Space-Efficient Identity Based Encryption Without Pairings*

Dan Boneh† Craig Gentry‡ Michael Hamburg

{dabo,cgentry,mhamburg}@cs.stanford.edu
Stanford University

Abstract

Identity Based Encryption (IBE) systems are often constructed using bilinear maps (a.k.a. pairings) on elliptic curves. One exception is an elegant system due to Cocks which builds an IBE based on the quadratic residuosity problem modulo an RSA composite N. The Cocks system, however, produces long ciphertexts. Since the introduction of the Cocks system in 2001 it has been an open problem to construct a space efficient IBE system without pairings. In this paper we present an IBE system in which ciphertext size is short: an encryption of an ℓ-bit message consists of a single element in $\mathbb{Z}/N\mathbb{Z}$ plus $\ell + 1$ additional bits. Security, as in the Cocks system, relies on the quadratic residuosity problem. The system is based on the theory of ternary quadratic forms and as a result, encryption and decryption are slower than in the Cocks system.

1 Introduction

In an Identity Based Encryption (IBE) system any string can function as a public key [37]. IBE systems have found numerous applications in cryptography (see [13, 14, 7, 23, 42, 8, 11, 5] to name a few) and computer security [2, 41, 31, 32, 38]. There are currently two approaches for constructing IBE systems. The first approach builds IBE systems using bilinear pairings [9, 6, 40, 36]. The resulting systems are efficient both in performance and ciphertext size. The rich structure of bilinear maps enables various extensions such as Hierarchical IBE [29, 26], anonymous IBE [8, 1, 12, 25], and many others.

The second approach, due to Cocks [17], builds an elegant IBE system based on the standard quadratic residuosity problem [33, p.99] modulo an RSA composite N (in the random oracle model). Ciphertexts in this system contain two elements in $\mathbb{Z}/N\mathbb{Z}$ for every bit of plaintext. Hence, the encryption of an ℓ-bit message key is of size $2\ell \cdot \log_2 N$. For example, when encrypting a 128-bit message key using 1024-bit modulus, one ends up with a ciphertext of size 32678 bytes. For comparison, pairing based methods produce a 36 byte ciphertext for comparable security.

A long standing open problem since [17] is the construction of a space efficient IBE system without pairings, namely a system with short ciphertexts. We construct such a system — an encryption of an ℓ bit message consists of a single element in $\mathbb{Z}/N\mathbb{Z}$ plus $(\ell + 1)$ additional bits. Hence, ciphertext size is about $\ell + \log_2 N$. When encrypting a 128-bit message key the result is

*This paper is the full version of [10].

†Supported by NSF and the Packard Foundation.

‡Supported by the Herbert Kunzel Stanford Graduate Fellowship.

a ciphertext of size $1024 + 129 = 1153$ bits or 145 bytes. The system makes extensive use of the theory of quadratic forms [15]. In particular, encryption and decryption are based on an effective version of Legendre's famous three squares theorem.

Our main proposed IBE system is presented in Section 6. The system is related to the Cocks system and security is similarly based on the quadratic residuosity (QR) problem. As in the Cocks system, our IBE proof of security makes use of the random oracle model. However, the random oracle model is needed only for proving that the underlying Rabin signature scheme is existentially unforgeable. To make this precise we first prove security in *the standard model* under the Interactive QR assumption (IQR), namely under the assumption that the QR problem is difficult in the presence of a hash square root oracle. We then note that IQR is equivalent to QR in the random oracle model. We observe that the system is an anonymous IBE under the quadratic residuosity assumption. (the Cocks system is known to not be anonymous, although a variant of it is [21]). In Appendix E, we describe a general framework for using hash proof systems (as defined in [19]) that have a trapdoor to construct IBE systems that are anonymous and secure against chosen-ciphertext attacks in the standard model under the Interactive Subset Membership (ISM) assumption, a generalization of the IQR assumption. We provide hash proof systems for quadratic residuosity, which induce a system based on IQR (in the standard model). We also provide a PKE system secure against chosen-ciphertext attacks under the QR assumption, which may be of independent interest.

Encryption time in our system is not ideal. Recall that encryption time in most practical public key systems such as RSA and existing IBE systems [9, 17] is cubic in the security parameter. Encryption time in our system is quartic in the security parameter per message bit. Decryption time, however, is cubic as in other systems. The bottleneck during encryption is the need to generate primes on the order of N. In Section 5.3 we show a time space tradeoff where the number of primes to generate can be significantly reduced at the cost of a modest increase in the ciphertext size.

2 Definitions

Recall from [37, 9] that an Identity Based Encryption system (IBE) consists of four algorithms: *Setup*, *KeyGen*, *Encrypt*, *Decrypt*. The *Setup* algorithm generates system parameters, denoted by PP, and a master key MSK. The *KeyGen* algorithm uses the master key to generate the private key d_{ID} corresponding to a given identity ID. The *Encrypt* algorithm encrypts messages for a given identity (using the system parameters) and the *Decrypt* algorithm decrypts ciphertexts using the private key.

An IBE must remain secure against an attacker who can request private keys for identities of his choice. This is captured in the standard IBE security game [9], which also captures chosen ciphertext attacks. Beyond the basic semantic security requirements for IBE encryption one can also require that the IBE be anonymous, namely that a ciphertext reveal no information about the identity used to create the ciphertext. Anonymous IBE is useful for a variety of applications such as searching on encrypted data [8, 1, 12, 38, 21]. Chosen ciphertext security, private key queries, and anonymity are captured in the following IBE security game:

Setup: The challenger runs *Setup*(λ) and gives the adversary $\mathcal{A}$ the resulting public parameters PP. It keeps MSK to itself. We set $\text{ID}_0^*,\ \text{ID}_1^* \leftarrow \perp$ and $C^* \leftarrow \perp$.

Queries: The adversary issues adaptive queries $q_1, q_2, \ldots$ where query q_i is one of:

- Private key query $\langle \mathrm{ID}_i \rangle$ where $\mathrm{ID}_i \neq \mathrm{ID}_0^*,\ \mathrm{ID}_1^*$. The challenger responds by running algorithm $\mathit{KeyGen}(\mathrm{MSK}, \mathrm{ID}_i)$ and sends the resulting private key d_i to $\mathcal{A}$.
- Decryption query (ID_i, C_i) where (ID_i, C_i) is neither (ID_0^*, C^*) nor (ID_1^*, C^*). The challenger responds by running algorithm $\mathit{KeyGen}(\mathrm{MSK}, \mathrm{ID}_i)$ to generate a private key d_i and then runs algorithm $\mathit{Decrypt}(d_i, C_i)$. It sends the resulting plaintext to the adversary.
- A *single* encryption query $\big((\mathrm{ID}_0, m_0),\ (\mathrm{ID}_1, m_1)\big)$ where $\mathrm{ID}_0, \mathrm{ID}_1$ are distinct from all previous private key queries and m_0, m_1 are two equal length plaintexts. The challenger picks a random bit $b \overset{\mathrm{R}}{\leftarrow} \{0,1\}$ and sets

$$C^* \leftarrow \mathit{Encrypt}(\mathrm{PP}, \mathrm{ID}_b, m_b)\ , \qquad \mathrm{ID}_0^* \leftarrow \mathrm{ID}_0, \quad \mathrm{ID}_1^* \leftarrow \mathrm{ID}_1$$

  It sends C^* to the adversary.

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

We define $\mathcal{A}$'s advantage in attacking the scheme $\mathcal{E}$ as

$$\mathrm{IBEAdv}_{\mathcal{A},\mathcal{E}}(\lambda) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

The probability is over the random bits used by the challenger and the adversary. An adversary $\mathcal{A}$ in this game is called an ANON-IND-ID-CCA adversary. We will also consider three types of weaker adversaries:

- If $\mathcal{A}$ makes no decryption queries we say that $\mathcal{A}$ is an ANON-IND-ID-CPA adversary. This models an anonymous IBE under a chosen plaintext attack.
- If in the single encryption query used to create the challenge ciphertext C^*, the adversary uses $\mathrm{ID}_0 = \mathrm{ID}_1$ then we say that the adversary is an IND-ID-CCA adversary. This models a chosen ciphertext secure IBE that is not necessary anonymous [9].
- An adversary that makes no decryption queries and sets $\mathrm{ID}_0 = \mathrm{ID}_1$ is said to be an IND-ID-CPA adversary. This is the standard IBE security model under a chosen plaintext attack.

Definition 2.1. Let $\mathcal{S}$ be one of {IND-ID-CPA, IND-ID-CCA, ANON-IND-ID-CPA}. We say that an IBE system $\mathcal{E}$ is $\mathcal{S}$ secure if for all polynomial time $\mathcal{S}$ adversaries $\mathcal{A}$ we have that $\mathrm{IBEAdv}_{\mathcal{A},\mathcal{E}}(\lambda)$ is a negligible function.

Notation: throughout the paper we let $\mathcal{ID}_\lambda$ denote the set of all identities ID. The size of the set grows with λ. As shorthand, we will often write $\mathcal{ID}$ instead of $\mathcal{ID}_\lambda$.

2.1 Jacobi symbols and the QR assumption

For a positive integer N, we use $J(N)$ to denote the set $\{x \in \mathbb{Z}/N\mathbb{Z} : \left(\frac{x}{N}\right) = 1\}$, where $\left(\frac{x}{N}\right)$ is the Jacobi symbol of x in $\mathbb{Z}/N\mathbb{Z}$. We use $\mathrm{QR}(N)$ to denote the set of quadratic residues in $J(N)$. We base the security of our system on the following computational assumption.

Definition 2.2 (Quadratic Residuosity Assumption (QR))**.** Let RSAgen(λ) be a PPT algorithm that generates two equal size primes p, q.

- Let $\mathcal{P}_{QR}(\lambda)$ be the distribution:

$$(p, q) \xleftarrow{\text{R}} \text{RSAgen}(\lambda), \quad N \leftarrow pq, \quad V \xleftarrow{\text{R}} \text{QR}(N), \quad \text{output}(N, V)$$

- Let $\mathcal{P}_{NQR}(\lambda)$ be the distribution:

$$(p, q) \xleftarrow{\text{R}} \text{RSAgen}(\lambda), \quad N \leftarrow pq, \quad V \xleftarrow{\text{R}} J(N) \setminus \text{QR}(N), \quad \text{output}(N, V)$$

We say that the **QR assumption** holds for RSAgen if for all PPT algorithms $\mathcal{A}$, the function

$$\left| \Pr[(N, V) \xleftarrow{\text{R}} \mathcal{P}_{QR}(\lambda) \ : \ \mathcal{A}(N, V) = 1] - \Pr[(N, V) \xleftarrow{\text{R}} \mathcal{P}_{NQR}(\lambda) \ : \ \mathcal{A}(N, V) = 1] \right|$$

is negligible. We call this the QR advantage $\text{QRAdv}_{\mathcal{A},\text{RSAgen}}(\lambda)$ of $\mathcal{A}$ against RSAgen.

For completeness, we also prove our system secure without relying on random oracles. This, however, requires a stronger assumption we call the *interactive QR assumption*. Basically, the assumption says that the QR assumption holds relative to a square root oracle.

Definition 2.3 (Interactive Quadratic Residuosity Assumption (IQR))**.** Let H be a hash function such that for every integer N the function $H_N(\cdot)$ maps $\{0, 1\}^*$ to $J(N)$. Let $\mathcal{O}$ be a square root oracle: for every N which is a product of two odd primes, $\mathcal{O}$ picks a quadratic non-residue $u_N \in J(N)$; it maps an input pair (N, x) to one of $H_N(x)^{1/2}$ or $(u_N \, H_N(x))^{1/2}$ in $\mathbb{Z}/N\mathbb{Z}$, depending on which value is a quadratic residue. The oracle $\mathcal{O}$ is chosen uniformly from the set of all such functions. We say that the **Interactive QR assumption** (IQR) holds for the pair (RSAgen, H) if for all PPT algorithms $\mathcal{A}$, the function

$$\left| \Pr[(N, V) \xleftarrow{\text{R}} \mathcal{P}_{QR}(\lambda) \ : \ \mathcal{A}^{\mathcal{O}}(N, V) = 1] - \Pr[(N, V) \xleftarrow{\text{R}} \mathcal{P}_{NQR}(\lambda) \ : \ \mathcal{A}^{\mathcal{O}}(N, V) = 1] \right|$$

is negligible. We call this the IQR advantage $\text{IQRAdv}_{\mathcal{A},(\text{RSAgen},H)}(\lambda)$ of $\mathcal{A}$ against RSAgen and H. Note that for IQR to hold, it must be difficult to find collisions in H, since each collision allows the adversary to factor N with probability 1/2.

Note that the oracle $\mathcal{O}$ is a Rabin signature oracle. For a messages $m \in \{0, 1\}^*$ and public key (N, u_N), the value $\mathcal{O}(N, m)$ is the Rabin signature on m.

When H is a full-domain hash function modeled as a random oracle, the QR assumption implies the IQR assumption. Hence, our system will be secure in the standard model based on IQR and in the random oracle model based on the standard QR assumption.

Our IBE system also uses a pseudorandom function (PRF) as defined in [27].

3 An abstract IBE system with short ciphertexts

We begin by describing an abstract IBE system that produces short ciphertexts. The system uses a deterministic algorithm $\mathcal{Q}$ with the following properties.

Definition 3.1. Let $\mathcal{Q}$ be a deterministic algorithm that takes as input (N, R, S) where $N \in \mathbb{Z}^+$ and $R, S \in \mathbb{Z}/N\mathbb{Z}$. The algorithm outputs two polynomials $f, g \in \mathbb{Z}/N\mathbb{Z}[x]$. We say that $\mathcal{Q}$ is **IBE compatible** if the following two conditions hold:

- (Condition 1) If R and S are quadratic residues, then $f(r)g(s)$ is a quadratic residue for all square roots r of R and s of S.

- (Condition 2) If R is a quadratic residue, then $f(r)f(-r)S$ is a quadratic residue for all square roots r of R.

We construct an IBE compatible algorithm $\mathcal{Q}$ in Sections 4 and 5. Condition 1 implies that the Legendre symbol $(f(r)/N)$ is equal to $(g(s)/N)$, a fact that will be used during decryption. Condition 2 will be used to prove security.

A single bit system. As a warm up to our IBE construction, consider briefly the following simple IBE for one bit messages.

Setup(λ): generate $(p, q) \overset{\mathrm{R}}{\leftarrow} \mathrm{RSAgen}(\lambda)$, $N \leftarrow pq$, and a random $u \overset{\mathrm{R}}{\leftarrow} J(N) \setminus \mathrm{QR}(N)$. Output public parameters $\mathrm{PP} = (N, u, H)$ where H is a hash function $H : \mathcal{ID} \to J(N)$. The master key MSK is the factorization of N.

KeyGen(MSK, ID): generate a private key by first setting $R \leftarrow H(\mathrm{ID})$. If $R \in \mathrm{QR}(N)$ set $r \leftarrow R^{1/2}$ and otherwise set $r \leftarrow (uR)^{1/2}$. Output r as the private key for ID. Note that the private key is essentially a Rabin signature on ID, as in the Cocks system.

Encrypt(PP, ID, m): to encrypt $m \in \{\pm 1\}$ with public key ID pick a random $s \in \mathbb{Z}/N\mathbb{Z}$ and compute $S \leftarrow s^2$. Let $R \leftarrow H(\mathrm{ID})$. Run $\mathcal{Q}$ twice:

$$(f, g) \leftarrow \mathcal{Q}(N,\ R,\ S) \quad \text{and} \quad (\bar{f}, \bar{g}) \leftarrow \mathcal{Q}(N,\ uR,\ S)$$

and encrypt m using the two Jacobi symbols: $c \leftarrow m \cdot \left(\frac{g(s)}{N}\right)$ and $\bar{c} \leftarrow m \cdot \left(\frac{\bar{g}(s)}{N}\right)$. Output the ciphertext $C \leftarrow (S, c, \bar{c})$.

Decrypt(C, r): decrypt $(S, c, \bar{c})$ using private key r. Let us first suppose that $R = H(\mathrm{ID})$ is in $\mathrm{QR}(N)$ so that $r^2 = R$. The decryptor runs $\mathcal{Q}(N,\ R,\ S)$ to obtain (f, g). By condition (1) of Definition 3.1 we know that

$$\left(\frac{g(s)}{N}\right) = \left(\frac{f(r)}{N}\right)$$

Hence the plaintext is obtained by setting $m \leftarrow c \cdot \left(\frac{f(r)}{N}\right)$. If R is a non-residue then uR is a quadratic residue and $r^2 = uR$. We decrypt by running $\mathcal{Q}(N,\ uR,\ S)$ and recovering m from $\bar{c}$. Since $\mathcal{Q}$ is deterministic, both sender and receiver always obtain the same pairs (f, g) and $(\bar{f}, \bar{g})$.

This completes the description of the one bit abstract system. Condition (1) implies soundness. Condition (2) is used to prove semantic security under the QR assumption. We note that the Cocks system, reviewed in Appendix A, is not an instance of this system.

Remark: Throughout the paper we let S, s be values chosen by the Sender (encryptor). We let R, r be values chosen by the Receiver (decryptor).

3.1 The Multi-Bit Abstract IBE System

Observe that a single value S chosen by the encryptor can be used to encrypt multiple bits. To encrypt an ℓ-bit message we hash ID multiple times by computing $R_i \leftarrow H(\mathrm{ID}, i)$ for $i = 1, \ldots, \ell$. Now each pair (S, R_i) can be used to encrypt one message bit. The ciphertext only grows by two bits $(c_i, \bar{c}_i)$ per message bit. Hence, when encrypting an ℓ bit message, the complete ciphertext is $C = (S, (c_1, \bar{c}_1), \ldots, (c_\ell, \bar{c}_\ell))$ whose length is $\lceil \log_2(N) \rceil + 2\ell$ bits. In Section 6 we show how to shrink the ciphertext length to $\lceil \log_2(N) \rceil + (\ell + 1)$ bits.

We describe the complete system, called "BasicIBE" in more detail.

Setup(λ): Generate $(p, q) \xleftarrow{\mathrm{R}} \mathrm{RSAgen}(\lambda)$, $\quad N \leftarrow pq$, and a random $u \xleftarrow{\mathrm{R}} J(N) \setminus \mathrm{QR}(N)$. Output public parameters $\mathrm{PP} = (N, u, H)$ where H is a hash function $H : \mathcal{ID} \times [1, \ell] \to J(N)$.
The master key MSK is the factorization of N and a random key K for a pseudorandom function $F_K : \mathcal{ID} \times [1, \ell] \to \{0, 1, 2, 3\}$.

