Efficient Identity-Based Encryption and Public-Key Signature from Trapdoor Subgroups*

Jong Hwan Park† Kwangsu Lee‡ Dong Hoon Lee§

Abstract

We present a new Identity-Based Encryption (IBE) scheme from a trapdoor subgroup of $\mathbb{Z}_n^*$ for an RSA modulus n. In a trapdoor subgroup of $\mathbb{Z}_n^*$, a subgroup order is hidden and can be used as a trapdoor. Our IBE scheme is efficient in both performance and space. Compared to practical pairing-based IBE schemes, ours is more efficient particularly in terms of computational performance. Following Naor's observation, we also suggest a new Public-Key Signature (PKS) scheme from a trapdoor subgroup of $\mathbb{Z}_n^*$. A favorable feature of our PKS scheme is that signing algorithm is exponentiation-free and requires only one modular inversion. This enables our PKS scheme to provide the fastest signing, compared to practical signature schemes such as RSA and ECDSA. We prove the security of our schemes in the random oracle model under new computational hardness problems that arguably hold in the trapdoor subgroup of $\mathbb{Z}_n^*$.

Keywords: identity-based encryption, trapdoor subgroup, RSA modulus, public-key signature.

1 Introduction

Identity-Based Encryption (IBE) is a special type of public key encryption where a public key can be any string such as an e-mail address or a device identifier. A private key that corresponds to such a string (i.e., identity) is generated by a trusted third party called a key generation center (KGC). A message is encrypted under a receiver's identity (as a public key) and public parameters, without any interaction with the receiver to obtain the public key. A ciphertext is then decrypted with a private key that corresponds to the receiver's identity.

Since Shamir [36] posed the initial question about the existence of such an IBE system, there have been three approaches for constructing IBE schemes. The first one is based on the bilinear map (i.e., pairing) that gives [9, 34, 5, 38, 39, 14, 21], and the second one is based on the quadratic residuosity (QR) problem that gives [15, 10], and the third one is based the lattice that gives [18, 13, 1]. Among the previous IBE schemes, three pairing-based schemes [9, 34, 5] have been perceived as being practical constructions in terms of both performance and space, and are now in the process of standardization in the IEEE P1363.3 and ISO/IEC SC27 18033-5, respectively

*Since our manuscript has been revealed at ePrint Archive, we have received several analysis from Marc Joye, Jung Yeon Hwang, and Olivier Sanders. All of them show that our schemes are all broken and only the inverse structure in a trapdoor subgroup of $\mathbb{Z}_n^*$ is not enough. We put the analysis by Marc Joye in the last part of this manuscript. Thus, it still remains an open problem to construct an IBE scheme over an RSA modulus (or to show impossibility result).

†Sangmyung University, Seoul, Korea. Email: jhpark@smu.ac.kr

‡Korea University, Seoul, Korea. Email: Kwangsu.lee@korea.ac.kr

§Korea University, Seoul, Korea. Email: donghlee@korea.ac.kr

1.1 Our Contribution

We present a new efficient (anonymous) IBE scheme that relies on a novel approach, what we call '*trapdoor subgroup*' of $\mathbb{Z}_n^*$ for an RSA modulus $n(=pq)$, where $p = 2p_1 + 1$ and $q = 2q_1 + 1$ for large primes p_1 and q_1. Roughly speaking, a trapdoor subgroup of $\mathbb{Z}_n^*$ is specified as (n, g), where g is a generator of the order-p_1q_1 subgroup of $\mathbb{Z}_n^*$. The important point is that the order of g modulo n must be hidden and can be used as a trapdoor in constructing our IBE scheme (and also our signature scheme). With a trapdoor subgroup of $\mathbb{Z}_n^*$ described, we present an IBE scheme that is secure against chosen plaintext attacks, and extend our scheme to achieve chosen ciphertext security by applying a variant [24] of Fujisaki-Okamoto [17] transformation. To prove the security of our IBE schemes in the random oracle model, we introduce a new computational hardness problem, called $\widetilde{q}$-Trapdoor Subgroup Diffie-Hellman (TSDH) problem for $\widetilde{q}$ private key queries, which arguably holds in the trapdoor subgroup of $\mathbb{Z}_n^*$. Our scheme is efficient in terms of both performance and space, where the 'performance' means the computational cost of key generation, encryption, and decryption, and the 'space' means the size of ciphertexts, private keys, and public parameters. Particularly, our scheme has the advantage in terms of performance, compared to the pairing-based schemes [9, 34, 5]. To add credence to the advantage, we give more concrete efficiency comparison that is based on our implementation using PBC (Pairing-Based Cryptography)[1] and Integer libraries.

