}})$ if

$$\Pr\left[\mathcal{A}(g, g^a, g^c, h, h^a, h^b) = e(g, h)^{abc}\right] \geq \epsilon$$

where the probability is over the random choice of generators $g \in \mathbb{G}$ and $h \in \hat{\mathbb{G}}^*$, the random choice of exponents $a, b, c \in \mathbb{Z}_p$, and the random bits used by $\mathcal{A}$.

Similarly, we say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the *decisional* BDH problem in $(\mathbb{G}, \hat{\mathbb{G}})$ if

$$\left|\Pr\left[\mathcal{B}(g, g^a, g^c, h, h^a, h^b, e(g, g)^{abc}) = 0\right] - \Pr\left[\mathcal{B}(g, g^a, g^c, h, h^a, h^b, T) = 0\right]\right| \geq \epsilon$$

where the probability is over the random choice of generators $g \in \mathbb{G}$ and $h \in \hat{\mathbb{G}}^*$, the random choice of exponents $a, b, c \in \mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_T$, and the random bits used by $\mathcal{B}$.

We say that the (t, ϵ)-BDH or Decision (t, ϵ)-BDH assumption holds in $(\mathbb{G}, \hat{\mathbb{G}})$ if no t-time algorithm has advantage at least ϵ in solving the BDH or Decision BDH problem in $(\mathbb{G}, \hat{\mathbb{G}})$, respectively.

Observe that we avoid specifying ϕ in the BDH problem instance by providing selected powers of both g and h to the adversary. Indeed, providing g and g^a would be unnecessary if ϕ had been given. We note that, even in the general case, providing both g^a and h^a may seem redundant, but it is necessary to preserve the formal equivalence between $\mathbb{G}$ and $\hat{\mathbb{G}}$; it is also harmless since given a problem instance it is easy to tell whether (g, g^a, h, h^a) is a legitimate Diffie-Hellman tuple using the bilinear map.

3 Secure Encryption from Adaptive-ID IBE

We now present our scheme which is a direct construction based off the Waters [29] identity-based encryption scheme. We first describe our construction and then present the intuition behind its security. The full proof will be given in the appendix.

3.1 Encryption System

Let $\mathbb{G}$ and $\hat{\mathbb{G}}$ be two cyclic groups of prime order, p, between which there exists an efficiently computable bilinear map into $\mathbb{G}_T$. Specifically, let $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ denote the bilinear map, and let $g \in \mathbb{G}$ and $h \in \hat{\mathbb{G}}$ be the corresponding generators. The size p of the groups is determined by the security parameter. We also assume the availability of some collision resistant function family (but not necessarily one-way). Without any further assumptions, we may use any fixed injective encoding $H_0 : \mathbb{G}_T \times \mathbb{G} \to \{0,1\}^n$, which by a counting argument demands that $n \geq \lceil 2\log_2(p) \rceil$. However, since the public key size and encryption time will be seen to grow linearly with n, it may be more economical as an alternative to H_0 to substitute a family of collision resistant functions $H_s : \mathbb{G}_T \times \mathbb{G} \to \{0,1\}^n$, indexed from some finite set $\{s\}$, in which case adequate collision resistance may be provided with an output size of only $n \approx \log_2(p)$.

The following description is written so as to provide the most compact ciphertexts under the assumption that $\mathbb{G}$'s elements have a shorter representation than $\hat{\mathbb{G}}$'s. An example of this is when the bilinear map is realized as the Weil or Tate pairing on certain algebraic curves, where $\mathbb{G}$ and $\hat{\mathbb{G}}$ are subgroups of points in the ground field and in an extension field, respectively. If the converse is true—namely, if the elements of $\hat{\mathbb{G}}$ have the shorter representation—it suffices to exchange all occurrences of g and h and then swap the arguments of all pairings $e(\cdot,\cdot)$ to restore the short ciphertext property. The same trick can be used if it is desirable to minimize the private key size rather than the ciphertext. Although this is not true in general, this trick is applicable throughout this paper because $\mathbb{G}$ and $\hat{\mathbb{G}}$ play equivalent roles in all our constructions.

The cryptosystem is described by the following three algorithms.

Key Generation: A user's public/private key pair generation algorithm proceeds as follows. First, a secret $\alpha \in \mathbb{Z}_p$ is chosen at random, from which the values $h_0 = h^\alpha$ and $Z = e(g, h_0)$ are calculated. Next, the algorithm chooses a random $y' \in \mathbb{Z}_p$ and a random n-length vector $\vec{y} = (y_1, \ldots, y_n)$, whose elements are chosen at random from $\mathbb{Z}_p$. It then calculates $u' = g^{y'}$ and $u_i = g^{y_i}$ for $i = 1$ to n. Finally, a random seed s for the collision resistant family is chosen, if needed (for notational convenience, we always write H_s, and peg $s = 0$ whenever the injective encoding H_0 is used).

The published public key is

$$\left(\ s,\ \ Z = e(g,h)^\alpha,\ \ u' = g^{y'},\ \ u_1 = g^{y_1},\ \ \ldots,\ \ u_n = g^{y_n}\ \right) \quad \in\ \{s\} \times \mathbb{G}_T \times \mathbb{G}^{n+1},$$

and the private key is

$$\left(\ h_0 = h^\alpha,\ \ y',\ \ y_1,\ \ \ldots,\ \ y_n\ \right) \quad \in\ \hat{\mathbb{G}} \times \mathbb{Z}_p^{n+1}.$$

Encryption: A message $M \in \mathbb{G}_T$ is encrypted as follows. First, a value $t \in \mathbb{Z}_p$ is randomly chosen. Next, the first two elements of the ciphertext are computed: $C_0 = M \cdot Z^t = M \cdot e(g,h)^{\alpha t}$ and $C_1 = g^t$. Next, a bit string $w \in \{0,1\}^n$ is derived as $w = H_s(C_0, C_1)$. Let $w_1 w_2 \ldots w_n$ denote the binary expansion of w, where each bit $w_i \in \{0,1\}$. The final step is to compute $C_2 = (u' \prod_{i=1}^n u_i^{w_i})^t$. The complete ciphertext, $C = (C_0, C_1, C_2)$, consists of the three group elements

$$\left(\ M \cdot Z^t,\ \ g^t,\ \ (u' \prod_{i=1}^{n} u_i^{w_i})^t\ \right) \quad \in\ \mathbb{G}_T \times \mathbb{G}^2.$$

Decryption: Let $C = (C_0, C_1, C_2)$ be a ciphertext and $w = H_s(C_0, C_1)$. In a well-formed ciphertext, the quadruple $(g, C_1, u' \prod_{i=1}^n u_i^{w_i}, C_2) \in \mathbb{G}^4$ will be a Diffie-Hellman tuple, which can be efficiently tested by the private key holder as follows.

Given a ciphertext C the algorithm first computes $w = H_s(C_0, C_1)$, expressed in binary as $w_1 w_2 \ldots w_n$. Next, it raises C_1 to the power of $w' = y' + \sum_{i=1}^n y_i w_i \bmod p$, and compares the result $(C_1)^{w'}$ with C_2. If these two values are unequal, then $(g, C_1, u' \prod_{i=1}^n u_i^{w_i}, C_2)$ is not a Diffie-Hellman tuple, and the algorithm outputs $\perp$.

Otherwise, the ciphertext is valid, and the algorithm decrypts the message as

$$C_0 \Big/ e(C_1, h_0) = M \quad \in\ \mathbb{G}_T.$$

3.2 Analogy to the Waters IBE

The above system bears a strong resemblence to Waters' adaptive-ID semantically secure IBE [29]: the public and private keys are essentially identical to the master public and secret parameters in the IBE system, and the bit string w plays the role of the recipient identity.

Other than notational differences, the distinguishing feature is that the identity is not chosen by the encryption party but determined by the first two ciphertext elements; it is this feature that conveys our scheme its chosen ciphertext security. Additionally, all the secret exponents u', $u_1, \ldots, u_n$ are retained in the private key, which allows for faster validity checking and decryption than in the IBE system. (In practice, these exponents could be generated from a pseudo random number generator seeded by h_0 in order to reduce the cost of secure storage for the private key.)

