Identity-Based Encryption from the Weil Pairing

Dan Boneh*
dabo@cs.stanford.edu

Matthew Franklin†
franklin@cs.ucdavis.edu

Abstract

We propose a fully functional identity-based encryption scheme (IBE). The scheme has chosen ciphertext security in the random oracle model assuming an elliptic curve variant of the computational Diffie-Hellman problem. Our system is based on bilinear maps between groups. The Weil pairing on elliptic curves is an example of such a map. We give precise definitions for secure identity based encryption schemes and give several applications for such systems.

1 Introduction

In 1984 Shamir [32] asked for a public key encryption scheme in which the public key can be an arbitrary string. In such a scheme there are four algorithms: (1) setup generates global system parameters and a master-key, (2) extract uses the master-key to generate the private key corresponding to an arbitrary public key string $\mathsf{ID} \in \{0,1\}^*$, (3) encrypt encrypts messages using the public key ID, and (4) decrypt decrypts messages using the corresponding private key.

Shamir's original motivation for identity-based encryption was to simplify certificate management in e-mail systems. When Alice sends mail to Bob at `bob@hotmail.com` she simply encrypts her message using the public key string "`bob@hotmail.com`". There is no need for Alice to obtain Bob's public key certificate. When Bob receives the encrypted mail he contacts a third party, which we call the Private Key Generator (PKG). Bob authenticates himself to the PKG in the same way he would authenticate himself to a CA and obtains his private key from the PKG. Bob can then read his e-mail. Note that unlike the existing secure e-mail infrastructure, Alice can send encrypted mail to Bob even if Bob has not yet setup his public key certificate. Also note that key escrow is inherent in identity-based e-mail systems: the PKG knows Bob's private key. We discuss key revocation, as well as several new applications for IBE schemes in the next section.

Since the problem was posed in 1984 there have been several proposals for IBE schemes [8, 34, 33, 24, 19] (see also [26, p. 561]). However, none of these are fully satisfactory. Some solutions require that users not collude. Other solutions require the PKG to spend a long time for each private key generation request. Some solutions require tamper resistant hardware. It is fair to say that constructing a usable IBE system is still an open problem. Interestingly, the related notions of identity-based signature and authentication schemes, also introduced by Shamir [32], do have satisfactory solutions [12, 11].

In this paper we propose a fully functional identity-based encryption scheme. The performance of our system is comparable to the performance of ElGamal encryption in $\mathbb{F}_p^*$. The security of our system is based on a natural analogue of the computational Diffie-Hellman assumption on elliptic curves. Based on this assumption we show that the new system has chosen ciphertext security in the random oracle

*Supported by DARPA contract F30602-99-1-0530 and the Packard Foundation.
†Supported by an NSF Career Award.

model. Using standard techniques from threshold cryptography [16, 17] the PKG in our scheme can be distributed so that the master-key is never available in a single location. Unlike common threshold systems, we show that robustness for our distributed PKG is free.

Our IBE system can be built from any bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between two groups $\mathbb{G}_1, \mathbb{G}_2$ as long as a variant of the Computational Diffie-Hellman problem in $\mathbb{G}_1$ is hard. We use the Weil pairing on elliptic curves as an example of such a map. Until recently the Weil pairing has mostly been used for attacking elliptic curve systems [25, 14]. Joux [20] recently showed that the Weil pairing can be used for "good" by using it in a protocol for three party one round Diffie-Hellman key exchange. Using similar ideas, Verheul [35] recently constructed an ElGamal encryption scheme where each public key has two corresponding private keys. In addition to our identity-based encryption scheme, we show how to construct an ElGamal encryption scheme with "built-in" key escrow, i.e., where one global escrow key can decrypt ciphertexts encrypted under any public key.

To argue about the security of our IBE system we define chosen ciphertext security for identity-based encryption. Our model gives the adversary more power than the standard model for chosen ciphertext security [29, 1]. The reason is that while mounting a chosen ciphertext attack on the public key ID, the attacker can obtain from the PKG the private key of some public key $\mathsf{ID}' \neq \mathsf{ID}$. This private key might help the attacker. Hence, during the chosen ciphertext attack we allow the attacker to obtain the private key for any public key of her choice other than the one on which the attacker is being challenged. Even with the help of such queries the attacker should have negligible advantage in defeating the semantic security of the system.

The rest of the paper is organized as follows. Several applications of identity-based encryption are discussed in Section 1.1. We then give precise definitions and security models in Section 2. We describe bilinear maps with certain properties in Section 3. Our identity-based encryption scheme is presented in Section 4 using general bilinear maps. Then a concrete identity based system from the Weil pairing is given in Section 5. Some extensions and variations (efficiency improvements, distribution of the master-key) are considered in Section 6. Our construction for ElGamal encryption with a global escrow key is described in Section 7. We end with conclusions and some open problems.

1.1 Applications for Identity-Based Encryption

The original motivation for identity-based encryption is to help the deployment of a public key infrastructure. In this section, we show several other unrelated applications.

1.1.1 Revocation of Public Keys

Public key certificates contain a preset expiration date. In an IBE system key expiration can be done by having Alice encrypt e-mail sent to Bob using the public key: "`bob@hotmail.com || current-year`". In doing so Bob can use his private key during the current year only. Once a year Bob needs to obtain a new private key from the PKG. Hence, we get the effect of annual private key expiration. Note that unlike the existing PKI, Alice does not need to obtain a new certificate from Bob every time Bob refreshes his certificate.

One could potentially make this approach more granular by encrypting e-mail for Bob using "`bob@hotmail.com || current-date`". This forces Bob to obtain a new private key every day. This might be feasible in a corporate PKI where the PKG is maintained by the corporation. With this approach key revocation is quite simple: when Bob leaves the company and his key needs to be revoked, the corporate PKG is instructed to stop issuing private keys for Bob's e-mail address. The interesting property is that Alice does not need to communicate with any third party to obtain Bob's daily public key. This approach enables Alice to send messages into the future: Bob will only be able to decrypt the e-mail on the date specified by Alice (see [30, 9] for methods of sending messages into the future using a stronger security model).

1.1.2 Delegation of Decryption Keys

Another application for IBE systems is delegation of decryption capabilities. We give two example applications. In both applications the user Bob plays the role of the PKG. Bob runs the setup algorithm to generate his own IBE system parameters params and his own master-key. Here we view params as Bob's public key. Bob obtains a certificate from a CA for his public key params. When Alice wishes to send mail to Bob she first obtains Bob's public key params from Bob's public key certificate. Note that Bob is the only one who knows his master-key and hence there is no key-escrow with this setup.

1. **Delegation to a laptop.** Suppose Alice encrypts mail to Bob using the current date as the IBE encryption key (she uses Bob's params as the IBE system parameters). Since Bob has the master-key he can extract the private key corresponding to this IBE encryption key and then decrypt the message. Now, suppose Bob goes on a trip for seven days. Normally, Bob would put his private key on his laptop. If the laptop is stolen the private key is compromised. When using the IBE system Bob could simply install on his laptop the seven private keys corresponding to the seven days of the trip. If the laptop is stolen, only the private keys for those seven days are compromised. The master-key is unharmed. This is analogous to the delegation scenario for *signature schemes* considered by Goldreich et al. [18].

2. **Delegation of duties.** Suppose Alice encrypts mail to Bob using the subject line as the IBE encryption key. Bob can decrypt mail using his master-key. Now, suppose Bob has several assistants each responsible for a different task (e.g. one is 'purchasing', another is 'human-resources', etc.). Bob gives one private key to each of his assistants corresponding to the assistant's responsibility. Each assistant can then decrypt messages whose subject line falls within its responsibilities, but it cannot decrypt messages intended for other assistants. Note that Alice only obtains a single public key from Bob (params), and she uses that public key to send mail with any subject line of her choice. The mail can only be read by the assistant responsible for that subject.

More generally, IBE can simplify various systems that manage a large number of public keys. Rather than storing a big database of public keys the system can either derive these public keys from usernames, or simply use the integers $1, \dots, n$ as distinct public keys.

2 Definitions

Identity-Based Encryption. An identity-based encryption scheme is specified by four randomized algorithms: Setup, Extract, Encrypt, Decrypt:

Setup: takes a security parameter k and returns params (system parameters) and master-key. The system parameters include a description of a finite message space $\mathcal{M}$, and a description of a finite ciphertext space $\mathcal{C}$. Intuitively, the system parameters will be publicly known, while the master-key will be known only to the "Private Key Generator" (PKG).

Extract: takes as input params, $\mathsf{master\text{-}key}$, and an arbitrary $\mathsf{ID} \in \{0,1\}^*$, and returns a private key d. Here ID is an arbitrary string that will be used as a public key, and d is the corresponding private decryption key. The $\mathsf{Extract}$ algorithm extracts a private key from the given public key.

Encrypt: takes as input params, ID, and $M \in \mathcal{M}$. It returns a ciphertext $C \in \mathcal{C}$.

Decrypt: takes as input params, $C \in \mathcal{C}$, and a private key d. It return $M \in \mathcal{M}$.

These algorithms must satisfy the standard consistency constraint, namely when d is the private key generated by algorithm $\mathsf{Extract}$ when it is given ID as the public key, then

$$\forall M \in \mathcal{M} : \mathsf{Decrypt}(\mathsf{params}, C, d) = M \quad \text{where} \quad C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, M)$$

Chosen ciphertext security. Chosen ciphertext security ($\mathsf{IND\text{-}CCA}$) is the standard acceptable notion of security for a public key encryption scheme [29, 1, 10]. Hence, it is natural to require that an identity-based encryption scheme also satisfy this strong notion of security. However, the definition of chosen ciphertext security must be strengthened a bit. The reason is that when an adversary attacks a public key ID in an identity-based system, the adversary might already possess the private keys of users $\mathsf{ID}_1, \ldots, \mathsf{ID}_n$ of her choice. The system should remain secure under such an attack. Hence, the definition of chosen ciphertext security must allow the adversary to obtain the private key associated with any identity ID_i of her choice (other than the public key ID being attacked). We refer to such queries as private key extraction queries. Another difference is that the adversary is challenged on a public key ID of her choice (as opposed to a random public key).

We say that an identity-based encryption scheme is semantically secure against an adaptive chosen ciphertext attack ($\mathsf{IND\text{-}ID\text{-}CCA}$) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following game:

Setup: The challenger takes a security parameter k and runs the Setup algorithm. It gives the adversary the resulting system parameters params. It keeps the $\mathsf{master\text{-}key}$ to itself.

Phase 1: The adversary issues queries $q_1, \ldots, q_m$ where query q_i is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$. The challenger responds by running algorithm $\mathsf{Extract}$ to generate the private key d_i corresponding to the public key $\langle \mathsf{ID}_i \rangle$. It sends d_i to the adversary.
- Decryption query $\langle \mathsf{ID}_i, C_i \rangle$. The challenger responds by running algorithm $\mathsf{Extract}$ to generate the private key d_i corresponding to ID_i. It then runs algorithm $\mathsf{Decrypt}$ to decrypt the ciphertext C_i using the private key d_i. It sends the resulting plaintext to the adversary.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \ldots, q_{i-1}$.

Challenge: Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and an identity ID on which it wishes to be challenged. The only constraint is that ID did not appear in any private key extraction query in Phase 1.

The challenger picks a random bit $b \in \{0,1\}$ and sets $C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, M_b)$. It sends C as the challenge to the adversary.

Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$ where query q_i is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}$. Challenger responds as in Phase 1.
- Decryption query $\langle \mathsf{ID}_i, C_i \rangle \neq \langle \mathsf{ID}, C \rangle$. Challenger responds as in Phase 1.

These queries may be asked adaptively as in Phase 1.

Guess: Finally, the adversary outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an $\mathsf{IND\text{-}ID\text{-}CCA}$ adversary. We define adversary $\mathcal{A}$'s advantage in attacking the scheme as: $\quad \mathrm{Adv}(\mathcal{A}) = \left|\Pr[b = b'] - \frac{1}{2}\right|$.
The probability is over the random bits used by the challenger and the adversary. We say that the IBE system is semantically secure against an adaptive chosen ciphertext attack ($\mathsf{IND\text{-}ID\text{-}CCA}$) if no polynomially bounded adversary has a non-negligible advantage in attacking the scheme. As usual, "non-negligible" should be understood as larger than $1/f(k)$ for some polynomial f (recall k is the security parameter). Note that the standard definition of chosen ciphertext security ($\mathsf{IND\text{-}CCA}$) [29, 1] is the same as above except that there are no private key extraction queries and the adversary is challenged on a random public key (rather than a public key of her choice).