Following Naor's observation (stated in [9]), we can suggest a new public-key signature scheme that is also based on a trapdoor subgroup of $\mathbb{Z}_n^*$. A notable feature of our signature scheme is that the signing algorithm is *exponentiation-free* and just requires one modular inversion in the hidden order p_1q_1. As a result, our signature scheme provides the *fastest* signing, compared to the current practical schemes such as RSA-FDH (Full-Domain-Hash) [3], RSA-PSS (Probabilistic Signature Scheme) [4], DSA, and ECDSA. Indeed, our signing algorithm can be about 62 times faster than RSA-{FDH, PSS}, and about 4 times faster than ECDSA at the current 112-bit security level. Instead, our verification algorithm is much slower than the others. Therefore, our signature scheme can be useful in applications where a device that has to perform signing algorithm is extremely resource-constrained. We prove the security of our signature scheme (in the random oracle model) based on a new computational hardness problem, called $\widetilde{q}$-Trapdoor Subgroup Exponent Inversion (TSEI) problem for $\widetilde{q}$ signature queries, which also arguably holds in the trapdoor subgroup of $\mathbb{Z}_n^*$. Finally, to provide confidence on the above two problems, we demonstrate that they hold in generic groups in the sense of [6, 37], under the assumption that factoring n is computationally infeasible .

1.2 Our Technique

With a simple example, we explain the basic idea behind our technique. Let (p, g, g^x) be public parameters where g is a generator of an order-p group and x is randomly chosen among $\{1, \ldots, p\}$. Assume that KGC generates a private key d for an identity ID as an exponential value $d = 1/(x + h(\mathsf{ID})) \in \mathbb{Z}_p$ using a collision-resistant hash function h. Apparently, this leads to an attack that easily allows an adversary to recover the master secret key x: with a private key d, the adversary has only to compute the inverse of d modulo p, and extract $h(\mathsf{ID})$ from the inverse. One method to prevent this attack is to raise the exponent d to the group element g, so that the private key for ID becomes $g^{1/(x+h(\mathsf{ID}))}$. This was the idea behind the Sakai-Kasahara IBE scheme [34] that relied on pairings.

A distinct method we consider is to hide the group order p from the public parameters so as to be hard to compute the inverse. To do so, we consider an order-p_1q_1 (trapdoor) subgroup of $\mathbb{Z}_n^*$ for a RSA modulus n described above. If factoring n is computationally infeasible and p_1q_1 is the product of large primes p_1 and

[1] https://crypto.stanford.edu/pbc/

q_1, then we can see that it is hard to find the order from the public n. One may wonder if a prime number p_1 or q_1 is a useful section of the subgroup order of $\mathbb{Z}_n^*$, but according to [27], setting such a prime order can easily lead to the factorization of n even without knowledge of the hidden order. From this observation, the public parameters are changed into (n, g, g^x), where g is a generator of a *composite* order-p_1q_1 subgroup of $\mathbb{Z}_n^*$ and x is randomly chosen among $\{1, \ldots, p_1q_1\}$. In this case, a private key d for an identity ID is generated as $d = 1/(x + h(\mathsf{ID})) \in \mathbb{Z}_{p_1q_1}$. The order of g modulo n is not revealed from the public parameters (n, g, g^x), so that it is computationally infeasible for the adversary to compute the inverse of d, unless the order p_1q_1 (or the exponent x) is found. This is the reason why such an exponential value d can be directly given out as a private key for an identity. In addition, because of the non-linear (i.e., inverse) structure of a private key, it is difficult for the adversary to generate a private key for a new identity even through collusion-attacks with (polynomially) many private keys. The key structure is applied to our signature scheme, where ID is replaced with a message m and then a signature on m is generated as $\sigma = 1/(x + h(m)) \in \mathbb{Z}_{p_1q_1}$. It follows that signing m can be done with a simple modular inversion without requiring any exponentiation, which makes our signing algorithm faster than the current practical signature schemes.