KeyGen$(\mathrm{MSK}, \mathrm{ID}, \ell)$: Takes as input $\mathrm{MSK}, \mathrm{ID}$, and a message length parameter ℓ. It generates a private key for decrypting encryptions of ℓ-bit messages. For $j = 1, \ldots, \ell$ do:

```
$R_j \leftarrow H(\mathrm{ID}, j) \in J(N)$   and   $w \leftarrow F_K(\mathrm{ID}, j) \in \{0,1,2,3\}$
let $a \in \{0,1\}$ be such that $u^a R_j \in \mathrm{QR}(N)$
let $\{z_0, z_1, z_2, z_3\}$ be the four square roots of $u^a R_j$ in $\mathbb{Z}/N\mathbb{Z}$
Set $r_j \leftarrow z_w$
```

Output the decryption key $d_{\mathrm{ID}} \leftarrow (\mathrm{PP}, r_1, \ldots, r_\ell)$. The PRF F ensures that the key generator always outputs the same square roots for a given ID, but an adversary cannot tell ahead of time which of the four square roots will be output.

Encrypt$(\mathrm{PP}, \mathrm{ID}, m)$: Takes as input PP, a user ID, and a message $m = m_1 \ldots m_\ell \in \{-1, 1\}^\ell$. It generates random $s \in \mathbb{Z}/N\mathbb{Z}$ and sets $S \leftarrow s^2$. For $j = 1, \ldots, \ell$ do:

```
(1)   $R_j \leftarrow H(\mathrm{ID}, j)$ ,   $(f, g) \leftarrow \mathcal{Q}(N, R_j, S)$,   and   $(\bar{f}, \bar{g}) \leftarrow \mathcal{Q}(N, uR_j, S)$
(2)   $c_j \leftarrow m_j \cdot \left(\frac{g_j(s)}{N}\right)$   and   $\bar{c}_j \leftarrow m_j \cdot \left(\frac{\bar{g}_j(s)}{N}\right)$ .
Set $c \leftarrow c_1 \cdots c_\ell$ and $\bar{c} \leftarrow \bar{c}_1 \cdots \bar{c}_\ell$ and output the ciphertext $C \leftarrow (S, c, \bar{c})$.
```

Decrypt(C, d_{ID}): Let $d_{\mathrm{ID}} = (\mathrm{PP}, r_1, \ldots, r_\ell)$. For $j = 1, \ldots, \ell$ let $R_j \leftarrow H(\mathrm{ID}, j)$ and do:

```
if $r_j^2 = R_j$  run $(f_j, g_j) \leftarrow \mathcal{Q}(N, R_j, S)$   and set $m_j \leftarrow c_j \cdot \left(\frac{f_j(r_j)}{N}\right)$
if $r_j^2 = uR_j$  run $(\bar{f}_j, \bar{g}_j) \leftarrow \mathcal{Q}(N, uR_j, S)$   and set $m_j \leftarrow \bar{c}_j \cdot \left(\frac{\bar{f}_j(r_j)}{N}\right)$
Output $m \leftarrow m_1 \ldots m_\ell$.
```

This completes the description of BasicIBE. Soundness of decryption follows from Condition 1.

Remark: The function H outputs elements in $J(N)$. This function can be easily implemented using a hash function H' that outputs elements in $\mathbb{Z}/N\mathbb{Z}$. Simply include in PP some element $z \in \mathbb{Z}/N\mathbb{Z}$ whose Jacobi symbol (z/N) is -1. Then, to compute $H(\mathrm{ID}, j)$ first compute $x \leftarrow H'(\mathrm{ID}, j)$. If $x \in J(N)$ output $H(\mathrm{ID}, j) = x$, otherwise output $H(\mathrm{ID}, j) = xz$. Either way, $H(\mathrm{ID}, j) \in J(N)$ as required.

3.2 Security

We prove security of the IBE system BasicIBE in the random oracle model based on the QR assumption.

Theorem 3.2. *Suppose the QR assumption holds for RSAgen and F is a secure PRF. Then the IBE system* BasicIBE *is IND-ID-CPA secure when H is modeled as a random oracle. In particular, suppose $\mathcal{A}$ is an efficient IND-ID-CPA adversary. Then there exist efficient algorithms $\mathcal{B}_1, \mathcal{B}_2$ (whose running time is about the same as that of $\mathcal{A}$) such that*

$$\mathrm{IBEAdv}_{\mathcal{A},\mathsf{BasicIBE}}(\lambda) \leq 2 \cdot \mathrm{QRAdv}_{\mathcal{B}_1, RSAgen}(\lambda) + \mathrm{PRFAdv}_{\mathcal{B}_2, F}(\lambda).$$

The proof is given in Appendix B as a sequence of games. We also describe the underlying public key encryption system and prove its semantic security with a simple reduction style proof to the QR assumption.

We note that the bounds in Theorem 3.2 are tight, and do not depend on the number of private key queries from the adversary. Since the theorem is set in the random oracle model, we could avoid the additional PRF assumption by implementing the PRF using the random oracle H. However, the proof is simpler and the result more concrete using a PRF.

The proof of Theorem 3.2 requires Condition (2) of Definition 3.1. Condition (2) is only used to satisfy the conditions of the following simple lemma, which is used to prove semantic security — it enables the challenger to create a challenge ciphertext that is well formed when S is in $\mathrm{QR}(N)$ but is random when S is not (see Appendix B).

Lemma 3.3. *Let $N = pq$ be an RSA modulus, $X \in \mathrm{QR}(N)$, and $S \in J(N)$. Let x be a random variable uniformly chosen from among the four square roots of X. Let f be a polynomial such that $f(x)f(-x)S$ is a quadratic residue for all four values of x. Then:*

- *when $S \notin \mathrm{QR}(N)$ the Jacobi symbol $(f(x)/N)$ is uniformly distributed in $\{\pm 1\}$;*
- *when $S \in \mathrm{QR}(N)$ then $(f(x)/N)$ is constant, namely the same for all four values of x.*

Proof. Let x, x' be two square roots of X such that $x' = x \bmod p$, but $x' = -x \bmod q$. Then the four square roots of X are $\{\pm x, \pm x'\}$. When $S \notin \mathrm{QR}(N)$, we have $\left(\frac{f(x)}{p}\right) = -\left(\frac{f(-x)}{p}\right)$, and the same on q. By the Chinese Remainder Theorem, $\left(\frac{f(x')}{p}\right) = \left(\frac{f(x)}{p}\right)$ but $\left(\frac{f(x')}{q}\right) = -1 \cdot \left(\frac{f(x)}{q}\right)$, so that $\left(\frac{f(x')}{N}\right) = -1 \cdot \left(\frac{f(x)}{N}\right)$. So of the four values $f(x), f(x'), f(-x), f(-x')$, the first two must have different Jacobi symbols, as must the last two. Hence, among the four symbols, two are $+1$ and two are -1. When $S \in \mathrm{QR}(N)$ all four symbols are equal. □

Remark: The system BasicIBE is CPA secure, but is not anonymous — there are instances of $\mathcal{Q}$ for which anyone can test which identity created a given ciphertext. In Section 6 we make the system anonymous. Moreover, we shrink the ciphertext length to $\lceil \log_2(N) \rceil + (\ell + 1)$ bits. The resulting system can be proven secure in the standard model under the IQR assumption. The construction, however, cannot be based on a general IBE compatible algorithm $\mathcal{Q}$. Instead, $\mathcal{Q}$ must satisfy additional properties. We view the construction in Section 6 as our main proposed construction.

4 Concrete instantiation: an IBE compatible algorithm

To make our abstract system concrete, we need to give an IBE compatible algorithm $\mathcal{Q}$. Recall that $\mathcal{Q}$ must generate polynomials f and g that meet Conditions 1 and 2 of Definition 3.1. The algorithm works as follows.

Algorithm $\mathcal{Q}(N, R, S)$:

1. Construct a solution $(x, y) \in (\mathbb{Z}/N\mathbb{Z})^2$ to the equation $\quad Rx^2 + Sy^2 = 1 \qquad (*)$
   In Section 5 we describe algorithms and optimizations for solving this equation.
   Solving this equation is the main bottleneck in our system.
2. Output the polynomials $\quad f(r) \leftarrow xr + 1 \quad$ and $\quad g(s) \leftarrow 2ys + 2$.

We show that $\mathcal{Q}$ is IBE compatible. Let $R, S \in \mathbb{Z}/N\mathbb{Z}$. Let r be a square root of R and s a square root of S, if one exists. Condition 1 is met, since

$$f(r) \cdot g(s) = 2xrys + 2xr + 2ys + 2 + (Rx^2 + Sy^2 - 1) = (xr + ys + 1)^2 \pmod{N} .$$

Condition 2 is met, since $\quad f(r) \cdot f(-r) \cdot S = (1 - Rx^2)S = (Sy)^2 \pmod{N}$.

Hence we have a valid instantiation for $\mathcal{Q}$.

5 Algorithms and Optimizations

The instantiation of algorithm $\mathcal{Q}$ in the previous section needs an algorithm that given an integer N and $R, S \in \mathbb{Z}/N\mathbb{Z}$ outputs a pair $(x, y) \in \mathbb{Z}/N\mathbb{Z}^2$ such that

$$Rx^2 + Sy^2 = 1 \tag{1}$$

Throughout the section we assume $R \neq S$ and RS is in $(\mathbb{Z}/N\mathbb{Z})^*$. Recall that when encrypting an ℓ-bit message using the abstract system (Section 3), the encryptor must solve 2ℓ such equations. In particular, one needs pairs $(x_i, y_i),\ (\bar{x}_i, \bar{y}_i) \in \mathbb{Z}/N\mathbb{Z}^2$ such that

$$R_i \cdot x_i^2 + S \cdot y_i^2 = 1 \qquad \text{and} \qquad (uR_i) \cdot \bar{x}_i^2 + S \cdot \bar{y}_i^2 = 1 \tag{2}$$

for $i = 1, \ldots, \ell$. The decryptor needs a solution to ℓ of these equations.

5.1 A product formula

Although the encryptor needs solutions to the 2ℓ equations in (2), we show that solving only $\ell + 1$ equations is sufficient. We do so using a product formula that, given a solution to two equations, builds a solution to a third equation.

Lemma 5.1. *For $i = 1, 2$ suppose (x_i, y_i) is a solution to $A_i x^2 + By^2 = 1$. Then (x_3, y_3) is a solution to*

$$(A_1 A_2) \cdot x^2 + B \cdot y^2 = 1 \tag{3}$$

where $x_3 = (x_1 x_2)/(By_1 y_2 + 1)$ and $y_3 = (y_1 + y_2)/(By_1 y_2 + 1)$.

Proof. The proof is by direct substitution into (3). □

During encryption, the encryptor first finds solutions in $\mathbb{Z}/N\mathbb{Z}$ to the $\ell + 1$ equations

$$u \cdot x^2 + S \cdot y^2 = 1 \qquad \text{and} \qquad R_i \cdot x^2 + S \cdot y^2 = 1 \quad \text{for } i = 1, \ldots, \ell \tag{4}$$

using the algorithms discussed in the next section. The encryptor can now use Lemma 5.1 to quickly find solutions to the remaining ℓ equations in (2). Simply apply Lemma 5.1 to the left equation in (4) and the ith right equation in (4) to obtain a solution to $(uR_i) \cdot x^2 + S \cdot y^2 = 1$, as required. The same applies during decryption.

Overall, during encryption and decryption we need only to construct solutions to $\ell + 1$ equations of the form (1). For convenience, from here on we set $R_0 = u$.

5.2 Computing Solutions to $Rx^2 + Sy^2 = 1$ in $\mathbb{Z}/N\mathbb{Z}$

Now we turn to solving (1). Pollard and Schnorr[35] provided, as part of their cryptanalysis of the Ong-Schnorr-Shamir signature scheme[34], a beautiful polynomial time algorithm (assuming the GRH) for finding solutions to the equation $x^2 - Sy^2 = R$ in $\mathbb{Z}/N\mathbb{Z}$. Clearly, a solution to the Pollard-Schnorr equation (with $x \neq 0$) also gives a solution to (1). The algorithm, however, is relatively slow requiring one to generate $O(\log \log N)$ primes on the order of N.

We, instead, take a different approach. We lift (1) to the integers and consider the ternary quadratic form

$$\tilde{R}x^2 + \tilde{S}y^2 - z^2 = 0 \tag{5}$$

where $\tilde{R}, \tilde{S}, x, y, z \in \mathbb{Z}$ and $\tilde{R} = R,\ \tilde{S} = S \bmod N$. A solution to (5) in $\mathbb{Z}$ gives a solution to (1) in $\mathbb{Z}/N\mathbb{Z}$. Since N is odd, by adding multiples of N to $\tilde{R}$ and $\tilde{S}$ we can assume that

$$\gcd(\tilde{R}, \tilde{S}) = 1, \quad \tilde{R} \text{ is odd}, \quad \text{and} \quad \tilde{S} \text{ is } 1 \bmod 4. \tag{6}$$

Then, by quadratic reciprocity, $\tilde{R}$ is a square mod $\tilde{S}$ if and only if $\tilde{S}$ is a square mod $\tilde{R}$. We let S' denote an integer in $[0, 4N]$ such that $S' = S \bmod N$ and $S' = 1 \bmod 4$. Then $\tilde{S} = S' \bmod 4N$.

A classic result of Legendre [15] says that for $\tilde{R}$ and $\tilde{S}$ satisfying (6), equation (5) has a solution (x, y, z) in $\mathbb{Z}$ whenever there exist integers r, s such that

$$r^2 = \tilde{R} \pmod{\tilde{S}} \quad \text{and} \quad s^2 = \tilde{S} \pmod{\tilde{R}} \tag{7}$$

Cremona and Rusin [20] present an effective version of Legendre's theorem. They present several algorithms that take as input integers $N, \tilde{R}, \tilde{S}, r, s$ satisfying (7) and output a solution to (5). At the heart of one of these algorithms is a lattice reduction in a simple 3-dimensional lattice of determinant $4\tilde{R}\tilde{S}$. For completeness, we present the algorithm in Appendix C.

Consequently, solving (1) reduces to finding integers r, s satisfying (7). To do so, one can look for the smallest (probable) prime $\tilde{R}$ along the arithmetic progression $\tilde{R} = R \bmod N$. Next one looks for the smallest (probable) prime $\tilde{S}$ along the arithmetic progression $\tilde{S} = S' \bmod 4N$ for which the Legendre symbol $(\tilde{S}/\tilde{R}) = 1$. Finally, one computes the appropriate modular square roots to find an r, s satisfying (7). Given r, s we obtain a solution to (5) which leads to a solution to (1) as required.

Fortunately, we do not need a full primality test – one can use a simple deterministic square root algorithm as a light weight primality test. If we obtain square roots r, s satisfying (7), we are

1024-bit Jacobi symbol:	0.135 ms
full 1024-bit exponentiation:	5.0 ms
1536-bit 3x3 LLL lattice reduction:	6.8 ms
1024-bit prime generation:	123.6 ms
512-bit prime generation:	11.0 ms

Table 1: Running times on 2.015GHz AMD dual-core Athlon64

done, even if $\tilde{R}, \tilde{S}$ are not primes. We give the details in Appendix C.1. Similarly, the number of candidates $\tilde{R}, \tilde{S}$ considered can be greatly reduced by sieving to eliminate all candidates divisible by small primes. To summarize, we obtain the following result:

Lemma 5.2. *There is a deterministic algorithm $\mathcal{Q}$ that on input N and $R, S \in (\mathbb{Z}/N\mathbb{Z})^*$ finds a solution to (1). $\mathcal{Q}$'s running time is dominated by the time to generate two primes $\tilde{R} = R \bmod N$ and $\tilde{S} = S' \bmod 4N$ satisfying $(\tilde{S}/\tilde{R}) = 1$.*

We present the running times for various parts of the algorithm in Table 1. As expected, the bottleneck is prime number generation. In Section 5.5 we show that it suffices to generate primes on the order of $\sqrt{N}$ corresponding to the last row in the table. These performance numbers were obtained using the SAGE library.

5.2.1 Solving the system of $\ell + 1$ equations

Recall that in our system the encryptor must solve the $\ell + 1$ equations in (4). Clearly one could apply Lemma 5.2 $(\ell + 1)$ times. However, we would like to minimize the number of (expensive) prime generations needed by our system. We show that a single prime $\tilde{S}$ can be used to solve all $\ell + 1$ equations. Moreover, we can eliminate prime generation altogether during decryption.

As a preliminary optimization, instead of storing u in PP, we let $\tilde{u}$ be the smallest prime such that $\tilde{u} = u \bmod N$ and $\tilde{u} = 3 \bmod 4$, and we store $\tilde{u}$ in PP. Let $\tilde{R}_0 = \tilde{u}$. Below, we provide two methods for generating the other primes needed by the system.

Method 1: The encryptor finds the smallest (probable) prime $\tilde{S}$ such that $\tilde{S} = S' \bmod 4N$ and $(\tilde{S}/\tilde{u}) = 1$. Then for $i = 1, \ldots, \ell$ the encryptor finds the smallest (probable) prime $\tilde{R}_i \in \mathbb{Z}^+$ such that

$$\tilde{R}_i = R_i \bmod N \quad \text{and} \quad \left(\frac{\tilde{S}}{\tilde{R}_i}\right) = 1$$

Note that by quadratic reciprocity we also have $\left(\frac{\tilde{R}_i}{\tilde{S}}\right) = 1$ for $i = 0, \ldots, \ell$. Using the primes $(\tilde{S}, \tilde{R}_i)$ the encryptor can find a solution to $R_i x^2 + S y^2 = 1 \bmod N$ needed for encrypting the i'th bit of the message. Overall, the encryptor needs to generate $(\ell + 1)$ primes.

On the decryption side, for all $i = 1, \ldots, \ell$, the decryptor precomputes and stores the first few primes $\tilde{R}_{i,1}, \tilde{R}_{i,2}, \ldots$ that are congruent to R_i modulo N. Moreover, the encryptor can embed $\tilde{S}$ in the ciphertext and as a result, the decryptor can directly solve the equations $R_i x^2 + S y^2 = 1 \bmod N$ for $i = 0, \ldots, \ell$ without generating any primes. Consequently, decryption in this system is much faster than encryption. Note, however, that the decryptor must precompute and store about $\ell \log_2 \ell$ primes.

Method 2: We reduce the decryptor's storage requirement to ℓ primes at the expense of slightly increasing the size of PP and the time needed to compute a solution to (5). In particular, we augment PP with values $p_1, \ldots, p_t \in \mathbb{Z}/N\mathbb{Z}$, where each $P_j \leftarrow p_j^2 \bmod N$ is a prime in $\mathbb{Z}$ congruent to 3 modulo 4. We can have $t = O(\log \ell)$. Let $\mathcal{P} = \{P_1, \ldots, P_t\}$ and notice that $\mathcal{P}$ can be generated without knowledge of N's factorization.