Private key extraction queries are related to the definition of chosen ciphertext security in the multiuser settings [5]. After all, our definition involves multiple public keys belonging to multiple users. In [5] the authors show that that multiuser $\mathsf{IND\text{-}CCA}$ is reducible to single user $\mathsf{IND\text{-}CCA}$ using a standard hybrid argument. This does not hold in the identity-based settings, $\mathsf{IND\text{-}ID\text{-}CCA}$, since the adversary gets to choose which public keys to corrupt during the attack. To emphasize the importance of private key extraction queries we note that our IBE system can be easily modified (by removing one of the hash functions) into a system which has chosen ciphertext security when private extraction queries are disallowed. However, the scheme is completely insecure when extraction queries are allowed.

One way identity-based encryption. The proof of security for our IBE system makes use of a weak notion of security called one-way encryption (OWE) [13]. OWE is defined for standard public key encryption schemes (not identity based) as follows: the adversary $\mathcal{A}$ is given a random public key K_{pub} and a ciphertext C which is the encryption of a random message M using K_{pub}. The adversary's goal is to recover the corresponding plaintext. It has advantage ϵ in attacking the system if $\Pr[\mathcal{A}(K_{pub}, C) = M] = \epsilon$. We say that the public key scheme is a one-way encryption scheme (OWE) if no polynomial time adversary has non-negligible advantage in attacking the scheme. See [13] for precise definitions.

For identity-based encryption, we strengthen the definition as follows. We say that an IBE scheme is a one-way identity-based encryption scheme ($\mathsf{ID\text{-}OWE}$) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following game:

Setup: The challenger takes a security parameter k and runs the Setup algorithm. It gives the adversary the resulting system parameters params. It keeps the $\mathsf{master\text{-}key}$ to itself.

Phase 1: The adversary issues private key extraction queries $\mathsf{ID}_1, \ldots, \mathsf{ID}_m$. The challenger responds by running algorithm $\mathsf{Extract}$ to generate the private key d_i corresponding to the public key ID_i. It sends d_i to the adversary. These queries may be asked adaptively.

Challenge: Once the adversary decides that Phase 1 is over it outputs a public key $\mathsf{ID} \neq \mathsf{ID}_1, \ldots, \mathsf{ID}_m$ on which it wishes to be challenged. The challenger picks a random $M \in \mathcal{M}$ and encrypts M using ID as the public key. It then sends the resulting ciphertext C to the adversary.

Phase 2: The adversary issues more extraction queries $\mathsf{ID}_{m+1}, \ldots, \mathsf{ID}_n$. The only constraint is that $\mathsf{ID}_i \neq \mathsf{ID}$. The challenger responds as in Phase 1.

Guess: Finally, the adversary outputs a guess $M' \in \mathcal{M}$. The adversary wins the game if $M = M'$.

We refer to such an adversary $\mathcal{A}$ as an $\mathsf{ID\text{-}OWE}$ adversary. We define adversary's $\mathcal{A}$'s advantage in attacking the scheme as: $\mathrm{Adv}(\mathcal{A}) = \Pr[M = M']$. The probability is over the random bits used by the challenger and the adversary. Note that the definition of OWE is the same as $\mathsf{ID\text{-}OWE}$ except that there are no private key extraction queries and the adversary is challenged on a random public key (rather than a public key of her choice).

3 Bilinear maps and the Bilinear Diffie-Hellman Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of order q for some large prime q. Our IBE system makes use of a *bilinear* map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between these two groups. The map must satisfy the following properties:

1. Bilinear: We say that a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is *bilinear* if $\hat{e}(aP, bQ) = \hat{e}(P,Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that since $\mathbb{G}_1, \mathbb{G}_2$ are groups of prime order this implies that if P is a generator of $\mathbb{G}_1$ then $\hat{e}(P,P)$ is a generator of $\mathbb{G}_2$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(P,Q)$ for any $P, Q \in \mathbb{G}_1$.

In Section 5 we give a concrete example of groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map between them. The group $\mathbb{G}_1$ is the additive group of points of an elliptic curve $E/\mathbb{F}_p$. The group $\mathbb{G}_2$ is the multiplicative group of a finite field $\mathbb{F}^*_{p^2}$. Therefore, throughout the paper we view $\mathbb{G}_1$ as an additive group and $\mathbb{G}_2$ as a multiplicative group. As we will see in Section 5.1, the Weil pairing gives rise to an efficiently computable non-degenerate bilinear map.

The existence of the bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ as above has two direct implications to these groups.

The MOV reduction: Menezes, Okamoto, and Vanstone [25] show that the discrete log problem in $\mathbb{G}_1$ is no harder than the discrete log problem in $\mathbb{G}_2$. To see this, let $P, Q \in \mathbb{G}_1$ be an instance of the discrete log problem in $\mathbb{G}_1$ where both P, Q have order q. We wish to find an $\alpha \in \mathbb{Z}_q$ such that $Q = \alpha P$. Let $g = \hat{e}(P,P)$ and $h = \hat{e}(Q,P)$. Then, by bilinearity of $\hat{e}$ we know that $h = g^\alpha$. By non-degeneracy of $\hat{e}$ both g, h have order q in $\mathbb{G}_2$. Hence, we reduced the discrete log problem in $\mathbb{G}_1$ to a discrete log problem in $\mathbb{G}_2$. It follows that for discrete log to be hard in $\mathbb{G}_1$ we must choose our security parameter so that discrete log is hard in $\mathbb{G}_2$ (see Section 5).

Decision Diffie-Hellman is Easy: The Decision Diffie-Hellman problem (DDH) [2] in $\mathbb{G}_1$ is to distinguish between the distributions $\langle P, aP, bP, abP\rangle$ and $\langle P, aP, bP, cP\rangle$ where a, b, c are random in $\mathbb{Z}_q$ and P is random in $\mathbb{G}_1$. Joux and Nguyen [21] point out that DDH in $\mathbb{G}_1$ is easy. To see this, observe that given $P, aP, bP, cP \in \mathbb{G}_1^*$ we have

$$c = ab \bmod q \quad \iff \quad \hat{e}(P, cP) = \hat{e}(aP, bP).$$

The Computational Diffie-Hellman problem (CDH) in $\mathbb{G}_1$ can still be hard (CDH in G_1 is to find abP given random $\langle P, aP, bP\rangle$). Joux and Nguyen [21] give examples of mappings $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where CDH in $\mathbb{G}_1$ is believed to be hard even though DDH in $\mathbb{G}_1$ is easy.

Notation: From here on we use $\mathbb{G}_1^*$ to denote the set $\mathbb{G}_1^* = \mathbb{G}_1 \setminus \{O\}$ where O is the identity element in the group $\mathbb{G}_1$. We use $\mathbb{Z}_q$ to denote the group $\{0, \ldots, q-1\}$ under addition modulo q.

3.1 The Bilinear Diffie-Hellman Assumption (BDH)

Since the Decision Diffie-Hellman problem (DDH) in $\mathbb{G}_1$ is easy we cannot use DDH to build cryptosystems in the group $\mathbb{G}_1$. Instead, the security of our IBE system is based on a variant of the Computational Diffie-Hellman assumption called the Bilinear Diffie-Hellman Assumption (BDH).[1]

Bilinear Diffie-Hellman Problem: Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups of prime order q. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map and let P be a generator of $\mathbb{G}_1$. The BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ for some $a, b, c \in \mathbb{Z}_q^*$ compute $W = \hat{e}(P, P)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ has advantage ϵ in solving BDH in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ if

$$\Pr\left[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc}\right] \geq \epsilon$$

where the probability is over the random choice of $\langle a, b, c \rangle$ in $\mathbb{Z}_q^*$, the random choice of $P \in \mathbb{G}_1^*$, and the random bits of $\mathcal{A}$.

BDH Parameter Generator: We say that a randomized algorithm $\mathcal{IG}$ is a *BDH parameter generator* if (1) $\mathcal{IG}$ takes a security parameter $0 < k \in \mathbb{Z}$, (2) $\mathcal{IG}$ runs in polynomial time in k, and (3) $\mathcal{IG}$ outputs the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ and the description of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. We require that the groups have the same prime order $q = |G_1| = |G_2|$. We denote the output of $\mathcal{IG}$ by $\mathcal{IG}(1^k)$. We give an example of a BDH parameter generator in Section 5.1.

Bilinear Diffie-Hellman Assumption: Let $\mathcal{IG}$ be a BDH parameter generator. We say that an algorithm $\mathcal{A}$ has advantage $\epsilon(k)$ in solving the BDH problem for $\mathcal{IG}$ if for sufficiently large k:

$$\mathrm{Adv}_{\mathcal{IG}}(\mathcal{A}) = \Pr\left[\mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P, P)^{abc} \;\middle|\; \begin{array}{c} \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle \leftarrow \mathcal{IG}(1^k), \\ P \leftarrow \mathbb{G}_1^*,\ a, b, c \leftarrow \mathbb{Z}_q^* \end{array}\right] > \epsilon(k)$$

We say that $\mathcal{IG}$ satisfies the BDH assumption if any randomized polynomial time (in k) algorithm $\mathcal{A}$ solves the BDH problem with advantage at most $1/f(k)$ for any polynomial $f \in \mathbb{Z}[x]$. In other words, given a random $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ generated by $\mathcal{IG}$, no efficient algorithm can solve BDH in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with non-negligible advantage. Occasionally we say that BDH is hard in groups generated by $\mathcal{IG}$.

In Section 5.1 we give some examples of BDH parameter generators that are believed to satisfy the BDH assumption. We note that Joux [20] (implicitly) used the BDH assumption to construct a one-round three party Diffie-Hellman protocol. The BDH assumption is also needed for constructions in [35, 31].

It is interesting to study the relationship of the BDH problem to other hard problems used in cryptography. Currently, all we can say is that the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is no harder than the CDH problem in $\mathbb{G}_1$. In other words, an algorithm for CDH in $\mathbb{G}_1$ is sufficient for solving BDH in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. The converse is currently an open problem: is an algorithm for BDH sufficient for solving CDH in $\mathbb{G}_1$?

We note that in all our examples (in Section 5.1) the isomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$ induced by the bilinear map are believed to be one-way functions. More specifically, for a point $Q \in \mathbb{G}_1^*$ define the

[1]This was called the Weil Diffie-Hellman assumption (WDH) in an earlier version of the paper [3]. However, since our system can be implemented using any of a number of bilinear maps (the Weil pairing, the Tate pairing, etc.) we opt for the more general name here.

isomorphism $f_Q : \mathbb{G}_1 \to \mathbb{G}_2$ by $f_Q(P) = \hat{e}(P, Q)$. Observe that an efficient algorithm for inverting f_Q would imply an a efficient algorithm for deciding DDH in the group $\mathbb{G}_2$. In all our examples DDH is believed to be hard in the group $\mathbb{G}_2$. Hence, all the isomorphisms $f_Q : \mathbb{G}_1 \to \mathbb{G}_2$ induced by the bilinear map are believed to be one-way functions.

4 Our Identity-Based Encryption Scheme

We describe our scheme in stages. First we give a basic identity-based encryption scheme which is not secure against an adaptive chosen ciphertext attack. The only reason for describing the basic scheme is to make the presentation easier to follow. Our full scheme, described in Section 4.2, extends the basic scheme to get security against an adaptive chosen ciphertext attack (IND-ID-CCA) in the random oracle model. In Section 4.3 we relax some of the requirements on the hash functions.

The presentation in this section uses an arbitrary BDH parameter generator $\mathcal{IG}$ satisfying the BDH assumption. In Section 5 we describe a concrete IBE system using the Weil pairing.

4.1 BasicIdent

To explain the basic ideas underlying our IBE system we describe the following simple scheme, called BasicIdent. We present the scheme by describing the four algorithms: Setup, Extract, Encrypt, Decrypt. We let k be the security parameter given to the setup algorithm. We let $\mathcal{IG}$ be some BDH parameter generator.

Setup: The algorithm works as follows:

Step 1: Run $\mathcal{IG}$ on input k to generate two prime order groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let q be the order of $\mathbb{G}_1, \mathbb{G}_2$. Choose an arbitrary generator $P \in \mathbb{G}_1$.

Step 2: Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.

Step 3: Choose a cryptographic hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$. Choose a cryptographic hash function $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some n. The security analysis will view H_1, H_2 as random oracles.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n$. The system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2 \rangle$. The master-key is $s \in \mathbb{Z}_q^*$.

Extract: For a given string $\mathsf{ID} \in \{0,1\}^*$ the algorithm does: (1) computes $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$, and (2) sets the private key d_{ID} to be $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ where s is the master key.

Encrypt: To encrypt $M \in \mathcal{M}$ under the public key ID do the following: (1) compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$, (2) choose a random $r \in \mathbb{Z}_q^*$, and (3) set the ciphertext to be

$$C = \langle rP, \;\; M \oplus H_2(g_{\mathsf{ID}}^r) \rangle \quad \text{where} \quad g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2^*$$

Decrypt: Let $C = \langle U, V \rangle \in \mathcal{C}$ be a ciphertext encrypted using the public key ID. To decrypt C using the private key $d_{\mathsf{ID}} \in \mathbb{G}_1^*$ compute:

$$V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = M$$

This completes the description of BasicIdent. We first verify consistency. When everything is computed as above we have:

1. During encryption M is bitwise exclusive-ored with the hash of: g_{ID}^r.
2. During decryption V is bitwise exclusive-ored with the hash of: $\hat{e}(d_{\mathsf{ID}}, U)$.