1.3 Related Work

Boneh and Franklin [9] presented the first practical IBE scheme based on pairings and defined the formal security notion for IBE . Since then, most of the subsequent IBE schemes [9, 34, 5, 38, 39, 14, 21] have been suggested to improve efficiency and security, based on pairings. Until now, three [9, 34, 5] of them have been considered as being practical constructions in terms of both performance and space.

Cocks [15] and Boneh et al. [10] constructed IBE schemes based on the QR problem. They use an RSA modulus n, but requires to compute the Jacobi symbol over the large composite n each time one-bit message (or one-bit ciphertext) is encrypted (or decrypted). For an ℓ-bit length of message, Cocks IBE scheme has a ciphertext that consists of 2ℓ elements in $\mathbb{Z}_n$ plus 2ℓ bits, which becomes a drawback in space although the size of private keys and public parameters is short. On the other hand, [10] has a shorter size of ciphertext that consists of 1 element in $\mathbb{Z}_n$ plus 2ℓ bits for an ℓ-bit message encrypted. However, the size of private key becomes of size $\mathcal{O}(\ell \cdot \log n)$ and the encryption time is $\mathcal{O}((\log n)^4)$ per a message bit.

Gentry et al. [18] demonstrated how to build an IBE system based on lattice. They constructed a dual scheme of Regev's public-key encryption [32], in which a public key corresponds to many equivalent private keys. Using the so-called 'preimage sampleable' trapdoor function as a basic primitive, they can extract a private key for an identity by mapping a hashed identity as a public key. The advantage of their scheme is the encryption and decryption time of $\mathcal{O}(\log n)$ per a message bit, but the drawback is in space: a size of ciphertext is $\mathcal{O}(\log n \cdot \log(\log n))$, and the size of private keys and public parameters is $O((\log n)^2)$.

The notion of IBE has been extended into hierarchical IBE [19], attribute-based encryption (ABE) [33], anonymous IBE [8], and Functional Encryption (FE) [11]. Many pairing-based works [7, 12, 23, 30, 26] (and more) have been suggested to realize those extended notions. Lattice-based IBE has also been extended toward hierarchical IBE [13, 1], and FE [2] constructions. As for the QR-based schemes, there does not exist any further extension, except the anonymous IBE [10], as far as we know.

Trapdoor subgroups. In 1991, Maurer and Yacobi [28] proposed an IBE scheme based on a trapdoor subgroup of $\mathbb{Z}_n^*$. A key idea in their construction is that a private key d for an identity ID can be generated as a discrete logarithm such that $g^d = h(\mathsf{ID}) \pmod n$ by using the trapdoor that is the factorization of an RSA modulus n. If d is easily computed to solve a discrete-logarithm problem (DLP), their scheme is simple and efficient. However, for some accepted level of security, n (and also factors of n) should be large (e.g., 2048 bits for the 112-bit security level) and then solving such a DLP needs a tremendous amount of work even if the factors of n are known as a trapdoor. Later, Paterson et al. [31] proved the security of [28] after a

slight modification, based on the FDH [3] technique. Lee et al. [25] suggested a pre-computation method to reduce the key generation time of [28], but for the current level of security their method still suffers from spending a long time in preparing a pre-computation table.

In 2005, Groth [20] demonstrated the cryptographic usefulness of a trapdoor subgroup of $\mathbb{Z}_n^*$ where an hidden order is used as a trapdoor, and by reducing the order of underlying groups they suggested a new homomorphic public-key encryption and commitment schemes. In 2009, Saxena and Soh [35] suggested an IBE scheme based on so-called 'Oracle-based Group with Infeasible Inversion', where 'Group with Infeasible Inversion' [22] has the property that it is easy to compute group operation, but hard to compute an inverse of group element. Their scheme, with no security proof, has a drawback that decryption algorithm has always to get access to an oracle in order to perform a group operation.

In 2015, independently to our work, Meshram [29] presented an IBE scheme based on both the integer factorization and discrete logarithm problems, which is substantially similar in philosophy to our technique. In comparison to our construction, their scheme can be viewed as using the similar encoding method to Waters' hash [38] so as to map a hashed identity to a relevant exponent in $\mathbb{Z}_p$ for the hidden prime order p. As a result, the drawback of their scheme is that the size of public parameters is expanded to $O(k \cdot \log n)$ for a security parameter k. Moreover, due to selecting a prime order of g modulo n, their scheme can be easily broken by factoring n.