The encryptor generates $\tilde{S}$ as before, except that $\tilde{S} = 3 \bmod 4$. For $i = 1, \ldots, \ell$, the encryptor finds the smallest (probable) prime $\tilde{R}_i \in \mathbb{Z}^+$ such that $\tilde{R}_i = R_i \bmod N$ and $\tilde{R}_i = 3 \bmod 4$. For all $i = 0, \ldots, \ell$ there are two cases:

- If $(\tilde{S}/\tilde{R}_i) = -1$, then $(\tilde{R}_i/\tilde{S}) = 1$ by quadratic reciprocity. The encryptor and decryptor deterministically select a prime $P_j \in \mathcal{P}$ such that $(P_j/\tilde{R}_i) = -1$, and set $\tilde{S}' = \tilde{S} \cdot P_j$. In the unlikely event that no such P_j exists, the encryptor looks for the next suitable prime along the arithmetic progression $\tilde{S} = S \bmod N$. Note that $\tilde{S}'$ is a square modulo $\tilde{R}_i$, and vice versa by quadratic reciprocity. Moreover, $\tilde{S}' = 1 \bmod 4$. For these reasons, and since the factors of $\tilde{S}'$ and $\tilde{R}_i$ are known, the encryptor and decryptor can each find a solution to $\tilde{R}_i x^2 + \tilde{S}' y^2 - z^2 = 0$ in $\mathbb{Z}$. This yields a solution $(x/z,\ p_j \cdot y/z)$ to $R_i x^2 + S y^2 = 1$ in $\mathbb{Z}/N\mathbb{Z}$, as needed for encrypting the i'th bit of the message.

- If $(\tilde{S}/\tilde{R}_i) = 1$, then $(\tilde{R}_i/\tilde{S}) = -1$. Encryption and decryption proceed as in the first case, except that we multiply $\tilde{R}_j$ by $P_j \in \mathcal{P}$ to obtain $\tilde{R}' = \tilde{R}_j \cdot P_j$ where $(\tilde{R}'/\tilde{S}) = 1$.

Since $\tilde{S}'$ has about twice as many bits as $\tilde{S}$, the lattice reduction step may take a little longer. However, empirically, the lattice reduction time is dictated by the smaller of the two coefficients (e.g., $|\tilde{R}_i| < |\tilde{S}'|$). In practice, the effect on lattice reduction time is therefore minimal. Since it takes two full $(\log N)$-bit exponentiations to compute $\tilde{R}_i^{1/2} \bmod \tilde{S}'$, this method impacts decryption time somewhat. But it has little impact on encryption time, which is dominated by prime generation.

5.3 A time space tradeoff

So far the encryption bottleneck is in generating the primes $\tilde{S}, \tilde{R}_1, \tilde{R}_2, \ldots$. This work can be greatly reduced by splitting the ℓ-bit plaintext into $\sqrt{\ell}$ blocks each containing $\sqrt{\ell}$ bits. We encrypt each block separately. Write $t \leftarrow \lceil \sqrt{\ell} \rceil$.

For example, the encryptor may instantiate this idea with method 2, using the same values of $\tilde{R}_1, \ldots, \tilde{R}_t$ with each block. For the i'th block, the encryptor generates $\tilde{S}_i$ as in method 2, and encryption proceeds as usual. Overall, the encryptor generates only $2t = 2\lceil \sqrt{\ell} \rceil$ primes, as opposed to $\ell + 1$ primes as discussed in the previous section. The downside is that the ciphertext size increases from a single element in $\mathbb{Z}/N\mathbb{Z}$ to t elements $S_1, \ldots, S_t$ in $\mathbb{Z}/N\mathbb{Z}$.

For concrete parameters, say $\ell = 128$ bits, this approach reduces the number of primes to generate during encryption from $\ell + 1 = 129$ (as in method 2) to $2\lceil \sqrt{\ell} \rceil = 24$, about a factor of 5 improvement. The ciphertext grows from one element in $\mathbb{Z}/N\mathbb{Z}$ to 12 elements $(\tilde{S}_1, \ldots, \tilde{S}_{12})$.

5.4 Hashing IDs to small elements in $\mathbb{Z}/N\mathbb{Z}$

To further speed up prime generation, we can change slightly how the primes $\tilde{R}_1, \ldots, \tilde{R}_\ell$ are generated. Recall that $R_1, \ldots, R_\ell \in \mathbb{Z}/N\mathbb{Z}$ are generated by hashing (ID, i) into $\mathbb{Z}/N\mathbb{Z}$ for $i = 1, \ldots, \ell$. We needed a a uniform distribution over $\mathbb{Z}/N\mathbb{Z}$ for the random oracle simulation. In particular, we needed to generate random pairs $(z, z^2) \in \mathbb{Z}/N\mathbb{Z}$ and then program the oracle so that $H(\mathrm{ID}, i) = z^2$.

Gentry [24], building on work by Vallée [39] and Coron [18], shows how to generate pairs $(z, z^2) \in \mathbb{Z}/N\mathbb{Z}$ where z^2 is indistinguishable from uniform in $QR(N) \cap [1, 8N^{2/3}]$ and z is a uniformly random square root of z^2. Consequently, we can hash (ID, i) into $[1, 8N^{2/3}]$ without hurting the simulation. This can potentially speed up prime number generation. We note that hashing into a shorter interval can become insecure due to the attacks of Desmedt and Odlyzko [22].

Let us assume that for $i = 1, \ldots, \ell$ the R_i are in the range $[1, 8N^{2/3}]$. We can no longer look for primes $\tilde{R}_i$ along the arithmetic progression $\tilde{R}_i = R_i \bmod N$ since that will negate the benefit of having small R_i. Instead, we generate the $\tilde{R}_i$ by hashing (ID, i, v) for $v = 1, 2, \ldots$. We define $\tilde{R}_i$ as the first prime along this sequence. Since the numbers are a little smaller than before (size $8N^{2/3}$ as opposed to N), finding such a prime is faster than looking along the arithmetic progression.

5.5 Hashing IDs to products of small primes

Another way to speed up prime generation is to compute $\tilde{R}_i$ as the product $\prod_{b=0}^{k-1} H(\mathrm{ID}, i, b)$, where H outputs $\lceil (\log N)/k \rceil$-bit primes, each with Jacobi symbol 1 modulo N. $\tilde{R}_0 = \tilde{u}$ can also be set in this way. Since prime generation takes quartic time (without sieving), generating $\tilde{R}_i$ in this way takes much less time for the sender – by a factor of $O(k^3)$ – than generating $\tilde{R}_i$ as in Section 5.2. The PKG computes the private key value r_i from $\tilde{R}_i \bmod N$ in the usual way.

When $k = 2$, this approach can be simulated very efficiently in the random oracle model. To define $H(\mathrm{ID}, i, b)$, the simulator generates random $q_i \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}/N\mathbb{Z}$ and $a_i \overset{\mathrm{R}}{\leftarrow} \{0, 1\}$, sets $Q_i \leftarrow u^{a_i} q_i^2 \in \mathbb{Z}/N\mathbb{Z}$ and uses continued fractions to find small numbers $x, y \in \mathbb{Z}/N\mathbb{Z}$ such that $Q_i = x/y$. It repeatedly picks new random (q_i, a_i) until x and y are $\lceil (\log N)/2 \rceil$-bit primes coming from the appropriate distribution. Next, it sets $r_i \leftarrow q_i \cdot y$ and $\tilde{R}_i \leftarrow Q_i \cdot y^2$ $(= x \cdot y)$. When queried, it returns $H(\mathrm{ID}, i, 0) \leftarrow x$, $H(\mathrm{ID}, i, 1) \leftarrow y$, and r_i as the private key component for $\tilde{R}_i$.

Moreover, when $k = 2$, we can construct a variant of our system that uses the faster approach to prime generation, without making the other steps (notably, the lattice reduction) much slower. In this variant, the PKG augments PP as in Method 3 in Section 5.2.1, except that each $P_j \in \mathcal{P}$ is a product of two $\lceil (\log N)/2 \rceil$-bit primes, each congruent to 3 modulo 4; let $P_{j,0}$ and $P_{j,1}$ denote the two factors of P_j. To instantiate H efficiently, $H(\mathrm{ID}, i, 0)$ is computed (using sieving) as the first prime following $H'(\mathrm{ID}, i, 0)$ that is congruent to 1 modulo 4, where H' outputs $\lceil (\log N)/2 \rceil$-bit integers; $H(\mathrm{ID}, i, 1)$ is computed identically, except it equals 3 modulo 4. To compute $\tilde{R}_i$, the encryptor sets $\tilde{R}_{i,0} \leftarrow H(\mathrm{ID}, i, 0)$, $\tilde{R}_{i,1} \leftarrow H(\mathrm{ID}, i, 1)$, and $\tilde{R}_i \leftarrow \tilde{R}_{i,0} \cdot \tilde{R}_{i,1}$. The encryptor generates $\tilde{S}$ in the usual way. (This single large prime generation seems unavoidable.) For $i = 0, \ldots, \ell$, if $(\tilde{R}_{i,0}/\tilde{S}) = (\tilde{R}_{i,1}/\tilde{S}) = 1$, the encryptor and decryptor compute a solution to $\tilde{R}_i x^2 + \tilde{S} y^2 - z^2 = 0$ in the usual way. If $(\tilde{R}_{i,0}/\tilde{S}) = -1$ and $(\tilde{R}_{i,1}/\tilde{S}) = 1$, they select $P_j, P_k \in \mathcal{P}$ such that $\tilde{R}_{i,0} \cdot P_{j,0}$, $\tilde{R}_{i,1}$ and $P_{j,1}$ are all squares modulo $\tilde{S}' \leftarrow \tilde{S} \cdot P_k$, and vice versa. Then, they solve the three equations $Tx^2 + \tilde{S}' y^2 - z^2 = 0$, where T is one of $\tilde{R}_{i,0} \cdot P_{j,0}$, $\tilde{R}_{i,1}$, or $P_{j,1}$. Afterwards, these three solutions can be combined to obtain a solution to $\tilde{R}'_i x^2 + \tilde{S}' y^2 = 1 \bmod N$ for $\tilde{R}'_i \leftarrow \tilde{R}_i \cdot P_j$ by using the product formula (Lemma 5.1) twice. Multiplying x by p_j and y by p_k gives a solution to $R_i x^2 + S y^2 = 1 \bmod N$, after which encryption and decryption proceed in the usual way. The remaining cases for $(\tilde{R}_{i,0}/\tilde{S})$ and $(\tilde{R}_{i,1}/\tilde{S})$ are similar.

Empirically, the time needed to reduce the lattice associated to the equation $Tx^2 + \tilde{S}' y^2 - z^2 = 0$ is dictated by the smaller of the two values T and $\tilde{S}'$. Since lattice reduction takes cubic time, and since T is $(\log N)$ bits once, and $(\log N)/2$ bits twice, in the three equations above, the three lattice reductions take only about 1.25 times as long as one "normal" lattice reduction in our system. So, aside from the generation of $\tilde{S}$, this approach allows significantly (8 times) faster prime generation almost for free.

Setting $k > 2$ is an interesting possibility, but it is unclear how to make the simulation efficient. Of course, the simulator could generate $\tilde{R}_i$ as follows: generate random $r_i \overset{\text{R}}{\leftarrow} \mathbb{Z}/N\mathbb{Z}$ and $a_i \overset{\text{R}}{\leftarrow} \{0,1\}$, set $\tilde{R}_i \leftarrow u^{a_i} r_i^2$, and try to factor $\tilde{R}_i$, hoping that it is (roughly speaking) $N^{1/k}$-smooth. But this leads to an extremely loose reduction.

6 Anonymous IBE and security without random oracles

The product formula (Lemma 5.1) allows us to quickly compute $\mathcal{Q}(N, uR_j, S)$ from $\mathcal{Q}(N, R_j, S)$. We show that for these derived solutions the decryptor can deduce the ciphertext bit $\bar{c}_j$ from the bit c_j. As a result, the bits $(\bar{c}_1, \ldots, \bar{c}_\ell)$ need not be included in the ciphertext. By eliminating the extra ciphertext bits we obtain an anonymous IBE and we can prove its security based on the IQR assumption *without* the random oracle model.

As in Section 3 we begin by stating the abstract properties needed for this construction. We then give an example instantiation and describe the resulting IBE system.

Definition 6.1. Let $\mathcal{Q}'$ be a deterministic algorithm that takes as input (N, u, R, S) where $N \in \mathbb{Z}^+$ and $u, R, S \in \mathbb{Z}/N\mathbb{Z}$. The algorithm outputs polynomials $f, \bar{f}, g, \tau \in \mathbb{Z}/N\mathbb{Z}[x]$. We say that $\mathcal{Q}'$ is **Enhanced IBE Compatible** if the following conditions hold:

- (Condition 1a) If R and S are quadratic residues, then $f(r)g(s)$ is also a quadratic residue for all square roots r of R and s of S.
- (Condition 1b) If uR and S are quadratic residues, then $\bar{f}(\bar{r})g(s)\tau(s)$ is also a quadratic residue for all square roots $\bar{r}$ of uR and s of S.
- (Condition 2a) If R is a quadratic residue, then $f(r)f(-r)S$ is also a quadratic residue for all square roots r of R.
- (Condition 2b) If uR is a quadratic residue, then $\bar{f}(\bar{r})\bar{f}(-\bar{r})S$ is also a quadratic residue for all square roots $\bar{r}$ of uR.
- (Condition 2c) If S is a quadratic residue, then $\tau(s)\tau(-s)u$ is also a quadratic residue for all square roots s of S.
- (Condition 3) τ is independent of R, that is, $\mathcal{Q}'(N, u, R_1, S)$ and $\mathcal{Q}'(N, u, R_2, S)$ produce the same τ for all N, u, R_1, R_2, S.

6.1 An enhanced IBE compatible example

As an instance of such a $\mathcal{Q}'$, we proceed from Section 5.1 as follows. Algorithm $\mathcal{Q}'(N, u, R, S)$:

```
1.  Construct a solution $(x, y) \in (\mathbb{Z}/N\mathbb{Z})^2$ to the equation $Rx^2 + Sy^2 = 1$
2.  Construct a solution $(\alpha, \beta) \in (\mathbb{Z}/N\mathbb{Z})^2$ to the equation $u \cdot \alpha^2 + S \cdot \beta^2 = 1$
3.  Output the polynomials:
        $f(r) \leftarrow xr + 1$          $\bar{f}(\bar{r}) \leftarrow 1 + Sy\beta + \alpha x\bar{r}$
        $g(s) \leftarrow 2ys + 2$         $\tau(s) \leftarrow 1 + \beta s.$
```

This satisfies Conditions 1a, 2a and 2c as in Section 4, and trivially satisfies Condition 3. As for Conditions 1b and 2b, we note that, per Lemma 5.1, $\bar{x} \leftarrow \alpha x/(Sy\beta+1)$ and $\bar{y} \leftarrow (y+\beta)/(Sy\beta+1)$ satisfy $uR \cdot \bar{x}^2 + S \cdot \bar{y}^2 = 1$, so that when uR and S are quadratic residues:

$$\begin{aligned}
\bar{f}(\bar{r})g(s)\tau(s) &= (1+Sy\beta+\alpha x\bar{r}) \cdot (2+2ys) \cdot (1+\beta s) \\
&= 2 \cdot (1+Sy\beta+\alpha x\bar{r}) \cdot (1+Sy\beta+ys+\beta s) \\
&= 2 \cdot \left(1+\bar{r}\frac{\alpha x}{Sy\beta+1}\right) \cdot \left(1+s\frac{y+\beta}{Sy\beta+1}\right) \cdot (Sy\beta+1)^2 \\
&= 2 \cdot (1+\bar{r}\bar{x}) \cdot (1+s\bar{y}) \cdot (Sy\beta+1)^2
\end{aligned}$$

This is a quadratic residue: $2 \cdot (1+\bar{r}\bar{x}) \cdot (1+s\bar{y})$ is a quadratic residue as in Section 4, and so is $(Sy\beta+1)^2$ by definition. Similarly, if uR is a quadratic residue, then

$$\bar{f}(\bar{r})\bar{f}(-\bar{r})S = (Sy\beta+1)^2(1+\bar{r}\bar{x})(1-\bar{r}\bar{x})S$$

is a quadratic residue as in Section 4.

6.2 The modified IBE system

With an Enhanced IBE Compatible $\mathcal{Q}'$ we construct an IBE, which we call AnonIBE. Its *Setup* and *KeyGen* algorithms are the same as in the simpler system of Section 3.1.

Encrypt$(\mathrm{PP}, \mathrm{ID}, m)$: To encrypt a message $m = m_1, m_2, \ldots, m_\ell \in \{\pm 1\}^\ell$ to some ID, pick a random $s \in \mathbb{Z}/N\mathbb{Z}$ and compute $S \leftarrow s^2 \bmod N$. Run $\mathcal{Q}'(N, u, 1, S)$ to obtain the polynomial τ, and compute $k \leftarrow \left(\frac{\tau(s)}{N}\right)$. Then for $j \in [1, \ell]$, set $R_j \leftarrow H(\mathrm{ID}, j)$, run $\mathcal{Q}'(N, u, R_j, S)$ to obtain g_j, and compute $c_j \leftarrow m_j \cdot \left(\frac{g_j(s)}{N}\right)$. Set $c = c_1, \ldots, c_\ell$; the ciphertext is then (S, k, c). The ciphertext size is now $(\log_2 N + \ell + 1)$ bits which is $\ell - 1$ bits shorter than before.

Decrypt(C, d_{ID}): To decrypt a ciphertext $C = (S, k, c)$ with a private key $d_{\mathrm{ID}} = (r_1, \ldots, r_\ell)$, for $j = 1, \ldots, \ell$ the receiver sets $R_j \leftarrow H(\mathrm{ID}, j)$ and runs $\mathcal{Q}'(N, u, R_j, S)$ to obtain f_j and $\bar{f}_j$.

$$\text{if } r_j^2 = R_j \;\text{ set }\; m_j \leftarrow c_j \cdot \left(\frac{f_j(r_j)}{N}\right) \qquad ; \qquad \text{if } r_j^2 = uR_j \;\text{ set }\; m_j \leftarrow c_j \cdot k \cdot \left(\frac{\bar{f}_j(r_j)}{N}\right)$$

Output $m \leftarrow m_1 \ldots m_\ell$. Conditions 1a, 1b and 3 above guarantee that this correctly recovers the message.