These masks used during encryption and decryption are the same since:

$$\hat{e}(d_{\mathsf{ID}}, U) = \hat{e}(sQ_{\mathsf{ID}}, rP) = \hat{e}(Q_{\mathsf{ID}}, P)^{sr} = \hat{e}(Q_{\mathsf{ID}}, P_{pub})^r = g_{\mathsf{ID}}^r$$

Thus, applying decryption after encryption produces the original message M as required. We note that there is no need to devise attacks against this basic scheme since it is only presented for simplifying the exposition. The next section describes the full scheme. Performance considerations of BasicIdent are discussed in Section 5.

Security. Next, we study the security of this basic scheme. The following theorem shows that BasicIdent is a one-way identity based encryption scheme (ID-OWE) assuming BDH is hard in groups generated by $\mathcal{IG}$.

Theorem 4.1 *Let the hash functions H_1, H_2 be random oracles. Suppose there is an* ID-OWE *adversary $\mathcal{A}$ that has advantage ϵ against the scheme* BasicIdent. *Suppose $\mathcal{A}$ make at most $q_E > 0$ private key extraction queries and $q_{H_2} > 0$ hash queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves BDH in groups generated by $\mathcal{IG}$ with advantage at least:*

$$Adv_{\mathcal{IG}}(\mathcal{B}) \geq \frac{\epsilon}{e(1+q_E) \cdot q_{H_2}} - \frac{1}{q_{H_2} \cdot 2^n}$$

Here $e \approx 2.71$ is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.

Note that the values ϵ, q_E, q_{H_2} in Theorem 4.1 are actually functions of the security parameter k, i.e. $\epsilon = \epsilon(k)$, $q_E = q_E(k)$, etc. However, to simplify the notation we drop the parameter k throughout the paper.

To prove the theorem we need to define a related Public Key Encryption scheme (not an identity scheme), called BasicPub. BasicPub is described by three algorithms: keygen, encrypt, decrypt.

keygen: The algorithm works as follows:

Step 1: Run $\mathcal{IG}$ on input k to generate two prime order groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let q be the order of $\mathbb{G}_1, \mathbb{G}_2$. Choose an arbitrary generator $P \in \mathbb{G}_1$.

Step 2: Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.
Pick a random point $Q_{\mathsf{ID}} \in \mathbb{G}_1^*$. Then Q_{ID} is in the group generated by P.

Step 3: Choose a cryptographic hash function $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some n.

Step 4: The public key is $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$. The private key is $d_{\mathsf{ID}} = sQ_{\mathsf{ID}} \in \mathbb{G}_1^*$.

encrypt: To encrypt $M \in \{0,1\}^n$ choose a random $r \in \mathbb{Z}_q^*$ and set the ciphertext to be:

$$C = \langle rP, \;\; M \oplus H_2(g^r) \rangle \quad \text{where} \quad g = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2^*$$

decrypt: Let $C = \langle U, V \rangle \in \mathcal{C}$ be a ciphertext encrypted using the public key $\langle p, n, P, P_{pub}, Q_{\mathsf{ID}}, H \rangle$. To decrypt C using the private key $d_{\mathsf{ID}} \in \mathbb{G}_1^*$ compute:

$$V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = M$$

This completes the description of BasicPub. We now prove Theorem 4.1 in two steps. We first show that an ID-OWE attack on BasicIdent can be converted to a OWE attack on BasicPub. This step shows that private key extraction queries do not help the adversary. We then show that BasicPub is OWE if the BDH assumption holds. The proofs of these two lemmas are given in Appendix A.

Lemma 4.2 *Let H_1 be a random oracle from $\{0,1\}^*$ to $\mathbb{G}_1^*$. Let $\mathcal{A}$ be an* ID-OWE *adversary that has advantage ϵ against* BasicIdent. *Suppose $\mathcal{A}$ makes at most $q_E > 0$ private key extraction queries. Then there is a* OWE *adversary $\mathcal{B}$ that has advantage at least $\epsilon/e(1+q_E)$ against* BasicPub. *Its running time is $O(time(\mathcal{A}))$.*

Lemma 4.3 *Let H_2 be a random oracle from $\mathbb{G}_2$ to $\{0,1\}^n$. Let $\mathcal{A}$ be a* OWE *adversary that has advantage ϵ against* BasicPub. *Suppose $\mathcal{A}$ makes a total of $q_{H_2} > 0$ queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem for $\mathcal{IG}$ with advantage at least $(\epsilon - \frac{1}{2^n})/q_{H_2}$ and a running time $O(time(\mathcal{A}))$.*

Proof of Theorem 4.1. The theorem follows directly from Lemma 4.2 and Lemma 4.3. Composing both reductions shows that an ID-OWE adversary on BasicIdent with advantage ϵ gives a BDH algorithm for $\mathcal{IG}$ with advantage at least $(\epsilon/e(1+q_E) - 1/2^n)/q_{H_2}$, as required. $\square$

4.2 Identity-Based Encryption with Chosen Ciphertext Security

We use a technique due to Fujisaki-Okamoto [13] to convert the BasicIdent scheme of the previous section into a chosen ciphertext secure IBE system (in the sense of Section 2) in the random oracle model. Let $\mathcal{E}$ be a probabilistic public key encryption scheme. We denote by $\mathcal{E}_{pk}(M; r)$ the encryption of M using the random bits r under the public key pk. Fujisaki-Okamoto define the hybrid scheme $\mathcal{E}^{hy}$ as:

$$\mathcal{E}^{hy}_{pk}(M) \quad = \quad \langle\ \mathcal{E}_{pk}(\sigma; H_3(\sigma, M)),\ \ H_4(\sigma) \oplus M\ \rangle$$

Here σ is generated at random and H_3, H_4 are cryptographic hash functions. Fujisaki-Okamoto show that if $\mathcal{E}$ is a one-way encryption scheme then $\mathcal{E}^{hy}$ is a chosen ciphertext secure system (IND-CCA) in the random oracle model (assuming $\mathcal{E}_{pk}$ satisfies some natural constraints).

We apply this transformation to BasicIdent and show that the resulting IBE system is IND-ID-CCA. We obtain the following IBE scheme which we call FullIdent. Recall that n is the length of the message to be encrypted.

Setup: As in the BasicIdent scheme. In addition, we pick a hash function $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$, and a hash function $H_4 : \{0,1\}^n \to \{0,1\}^n$.

Extract: As in the BasicIdent scheme.

Encrypt: To encrypt $M \in \{0,1\}^n$ under the public key ID do the following: (1) compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$, (2) choose a random $\sigma \in \{0,1\}^n$, (3) set $r = H_3(\sigma, M)$, and (4) set the ciphertext to be

$$C = \langle rP,\ \ \sigma \oplus H_2(g_{\mathsf{ID}}^r),\ \ M \oplus H_4(\sigma)\rangle \quad \text{where} \quad g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2$$

Decrypt: Let $C = \langle U, V, W\rangle$ be a ciphertext encrypted using the public key ID. If $U \notin \mathbb{G}_1^*$ reject the ciphertext. To decrypt C using the private key $d_{\mathsf{ID}} \in \mathbb{G}_1^*$ do:

1. Compute $\quad V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = \sigma$.
2. Compute $\quad W \oplus H_4(\sigma) = M$.
3. Set $r = H_3(\sigma, M)$. Test that $U = rP$. If not, reject the ciphertext.
4. Output M as the decryption of C.

This completes the description of FullIdent. Note that M is encrypted as $W = M \oplus H_4(\sigma)$. This can be replaced by $W = E_{H_4(\sigma)}(M)$ where E is a semantically secure symmetric encryption scheme (see [13]).

Security. The following theorem shows that FullIdent is a chosen ciphertext secure IBE (i.e. IND-ID-CCA), assuming BDH is hard in groups generated by $\mathcal{IG}$.

Theorem 4.4 *Let the hash functions H_1, H_2, H_3, H_4 be random oracles. Let $\mathcal{A}$ be a t-time* IND-ID-CCA *adversary on* FullIdent *that achieves advantage ϵ. Suppose $\mathcal{A}$ makes at most q_E extraction queries, at most q_D decryption queries, and at most $q_{H_2}, q_{H_3}, q_{H_4}$ queries to the hash functions H_2, H_3, H_4 respectively. Then there is a BDH algorithm $\mathcal{B}$ for $\mathcal{IG}$ where:*

$$\begin{aligned} time(\mathcal{B}) &\leq FO_{time}(t, q_{H_4}, q_{H_3}) \\ Adv_{\mathcal{IG}}(\mathcal{B}) &\geq \left(FO_{adv}(\tfrac{\epsilon}{e(1+q_E+q_D)}, q_{H_4}, q_{H_3}, q_D) - 1/2^n \right) / q_{H_2} \end{aligned}$$

where the functions FO_{time} and FO_{adv} are defined in Theorem 4.5.

The proof of the theorem is based on the theorem below due to Fujisaki and Okamoto (Theorem 14 in [13]). We state their theorem as it applies to the public key encryption scheme BasicPub of the previous section. Let $\mathsf{BasicPub}^{hy}$ be the result of applying the Fujisaki-Okamoto transformation to BasicPub.

Theorem 4.5 (FO) *Suppose there is a $(t, q_D, q_{H_3}, q_{H_4})$* IND-CCA *adversary that achieves advantage ϵ when attacking* $\mathsf{BasicPub}^{hy}$. *Then there is a (t_1, ϵ_1)* OWE *adversary on* BasicPub *where*

$$\begin{aligned} t_1 &= FO_{time}(t, q_{H_4}, q_{H_3}) = t + O((q_{H_4} + q_{H_3}) \cdot n), \quad \textit{and} \\ \epsilon_1 &= FO_{adv}(\epsilon, q_{H_4}, q_{H_3}, q_D) = \frac{1}{2(q_{H_4} + q_{H_3})} \left[(\epsilon + 1)(1 - 2/q)^{q_D} - 1 \right] \end{aligned}$$

We also need the following lemma to translate between an IND-ID-CCA chosen ciphertext attack on FullIdent and an IND-CCA chosen ciphertext attack on $\mathsf{BasicPub}^{hy}$. The proof is given in Appendix A.

Lemma 4.6 *Let $\mathcal{A}$ be an* IND-ID-CCA *adversary that has advantage ϵ against the IBE scheme* FullIdent. *Suppose $\mathcal{A}$ makes at most $q_E > 0$ private key extraction queries and at most q_D decryption queries. Then there is an* IND-CCA *adversary $\mathcal{B}$ that has advantage at least $\frac{\epsilon}{e(1+q_E+q_D)}$ against* $\mathsf{BasicPub}^{hy}$. *Its running time is $O(time(\mathcal{A}))$.*

Proof of Theorem 4.4. By Lemma 4.6 an IND-ID-CCA adversary on FullIdent implies an IND-CCA adversary on $\mathsf{BasicPub}^{hy}$. By Theorem 4.5 an IND-CCA adversary on $\mathsf{BasicPub}^{hy}$ implies a OWE adversary on BasicPub. By Lemma 4.3 a OWE adversary on BasicPub implies an algorithm for BDH. Composing all these reductions gives the required bounds. $\square$

4.3 Relaxing the hashing requirements

Recall that the IBE system of Section 4.2 uses a hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$. The concrete IBE system presented in the next section uses $\mathbb{G}_1$ as a subgroup of the group of points on an elliptic curve. In practice, it is difficult to build hash functions that hash directly onto such groups. We therefore show how to relax the requirement of hashing directly onto $\mathbb{G}_1^*$. Rather than hash onto $\mathbb{G}_1^*$ we hash onto some set $A \subseteq \{0,1\}^*$ and then use a deterministic encoding function to map A onto $\mathbb{G}_1^*$.

Admissible encodings: Let $\mathbb{G}_1$ be a group and let $A \in \{0,1\}^*$ be a finite set. We say that an encoding function $L : A \to \mathbb{G}_1^*$ is *admissible* if it satisfies the following properties:

1. Computable: There is an efficient deterministic algorithm to compute $L(x)$ for any $x \in A$.

2. ℓ-to-1: For any $y \in \mathbb{G}_1^*$ the preimage of y under L has size exactly ℓ. In other words, $|L^{-1}(y)| = \ell$ for all $y \in \mathbb{G}_1^*$. Note that this implies that $|A| = \ell \cdot |\mathbb{G}_1^*|$.
3. Samplable: There is an efficient randomized algorithm $\mathcal{L}_S$ such that $\mathcal{L}_S(y)$ induces a uniform distribution on $L^{-1}(y)$ for any $y \in \mathbb{G}_1^*$. In other words, $\mathcal{L}_S(y)$ is a random element in $L^{-1}(y)$.