Again, we note that while our scheme is derived from the Waters IBE scheme, an IBE private key is never generated, since it is more efficient to decrypt directly using the "master key".

3.3 Security

We now give the intuition for the security of our system.

As noted before a ciphertext in our scheme is essentially an IBE ciphertext where the identity is determined from the first two elements. It is possible to generate ciphertexts this way since in the Waters scheme only the third ciphertext element depends on the identity.

Our simulation roughly works as follows. For all decryption requests of a ciphertext C the simulator first checks that the ciphertext is well formed. This amounts to checking the DDH property, which the simulator can do without the private exponents by using the bilinear map. If the ciphertext is well formed, the simulator creates a private key for the identity string determined from the first two elements of the ciphertext, and uses this to decrypt the ciphertext. The simulator will then create a challenge ciphertext $C^* = (C_0^*, C_1^*, C_2^*)$ which will be equivalent to an identity-based encryption under the identity $w^* = H_s(C_0^*, C_1^*)$. Since H_s is collision resistant (or injective), the adversary will not be able to make any well-formed ciphertext queries that would require the simulator to use an IBE key for the same identity string w^*. Thus, the security of our scheme follows by virtue of the underlying IBE security. We emphasize that even though C_2 acts only as a checksum in the regular decryption algorithm, it plays an active role in the decryption process conducted by the simulator.

We remark that the above argument is not a generic reduction from the underlying IBE. The problem is that in the challenge phase of the IBE game, the adversary is allowed to choose the identity it wants to attack, whereas here it is the challenge ciphertext itself that determines the target identity w^*. Additionally, we note that since C_0^* depends partially on input from the adversary; the value of $w^* = H_s(C_0^*, \ldots)$ cannot be determined at setup time by the simulator as in previous IBE-to-CCA2 transformations such as CHK [11] and BK [9]. This is the reason why our system is based on adaptive-identity secure IBE (although we will see in Section 4 that selective-identity secure IBE is enough if we forgo direct encryption and meander through key encapsulation).

Formally, we have the following result, stated for the fixed injective encoding implementation for simplicity. Since the formal argument for CCA2 security very much resembles that of Waters [29] for adaptive-ID security, we defer the proof to the full paper.[2]

Theorem 3.1. *Suppose the (t', ϵ')-Decision BDH assumption holds in $(\mathbb{G}, \hat{\mathbb{G}})$, and assume that $H_0 : \mathbb{G}_T \times \mathbb{G} \to \{0,1\}^n$ is an efficiently computable injection for some n. Then the encryption system of Section 3.1 is (t, q, ϵ)-chosen ciphertext (IND-CCA2) secure for any $q < p$ provided that $\epsilon \geq 32(n+1)q\epsilon'$ and $t \leq t' - \Theta(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1}))$, where $\lambda = \frac{1}{8(n+1)q}$, and where it is assumed that each exponentiation, pairing, and evaluation of H_0 takes constant time.*

3.4 Efficiency

Encryption in our scheme requires requires one exponentiation in $\mathbb{G}_T$, two exponentiations in $\mathbb{G}$, and an average of $n/2$ (at most n) group operations in $\mathbb{G}$ which amount to much less than an exponentiation. If the encryption party is to send multiple messages under the same public key, then all but one of the above exponentiations can be greatly accelerated by using many well-known pre-computation techniques for fixed-base exponentiation. (In the same vein, the product $\prod u_i^{w_i}$ can also be pre-computed and cached factor-by-factor or using fixed-sized windows.)

Decryption requires one exponentiation in $\mathbb{G}$ and one bilinear pairing into $\mathbb{G}_T$.

4 Tight Key Encapsulation from Selective-ID IBE

It is easy to turn our CCA2-secure encryption system into a CCA2-secure Key Encapsulation Mechanism (KEM). However, we can get a simpler construction with a *tight* security reduction if we take the lesser requirements of the KEM to our advantage and start from the Boneh-Boyen IBE [4].

[2] Instead, we give a formal proof for the simpler and equally informative KEM construction of the next section.

4.1 KEM Construction

As before, we let $\mathbb{G}$ and $\hat{\mathbb{G}}$ be cyclic groups of prime order, p, generated by g and h, and equipped with a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. We also assume the availability of either a collision resistant family $H_s : \mathbb{G} \to \mathbb{Z}_p$ or a fixed injective encoding $H_0 : \mathbb{G} \to \mathbb{Z}_p$. Notice that unlike in Section 3 this time the domain of H_s or H_0 is just $\mathbb{G}$, and the range is now $\mathbb{Z}_p$. As before we assume that the elements of $\mathbb{G}$ have at least as small a representation as those of $\hat{\mathbb{G}}$. The key encapsulation scheme is described by the following three algorithms.

Key Generation: A user's public/private key pair generation algorithm proceeds as follows. First, a secret $\alpha \in \mathbb{Z}_p$ is chosen at random, from which the values $h_0 = h^\alpha$ and $Z = e(g, h_0)$ are calculated. Next, the algorithm chooses y_1 and y_2 at random from $\mathbb{Z}_p$. It then calculates $u_1 = g^{y_1}$ and $u_2 = g^{y_2}$. Finally, a random seed s for the collision resistant family is chosen, if needed (for convenience, we assume that $s = 0$ whenever the fixed encoding H_0 is used).

The published public key is

$$\left(\; s, \quad Z = e(g,h)^\alpha, \quad u_1 = g^{y_1}, \quad u_2 = g^{y_2} \; \right) \quad \in \; \{s\} \times \mathbb{G}_T \times \mathbb{G}^2,$$

and the private key is

$$\left(\; h_0 = h^\alpha, \quad y_1, \quad y_2 \; \right) \quad \in \; \hat{\mathbb{G}} \times \mathbb{Z}_p^2.$$

Encapsulation: The generation and encapsulation of a random session key works as follows. First, a value $t \in \mathbb{Z}_p$ is randomly chosen, and the algorithm computes the first element of the ciphertext: $C_1 = g^t$. Next, it computes $w \in \mathbb{Z}_p$ as $w = H_s(C_1)$, and then the second ciphertext element: $C_2 = u_1^t u_2^{tw} = (u_1 u_2^w)^t$. The complete ciphertext, or encapsulated key, C, consists of the two group elements

$$\left(\; g^t, \quad u_1^t u_2^{tw} \; \right) \quad \in \; \mathbb{G}^2.$$

The session key, K, is calculated by the sender as the group element $K = Z^t = e(g,h)^{\alpha t} \in \mathbb{G}_T$.

Decapsulation: Let $C = (C_1, C_2)$ be a ciphertext encapsulating some session key K. Before recovering K, the algorithm must verify that the ciphertext is legitimate. To do so, the algorithm computes $w = H_s(C_1)$ and $w' = y_1 + y_2 \cdot w \pmod p$. It then computes $C_1^{w'}$ and compares it with C_2. If these two values are unequal, then $(g, C_1, u_1 u_2^w, C_2)$ is not a Diffie-Hellman tuple and the ciphertext is invalid. In this case, the algorithm outputs $\perp$ and halts. Otherwise, the algorithm outputs the session key, K, which it obtains by computing

$$e(C_1, h_0) = K \quad \in \; \mathbb{G}_T.$$

4.2 Analogy to the Boneh-Boyen IBE

The encapsulation algorithm may be viewed as a variant of encryption where the message to be encrypted is the constant $M = 1 \in \mathbb{G}_T$. With $M = 1$, the first ciphertext component would be $C_0 = M \cdot Z^t$, which reduces to the random blinding factor Z^t. For key encapsulation, we suppress this component from the ciphertext output, and instead use it as the randomly generated session key K.

Aside from the fact that the above algorithms provide only key encapsulation rather than true encryption, the system bears the same relation to Boneh and Boyen's selective-ID semantically secure IBE [4] as the system of Section 3.1 did to Waters' IBE. The two differ mainly in the construction of the checksum component C_2, and the related changes to the public and private keys, as in one case the hashed identity is used bit by bit and all at once in the other case.