Security We prove that the system is ANON-IND-ID-CPA based on the IQR assumption.

Theorem 6.2. *Let $\mathcal{A}$ be an efficient ANON-IND-ID-CPA adversary against* AnonIBE. *Then there exist efficient algorithms $\mathcal{B}_1, \mathcal{B}_2$, running in about the same time as $\mathcal{A}$, such that*

$$\mathrm{IBEAdv}_{\mathcal{A},\mathsf{AnonIBE}}(\lambda) \leq \mathrm{PRFAdv}_{\mathcal{B}_1,F}(\lambda) + \mathrm{IQRAdv}_{\mathcal{B}_2,(RSAgen,H)}(\lambda)$$

Proof. Let $\mathcal{A}$ be an ANON-IND-ID-CPA adversary attacking the IBE system AnonIBE. We wish to bound its advantage $\mathrm{IBEAdv}_{\mathcal{A},\mathsf{AnonIBE}}(\lambda)$. We present the proof as a series of games. We let W_i denote the event that the adversary wins Game i.

Game 0. The first game is identical to the $\mathsf{ANON\text{-}IND\text{-}ID\text{-}CPA}$ game defined in Section 2. Hence, we know that

$$\left|\Pr[W_0] - \frac{1}{2}\right| = \mathrm{IBEAdv}_{\mathcal{A},\mathsf{AnonIBE}}(\lambda) \tag{8}$$

Game 1. In Game 1 we slightly modify the challenger in Game 0. Instead of using a PRF F_K to respond to $\mathcal{A}$'s private key queries we use a truly random function $f : \mathcal{ID} \times [1, \ell] \to \{0, 1, 2, 3\}$. Clearly, if F is a secure PRF, $\mathcal{A}$ will not notice the difference between Game 0 and Game 1. In particular, there exists an algorithm $\mathcal{B}_1$ (whose running time is about the same as that of $\mathcal{A}$) such that

$$|\Pr[W_1] - \Pr[W_0]| = \mathrm{PRFAdv}_{\mathcal{B}_1,F}(\lambda) \tag{9}$$

Game 2. In Game 2, we modify the challenger so that it does not need the factorization of N. We will work in the setting of the IQR assumption, so we assume a randomly chosen square root oracle $\mathcal{O}$ with its associated quadratic non-residue u as in Definition 2.3. As before, the challenger chooses $p, q \overset{\mathrm{R}}{\leftarrow} \mathrm{RSAgen}(\lambda)$ and $N \leftarrow pq$, but now it does not need the factorization during the game.

The public parameters are (N, u, H); they are distributed as in Game 1. The challenger responds to private key queries using the oracle $\mathcal{O}$. Given a private key query for ID the challenger must return ℓ values $r_j \in \mathbb{Z}/N\mathbb{Z}$ where $r_j^2 = u^a H(\mathrm{ID}, j)$ for some $a \in \{0, 1\}$. This is precisely what the oracle does, enabling the challenger to respond to private key queries. The encryption query is unchanged.

Because the square root oracle is chosen at random (replacing the random function f), the distributions of query responses are identical to those in Game 1. Therefore,

$$\Pr[W_2] = \Pr[W_1] \tag{10}$$

Game 3. Game 3 proceeds exactly as Game 2 except that the encryption query is handled differently. To encrypt a message m_b to a particular ID_b, the challenger does:

$$\begin{array}{ll} & \text{construct the private key } d_{\mathrm{ID}_b} = (\mathrm{PP}, r_1, \ldots, r_\ell) \text{ using } \mathcal{O} \\ (*) & \text{choose } s \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}/N\mathbb{Z} \ \text{ and set } \ S \leftarrow s^2 \\ & \text{choose } k \overset{\mathrm{R}}{\leftarrow} \{\pm 1\}, \ \text{ and set } \ c_b \leftarrow \mathit{Decrypt}\big(d_{\mathrm{ID}_b},\ (S, k, m_b)\big) \end{array}$$

Send the challenge ciphertext (S, k, c_b) to the adversary.

This process produces the same distribution for (S, k, c_b) as the real *Encrypt* in Game 2. Indeed, in Game 2 the quantity S is uniformly distributed among quadratic residues, and by Lemma 3.3, the bit k is independently uniformly distributed in $\{\pm 1\}$. Furthermore, c_b is the unique string such that (S, k, c_b) is an encryption of m_b. Hence, (S, k, c_b) in Game 3 is distributed as in Game 2 and therefore

$$\Pr[W_3] = \Pr[W_2] \tag{11}$$

Game 4. Game 4 proceeds exactly as Game 3 except that when creating the challenge ciphertext the challenger chooses S as a random non-residue. That is, we change the line marked $(*)$ in

Game 3 to $S \overset{\mathrm{R}}{\leftarrow} J(N)\backslash \mathrm{QR}(N)$. Since this is the only change, under the IQR assumption, Game 4 will be indistinguishable from Game 3. More precisely, there exists an algorithm $\mathcal{B}_2$ whose running in about the same time as $\mathcal{A}$, such that

$$|\Pr[W_3] - \Pr[W_4]| = \mathrm{IQRAdv}_{\mathcal{B}_2,(\mathrm{RSAgen},H)}(\lambda) \tag{12}$$

We claim that the adversary wins Game 4 exactly half the time. We show that in Game 4 the challenge ciphertext (S, k, c) is random and independent of b. The values S and k are clearly independent of b. To show that $c = c_1 \dots c_\ell$ are independent of b, consider the jth bit c_j. Since $\mathcal{O}$ was chosen at random, in the adversary's view, r_j is uniformly distributed among the four square roots of $R_j = H(\mathrm{ID}_b, j)$ or uR_j. Hence, since S is not a quadratic residue, Lemma 3.3 tells us that $(f_j(r_j)/N)$ or $(\bar{f}_j(r_j)/N)$ is uniform in $\{\pm 1\}$. Either way, the bit c_j is uniform in $\{\pm 1\}$ and independent of b. Overall, (S, k, c) is independent of b and hence

$$Pr[W_4] = \frac{1}{2} \tag{13}$$

Combining equations (8) through (13), we obtain:

$$\mathrm{IBEAdv}_{\mathcal{A},\mathsf{AnonIBE}}(\lambda) \leq \mathrm{PRFAdv}_{\mathcal{B}_1,F}(\lambda) + \mathrm{IQRAdv}_{\mathcal{B}_2,(\mathrm{RSAgen},H)}(\lambda)$$

This is the desired bound. □

Note that the IQR assumption follows from the QR assumption where H (i.e., H_N) is a full-domain hash function modeled as a random oracle. The challenger picks some $v \in J(n)$, and can replace calls to $H_N(X)$ by choosing $x \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}/N\mathbb{Z}, a \overset{\mathrm{R}}{\leftarrow} \{0, 1\}$, and returning $H_N(X) = v^{-a}x^2$. When queried for $\mathcal{O}(N, X)$, it returns x. If v is not a quadratic residue, this gives a properly random distribution of H_N. If v is a quadratic residue, the results are still indistinguishable by the QR assumption. In summary, AnonIBE is secure in the standard model under the IQR assumption, and in the random oracle model under the QR assumption.

7 Extensions and Open Problems

7.1 Chosen ciphertext security

In Appendix D, we review a generic construction that converts an IND-ID-CPA IBE system into an IND-ID-CCA IBE in the random oracle model [4], and discuss how it applies to our scheme.

More importantly, in Appendix E we build an ANON-IND-ID-CCA IBE system in the standard model. First, we describe a generic ANON-IND-ID-CCA system that uses hash proof systems (as defined in [19]) for a subset membership problem that has a trapdoor permitting private key generation. We prove this system secure under the *interactive subset membership* (ISM) assumption, a generic analogue to the IQR assumption. We then describe our hash proof systems for quadratic residuosity (employing the ideas described earlier in this paper) and show that these systems meet the requirements of our generic framework. As a result, we obtain an ANON-IND-ID-CCA-secure IBE system in the standard model under the IQR assumption. We also briefly describe an anonymous PKE system based on our hash proof systems that is anonymous in the multi-user setting and IND-CCA secure in the standard model under the (*non-interactive*) QR assumption.

7.2 Multivariate quadratics and higher residue symbols

A possible way to extend our IBE system is to use a higher-order character in place of the Jacobi symbol. This may improve efficiency by allowing the sender and receiver to derive more than one bit of shared secret data with each computation. Another possible extension is to use multivariate polynomials f and g instead of univariate ones. This may improve efficiency if it allows us to replace $\mathcal{Q}$ by a faster algorithm.

8 Conclusions

We described a space-efficient anonymous IBE system without pairings based on the quadratic residuosity assumption in the random oracle model. In the standard model, the system is secure under the interactive quadratic residuosity assumption. Our system is more space efficient than Cocks' IBE, but is less time efficient. Since the bottleneck in our system is due to algorithm $\mathcal{Q}$ it is natural to look for other instantiations of our abstract system that are more efficient. Another natural important problem is to design an IBE system secure without random oracles based on quadratic residuosity. Currently, both the Cocks system and ours need the random oracle model to argue that the underlying Rabin signature system is existentially unforgeable.

Acknowledgments

We thank Peter Montgomery for his comments on extensions to Section 4.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *CRYPTO*, pages 205–222, 2005.

[2] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *Proceedings of the IEEE Symposium and Security and Privacy*, pages 180–196, 2003.

[3] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[4] K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart. Generic constructions of identity-based and certificateless kems. `http://eprint.iacr.org/2005/058`.

[5] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of 2007 IEEE Symposium on Security and Privacy*, 2007.

[6] Dan Boneh and Xavier Boyen. Efficient selective-ID identity based encryption without random oracles. In *Proceedings of Eurocrypt 2004*, LNCS, pages 223–238. Springer-Verlag, 2004.

[7] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. of Computing (SICOMP)*, 36(5):915–942, 2006.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafial Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Proceedings of Eurocrypt '04*, 2004.

[9] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003. extended abstract in Crypto '01.

[10] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *Proceedings of FOCS'07*, 2007. full version at `http://eprint.iacr.org/2007/177`.

[11] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO '05*, pages 258–275, 2005.

[12] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Crypto '06*, 2006.

[13] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*. Springer, 2003.

[14] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Proceedings of Eurocrypt 2004*, LNCS, 2004.

[15] J. W. S. Cassels. *Rational quadratic forms*, volume 13 of *London Mathematical Society Monographs*. Academic Press, 1978.

[16] T. Cochrane and P. Woods. Small solutions of the legendre equation. *Journal of Number Theory*, 70(1):62–66, 1998.

[17] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 26–8, 2001.

[18] Jean-Sbastien Coron. Security proof for partial-domain hash signature schemes. In *Proceedings of Crypto '02*, LNCS, pages 613–626. Springer, 2002.

[19] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. In *Proceedings of Eurocrypt 2002*, 2002.

[20] J. Cremona and D. Rusin. Efficient solution of rational conics. *Math. of computation*, 72(243):1417–1441, 2003.

[21] Giovanni Di Crescenzo and Vishal Saraswat. Searchable public-key encryption based on jacobi symbols. manuscript, Feb. 2007.

[22] Y. Desmedt and A. Oldyzko. A chosen text attack on the rsa cryptosystem and some discrete logarithm schemes. In *Proceedings of Crypto '85*, volume 218 of *LNCS*, pages 516–521. Springer-Verlag, 1985.

[23] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[24] Craig Gentry. How to compress rabin ciphertexts and signatures (and more). In *Proceedings of Crypto '04*, pages 179–200, 2004.

[25] Craig Gentry. Practical identity-based encryption without random oracles. In *Proceedings of Eurocrypt '06*, volume 4004 of *LNCS*, pages 445–464, 2006.

[26] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Proceedings of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 548–66, 2002.

[27] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness.* Springer, 1999.

[28] L. Lovasz H. Lensta, L. Lensta. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[29] Jeremy Horwitz and Ben Lynn. Towards hierarchical identity-based encryption. In Lars Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 466–81. Springer, 2002.

[30] K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *Proceedings of Crypto '04*, volume 3152 of *LNCS*, pages 426–442. Springer, 2004.

[31] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In *In proceedings of PODC '03*, pages 182–189, 2003.

[32] K. Mccusker, N.O'Connor, and D. Diamond. Low-energy finite field arithmetic primitives for implementing security in wireless sensor networks. In *Proceedings of 2006 conference on Communications, Circuits and Systems*, 2006.

[33] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[34] H. Ong, C. P. Schnorr, and A. Shamir. An efficient signature scheme based on quadratic equations. In *Proceedings of STOC '84*, pages 208–216, 1984.

[35] J. M. Pollard and C. P. Schnorr. An efficient solution of the congruence $x^2 + ky^2 = m(modn)$. *IEEE Transactions on Information Theory*, 33(5):702–709, 1987.

[36] R. Sakai and M. Kasahara. ID based cryptosystems with pairing over elliptic curve. `http://eprint.iacr.org/2003/054`, 2003.

[37] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology—CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.

[38] Elaine Shi, John Bethencourt, T.-H. Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *Proceedings of the IEEE Symposium and Security and Privacy*, 2007.

[39] B. Vallee. Generation of elements with small modular squares and provably fast integer factoring algorithms. *Mathematics of Computation*, 56(194):823–849, 1991.

[40] B. Waters. Efficient identity-based encryption without random oracles. In *Proceedings of Eurocrypt 2005*, LNCS, 2005.

[41] Brent Waters, Dirk Balfanz, Glenn Durfee, and D. Smetters. Building an encrypted and searchable audit log. In *Proceedings of Network and Distributed System Security Symposium (NDSS 2004)*, 2004.

[42] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In Birgit Pfitzmann, editor, *Proceedings of the ACM Conference on Computer and Communications Security 2004*, pages 354–63, 2004.

A Review of Cocks IBE

The Cocks cryptosystem [17] depends on the sender and receiver knowing solutions to public quadratic equations mod a large composite number N. Specifically:

Setup(λ) generates $(p, q) \overset{\text{R}}{\leftarrow} \text{RSAgen}(\lambda)$, $N \leftarrow pq$, and a random $u \overset{\text{R}}{\leftarrow} J(N) \setminus QR(N)$. It outputs public parameters $\text{PP} = (N, u, H)$ where H is a hash function $H : \mathcal{ID} \to J(N)$. The master key MSK is the factorization of N.

KeyGen(MSK, ID) generates a private key by first setting $R \leftarrow H(\text{ID})$. If $R \in \text{QR}(N)$ it sets $r \leftarrow R^{1/2}$ and otherwise it sets $r \leftarrow (uR)^{1/2}$. It outputs r as the private key for ID.

Encrypt(PP, ID, m): Let $R \leftarrow H(\text{ID})$. To encrypt an ℓ-bit message $m = m_1 \cdots m_\ell$ choose 2ℓ random $t_{j,a} \in \mathbb{Z}/N\mathbb{Z}$ for $j \in \{1, \ldots, \ell\}$ and $a \in \{0, 1\}$. Compute

$$d_{j,a} \leftarrow \frac{t_{j,a}^2 + u^a R}{t_{j,a}} \qquad \text{and} \qquad c_{j,a} \leftarrow m_j \cdot \left(\frac{t_{j,a}}{N}\right)$$

Output the 2ℓ pairs $C \leftarrow \big(\ (d_{j,a}, c_{j,a})\ \big)$ for $j \in \{1, \ldots, \ell\}$ and $a \in \{0, 1\}$.

Decrypt(C, r): Set $a \in \{0, 1\}$ such that $r^2 = u^a R$. For $j = 1, \ldots, \ell$ set $g_j \leftarrow d_{j,a} + 2r$. Note that

$$g_j = \frac{(t_{j,a} + r)^2}{t_{j,a}} \qquad \text{and hence} \qquad \left(\frac{g_j}{N}\right) = \left(\frac{t_{j,a}}{N}\right)$$

Therefore the receiver can compute $m_j \leftarrow c_{j,a} \cdot \big(\frac{g_j}{N}\big)$ and outputs $m \leftarrow m_1 \cdots m_\ell$.

B Proof of Theorem 3.2

We prove security of the abstract multi-bit IBE system from Section 3.1. We prove Theorem 3.2 in two steps. First, we present a public key system, called BasicPKE, that is semantically secure in the standard model under the quadratic residuosity (QR) assumption. Then we deduce security of the IBE system; however this second step requires the random oracle model. We note that BasicPKE may be of independent interest since it gives a CCA-secure system in the standard model based on the QR assumption (see Appendix E).

B.1 The public key system BasicPKE.

Our public key system (PKE) allows multiple users to use the same modulus N, and we treat N as a common reference string. Hence, the PKE consists of four algorithms: a setup algorithm G to generate the common reference string $\mathcal{C}$, a key generation algorithm K to generate private/public key pairs, and algorithms E, D to encrypt/decrypt. Note that if for some reason one does not want all users to share the same N then the pair of algorithms (G, K) can be viewed as a single key generation algorithm.
Algorithms $\mathcal{E} = (G, K, E, D)$ work as follows:

Algorithm $G(\lambda)$: generate $(p, q) \overset{\text{R}}{\leftarrow} \text{RSAgen}(\lambda)$, $\;N \leftarrow pq$, and output N as the common reference string. The factorization of N is erased.

Algorithm $K(N, \ell)$: Takes as input the common reference string N and a message length parameter ℓ. For $j = 1, \ldots, \ell$ it picks a random $r_j \overset{\text{R}}{\leftarrow} \mathbb{Z}/N\mathbb{Z}$ and sets $R_j \leftarrow r_j^2$. It then outputs the public key $\text{PK} \leftarrow (R_1, \ldots, R_\ell)$ and private key $\text{SK} \leftarrow (N, r_1, \ldots, r_\ell)$.

Algorithm $E(N, \text{PK}, m)$: Let $\text{PK} = (R_1, \ldots, R_\ell)$ and $m = m_1 \ldots m_\ell \in \{-1, 1\}^\ell$. The algorithm picks a random $s \in \mathbb{Z}/N\mathbb{Z}$ and sets $S \leftarrow s^2$. For $j = 1, \ldots, \ell$, it does:

$$(f_j, g_j) \leftarrow \mathcal{Q}(N, R_j, S) \quad \text{and} \quad c_j \leftarrow m_j \cdot \left(\frac{g_j(s)}{N}\right)$$

Set $c \leftarrow c_1 \cdots c_\ell$ and output the ciphertext $C \leftarrow (S, c)$.

Algorithm $D(\text{SK}, C)$: Let $\text{SK} = (N, r_1, \ldots, r_\ell)$ and $C = (S, c_1, \ldots, c_\ell)$. For $j = 1, \ldots, \ell$ do:

$$R_j \leftarrow r_j^2\ , \quad (f_j, g_j) \leftarrow \mathcal{Q}(N, R_j, S)\ , \quad m_j \leftarrow c_j \cdot \left(\frac{f_j(r_j)}{N}\right)$$

Output $m \leftarrow m_1 \ldots m_\ell$.

