*Appears in SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003. An extended abstract of this paper appears in the Proceedings of Crypto 2001, volume 2139 of Lecture Notes in Computer Science, pages 213–229, Springer-Verlag, 2001.*

# 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 a 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 [41] 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@company.com` she simply encrypts her message using the public key string "`bob@company.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 [11, 45, 44, 31, 25] (see also [33, 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 until the results in [5] constructing a usable IBE system was an open problem. Interestingly, the related notions of identity-based signature and authentication schemes, also introduced by Shamir [41], do have satisfactory solutions [15, 14].

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. Based on

*Supported by DARPA contract F30602-99-1-0530, NSF, and the Packard Foundation.

†Supported by an NSF Career Award and the Packard Foundation.

this assumption we show that the new system has chosen ciphertext security in the random oracle model. Using standard techniques from threshold cryptography [20, 22] 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 [32, 17]. Joux [26] recently showed that the Weil pairing can be used for "good" by using it for a protocol for three party one round Diffie-Hellman key exchange. Sakai et al. [40] used the pairing for key exchange and Verheul [46] used it to construct 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 [37, 2]. First, we allow the attacker to attack an arbitrary public key ID of her choice. Second, while mounting a chosen ciphertext attack on ID we allow the attacker to obtain from the PKG the private key for any public key of her choice, other than the private key for ID. This models an attacker who obtains a number of private keys corresponding to some identities of her choice and then tries to attack some other public key ID of her choice. 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. Section 8 gives conclusions and some open problems. The Appendix contains a more detailed discussion of the Weil pairing.

## 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@company.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 private key.

One could potentially make this approach more granular by encrypting e-mail for Bob using "`bob@company.com` $\|$ `current-date`". This forces Bob to obtain a new private key every day.

This might be possible in a corporate PKI where the PKG is maintained by the corporation. With this approach key revocation is very 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. As a result, Bob can no longer read his email. The interesting property is that Alice does not need to communicate with any third party certificate directory to obtain Bob's daily public key. Hence, identity based encryption is a very efficient mechanism for implementing ephemeral public keys. Also note that 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 [38, 12] for methods of sending messages into the future using a stronger security model).

**Managing user credentials.** A simple extension to the discussion above enables us to manage user credentials using the IBE system. Suppose Alice encrypts mail to Bob using the public key: "`bob@company.com` ‖ `current-year` ‖ `clearance=secret`". Then Bob will only be able to read the email if on the specified date he has secret clearance. Consequently, it is easy to grant and revoke user credentials using the PKG.

### 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. [23].
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 security 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 $1, \ldots, n$ as distinct public keys.

## 2 Definitions

**Identity-Based Encryption.** An identity-based encryption scheme $\mathcal{E}$ 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, master-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 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 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 (IND-CCA) is the standard acceptable notion of security for a public key encryption scheme [37, 2, 13]. 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 $\mathsf{ID}_i$ 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 $\mathcal{E}$ is semantically secure against an adaptive chosen ciphertext attack (IND-ID-CCA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following IND-ID-CCA 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 master-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 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 Extract to generate the private key $d_i$ corresponding to $\mathsf{ID}_i$. It then runs algorithm 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 $\mathsf{ID}$ on which it wishes to be challenged. The only constraint is that $\mathsf{ID}$ 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 $b' \in \{0,1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking the scheme $\mathcal{E}$ as the following function of the security parameter $k$ ($k$ is given as input to the challenger): $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}(k) = \left|\Pr[b = b'] - \frac{1}{2}\right|$.
The probability is over the random bits used by the challenger and the adversary.

Using the IND-ID-CCA game we can define chosen ciphertext security for IBE schemes. As usual, we say that a function $g : \mathbb{R} \to \mathbb{R}$ is *negligible* if for any $d > 0$ we have $|g(k)| < 1/k^d$ for sufficiently large $k$.

**Definition 2.1.** *We say that the IBE system $\mathcal{E}$ is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time* IND-ID-CCA *adversary $\mathcal{A}$ the function $Adv_{\mathcal{E},\mathcal{A}}(k)$ is negligible. As shorthand, we say that $\mathcal{E}$ is* IND-ID-CCA *secure.*

Note that the standard definition of chosen ciphertext security (IND-CCA) [37, 2] 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 [7]. After all, our definition involves multiple public keys belonging to multiple users. In [7] the authors show that that multiuser IND-CCA is reducible to single user IND-CCA using a standard hybrid argument. This does not hold in the identity-based settings, IND-ID-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.

**Semantically secure identity based encryption.** The proof of security for our IBE system makes use of a weaker notion of security known as semantic security (also known as semantic security against a chosen plaintext attack) [24, 2]. Semantic security is similar to chosen ciphertext security (IND-ID-CCA) except that the adversary is more limited; it cannot issue decryption queries while attacking the challenge public key. For a standard public key system (not an identity based system) semantic security is defined using the following game: (1) the adversary is given a random public key generated by the challenger, (2) the adversary outputs two equal length messages $M_0$ and $M_1$ and receives the encryption of $M_b$ from the challenger where $b$ is chosen at random in $\{0,1\}$, (3) the adversary outputs $b'$ and wins the game if $b = b'$. The public key system is said to be semantically secure if no polynomial time adversary can win the game with a non-negligible advantage. As shorthand we say that a semantically secure public key system is IND-CPA secure. Semantic security captures our intuition that given a ciphertext the adversary learns nothing about the corresponding plaintext.

To define semantic security for identity based systems (denoted IND-ID-CPA) we strengthen the standard definition by allowing the adversary to issue chosen private key extraction queries. Similarly, the adversary is challenged on a public key ID of her choice. We define semantic security for identity based encryption schemes using an IND-ID-CPA game. The game is identical to the IND-ID-CCA game defined above except that the adversary cannot make any decryption queries. The adversary can only make private key extraction queries. We say that an identity-based encryption scheme $\mathcal{E}$ is semantically secure (IND-ID-CPA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following IND-ID-CPA 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 master-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 Extract to generate the private key $d_i$ corresponding to the public key $\mathsf{ID}_i$. 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 two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and a public key 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 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 $b' \in \{0,1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CPA adversary. As we did above, the advantage of an IND-ID-CPA adversary $\mathcal{A}$ against the scheme $\mathcal{E}$ is the following function of the security parameter $k$: $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}(k) = \left|\Pr[b = b'] - \frac{1}{2}\right|$.
The probability is over the random bits used by the challenger and the adversary.

**Definition 2.2.** *We say that the IBE system $\mathcal{E}$ is semantically secure if for any polynomial time* IND-ID-CPA *adversary $\mathcal{A}$ the function $Adv_{\mathcal{E},\mathcal{A}}(k)$ is negligible. As shorthand, we say that $\mathcal{E}$ is* IND-ID-CPA *secure.*

**One way identity-based encryption.** One can define an even weaker notion of security called one-way encryption (OWE) [16]. Roughly speaking, a public key encryption scheme is a one-way encryption if given the encryption of a random plaintext the adversary cannot produce the plaintext in its entirety. One way encryption is a weak notion of security since there is nothing preventing the adversary from, say, learning half the bits of the plaintext. Hence, one-way encryption schemes do not generally provide secure encryption. In the random oracle model one-way encryption schemes can be used for encrypting session-keys (the session-key is taken to be the hash of the plaintext). We note that one can extend the notion of one-way encryption to identity based systems by adding private key extraction queries to the definition. We do not give the full definition here since in this paper we use semantic security as the weakest notion of security. See [5] for the full definition of identity based one-way encryption, and its use as part of an alternative proof strategy for our main result.