2 Preliminaries

2.1 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme consists of the following algorithms:

- **Setup**(k) takes a security parameter k as input and outputs a public parameter PP and a master secret key msk.
- **KeyGen**(msk, ID) takes a master secret key msk, a public parameter PP and an identity $\mathsf{ID} \in \mathcal{ID}$ as inputs, where $\mathcal{ID}$ is an identity space. It outputs $\mathsf{sk}_{\mathsf{ID}}$, a private key for ID.
- **Encrypt**(PP, M, ID) takes a public parameter PP, a message $M \in \mathcal{M}$, and an identity $\mathsf{ID} \in \mathcal{ID}$ as inputs, where $\mathcal{M}$ is a message space. It outputs CT under ID, a ciphertext under ID.
- **Decrypt**(CT, PP, $\mathsf{sk}_{\mathsf{ID}}$) takes a ciphertext CT under ID', a public parameter PP, and a private key $\mathsf{sk}_{\mathsf{ID}}$ as inputs. It outputs a message M or $\perp$.

Correctness. For all $\mathsf{ID} \in \mathcal{ID}$ and all $M \in \mathcal{M}$, let $(\mathsf{PP}, \mathsf{msk}) \leftarrow$ **Setup**(k), $\mathsf{sk}_{\mathsf{ID}} \leftarrow$ **KeyGen**$(\mathsf{msk}, \mathsf{PP}, \mathsf{ID})$, $\mathsf{CT} \leftarrow$ **Encrypt**$(\mathsf{PP}, M, \mathsf{ID})$. We have $M \leftarrow$ **Decrypt**$(\mathsf{sk}_{\mathsf{ID}}, \mathsf{PP}, \mathsf{CT})$.

We next define the chosen ciphertext security [9] of an IBE scheme via the following game interacted by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- **Setup**: $\mathcal{C}$ runs the setup algorithm to obtain a public parameter PP and a master secret key msk. $\mathcal{C}$ gives PP to $\mathcal{A}$.
- **Query Phase 1**: $\mathcal{A}$ adaptively issues a number of queries where each query is one of:
  - Private key query on ID: $\mathcal{C}$ runs the key generation algorithm to obtain a private key for ID and gives the key $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$.

– Decryption query on $(\mathsf{CT}, \mathsf{ID})$: $\mathcal{C}$ runs the key generation algorithm to obtain $\mathsf{sk}_{\mathsf{ID}}$ to $\mathcal{A}$ and then runs the decryption algorithm using $\mathsf{CT}_{\mathsf{ID}}$ and $\mathsf{sk}_{\mathsf{ID}}$. It gives the resulting message to $\mathcal{A}$.

- **Challenge**: $\mathcal{A}$ outputs two equal-length messages M_0, M_1 and an identity ID^* on which it wishes to be challenged. The only restriction is that ID is not queried in Query Phase 1. $\mathcal{C}$ flips a coin $\sigma \in \{0,1\}$. $\mathcal{C}$ gives $\mathsf{CT}^* \leftarrow$ **Encrypt**$(\mathsf{PP}, M_\sigma, \mathsf{ID}^*)$ as a challenge ciphertext to $\mathcal{A}$.

- **Query Phase 2**: $\mathcal{A}$ adaptively issues a number of additional queries where each query is one of:

  – Private key query on ID, where $\mathsf{ID} \neq \mathsf{ID}^*$: $\mathcal{C}$ responds as in Phase 1.

  – Decryption query on $(\mathsf{CT}, \mathsf{ID})$, where $(\mathsf{CT}, \mathsf{ID}) \neq (\mathsf{CT}^*, \mathsf{ID}^*)$: $\mathcal{C}$ responds as in Query Phase 1.

- **Guess**: $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$. $\mathcal{A}$ wins if $\sigma' = \sigma$.

The advantage of $\mathcal{A}$ in breaking the chosen ciphertext security of an IBE scheme $\mathcal{IBE}$ is defined as $\mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{IBE},\mathcal{A}} = \left|\Pr[b' = b] - 1/2\right|$.