This completes the description of the PKE system BasicPKE. Soundness follows from condition (1) of Definition 3.1.

Security of BasicPKE. We now prove that BasicPKE is semantically secure *without* random oracles. The standard semantic security game proceeds by the challenger first running $G(\lambda)$ to generate the common reference string $\mathcal{C}$. It then runs $K(\mathcal{C}, \ell)$ to generate a public/private key pair (PK, SK) for ℓ-bit messages. It sends $(\mathcal{C}, \text{PK})$ to $\mathcal{A}$. Finally, it picks a random $b \overset{\text{R}}{\leftarrow} \{0, 1\}$. Next, $\mathcal{A}$ gives the challenger two equal length messages $m^{(0)}, m^{(1)}$ and gets back a challenge ciphertext $C^* \leftarrow E(\mathcal{C}, \text{PK}, m^{(b)})$. Finally $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for the bit b. We say that $\mathcal{A}$ wins the game if $b = b'$. We refer to such adversary $\mathcal{A}$ as an IND-CPA adversary. We define the adversary's advantage in attacking the PKE scheme $\mathcal{E}$ as

$$\text{PKEAdv}_{\mathcal{E},\mathcal{A}}(\lambda) = \left|\Pr[b = b'] - \frac{1}{2}\right|$$

The probability is over the random bits used by the challenger and the adversary.

Definition B.1. We say that a PKE system $\mathcal{E}$ is $\mathsf{IND\text{-}CPA}$ secure if for all polynomial time adversaries $\mathcal{A}$ we have that $\mathrm{PKEAdv}_{\mathcal{E},\mathcal{A}}$ is a negligible function.

We can now state and prove the security theorem of $\mathsf{BasicPKE}$.

Lemma B.2. *The PKE system* $\mathsf{BasicPKE} = (G, K, E, D)$ *is* $\mathsf{IND\text{-}CPA}$ *secure in the standard model if the QR assumption holds for RSAgen. In particular, suppose* $\mathcal{A}$ *is a polynomial time* $\mathsf{IND\text{-}CPA}$ *adversary attacking* $\mathsf{BasicPKE}$*. Then there exists an efficient QR algorithm* $\mathcal{B}$ *(whose running time is about the same as that of* $\mathcal{A}$*) such that*

$$\mathrm{PKEAdv}_{\mathcal{A},\mathsf{BasicPKE}}(\lambda) = \mathrm{QRAdv}_{\mathcal{B},RSAgen}(\lambda).$$

Proof. The proof is by direct reduction to the QR assumption. Algorithm $\mathcal{B}$ is given a random tuple (N, V) where $N = pq,\ (p, q) \xleftarrow{\mathrm{R}} \mathrm{RSAgen}(\lambda)$ and $V \in J(N)$. It tries to output 1 when $V \in \mathrm{QR}(N)$ and 0 otherwise. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and plays the role of challenger to $\mathcal{A}$. First, $\mathcal{B}$ sets N as the common reference string and runs $K(N, \ell)$ to obtain a public/private key pair $(\mathrm{PK}, \mathrm{SK})$ for ℓ-bit messages. It gives $\mathcal{A}$ the pair (N, PK).

Adversary $\mathcal{A}$ now responds with two equal length messages $m^{(0)}, m^{(1)} \in \{\pm 1\}^{\ell}$. Now $\mathcal{B}$ creates a challenge ciphertext C^*. It chooses a random $b \xleftarrow{\mathrm{R}} \{0, 1\}$ and creates an encryption of $m^{(b)}$ using the given V as follows:

- Let $\mathrm{PK} = (R_1, \ldots, R_\ell)$ and let $\mathrm{SK} = (r_1, \ldots, r_\ell)$.
- Write $m^{(b)} = m_1 \ldots m_\ell \in \{\pm 1\}^{\ell}$. For $u = 1, \ldots, \ell$, do:

$$(f_u, g_u) \leftarrow \mathcal{Q}(N, R_u, V) \quad \text{and} \quad c_u \leftarrow m_u \cdot \left(\frac{f_u(r_u)}{N}\right)$$

  Set $c \leftarrow c_1 \cdots c_\ell$ and send $\mathcal{A}$ the challenge ciphertext $C^* \leftarrow (V, c)$.

$\mathcal{A}$ now outputs a guess $b' \in \{0, 1\}$ for b. If $b = b'$ then $\mathcal{B}$ outputs 1. Otherwise it outputs 0. This completes the description of $\mathcal{B}$.

We argue that $\mathcal{B}$ breaks the QR assumption with the same advantage as $\mathcal{A}$ breaking the public key system. This will follow from the the following two claims.

Claim 1: When $(N, V) \xleftarrow{\mathrm{R}} \mathcal{P}_{QR}$ (i.e. V is uniformly distributed in $\mathrm{QR}(N)$) then

$$\left|\Pr[\mathcal{B}(N, V) = 1] - \frac{1}{2}\right| = \left|\Pr[b = b'] - \frac{1}{2}\right| = \mathrm{PKEAdv}_{\mathcal{A},\mathsf{BasicPKE}}(\lambda) \tag{14}$$

Proof. When (N, V) are distributed in $\mathcal{P}_{QR}$ then $\mathcal{B}$ emulates perfectly an $\mathsf{IND\text{-}CPA}$ challenger. The reference string N and public key PK are as in the real game. To see that the challenge ciphertext C^* is distributed as in the real attack game, let v be a square root of V, then by condition (1) of Definition 3.1 we have $(g_u(v)/N) = (f_u(r_u)/N)$ for all $u = 1, \ldots, \ell$, as required. Hence, the claim follows. □

Claim 2: When $(N, V) \xleftarrow{\mathrm{R}} \mathcal{P}_{NQR}$ (i.e. V is uniformly distributed in $J(N) \setminus \mathrm{QR}(N)$) then

$$\Pr[\mathcal{B}(N, V) = 1] = \Pr[b = b'] = \frac{1}{2} \tag{15}$$

Proof. When (N, V) are distributed in $\mathcal{P}_{NQR}$ we claim that the bit b is independent of $\mathcal{A}$'s view. In particular, the challenge ciphertext $C^* = (V, (c_1 \cdots c_\ell))$ is independent of b. To see why, consider the bit $c_u = m_u \cdot (f_u(r_u)/N) \in \{\pm 1\}$ for $u = 1, \ldots, \ell$. Recall that $\mathrm{PK} = (R_1, \ldots, R_\ell)$. The only information $\mathcal{A}$ has about r_u is the value $R_u \in \mathbb{Z}/N\mathbb{Z}$. Hence, from $\mathcal{A}$'s view, r_u is uniformly distributed in the set of four square roots of R_u. Since $V \in J(N) \setminus \mathrm{QR}(N)$, it follows by Condition (2) of Definition 3.1 and Lemma 3.3 that the symbol $\left(\frac{f_u(r_u)}{N}\right)$ is uniformly distributed in $\{\pm 1\}$. Hence, c_u is independent of m_u. The same argument holds for all $j = 1, \ldots, \ell$ and therefore C^* is independent of the message being encrypted. Overall, when (N, V) are distributed in $\mathcal{P}_{NQR}$ we have $\Pr[\mathcal{B}(N, V) = 1] = \Pr[b = b'] = \frac{1}{2}$. □

By combining (14) and (15) we obtain $\mathrm{QRAdv}_{\mathcal{B},\mathrm{RSAgen}}(\lambda) = \mathrm{PKEAdv}_{\mathcal{A},\mathsf{BasicPKE}}(\lambda)$ as required. This completes the proof of Lemma B.2. □

B.2 Security of the IBE system

Now, we turn to the proof of Theorem 3.2.

Proof. Let $\mathcal{A}$ be an $\mathsf{IND\text{-}ID\text{-}CPA}$ adversary attacking the IBE system $\mathsf{BasicIBE}$. We show that under the hypothesis of the theorem, $\mathrm{IBEAdv}_{\mathcal{A},\mathsf{BasicIBE}}(\lambda)$ is a negligible function. We present the proof as a sequence of games. We let W_i denote the event that the adversary wins game i.

Game 0. The first game is identical to the $\mathsf{IND\text{-}ID\text{-}CPA}$ game defined in Section 2. Hence, we know that

$$\left|\Pr[W_0] - \frac{1}{2}\right| = \mathrm{IBEAdv}_{\mathcal{A},\mathsf{BasicIBE}}(\lambda) \tag{16}$$

The challenger picks the random oracle $H : \mathcal{ID} \times [1, \ell] \to J(N)$ at random from the set of all such functions and allows $\mathcal{A}$ to query H at arbitrary points.

Game 1. We slightly modify the challenger in Game 0. Instead of using a PRF to respond to $\mathcal{A}$'s private key queries we use a truly random function $f : \mathcal{ID} \times [1, \ell] \to \{0, 1, 2, 3\}$. Clearly, if F is a secure PRF, $\mathcal{A}$ will not notice the difference between Game 0 and Game 1. In particular, there exists an algorithm $\mathcal{B}_1$ (whose running time is about the same as that of $\mathcal{A}$) such that

$$|\Pr[W_1] - \Pr[W_0]| = \mathrm{PRFAdv}_{\mathcal{B}_1,F}(\lambda) \tag{17}$$

Game 2. Recall that in Game 1 the public parameters PP given to $\mathcal{A}$ contain (N, u, H) where u is uniform in $J(N) \setminus \mathrm{QR}(N)$. Furthermore, in Game 1 the random oracle H is a random function $H : \mathcal{ID} \times [1, \ell] \to J(N)$. In Game 2 we slightly change H. The challenger responds to a query to $H(\mathrm{ID}, j)$ by picking random $a \overset{\mathrm{R}}{\leftarrow} \{0, 1\}$ and $v \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}/N\mathbb{Z}$ and settings $H(\mathrm{ID}, j) = u^a \cdot v^2$. Clearly this challenger implements a random function $H : \mathcal{ID} \times [1, \ell] \to J(N)$ as in Game 1.

Let $R_j \leftarrow H(\mathrm{ID}, j)$ for some (ID, j). Recall that in Game 1 the challenger responds to a private key query for ID by outputting a *random* square root of R_j or uR_j for $j = 1, \ldots, \ell$. Indeed, the value $f(R_j)$, which selects which square root to output, is uniform in the set $\{0, 1, 2, 3\}$.

In Game 2 let $R_j \leftarrow H(\mathrm{ID}, j) = u^a \cdot v^2$ for some (a, v). The challenger responds to a private key query for ID by outputting either $R_j^{1/2} = v$ (used if $a = 0$) or $(uR_j)^{1/2} = uv$ (used if $a = 1$)

for $j = 1, \ldots, \ell$. Since v is uniform in $\mathbb{Z}/N\mathbb{Z}$ this will produce a square root of R_j or uR_j that is uniformly chosen from amongst the four square roots, as in Game 1. In summary, $\mathcal{A}$'s view in games 1 and 2 is identical and therefore,

$$\Pr[W_2] = \Pr[W_1] \tag{18}$$

Note that in Game 2 the challenger no longer needs the factorization of N to respond to $\mathcal{A}$'s queries.

Game 3. We slightly modify the challenger in Game 2 by choosing u uniformly in $\mathrm{QR}(N)$ instead of in $J(N) \setminus \mathrm{QR}(N)$. Since this is the only change between games 2 and 3, adversary $\mathcal{A}$ will not notice the difference assuming the QR assumption holds for RSAgen. In particular, there exists an algorithm $\mathcal{B}_2$ (whose running time is about the same as that of $\mathcal{A}$) such that

$$|\Pr[W_3] - \Pr[W_2]| = \mathrm{QRAdv}_{\mathcal{B}_2,\mathrm{RSAgen}}(\lambda) \tag{19}$$

We note that since the random oracle is defined as $H(\mathrm{ID}, j) = u^a \cdot v^2$ and $u \in \mathrm{QR}(N)$ we now know that $H(\mathrm{ID}, j)$ always outputs elements in $\mathrm{QR}(N)$. From here on we let u_0 be a root of u.

Game 4. We slightly change how the challenger picks the challenge ciphertext C^*. We pick C^* as in the proof of Lemma B.2. To respond to the encryption query $(\mathrm{ID}, m^{(0)}, m^{(1)})$ from $\mathcal{A}$ the challenger chooses $b \xleftarrow{\mathrm{R}} \{0, 1\}$ and does:

$R_i \leftarrow H(\mathrm{ID}, i) = u^{a_i} \cdot v_i^2 \quad$ and $\quad r_i \leftarrow u_0^{a_i} \cdot v_i \quad$ for $i = 1, \ldots, \ell$
(then r_i is a root of R_i and $u_0 r_i$ is a root of uR_i)

$(*)\qquad s \xleftarrow{\mathrm{R}} \mathbb{Z}/N\mathbb{Z} \quad$ and $\quad S \leftarrow s^2$

write $m^{(b)} = m_1 \ldots m_\ell \in \{\pm 1\}^\ell$

for $k = 1, \ldots, \ell$, do:

$\qquad (f_k, g_k) \leftarrow \mathcal{Q}(N, R_k, S) \quad$ and $\quad (\bar{f}_k, \bar{g}_k) \leftarrow \mathcal{Q}(N, uR_k, S)$

$(**)\qquad c_k \leftarrow m_k \cdot \left(\frac{f_k(r_k)}{N}\right) \quad$ and $\quad \bar{c}_k \leftarrow m_k \cdot \left(\frac{\bar{f}_k(u_0 r_k)}{N}\right)$

$c \leftarrow (c_1 \cdots c_\ell)$ and $\bar{c} = (\bar{c}_1 \cdots \bar{c}_\ell)$. Send $\mathcal{A}$ the challenge ciphertext $C^* \leftarrow (S, c, \bar{c})$

Since S, R_k, uR_k are all in $\mathrm{QR}(N)$ we know by condition (1) that $(f_k(r_k)/N) = (g_k(s)/N)$ for all $k = 1, \ldots, \ell$ and that the same holds for $(\bar{f}_k, \bar{g}_k)$. Hence, the ciphertext C^* created in this way is identical to the challenge ciphertext created in Game 3. Therefore,

$$\Pr[W_3] = \Pr[W_4] \tag{20}$$

It is important to note that s is not used in the creation of C^*.

Game 5. Game 5 is identical to Game 4 except that the S used to create the challenge ciphertext C^* is chosen uniformly in $J(N) \setminus \mathrm{QR}(N)$ instead of $\mathrm{QR}(N)$. That is we change the line marked with a $(*)$ in Game 4 to be

$(*)\qquad S \xleftarrow{\mathrm{R}} J(N) \setminus \mathrm{QR}(N)$

Since this is the only change from game 4 to game 5, adversary $\mathcal{A}$ will not notice the difference, assuming the QR assumption holds for RSAgen. In particular, there exists an algorithm $\mathcal{B}_2'$ (whose running time is about the same as that of $\mathcal{A}$) such that

$$|\Pr[W_5] - \Pr[W_4]| = \mathrm{QRAdv}_{\mathcal{B}_2',\mathrm{RSAgen}}(\lambda) \tag{21}$$

Game 6. We can now change Game 5 and make the challenge ciphertext C^* be independent of the challenge bit b. We change the line marked $(**)$ in Game 4 as follows:

$$(**) \qquad z_k \xleftarrow{\text{R}} \{\pm 1\}, \qquad c_k \leftarrow z_k \cdot \left(\frac{f_k(r_k)}{N}\right) \quad \text{and} \quad \bar{c}_k \leftarrow z_k \cdot \left(\frac{\bar{f}_k(u_0 r_k)}{N}\right)$$

As a result, the challenge ciphertext C^* is an encryption of a random message $z_1 \dots z_\ell$, independent of the bit b.

We argue that because S is a non-residue, Games 5 and 6 are indistinguishable due to Condition (2) of Definition 3.1 and Lemma 3.3. The argument is similar to the argument in the proof of Lemma B.2. The challenge ciphertext is created using 2ℓ elements $\{R_1, uR_1, \dots, R_\ell, uR_\ell\}$ all in $\text{QR}(N)$. For each R in $\{R_1, \dots, R_\ell\}$ the adversary does not know which of the four square roots of R is used in the creation of C^*.

Consider a specific k in $\{1, \dots, \ell\}$ and let $x = \left(\frac{f_k(r_k)}{N}\right)$ and $y = \left(\frac{\bar{f}_k(u_0 r_k)}{N}\right)$. Using Condition (2) of Definition 3.1 and the proof of Lemma 3.3 one can show that, when r_k is chosen at random from among the four square roots of R_k, the pair (x, y) satisfies

$$\Pr\big[(x,y) = (1,1)\big] = \Pr\big[(x,y) = (-1,-1)\big] \quad \text{and} \quad \Pr\big[(x,y) = (1,-1)\big] = \Pr\big[(x,y) = (-1,1)\big]$$

It follows that the pair (x, y), which is an encryption of $+1$, is distributed identically as the pair $(-x, -y)$, which is an encryption of -1. Hence, in the adversary's view, the bits $(c_k, \bar{c}_k)$ are distributed identically whether the plaintext is $+1$ or -1. Since this holds for all $k = 1, \dots, \ell$ it follows that C^* is distributed identically in Games 5 and 6. We obtain

$$\Pr[W_6] = \Pr[W_5] \tag{22}$$

End. Clearly in Game 6 we have

$$\Pr[W_6] = \frac{1}{2} \tag{23}$$

Now combining equations (16) through (23) proves the theorem. □

Remark 1: We point out that the reduction is tight, and in particular independent of the number of private key queries and oracle queries made by $\mathcal{A}$.

Remark 2: Recall that the element u in the public parameters is a random non-residue in $J(N)$. One might be tempted to set $u = -1$ and use a modulus $N = pq$ which is a Blum integer (namely, $p = q = 3 \bmod 4$). This, however, would ruin our proof. In the transition from Game 2 to Game 3 we switched u from a residue to a non-residue which caused the random oracle H to always output elements in $\text{QR}(N)$. This was necessary for a later part of the proof. Choosing $u = -1$ would preclude this step. We note however that the transition from Game 2 to Game 3 would go through with $u = -1$ if, in addition to QR, one also made the assumption that distinguishing $N = pq$ with $p = q = 3 \bmod 4$ from $p = q = 1 \bmod 4$ is intractable. Clearly we recommend using a random non-residue u rather than $u = -1$.