**Random oracle model.** To analyze the security of certain natural cryptographic constructions Bellare and Rogaway introduced an idealized security model called the random oracle model [3]. Roughly

speaking, a random oracle is a function $H : X \to Y$ chosen uniformly at random from the set of all functions $\{h : X \to Y\}$ (we assume $Y$ is a finite set). An algorithm can query the random oracle at any point $x \in X$ and receive the value $H(x)$ in response. Random oracles are used to model cryptographic hash functions such as SHA-1. Note that security in the random oracle model does not imply security in the real world. Nevertheless, the random oracle model is a useful tool for validating natural cryptographic constructions. Security proofs in this model prove security against attackers that are confined to the random oracle world.

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

## 3 Bilinear maps and the Bilinear Diffie-Hellman Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two 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 $P, Q \in \mathbb{G}_1$.

A bilinear map satisfying the three properties above is said to be an *admissible* bilinear map. In Section 5 we give a concrete example of groups $\mathbb{G}_1, \mathbb{G}_2$ and an admissible bilinear map between them. The group $\mathbb{G}_1$ is a subgroup of the additive group of points of an elliptic curve $E/\mathbb{F}_p$. The group $\mathbb{G}_2$ is a subgroup of 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 can be used to construct an admissible bilinear map between these two groups.

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 [32] 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) [4] 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 [28] 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

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

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 [28] 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.

### 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).

**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 an admissible 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 $\epsilon$ 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 $a, b, c$ 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 $\mathcal{G}$ is a *BDH parameter generator* if (1) $\mathcal{G}$ takes a security parameter $k \in \mathbb{Z}^+$, (2) $\mathcal{G}$ runs in polynomial time in $k$, and (3) $\mathcal{G}$ outputs a prime number $q$, the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ of order $q$, and the description of an admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. We denote the output of $\mathcal{G}$ by $\mathcal{G}(1^k) = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$. The security parameter $k$ is used to determine the size of $q$; for example, one could take $q$ to be a random $k$-bit prime. For $i = 1, 2$ we assume that the description of the group $\mathbb{G}_i$ contains polynomial time (in $k$) algorithms for computing the group action in $\mathbb{G}_i$ and contains a generator of $\mathbb{G}_i$. The generator of $\mathbb{G}_i$ enables us to generate uniformly random elements in $\mathbb{G}_i$. Similarly, we assume that the description of $\hat{e}$ contains a polynomial time algorithm for computing $\hat{e}$. We give an example of a BDH parameter generator in Section 5.1.

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

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

We say that $\mathcal{G}$ satisfies the BDH assumption if for any randomized polynomial time (in $k$) algorithm $\mathcal{A}$ we have that $\mathrm{Adv}_{\mathcal{G},\mathcal{A}}(k)$ is a negligible function. When $\mathcal{G}$ satisfies the BDH assumption we say that BDH is hard in groups generated by $\mathcal{G}$.

In Section 5.1 we give some examples of BDH parameter generators that are believed to satisfy the BDH assumption. We note that Joux [26] (implicitly) used the BDH assumption to construct a one-round three party Diffie-Hellman protocol. The BDH assumption is also needed for constructions in [46, 40].

**Hardness of BDH.** 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 $\mathbb{G}_1$ or $\mathbb{G}_2$. In other words, an algorithm for CDH in $\mathbb{G}_1$ or $\mathbb{G}_2$ 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 $\mathbb{G}_1$ or in $\mathbb{G}_2$? We refer to a survey by Joux [27] for a more detailed analysis of the relationship between BDH and other standard problems.

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 isomorphism $f_Q : \mathbb{G}_1 \to \mathbb{G}_2$ by $f_Q(P) = \hat{e}(P, Q)$. If any one of these isomorphisms turns out to be invertible then BDH is easy in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. Fortunately, an efficient algorithm for inverting $f_Q$ for some fixed $Q$ would imply an 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 $\mathcal{G}$ 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 $\mathcal{G}$ be some BDH parameter generator.

**Setup:** Given a security parameter $k \in \mathbb{Z}^+$, the algorithm works as follows:

Step 1: Run $\mathcal{G}$ on input $k$ to generate a prime $q$, two groups $\mathbb{G}_1, \mathbb{G}_2$ of order $q$, and an admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Choose a random 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 $\mathsf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2 \rangle$. 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 $d_{\mathsf{ID}}$ 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 $\mathsf{ID}$ 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: $g_{\mathsf{ID}}^r$.
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. Performance considerations of BasicIdent are discussed in Section 5. Note that the value of $g_{\mathsf{ID}}$ in Algorithm Encrypt is independent of the message to be encrypted. Hence there is no need to recompute $g_{\mathsf{ID}}$ on subsequent encryptions to the same public key ID.

**Security.** Next, we study the security of this basic scheme. The following theorem shows that BasicIdent is a semantically secure identity based encryption scheme (IND-ID-CPA) assuming BDH is hard in groups generated by $\mathcal{G}$.

**Theorem 4.1.** *Suppose the hash functions $H_1, H_2$ are random oracles. Then* BasicIdent *is a semantically secure identity based encryption scheme (*IND-ID-CPA*) assuming BDH is hard in groups generated by $\mathcal{G}$. Concretely, suppose there is an* IND-ID-CPA *adversary $\mathcal{A}$ that has advantage $\epsilon(k)$ against the scheme* BasicIdent*. Suppose $\mathcal{A}$ makes 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 $\mathcal{G}$ with advantage at least:*

$$Adv_{\mathcal{G},\mathcal{B}}(k) \geq \frac{2\epsilon(k)}{e(1+q_E) \cdot q_{H_2}}$$

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

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

**keygen:** Given a security parameter $k \in \mathbb{Z}^+$, the algorithm works as follows:

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

Step 2: Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$. Pick a random $Q_{\mathsf{ID}} \in \mathbb{G}_1^*$.

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

Step 4: The public key is $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$. 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 $C = \langle U, V \rangle$ be a ciphertext created using the public key $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$. 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 IND-ID-CPA attack on BasicIdent can be converted to a IND-CPA attack on BasicPub. This step shows that private key extraction queries do not help the adversary. We then show that BasicPub is IND-CPA secure if the BDH assumption holds.

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

**Proof.** We show how to construct an IND-CPA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\epsilon/e(1+q_E)$ against 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 BasicPub. The result is a public key $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2\rangle$ and a private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. As usual, $q$ is the order of $\mathbb{G}_1, \mathbb{G}_2$. The challenger gives $K_{pub}$ to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ is supposed to output two messages $M_0$ and $M_1$ and expects to receive back the BasicPub encryption of $M_b$ under $K_{pub}$ where $b \in \{0,1\}$. Then algorithm $\mathcal{B}$ outputs its guess $b' \in \{0,1\}$ for $b$.

Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in an IND-ID-CPA game as follows ($\mathcal{B}$ simulates the challenger for $\mathcal{A}$):

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

**$H_1$-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 $\mathsf{ID}_i$ algorithm $\mathcal{B}$ responds as follows:

1. If the query $\mathsf{ID}_i$ 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 $\delta$ that will be determined later.
3. Algorithm $\mathcal{B}$ picks a random $b \in \mathbb{Z}_q^*$.
   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 $\mathsf{ID}_i$ 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 BasicPub failed.
2. We know $coin_i = 0$ and hence $Q_i = b_iP$. Define $d_i = b_iP_{pub} \in \mathbb{G}_1^*$. Observe that $d_i = sQ_i$ and therefore $d_i$ is the private key associated to the public key $\mathsf{ID}_i$. Give $d_i$ to algorithm $\mathcal{A}$.

**Challenge:** Once algorithm $\mathcal{A}$ decides that Phase 1 is over it outputs a public key $\mathsf{ID}_{ch}$ 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 its challenger the messages $M_0, M_1$. The challenger responds with a BasicPub ciphertext $C = \langle U, V\rangle$ 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 $H_1(\mathsf{ID}_{ch}) =$

$Q$. Let $\langle \mathsf{ID}_{ch}, Q, b, coin \rangle$ be the corresponding tuple on the $H_1^{list}$. If $coin = 0$ then $\mathcal{B}$ reports failure and terminates. The attack on BasicPub failed.

3. We know $coin = 1$ and therefore $Q = bQ_{\mathsf{ID}}$. Recall that when $C = \langle U, V \rangle$ we have $U \in \mathbb{G}_1^*$. 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 ciphertext $C'$. Note that $C'$ is a proper BasicIdent encryption of $M_c$ under the public key $\mathsf{ID}_{ch}$ as required. To see this first observe that, since $H_1(\mathsf{ID}_{ch}) = Q$, the private key corresponding to $\mathsf{ID}_{ch}$ is $d_{ch} = sQ$. Second, observe that

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

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

**Phase 2:** Algorithm $\mathcal{B}$ responds to private key extraction queries as in Phase 1.

**Guess:** Eventually algorithm $\mathcal{A}$ outputs 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 are valid. Finally, the challenge ciphertext $C'$ given to $\mathcal{A}$ is the BasicIdent encryption of $M_c$ for some random $c \in \{0, 1\}$. Therefore, by definition of algorithm $\mathcal{A}$ we have that $|\Pr[c = c'] - \frac{1}{2}| \geq \epsilon$. □

To complete the proof of Lemma 4.2 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 $\delta^{q_E}$. 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 $\delta_{opt}$, 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 [9]. Next, we show that BasicPub is a semantically secure public key system if the BDH assumption holds.

**Lemma 4.3.** *Let $H_2$ be a random oracle from $\mathbb{G}_2$ to $\{0, 1\}^n$. Let $\mathcal{A}$ be an* IND-CPA *adversary that has advantage $\epsilon(k)$ 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 $\mathcal{G}$ with advantage at least $2\epsilon(k)/q_{H_2}$ and a running time $O(time(\mathcal{A}))$.*

**Proof.** Algorithm $\mathcal{B}$ is given as input the BDH parameters $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ produced by $\mathcal{G}$ 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 $a, b, c$ 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:

**Setup:** Algorithm $\mathcal{B}$ creates the BasicPub public key $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$ by setting $P_{pub} = P_1$ and $Q_{\mathsf{ID}} = P_2$. Here $H_2$ is a random oracle controlled by $\mathcal{B}$ as described below. Algorithm $\mathcal{B}$ gives $\mathcal{A}$ the BasicPub public key $K_{pub}$. Observe that the (unknown) private key associated to $K_{pub}$ is $d_{\mathsf{ID}} = aQ_{\mathsf{ID}} = abP$.

**$H_2$-queries:** At any time algorithm $\mathcal{A}$ may issue queries to the random oracle $H_2$. 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 $\langle X_i, H_i \rangle$ then respond with $H_2(X_i) = H_i$.
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 $H_2(X_i) = H_i$.

**Challenge:** Algorithm $\mathcal{A}$ outputs two messages $M_0, M_1$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ 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 $C$ as the challenge to $\mathcal{A}$. Observe that, by definition, the decryption of $C$ is $R \oplus H_2(\hat{e}(P_3, d_{\mathsf{ID}})) = R \oplus H_2(D)$.

**Guess:** Algorithm $\mathcal{A}$ outputs its guess $c' \in \{0,1\}$. 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.

Algorithm $\mathcal{B}$ is simulating a real attack environment for algorithm $\mathcal{A}$ (it simulates the challenger and the oracle for $H_2$). We show that algorithm $\mathcal{B}$ outputs the correct answer $D$ with probability at least $2\epsilon/q_{H_2}$ as required. The proof is based on comparing $\mathcal{A}$'s behavior in the simulation to its behavior in a real IND-CPA attack game (against a real challenger and a real random oracle for $H_2$).

Let $\mathcal{H}$ be the event that algorithm $\mathcal{A}$ issues a query for $H_2(D)$ at some point during the simulation above (this implies that at the end of the simulation $D$ appears in some tuple on the $H_2^{list}$). We show that $\Pr[\mathcal{H}] \geq 2\epsilon$. This will prove that algorithm $\mathcal{B}$ outputs $D$ with probability at least $2\epsilon/q_{H_2}$. We also study event $\mathcal{H}$ in the real attack game, namely the event that $\mathcal{A}$ issues a query for $H_2(D)$ when communicating with a real challenger and a real random oracle for $H_2$.

**Claim 1:** $\Pr[\mathcal{H}]$ in the simulation above is equal to $\Pr[\mathcal{H}]$ in the real attack.

Proof of claim. Let $\mathcal{H}_\ell$ be the event that $\mathcal{A}$ makes a query for $H_2(D)$ in one of its first $\ell$ queries to the $H_2$ oracle. We prove by induction on $\ell$ that $\Pr[\mathcal{H}_\ell]$ in the real attack is equal to $\Pr[\mathcal{H}_\ell]$ in the simulation for all $\ell \geq 0$. Clearly $\Pr[\mathcal{H}_0] = 0$ in both the simulation and in the real attack. Now suppose that for some $\ell > 0$ we have that $\Pr[\mathcal{H}_{\ell-1}]$ in the simulation is equal to $\Pr[\mathcal{H}_{\ell-1}]$ in the real attack. We show that the same holds for $\mathcal{H}_\ell$. We know that:

$$\begin{aligned} \Pr[\mathcal{H}_\ell] &= \Pr[\mathcal{H}_\ell \,|\, \mathcal{H}_{\ell-1}] \Pr[\mathcal{H}_{\ell-1}] + \Pr[\mathcal{H}_\ell \,|\, \neg\mathcal{H}_{\ell-1}] \Pr[\neg\mathcal{H}_{\ell-1}] \\ &= \Pr[\mathcal{H}_{\ell-1}] + \Pr[\mathcal{H}_\ell \,|\, \neg\mathcal{H}_{\ell-1}] \Pr[\neg\mathcal{H}_{\ell-1}] \end{aligned} \tag{1}$$