It is easy to hash elements of $\mathbb{G}$ to $\mathbb{Z}_p$ using a family of collision resistant functions H_s. A difficulty arises in the case where we wish to use a fixed injective encoding H_0. Since $|\mathbb{G}| = |\mathbb{Z}_p| = p$, it follows that H_0 should be an (efficiently computable) bijection, which might not be easy to come by. In reality, we only need H_0 to be injective on most of its domain, as long as we exclude the rest by using an iterated hashing strategy until we hit the adequate portion of the domain. We describe how to do this in a couple of concrete elliptic curve implementations in Appendix C.

4.3 Generalized Identity Hashing

A more general way to enable the use of a fixed injective encoding H_0 is to generalize the identity-dependent function in the underlying Boneh-Boyen IBE. Specifically, we can replace all occurrences of $(u_1 \cdot u_2^w)$ by $(u_1 \cdot u_{2,1}^{w_1} u_{2,2}^{w_2} \cdots u_{2,\nu}^{w_\nu})$, where $w = (w_1, w_2, \ldots, w_\nu)_b$ is the representation of the "identity" w in radix p (or some fixed radix $p' \leq p$), and $u_{2,1}, \ldots, u_{2,\nu}$ are random elements of $\mathbb{G}$ we add to the public key. We then select an injection $H_0 : \mathbb{G} \to (\mathbb{Z}_p)^\nu$, which for large enough ν is trivial to construct. The modifications to the scheme and the security reduction are straightforward.

Note that we did essentially the same thing in Section 3 but with $p' = 2$. Here, since $p' \approx p$, the number ν of group elements in the public key is significantly smaller than its counterpart n in Section 3 (and the underlying Waters IBE scheme); specifically, we save a factor $\frac{n}{\nu} \approx \log_2 p$.

4.4 Security and Efficiency

The security reasoning is similar to that of the encryption system, with one important difference. Since with key encapsulation there is no message to encrypt, the challenge ciphertext given to the adversary in the attack game does not depend on input from the adversary. Since the simulator can choose ahead of time the randomization value t^* to be used in the challenge ciphertext, it can therefore determine (in IBE parlance) the challenge "identity" $w^* = H_s(C_1) = H_s(g^{t^*})$ before interacting with the adversary. This brings us in a similar situation as in the selective-identity IBE proof of security of the Boneh-Boyen system, which is the reason why we are able to construct a KEM using just the equivalent of a selective-ID IBE system.

The security of the KEM follows from that C_2 is functionally dependent on C_1 via w. The first role of C_2 is thus to act as a checksum preventing the adversary from making decryption queries on algebraic transformations of the C_1 component of the challenge ciphertext. As before, C_2 has a second role, which is to help the simulator answer decryption queries (except on the challenge ciphertext), by way of the underlying IBE system.

A crucial point to note is that, in the Boneh-Boyen IBE, C_1 is based on a generator g that in the simulation is passed on unchanged to the adversary and does not depend on the target identity.[3] In the KEM, this translates into a chain of maps $C_1 \mapsto w \mapsto C_2$ that are preserved in the simulation, and allow the challenge ciphertext to be constructed without cyclic dependencies.[4]

[3] This would not be true of an IBE scheme whose proof of security required the simulator to let the generator depend on the challenge identity, since our transformation would then create a circular dependency.

[4] Stated differently, our KEM exploits the fact that the first Boneh-Boyen IBE (based on BDH) is secure against an adversary that may postpone the choice of target ID until the challenger has output "some" of the public parameters. In the usual selective-ID security model, the adversary much make that choice at the onset of the attack.

Recently, Hovav Shacham pointed out to us that the second Boneh-Boyen IBE (the one based on the BDHI assumption) [4, §5] also obeys this property, which means that our transformation into a CCA2-secure KEM applies equally well to that scheme (and results in a surprisingly similar KEM).

Returning to our original construction, we have the following result, stated in the context of a fixed encoding for simplicity. The formal proof of security is given in Appendix B.

Theorem 4.1. *Suppose the (t', ϵ')-Decision BDH assumption holds in $(\mathbb{G}, \hat{\mathbb{G}})$, and let $H_0 : \mathbb{G} \to \mathbb{Z}_p$ be an efficiently computable injection. Then the KEM system of Section 4.1 is (t, q, ϵ)-chosen ciphertext (KEM-CCA2) secure for any $q < p$, any $\epsilon \geq \epsilon' + q/2p$, and any $t \leq t' - \Theta(q)$, where it is assumed that each exponentiation, pairing, and evaluation of H_0 takes unit time.*

Proof. See Appendix B. □

In terms of efficiency, the encapsulation algorithm requires one exponentation in $\mathbb{G}$ and one multi-exponentiation in $\mathbb{G}$. Decapsulation requires one exponentiation in $\mathbb{G}$ and one bilinear map.

In summary, compared to the encryption system, the key encapsulation scheme benefits from these advantages:

- Tight security reduction from the BDH assumption, since the Boneh-Boyen IBE has tight security in selective-identity attacks;
- Shorter public and private keys, requiring only $O(1)$ components, by contrast to the $O(n)$ group elements needed in the encryption system.

4.5 From KEM to Full Encryption

Naturally, once we have a KEM, it is easy to obtain a full encryption system, where the sending party can choose the message it wishes to transmit. This can be done with a hybrid system where the KEM key is used as a session key for a symmetric cipher with a chosen ciphertext secure mode of operation (itself possibly constructed using a MAC). The benefit of this construction is that it retains the tight security reduction of our KEM. The drawback is that the symmetric cipher requires additional randomness, and for short messages will result in longer ciphertexts overall than our direct encryption system of the previous section.

5 Practical Extensions

We now describe a few extensions to the encryption and encapsulation schemes of Sections 3 and 4.

5.1 Public Validity Testing

Recall that in both systems, the decryptor needs to verify the ciphertext before attempting to decrypt or decapsulate it. In our descriptions, this test is efficiently performed using a single exponentiation in $\mathbb{G}$, but requires knowledge of the private key (the exponents $y', y_1, \ldots, y_n$ in the encryption system, or the exponents y_1, y_2 in the KEM).

In the encryption system of Section 3.1, for example, if the public key had included the $\hat{\mathbb{G}}$-elements $h^{y'}, h^{y_1}, \ldots, h^{y_n}$ in addition to the $\mathbb{G}$-elements $u', u_1, \ldots, u_n$, then the validity test could have been performed publicly, using additional applications of the bilinear map, by testing whether the following ratio of bilinear pairings equals the identity element in $\mathbb{G}_T$:

$$\frac{e\big(C_1,\ (h^{y'})\prod_{i=1}^{n}(h^{y_i})^{w_i}\big)}{e(C_2,\ h)} \stackrel{?}{=} 1.$$

Since under such modification the ciphertext validity test does no longer require the private key, we refer to it as the *public validity testing* variant. The principle is the same for the KEM scheme.

Public validity testing still results in chosen ciphertext security,[5] but it requires a lengthier public key. It also increases the decryption burden by an amount of work comparable to a pairing computation (the public test depends on the computation of a ratio of two pairings, which, in the case of the Weil and Tate pairings can be done almost as efficiently as a single pairing, by modifying Miller's algorithm in a manner akin to multi-exponentiation [23]). Public validity testing is used below in the direct construction of a provably CCA2-secure non-interactive threshold system.

5.2 Threshold Decryption

A k-out-of-m threshold public key encryption system is one that allows the private key to be divided up into m shares; each share can then be used to obtain a partial decryption of any given ciphertext, in such a way that the decrypted message can be reconstituted using any k partial decryptions. In a non-interactive threshold system, no communication is needed amongst the k parties performing the partial decryptions, other than their (independent) transmission of the decryption shares to the entity that performs the final reconstitution.