Definition 1. *We say that an IBE scheme is $(t, \varepsilon, q_K, q_D)$-IND-ID-CCA secure if for any polynomial time adversary $\mathcal{A}$ that runs in time at most t, issues at most q_K private key queries and at most q_D decryption queries in chosen ciphertext security games, we have that $\mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{IBE},\mathcal{A}} < \varepsilon$.*

As usual, we consider chosen plaintext security of IBE scheme by disallowing $\mathcal{A}$ to issue any decryption query in the above game. In that case, we say that an IBE scheme is (t, ε, q_K)-IND-ID-CPA secure.

2.2 Public Key Signature

A public key signature (PKS) scheme consists of the following algorithms:

- **Setup**(k): takes as input a security parameter k and outputs a public key PK and a secret key sk.

- **Sign**(sk, PK, m): takes a secret key sk, the public key PK, and a message $m \in \mathcal{M}$ as input and returns a signature σ.

- **Verify**(PK, m, σ): takes a public key PK, a message m , and a signature σ as input and returns accept or reject.

Correctness. For all $(\mathsf{PK}, \mathsf{sk})$ output by **Setup** and all $m \in \mathcal{M}$, we have **Verify**$(\mathsf{PK}, m,$ **Sign**$(\mathsf{sk}, \mathsf{PK}, m)) = \mathsf{accept}$.

We next give the definition of *strong* unforgeability under adaptive chosen message attacks [6] via an interaction between an adversary $\mathcal{A}$ (i.e., a forger against a signature scheme) and a challenger $\mathcal{C}$:

- **Setup**: $\mathcal{C}$ runs the setup algorithm to obtain a pair $(\mathsf{PK}, \mathsf{sk})$. It gives PK to $\mathcal{A}$ and keeps sk secret.

- **Query Phase**: $\mathcal{A}$ issues signature queries on messages $\{m_i\}$ that can be adaptively chosen, depending on previous signatures and messages. Using sk, $\mathcal{C}$ runs the signing algorithm for each message and returns a resulting signature as a response.

- **Output**: $\mathcal{A}$ outputs a valid signature σ^* and a message m^* such that: (1) **Verify**$(\mathsf{PK}, m^*, \sigma^*) = \mathsf{accept}$, and (2) $(m^*, \sigma^*) \notin \Sigma$, where Σ is the set of pairs (m_i, σ_i) such that σ_i was the response to a signature query.

The advantage of $\mathcal{A}$ that breaks the strong unforgeability of a signature scheme $\mathcal{PKS}$ is defined as

$$\begin{aligned}&\mathbf{Adv}^{\mathrm{suf}}_{\mathcal{PKS},\mathcal{A}}(k)\\&\quad=\Pr\Big[\mathcal{A}\to(m^*,\sigma^*):\mathbf{Verify}(\mathsf{PK},m^*,\sigma^*)=\mathsf{accept}\bigwedge(m^*,\sigma^*)\notin\Sigma\Big].\end{aligned}$$

Definition 2. *We say that a signature scheme $\mathcal{PKS}$ is (t,ε,q_S)-secure in the sense of strong unforgeability if no adversary that runs in time at most t and issues at most q_S signature queries breaks the strong unforgeability with advantage at most ε.*

2.3 Trapdoor Subgroups and Complexity Assumptions

Trapdoor Subgroups: Let n be a product of two primes p and q such that $p = 2p_1\cdots p_i+1$ and $q = 2q_1\cdots q_j+1$ for odd primes p_k for $k=1,\ldots,i$ and q_k for $k=1,\ldots,j$. We simply consider the case when $i=1$ and $j=1$, in which case such a prime p (and q) is called a safe prime. The notation '$\mathrm{ord}_n g$' is defined as the least positive integer x such that $g^x \equiv 1 \pmod n$.

A number theory shows that there exists the multiplicative group $\mathbb{Z}_n^* := \{u\in\mathbb{Z}_n \mid \gcd(u,n)=1, n=pq\}$ that has $\phi(n)$ group elements where ϕ is the Euler-phi function. Notice that $\phi(n) = 2p_1\cdot 2q_1$. Then, we can consider a (cyclic) subgroup of $\mathbb{Z}_n^*$ whose order is the *composite* number p_1q_1 by fining a generator $g\in\mathbb{Z}_n^*$ such that $\mathrm{ord}_n g = p_1q_1$. The point is that the composite order must be the product of two primes, each of which comes from p and q, respectively. As pointed out in [27], the prime number p_1 or q_1 cannot be a subgroup order, since otherwise such a prime order setting easily leads to the factorization of n.