Remark 3: Key generation in our system outputs a *random* square root of $H(\text{ID}, j)$. The reason for outputting a random root is evident in the transition from Game 2 to Game 3. That transition would break if instead, the system used a deterministic square root algorithm (say by using $N = pq$ with $p = q = 3 \bmod 4$). Indeed, a powerful adversary could distinguish Game 2 from Game 3 by testing whether the "correct" square root was given as a response to a private key query. We note a modified proof can work with a deterministic square root algorithm, but the reduction is very loose.

C Solving $R \cdot x^2 + S \cdot y^2 = z^2$ in $\mathbb{Z}$

In section 5.2, we needed an algorithm to solve $R \cdot x^2 + S \cdot y^2 = z^2$. Here we describe such an algorithm. We are given integers R, S, r, s where

- $\gcd(R, S) = 1$, R is odd, and $S = 1 \bmod 4$,
- $r^2 = R \bmod S$ and $s^2 = S \bmod R$.

We show how to construct a solution $(x, y, z) \in \mathbb{Z}^3$ to

$$R \cdot x^2 + S \cdot y^2 - z^2 = 0 \tag{24}$$

using an algorithm described in [20]. The algorithm presented here is based on the proof of Legendre's theorem due to Mordell. Consider the three dimensional lattice L in $\mathbb{Z}^3$ containing all triples (x, y, z) satisfying

$$\begin{aligned} s \cdot y = z &\pmod{R} \\ r \cdot x = z &\pmod{S} \\ x = 0 &\pmod{2} \\ y = z &\pmod{2} \end{aligned}$$

It is easy to see that any point (x, y, z) in L satisfies

$$Rx^2 + Sy^2 - z^2 = 0 \pmod{4RS}$$

We define the norm $q(x, y, z) = |R|x^2 + |S|y^2 + z^2$ and look for the shortest vector $v \in L$ under the norm q. A simple determinant calculation (and an application Minkowski's theorem) shows that a vector $v = (x_0, y_0, z_0) \in L$ exists whose q-norm is less than $4RS$. Then $|Rx_0^2 + Sy_0^2 - z_0^2| < |4RS|$. But since $Rx_0^2 + Sy_0^2 - z_0^2 = 0 \bmod 4RS$, it follows that $Rx_0^2 + Sy_0^2 - z_0^2 = 0$ holds over the integers, as required.

To find the short vector $v \in L$ Cremona and Rusin [20] use the LLL algorithm [28]. They show that in three dimensions, one can easily derive the shortest vector from an LLL reduced basis. Cremona and Rusin present other algorithms for solving (24) that do not use LLL and are claimed to be twice as fast as the algorithm above. We also note that Cochrane and Mitchell [16] show that the lattice L without the 2-adic components already contains a solution to (24).

C.1 Light weight primality tests

We give more details regarding the heuristic light weight primality test discussed in Section 5.2. The idea is to implement the primality test via a deterministic square root algorithm. For convenience, we require that $\tilde{R} = 3 \bmod 4$ and $\tilde{S} = 5 \bmod 8$.

We first look for a (probable) prime along the $\tilde{S}$ arithmetic progression by sequentially testing if $2^{(\tilde{S}-1)/2} = \pm 1 \bmod \tilde{S}$ until we find the first such $\tilde{S}$. Once found, we treat $\tilde{S}$ as though it was prime. Next, we look for the first element along the $\tilde{R}$ arithmetic progression such that

$$s \leftarrow \tilde{S}^{(\tilde{R}+1)/4} \bmod \tilde{R} \quad \text{satisfies} \quad s^2 = \tilde{S} \bmod \tilde{R} \tag{25}$$

If $\tilde{R}$ and $(\frac{\tilde{S}}{\tilde{R}}) = 1$ then s will be a square root of $\tilde{S}$ modulo $\tilde{R}$. Finally, we compute the square root of $\tilde{R}$ modulo $\tilde{S}$ assuming $\tilde{S}$ is prime. Since $\tilde{S} = 5 \bmod 8$ the square root is one of

$$r_0 \leftarrow R_0^{(\tilde{S}+3)/8} \bmod \tilde{S} \quad \text{or} \quad r_1 \leftarrow R_0^{(\tilde{S}+3)/8} \cdot 2^{(\tilde{S}-1)/4} \bmod \tilde{S} \tag{26}$$

We test if $r_i^2 = \tilde{R} \bmod \tilde{S}$ for $i = 0, 1$. If not, then we start over by looking for the next prime along the $\tilde{S}$ arithmetic progression.

D IND-ID-CCA IBE With Random Oracles Based on QR

We can construct an IND-ID-CCA system FullIBE from BasicIBE in the random oracle model using a generic method, such as Construction 3 in [4]. That construction uses hash functions H_1 and H_2, modeled as random oracles, and combines them with a generic probabilistic IBE scheme $\mathcal{E}_{ID}$ with algorithms $Encrypt_{ID}(ID, m; r)$ and $Decrypt_{ID}(d_{ID}, C)$. It outputs an ID-KEM scheme $\mathcal{E}_{ID-KEM}$ where $Encrypt_{ID-KEM}$ outputs (k, C) with $C = Encrypt_{ID}(ID, m; H_1(m))$ and $k = H_2(m)$, and $Decrypt_{ID-KEM}(d_{ID}, C)$ first checks that $m \leftarrow Decrypt_{ID}(d_{ID}, C) \neq \perp$, then checks that $C = Encrypt_{ID}(ID, m; H_1(m))$, and finally outputs $k = H_2(m)$. In our system, the second check does not require a complete encryption; rather, the recipient merely checks that randomness $H_1(m)$ leads to the sender's choice of S. Bentahar et al. prove (in the random oracle model) that if $\mathcal{E}_{ID}$ is one-way, then $\mathcal{E}_{ID-KEM}$ is secure against adaptive chosen ciphertext attack.

E ANON-IND-ID-CCA IBE Without Random Oracles Based on Interactive Subset Membership Problems

In this section, we describe how to achieve ANON-IND-ID-CCA security for our IBE system.

E.1 Supplementary Definitions

First, we review Cramer and Shoup's universal hash proof framework [19].

Let X, Π, K and T be finite non-empty sets. Let $H = \{H_k : X \to \Pi\}_{k \in K}$ be a set of functions. Let $\alpha : K \to T$ be a function from K to T, which may be seen as a "projection."

Definition E.1 (Projective hash family (PHF)). Let H, K, X, Π, T, and α be as above. Then, for a given proper subset $L \subset X$, we refer to the tuple $\mathbf{H} = (H, K, X, L, \Pi, T, \alpha)$ as a projective hash family for (X, L) if for all $k \in K$, the action of H_k on L is determined by $\alpha(k)$ – i.e., given $\alpha(k)$ and $x \in L$, $H_k(x)$ is uniquely determined.

Definition E.2 (ϵ-universal). PHF $\mathbf{H}$ above is ϵ-universal if, for all $t \in T$, $x \in X \setminus L$ and $\pi \in \Pi$, it holds that

$$\Pr[H_k(x) = \pi \mid \alpha(k) = t] \leq \epsilon \ ,$$

i.e., the probability of guessing $H_k(x)$ from x and $\alpha(k)$ is at most ϵ.

Definition E.3 (ϵ-smooth)**.** PHF **H** above is ϵ-smooth if the statistical distance between the distributions of $(x, t, H_k(x))$ and (x, t, π) is at most ϵ when k, x, and π are chosen uniformly at random from K, $X \setminus L$, and Π respectively, and $t \leftarrow \alpha(k)$.

Definition E.4 (Strongly smooth)**.** PHF **H** above is strongly smooth if it is 0-smooth.

Definition E.5 (ϵ-universal$_2$)**.** PHF **H** above is ϵ-universal$_2$ if, for all $t \in T$, distinct $x, x^* \in X \setminus L$, and $\pi, \pi^* \in \Pi$, it holds that

$$\Pr[H_k(x) = \pi \mid H_k(x^*) = \pi^* \wedge \alpha(k) = t] \leq \epsilon \ ,$$

i.e., the probability of guessing $H_k(x)$ from x, $\alpha(k)$ and $H_k(x^*)$ is at most ϵ.

Note that if **H** is ϵ-universal$_2$, it is ϵ-universal$_2$.

Definition E.6 (Strongly universal$_2$)**.** PHF **H** above is strongly universal$_2$ if it is $(1/|\Pi|)$-universal$_2$.

Cramer and Shoup use universal hash proofs in connection with hard subset membership problems, defined as follows.

Definition E.7 (Subset Membership Problem)**.** A subset membership problem **M** specifies a collection $(D_\lambda)_{\lambda \geq 0}$ of distributions. For every value of the security parameter $\lambda \geq 0$, D_λ is a probability distribution of instance descriptions. We let $\mathrm{Gen}(\lambda)$ be an algorithm that generates a random instance description according to the distribution D_λ in time polynomial in λ. Each instance description Λ specifies the following:

- Finite non-empty sets X, L and W, with L a proper subset of X.
- A binary relation $\mathcal{R} \subset X \times W$.

M specifies some basic algorithms – i.e., for sampling instances, for generating an element $x \in L$ together with a witness w, and for checking that a bit string is an element of X. We say that **M** is a hard subset membership problem if it is hard to distinguish whether an element x has been chosen at random from L or from $X \setminus L$. The subset membership assumption for **M** is the assumption that **M** is a hard subset membership problem.

Definition E.8 (Hash Proof System)**.** A *hash proof system* (HPS) **P** for subset membership problem **M** associates with each instance $\Lambda[X, L, W, R]$ of **M** a PHF $H = (H, K, X, L, \Pi, T, \alpha)$ for (X, L). The HPS provides some basic algorithms – e.g., to sample $k \in K$ at random, to compute $\alpha(k)$ given $k \in K$, and to check that a bit string is an element of Π. The most important algorithms are the *private evaluation algorithm*, which allows one to efficiently compute $H_k(x)$ from the instance Λ, $k \in K$ and $x \in X$, and the *public evaluation algorithm*, which allows one to efficiently compute $H_k(x)$ if $x \in L$ when given the instance Λ, the value t (where $t = \alpha(k)$), the value x and a witness $w \in W$ for x's membership in L.

We introduce the following general notion, which we will use to construct an IBE system.

Definition E.9 (Hash Proof System with Trapdoor)**.** **P** is a HPS with trapdoor for subset membership problem **M** if instances of **M** can be generated (with the appropriate distribution) together with a trapdoor MSK that enables an efficient *lifting function* $\sigma_{\mathrm{MSK}} : T \to K$ that is pseudorandom among functions satisfying $\alpha(\sigma_{\mathrm{MSK}}(t)) = t$ for $t \in T$.

We also introduce the following computational assumption, which is defined analogously to the IQR assumption.

Definition E.10 (Interactive Subset Membership (ISM) Assumption)**.** Let $\mathbf{P}$ be a hash proof system for subset membership problem $\mathbf{M}$, as above. Let I be a set. Let $\Gamma : I \rightarrow T$ be a function. Let $\mathcal{O}$ be an oracle chosen uniformly from the set of all functions from $I \rightarrow K$ satisfying $\alpha(O(i)) = \Gamma(i)$ for all $i \in I$. We say the interactive subset membership assumption holds for $(\mathbf{M}, \mathbf{P}, I, \Gamma)$ if for all PPT algorithms $\mathcal{A}$, the function

$$\mathrm{ISMAdv}_{\mathcal{A}}(\lambda) = \left|\Pr[\Lambda \overset{\mathrm{D}_\lambda}{\leftarrow} \mathrm{Gen}(\lambda); x \overset{\mathrm{R}}{\leftarrow} L \ : \ \mathcal{A}^{\mathcal{O}}(\Lambda, x) = 1] - \Pr[\Lambda \overset{\mathrm{D}_\lambda}{\leftarrow} \mathrm{Gen}(\lambda); x \overset{\mathrm{R}}{\leftarrow} X \setminus L \ : \ \mathcal{A}^{\mathcal{O}}(\Lambda, x) = 1]\right|$$

is a negligible function. If $\mathbf{P}$ and $\hat{\mathbf{P}}$ are two HPS's for $\mathbf{M}$, with associated sets I and $\hat{I}$, functions Γ and $\hat{\Gamma}$, and oracles $\mathcal{O}$ and $\hat{\mathcal{O}}$, we say the ISM assumption holds for $(\mathbf{M}, \mathbf{P}, \hat{\mathbf{P}}, I, \hat{I}, \Gamma, \hat{\Gamma})$ when $\mathcal{A}$'s advantage is negligible when given access to both $\mathcal{O}$ and $\hat{\mathcal{O}}$.

Note that the ISM problem may not actually be hard unless Γ has at least minimal security properties, such as collision resistance.

Also, note that when Γ is a full-domain hash function onto T that is modeled as a random oracle and $|\alpha^{-1}(t)|$ is constant for all $t \in T$, the subset membership assumption for $\mathbf{M}$ implies the ISM assumption for $(\mathbf{M}, \mathbf{P}, I, \Gamma)$. This is basically because the simulator can use its control over the random oracle to implement the oracle $\mathcal{O}$; specifically, for $i \in I$, it chooses random $k_i \in K$ and sets $\mathcal{O}(i) \leftarrow k_i$ and $\Gamma(i) \leftarrow \alpha(k_i)$.

E.2 ANON-IND-ID-CCA Identity Based Encryption from Hash Proof Systems

Here, we describe a simple generic construction of ANON-IND-ID-CCA identity based encryption using hash proof systems. The construction is analogous to Cramer and Shoup's generic construction of IND-CCA PKE from hash proof systems [19]. In Section E.3, we specify how to construct the necessary hash proof systems for quadratic residuosity.

Let $\mathbf{P}$ be a hash proof system with trapdoor for subset membership problem $\mathbf{M}$, which for instance $\Lambda[X, L, W, \mathcal{R}]$ of $\mathbf{M}$, has an associated PHF $\mathbf{H} = (H, K, X, L, \Pi, T, \alpha)$ for (X, L) and pseudorandom lifting function $\sigma_{\mathrm{MSK}} : T \rightarrow K$. Let $\hat{\mathbf{P}}$ be an extended hash proof system with trapdoor for $\mathbf{M}$, which for instance $\hat{\Lambda}[\hat{X}, \hat{L}, \hat{W}, \hat{\mathcal{R}}]$ of $\mathbf{M}$, has an associated PHF $\hat{\mathbf{H}} = (\hat{H}, \hat{K}, \hat{X}, \hat{L}, \hat{\Pi}, \hat{T}, \hat{\alpha})$ for $(\hat{X}, \hat{L})$ and pseudorandom lifting function $\hat{\sigma}_{\mathrm{MSK}} : \hat{T} \rightarrow \hat{K}$. (As in [19], "extended" simply means that $\hat{X} = X \times Y$ and $\hat{L} = L \times Y$ for some set Y, with straightforward modifications to the associated algorithms.)

Our generic construction, FullIBE, is as follows.

Setup(1^λ): Generate an instance $\Lambda[X, L, W, \mathcal{R}]$ of $\mathbf{M}$, together with trapdoor MSK. Let $Y = \Pi$. It is required that Π be an abelian group (with operations + and −). Let $\mathcal{ID}$ be the set of possible identities, and let $\Gamma : \mathcal{ID} \rightarrow T$ and $\hat{\Gamma} : \mathcal{ID} \rightarrow \hat{T}$ be functions. Output the public parameters PP, which include Λ, $\mathbf{P}$, $\hat{\mathbf{P}}$, Γ and $\hat{\Gamma}$. The master secret is MSK (with the pseudorandom lifting functions σ_{MSK} and $\hat{\sigma}_{\mathrm{MSK}}$).

KeyGen(PP,MSK,ID): Takes as input PP, MSK, and ID $\in \mathcal{ID}$. It sets $k \leftarrow \sigma_{\mathrm{MSK}}(\Gamma(\mathrm{ID}))$ and $\hat{k} \leftarrow \hat{\sigma}_{\mathrm{MSK}}(\hat{\Gamma}(\mathrm{ID}))$. It outputs the decryption key $d_{\mathrm{ID}} = (\mathrm{PP}, k, \hat{k})$.

Encrypt(PP,ID,m): Takes as input PP, an identity ID, and a message $m \in \Pi$. It generates a random $x \in L$ together with a witness w for x, computes $t \leftarrow \Gamma(\mathrm{ID})$ and $\hat{t} \leftarrow \hat{\Gamma}(\mathrm{ID})$, computes

$\pi = H_k(x)$ using the public evaluation algorithm for $\mathbf{P}$, sets $e = m + \pi$, computes $\hat{\pi} = \hat{H}_{\hat{k}}(x, e)$ using the public evaluation algorithm for $\hat{\mathbf{P}}$, and outputs the ciphertext $C = (x, e, \hat{\pi})$.

Decrypt(C,d_{ID}): Takes as input ciphertext C and decryption key d_{ID}. It parses C and outputs $\perp$ if C does not encode some $(x, e, \hat{\pi}) \in X \times \Pi \times \hat{\Pi}$. Otherwise, it computes $\hat{\pi}' = \hat{H}_{\hat{k}}(x, e)$ using the private evaluation algorithm for $\hat{\mathbf{P}}$. If $\hat{\pi}' \neq \hat{\pi}$, it outputs $\perp$. Otherwise, it computes $\pi = H_k(x)$ using the private evaluation algorithm for $\mathbf{P}$, and outputs $m = e - \pi$.

The following theorem is analogous to Theorem 1 in [19].

Theorem E.11. *Let* $\mathbf{P}$ *be a strongly smooth HPS with trapdoor and* $\hat{\mathbf{P}}$ *be a strongly universal*$_2$ *extended HPS with trapdoor for the subset membership problem* $\mathbf{M}$. *Let* $\Gamma : \mathcal{ID} \to T$ *and* $\hat{\Gamma} : \mathcal{ID} \to \hat{T}$ *be injective. Then,* FullIBE *is ANON-IND-ID-CCA secure (in the standard model) under the interactive subset membership assumption for* $(\mathbf{M}, \mathbf{P}, \hat{\mathbf{P}}, \mathcal{ID}, \mathcal{ID}, \Gamma, \hat{\Gamma})$.

As usual in IBE systems, we can set $\mathcal{ID} = \{0,1\}^*$ and remove the assumption that Γ and $\hat{\Gamma}$ are injective by assuming that Γ and $\hat{\Gamma}$ are collision-resistant and making the appropriate changes to the proof.

Note that it follows that FullIBE is also ANON-IND-ID-CCA secure in the random oracle model under the (non-interactive) subset membership problem for $\mathbf{M}$ when Γ and $\hat{\Gamma}$ are full-domain hash functions.

We prove Theorem E.11 in Section E.4.

Following Kurosawa and Desmedt [30], we can modify the above IBE system into a more efficient IBE KEM-DEM. The generic IBE-KEM-DEM, called FullIBE-KD, is as follows.