We argue that $\Pr[\mathcal{H}_\ell \,|\, \neg\mathcal{H}_{\ell-1}]$ in the simulation is equal to $\Pr[\mathcal{H}_\ell \,|\, \neg\mathcal{H}_{\ell-1}]$ in the real attack. To see this observe that as long as $\mathcal{A}$ does not issue a query for $H_2(D)$ its view during the simulation is identical to its view in the real attack (against a real challenger and a real random oracle for $H_2$). Indeed, the public-key and the challenge are distributed as in the real attack. Similarly, all responses to $H_2$-queries are uniform and independent in $\{0,1\}^n$. Therefore, $\Pr[\mathcal{H}_\ell \,|\, \neg\mathcal{H}_{\ell-1}]$ in the simulation is equal to $\Pr[\mathcal{H}_\ell \,|\, \neg\mathcal{H}_{\ell-1}]$ in the real attack. It follows by (1) and the inductive hypothesis that $\Pr[\mathcal{H}_\ell]$ in the real attack is equal to $\Pr[\mathcal{H}_\ell]$ in the simulation. By induction on $\ell$ we obtain that $\Pr[\mathcal{H}]$ in the real attack is equal to $\Pr[\mathcal{H}]$ in the simulation. □

**Claim 2:** In the real attack we have $\Pr[\mathcal{H}] \geq 2\epsilon$.

Proof of claim. In the real attack, if $\mathcal{A}$ never issues a query for $H_2(D)$ then the decryption of $C$ is independent of $\mathcal{A}$'s view (since $H_2(D)$ is independent of $\mathcal{A}$'s view). Therefore, in the real attack $\Pr[c = c' \mid \neg\mathcal{H}] = 1/2$. By definition of $\mathcal{A}$, we know that in the real attack $|\Pr[c = c'] - 1/2| \geq \epsilon$.

We show that these two facts imply that $\Pr[\mathcal{H}] \geq 2\epsilon$. To do so we first derive simple upper and lower bounds on $\Pr[c = c']$:

$$\begin{aligned}
\Pr[c = c'] &= \Pr[c = c' | \neg\mathcal{H}] \Pr[\neg\mathcal{H}] + \Pr[c = c' | \mathcal{H}] \Pr[\mathcal{H}] \leq \\
&\leq \Pr[c = c' | \neg\mathcal{H}] \Pr[\neg\mathcal{H}] + \Pr[\mathcal{H}] = \frac{1}{2} \Pr[\neg\mathcal{H}] + \Pr[\mathcal{H}] = \frac{1}{2} + \frac{1}{2} \Pr[\mathcal{H}] \\
\Pr[c = c'] &\geq \Pr[c = c' | \neg\mathcal{H}] \Pr[\neg\mathcal{H}] = \frac{1}{2} - \frac{1}{2} \Pr[\mathcal{H}]
\end{aligned}$$

It follows that $\epsilon \leq |\Pr[c = c'] - 1/2| \leq \frac{1}{2} \Pr[\mathcal{H}]$. Therefore, in the real attack $\Pr[\mathcal{H}] \geq 2\epsilon$. □

To complete the proof of Lemma 4.3 observe that by Claims 1 and 2 we know that $\Pr[\mathcal{H}] \geq 2\epsilon$ in the simulation above. Hence, at the end of the simulation, $D$ appears in some tuple on the $H_2^{list}$ with probability at least $2\epsilon$. It follows that $\mathcal{B}$ produces the correct answer with probability at least $2\epsilon/q_{H_2}$ as required. □

We note that one can slightly vary the reduction in the proof above to obtain different bounds. For example, in the 'Guess' step above one can avoid having to pick a random element from the $H_2^{list}$ by using the random self reduction of the BDH problem. This requires running algorithm $\mathcal{A}$ multiple times (as in Theorem 7 of [42]). The success probability for solving the given BDH problem increases at the cost of also increasing the running time.

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

## 4.2 Identity-Based Encryption with Chosen Ciphertext Security

We use a technique due to Fujisaki-Okamoto [16] 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:

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

Here $\sigma$ 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 note that semantic security implies one-way encryption and hence the Fujisaki-Okamoto result also applies if $\mathcal{E}$ is semantically secure (IND-CPA).

We apply the Fujisaki-Okamoto transformation to BasicIdent and show that the resulting IBE system is IND-ID-CCA secure. 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 $\mathsf{ID}$ 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 $\mathsf{ID}$. 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 $\mathsf{FullIdent}$. 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 [16]).

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

**Theorem 4.4.** *Let the hash functions $H_1, H_2, H_3, H_4$ be random oracles. Then* $\mathsf{FullIdent}$ *is a chosen ciphertext secure IBE (*$\mathsf{IND\text{-}ID\text{-}CCA}$*) assuming BDH is hard in groups generated by $\mathcal{G}$.*
*Concretely, suppose there is an* $\mathsf{IND\text{-}ID\text{-}CCA}$ *adversary $\mathcal{A}$ that has advantage $\epsilon(k)$ against the scheme* $\mathsf{FullIdent}$ *and $\mathcal{A}$ runs in time at most $t(k)$. 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 $\mathcal{G}$ with running time $t_1(k)$ where:*

$$\begin{aligned} Adv_{\mathcal{G},\mathcal{B}}(k) &\geq 2FO_{adv}(\tfrac{\epsilon(k)}{e(1+q_E+q_D)},\ q_{H_4},\ q_{H_3},\ q_D)/q_{H_2} \\ t_1(k) &\leq FO_{time}(t(k), q_{H_4}, q_{H_3}) \end{aligned}$$

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

The proof of Theorem 4.4 is based on the following result of Fujisaki and Okamoto (Theorem 14 in [16]). Let $\mathsf{BasicPub}^{hy}$ be the result of applying the Fujisaki-Okamoto transformation to $\mathsf{BasicPub}$.

**Theorem 4.5 (Fujisaki-Okamoto).** *Suppose $\mathcal{A}$ is an* $\mathsf{IND\text{-}CCA}$ *adversary that achieves advantage $\epsilon(k)$ when attacking* $\mathsf{BasicPub}^{hy}$*. Suppose $\mathcal{A}$ has running time $t(k)$, makes at most $q_D$ decryption queries, and makes at most $q_{H_3}, q_{H_4}$ queries to the hash functions $H_3, H_4$ respectively. Then there is an* $\mathsf{IND\text{-}CPA}$ *adversary $\mathcal{B}$ against* $\mathsf{BasicPub}$ *with running time $t_1(k)$ and advantage $\epsilon_1(k)$ where*

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

*Here $q$ is the size of the groups $\mathbb{G}_1, \mathbb{G}_2$ and $n$ is the length of $\sigma$.*

In fact, Fujisaki-Okamoto prove a stronger result: Under the hypothesis of Theorem 4.5, $\mathsf{BasicPub}^{hy}$ would not even be a one-way encryption scheme. For our purposes the result in Theorem 4.5 is sufficient. To prove Theorem 4.4 we also need the following lemma to translate between an $\mathsf{IND\text{-}ID\text{-}CCA}$ chosen ciphertext attack on $\mathsf{FullIdent}$ and an $\mathsf{IND\text{-}CCA}$ chosen ciphertext attack on $\mathsf{BasicPub}^{hy}$.