We slightly modify FullIdent to obtain an IND-ID-CCA secure IBE system where H_1 is replaced by a hash function into some set A. Since the change is so minor we refer to this new scheme as FullIdent':

Setup: As in the FullIdent scheme. The only difference is that H_1 is replaced by a hash function $H_1' : \{0,1\}^* \to A$. The system parameters also include a description of an admissible encoding function $L : A \to \mathbb{G}_1^*$.

Extract, Encrypt: As in the FullIdent scheme. The only difference is that in Step 1 these algorithms compute $Q_{\mathsf{ID}} = L(H_1'(\mathsf{ID})) \in \mathbb{G}_1^*$.

Decrypt: As in the FullIdent scheme.

This completes the description of FullIdent'. The following theorem shows that FullIdent' is a chosen ciphertext secure IBE (i.e. IND-ID-CCA), assuming FullIdent is.

Theorem 4.7 *Let $\mathcal{A}$ be an* IND-ID-CCA *adversary on* FullIdent' *that achieves advantage ϵ. Suppose $\mathcal{A}$ makes at most q_{H_1} queries to the hash function H_1'. Then there is an* IND-ID-CCA *adversary $\mathcal{B}$ on* FullIdent *that achieves the same advantage ϵ and $time(\mathcal{B}) = time(\mathcal{A}) + q_{H_1} \cdot time(L_S)$*

Proof Sketch Algorithm $\mathcal{B}$ attacks FullIdent by running algorithm $\mathcal{A}$. It relays all decryption queries, extraction queries, and hash queries directly to the challenger and relays the challenger's response back to $\mathcal{A}$. It only behaves differently when $\mathcal{A}$ issues a hash query to H_1'. Recall that $\mathcal{B}$ only has access to a hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$. To respond to H_1' queries algorithm $\mathcal{B}$ maintains a list of tuples $\langle \mathsf{ID}_j, y_j \rangle$ as explained below. We refer to this list as the H_1^{list}. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1' at a point ID_i algorithm $\mathcal{B}$ responds as follows:

1. If the query ID_i already appears on the H_1^{list} in a tuple $\langle \mathsf{ID}_i, y_i \rangle$ then respond with $H_1'(\mathsf{ID}_i) = y_i \in A$.
2. Otherwise, $\mathcal{B}$ issues a query for $H_1(\mathsf{ID}_i)$. Say, $H_1(\mathsf{ID}_i) = \alpha \in \mathbb{G}_1^*$.
3. $\mathcal{B}$ runs the sampling algorithm $L_S(\alpha)$ to generate a random element $y \in L^{-1}(\alpha)$.
4. $\mathcal{B}$ adds the tuple $\langle \mathsf{ID}_i, y \rangle$ to the H_1^{list} and responds to $\mathcal{A}$ with $H_1'(\mathsf{ID}_i) = y \in A$. Note that y is uniformly distribute in A as required since α is uniformly distributed in $\mathbb{G}_1^*$ and L is an ℓ-to-1 map.

Algorithm $\mathcal{B}$'s responses to all of $\mathcal{A}$'s queries, including H_1' queries, are identical to $\mathcal{A}$'s view in the real attack. Hence, $\mathcal{B}$ will have the same advantage ϵ in winning the game with the challenger. $\square$

5 A concrete IBE system using the Weil pairing

In this section we use FullIdent' to describe a concrete IBE system based on the Weil pairing. We first review some properties of the pairing.

5.1 Properties of the Weil Pairing

Let p be a prime satisfying $p = 2 \bmod 3$ and $p = 6q - 1$ for some prime q. Let E be the elliptic curve defined by the equation $y^2 = x^3 + 1$ over $\mathbb{F}_p$. We state a few elementary facts about this curve:

Fact 1: Since $x^3 + 1$ is a permutation on $\mathbb{F}_p$ it easily follows that $E/\mathbb{F}_p$ contains $p+1$ points. We let O denote the point at infinity. Let $P \in E/\mathbb{F}_p$ be a generator of the group of points of order $q = (p+1)/6$. We denote this group by $\mathbb{G}_1$.

Fact 2: For any $y_0 \in \mathbb{F}_p$ there is a unique point (x_0, y_0) on $E/\mathbb{F}_p$, namely $x_0 = (y_0^2 - 1)^{1/3} \in \mathbb{F}_p$. Hence, if (x,y) is a random non-zero point on $E/\mathbb{F}_p$ then y is uniform in $\mathbb{F}_p$. We use this property to simplify the proof of security.

Fact 3: Let $1 \neq \zeta \in \mathbb{F}_{p^2}$ be a solution of $x^3 - 1 = 0 \bmod p$. Then the map $\phi(x,y) = (\zeta x, y)$ is an automorphism of the group of points on the curve E. Note that when $P = (x,y) \in E/\mathbb{F}_p$ we have that $\phi(P) \in E/\mathbb{F}_{p^2}$, but $\phi(P) \notin E/\mathbb{F}_p$. Hence, $P \in E/\mathbb{F}_p$ is linearly independent of $\phi(P) \in E/\mathbb{F}_{p^2}$.

Fact 4: Since the points P and $\phi(P)$ are linearly independent they generate a group isomorphic to $\mathbb{Z}_q \times \mathbb{Z}_q$. We denote this group of points by $E[q]$.

Let $\mathbb{G}_2$ be the subgroup of $\mathbb{F}_{p^2}^*$ containing all elements of order $q = (p+1)/6$. The Weil pairing on the curve $E/\mathbb{F}_{p^2}$ is a mapping $e : E[q] \times E[q] \to \mathbb{G}_2$ defined in Appendix B. We define the modified Weil pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ to be:

$$\hat{e}(P,Q) = e(P, \phi(Q))$$

The modified Weil pairing satisfies the following properties:

1. Bilinear: For all $P, Q \in \mathbb{G}_1$ and for all $a, b \in \mathbb{Z}$ we have $\hat{e}(aP, bQ) = \hat{e}(P,Q)^{ab}$.
2. Non-degenerate: If P is a generator of $\mathbb{G}_1$ then $\hat{e}(P,P) \in \mathbb{F}_{p^2}^*$ is a generator of $\mathbb{G}_2$.
3. Computable: Given $P, Q \in \mathbb{G}_1$ there is an efficient algorithm, due to Miller, to compute $\hat{e}(P,Q) \in \mathbb{G}_2$. This algorithm is described in Appendix C. Its running time is comparable to exponentiation in $\mathbb{F}_p$.

Joux and Nguyen [21] point out that although the Computational Diffie-Hellman problem (CDH) appears to be hard in the group $\mathbb{G}_1$, the Decisional Diffie-Hellman problem (DDH) is easy in $\mathbb{G}_1$ (as discussed in Section 3).

BDH Parameter Generator $\mathcal{IG}_1$: Given a security parameter $0 < k \in \mathbb{Z}$ the BDH parameter generator picks a random k-bit prime p such that $p = 2 \bmod 3$ and $p = 6q - 1$ for some prime q. The group $\mathbb{G}_1$ is the subgroup of order q of the group of points on the curve $y^2 = x^3 + 1$ over $\mathbb{F}_p$. The group $\mathbb{G}_2$ is the subgroup of order q of $\mathbb{F}_{p^2}^*$. The bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is the modified Weil pairing defined above.

As pointed out in Section 3 the discrete log problem in $\mathbb{G}_1$ is efficiently reducible to discrete log in $\mathbb{G}_2$ (see [25, 14]). Hence, computing discrete log in $\mathbb{F}_{p^2}^*$ is sufficient for computing discrete log in $\mathbb{G}_1$. For proper security of discrete log in $\mathbb{F}_{p^2}^*$ one often uses primes p that are at least 1024-bits long. Since we need discrete log in $\mathbb{G}_1$ to be difficult our system also uses primes p that are at least 1024-bits long. In fact, a 512-bit prime p may be sufficient since the MOV reduction then leads to a discrete log problem in a finite field $\mathbb{F}_{p^2}$ of size approximately 2^{1024}.

5.2 An admissible encoding function: MapToPoint

Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups generated by $\mathcal{IG}_1$ as defined above. Recall that the IBE system of Section 4.2 uses a hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$. By Theorem 4.7, it suffices to have a hash function $H_1 : \{0,1\}^* \to A$ for some set A, and an admissible encoding function $L : A \to \mathbb{G}_1^*$. In what follows the set A will be $\mathbb{F}_p$, and the admissible encoding function L will be called MapToPoint.

Let p be a prime satisfying $p = 2 \bmod 3$ and $p = 6q - 1$ for some prime $q > 3$. Let E be the elliptic curve $y^2 = x^3 + 1$ over $\mathbb{F}_p$. Let $\mathbb{G}_1$ be the subgroup of points on E of order q. Suppose we already have a hash function $H_1 : \{0,1\}^* \to \mathbb{F}_p$.

Algorithm $\mathsf{MapToPoint}$ works as follows on input $y_0 \in \mathbb{F}_p$:

1. Compute $x_0 = (y_0^2 - 1)^{1/3} = (y_0^2 - 1)^{(2p-1)/3} \in \mathbb{F}_p$.
2. Let $Q = (x_0, y_0) \in E/\mathbb{F}_p$ and set $Q_{\mathsf{ID}} = 6Q \in \mathbb{G}_1$.
3. Output $\mathsf{MapToPoint}(y_0) = Q_{\mathsf{ID}}$.

This completes the description of $\mathsf{MapToPoint}$.

We note that there are 5 values of $y_0 \in \mathbb{F}_p$ for which $6Q = (x_0, y_0) = O$ (these are the non-O points of order dividing 6). When $H_1(\mathsf{ID})$ is one of these 5 values Q_{ID} is the identity element of $\mathbb{G}_1$. Since it is extremely unlikely for $H_1(\mathsf{ID})$ to hit one of these five points, for simplicity we say that such ID's are invalid. Algorithm $\mathsf{MapToPoint}$ can be easily extended to handle these five y_0 values as well by hashing ID multiple times using different hash functions.

Claim 5.1 $\mathsf{MapToPoint} : \mathbb{F}_p \to \mathbb{G}_1^*$ *is an admissible encoding function.*

Proof The map is clearly computable and is a $6 - to - 1$ mapping. It remains to show that L is samplable. Let P be a generator of $E/\mathbb{F}_p$. Given a $Q \in \mathbb{G}_1^*$ the sampling algorithm $\mathcal{L}_S$ does the following: (1) pick a random $b \in \{0, \ldots, 5\}$, (2) compute $Q' = 6^{-1} \cdot Q + bqP = (x, y)$, and (3) output $\mathcal{L}_S(Q) = y \in \mathbb{F}_p$. Here 6^{-1} is the inverse of 6 in $\mathbb{Z}_q^*$. This algorithm outputs a random element from the 6 elements in $L^{-1}(Q)$ as required. □

5.3 A concrete IBE system

Using the BDH parameter generator $\mathcal{IG}_1$ and the admissible encoding function $\mathsf{MapToPoint}$ we obtain the following concrete IBE system.

Setup: The algorithm works as follows:

Step 1: Choose a large k-bit prime p such that $p = 2 \bmod 3$ and $p = 6q - 1$ for some prime $q > 3$. Let E be the elliptic curve defined by $y^2 = x^3 + 1$ over $\mathbb{F}_p$. Choose an arbitrary $P \in E/\mathbb{F}_p$ of order q.

Step 2: Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.

Step 3: Pick four hash functions: $H_1 : \{0,1\}^* \to \mathbb{F}_p$; $H_2 : \mathbb{F}_{p^2} \to \{0,1\}^n$ for some n ; $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$, and a hash function $H_4 : \{0,1\}^n \to \{0,1\}^n$.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = E/\mathbb{F}_p \times \{0,1\}^n$. The system parameters are $\mathsf{params} = \langle p, n, P, P_{pub}, H_1, \ldots, H_4 \rangle$. The $\mathsf{master\text{-}key}$ is $s \in \mathbb{Z}_q^*$.

Extract: For a given string $\mathsf{ID} \in \{0,1\}^*$ the algorithm builds a private key d as follows:

Step 1: Compute $\mathsf{MapToPoint}(H_1(\mathsf{ID})) = Q_{\mathsf{ID}} \in E/\mathbb{F}_p$ of order q.

Step 2: Set the private key d_{ID} to be $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ where s is the master key.

Encrypt: To encrypt $M \in \{0,1\}^n$ under the public key ID do the following:

Step 1: Compute $\mathsf{MapToPoint}(H_1(\mathsf{ID})) = Q_{\mathsf{ID}} \in E/\mathbb{F}_p$ of order q.

Step 2: Choose a random $\sigma \in \{0,1\}^n$.

Step 3: Set $r = H_3(\sigma, M)$.