Setup(1^λ): Generate an instance $\Lambda[X, L, W, \mathcal{R}]$ of $\mathbf{M}$, together with trapdoor MSK. Let $Y = \emptyset$. Define $\mathcal{ID}$ and $\hat{\Gamma}$ as in FullIBE. Output the public parameters PP, which include Λ, $\hat{\mathbf{P}}$, $\hat{\Gamma}$, a function $F : \hat{\Pi} \to \{0,1\}^d$ and a one-time symmetric encryption system SKE with algorithms SKE.Encrypt and SKE.Decrypt. SKE.Encrypt takes input 1^λ, a d-bit key and a message. (We discuss the requirements on F and SKE later.) The master secret is MSK (with the pseudorandom lifting function $\hat{\sigma}_{\mathrm{MSK}}$).

KeyGen(PP,MSK,ID): Takes as input PP, MSK, and ID $\in \mathcal{ID}$. It sets $\hat{k} \leftarrow \hat{\sigma}_{\mathrm{MSK}}(\hat{\Gamma}(\mathrm{ID}))$. It outputs the decryption key $d_{\mathrm{ID}} = (\mathrm{PP}, \hat{k})$.

Encrypt(PP,ID,m): Takes as input PP, an identity ID, and a message $m \in \Pi$. It generates a random $x \in L$ together with a witness w for x, computes $\hat{t} \leftarrow \hat{\Gamma}(\mathrm{ID})$, computes $\hat{\pi} = \hat{H}_{\hat{k}}(x)$ using the public evaluation algorithm for $\hat{\mathbf{P}}$, and computes $\kappa = F(\hat{\pi})$ and $\chi = \mathsf{SKE.Encrypt}(1^\lambda, \kappa, m)$. The ciphertext is $C = (x, \chi)$.

Decrypt(C,d_{ID}): Takes as input ciphertext C and decryption key d_{ID}. It parses C and outputs $\perp$ if C does not encode some (x, χ) with $x \in X$. Otherwise, it computes $\hat{\pi}' = \hat{H}_{\hat{k}}(x)$ using the private evaluation algorithm for $\hat{\mathbf{P}}$, $\kappa' = F(\hat{\pi}')$, and $m' = \mathsf{SKE.Decrypt}(1^\lambda, \kappa', \chi)$. It outputs m' if SKE.Decrypt does not register an error.

The following theorem is analogous to Theorem 2 in [30].

Theorem E.12. *Let* $\hat{\mathbf{P}}$ *be a strongly universal*$_2$ *extended HPS with trapdoor for the subset membership problem* $\mathbf{M}$, $\hat{\Gamma} : \mathcal{ID} \to \hat{T}$ *be injective,* $F(\hat{\pi})$ *be uniformly distributed over* $\{0,1\}^d$ *if* $\hat{\pi}$ *is uniformly distributed over* $\hat{\Pi}$, *and* SKE *be secure in the sense of IND-CCA and* ϵ*-rejection secure for negligible* ϵ. *Then,* FullIBE-KD *is ANON-IND-ID-CCA secure (in the standard model) under the interactive subset membership assumption for* $(\mathbf{M}, \hat{\mathbf{P}}, \mathcal{ID}, \hat{\Gamma})$.

We say that SKE is ϵ-rejection secure if $\Pr[\mathsf{SKE.Decrypt}(1^\lambda, \kappa, \chi) = \mathit{reject}] \geq 1 - \epsilon$ for any bit-string χ, where the probability is taken over the choice of keys κ. We omit the proof of Theorem E.12, since it adapts the proof of Theorem E.11 in the same way that the proof of Kurosawa and Desmedt's Theorem 2 adapts the proof of Cramer and Shoup's Theorem 1.

To instantiate FullIBE and FullIBE-KD, we use the HPS's defined in Section E.3 for quadratic residuosity. We claim the following lemmas are true.

Lemma E.13. *The HPS* $\mathbf{P}$*, as defined in Section E.3, is a strongly smooth HPS with trapdoor.*

Lemma E.14. *The HPS* $\hat{\mathbf{P}}$*, as defined in Section E.3, is a strongly univeral*$_2$ *extended HPS with trapdoor.*

We prove Lemmas E.13 and E.14 in Sections E.5 and E.6, respectively. In our proof of Lemma E.14, we assume (as in [19]) that a certain function Γ' is injective. As mentioned in [19], it is straightforward to adapt the model account for the case that Γ' is merely collision-resistant.

As an aside, we note that Gentry's pairing-based anonymous IBE system [25] falls within the above framework for ANON-IND-ID-CCA IBE from hash proof systems (though his scheme has minor differences from our generic scheme). That scheme is secure in the standard model without interactive assumptions basically because the simulator can actually implement the oracles O and $\hat{O}$ itself by using an instance to a different subset membership problem $\mathbf{M}_0$ to build an instance of the problem $\mathbf{M}$. However, we note that the size of the $\mathbf{M}_0$-instance in [25] must be proportional to the total number of private key queries permitted to the adversary; otherwise, the simulator's lifting function σ will not appear pseudorandom.

E.3 Hash Proof Systems for Quadratic Residuosity

Now, we define two HPS's $\mathbf{P}$ and $\hat{\mathbf{P}}$ for the quadratic residuosity problem. When implemented with these HPS's, and when $\hat{\mathbf{P}}$ implements Γ' as a collision-resistant hash with $\hat{\ell}$-bit output, ciphertexts in FullIBE are $1 + \ell + \hat{\ell} + \log_2 N$ bits, where ℓ is the message length parameter.

E.3.1 A Strongly Smooth Universal Hash Proof System

In our strongly smooth HPS $\mathbf{P}$ for quadratic residuosity, we generate an instance of the quadratic residuosity subset membership problem for security parameter λ by setting $(p, q) \xleftarrow{\text{R}} \text{RSAgen}(\lambda)$, $N = pq$, $X = J(N) \times \{-1, 1\}$, $L = QR(N) \times \{-1, 1\}$; we say $((S, b), s)$ is a relation in $\mathcal{R} = X \times W$ when $S, s \in \mathbb{Z}/N\mathbb{Z}$, $S = s^2$, and $(\tau(s)/N) = b$, where the polynomial τ is generated using $\mathcal{Q}'$ as in Section 6. $\mathbf{P}$ uses the projective hash function $\mathbf{H} = (H, K, X, L, \Pi, T, \alpha)$. We set $K = \{(r_1, \ldots, r_\ell) \in (\mathbb{Z}/N\mathbb{Z}^*)^\ell\}$. T is a set of equivalence classes in $(J(N))^\ell$, where $(R_1, \ldots, R_\ell) \cong (R'_1, \ldots, R'_\ell)$ if there exists $(r_1, \ldots, r_\ell)$ such that $R_i = u^{-a_i} r_i^2$ and $R'_i = u^{-a'_i} r_i^2$ for $a_i, a'_i \in \{0, 1\}$ for all $i \in [1, \ell]$, where u is a fixed element of $J(N) \setminus QR(N)$. The projection $\alpha : K \to T$ is component-wise squaring – i.e., $\alpha(r_1, \ldots, r_\ell) = (R_1, \ldots, R_\ell)$ where $R_i = r_i^2$, and where the rhs is understood to represent an equivalence class. Finally, $\Pi = \{-1, 1\}^\ell$, where ℓ is the message length parameter. Note Π is an abelian group with operation '+' given by $\pi^{(1)} + \pi^{(2)} = (\pi_1^{(1)} \cdot \pi_1^{(2)}, \ldots, \pi_\ell^{(1)} \cdot \pi_\ell^{(2)})$, where $\pi^{(1)} = (\pi_1^{(1)}, \ldots, \pi_\ell^{(1)})$ and $\pi^{(2)} = (\pi_1^{(2)}, \ldots, \pi_\ell^{(2)})$ are elements of Π.

Now, we define the private and public evaluation algorithms for $\mathbf{P}$. Let $x = (S, b) \in X$, let $k = (r_1, \ldots, r_\ell) \in K$, and let $t = \alpha(k) = (u^{-a_1} r_1^2, \ldots, u^{-a_\ell} r_\ell^2)$ for $a_j \in \{0, 1\}$ be the class representative of $\alpha(k)$ (with some abuse of notation). To compute $H_k(x)$ using the private evaluation algorithm, one does the following:

- For each $j = 1, \dots, \ell$, use $\mathcal{Q}'$ to generate f_j (resp. $\bar{f}_j$) for S and r_j^2 when $a_j = 0$ (resp. when $a_j = 1$), as described in Section 6.
- Output $H_k(x) \in \{-1, 1\}^\ell$ as the sequence $B_1, \dots, B_\ell$, where $B_j = \left(\frac{f_j(r_j)}{N}\right)$ or $B_j = b\left(\frac{\bar{f}_j(r_j)}{N}\right)$, depending on whether a_j equals 0 or 1.

To compute $H_k(x)$ using the public evaluation algorithm on input t, $(S, b) \in L$, and a witness s for (S, b) satisfying $S = s^2$ and $(\tau(s)/N) = b$, one does the following:

- For $j = 1, \dots, \ell$, use $\mathcal{Q}'$ to generate g_j as described in Section 6.
- Output $H_k(x) \in \{-1, 1\}^\ell$ as the sequence $B_1, \dots, B_\ell$, where $B_j = \left(\frac{g_j(s)}{N}\right)$.

E.3.2 A Strongly Universal$_2$ Hash Proof System

In our strongly universal$_2$ HPS $\hat{\mathbf{P}}$ for quadratic residuosity, N and u are generated as in $\mathbf{P}$, $\hat{X} = X \times Y$ and $\hat{L} = L \times Y$ for some set Y, and $((S, b, y), s)$ is a relation in $\hat{X} \times \hat{W}$ when $y \in Y$, $S, s \in \mathbb{Z}/N\mathbb{Z}$, $S = s^2$, and $(\tau(s)/N) = b$, where the polynomial τ is generated using $\mathcal{Q}'$ as in Section 6. $\hat{K} = \{(r_1, \dots, r_{\hat{\ell}}, r_1^+, \dots, r_{\hat{\ell}}^+) \in (\mathbb{Z}/N\mathbb{Z}^*)^{2\hat{\ell}}\}$, where $\hat{\ell}$ is parameter depending on λ; $\hat{T}$ and $\hat{\alpha}$ are defined in the obvious way. We set $\hat{\Pi} = \mathbb{F} = \mathbb{F}_2[z]/h(z)$, where $h(z)$ is a monic polynomial of degree $\hat{\ell}$ that is irreducible in $\mathbb{F}_2[z]$. Finally, $\Gamma' : \hat{X} \to \hat{\Pi}$ is a hash function. In the proof of Lemma E.14, Γ' is assumed to be injective, but, as mentioned in [19], the hash proof system framework can be modified to handle the case that Γ' is merely collision resistant.

To compute $\hat{H}_{\hat{k}}(\hat{x})$ using the private evaluation algorithm on input $\hat{k} = (r_1, \dots, r_{\hat{\ell}}, r_1^+, \dots, r_{\hat{\ell}}^+) \in \hat{K}$ and $\hat{x} = (S, b, y) \in \hat{X}$, one does the following:

- For each $j = 1, \dots, \hat{\ell}$, use $\mathcal{Q}'$ to generate f_j (resp. $\bar{f}_j$) for S and r_j^2 when $a_j = 0$ (resp. when $a_j = 1$), as described in Section 6. Generate f_j^+ or $\bar{f}_j^+$ analogously.
- For each $j = 1, \dots, \hat{\ell}$, set $B_j = (1 + \left(\frac{f_j(r_j)}{N}\right))/2$ or $B_j = (1 + b\left(\frac{\bar{f}_j(r_j)}{N}\right))/2$, depending on whether a_j equals 0 or 1. Set B_j^+ similarly.
- Set $\mathfrak{e} = \sum_{j=1}^{\hat{\ell}} B_j z^{j-1} \in \mathbb{F}$. Set $\mathfrak{e}^+$ similarly.
- Set $\mathfrak{a} \leftarrow \Gamma'(S, b, y)$.
- Output $\hat{\pi} \leftarrow \mathfrak{e} + \mathfrak{a} \cdot \mathfrak{e}^+ \in \mathbb{F} = \hat{\Pi}$.

To compute $\hat{H}_{\hat{k}}(\hat{x})$ using the public evaluation algorithm on input $\hat{t}$, $\hat{x} = (S, b, y) \in \hat{X}$, and witness s for (S, b, y) satisfying $S = s^2$ and $(\tau(s)/N) = b$, one does the following:

- For each $j = 1, \dots, \hat{\ell}$, use $\mathcal{Q}'$ to generate g_j and g_j^+ as described in Section 6.
- For each $j = 1, \dots, \hat{\ell}$, set $B_j = (1 + \left(\frac{g_j(s)}{N}\right))/2$. Set B_j^+ similarly.
- Set $\mathfrak{e} = \sum_{j=1}^{\hat{\ell}} B_j z^{j-1} \in \mathbb{F}$. Set $\mathfrak{e}^+$ similarly.
- Set $\mathfrak{a} \leftarrow \Gamma'(S, b, y)$.
- Output $\hat{\pi} \leftarrow \mathfrak{e} + \mathfrak{a} \cdot \mathfrak{e}^+ \in \mathbb{F} = \hat{\Pi}$.

E.4 Proof of Theorem E.11

We prove that the system is ANON-IND-ID-CCA under the ISM assumption for $(\mathbf{M}, \mathbf{P}, \hat{\mathbf{P}}, \mathcal{ID}, \mathcal{ID}, \Gamma, \hat{\Gamma})$ when $\mathbf{P}$ is a strongly smooth HPS with trapdoor, $\hat{\mathbf{P}}$ is a strongly universal$_2$ extended HPS with trapdoor for the subset membership problem $\mathbf{M}$, and Γ and $\hat{\Gamma}$ are injective.

Let $\mathcal{A}$ be an ANON-IND-ID-CCA adversary attacking the IBE system FullIBE. We wish to bound its advantage $\mathrm{IBEAdv}_{\mathcal{A},\mathsf{FullIBE}}(\lambda)$. We present the proof as a series of games. We let W_i denote the event that the adversary wins Game i.

Game 0. The first game is identical to the ANON-IND-ID-CCA game defined in Section 2. Hence, we know that

$$\left|\Pr[W_0] - \frac{1}{2}\right| = \mathrm{IBEAdv}_{\mathcal{A},\mathsf{FullIBE}}(\lambda) \tag{27}$$

Game 1. In Game 1 we slightly modify Game 0. Instead of using the pseudorandom lifting functions σ_{MSK} and $\hat{\sigma}_{\mathrm{MSK}}$ to respond to $\mathcal{A}$'s private key queries, the challenger uses truly random functions $F : T \to K$ and $\hat{F} : \hat{T} \to \hat{K}$ satisfying $\alpha(F(t)) = t$ and $\hat{\alpha}(\hat{F}(\hat{t})) = \hat{t}$ for all $t \in T$ and $\hat{t} \in \hat{T}$. To generate a private key $(k, \hat{k})$ for ID, the challenger sets $k = F(\Gamma(\mathrm{ID}))$ and $\hat{k} = \hat{F}(\hat{\Gamma}(\mathrm{ID}))$. Clearly, if σ and $\hat{\sigma}$ are secure PRFs, $\mathcal{A}$ will not notice the difference between Game 0 and Game 1. In particular, there exists an algorithm $\mathcal{B}_1$ (whose running time is about the same as that of $\mathcal{A}$) such that

$$|\Pr[W_1] - \Pr[W_0]| = \mathrm{PRFAdv}_{\mathcal{B}_1,(\sigma,\hat{\sigma})}(\lambda) \tag{28}$$

Game 2. In Game 2 we remove knowledge of the master private key from the challenger. The challenger is given an instance Λ of M, the HPS's $\mathbf{P}$ and $\hat{\mathbf{P}}$, hash functions Γ and $\hat{\Gamma}$, and access to randomly chosen oracles $\mathcal{O}$ and $\hat{\mathcal{O}}$ as in Definition E.10.

This changes how the challenger responds to private key queries, but Game 2 is otherwise identical to Game 1. In particular, to respond to a private key query on ID, the challenger feeds ID to $\mathcal{O}$ and $\hat{\mathcal{O}}$ and forwards the oracles' responses k and $\hat{k}$ to $\mathcal{A}$. However, the distribution of the challenger's responses to private key queries (and decryption queries) is unchanged, since $\mathcal{O}$ and $\hat{\mathcal{O}}$ have the same distribution as $F \circ \Gamma$ and $\hat{F} \circ \hat{\Gamma}$ on domain $\mathcal{ID}$ (and the challenger uses *Decrypt* with its private key to respond to decryption queries).

Therefore,

$$\Pr[W_2] = \Pr[W_1] \tag{29}$$

Game 3. Game 3 proceeds exactly as Game 2 except that the encryption query is handled differently. To encrypt a message m_b to a particular ID_b, the challenger finds the private key $d_{\mathrm{ID}_b} = (k, \hat{k})$. It chooses $x^* \overset{\mathrm{R}}{\leftarrow} L$, computes $\pi \leftarrow H_k(x^*)$ using the private evaluation algorithm for $\mathbf{P}$, sets $e^* \leftarrow m + \pi$, computes $\hat{\pi}^* \leftarrow \hat{H}_{\hat{k}}(x, e)$ using the private evaluation algorithm for $\hat{\mathbf{P}}$, and returns $(x^*, e^*, \hat{\pi}^*)$ to the adversary.

Since $x^* \in L$, the private and public evaluation algorithms compute the same result for $H_k(x^*)$ and $\hat{H}_{\hat{k}}(x^*, e^*)$. Since x^* is a uniform L-element in Games 2 and 3, the distribution of the challenger's response to encryption query is identical. Since no other aspect of Games 2 and 3 differs,

$$\Pr[W_3] = \Pr[W_2] \tag{30}$$

Game 4. Game 4 proceeds exactly as Game 3 except that when responding to the encryption query, instead of choosing $x^* \overset{\text{R}}{\leftarrow} L$, the challenger sets $x^* \overset{\text{R}}{\leftarrow} X \setminus L$. The adversary should not notice the difference unless it can distinguish subset membership (in the interactive setting). In particular, there exists an algorithm $\mathcal{B}_2$ (whose running time is about the same as that of $\mathcal{A}$) such that

$$|\Pr[W_4] - \Pr[W_3]| = \text{ISMAdv}_{\mathcal{B}_2,\text{Gen}}(\lambda) \tag{31}$$

Game 5. In Game 5 we modify how the challenger handles decryption queries. In particular, the challenger is given the power to determine subset membership (in L), and the challenger immediately outputs $\perp$ in response to a decryption query $(x, e, \hat{\pi}, \text{ID})$ if $x \notin L$. Aside from this additional ciphertext check, the challenger handles decryption queries as in Game 4.