Existing CCA2-secure threshold systems in the standard model, due to Canetti and Goldwasser [12], are based on the Cramer-Shoup system [13]; their system requires interaction between the decryption parties, due to the fact that in the Cramer-Shoup system only parties possessing the private key can check ciphertext validity, which makes threshold decryption non-trivial. Boneh and Boyen [4] and Canetti, Halevi, and Katz [11] recently suggested (without details) to use the BB scheme in combination with the CHK transformation to construct non-interactive CCA2 threshold cryptosystems without random oracles; the details of such a construction were recently worked out in [6]. Their approach starts from the Boneh-Boyen IBE system, suitably modified to provide threshold private key generation; it is then generically transformed into a CCA2-secure threshold public key system using a generalization of the CHK conversion, taking advantage of the fact that in the conversion anyone can check that a ciphertext is valid.

By contrast, we propose a direct approach that trades generality for even more efficiency. Indeed, by applying a simple secret sharing scheme to either the encryption system of Section 3 or the KEM of Section 4, and using the public validity test of Section 5.1, we directly obtain a CCA2-secure non-interactive threshold system in the standard model. Although this approach bears a lot of resemblance to the threshold system from [5] due to its roots in identity-based techniques, we are able to sidestep the generic IBE-to-CCA2 conversion (and the signing step it requires) by virtue of the inherently chosen ciphertext security of our underlying public-key construction.

A detailed construction of the key encapsulation version of the threshold system may be found in Appendix D.

[5] The feasibility of public testing is actually essential to chosen ciphertext security with our approach, since the simulator relies on it to properly answer the decryption queries in the proofs of security (see, e.g., Appendix B).

5.3 Hierarchical Identity-Based Encryption and (H)IB-KEM

Since our constructions are based on the Waters and Boneh-Boyen IBE systems, both of which support hierarchical key generation [19, 18], a natural question to ask is whether the same approach can be applied to directly obtain CCA2-secure HIBE systems without having to resort to use a signature, a MAC, or any other manner of exogenous integrity check.

It is easy to see that we obtain the desired result very simply, by extending the hierarchy in either HIBE construction by one level, and setting the "identity" for that last level to be the hash value of the previous ciphertext components. This gives us (in the Waters case) an adaptive-identity CCA2-secure HIBE, and (in the Boneh-Boyen case) a selective-identity CCA2-secure HIB-KEM.

With a twist, the same approach can be also used with the constant size ciphertext HIBE recently proposed by Boneh, Boyen, and Goh [5]. One of the features of their system is that it is algebraically compatible with the Boneh-Boyen (H)IBE; indeed its authors show how to design a hybrid of the two systems. In such a hybrid, it is straightforward to design the last level of the hierarchy to be a "Boneh-Boyen" level, and use it for our "checksum" construction. This results in a selective-identity, CCA2-secure hierarchical identity-based KEM with short ciphertexts.

6 Discussion and Comparisons

In this section we draw comparisons between our scheme and the other CCA2-secure encryption schemes built from identity-based encryption. Additionally, we describe qualatative differences between the methods of deriving a CCA2-secure encryption scheme from IBE and previous methods that fit under the two-key paradigm as described by Elkind and Sahai [16].

We begin by examining the commonalities between the three CCA2-secure schemes derived from IBE: CHK [11], BK [9] and ours. All three techniques follow a similar method in their proof simulation. After the setup phase there are a certain set of well-formed ciphertexts that the simulator can decrypt corresponding to "identities" that the simulator knows the private keys for. The simulator is unable to decrypt the remainder of the well-formed ciphertexts; these can therefore be used as challenge ciphertexts in the simulation. These ciphertexts that the simulator cannot decrypt correspond to identities for which the simulator does not know the private key.

All three IBE-like methods are fundamentally different from those that fit under the two-key paradigm [16], where a ciphertext consists of the dual encryption of the same message, accompanied with some non-interactive zero-knowledge proof that the two messages are the same. In these systems, the simulator always possesses one of the two keys, and is thus able to decrypt *all* well-formed ciphertexts by decrypting one or the other component. Consequently, the adversary must be challenged on a ciphertext that is not well formed. Using this type of simulation, clearly an adversary must not be able to tell whether a ciphertext is well formed or not; otherwise, the adversary could distinguish the challenge ciphertext from a normal one. In contrast, challenge ciphertexts in IBE-like schemes are always well formed, and (except in BK) anyone is able to tell.

We now focus on the differences between the IBE-like constructions. We just saw that in these, there are a small fraction of ciphertexts that the simulator cannot decrypt, which correspond to a set for which the simulator cannot generate private keys; these identities form the "challenge set".

In the CHK and BK schemes, the challenge set corresponds to a single identity. The well-formed ciphertexts are defined to be all ciphertexts that include a valid signature (or MAC) on the rest of the ciphertext, which the simulator uses to check that query ciphertexts are well formed. Thus, the

PKE & KEM	Ciphertext overhead (#bits, extras...)	Encryption operations (#pairings, [m-exp, r-exp,	Decryption opers. f-exp]:[w/o pre-comp], ...)	Generality	Threshold decryption
KD (KEM)	$2\|p\|$ (+hybr. redund.)	0, [0, 0, 4]:[1, 2, 0]	0, [1, 0, 0]	—	—
CHK/BB_1	$2\|p\|$, verific. key, sig.	0, [0, 0, 4]:[1, 2, 0], Sign	1, [1, 0, 0], Verify	∀ IBE	✓
CHK/BB_2	$2\|p\|$, verific. key, sig.	0, [0, 0, 4]:[1, 2, 0], Sign	1, [0, 1, 1], Verify	∀ IBE	✓
BK/BB_1	$2\|p\|$, commitm., mac	0, [0, 0, 4]:[1, 2, 0]	1, [1, 0, 0]	∀ IBE	—
BK/BB_2	$2\|p\|$, commitm., mac	0, [0, 0, 4]:[1, 2, 0]	1, [0, 1, 1]	∀ IBE	—
§3: PKE	$2\|p\|$	0, [0, 1, $2+1^{\dagger}$]:[0, $3+1^{\ddagger}$, 0]	1, [0, 1, 0]	—	✓
§4: KEM	$2\|p\|$	0, [0, 0, 4]:[1, 2, 0]	1, [0, 1, 0]	—	✓

† With pre-computation, the subset product in §3 can be computed as efficiently as a fixed-base exponentiation.

‡ Without pre-computation, the subset product in §3 is comparable to a regular exponentiation.

Table 1: Summary of the aspects of the various CCA2-secure PK systems from IBE. When applicable, both the encryption and KEM are considered together. The CHK [11] and BK [9] methods are each instianted with the main and second Boneh-Boyen IBE schemes (BB_1 and BB_2) from [4]. The Kurosawa-Desmedt (KD) [21] KEM/DEM hybrid system is also listed for reference; it incurs some symmetric-key encryption overhead that should be excluded for comparison purposes. In determining overheads, the size of any message is discounted to place encryption and KEM on equal footing. In counting numbers of operations, exponentiations are allocated optimally between regular exponentiations, multi-exponentiations, and exponentiations to a fixed base, with increasing preference. Tallies that exclude fixed bases are listed to capture situations where pre-computations are impractical for encryption. Indicative relative timings for the common operations follow: pairing ≈ 2–5, multi-exponentiation ≥ 1.5, regular exponentiation $\equiv 1$, fixed-base exponentiation $\ll 0.2$.

simulator has to arrange for the identity in the challenge set to match the identity that corresponds to the challenge it wishes to craft. In both CHK and BK, this is done "externally" by letting the identity depend on a MAC or signature key.

In our full encryption scheme, the challenge set of identities, for which the simulator does not know the private keys, is larger. Also, rather than being dependent on a MAC or signature key, the encryption identity is derived from the first two elements of the ciphertext. In our KEM scheme, we are back to using a challenge set containing a single identity, but instead of using an external MAC or signature key to select that identity independently of the adversary input, we simply relax the encryption to a key encapsulation. In both cases, a well-formed ciphertext is one whose some of the components (including the one that depends on the identity) form a Diffie-Hellman tuple, which can be easily checked by the simulator using the bilinear map.

Essentially we are able to take advantage of specific properties of the Waters and Boneh-Boyen IBE constructions, respectively, to test ciphertexts for being well formed without *any* additional overhead without having to rely on an external integrity check (such as a signature, a MAC, or an auxiliary symmetric encryption component in a hybrid system). However, since we take advantage of the algebraic properties of the underlying IBE scheme, without actually constructing a true *identity-based* ciphertext, our approach is not generic.