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

**Proof.** 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 $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2, H_3, H_4\rangle$ 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 $\mathsf{ID}_i$. 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 $\mathsf{ID}_i$. Then the FullIdent decryption of $C_i$ using $d_i$ is the same as the $\mathsf{BasicPub}^{hy}$ decryption of $C_i'$ using $d_{\mathsf{ID}}$. To see this observe that:

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

   - 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 $\mathsf{ID}_{ch}$ 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 $H_1(\mathsf{ID}_{ch}) = Q$. Let $\langle \mathsf{ID}_{ch}, Q, b, coin\rangle$ 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 $b$ mod $q$. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the challenge $C'$. Note that, as in the proof of Lemma 4.2, $C'$ is a FullIdent encryption of $M_c$ under the public key $\mathsf{ID}_{ch}$ 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 $\mathcal{A}$ outputs 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 $\mathsf{ID}_{ch}$ 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 $\mathsf{ID}_{ch}$ 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 $\mathcal{E}_1 \vee \mathcal{E}_3$ happens after $\mathcal{A}$ issues at most $i$ queries. Similarly, let $\mathcal{E}^i$ be the event that $\mathcal{E}_1 \vee \mathcal{E}_3$ happens for the first time when $\mathcal{A}$ issues the $i$'th query. We prove by induction on $i$ that $\Pr[\neg\mathcal{E}^{0\ldots i} \mid \neg\mathcal{E}_2] \geq \delta^i$. The claim follows because $\Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2 \wedge \neg\mathcal{E}_3] = \Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_3 \mid \neg\mathcal{E}_2] \Pr[\neg\mathcal{E}_2] \geq \Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_3 \mid \neg\mathcal{E}_2](1-\delta)$.

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

$$\begin{aligned}\Pr[\neg\mathcal{E}^{0\ldots i} \mid \neg\mathcal{E}_2] &= \Pr[\neg\mathcal{E}^{0\ldots i} \mid \neg\mathcal{E}^{0\ldots i-1} \wedge \neg\mathcal{E}_2] \Pr[\neg\mathcal{E}^{0\ldots i-1} \mid \neg\mathcal{E}_2] \\ &= \Pr[\neg\mathcal{E}^{i} \mid \neg\mathcal{E}^{0\ldots i-1} \wedge \neg\mathcal{E}_2] \Pr[\neg\mathcal{E}^{0\ldots i-1} \mid \neg\mathcal{E}_2] \geq \Pr[\neg\mathcal{E}^{i} \mid \neg\mathcal{E}^{0\ldots i-1} \wedge \neg\mathcal{E}_2]\delta^{i-1}\end{aligned}$$

Hence, it suffices to bound $q_i = \Pr[\neg\mathcal{E}^i \mid \neg\mathcal{E}^{0\ldots i-1} \wedge \neg\mathcal{E}_2]$. 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, and given that $\mathcal{E}_2$ does not occur. 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 $\mathsf{ID}_i$ is not equal to the public key $\mathsf{ID}_{ch}$ on which $\mathcal{A}$ is being challenged.

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

Case 2. The public key $\mathsf{ID}_i$ 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 $\mathsf{ID}_i$ 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 $\mathsf{ID}_i$ is equal to the public key $\mathsf{ID}_{ch}$ 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, we have that $\Pr[\neg\mathcal{E}^{0\ldots i} \mid \neg\mathcal{E}_2] \geq \delta^i$ 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 $\delta$. 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 $\delta_{opt}$, 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. □

**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 an IND-CPA adversary on BasicPub. By Lemma 4.3 an IND-CPA adversary on BasicPub implies an algorithm for BDH. Composing all these reductions gives the required bounds. □

### 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. $\ell$-to-1: For any $y \in \mathbb{G}_1^*$ the preimage of $y$ under $L$ has size exactly $\ell$. 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 uniform 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 $\epsilon(k)$. 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 $\epsilon(k)$ 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 from $\mathcal{A}$ 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 $\mathsf{ID}_i$ algorithm $\mathcal{B}$ responds as follows:

1. If the query $\mathsf{ID}_i$ already appears on the $(H_1')^{list}$ in a tuple $\langle \mathsf{ID}_i, y_i \rangle$, 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 $\mathcal{L}_S(\alpha)$ 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 distributed in $A$ as required since $\alpha$ is uniformly distributed in $\mathbb{G}_1^*$ and $L$ is an $\ell$-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 $\epsilon(k)$ in winning the game with the challenger. □

# 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 (see the Appendix for more details).

## 5.1 Properties of the Weil Pairing

Let $p$ be a prime satisfying $p = 2 \bmod 3$ and let $q > 3$ be some prime factor of $p + 1$. 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 $E$ (see [43] for more information). From here on we let $E(\mathbb{F}_{p^r})$ denote the group of points on $E$ defined over $\mathbb{F}_{p^r}$.

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

**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 build a simple admissible encoding function.

**Fact 3:** Let $1 \neq \zeta \in \mathbb{F}_{p^2}$ be a solution of $x^3 - 1 = 0$ in $\mathbb{F}_{p^2}$. Then the map $\phi(x, y) = (\zeta x, y)$ is an automorphism of the group of points on the curve $E$. Note that for any point $Q = (x, y) \in E(\mathbb{F}_p)$

we have that $\phi(Q) \in E(\mathbb{F}_{p^2})$, but $\phi(Q) \notin E(\mathbb{F}_p)$. Hence, $Q \in E(\mathbb{F}_p)$ is linearly independent of $\phi(Q) \in E(\mathbb{F}_{p^2})$.

**Fact 4:** Since the points $P \in \mathbb{G}_1$ 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 $\mathbb{F}^*_{p^2}$ of order $q$. 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 the Appendix. For any $Q, R \in E(\mathbb{F}_p)$ the Weil pairing satisfies $e(Q, R) = 1$. In other words, the Weil pairing is degenerate on $E(\mathbb{F}_p)$, and hence degenerate on the group $\mathbb{G}_1$. To get a non-degenerate map we define the modified Weil pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ as follows:

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

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 the Appendix. Its running time is comparable to exponentiation in $\mathbb{F}_p$.

Joux and Nguyen [28] 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 $\mathcal{G}_1$:** Given a security parameter $2 < k \in \mathbb{Z}$ the BDH parameter generator picks a random $k$-bit prime $q$ and finds the smallest prime $p$ such that (1) $p = 2 \bmod 3$, (2) $q$ divides $p + 1$, and (3) $q^2$ does not divide $p + 1$. We write $p = \ell q + 1$. 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.

The BDH parameter generator $\mathcal{G}_1$ is believed to satisfy the BDH assumption asymptotically. However, there is still the question of what values of $p$ and $q$ can be used in practice to make the BDH problem sufficiently hard. At the very least, we must ensure that the discrete log problem in $\mathbb{G}_1$ is sufficiently hard. 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 [32, 17]). Hence, computing discrete log in $\mathbb{F}^*_{p^2}$ is sufficient for computing discrete log in $\mathbb{G}_1$. In practice, for proper security of discrete log in $\mathbb{F}^*_{p^2}$ one often uses primes $p$ that are at least 512-bits long (so that the group size is at least 1024-bits long). Consequently, one should not use this BDH parameter generator with primes $p$ that are less than 512-bits long.

### 5.2 An admissible encoding function: MapToPoint

Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups generated by $\mathcal{G}_1$ 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 $p = \ell q - 1$ for some prime $q > 3$. We require that $q$ does not divide $\ell$ (i.e. that $q^2$ does not divide $p+1$). 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 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 $Q_{\mathsf{ID}} = \ell Q \in \mathbb{G}_1$.
3. Output $\mathsf{MapToPoint}(y_0) = Q_{\mathsf{ID}}$.