We say that the subgroup $\mathbb{G}$, which is determined by (n,g), is a *trapdoor subgroup* of $\mathbb{Z}_n^*$. The term 'trapdoor' means that $\mathrm{ord}_n g$ of the subgroup of $\mathbb{Z}_n^*$ should be hidden and can be used as a trapdoor. Under the assumption that factoring n is computationally infeasible, the trapdoor subgroup $\mathbb{G}$ of $\mathbb{Z}_n^*$ has the following properties:

1. *Hidden order*: finding $\mathrm{ord}_n g$ is computationally infeasible,
2. *Exponentiation-computable*: for a positive integer x and a group element $g_1\in\mathbb{G}$, it is easy to compute $g_1^x \pmod n$, without knowing $\mathrm{ord}_n g$.
3. *Inversion-infeasible*: for a positive integer x, it is hard to compute the inverse x^{-1} such that $x\cdot x^{-1}\equiv 1 \pmod{\mathrm{ord}_n g}$, without knowing $\mathrm{ord}_n g$.

Also, we introduce the following (well-known) lemma that plays an important role in our security analysis. We skip the proof of Lemma 1.

Lemma 1. *If g and n are relatively prime integers with $n>0$, then $g^i \equiv g^j \pmod n$, where i and j are nonnegative integers, if and only if $i\equiv j \pmod{\mathrm{ord}_n g}$.*

The $\widetilde{q}$-Trapdoor Subgroup Diffie-Hellman (TSDH) Problem: The $\widetilde{q}$-TSDH problem is defined as follows: given $\big(n,g,g^x,g^{(x+r^*)y},r^*,\{1/(x+r_i),r_i\}_{i=1}^{\widetilde{q}}\big)$ as input under the condition that (1) g is the generator of order-p_1q_1 trapdoor subgroup of $\mathbb{Z}_n^*$, (2) g, g^x, and $g^{(x+r^*)y}$ are in $\mathbb{Z}_n^*$, (3) r^* and r_i for $i=1,\ldots,\widetilde{q}$ are in $\{0,1\}^\ell$ for some ℓ (less than $\log(\mathrm{ord}_n g)$), and (4) $1/(x+r_i)$ for $i=1,\ldots,\widetilde{q}$ are in $\mathbb{Z}_{\mathrm{ord}_n g}$, output g^y in $\mathbb{Z}_n^*$. We say that an algorithm $\mathcal{A}$ that outputs g^y has an advantage $\mathbf{Adv}^{\widetilde{q}\text{-TSDH}}_{(n,g),\mathcal{A}}=\varepsilon$ in solving the $\widetilde{q}$-TSDH problem in the trapdoor subgroup of $\mathbb{Z}_n^*$ if

$$\Pr\Big[\mathcal{A}\big(n,g,g^x,g^{(x+r^*)y},r^*,\{1/(x+r_i),r_i\}_{i=1}^{\widetilde{q}}\big)=g^y\Big]\geq\varepsilon,$$

where the probability is taken over the random choice of x, $y \in \mathbb{Z}_{\mathrm{ord}_n g}$, the random choice of $r^*, \{r_i\}_{i=1}^{\widetilde{q}}$ in $\{0,1\}^\ell$, and the random bits used by $\mathcal{A}$.

Definition 3. *We say that the $(t, \varepsilon, \widetilde{q})$-TSDH assumption holds in the trapdoor subgroup of $\mathbb{Z}_n^*$ if no polynomial time adversary $\mathcal{A}$ that runs in time at most t has at least advantage ε in solving the $\widetilde{q}$-TSDH problem in the trapdoor subgroup of $\mathbb{Z}_n^*$.*