Let E be the event the challenger rejects some decryption query C in Game 5 that would have been accepted in Game 4. (Note that the opposite is impossible; if C is rejected in Game 4, it is certainly rejected in Game 5.) Since Games 4 and 5 proceed identically until E occurs,

$$|\Pr[W_5] - \Pr[W_4]| \leq \Pr[E] \tag{32}$$

Below, we prove the following:

Lemma E.15. $\Pr[E] \leq Q/|\hat{\Pi}|$, *where Q is the number of decryption queries made by the adversary.*

Game 6. In Game 6, we again modify how the encryption query is handled. In particular, the challenger generates the x^* component of its encryption query response as in Game 5 ($x^* \overset{\text{R}}{\leftarrow} X \setminus L$), but generates the e^* and $\hat{\pi}^*$ components as uniformly random elements of Π and $\hat{\Pi}$. Below, we prove the following:

Lemma E.16.

$$\Pr[W_6] = \Pr[W_5] \tag{33}$$

Since the challenger does not use m_b or ID_b to generate its response, the response is independent of m_b or ID_b, and we have,

$$\Pr[W_6] = \frac{1}{2} \tag{34}$$

Combining equations (27) through (34), and subject to Lemmas E.15 and E.16, we obtain:

$$\text{IBEAdv}_{\mathcal{A},\mathsf{FullIBE}}(\lambda) \leq \text{PRFAdv}_{\mathcal{B}_1,(\sigma,\hat{\sigma})}(\lambda) + \text{ISMAdv}_{\mathcal{B}_2,\text{Gen}}(\lambda) + Q/|\hat{\Pi}|$$

□

Now, we prove Lemmas E.15 and E.16.

Proof of Lemma E.15. Suppose $\mathcal{A}$ makes the decryption query $(x, e, \hat{\pi}, \text{ID})$ with $x \in X \setminus L$. Let $(k, \hat{k})$ be the private key for ID that the challenger obtains from $(\mathcal{O}, \hat{\mathcal{O}})$. We consider three cases:

- Case 1: $(x, e, \text{ID}) = (x^*, e^*, \text{ID}_b)$. Since we must have $(x, e, \hat{\pi}, \text{ID}) \neq (x^*, e^*, \hat{\pi}^*, \text{ID}_b)$, we conclude that $\hat{\pi} \neq \hat{\pi}^*$. However, this implies that $\hat{\pi} \neq \hat{H}_{\hat{k}}(x, e)$, and so the decryption query is certainly rejected in either game.

- Case 2: $\mathrm{ID} = \mathrm{ID}_b$ but $(x, e) \neq (x^*, e^*)$. In $\mathcal{A}$'s view, the only constraint on $\hat{k}$ is that $\hat{\alpha}(\hat{k}) = \hat{\Gamma}(\mathrm{ID})$ and $\hat{\pi}^* = \hat{H}_{\hat{k}}(x^*, e^*)$. Even if $\mathcal{A}$ knows b, the $(1/|\hat{\Pi}|)$-universality$_2$ of $\hat{\mathbf{H}}$ implies that the probability that $\hat{H}_{\hat{k}}(x, e) = \hat{\pi}$ is only $1/|\hat{\Pi}|$.

- Case 3: $\mathrm{ID} \neq \mathrm{ID}_b$. Here, $\mathcal{A}$ has even less information about $\hat{k}$ than in Case 2: just that $\hat{\alpha}(\hat{k}) = \hat{\Gamma}(\mathrm{ID})$. Again, the $(1/|\hat{\Pi}|)$-universality$_2$ of $\hat{\mathbf{H}}$ implies that the probability that $\hat{H}_{\hat{k}}(x, e) = \hat{\pi}$ is only $1/|\hat{\Pi}|$.

Thus, the desired bound follows. □

Proof of Lemma E.16. Let $(k, \hat{k})$ be the private key for ID_b that the challenger obtains from $(\mathcal{O}, \hat{\mathcal{O}})$. Since the challenger rejects all decryption queries $(x, e, \hat{\pi}, \mathrm{ID})$ with $x \notin L$, since *Decrypt* outputs the same result when applied to $(x, e, \hat{\pi}, \mathrm{ID}_b)$ irrespective of which private keys for ID_b is used when $x \in L$, and since $(\mathcal{O}, \hat{\mathcal{O}})$ are truly random functions (i.e., the private keys for distinct ID's are generated independently since Γ and $\hat{\Gamma}$ are injections), $\mathcal{A}$ has no information about $(k, \hat{k})$ other than that $(\alpha(k), \hat{\alpha}(\hat{k})) \in (\Gamma(\mathrm{ID}_b), \hat{\Gamma}(\mathrm{ID}_b))$, $e^* = m_b + H_k(x^*)$, and $\hat{\pi}^* = \hat{H}_{\hat{k}}(x^*, e^*)$. Since both $\mathbf{H}$ and $\hat{\mathbf{H}}$ are strongly smooth, the values of $H_k(x^*)$ and $\hat{H}_{\hat{k}}(x^*, e^*)$ remain uniformly random in Π and $\hat{\Pi}$ given $(\alpha(k), \hat{\alpha}(\hat{k}))$. The result follows. □

E.5 Proof of Lemma E.13

First, we make sure that $\mathbf{P}$ meets the basic requirements of a hash proof system with trapdoor. It is clear that the QR problem is a subset membership problem, complete with efficient algorithms for sampling instances, for checking whether an element S is in $X = J(N) \times \{-1, 1\}$, and for generating a random element $(S, b) \in L = QR(N) \times \{-1, 1\}$ together with a witness s satisfying $S = s^2$ and $(\tau(s)/N) = b$. (An algorithm for the latter is given in Section 6.) For PHF $H = (H, K, X, L, \Pi, T, \alpha)$ for (X, L), it is also clear that there are efficient algorithms for sampling k from K at random, computing $\alpha(k)$, and checking whether a bit string is an element of Π. Given the factorization of N, it is straightforward to construct the pseudorandom lifting function. In the body of the paper, we discussed efficient algorithms for implementing the private and public evaluation algorithms.

It remains to prove that $\mathbf{H}$ is strongly smooth – i.e., that for all $S \in X \setminus L$ and all values of $t \in T$, the distribution of $H_k(S)$ is uniform in Π when $k \in K$ is chosen uniformly at random from the values satisfying $t = \alpha(k)$. Since the j-th bit of $H_k(S)$ for $t = (R_1, \ldots, R_\ell)$ is independent of $(R_1, \ldots, R_{j-1}, R_{j+1}, \ldots, R_\ell)$, it suffices to show that the j-th bit of $H_k(S)$ is uniform in $\{0, 1\}$ when r_j (in the key $k = (r_1, \ldots, r_\ell)$) is chosen uniformly from the square roots of R_j. However, this is immediate consequence of Lemma 3.3. □

E.6 Proof of Lemma E.14

As in the proof of Lemma E.13, it is clear that $\hat{\mathbf{P}}$ meets the basic requirements of being a hash proof system. It remains to prove that $\hat{\mathbf{H}}$ is strongly universal$_2$. We prove this by showing that the cardinality of the set $\hat{K}^\dagger = \{\hat{k} \in \hat{K} : \hat{t} = \hat{\alpha}(\hat{k}) \wedge \hat{\pi} = \hat{H}_{\hat{k}}(\hat{x}) \wedge \hat{\pi}^* = \hat{H}_{\hat{k}}(\hat{x}^*)\}$ is constant over all distinct $\hat{x}, \hat{x}^* \in \hat{X} \setminus \hat{L} = (J(N) \setminus QR(N)) \times \{-1, 1\} \times Y$, all $\hat{\pi}, \hat{\pi}^* \in \hat{\Pi}$, and all $\hat{t} \in \hat{T}$. In particular, $|\hat{K}^\dagger| = 2^{2\hat{\ell}}$.

To prove this, let us consider the structure of $\mathbb{Z}/N\mathbb{Z}^*$ for $N = pq$. Let $\mathcal{W} \subset \mathbb{Z}/N\mathbb{Z}^*$ be such that $|\mathcal{W}|$ is maximal for sets satisfying $w^2 \neq w'^2$ for distinct $w, w' \in \mathcal{W}$. Let $v \in \mathbb{Z}/N\mathbb{Z}$ be such $v = 1 \bmod p$ and $v = -1 \bmod q$. Then, each element in $r \in \mathbb{Z}/N\mathbb{Z}^*$ has a unique representation

$(a, b, w) \in \{0,1\} \times \{0,1\} \times \mathcal{W}$ with $r = (-1)^a \cdot v^b \cdot w$. We can extend this representation componentwise to a representation for $\hat{k} = (r_1, \ldots, r_{\hat{\ell}}, r_1^+, \ldots, r_{\hat{\ell}}^+) \in \hat{K}$ as $(\vec{a}, \vec{b}, \vec{w})$ in the obvious way. Also, we use mappings from the $\vec{b}$ component to $\mathbb{F}$-elements as follows: $\mathfrak{b} \leftarrow \theta(\vec{b}) \leftarrow \sum_{j=1}^{\hat{\ell}} b_j z^{j-1}$ and $\mathfrak{b}^+ \leftarrow \theta^+(\vec{b}) \leftarrow \sum_{j=1}^{\hat{\ell}} b_j^+ z^{j-1}$.

Now, we state some useful facts. Let $\hat{k}$ and $\hat{k}'$ be distinct elements of $\hat{\alpha}^{-1}(\hat{t})$ with representations $(\vec{a}, \vec{b}, \vec{w})$ and $(\vec{a}', \vec{b}', \vec{w}')$, respectively, with $\mathfrak{b} \leftarrow \theta(\vec{b})$, $\mathfrak{b}^+ \leftarrow \theta^+(\vec{b})$, $\mathfrak{b}' \leftarrow \theta(\vec{b}')$, and $\mathfrak{b}^{+\prime} \leftarrow \theta^+(\vec{b}')$. Obviously, $\vec{w} = \vec{w}'$, since $\hat{t}$ determines $\vec{w}$. Let $\mathfrak{e}$ and $\mathfrak{e}^+$ (resp. $\mathfrak{e}'$ and $\mathfrak{e}^{+\prime}$) be the $\mathbb{F}$-elements that arise in Step E.3.2 of the private evaluation algorithm while computing $\hat{H}_{\hat{k}}(\hat{x})$ (resp. $\hat{H}_{\hat{k}'}(\hat{x})$). Then, it turns out that $\mathfrak{b} - \mathfrak{e} = \mathfrak{b}' - \mathfrak{e}'$ and $\mathfrak{b}^+ - \mathfrak{e}^+ = \mathfrak{b}^{+\prime} - \mathfrak{e}^{+\prime}$. In other words, $\Delta_{\hat{t},\hat{x}} = \mathfrak{b} - \mathfrak{e}$ and $\Delta^+_{\hat{t},\hat{x}} = \mathfrak{b}^+ - \mathfrak{e}^+$ depend only on $\hat{t}$ and $\hat{x}$ (and $\mathcal{W}$), and are invariant under keys in $\hat{\alpha}(\hat{t})$. To see this, note that the coefficients of z^{j-1} in $\mathfrak{e}$ and $\mathfrak{e}'$ differ precisely when $\left(\frac{f_j(r_j)}{N}\right) \neq \left(\frac{f_j(r_j')}{N}\right)$ (or $\left(\frac{\bar{f}_j(r_j)}{N}\right) \neq \left(\frac{\bar{f}_j(r_j')}{N}\right)$), which occurs iff $1 < \gcd(N, r_j - r_j') < N$, which occurs iff $b_j \neq b_j'$ (in the representations of r_j and r_j'), which occurs precisely when the coefficients of z^{j-1} in $\mathfrak{b}$ and $\mathfrak{b}'$ differ.

So, for any fixed distinct $\hat{x}, \hat{x}^* \in \hat{X} \setminus \hat{L}$, fixed $\hat{\pi}, \hat{\pi}^* \in \hat{\Pi}$, and fixed $\hat{t} \in \hat{T}$, we obtain:

$$
\begin{aligned}
|\hat{K}^\dagger| = \quad & |\{(\vec{a}, \mathfrak{b}, \mathfrak{b}^+, \vec{w}) \in \{0,1\}^{2\hat{\ell}} \times \mathbb{F} \times \mathbb{F} \times \mathcal{W}^{2\hat{\ell}} : \hat{t} = \hat{\alpha}(\vec{w}) \wedge \hat{\pi} = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}}) + \Gamma(\hat{x})(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}}) \\
& \quad \wedge \hat{\pi}^* = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}^*}) + \Gamma(\hat{x}^*)(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}^*})\}| \\
= \quad & |\{(\vec{a}, \mathfrak{b}, \mathfrak{b}^+) \in \{0,1\}^{2\hat{\ell}} \times \mathbb{F} \times \mathbb{F} : \hat{\pi} = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}}) + \Gamma(\hat{x})(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}}) \\
& \quad \wedge \hat{\pi}^* = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}^*}) + \Gamma(\hat{x}^*)(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}^*})\}| \\
= \quad & 2^{2\hat{\ell}} |\{(\bar{\mathfrak{b}}, \hat{\mathfrak{b}}) \in \mathbb{F} \times \mathbb{F} : \hat{\pi} = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}}) + \Gamma(\hat{x})(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}}) \\
& \quad \wedge \hat{\pi}^* = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}^*}) + \Gamma(\hat{x}^*)(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}^*})\}|
\end{aligned}
$$

From the equations $\hat{\pi} = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}}) + \Gamma(\hat{x})(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}})$ and $\hat{\pi}^* = (\mathfrak{b} - \Delta_{\hat{t},\hat{x}^*}) + \Gamma(\hat{x}^*)(\mathfrak{b}^+ - \Delta^+_{\hat{t},\hat{x}^*})$, we conclude that

$$
(\Gamma(\hat{x}) - \Gamma(\hat{x}^*))\mathfrak{b}^+ = \hat{\pi} - \hat{\pi}^* + \Delta_{\hat{t},\hat{x}} - \Delta_{\hat{t},\hat{x}^*} + \Gamma(\hat{x})\Delta^+_{\hat{t},\hat{x}} - \Gamma(\hat{x}^*)\Delta^+_{\hat{t},\hat{x}^*} \ ,
$$

where the right side of the equation is a constant. Since Γ is injective, $\Gamma(\hat{x}) - \Gamma(\hat{x}^*) \neq 0$, and $\mathfrak{b}^+$ is uniquely determined. Similarly, $\mathfrak{b}$ is uniquely determined. Thus, $|\hat{K}^\dagger| = 2^{2\hat{\ell}}$. □

E.7 IND-CCA PKE from Hash Proof Systems

We note that our hash proof systems allow us to construct an IND-CCA PKE system in the standard model under the (*non-interactive*) quadratic residuosity assumption. This result may be of independent interest.

Cramer and Shoup describe the following PKE system, for which we can use our instantiations of $\mathbf{P}$ and $\hat{\mathbf{P}}$.

The PKE system FullPKE.

Setup(1^λ): Generate an instance $\Lambda[X, L, W, R]$ of $\mathbf{M}$. Let $Y = \Pi$. It is required that Π be an abelian group (with operations + and −). Publish the parameters (common reference string) PP, which include Λ, $\mathbf{P}$ and $\hat{\mathbf{P}}$.

KeyGen(PP): Generate random $k \in K$ and $\hat{k} \in \hat{K}$. The private key is $k, \hat{k}$. The public key is $(t, \hat{t})$ with $t = \alpha(k)$ and $\hat{t} = \hat{\alpha}(\hat{k})$.

Encrypt(PP,t,$\hat{t}$,m): Takes as input PP, a user's public key $(t, \hat{t})$, and a message $m \in \Pi$. It generates a random $x \in L$ together with a witness w for x, computes $\pi = H_k(x)$ using the public evaluation algorithm for $\mathbf{P}$, sets $e = m + \pi$, computes $\hat{\pi} = \hat{H}_{\hat{k}}(x, e)$ using the public evaluation algorithm for $\hat{\mathbf{P}}$, and outputs the ciphertext $C = (x, e, \hat{\pi})$.

Decrypt(PP,C,k,$\hat{k}$): Takes as input ciphertext C and decryption key $(k, \hat{k})$. It parses C and outputs $\perp$ if C does not encode some $(x, e, \hat{\pi}) \in X \times \Pi \times \hat{\Pi}$. Otherwise, it computes $\hat{\pi}' = \hat{H}_{\hat{k}}(x, e)$ using the private evaluation algorithm for $\hat{\mathbf{P}}$. If $\hat{\pi}' \neq \hat{\pi}$, it outputs $\perp$. Otherwise, it computes $\pi = H_k(x)$ using the private evaluation algorithm for $\mathbf{P}$, and outputs $m = e - \pi$.

Cramer and Shoup [19] proved the following theorem.

Theorem E.17 (Cramer-Shoup)**.** *Suppose* $\mathbf{P}$ *is a strongly smooth HPS and* $\hat{\mathbf{P}}$ *is a strongly universal*$_2$ *extended HPS for the subset membership problem* $\mathbf{M}$*. Then, the system above is secure against chosen ciphertext attack (in the standard model) assuming* $\mathbf{M}$ *is a hard problem.*

Our hash proof systems also allow us to construct an efficient PKE KEM-DEM in the standard model under the quadratic residuosity assumption, à la Kurosawa and Desmedt [30]. Both PKE systems are recipient-anonymous, or "key-private" in the terminology of [3].

We note that, since the above system is not identity-based, we could use a slightly simpler pair of HPS's for quadratic residuosity. Basically, the reason is that the user's public key can be a set of elements from $QR(N)$, rather than $J(N)$, making the parameter $u \in J(N) \setminus QR(N)$ (and the algorithmic steps involving u) unnecessary. (See Appendix B for the IND-CPA version of this system.)

Cramer and Shoup [19] also describe an IND-CCA PKE system that uses HPS's for quadratic residuosity. In their scheme, the lengths of the ciphertext and public key are basically the same as in our scheme. The decryption times are also comparable, assuming that in our scheme the ciphertext value S and the public key values $\{R_j\}$ are presented as primes already satisfying the conditions mentioned in Section 5.2. Encryption in our system may take longer, only because the sender must generate $(s, \tilde{S})$ such that s is uniformly random in $\mathbb{Z}/N\mathbb{Z}$, and $\tilde{S}$ is a prime satisfying $\tilde{S} = s^2 \bmod N$ and $\tilde{S} = 1 \bmod 4$. However, we note that $(s, \tilde{S})$ could be computed offline, since it is independent of the plaintext and the recipient's public key.