This completes the description of MapToPoint.

We note that there are $\ell - 1$ values of $y_0 \in \mathbb{F}_p$ for which $\ell Q = \ell(x_0, y_0) = O$ (these are the non-$O$ points of order dividing $\ell$). Let $B \subset \mathbb{F}_p$ be the set of these $y_0$. When $H_1(\mathsf{ID})$ is one of these $\ell - 1$ values $Q_{\mathsf{ID}}$ is the identity element of $\mathbb{G}_1$. It is extremely unlikely for $H_1(\mathsf{ID})$ to hit one of these points – the probability is $1/q < 1/2^k$. Hence, for simplicity we say that $H_1(\mathsf{ID})$ only outputs elements in $\mathbb{F}_p \setminus B$, i.e. $H_1 : \{0,1\}^* \to \mathbb{F}_p \setminus B$. Algorithm MapToPoint can be easily extended to handle the values $y_0 \in B$ by hashing ID multiple times using different hash functions.

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

**Proof.** The map is clearly computable and is a $\ell - to - 1$ 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 $b \in \{0, \ldots, \ell - 1\}$, (2) compute $Q' = \ell^{-1} \cdot Q + bqP = (x, y)$, and (3) output $\mathcal{L}_S(Q) = y \in \mathbb{F}_p$. Here $\ell^{-1}$ is the inverse of $\ell$ in $\mathbb{Z}_q^*$. This algorithm outputs a random element from the $\ell$ elements in $\mathsf{MapToPoint}^{-1}(Q)$ as required. □

### 5.3 A concrete IBE system

Using FullIdent' from Section 4.3 with the BDH parameter generator $\mathcal{G}_1$ and the admissible encoding function MapToPoint we obtain a concrete IBE system. Note that in this system, $H_1$ is a hash function from $\{0,1\}^*$ to $\mathbb{F}_p$ (where $p$ is the finite field output by $\mathcal{G}_1$). The security of the system follows directly from Theorem 4.4 and Theorem 4.7. We summarize this in the following corollary.

**Corollary 5.2.** *The IBE system* FullIdent' *using the BDH parameter generator $\mathcal{G}_1$ and the admissible encoding* MapToPoint *is a chosen ciphertext secure IBE (i.e.* IND-ID-CCA *in the random oracle model) assuming $\mathcal{G}_1$ satisfies the BDH assumption.*

**Performance.** Algorithms Setup and Extract are very simple. 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 $Q_{\mathsf{ID}}$ and $P_{pub}$. Note that this computation is independent of the message to be encrypted, and hence can be done once and for all. Once $g_{\mathsf{ID}}$ is computed the performance of the system is almost identical to standard ElGamal encryption. Decryption is a single Weil pairing computation. We note that the ciphertext length of BasicIdent using $\mathcal{G}_1$ is the same as in regular ElGamal encryption in $\mathbb{F}_p$.

## 6 Extensions and Observations

**Tate pairing and other curves.** Our IBE system works with 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. Many different curves, or more generally Abelian varieties, are believed to give rise to such maps. For example, one could use the curve $y^2 = x^3 + x$ over $\mathbb{F}_p$ with $p = 3 \bmod 4$ and its endomorphism $\phi : (x, y) \to (-x, iy)$ where $i^2 = -1$. As another example, Galbraith [18] suggests using supersingular

elliptic curves over a field of small characteristic to reduce the ciphertext size in our system. More general Abelian varieties are proposed by Rubin and Silverberg [39]. We 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 [19, 1].

**Asymmetric pairings.** Our IBE system can use slightly more general bilinear maps, namely maps of the form $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_2$ where $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_2$ are three groups of prime order $q$. Using the notation of Section 4.1 the only change to BasicIdent is that we take $P$ and $P_{pub}$ as elements in $\mathbb{G}_0$ and let $H_1$ be a hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$. Everything else remains the same. However, to make the proof of security go through (Lemma 4.2 in particular) we need a different complexity assumption which we call the co-BDH assumption: given random $P, aP, bP \in \mathbb{G}_0$ and $Q, aQ, cQ \in \mathbb{G}_1$ no polynomial time algorithm can compute $\hat{e}(P,Q)^{abc}$ with non-negligible probability. If one is willing to accept this assumption then we can avoid using supersingular curves and instead use elliptic curves over $\mathbb{F}_p$, $p > 3$ proposed by Miyaji et al. [35]. Curves $E/\mathbb{F}_p$ in this family are not supersingular and have the property that if $q$ divides $|E(F_p)|$ then $E[q] \subseteq E(F_{p^6})$ (recall that $E[q]$ is the group containing all point in $E$ of order dividing $q$). One way to use these curves is to set $\mathbb{G}_1$ to be a cyclic subgroup of $E(\mathbb{F}_p)$ of order $q$ and $\mathbb{G}_0$ to be a different cyclic subgroup of $E(\mathbb{F}_{p^6})$ of the same order $q$. The standard Weil or Tate pairings on $\mathbb{G}_0 \times \mathbb{G}_1$ can be used as the bilinear map $\hat{e}$. Note that hashing public keys onto $\mathbb{G}_1 \subseteq E(\mathbb{F}_p)$ is easily done. Alternatively, to reduce the ciphertext size (which contains an element from $\mathbb{G}_0$) one could take $\mathbb{G}_0$ as a subgroup of order $q$ of $E(\mathbb{F}_p)$ and $\mathbb{G}_1$ as a different subgroup of $E(\mathbb{F}_{p^6})$ of the same order. The question is how to hash public keys into $\mathbb{G}_1$. To do so, let $\mathrm{tr} : E(\mathbb{F}_{p^6}) \to E(\mathbb{F}_p)$ be the trace map on the curve and define $\mathbb{G}_1$ to be the subgroup of $E[q]$ containing all points $P$ whose trace is $O$, i.e., $\mathrm{tr}(P) = O$. Then given a hash function $H : \{0,1\}^* \to E[q]$ we can hash a public key ID into $\mathbb{G}_1$ by computing: $H_1(\mathsf{ID}) = 6H(\mathsf{ID}) - \mathrm{tr}(H(\mathsf{ID})) \in \mathbb{G}_1$. Finally, we note that by modifying the security proof appropriately one can take $\mathbb{G}_1 = E[q]$ (a non-cyclic group) and then avoid computing traces while hashing into $\mathbb{G}_1$ (see also [18]).

**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 [20]. 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 $Q_{\mathsf{ID}}$ 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 $Q_{priv}^{(i)} = s_i Q_{\mathsf{ID}}$. The user can then construct $Q_{priv}$ as $Q_{priv} = \sum \lambda_i Q_{priv}^{(i)}$ where the $\lambda_i$'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 $P_{pub}^{(i)} = s_i P$. During a key generation request the user can verify that the response from the $i$'th PKG is valid by testing that:

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

Thus, a misbehaving PKG will be immediately caught. There is no need for zero-knowledge proofs as in regular robust threshold schemes [21]. The PKG's master-key can be generated in a distributed fashion using the techniques of [22].

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 $\hat{e}(s_i Q_{\mathsf{ID}}, U)$.

**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 $\mathsf{ID}$ is converted to a group point by hashing $\mathsf{ID}$ 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 [6].