Step 4: Set the ciphertext to be

$$C = \langle rP, \;\; \sigma \oplus H_2(g_{\mathsf{ID}}^r), \;\; M \oplus H_4(\sigma) \rangle \quad \text{where} \quad g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{F}_{p^2}$$

Decrypt: Let $C = \langle U, V, W\rangle \in \mathcal{C}$ be a ciphertext encrypted using the public key ID. If $U \in E/\mathbb{F}_p$ is not a point of order q reject the ciphertext. To decrypt C using the private key d_{ID} do:
Step 1. Compute $V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = \sigma$.
Step 2. Compute $W \oplus H_4(\sigma) = M$.
Step 3. Set $r = H_3(\sigma, M)$. Test that $U = rP$. If not, reject the ciphertext.
Step 4. Output M as the decryption of C.

Performance. Algorithms Setup and Extract are very simple algorithms. At the heart of both algorithms is a standard multiplication on the curve $E/\mathbb{F}_p$. Algorithm Encrypt requires that the encryptor compute the Weil pairing of Q_{ID} and P_{pub}. Note that this computation is independent of the message, and hence can be done once and for all. Once g_{ID} is computed the performance of the system is almost identical to standard ElGamal encryption. We also note that the ciphertext length of BasicIdent is the same as in regular ElGamal encryption in $\mathbb{F}_p$. Decryption is a simple Weil pairing computation.

Security. The security of the concrete IBE system follows directly from Theorem 4.4 and Theorem 4.7.

Corollary 5.2 *the concrete IBE system above is a chosen ciphertext secure IBE (i.e.* IND-ID-CCA *in the random oracle model) assuming the BDH parameter generator* $\mathcal{IG}_1$ *satisfies the BDH assumption.*

6 Extensions and Observations

Tate pairing and other curves. Our IBE system has some flexibility in terms of the curves being used and the definition of the pairing. For example, one could use the curve $y^2 = x^3 + x$ with its endomorphism $\phi : (x, y) \to (-x, iy)$ where $i^2 = -1$. As another example, Galbraith [15] suggests using special supersingular elliptic curves over a field of small characteristic to reduce the ciphertext size in our system. We also note that both encryption and decryption in FullIdent can be made faster by using the Tate pairing on elliptic curves rather than the Weil pairing. In general, one can use any efficiently computable bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between two groups $\mathbb{G}_1, \mathbb{G}_2$ as long as the BDH assumption holds. One would also need an admissible encoding function onto $\mathbb{G}_1$.

Distributed PKG. In the standard use of an IBE in an e-mail system the master-key stored at the PKG must be protected in the same way that the private key of a CA is protected. One way of protecting this key is by distributing it among different sites using techniques of threshold cryptography [16]. Our IBE system supports this in a very efficient and robust way. Recall that the master-key is some $s \in \mathbb{Z}_q^*$. in order to generate a private key the PKG computes $Q_{priv} = sQ_{\mathsf{ID}}$, where Q_{ID} is derived from the user's public key ID. This can easily be distributed in a t-out-of-n fashion by giving each of the n PKGs one share s_i of a Shamir secret sharing of $s \bmod q$. When generating a private key each of the t chosen PKGs simply responds with $Q^{(i)}_{priv} = s_i Q_{\mathsf{ID}}$. The user can then construct Q_{priv} as $Q_{priv} = \sum \lambda_i Q^{(i)}_{priv}$ where the λ_i's are the appropriate Lagrange coefficients.
Furthermore, it is easy to make this scheme robust against dishonest PKGs using the fact that DDH is easy in $\mathbb{G}_1$. During setup each of the n PKGs publishes $P^{(i)}_{pub} = s_i P$. During a key generation request the user can verify that the response from the i'th PKG is valid by testing that:

$$\hat{e}(Q^{(i)}_{priv}, P) = \hat{e}(Q_{\mathsf{ID}}, P^{(i)}_{pub})$$

Thus, a misbehaving PKG will be immediately caught. There is no need for zero-knowledge proofs as in regular robust threshold schemes. The PKG's master-key can be generated in a distributed fashion using the techniques of [17].

Note that a distributed master-key also enables threshold decryption on a *per-message* basis, without any need to derive the corresponding decryption key. For example, threshold decryption of BasicIdent ciphertext (U, V) is straightforward if each PKG responds with $\hat{e}(s_i Q_{ID}, U)$.

Working in subgroups. The performance of our IBE system (Section 5) can be improved if we work in a small subgroup of the curve. For example, choose a 1024-bit prime $p = 2 \bmod 3$ with $p = aq - 1$ for some 160-bit prime q. The point P is then chosen to be a point of order q. Each public key ID is converted to a group point by hashing ID to a point Q on the curve and then multiplying the point by a. The system is secure if the BDH assumption holds in the group generated by P. The advantage is that the Weil computation is done on points of small order, and hence is much faster.

IBE implies signatures. Moni Naor has observed that an IBE scheme can be immediately converted into a public key signature scheme. The intuition is as follows. The private key for the signature scheme is the master key for the IBE scheme. The public key for the signature scheme is the global system parameters for the IBE scheme. The signature on a message M is the IBE decryption key for $\mathsf{ID} = M$. To verify a signature, choose a random message M', encrypt M' using the public key $\mathsf{ID} = M$, and then attempt to decrypt using the given signature on M as the decryption key. If the IBE scheme is IND-ID-CCA, then the signature scheme is existentially unforgeable against a chosen message attack. Note that, unlike most signature schemes, the signature verification algorithm here is randomized. This shows that secure IBE schemes incorporate both public key encryption and digital signatures. We note that the signature scheme derived from our IBE system has some interesting properties [4].

7 Escrow ElGamal encryption

In this section we note that the Weil pairing enables us to add a global escrow capability to the ElGamal encryption system. A single escrow key enables the decryption of ciphertexts encrypted under any public key. Paillier and Yung have shown how to add a global escrow capability to the Paillier encryption system [28]. Our ElGamal escrow system works as follows:

Setup: The algorithm works as follows:

Step 1: Choose a large k-bit prime p such that $p = 2 \bmod 3$ and $p = 6q - 1$ for some prime $q > 3$. Let E be the elliptic curve defined by $y^2 = x^3 + 1$ over $\mathbb{F}_p$. Choose an arbitrary $P \in E/\mathbb{F}_p$ of order q.

Step 2: Pick a random $s \in \mathbb{Z}_q^*$ and set $Q = sP$.

Step 3: Choose a cryptographic hash function $H : \mathbb{F}_{p^2} \to \{0,1\}^n$.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = E/\mathbb{F}_p \times \{0,1\}^n$. The system parameters are $\mathsf{params} = \langle p, n, P, Q, H \rangle$. The escrow key is $s \in \mathbb{Z}_q^*$.

keygen: A user generates a public/private key pair for herself by picking a random $x \in \mathbb{Z}_q^*$ and computing $P_{pub} = xP$. Her private key is x, her public key is P_{pub}.

Encrypt: To encrypt $M \in \{0,1\}^n$ under the public key P_{pub} do the following: (1) pick a random $r \in \mathbb{Z}_q^*$, and (2) set the ciphertext to be:

$$C = \langle rP,\ M \oplus H(g^r) \rangle \quad \text{where} \quad g = \hat{e}(P_{pub}, Q) \in \mathbb{F}_{p^2}$$

Decrypt: Let $C = \langle U, V \rangle$ be a ciphertext encrypted using P_{pub}. If $U \in E/\mathbb{F}_p$ is not a point of order q reject the ciphertext. To decrypt C using the private key x do:

$$V \oplus H(\hat{e}(U, xQ)) = M$$

Escrow-decrypt: To decrypt $C = \langle U, V \rangle$ using the escrow key s do:

$$V \oplus H(\hat{e}(U, sP_{pub})) = M$$

A standard argument shows that assuming BDH the system has semantic security in the random oracle model (recall that since DDH is easy we cannot prove semantic security based on DDH). Yet, the escrow agent can decrypt any ciphertext encrypted using any user's public key. The decryption capability of the escrow agent can be distributed using the PKG distribution techniques described in Section 6.

Using a similar hardness assumption, Verheul [35] has recently described an ElGamal encryption system with non-global escrow. Each user constructs a public key with two corresponding private keys, and gives one of the private keys to the trusted third party. Although both private keys can be used to decrypt, only the user's private key can be used simultaneously as the signing key for a discrete logarithm based signature scheme.

8 Summary and open problems

We defined chosen ciphertext security for identity-based systems and proposed a fully functional IBE scheme. The scheme has chosen ciphertext security in the random oracle model assuming BDH, a natural analogue of the computational Diffie-Hellman problem. The BDH assumption deserves further study considering the powerful cryptosystems derived from it. For example, it could be interesting to see whether the techniques of [23] can be used to prove that the BDH assumption is equivalent to the discrete log assumption on the curve for certain primes p.

It is natural to try and build chosen ciphertext secure identity based systems that are secure under standard complexity assumptions (rather than the random oracle model). One might hope to use the techniques of Cramer-Shoup [7] to provide chosen ciphertext security based on DDH. Unfortunately, as mentioned in Section 3, the DDH assumption is false in the group of points on the curve E. However, simple variants of DDH do seem to hold. In particular, the following two distributions appear to be computationally indistinguishable: $\langle P, aP, bP, cP, abcP \rangle$ and $\langle P, aP, bP, cP, rP \rangle$ where a, b, c, r are random in $\mathbb{Z}_q$. We refer to this assumption as BDDH. A chosen ciphertext secure identity-based system strictly based on BDDH would be a plausible analogue of the Cramer-Shoup system.

Acknowledgments

The authors thank Moni Naor and Alice Silverberg for helpful discussions about this work.

References

[1] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, "Relations among notions of security for public-key encryption schemes", Proc. Crypto '98, pp. 26–45, 1998.

[2] D. Boneh, “The decision Diffie-Hellman problem”, In Proceedings of the Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science, Vol. 1423, Springer-Verlag, pp. 48–63, 1998.

[3] D. Boneh, M. Franklin, “Identity based encryption from the Weil pairing”, extended abstract in Proc. of Crypto ’2001.

[4] D. Boneh, B. Lynn, H. Shacham, “Short signatures from the Weil pairing”, in Proceedings of Asiacrypt ’2001.

[5] M. Bellare, A. Boldyreva, S. Micali, “Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements”, Proc. Eurocrypt 2000, LNCS 1807, 2000.

[6] J. Coron, “On the exact security of Full-Domain-Hash”, Proc. of Crypto 2000.

[7] R. Cramer and V. Shoup, “A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack”, in proc. Crypto ’98, pp. 13–25.

[8] Y. Desmedt and J. Quisquater, “Public-key systems based on the difficulty of tampering”, Proc. Crypto ’86, pp. 111-117, 1986.

[9] G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan, “Conditional Oblivious Transfer and Timed-Release Encryption”, Proc. of Eurocrypt ’99.

[10] D. Dolev, C. Dwork, M. Naor, “Non-malleable cryptography”, SIAM J. of Computing, Vol. 30(2), pp. 391–437, 2000.

[11] U. Feige, A. Fiat and A. Shamir, “Zero-knowledge proofs of identity”, *J. Cryptology*, vol. 1, pp. 77–94, 1988.

[12] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems”, Proc. Crypto ’86, pp. 186–194, 1986.

[13] E. Fujisaki and T. Okamoto, “Secure integration of asymmetric and symmetric encryption schemes”, Proc. Crypto ’99, pp. 537–554, 1999.

[14] G. Frey, M. Müller, H. Rück, “The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems”, IEEE Tran. on Info. Th., Vol. 45, pp. 1717–1718, 1999.

[15] S. Galbraith, “Supersingular curves in cryptography”, in proc. of Asiacrypt ’2001.

[16] P. Gemmell, “An introduction to threshold cryptography”, in CryptoBytes, a technical newsletter of RSA Laboratories, Vol. 2, No. 7, 1997.

[17] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, “Secure Distributed Key Generation for Discrete-Log Based Cryptosystems”, *Advances in Cryptology – Eurocrypt ’99*, Springer-Verlag LNCS 1592, pp. 295–310, 1999.

[18] O. Goldreich, B. Pfitzmann and R. Rivest, “Self-delegation with controlled propagation -or- What if you lose your laptop”, Proc. Crypto ’98, pp. 153–168, 1998.

[19] D. Hühnlein, M. Jacobson, D. Weber, “Towards Practical Non-interactive Public Key Cryptosystems Using Non-maximal Imaginary Quadratic Orders”, Proc. SAC 2000, Springer-Verlag LNCS 2012, pp. 275–287, 2000.

[20] A. Joux, "A one round protocol for tripartite Diffie-Hellman", Proc of ANTS 4, LNCS 1838, pp. 385–394, 2000.

[21] A. Joux, K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups", available from `eprint.iacr.org`.

[22] S. Lang, "Elliptic functions", Addison-Wesley, Reading, 1973.

[23] U. Maurer, "Towards proving the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms", Proc. Crypto '94, pp. 271–281.