To wrap up, we have summarized in Table 6 a performance comparison between PK systems and KEMs constructed using either of the three methods. The table borrows some figures from Boneh et al. [7], and also includes the Kurosawa-Desmedt [21] hybrid system for reference.

7 Conclusions

We described a CCA2-secure encryption system and a CCA2-secure key encapsulation mechanism respectively based off the identity-based encryption schemes of Waters, and Boneh and Boyen. Our method takes advantage of special properties of these systems that we use to improve upon previous CCA2-secure systems constructed from identity-based encryption schemes [11, 9]. In particular, we showed that our schemes have advantages in terms of simplicity, ciphertext size, and decryption time, and are well suited for threshold cryptography.

Acknowledgments

The authors wish to thank Eike Kiltz and the anonymous ACM CCS referees for helpful comments. We also thank Hovav Shacham for pointing out that our approach applies to both BB_1 and BB_2.

References

[1] Masayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 128–146. Springer-Verlag, 2005.

[2] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. `http://eprint.iacr.org/`.

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

[4] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, Lecture Notes in Computer Science. Springer Verlag, 2004.

[5] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer-Verlag, 2005.

[6] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *Proceedings of RSA-CT 2006*. Springer-Verlag, 2006. To appear. Available at `http://crypto.stanford.edu/~dabo/abstracts/threshold.html`.

[7] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption, 2005. Journal submission. Available at `http://crypto.stanford.edu/~dabo/papers/ccaibejour.pdf`.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology—CRYPTO 2001*, pages 213–229. Springer-Verlag, 2001.

[9] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In *Proceedings of RSA-CT 2005*. Springer-Verlag, 2005.

[10] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology—EUROCRYPT 2003*. Springer-Verlag, 2003.

[11] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology—EUROCRYPT 2004*. Springer-Verlag, 2004.

[12] Ron Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen message attack. In *Advances in Cryptology—EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 90–106. Springer-Verlag, 1999.

[13] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO 1998*, volume 1462 of *LNCS*, 1998.

[14] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology—EUROCRYPT 2002*, volume 2729 of *LNCS*, pages 45–64, 2002.

[15] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *ACM Symposium on Theory of Computing—STOC 1991*, pages 542–552. ACM Press, 1991.

[16] Edith Elkind and Amit Sahai. A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack. Cryptology ePrint Archive, Report 2002/042, 2002. `http://eprint.iacr.org/`.

[17] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 295–310. Springer-Verlag, 1999.

[18] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology—ASIACRYPT 2002*, 2002.

[19] Jeremy Horwitz and Ben Lynn. Towards hierarchical identity-based encryption. In *Advances in Cryptology—EUROCRYPT 2002*, pages 466–81, 2002.

[20] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, 2004.

[21] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *Advances in Cryptology—CRYPTO 2004*, LNCS, pages 426–442. Springer-Verlag, 2004.

[22] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Palo Alto, 1979.

[23] Victor Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4), 2004.

[24] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *ACM Symposium on Theory of Computing—STOC 1990*, pages 427–437, 1990.

[25] Charles Rackoff and Daniel Simon. Non-interactive zeroknowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology—CRYPTO 1991*, volume 576 of *LNCS*. Springer-Verlag, 1991.

[26] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *IEEE Symposium on Foundations of Computer Science—FOCS 1999*, 1999.

[27] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Advances in Cryptology—EUROCRYPT 2000*, LNCS, pages 275–288. Springer-Verlag, 2000.

[28] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, 2002.

[29] Brent Waters. Efficient identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2005*, Lecture Notes in Computer Science. Springer Verlag, 2005.

A Security Proof Sketch for the Encryption Scheme

We now sketch the proof of Theorem 3.1, considered for simplicity in the case where a fixed injective encoding $H_0 : \mathbb{G}_T \times \mathbb{G} \to \{0,1\}^n$ is used.

Proof sketch. Suppose there exists a (t, q, ϵ)-adversary $\mathcal{A}$ against our scheme. We construct a simulator, $\mathcal{B}$, to play the decisional BDH game. The simulator takes as input a BDH challenge $(g, g^a, g^c, h, h^a, h^b, T) \in \mathbb{G}^3 \times \hat{\mathbb{G}}^3 \times \mathbb{G} \times \mathbb{G}_T$, that is sampled either from the distribution of "true" instances, $\mathcal{P}_{BDH}$, in which $T = e(g,h)^{abc}$, or from the distribution of "false" instances, $\mathcal{R}_{BDH}$, in which T is uniform and independent in $\mathbb{G}_T$. The output of the simulator is a guess, β', as to whether the challenge is a BDH tuple. The simulator runs $\mathcal{A}$ executing the following steps.

Setup. The hash index s is fixed to $s = 0$ since we have assumed a fixed injective encoding H_0. To generate a key pair, the simulator first sets an integer, $m = 4q$, and chooses an integer, k, uniformly at random between 0 and n. It then chooses a random n-length vector, $\overrightarrow{x} = (x_i)$, where the elements of $\overrightarrow{x}$ are chosen uniformly at random from the integers between 0 and $m-1$ and a value, x', chosen uniformly at random between 0 and $m-1$. Let X^* denote the pair $(x', \overrightarrow{x})$ Additionally, the simulator chooses a random $y' \in \mathbb{Z}_p$ and an n-length vector, $\overrightarrow{y} = (y_i)$, where the elements of $\overrightarrow{y}$ are chosen at random in $\mathbb{Z}_p$. These values are all kept internal to the simulator.

For a bit string $w \in \{0,1\}^n$ as returned by H_s, we will let $\mathcal{W} \subseteq \{1, \ldots, n\}$ be the set of all i for which the i-th bit $w_i = 1$. For ease of analysis we define three functions. We define $F(v) = (p - mk) + x' + \sum_{i \in \mathcal{W}} x_i$ and define $J(v) = y' + \sum_{i \in \mathcal{W}} y_i$. Finally, we define a binary function $K(v)$ as

$$K(v) = \begin{cases} 0, & \text{if } x' + \sum_{i \in \mathcal{W}} x_i \equiv 0 \pmod{m} \\ 1, & \text{otherwise.} \end{cases}$$

The simulator assigns $g_1 = g^a$ and $g_2 = g^b$ (recall it does not know a or b). It then assigns the public parameters $u' = g_2^{p-km+x'} g^{y'}$ and $u_i = g_2^{x_i} g^{y_i}$ for $i = 1, \ldots, n$. From the perspective of the adversary the distribution of the public parameters is identical to the real construction.

For future internal use, the simulator also assigns $h_1 = h^a$ and $h_2 = h^b$, and computes $v' = h_2^{p-km+x'} h^{y'}$ and $v_i = h_2^{x_i} h^{y_i}$ for $i = 1, \ldots, n$.

Phase 1. The adversary, $\mathcal{A}$, will issue decryption queries. Suppose the adversary issues a query for a ciphertext $C = (C_0, C_1, C_2)$. The simulator determines $w = H_s(C_0, C_1)$. If $K(w) = 0$ the simulator aborts and randomly chooses its guess β' of the challenger's value β. Otherwise, the simulator chooses a random $r \in \mathbb{Z}_p$. Using the technique described by Boneh and Boyen [4] it constructs the "private key", d, that corresponds to the "identity", w, as

$$d = (d_1, d_2) = \left(g_1^{\frac{-J(v)}{F(w)}} (u' \prod_{i \in \mathcal{W}} u_i)^r,\ g_1^{\frac{-1}{F(w)}} g^r \right).$$

We note that for $\tilde{r} = r - \frac{a}{F(w)}$, we have

$$d_1 = g_1^{\frac{-J(v)}{F(v)}} (u' \prod_{i \in v} u_i)^r = g_2^a (u' \prod_{i \in \mathcal{V}} u_i)^{\tilde{r}}, \qquad d_2 = g_1^{\frac{-1}{F(v)}} g^r = g^{\tilde{r}}.$$

This simulator will be able to perform this computation iff $F(v) \neq 0 \mod p$. For ease of analysis the simulator will only continue (not abort) in the sufficient condition where $K(v) \neq 0$. (If we have $K(v) \neq 0$ this implies $F(v) \neq 0 \mod p$ since we can assume $p > nm$ for any reasonable values of p, n, and m).