# 7 Escrow ElGamal encryption

In this section we show 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 [36]. Our ElGamal escrow system works as follows:

**Setup:** Let $\mathcal{G}$ be some BDH parameter generator. Given a security parameter $k \in \mathbb{Z}^+$, the algorithm works as follows:

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

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

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

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 $\mathsf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, Q, H \rangle$. 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 $P_{pub} = xP \in \mathbb{G}_1$. 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:

$$C = \langle rP,\ M \oplus H(g^r) \rangle \quad \text{where} \quad g = \hat{e}(P_{pub}, Q) \in \mathbb{G}_2$$

**Decrypt:** Let $C = \langle U, V \rangle$ be a ciphertext encrypted using $P_{pub}$. Then $U \in \mathbb{G}_1$. 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 that BDH is hard for groups generated by $\mathcal{G}$ 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 [46] 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. The trusted third party must maintain a database of all private keys given to it by the various users.

# 8 Summary and open problems

We defined chosen ciphertext security for identity-based systems and proposed a fully functional IBE system. The system 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 [30] can be used to prove that the BDH assumption is equivalent to the discrete log assumption on the curve for certain primes $p$.

Cocks [8] recently proposed another IBE system whose security is based on the difficulty of distinguishing quadratic residues from non-residues in the ring $\mathbb{Z}/N\mathbb{Z}$ where $N$ is an RSA modulus (i.e., a product of two large primes). Cocks' system is somewhat harder to use in practice that the IBE system in this paper. Cocks' system uses bit-by-bit encryption and consequently outputs long ciphertexts. Also, encryption/decryption is a bit slower than the system described in this paper. Nevertheless, it is encouraging to see that IBE systems can be built using very different complexity assumptions.

It is an open problem to build chosen ciphertext secure identity based systems that are secure in the standard computation model (rather than the random oracle model). One might hope to use the techniques of Cramer-Shoup [10] 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. Building a chosen ciphertext secure IBE (IND-ID-CCA) in the standard model is currently an open problem.

# Acknowledgments

The authors thank Moni Naor, Alice Silverberg, Ben Lynn, Steven Galbraith, Kenny Paterson, and Mike Scott for helpful discussions about this work.

# References

[1] P. Barreto, H. Kim, B. Lynn, M. Scott, "Efficient Algorithms for Pairing-based Cryptosystems", in *Advances in Cryptology – Crypto 2002*, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[2] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, "Relations among notions of security for public-key encryption schemes", in *Advances in Cryptology – Crypto '98*, Lecture Notes in Computer Science, Vol. 1462, Springer-Verlag, pp. 26–45, 1998.

[3] M. Bellare, P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", In ACM conference on Computers and Communication Security, pp. 62–73, 1993.

[4] D. Boneh, "The decision Diffie-Hellman problem", in *Proc. Third Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, Vol. 1423, Springer-Verlag, pp. 48–63, 1998.

[5] D. Boneh, M. Franklin, "Identity based encryption from the Weil pairing", extended abstract in *Advances in Cryptology – Crypto 2001*, Lecture Notes in Computer Science, Vol. 2139, Springer-Verlag, pp. 231–229, Aug. 2001. See also `http://eprint.iacr.org/2001/090/`

[6] D. Boneh, B. Lynn, H. Shacham, "Short signatures from the Weil pairing", in *Advances in Cryptology – AsiaCrypt 2001*, Lecture Notes in Computer Science, Vol. 2248, Springer-Verlag, pp. 514-532, 2001.

[7] M. Bellare, A. Boldyreva, S. Micali, "Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements", in *Advances in Cryptology – Eurocrypt 2000*, Lecture Notes in Computer Science, Vol. 1807, Springer-Verlag, pp. 259–274, 2000.

[8] C. Cocks, "An identity based encryption scheme based on quadratic residues", Eighth IMA International Conference on Cryptography and Coding, Dec. 2001, Royal Agricultural College, Cirencester, UK.

[9] J. Coron, "On the exact security of Full-Domain-Hash", in *Advances in Cryptology – Crypto 2000*, Lecture Notes in Computer Science, Vol. 1880, Springer-Verlag, pp. 229–235, 2000.

[10] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", in *Advances in Cryptology – Crypto '98*, Lecture Notes in Computer Science, Vol. 1462, Springer-Verlag, pp. 13–25, 1998.

[11] Y. Desmedt and J. Quisquater, "Public-key systems based on the difficulty of tampering", in *Advances in Cryptology – Crypto '86*, Lecture Notes in Computer Science, Vol. 263, Springer-Verlag, pp. 111–117, 1986.

[12] G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan, "Conditional Oblivious Transfer and Timed-Release Encryption", in *Advances in Cryptology – Eurocrypt '99*, Lecture Notes in Computer Science, Vol. 1592, pp. 74–89, 1999.

[13] D. Dolev, C. Dwork, M. Naor, "Non-malleable cryptography", *SIAM J. Computing*, Vol. 30(2), pp. 391–437, 2000.

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

[15] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", in *Advances in Cryptology – Crypto '86*, Lecture Notes in Computer Science, Vol. 263, Springer-Verlag, pp. 186–194, 1986.

[16] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes", in *Advances in Cryptology – Crypto '99*, Lecture Notes in Computer Science, Vol. 1666, Springer-Verlag, pp. 537–554, 1999.

[17] 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.

[18] S. Galbraith, "Supersingular curves in cryptography", in *Advances in Cryptology – AsiaCrypt 2001*, Lecture Notes in Computer Science, Vol. 2248, Springer-Verlag, pp. 495–513, 2001.

[19] S. Galbraith, K. Harrison, D. Soldera, "Implementing the Tate-pairing", in *Proc. Fifth Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, Springer-Verlag, 2002.

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

[21] R.Gennaro, S.Jarecki, H.Krawczyk, T.Rabin, "Robust and Efficient Sharing of RSA Functions", *J. Cryptology*, Vol. 13(2), pp. 273-300, 2000.

[22] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems", *Advances in Cryptology – Eurocrypt '99*, Lecture Notes in Computer Science, Vol. 1592, Springer-Verlag, pp. 295–310, 1999.

[23] O. Goldreich, B. Pfitzmann and R. Rivest, "Self-delegation with controlled propagation -or- What if you lose your laptop", in *Advances in Cryptology – Crypto '98*, Lecture Notes in Computer Science, Vol. 1462, Springer-Verlag, pp. 153–168, 1998.

[24] S. Goldwasser, S. Micali, "Probabilistic Encryption", *J. Computer and System Sciences*, vol. 28, pp. 270-299, 1984.

[25] D. Hühnlein, M. Jacobson, D. Weber, "Towards Practical Non-interactive Public Key Cryptosystems Using Non-maximal Imaginary Quadratic Orders", in *Selected Areas in Cryptography*, Lecture Notes in Computer Science, Vol. 2012, Springer-Verlag, pp. 275–287, 2000.

[26] A. Joux, "A one round protocol for tripartite Diffie-Hellman", *Proc. Fourth Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, Vol. 1838, Springer-Verlag, pp. 385–394, 2000.

[27] A. Joux, "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems", in *Proc. Fifth Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[28] A. Joux, K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups", J. Cryptology 16(4), pp. 239-247, 2003.

[29] S. Lang, *Elliptic functions*, Addison-Wesley, Reading, 1973.