[24] U. Maurer and Y. Yacobi, "Non-interactive public-key cryptography", proc. Eurocrypt '91, pp. 498–507.

[25] A. Menezes, T. Okamoto, S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field", IEEE Tran. on Info. Th., Vol. 39, pp. 1639–1646, 1993.

[26] A. Menezes, P. van Oorrschot and S. Vanstone, "Handbook of applied cryptography", CRC Press, Boca Raton, FL, 1996.

[27] V. Miller, "Short programs for functions on curves", unpublished manuscript.

[28] P. Paillier and M. Yung, "Self-escrowed public-key infrastructures" in Proc. ICISC, pp. 257–268, 1999.

[29] C. Rackoff, D. Simon, "Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack", in proc. Crypto '91, pp. 433–444, 1991.

[30] R. Rivest, A. Shamir and D. Wagner, "Time lock puzzles and timed release cryptography," Technical report, MIT/LCS/TR-684

[31] R. Sakai, K. Ohgishi, M. Kasahara, "Cryptosystems based on pairings", SCIS 2001, Oiso, Japan.

[32] A. Shamir, "Identity-based cryptosystems and signature schemes", Proc. Crypto '84, pp. 47–53.

[33] S. Tsuji and T. Itoh, "An ID-based cryptosystem based on the discrete logarithm problem", *IEEE Journal on Selected Areas in Communication*, vol. 7, no. 4, pp. 467–473, 1989.

[34] H. Tanaka, "A realization scheme for the identity-based cryptosystem", Proc. Crypto '87, pp. 341–349, 1987.

[35] E. Verheul, "Evidence that XTR is more secure than supersingular elliptic curve cryptosystems", Proc. Eurocrypt 2001.

A Proofs of Key Lemmas

Proof of Lemma 4.2: We show how to construct a OWE adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\epsilon/e(1+q_E)$ against $\mathsf{BasicPub}$. The game between the challenger and the adversary $\mathcal{B}$ starts with the challenger first generating a random public key by running algorithm keygen of $\mathsf{BasicPub}$. The result is a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$ and a private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. Let q be the order of $\mathbb{G}_1, \mathbb{G}_2$. The challenger then picks a random plaintext $M \in \mathcal{M}$ and encrypts M using algorithm $\mathsf{encrypt}$ of $\mathsf{BasicPub}$. It gives K_{pub} and the resulting ciphertext $C = \langle U, V \rangle$ to algorithm $\mathcal{B}$.

Algorithm $\mathcal{B}$ is supposed to output a guess for M. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ gives algorithm $\mathcal{A}$ the $\mathsf{BasicIdent}$ system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2 \rangle$. Here $\mathbb{G}_1$,$\mathbb{G}_2$, $\hat{e}$, n, P, P_{pub}, H_2 are taken from K_{pub}, and H_1 is a random oracle controlled by $\mathcal{B}$.

H_1-queries: At any time algorithm $\mathcal{A}$ can query the random oracle H_1. To respond to these queries algorithm $\mathcal{B}$ maintains a list of tuples $\langle \mathsf{ID}_j, Q_j, b_j, c_j \rangle$ as explained below. We refer to this list as the H_1^{list}. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point ID_i algorithm $\mathcal{B}$ responds as follows:

1. If the query ID_i already appears on the H_1^{list} in a tuple $\langle \mathsf{ID}_i, Q_i, b_i, c_i \rangle$ then Algorithm $\mathcal{B}$ responds with $H_1(\mathsf{ID}_i) = Q_i \in \mathbb{G}_1^*$.
2. Otherwise, $\mathcal{B}$ generates a random $coin \in \{0, 1\}$ so that $\Pr[coin = 0] = \delta$ for some δ that will be determined later.
3. Algorithm $\mathcal{B}$ picks a random $0 \neq b \in \mathbb{Z}_q^*$.
   If $coin = 0$ compute $Q_i = bP \in \mathbb{G}_1^*$. If $coin = 1$ compute $Q_i = bQ_{\mathsf{ID}} \in \mathbb{G}_1^*$.
4. Algorithm $\mathcal{B}$ adds the tuple $\langle \mathsf{ID}_i, Q_i, b, coin \rangle$ to the H_1^{list} and responds to $\mathcal{A}$ with $H_1(\mathsf{ID}_i) = Q_i$.

Note that either way Q_i is uniform in $\mathbb{G}_1^*$ and is independent of $\mathcal{A}$'s current view as required.

Phase 1: Let ID_i be a private key extraction query issued by algorithm $\mathcal{A}$. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Run the above algorithm for responding to H_1-queries to obtain a $Q_i \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_i) = Q_i$. Let $\langle \mathsf{ID}_i, Q_i, b_i, coin_i \rangle$ be the corresponding tuple on the H_1^{list}. If $coin_i = 1$ then $\mathcal{B}$ reports failure and terminates. The attack on $\mathsf{BasicPub}$ failed.
2. We know $coin_i = 0$ and hence $Q_i = b_i P$. Define $d_i = b_i P_{pub} \in \mathbb{G}_1^*$. Observe that $d_i = sQ_i$ and therefore d_i is the private key associated to the public key ID_i. Give d_i to algorithm $\mathcal{A}$.

Challenge: Once algorithm $\mathcal{A}$ decides that Phase 1 is over it outputs a public key $\mathsf{ID} \in \{0, 1\}^*$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ responds as follows:

1. Run the above algorithm for responding to H_1-queries to obtain a $Q \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}) = Q$. Let $\langle \mathsf{ID}, Q, b, coin \rangle$ be the corresponding tuple on the H_1^{list}. If $coin = 0$ then $\mathcal{B}$ reports failure and terminates. The attack on $\mathsf{BasicPub}$ failed.
2. We know $coin = 1$ and hence $Q = bQ_{\mathsf{ID}}$.
3. Let $C = \langle U, V \rangle$ be the challenge ciphertext given to algorithm $\mathcal{B}$. Here $U \in E/\mathbb{F}_p$. Set $C' = \langle b^{-1}U, V \rangle$, where b^{-1} is the inverse of $b \bmod q$. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the challenge C'. Note that C' is a $\mathsf{BasicIdent}$ encryption of M under the public key ID as required. To see this first observe that, since $H_1(\mathsf{ID}) = Q$, the private key corresponding to ID is $d'_{\mathsf{ID}} = sQ$. Second, observe that

$$\hat{e}(b^{-1}U, d'_{\mathsf{ID}}) = \hat{e}(b^{-1}U, sQ) = \hat{e}(U, sb^{-1}Q) = \hat{e}(U, sQ_{\mathsf{ID}}) = \hat{e}(U, d_{\mathsf{ID}}).$$

   Hence, the $\mathsf{BasicIdent}$ decryption of C' using d'_{ID} is the same as the $\mathsf{BasicPub}$ decryption of C using d_{ID}.

Phase 2: Algorithm $\mathcal{B}$ responds to private key extraction queries in the same way it did in Phase 1.

Guess: Eventually algorithm $\mathcal{A}$ will produce a guess M'. Algorithm $\mathcal{B}$ outputs M' as its guess for the decryption of C.

Claim: If algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, if $\mathcal{B}$ does not abort then $\Pr[M = M'] \geq \epsilon$. The probability is over the random bits used by $\mathcal{A}, \mathcal{B}$ and the challenger.

Proof of Claim: All responses to H_1-queries are as in the real attack since each response is uniformly and independently distributed in $\mathbb{G}_1^*$. All responses to private key extraction queries are valid. Finally, the challenge ciphertext C' given to $\mathcal{A}$ is the BasicIdent encryption of the random plaintext M under the public key ID chosen by $\mathcal{A}$. Therefore, by definition of algorithm $\mathcal{A}$, it will output $M' = M$ with probability at least ϵ. □

It remains to calculate the probability that algorithm $\mathcal{B}$ aborts during the simulation. Suppose $\mathcal{A}$ makes a total of q_E private key extraction queries. Then the probability that $\mathcal{B}$ does not abort in phases 1 or 2 is δ^{q_E}. The probability that it does not abort during the challenge step is $1-\delta$. Therefore, the probability that $\mathcal{B}$ does not abort during the simulation is $\delta^{q_E}(1-\delta)$. This value is maximized at $\delta_{opt} = 1 - 1/(q_E + 1)$. Using δ_{opt}, the probability that $\mathcal{B}$ does not abort is at least $1/e(1 + q_E)$. This shows that $\mathcal{B}$'s advantage is at least $\epsilon/e(1 + q_E)$ as required. □

The analysis used in the proof of Lemma 4.2 uses a similar technique to Coron's analysis of the Full Domain Hash signature scheme [6].

Proof of Lemma 4.3: Algorithm B is given as input the BDH parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ produced by $\mathcal{IG}$ and and a random instance $\langle P, aP, bP, cP \rangle = \langle P, P_1, P_2, P_3 \rangle$ of the BDH problem for these parameters, i.e. P is random in $\mathbb{G}_1^*$ and $\langle a, b, c \rangle$ are random in $\mathbb{Z}_q^*$ where q is the order of $\mathbb{G}_1, \mathbb{G}_2$. Let $D = \hat{e}(P, P)^{abc} \in \mathbb{G}_2$ be the solution to this BDH problem. Algorithm $\mathcal{B}$ finds D by interacting with $\mathcal{A}$ as follows:

Challenge: Algorithm $\mathcal{B}$ creates the BasicPub public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$ by setting $P_{pub} = P_1$ and $Q_{\mathsf{ID}} = P_2$. It then picks a random string $R \in \{0,1\}^n$ and defines C to be the ciphertext $C = \langle P_3, R \rangle$. Algorithm $\mathcal{B}$ gives K_{pub} and C as the challenge to $\mathcal{A}$. Observe that the (unknown) private key associated to K_{pub} is $d_{\mathsf{ID}} = aQ_{\mathsf{ID}} = abP$. Furthermore, the (unknown) decryption of C is $R \oplus H_2(\hat{e}(P_3, d_{\mathsf{ID}})) = R \oplus H_2(D)$. We set $M = R \oplus H_2(D)$.

H_2-queries: At any time algorithm $\mathcal{A}$ may issue queries to H_2. To respond to these queries $\mathcal{B}$ maintains a list of tuples called the H_2^{list}. Each entry in the list is a tuple of the form $\langle X_j, H_j \rangle$. Initially the list is empty. To respond to query X_i algorithm $\mathcal{B}$ does the following:

1. If the query X_i already appears on the H_2^{list} in a tuple $\langle X_j, H_j \rangle$ then respond with $H_2(X_j) = H_j$.
2. Otherwise, $\mathcal{B}$ just picks a random string $H_i \in \{0,1\}^n$ and adds the tuple $\langle X_i, H_i \rangle$ to the H_2^{list}. It responds to $\mathcal{A}$ with $H_2(X_i) = H_i$.

Guess: Eventually, algorithm $\mathcal{A}$ will output its guess M' to the decryption of C. At this point $\mathcal{B}$ picks a random tuple $\langle X_j, H_j \rangle$ from the H_2^{list} and outputs X_j as the solution to the given instance of BDH.

It is easy to see that $\mathcal{A}$'s view is identical to its view in the real attack. Indeed, the challenge is distributed as in the real attack, and all responses to H_2-queries are uniform and independent in $\{0,1\}^n$. Hence, $\Pr[M = M'] \geq \epsilon$. It remains to calculate the probability that $\mathcal{B}$ outputs the required D.

Let $\mathcal{H}$ be the event that at the end of the simulation D appears in a tuple on the H_2^{list}. Let $\delta = \Pr[\mathcal{H}]$. Note that when D does not appear in a tuple on the H_2^{list} the decryption of C is independent of $\mathcal{A}$'s view (since $H_2(D)$ is independent of $\mathcal{A}$'s view). Therefore, $\Pr[M = M' \mid \neg\mathcal{H}] \le 1/2^n$. Then, we have:

$$\begin{aligned} \epsilon \;\le\; \Pr[M' = M] \;&=\; \Pr[M = M' \mid \mathcal{H}] \cdot \Pr[\mathcal{H}] + \Pr[M = M' \mid \neg\mathcal{H}] \cdot \Pr[\neg\mathcal{H}] \\ \le\; \Pr[\mathcal{H}] + \Pr[M = M' \mid \neg\mathcal{H}] \cdot \Pr[\neg\mathcal{H}] \;&\le\; \delta + \frac{1}{2^n}(1-\delta) \end{aligned}$$

Solving for δ we get $\Pr[\mathcal{H}] = \delta > \delta(1 - 1/2^n) \ge \epsilon - 1/2^n$. It follows that $\mathcal{B}$ produces the correct answer with probability at least $\delta/q_{H_2} \ge (\epsilon - 1/2^n)/q_{H_2}$ as required. □

Proof of Lemma 4.6: We construct an IND-CCA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\epsilon/e(1 + q_E + q_D)$ against $\mathsf{BasicPub}^{hy}$. The game between the challenger and the adversary $\mathcal{B}$ starts with the challenger first generating a random public key by running algorithm keygen of $\mathsf{BasicPub}^{hy}$. The result is a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2, H_3, H_4 \rangle$ and a private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. The challenger gives K_{pub} to algorithm $\mathcal{B}$.