Finally, the simulator responds to the decryption query with

$$M = C_0 \cdot e(C_2, d_2) / e(C_1, d_1).$$

Challenge. The adversary next will submit two messages $M_0, M_1 \in \mathbb{G}_T$. The simulator flip a fair coin, γ, and constructs the first two elements of the challenge: $C_0^* = T \cdot M_\gamma$ and $C_1^* = g^c$ (which was part of the tuple initially given to the simulator). It then evaluates $w^* = H_s(C_0^*, C_1^*)$. If $x' + \sum_{i \in \mathcal{W}^*} x_i \neq km$, the simulator aborts and submits a random guess for β'. Otherwise, we have $F(w^*) \equiv 0 \pmod p$ and the simulator computes the third ciphertext element: $C_2^* = C^{J(v^*)}$. It outputs the challenge ciphertext

$$C^* = (C_0^*, C_1^*, C_2^*).$$

We note that C^* is a valid encryption of M_γ if the simulator was given a BDH tuple. Otherwise, C^* is independent of γ in the adversary's view.

Phase 2. The simulator responds to further decryption queries on any ciphertext $C \neq C^*$ using the same method it used in Phase 1.

Guess. Finally, the adversary $\mathcal{A}$ outputs a guess γ' of γ.

Artificial Abort. At this point the simulator is still unable to use the output from the adversary. An adversary's probability of success could be correlated with the probability that the simulator needs to abort. This stems from the fact that two different sets of q private key queries may cause the simulator to abort with different probabilities.

The simulator corrects for this by forcing all possible sets of queries of the adversary to cause the simulator to abort with (almost) the same probability (1-λ), where (1-λ) is a lower bound on any set of private key queries causing an abort before this stage.

Let $\overrightarrow{w} = w_1, \ldots w_q$ denote the "identities" corresponding to the decryption queries made in Phase 1 and Phase 2. Let w^* denote the challenge identity and let $\mathcal{W}^* \subseteq \{1, \ldots, n\}$ be the set of all i for which $w_i^* = 1$. (All of these values are defined at this point in the simulation.) First, we define the function $\tau(X', \overrightarrow{w}, w^*)$, where W' is a set of simulation values $x', x_1, \ldots, x_n$, as

$$\tau(X', \overrightarrow{w}, w^*) = \begin{cases} 0, & \text{if } (\bigwedge_{i=1}^{q} K(w_i) = 1) \wedge x' + \sum_{i \in \mathcal{W}^*} x_i = km \\ 1, & \text{otherwise.} \end{cases}$$

The function $\tau(X', \overrightarrow{w}, w^*)$ will evaluate to 0 if the private key and challenge queries $\overrightarrow{w}, w^*$ will not cause an abort for a given choice of simulation values, X'. We can now consider the probability over the simulation values for a given set of queries, $\overrightarrow{w}, w^*$, as $\eta = \Pr_{X'}[\tau(X', \overrightarrow{w}, w^*) = 0]$.

The simulator samples $O(\epsilon^{-2}\ln(\epsilon^{-1})\lambda^{-1}\ln(\lambda^{-1}))$ times the probability η by choosing a random X' and evaluating $\tau(X', \overrightarrow{w}, w^*)$ to compute an estimate η'. We emphasize that the sampling does not involve running the adversary again. Let $\lambda = \frac{1}{8nq}$, be the lower bound on the probability of not aborting for any set of queries. (We show how to calculate λ below.) Then if $\eta' \geq \lambda$ the simulator will abort with probability $\frac{\eta'-\lambda}{\eta'}$ (not abort with probability $\frac{\lambda}{\eta'}$) and take a random guess β'. Otherwise, the simulator will not abort.

Output. If the simulator has not aborted at this point it will take check to see if the adversary's guess, $\gamma' = \gamma$. If $\gamma' = \gamma$ then the simulator outputs a guess $\beta' = 1$, otherwise it outputs $\beta = 0$.

This concludes the description of the simulator. By analogy with the simulator in the Waters IBE scheme, we refer to [29] for the derivation of the claimed bounds. □

B Security Proof for the Key Encapsulation Scheme

We now prove Theorem 4.1 by showing a (tight) reduction from Decision BDH to the KEM security. For simplicity, we consider the case where a fixed injective encoding $H_0 : \mathbb{G} \to \mathbb{Z}_p$ is used. (With the collision resistant family H_s, one would add a correcting term ϵ'' to account for the collision probability.)

We start with the definition of the KEM-CCA2 attack, which was omitted from Section 2.2.

Setup. The challenger runs *KeyGen*(λ) to obtain a random instance of public and private key pair (PK, SK). It gives the public key to the adversary.

Query phase 1. The adversary adaptively issues decapsulation queries C where $C \in \{0,1\}^*$. The challenger responds with *Decapsulate*(SK, C).

Challenge. The adversary signals that it is ready to be challenged. The challenger runs *Encapsulate*(PK) to obtain a ciphertext C^* and a session key $K^\dagger$. It then picks a random $b \in \{0,1\}$. If $b = 1$ it sets $K^* = K^\dagger$, otherwise it sets K^* to a random session key of equal length. It gives C^* and K^* to the adversary.

Query phase 2. The adversary continues to issue decapsulation queries C as in phase 1, with the added constraint that $C \neq C^*$. The challenger responds with *Decapsulate*(SK, C).

Guess. Algorithm $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

Proof of Theorem 4.1. Suppose $\mathcal{A}$ has advantage ϵ in attacking the KEM. We build an algorithm $\mathcal{B}$ that solves the Decision BDH problem in $(\mathbb{G}, \hat{\mathbb{G}})$ with advantage $\epsilon' \geq (\epsilon - q/2p)$. Algorithm $\mathcal{B}$ is given as input a random 7-tuple $(g, g^a, g^c, h, h^a, h^b, T) \in \mathbb{G}^3 \times \hat{\mathbb{G}}^3 \times \mathbb{G}_T$, that is sampled either from the distribution of "true" instances, $\mathcal{P}_{BDH}$, where $T = e(g,h)^{abc}$, or from the distribution of "false" instances, $\mathcal{R}_{BDH}$, where T is uniform and independent in $\mathbb{G}_T$. Algorithm $\mathcal{B}$'s goal is to output 1 if $T = e(g,h)^{abc}$ and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in an KEM-CCA2 game as follows:

Setup. The hash index s is fixed to $s = 0$ since we have assumed a fixed injective encoding H_0. To generate a key pair, the simulator $\mathcal{B}$ starts by calculating $w^* = H_s(g^c)$ from the given g^c. Then, $\mathcal{B}$ selects a random $\delta \in \mathbb{Z}_p$ and defines

$$\begin{aligned} u_1 &= (g^a)^{-w^*} g^\delta \in \mathbb{G}, & u_2 &= (g^a) \in \mathbb{G}, \\ v_1 &= (h^a)^{-w^*} h^\delta \in \hat{\mathbb{G}}, & v_2 &= (h^a) \in \hat{\mathbb{G}}, \end{aligned}$$
$$Z = e(g^a, h^b) \in \mathbb{G}_T.$$

The simulator $\mathcal{B}$ gives to $\mathcal{A}$ the public key: $\big(s,\ Z,\ u_1,\ u_2\big)$. The corresponding private key is $(h_0 = h^{ab},\ y_1 = \delta - aw^*,\ y_2 = a)$, although $\mathcal{B}$ cannot compute any of its components.

Phase 1. The adversary $\mathcal{A}$ issues up to q decapsulation queries, one at a time. Consider a query for a ciphertext $C = (C_1, C_2)$, and let $w = H_s(C_1)$. Algorithm $\mathcal{B}$ first determines whether the ciphertext is valid, by checking that, in the group $\mathbb{G}_T$,

$$e(C_1,\ v_1 v_2^w) \,/\, e(C_2,\ h) \stackrel{?}{=} 1.$$

If the equality does not hold, then the ciphertext is invalid and $\mathcal{B}$ responds to the query with $\perp$. Otherwise, the ciphertext is valid, thus $C = (C_1, C_2) = (g^t, u_1^t u_2^{tw})$ for some unknown $t \in \mathbb{Z}_p$.