[30] U. Maurer, "Towards proving the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms", in *Advances in Cryptology – Crypto '94*, Lecture Notes in Computer Science, Vol. 839, pp. 271–281, 1994.

[31] U. Maurer and Y. Yacobi, "Non-interactive public-key cryptography", in *Advances in Cryptology – Crypto '91*, Lecture Notes in Computer Science, Vol. 547, Springer-Verlag, pp. 498–507, 1991.

[32] 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.

[33] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of applied cryptography*, CRC Press, Boca Raton, FL, 1996.

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

[35] A. Miyaji, M. Nakabayashi, S. Takano, "New explicit condition of elliptic curve trace for FR-reduction", *IEICE Trans. Fundamentals*, Vol. E84 A, No. 5, May 2001.

[36] P. Paillier and M. Yung, "Self-escrowed public-key infrastructures" in *Information Security and Cryptology – ICISC '99*, Lecture Notes in Computer Science, Vol. 1787, Springer-Verlag, pp. 257–268, 1999.

[37] C. Rackoff, D. Simon, "Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack", in *Advances in Cryptology – Crypto '91*, Lecture Notes in Computer Science, Vol. 547, Springer-Verlag, pp. 433–444, 1991.

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

[39] K. Rubin, A. Silverberg, "Supersingular abelian varieties in cryptography", in *Advances in Cryptology – Crypto 2002*, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[40] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairings," In Proceedings of Symposium on Cryptography and Information Security, Japan, 2000.

[41] A. Shamir, "Identity-based cryptosystems and signature schemes", in *Advances in Cryptology – Crypto '84*, Lecture Notes in Computer Science, Vol. 196, Springer-Verlag, pp. 47–53, 1984.

[42] V. Shoup, 'Lower bounds for discrete logarithms and related problems', *In Proc. Eurocrypt '97*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **1233** (1997), 256–266.

[43] J. Silverman, *The arithmetic of elliptic curve*, Springer-Verlag, 1986.

[44] 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.

[45] H. Tanaka, "A realization scheme for the identity-based cryptosystem", in *Advances in Cryptology – Crypto '87*, Lecture Notes in Computer Science, Vol. 293, Springer-Verlag, pp. 341–349, 1987.

[46] E. Verheul, "Evidence that XTR is more secure than supersingular elliptic curve cryptosystems", in *Advances in Cryptology – Eurocrypt 2001*, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, pp. 195–210, 2001.

# A Definition of the Weil pairing

We define the Weil pairing and show how to efficiently compute it using an algorithm due to Miller [34]. To be concrete we present the algorithm as it applies to supersingular elliptic curves defined over a prime field $\mathbb{F}_p$ with $p > 3$ (the curve $y^2 = x^3 + 1$ over $\mathbb{F}_p$ with $p = 2 \bmod 3$ is an example of such a curve). The definition and algorithm easily generalize to computing the Weil pairing over other elliptic curves. We state a few elementary facts about such curves [43]:

**Fact 1:** A supersingular curve $E/\mathbb{F}_p$ (with $p > 3$) contains $p + 1$ points in $\mathbb{F}_p$. We let $O$ denote the point at infinity. The group of points over $\mathbb{F}_p$ forms a cyclic group of order $p + 1$. Let $P \in E(\mathbb{F}_p)$ be a point order $n$ where $n$ divides $p + 1$.

**Fact 2:** The group of points $E(\mathbb{F}_{p^2})$ 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 a subgroup which is isomorphic to the group $\mathbb{Z}_n^2$. The group is generated by $P \in E(\mathbb{F}_p)$ and $Q \in E(\mathbb{F}_{p^2})$. We denote this group by $E[n]$.

Throughout this section we let $\mathbb{G}_2$ denote the subgroup of $\mathbb{F}_{p^2}^*$ of order $n$. We will be working with the Weil pairing $e$ which maps pairs of points in $E[n]$ to $\mathbb{G}_2$, i.e. $e : E[n] \times E[n] \to \mathbb{G}_2$. To define the pairing, we review a few basic concepts (see [29, pp. 243–245]). In what follows we let $P$ and $Q$ be arbitrary points in $E(\mathbb{F}_{p^2})$.

**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 $(f) = \sum_P \mathrm{ord}_P(f) \cdot (P)$. 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 a 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 $e(P,Q)$.

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 A.1.** *The Weil pairing has the following properties for points in $E[n]$:*

- *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 $e(P,Q)^n = 1$, i.e. $e(P,Q) \in \mathbb{G}_2$.*
- *Non-degenerate in the following sense: if $P \in E[n]$ 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 $\phi$ is an automorphism on the group of points of $E$.

**Tate pairing.** The Tate pairing [17] is another bilinear pairing that has the required properties for our system. We slightly modify the original definition to fit our purpose. Define the Tate pairing of two points $P, Q \in E[n]$ as $T(P,Q) = f_P(\mathcal{A}_Q)^{|\mathbb{F}_{p^2}^*|/n}$ where $f_P$ and $\mathcal{A}_Q$ are defined as above. This definition gives a computable bilinear pairing $T : E[n] \times E[n] \to \mathbb{G}_2$.

# B 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 B.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 $a_1x + b_1y + c_1 = 0$ be the line passing through the points $bP$ and $cP$ (if $b = c$ then let $a_1x + b_1y + c_1 = 0$ be the line tangent to $E$ at $bP$). Define $g_1(x,y) = a_1x + b_1y + c_1$.
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:

$$f_{b+c}(\mathcal{A}_Q) = f_b(\mathcal{A}_Q) \cdot f_c(\mathcal{A}_Q) \cdot \frac{g_1(\mathcal{A}_Q)}{g_2(\mathcal{A}_Q)} \tag{2}$$

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 2. 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. □

Let $\mathcal{D}\big(f_b(\mathcal{A}_Q), f_c(\mathcal{A}_Q), bP, cP, (b+c)P\big) = f_{b+c}(\mathcal{A}_Q)$ denote the output of Algorithm $\mathcal{D}$ of Lemma B.1 above. Then one can compute $f_P(\mathcal{A}_Q) = f_n(\mathcal{A}_Q)$ using the following standard repeated doubling procedure. Let $n = b_m b_{m-1} \ldots b_1 b_0$ be the binary representation of $n$, i.e. $n = \sum_{i=0}^{m} b_i 2^i$.

**Init:** Set $Z = O$, $V = f_0(\mathcal{A}_Q) = 1$, and $k = 0$.

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

1: If $b_i = 1$ then do: Set $V = \mathcal{D}(V, f_1(\mathcal{A}_Q), Z, P, Z+P)$, set $Z = Z + P$, and set $k = k + 1$.
2: If $i > 0$ set $V = \mathcal{D}(V, V, Z, Z, 2Z)$, set $Z = 2Z$, and set $k = 2k$.

3: Observe that at the end of each iteration we have $Z = kP$ and $V = f_k(\mathcal{A}_Q)$.

**Output:** After the last iteration we have $k = n$ and therefore $V = f_n(\mathcal{A}_Q)$ 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 $f_Q(\mathcal{A}_P)$. The Tate pairing is evaluated similarly. 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 $a_1x + b_1y + c_1 = 0$ be the line passing through the points $P$ and $R_1$. Define the function: $g_1(x, y) = a_1x + b_1y + c_1$.
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 $\mathbb{F}_{p^2}$.