Algorithm $\mathcal{B}$ mounts an IND-CCA attack on the key K_{pub} using the help of algorithm $\mathcal{A}$. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup: Same as in Lemma 4.2 (with H_3, H_4 included in the system parameters given to $\mathcal{A}$).

H_1**-queries:** These queries are handled as in Lemma 4.2.

Phase 1: Private key queries. Handled as in Lemma 4.2.

Phase 1: Decryption queries. Let $\langle \mathsf{ID}_i, C_i \rangle$ be a decryption query issued by algorithm $\mathcal{A}$. Let $C_i = \langle U_i, V_i, W_i \rangle$. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Run the above algorithm for responding to H_1-queries to obtain a $Q_i \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_i) = Q_i$. Let $\langle \mathsf{ID}_i, Q_i, b_i, coin_i \rangle$ be the corresponding tuple on the H_1^{list}.
2. Suppose $coin_i = 0$. In this case run the algorithm for responding to private key queries to obtain the private key for the public key ID_i. Then use the private key to respond to the decryption query.
3. Suppose $coin_i = 1$. Then $Q_i = b_i Q_{\mathsf{ID}}$.
   - Recall that $U_i \in \mathbb{G}_1$. Set $C_i' = \langle b_i U_i,\ V_i,\ W_i \rangle$. Let $d_i = sQ_i$ be the (unknown) FullIdent private key corresponding to ID_i. Then the FullIdent decryption of C_i using d_i is the same as the $\mathsf{BasicPub}^{hy}$ decryption of C_i' using d_{ID}. To see this observe that:

   $$\hat{e}(b_i U_i, d_{\mathsf{ID}}) = \hat{e}(b_i U_i, sQ_{\mathsf{ID}}) = \hat{e}(U_i, s b_i Q_{\mathsf{ID}}) = \hat{e}(U_i, sQ_i) = \hat{e}(U_i, d_i).$$

   - Relay the decryption query $\langle C_i' \rangle$ to the challenger and relay the challenger's response back to $\mathcal{A}$.

Challenge: Once algorithm $\mathcal{A}$ decides that Phase 1 is over it outputs a public key ID and two messages M_0, M_1 on which it wishes to be challenged. Algorithm $\mathcal{B}$ responds as follows:

1. Algorithm $\mathcal{B}$ gives the challenger M_0, M_1 as the messages that it wishes to be challenged on. The challenger responds with a $\mathsf{BasicPub}^{hy}$ ciphertext $C = \langle U, V, W \rangle$ such that C is the encryption of M_c for a random $c \in \{0, 1\}$.
2. Next, $\mathcal{B}$ runs the algorithm for responding to H_1-queries to obtain a $Q \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}) = Q$. Let $\langle \mathsf{ID}, Q, b, coin \rangle$ be the corresponding tuple on the H_1^{list}. If $coin = 0$ then $\mathcal{B}$ reports failure and terminates. The attack on $\mathsf{BasicPub}^{hy}$ failed.
3. We know $coin = 1$ and therefore $Q = bQ_{\mathsf{ID}}$. Recall that when $C = \langle U, V, W \rangle$ we have $U \in \mathbb{G}_1^*$. Set $C' = \langle b^{-1}U,\ V,\ W \rangle$, where b^{-1} is the inverse of $b \bmod q$. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the

challenge C'. Note that, as in the proof of Lemma 4.2, C' is a $\mathsf{FullIdent}$ encryption of M_c under the public key ID as required.

Phase 2: Private key queries. Algorithm $\mathcal{B}$ responds to private key extraction queries in the same way it did in Phase 1.

Phase 2: Decryption queries. Algorithm $\mathcal{B}$ responds to decryption queries in the same way it did in Phase 1. However, if the resulting decryption query relayed to the challenger is equal to the challenge ciphertext $C = \langle U, V, W \rangle$ then $\mathcal{B}$ reports failure and terminates. The attack on $\mathsf{BasicPub}^{hy}$ failed.

Guess: Eventually algorithm $\mathcal{A}$ produces a guess c' for c. Algorithm $\mathcal{B}$ outputs c' as its guess for c.

Claim: If algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, if $\mathcal{B}$ does not abort then $|\Pr[c = c'] - \frac{1}{2}| \geq \epsilon$. The probability is over the random bits used by $\mathcal{A}, \mathcal{B}$ and the challenger.

Proof of Claim: The responses to H_1-queries are as in the real attack since each response is uniformly and independently distributed in $\mathbb{G}_1^*$. All responses to private key extraction queries and decryption queries are valid. Finally, the challenge ciphertext C' given to $\mathcal{A}$ is the $\mathsf{FullIdent}$ encryption of M_c for some random $c \in \{0, 1\}$. Therefore, by definition of algorithm $\mathcal{A}$ we have that $|\Pr[c = c'] - \frac{1}{2}| \geq \epsilon$. $\square$

It remains to bound the probability that algorithm $\mathcal{B}$ aborts during the simulation. The algorithm could abort for three reasons: (1) a bad private key query from $\mathcal{A}$ during phases 1 or 2, (2) $\mathcal{A}$ chooses a bad ID to be challenged on, or (3) a bad decryption query from $\mathcal{A}$ during phase 2. We define three corresponding events:

$\mathcal{E}_1$ is the event that $\mathcal{A}$ issues a private key query during phase 1 or 2 that causes algorithm $\mathcal{B}$ to abort.

$\mathcal{E}_2$ is the event that $\mathcal{A}$ choose a public key ID to be challenged on that causes algorithm $\mathcal{B}$ to abort.

$\mathcal{E}_3$ is the event that during phase 2 of the simulation Algorithm $\mathcal{A}$ issues a decryption query $\langle \mathsf{ID}_i, C_i \rangle$ so that the decryption query that $\mathcal{B}$ would relay to the $\mathsf{BasicPub}^{hy}$ challenger is equal to C. Recall that $C = \langle U, V, W \rangle$ is the challenge ciphertext from the $\mathsf{BasicPub}^{hy}$ challenger.

Claim: $\Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2 \wedge \neg\mathcal{E}_3] \geq \delta^{q_E + q_D}(1 - \delta)$

Proof of Claim: We prove the claim by induction on the maximum number of queries $q_E + q_D$ made by the adversary. Let $i = q_E + q_D$ and let $\mathcal{E}^{0\ldots i}$ be the event that $\mathcal{E}_1 \vee \mathcal{E}_2 \vee \mathcal{E}_3$ happens after $\mathcal{A}$ issues at most i queries. Similarly, let $\mathcal{E}^i$ be the event that $\mathcal{E}_1 \vee \mathcal{E}_2 \vee \mathcal{E}_3$ happens for the first time when $\mathcal{A}$ issues the i'th query. We prove by induction on i that $\Pr[\neg\mathcal{E}^{0\ldots i}] \geq \delta^i(1 - \delta)$.

For $i = 0$ the claim is trivial since by definition $\Pr[\neg\mathcal{E}^{0\ldots 0}] \geq 1 - \delta$. Now, suppose the claim holds for $i - 1$. Then

$$\begin{aligned}
\Pr[\neg\mathcal{E}^{0\ldots i}] &= \Pr[\neg\mathcal{E}^{0\ldots i} \mid \neg\mathcal{E}^{0\ldots i-1}] \Pr[\neg\mathcal{E}^{0\ldots i-1}] \\
&= \Pr[\neg\mathcal{E}^{i} \mid \neg\mathcal{E}^{0\ldots i-1}] \Pr[\neg\mathcal{E}^{0\ldots i-1}] \geq \Pr[\neg\mathcal{E}^{i} \mid \neg\mathcal{E}^{0\ldots i-1}] \delta^{i-1}(1 - \delta)
\end{aligned}$$

Hence, it suffices to bound $q_i = \Pr[\neg\mathcal{E}^i \mid \neg\mathcal{E}^{0\ldots i-1}]$. In other words, we bound the probability that the i'th query does not cause $\mathcal{E}^i$ to happen given that the first $i-1$ queries did not. Consider the i'th query issued by $\mathcal{A}$ during the simulation. The query is either a private key query for $\langle \mathsf{ID}_i \rangle$ or a decryption query for $\langle \mathsf{ID}_i, C_i \rangle$ where $C_i = \langle U_i, V_i, W_i \rangle$. If the query is a decryption query we assume it takes place during phase 2 since otherwise it has no effect on $\mathcal{E}_3$.

Let $H_1(\mathsf{ID}_i) = Q_i$ and let $\langle \mathsf{ID}_i, Q_i, b_i, coin_i \rangle$ be the corresponding tuple on the H_1^{list}. Recall that when $coin_i = 0$ the query cannot cause event $\mathcal{E}_1$ to happen. Similarly, when $coin_i = 0$ the query cannot cause event $\mathcal{E}_3$ to happen since in this case $\mathcal{B}$ does not relay a decryption query to the $\mathsf{BasicPub}^{hy}$ challenger. We use these facts to bound q_i. There are four cases to consider. In the first three cases we assume ID_i is not equal to the public key ID on which $\mathcal{A}$ is being challenged.

Case 1. The i'th query is the first time $\mathcal{A}$ issues a query containing ID_i. In this case $\Pr[coin_i = 0] = \delta$ and hence $q_i \geq \delta$.

Case 2. The public key ID_i appeared in a previous private key query. Since by assumption this earlier private key query did not cause $\mathcal{E}^{0\ldots i-1}$ to happen we know that $coin_i = 0$. Hence, we have $q_i = 1$.

Case 3. The public key ID_i appeared in a previous decryption query. Since by assumption this earlier decryption query did not cause event $\mathcal{E}^{0\ldots i-1}$ to happen we have that either $coin_i = 0$ or $coin_i$ is independent of $\mathcal{A}$'s current view. Either way we have that $q_i \geq \delta$.

Case 4. The public key ID_i is equal to the public key ID on which $\mathcal{A}$ is being challenged. Then, by definition, the i'th query cannot be a private key query. Therefore, it must be a decryption query $\langle \mathsf{ID}_i, C_i \rangle$. Furthermore, since $\mathcal{E}_2$ did not happen we know that $coin_i = 1$ and hence $\mathcal{B}$ will relay a decryption query C'_i to the $\mathsf{BasicPub}^{hy}$ challenger. Let C' be the challenge ciphertext given to $\mathcal{A}$. By definition we know that $C_i \neq C'$. It follows that $C'_i \neq C$. Therefore this query cannot cause event $\mathcal{E}_3$ to happen. Hence, in this case $q_i = 1$.

To summarize, we see that whatever the i'th query is, we have that $q_i \geq \delta$. Therefore, $\Pr[\neg\mathcal{E}^{0\ldots i}] \geq \delta^i(1-\delta)$ as required. The claim now follows by setting $i = q_E + q_D$. □

To conclude the proof of Lemma 4.6 it remains to optimize the choice of δ. Since $\Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2 \wedge \neg\mathcal{E}_3] \geq \delta^{q_E+q_D}(1-\delta)$ the success probability is maximized at $\delta_{opt} = 1 - 1/(q_E + q_D + 1)$. Using δ_{opt}, the probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(1+q_E+q_D)}$. This shows that $\mathcal{B}$'s advantage is at least $\epsilon/e(1+q_E+q_D)$ as required. □

B Definition of the Weil pairing

We define the Weil pairing and show how to efficiently compute it using an algorithm due to Miller [27]. To be concrete we present the algorithm as it applies to supersingular elliptic curves. The definitions and algorithm easily generalizes to computing the Weil pairing over other elliptic curves.

Let p be a prime satisfying $p = 2 \bmod 3$. Consider the elliptic curve E defined by the equation $y^2 = x^3 + 1$ over $\mathbb{F}_p$. We state a few elementary facts about this curve:

Fact 1: Since $x^3 + 1$ is a permutation on $\mathbb{F}_p$ it easily follows that $E/\mathbb{F}_p$ contains $p+1$ points. We let O denote the point at infinity. Furthermore, the group of points over $\mathbb{F}_p$ forms a cyclic group of order $p+1$. Let P be a generator of this group. For simplicity, set $n = p + 1$.

Fact 2: The group of points of E over $\mathbb{F}_{p^2}$ contains a point Q of order n which is linearly independent of the points in $E/\mathbb{F}_p$. Hence, $E/\mathbb{F}_{p^2}$ contains the group $\mathbb{Z}_n^2$. The group is generated by $P \in E/\mathbb{F}_p$ and $Q \in Q/\mathbb{F}_{p^2}$. We denote this group by $E[p+1] = E[n]$.