If $w = w^*$, then $\mathcal{B}$ is unable to respond; in this case, $\mathcal{B}$ terminates the simulation with $\mathcal{A}$, outputs a random bit $b \in \{0,1\}$, and halts. If instead $w \neq w^*$, it computes, in the group $\mathbb{G}_T$,

$$K = \frac{e\big(C_1,\ (h^b)^{\frac{-\delta}{w-w^*}} v_1 v_2^w\big)}{e\big(C_2,\ (h^b)^{\frac{-1}{w-w^*}} h\big)} = \left(\frac{e(g,h)^{\delta + a(w-w^*) - b\frac{\delta}{w-w^*}}}{e(g,h)^{\delta + a(w-w^*) - b\frac{\delta}{w-w^*} - ab\frac{w-w^*}{w-w^*}}} \right)^t = e\big(g,\ h\big)^{abt} = Z^t.$$

The simulator $\mathcal{B}$ responds to the query with K, which is the decapsulated session key.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it signals to $\mathcal{B}$ that it is ready to accept the challenge. In response, $\mathcal{B}$ gives $\mathcal{A}$ the ciphertext $C^* = \big(g^c,\ (g^c)^\delta\big) \in \mathbb{G}^2$ and the session key $K^* = T \in \mathbb{G}_T$.

Notice that the challenge ciphertext is valid since $C^* = (g^c, u_1^c u_2^{cw^*})$ for $w^* = H_s(g^c)$. The corresponding randomization exponent, c, is unknown to $\mathcal{B}$. The challenge session key, K^*, will be valid when $T = e(g,h)^{abc}$, that is, when $\mathcal{B}$'s input is sampled from $\mathcal{P}_{BDH}$; otherwise, when $\mathcal{B}$'s input is sampled from $\mathcal{R}_{BDH}$, then K^* is independent of C^* in the adversary's view.

Phase 2. The adversary $\mathcal{A}$ continues to issue any remaining queries from its original quota of q. The simulator $\mathcal{B}$ responds as in Phase 1.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$, where $b' = 1$ is a guess that K^* is the correct key. Algorithm $\mathcal{B}$ concludes its own game by forwarding b' as its own output b, where $b = 1$ means that it guesses that $T = e(g,h)^{abc}$.

To conclude, we observe that as long as the simulator does not abort, the adversary's view in the simulation is identical to its view in a real attack. By our hypothesis $\mathcal{A}$ must make a correct guess with probability at least $\frac{1}{2} + \epsilon$. Since for q distinct queries the simulator aborts with probability q/p, and when this happens makes a wrong guess with probability $\frac{1}{2}$, it follows that $\mathcal{B}$'s success probability in its own game at least $\frac{1}{2} + \epsilon - \frac{q}{2p}$. Therefore, $\mathcal{B}$'s advantage in the Decision BDH game is $(\epsilon - q/2p) \geq \epsilon'$, which completes the proof of the theorem. $\square$

C Concrete Injective Encodings on Curves

We now briefly revisit the issue of devising an efficiently computable injective encoding H_0 for the schemes of Sections 3 and 4.

In the encryption scheme of Section 3, we needed an injective function $H_0 : \mathbb{G}_T \times \mathbb{G} \to \{0,1\}^n$. Such a function is not difficult construct, if we are allowed to choose a large enough value of n. The only constraint is that $n \geq \lceil 2 \log_2 p \rceil$ since $|\mathbb{G}_T| = |\mathbb{G}| = p$. (For example, a trivial construction is to let H_0 return the concatenation of the binary representations of the two input elements.)

In the KEM scheme of Section 4, the difficulty is that there is no flexibility to choose the codomain of $H_0 : \mathbb{G} \to \mathbb{Z}_p$. Since $|\mathbb{G}| = |\mathbb{Z}_p| = p$, the function H_0 would have to be a bijection, which might complicate its contruction given that H_0 is required to be be efficiently computable. However, we can relax these requirements if we take $H_0 : \mathbb{G} \to \mathbb{Z}_p \cup \{\perp\}$ and allow it to assume the distinguished value $\perp$ on some portion of its domain, only requiring it to be injective over the complement. Then, we can modify the encapsulation algorithm to try new randomization values t until it finds one that results in $H_0(C_1) = H_0(g^t) \neq \perp$.

These small modifications can greatly simplify the construction of H_0 in the context of bilinear groups defined on elliptic curves. Let $E(\mathbb{F}_q)$ be the group of points on a curve E over some finite field $\mathbb{F}_q$. Except for the "point at infinity", all points on the curve are represented by pairs of coordinates $(x, y) \in (\mathbb{F}_q)^2$. We mention two cases of practical interest:

1. If $E(\mathbb{F}_q)$ has prime order p, as is the case for the recent BN family of pairing-friendly curves [2], then we take $\mathbb{G} = E(\mathbb{F}_q)$, and by Hasse's theorem we know that $p \in [q+1-2\sqrt{q},\, q+1+2\sqrt{q}]$. In this case, the map $Y : E(\mathbb{F}_q) \to \mathbb{F}_q : (x, y) \mapsto y$ that merely returns a point's y coordinate is injective on an overwhelming fraction of its domain. To obtain $H_0 : \mathbb{G} \to \mathbb{F}_p \cup \{\perp\}$, we start from the map Y and modify it to output $\perp$ on any input point (x, y) whose y coordinate either is greater than $p - 1$ or admits a second point $(x', y) \in E(\mathbb{F}_q)$.

2. If $E(\mathbb{F}_q)$ is a supersingular curve of order $q + 1 = rp$ for a large prime p and a tiny cofactor r (such as $r = 12$), then the map $Y : E(\mathbb{F}_q) \to \mathbb{F}_q : (x, y) \mapsto y$ is a bijection provided that we remove the point at infinity from the domain. In this case, $\mathbb{G}$ will be a subgroup of $E(\mathbb{F}_p)$ of order p, and hence a random point $g^t \in \mathbb{G}$ will have a probability $1/r$ of satisfying $y \leq p - 1$. To evaluate H_0, we use the map Y and change its output to $\perp$ whenever it exceeds $p - 1$.

In the first case, there is hardly any efficiency penalty. The practicality of the method in the second case will depend on the practicality of conducting the expected $r - 1$ extra exponentiations in $\mathbb{G}$ (i.e., point multiplications on the curve) for each encapsulation.

As mentioned, we can sidestep these issues with the generalized injective encoding $H_0 : \mathbb{G} \to \mathbb{Z}_p^\nu$ presented in Section 4.3, or a collision resistant family $\{H_s\}$ as described earlier.

D Direct Non-Interactive Threshold KEM Construction

We now give a detailed description of the non-interactive threshold system constructed using our identity-based technique, adapted from the IBE-based threshold system of [6]. We give the key encapsulation version, which has tight CCA2-security against (statically corrupted) coalitions of malicious decryption (or decapsulation) servers. For simplicity, we only describe the key generation procedure using a centralized dealer, although it is straightforward to extend it into a distributed key generation protocol using known methods [17].

We refer to [12, 28] for the formal definition of threshold encryption and the various subtleties of its security model. The security of our system is based on the Decision BDH assumption, and does not require random oracles (or even collision resistance). We refer the reader to [6] for precisions on how to show a security reduction for this type of constructions.

Key Generation: For k-out-of-m secret sharing, a user's key generation proceeds as follows.

First, a secret $\alpha_0 \in \mathbb{Z}_p$ is chosen at random, from which the value $Z_0 = e(g,h)^{\alpha_0}$ is calculated. Next, the algorithm chooses y_1 and y_2 at random from $\mathbb{Z}_p$. It then calculates $u_1 = g^{y_1}$ and $u_2 = g^{y_2}$ in $\mathbb{G}$, and also calculates $v_1 = h^{y_1}$ and $v_2 = h^{y_2}$ in $\hat{\mathbb{G}}$. Last, a random seed s for the collision resistant family is chosen (letting $s = 0$ when a fixed injection H_0 is used).

To obtain the m private key shares, select a random polynomial $f = \alpha_0 + \alpha_1 X + \ldots + \alpha_{k-1} X^{k-1}$ of degree $k-1$ in $\mathbb{Z}_p[X]$, where the constant term is α_0. Let $f(i) \in \mathbb{Z}_p$ be the value of f evaluated at $i = 1, \ldots, m$. The m private key shares respectively given to the m decapsulation servers are

$$\left(\ h_1 = h^{f(1)},\ \ \ldots,\ \ h_m = h^{f(m)}\ \right) \quad \in\ \hat{\mathbb{G}}^m.$$

The public key is published as

$$\Big(\ s,\ \ Z_0 = e(g,h)^{\alpha_0},\ \ u_1 = g^{y_1},\ \ u_2 = g^{y_2},\ \ v_1 = h^{y_1},\ \ v_2 = h^{y_2}\ \Big) \quad \in\ \{s\} \times \mathbb{G}_T \times \mathbb{G}^2 \times \hat{\mathbb{G}}^2.$$

Additionally, there is a public verification key, which allows the various parties to verify the private key shares and decapsulation shares given to them. A random $\delta \in \mathbb{Z}_p$ is chosen, and the verification key is published as

$$\Big(\ Z_1 = e(g,h)^{\alpha_1},\ \ \ldots,\ \ Z_{k-1} = e(g,h)^{\alpha_{k-1}},$$
$$\ell = g^{\delta},\ \ \ell_1 = u_1^{\delta},\ \ \ell_2 = u_2^{\delta},\quad \ell'_1 = g^{f(1)\delta},\ \ \ldots,\ \ \ell'_m = g^{f(m)\delta}\ \Big) \quad \in\ \mathbb{G}_T^{k-1} \times \mathbb{G}^{3+m}.$$

Private Share Verification: The i-th decapsulation server can verify that his assigned private key share h_i is correct, using Feldman's method. To do so, the i-th share holder tests whether

$$e(g,h_i) \stackrel{?}{=} \prod_{j=0}^{k-1} Z_j^{(i^j)} = e(g,h)^{\sum_{j=0}^{k-1}(\alpha_j X^j)}\big|_{X=i} = e(g,h)^{f(i)}.$$

Encapsulation: The process works essentially as in Section 4.1 (substituting $\alpha = \alpha_0$ and $Z = Z_0$).

First, a value $t \in \mathbb{Z}_p$ is randomly chosen, and the algorithm computes the first component of the ciphertext: $C_1 = g^t \in \mathbb{G}$. Next, it computes $w = H_s(C_1) \in \mathbb{Z}_p$, and then the second ciphertext component: $C_2 = u_1^t u_2^{tw} \in \mathbb{G}$. The complete ciphertext, C, is the pair

$$\left(\ C_1 = g^t,\ \ C_2 = u_1^t u_2^{tw}\ \right) \quad \in\ \mathbb{G}^2.$$

The corresponding session key, K, is the group element $\left(K = Z_0^t\right)\ \in \mathbb{G}_T$.

Partial Decapsulation: A ciphertext $C = (C_1, C_2)$ is partially decapsulated by the holder of the i-th decapsulation share $h_i = h^{f(i)}$ as follows.

First, the algorithm verifies that the ciphertext is well formed, using the v-values in the public key. To do so, the algorithm computes $w = H_s(C_1) \in \mathbb{Z}_p$, and determines whether, in $\mathbb{G}_T$,

$$e(C_1,\ v_1 v_2^w) \,/\, e(C_2,\ h) \stackrel{?}{=} 1.$$

If the two sides are unequal, then $(g, C_1, u_1 u_2^w, C_2)$ is not a Diffie-Hellman tuple, and the algorithm outputs $(i, \perp)$. Otherwise, the ciphertext is valid, in which case the algorithm picks a random $r_i \in \mathbb{Z}_p$, computes $d_i = h_i v_1^{r_i} v_2^{r_i w}$ and $d_i' = h^{r_i}$, and outputs the triple

$$\Big(\ i,\ \ d_i = h_i\, v_1^{r_i}\, v_2^{r_i w},\ \ d_i' = h^{r_i}\ \Big) \quad \in\ \{1, \ldots, m\} \times \hat{\mathbb{G}}^2.$$

The pair (d_i, d_i') can be viewed as the i-th share of a (randomized) identity-based private key tailored to the specific "identity" w of the given ciphertext. The index i is also returned as it is needed for the final reconstitution.

Decapsulation Share Verification: The combiner can verify that the partial decapsulation of a given ciphertext $C = (C_1, C_2)$ by the i-th decapsulation server is correct, as follows. Let $w = H_s(C_1)$. The first task is to test that C is valid, by checking that, in the group $\mathbb{G}_T$,

$$e(C_1,\ v_1 v_2^w) \,/\, e(C_2,\ h) \stackrel{?}{=} 1.$$

If C is invalid, then the algorithm outputs `valid` if the decapsulation share is of the form $(i, \perp)$, and `invalid` if not. If the decapsulation share is of the form $(i, \perp)$ even though C is well formed, the algorithm also outputs `invalid`.

At this point, we know that C is well formed and that the decapsulation share to verify is of the form (i, d_i, d_i'). The algorithm verifies that, in the group $\mathbb{G}_T$,

$$e(\ell_i', h) \cdot e(\ell_1 \ell_2^w, d_i') \,/\, e(\ell, d_i) \stackrel{?}{=} 1.$$

If this holds, the algorithm output `valid`, otherwise it outputs `invalid`.

Reconstitution: The session key is reconstituted from k partial decapsulations of a ciphertext $C = (C_1, C_2)$ as follows. Let $w = H_s(C_1)$. The combiner must first verify that C is a valid ciphertext, which is done as usual by checking that

$$e(C_1,\ v_1 v_2^w) \,/\, e(C_2,\ h) \stackrel{?}{=} 1.$$

Let then $S = \{(i, d_i, d_i')\}$ be the (indexed) set of available partial decapsulations of C, where S has size at least k. We require that all indices i in S be distinct, and that all partial decapsulations in S be valid for the given ciphertext, which can be checked using the Decapsulation Share Verification method. Let I be the set of indices i that appear in S. W.l.o.g., we assume that $|S| = |I| = k$, as can always discard decapsulation shares in excess.

The reconstitution algorithm first reassembles a complete identity-based private key for the target ciphertext, as

$$\Big(\ d = \prod_{i \in I} (d_i)^{\Lambda_i(0)},\ \ d' = \prod_{i \in I} (d_i')^{\Lambda_i(0)}\ \Big) \quad \in\ \hat{\mathbb{G}}^2, \qquad \text{where}\quad \Lambda_i(x) = \prod_{j \in I \setminus \{i\}} \frac{x - j}{i - j}\ \in\ \mathbb{Z}_p.$$

Then, the session key is reconstituted and output, as

$$K = \frac{e(C_1,\ d)}{e(C_2,\ d')} \quad \in\ \mathbb{G}_T.$$

Indeed, the $\Lambda_i(0)$ defined above are the Lagrange coefficients for interpolating a $(k-1)$-degree polynomial from I to 0. Thus, for any set S of k valid decapsulation shares, we easily find that

$$d = \prod_{i \in I} (h_i\, v_1^{r_i}\, v_2^{r_i w})^{\Lambda_i(0)} = h_0\, v_1^{r}\, v_2^{rw}, \quad \text{and} \quad d' = \prod_{i \in I} (h^{r_i})^{\Lambda_i(0)} = h^r, \quad \text{for some } r \in \mathbb{Z}_p.$$

Therefore, $K = e(g^t, d)/e(u_1^t u_2^{tw}, d') = e(g, h_0)^t \cdot e(g, v_1 v_2^w)^{rt}/e(u_1 u_2^w, h)^{rt} = e(g, h_0)^t \cdot 1 = Z^t$.