We will be working with the Weil pairing e which maps pairs of points in $E[n]$ into $\mathbb{F}_{p^2}^*$, i.e. $e : E[n] \times E[n] \to \mathbb{F}_{p^2}^*$. To define the pairing, we review the following concepts (see [22, pp. 243–245]):

Divisors A divisor is a formal sum of points on the curve $E/\mathbb{F}_{p^2}$. We write divisors as $\mathcal{A} = \sum_P a_p(P)$ where $a_P \in \mathbb{Z}$ and $P \in E/\mathbb{F}_{p^2}$. For example, $\mathcal{A} = 3(P_1) - 2(P_2) - (P_3)$ is a divisor. We will only consider divisors $\mathcal{A} = \sum_P a_p(P)$ where $\sum_P a_p = 0$.

Functions Roughly speaking, a function f on the curve $E/\mathbb{F}_{p^2}$ can be viewed as a rational function $f(x,y) \in \mathbb{F}_{p^2}(x,y)$. For any point $P=(x,y) \in E/\mathbb{F}_{p^2}$ we define $f(P)=f(x,y)$.

Divisors of functions Let f be a function on the curve $E/\mathbb{F}_{p^2}$. We define its divisor, denoted by (f), as $(f) = \sum_P \mathrm{ord}_P(f)$. Here $\mathrm{ord}_P(f)$ is the order of the zero that f has at the point P. For example, let $ax+by+c=0$ be the line passing through the points $P_1, P_2 \in E/\mathbb{F}_{p^2}$ with $P_1 \neq \pm P_2$. This line intersects the curve at third point $P_3 \in E/\mathbb{F}_{p^2}$. Then the function $f(x,y) = ax+by+c$ has three zeroes P_1, P_2, P_3 and a pole of order 3 at infinity. The divisor of f is $(f) = (P_1)+(P_2)+(P_3)-3(O)$.

Principal divisors Let $\mathcal{A}$ be a divisor. If there exists a function f such that $(f)=\mathcal{A}$ then we say that $\mathcal{A}$ is a principal divisor. We know that a divisor $\mathcal{A} = \sum_P a_p(P)$ is principal if and only if $\sum_P a_p = 0$ and $\sum_P a_P P = O$. Note that the second summation is using the group action on the curve. Furthermore, given a principal divisor $\mathcal{A}$ there exists a *unique* function f (up to constant multiples) such that $(A)=(f)$.

Equivalence of divisors We say that two divisors $\mathcal{A}, \mathcal{B}$ are equivalent if their difference $\mathcal{A}-\mathcal{B}$ is a principal divisor. We know that any divisor $\mathcal{A} = \sum_P a_p(P)$ (with $\sum_P a_P = 0$) is equivalent to a divisor of the form $\mathcal{A}' = (Q)-(O)$ for some $Q \in E$. Observe that $Q = \sum_P a_P P$.

Notation Given a function f and a divisor $\mathcal{A} = \sum_P a_p(P)$ we define $f(\mathcal{A})$ as $f(\mathcal{A}) = \prod_P f(P)^{a_P}$. Note that since $\sum_P a_P = 0$ we have that $f(\mathcal{A})$ remains unchanged if instead of f we use cf for any $c \in \mathbb{F}_{p^2}$.

We are now ready to define the Weil pairing of two points $P, Q \in E[n]$. Let $\mathcal{A}_P$ be some divisor equivalent to the divisor $(P)-(O)$. We know that $n\mathcal{A}_P$ is a principal divisor (it is equivalent to $n(P)-n(O)$ which is clearly a principal divisor). Hence, there exists a function f_P such that $(f_P) = n\mathcal{A}_P$. Define $\mathcal{A}_Q$ and f_Q analogously. The Weil pairing of P and Q is defined as:

$$e(P,Q) = \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)}$$

This ratio defines the Weil pairing of P and Q whenever it is well defined (no division by zero occurred). If this ratio is undefined we use different divisors $\mathcal{A}_P, \mathcal{A}_Q$ to define $e(P,Q)$. When $P, Q \in E/\mathbb{F}_{p^2}$ we have that $e(P,Q) \in \mathbb{F}_{p^2}$.

We briefly show that the Weil pairing is well defined. That is, the value of $e(P,Q)$ is independent of the choice of the divisor $\mathcal{A}_P$ as long as $\mathcal{A}_P$ is equivalent to $(P)-(O)$ and $\mathcal{A}_P$ leads to a well defined value. The same holds for $\mathcal{A}_Q$. Let $\hat{\mathcal{A}}_P$ be a divisor equivalent to $\mathcal{A}_P$ and let $\hat{f}_P$ be a function so that $(\hat{f}_P) = n\hat{\mathcal{A}}_P$. Then $\hat{\mathcal{A}}_P = \mathcal{A}_P + (g)$ for some function g and $\hat{f}_P = f_P \cdot g^n$. We have that:

$$e(P,Q) = \frac{\hat{f}_P(\mathcal{A}_Q)}{f_Q(\hat{\mathcal{A}}_P)} = \frac{f_P(\mathcal{A}_Q) g(\mathcal{A}_Q)^n}{f_Q(\mathcal{A}_P) f_Q((g))} = \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)} \cdot \frac{g(n\mathcal{A}_Q)}{f_Q((g))} = \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)} \cdot \frac{g((f_Q))}{f_Q((g))} = \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)}$$

The last equality follows from the following fact known as Weil reciprocity: for any two functions f, g we have that $f(\,(g)\,) = g(\,(f)\,)$. Hence, the Weil pairing is well defined.

Fact B.1 *The Weil pairing has the following properties:*

- *For all* $P \in E[n]$ *we have:* $e(P,P) = 1$.
- *Bilinear:* $e(P_1+P_2, Q) = e(P_1,Q) \cdot e(P_2,Q)$ *and* $e(P, Q_1+Q_2) = e(P,Q_1) \cdot e(P,Q_2)$.

- *When $P, Q \in E[n]$ are collinear then $e(P,Q) = 1$. Similarly, $e(P,Q) = e(Q,P)^{-1}$.*
- *n'th root: for all $P, Q \in E[n]$ we have $e(P,Q)^n = 1$.*
- *Non-degenerate: if P satisfies $e(P,Q) = 1$ for all $Q \in E[n]$ then $P = O$.*

As discussed in Section 5, our concrete IBE scheme uses the modified Weil pairing $\hat{e}(P,Q) = e(P,\phi(Q))$, where ϕ is an automorphism on the group of points of E given by $\phi(x,y) = (\zeta x, y)$, for $\zeta \in \mathbb{F}_{p^2}$ a non-trivial cube root of unity modulo p.

C Computing the Weil pairing

Given two points $P, Q \in E[n]$ we show how to compute $e(P,Q) \in \mathbb{F}_{p^2}^*$ using $O(\log p)$ arithmetic operations in $\mathbb{F}_p$. We assume $P \neq Q$. We proceed as follows: pick two random points $R_1, R_2 \in E[n]$. Consider the divisors $\mathcal{A}_P = (P + R_1) - (R_1)$ and $\mathcal{A}_Q = (Q + R_2) - (R_2)$. These divisors are equivalent to $(P) - (O)$ and $(Q) - (O)$ respectively. Hence, we can use $\mathcal{A}_P$ and $\mathcal{A}_Q$ to compute the Weil pairing as:

$$e(P,Q) = \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_P)} = \frac{f_P(Q+R_2) f_Q(R_1)}{f_P(R_2) f_Q(P+R_1)}$$

This expression is well defined with very high probability over the choice of R_1, R_2 (the probability of failure is at most $O(\frac{\log p}{p})$). In the rare event that a division by zero occurs during the computation of $e(P,Q)$ we simply pick new random points R_1, R_2 and repeat the process.

To evaluate $e(P,Q)$ it suffices to show how to evaluate the function f_P at $\mathcal{A}_Q$. Evaluating $f_Q(\mathcal{A}_P)$ is done analogously. We evaluate $f_P(\mathcal{A}_Q)$ using repeated doubling. For a positive integer b define the divisor

$$\mathcal{A}_b = b(P + R_1) - b(R_1) - (bP) + (O)$$

It is a principal divisor and therefore there exists a function f_b such that $(f_b) = \mathcal{A}_b$. Observe that $(f_P) = (f_n)$ and hence, $f_P(\mathcal{A}_Q) = f_n(\mathcal{A}_Q)$. It suffices to show how to evaluate $f_n(\mathcal{A}_Q)$.

Lemma C.1 *There is an algorithm $\mathcal{D}$ that given $f_b(\mathcal{A}_Q), f_c(\mathcal{A}_Q)$ and $bP, cP, (b+c)P$ for some $b, c > 0$ outputs $f_{b+c}(\mathcal{A}_Q)$. The algorithm only uses a (small) constant number of arithmetic operations in $\mathbb{F}_{p^2}$.*

Proof We first define two auxiliary linear functions g_1, g_2:

1. Let $a_1 x + b_1 y + c_1 = 0$ be the line passing through the points bP and cP (if $b = c$ then let $a_1 x + b_1 y + c_1 = 0$ be the line tangent to E at bP). Define $g_1(x,y) = a_1 x + b_1 y + c_1$.
2. Let $x + c_2 = 0$ be the vertical line passing through the point $(b+c)P$. Define $g_2(x,y) = x + c_2$

The divisors of these functions are:

$$\begin{aligned}(g_1) &= (bP) + (cP) + (-(b+c)P) - 3(O) \\ (g_2) &= ((b+c)P) + (-(b+c)P) - 2(O)\end{aligned}$$

By definition we have that:

$$\begin{aligned}\mathcal{A}_b &= b(P + R_1) - b(R_1) - (bP) + (O) \\ \mathcal{A}_c &= c(P + R_1) - c(R_1) - (cP) + (O) \\ \mathcal{A}_{b+c} &= (b+c)(P + R_1) - (b+c)(R_1) - ((b+c)P) + (O)\end{aligned}$$

It now follows that: $\mathcal{A}_{b+c} = \mathcal{A}_b + \mathcal{A}_c + (g_1) - (g_2)$. Hence:

$$f_{b+c}(\mathcal{A}_Q) = f_b(\mathcal{A}_Q) \cdot f_c(\mathcal{A}_Q) \cdot \frac{g_1(\mathcal{A}_Q)}{g_2(\mathcal{A}_Q)} \tag{1}$$

This shows that to evaluate $f_{b+c}(\mathcal{A}_Q)$ it suffices to evaluate $g_i(\mathcal{A}_Q)$ for all $i = 1, 2$ and plug the results into equation 1. Hence, given $f_b(\mathcal{A}_Q), f_c(\mathcal{A}_Q)$ and $bP, cP, (b+c)P$ one can compute $f_{b+c}(\mathcal{A}_Q)$ using a constant number of arithmetic operations. □

Denote the output of Algorithm $\mathcal{D}$ of Lemma C.1 by $\mathcal{D}(f_b(\mathcal{A}_Q), f_c(\mathcal{A}_Q), bP, cP, (b+c)P) = f_{b+c}(\mathcal{A}_Q)$. Then one can compute $f_P(\mathcal{A}_Q) = f_n(\mathcal{A}_Q)$ using the following repeated doubling procedure. Let $n = b_m b_{m-1} \ldots b_1 b_0$ be the binary representation of n, i.e. $n = \sum b_i 2^i$.

Init: Set $Z = O$ and $V = f_0(\mathcal{A}_Q) = 1$.

Iterate: For $i = m, m-1, \ldots, 1, 0$ do:

1: If $b_i = 1$ then do: (1) Set $V = \mathcal{D}(V, f_1(\mathcal{A}_Q), Z, P, Z + P)$, and (2) Set $Z = Z + P$.
2: Set $V = \mathcal{D}(V, V, Z, Z, 2Z)$ and $Z = 2Z$.

At the end of each iteration we have that $Z = bP$ and $V = f_b(\mathcal{A}_Q)$ for some $0 \le b \le n$. At the end we have $Z = nP$ and $V = f_n(\mathcal{A}_Q) = f_P(\mathcal{A}_Q)$ as required. To evaluate the Weil pairing $e(P, Q)$ we run the above algorithm once to compute $f_P(\mathcal{A}_Q)$ and once to compute $f_Q(\mathcal{A}_P)$.

Note that the repeated squaring algorithm needs to evaluate $f_1(\mathcal{A}_Q)$. This is easily done since the function $f_1(x, y)$ (whose divisor is $(f_1) = (P + R_1) - (R_1) - (P) + (O)$) can be written out explicitly as follows:

1. Let $a_1 x + b_1 y + c_1 = 0$ be the line passing through the points P and R_1. Define the function: $g_1(x, y) = a_1 x + b_1 y + c_1$.
2. Let $x + c_2 = 0$ be the vertical line passing through the point $P + R_1$. Define the function: $g_2(x, y) = x + c_2$.
3. The function $f_1(x, y)$ is simply $f_1(x, y) = g_2(x, y)/g_1(x, y)$ which is easy to evaluate in $\mathbb{F}_{p^2}$.