The $\widetilde{q}$-Trapdoor Subgroup Exponent Inversion (TSEI) Problem: The $\widetilde{q}$-TSEI problem is defined as follows: given $\big(n, g, g^x, \{1/(x+r_i), r_i\}_{i=1}^{\widetilde{q}}\big)$ as input under the condition that (1) g is the generator of order-p_1q_1 trapdoor subgroup of $\mathbb{Z}_n^*$, (2) g and g^x are in $\mathbb{Z}_n^*$, (3) r_i for $i = 1, \ldots, \widetilde{q}$ are in $\{0,1\}^\ell$ for some ℓ (less than $\log(\mathrm{ord}_n g)$), and (4) $1/(x+r_i)$ for $i = 1, \ldots, \widetilde{q}$ are in $\mathbb{Z}_{\mathrm{ord}_n g}$, output a new pair $(1/(x+r^*), r^*) \in \mathbb{Z}_{\mathrm{ord}_n g} \times \{0,1\}^\ell$. We say that an algorithm $\mathcal{A}$ that outputs $(1/(x+r^*), r^*)$ has an advantage $\mathbf{Adv}_{(n,g),\mathcal{A}}^{\widetilde{q}\text{-TSEI}} = \varepsilon$ in solving the $\widetilde{q}$-TSEI problem in the trapdoor subgroup of $\mathbb{Z}_n^*$ if

$$\Pr\Big[\mathcal{A}\big(n, g, g^x, \{1/(x+r_i), r_i\}_{i=1}^{\widetilde{q}}\big) = (1/(x+r^*), r^*)\Big] \geq \varepsilon,$$

where the probability is taken over the random choice of $x \in \mathbb{Z}_{\mathrm{ord}_n g}$, the random choice of $\{r_i\}_{i=1}^{\widetilde{q}}$ in $\{0,1\}^\ell$, and the random bits used by $\mathcal{A}$.

Definition 4. *We say that the $(t, \varepsilon, \widetilde{q})$-TSEI assumption holds in the trapdoor subgroup of $\mathbb{Z}_n^*$ if no polynomial time adversary $\mathcal{A}$ that runs in time at most t has at least advantage ε in solving the $\widetilde{q}$-TSEI problem in the trapdoor subgroup of $\mathbb{Z}_n^*$.*

To gain confidence on the above two problems, we prove in Section 5 that they hold in generic groups in the sense of [37, 6], under the assumption that factoring n is computationally infeasible .

3 New Identity-Based Encryption Scheme

3.1 CPA-Secure Construction

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

1. Generate two large random safe primes p and q, each roughly the same size, where $p = 2p_1 + 1$ and $q = 2q_1 + 1$ for primes p_1 and q_1.
2. Compute $n = pq$.
3. Select a random $g \in \mathbb{Z}_n^*$ such that $\mathrm{ord}_n g = p_1 q_1$.
4. Pick a random $x \in \mathbb{Z}_{\mathrm{ord}_n g}$ and set $g_1 = g^x \pmod n$.
5. Select two hash functions $h : \{0,1\}^* \to \{0,1\}^\ell$, where $\ell < \log(\mathrm{ord}_n g)$, and $H : \mathbb{Z}_n \to \{0,1\}^\delta$.
6. Output $\mathsf{PP} = (n, g, g_1, h, H)$ and $\mathsf{msk} = (x, \mathrm{ord}_n g)$.

KeyGen(msk, ID)**:** To create a private key $\mathsf{sk}_{\mathsf{ID}}$ for an identity $\mathsf{ID} \in \mathcal{ID} = \{0,1\}^*$, the key generation algorithm does as follows:

1. Compute $h(\mathsf{ID}) \in \{0,1\}^\ell$. Note that $h(\mathsf{ID}) \in \mathbb{Z}_{\mathrm{ord}_n g}$, since $\ell < \log(\mathrm{ord}_n g)$.
2. Check if $\gcd(x + h(\mathsf{ID}), \mathrm{ord}_n g) \neq 1$. If so, abort.

3. Otherwise, compute $\mathsf{sk}_{\mathsf{ID}}$ such that $(x+h(\mathsf{ID}))\cdot \mathsf{sk}_{\mathsf{ID}} \equiv 1 \pmod{\mathrm{ord}_n g}$.

4. Output the private key $\mathsf{sk}_{\mathsf{ID}} = 1/(x+h(\mathsf{ID})) \in \mathbb{Z}_{\mathrm{ord}_n g}$.

Encrypt($\mathsf{PP}, \mathsf{ID}, M$)**:** To encrypt a message $M \in \{0,1\}^\delta$ under an identity $\mathsf{ID} \in \mathcal{ID} = \{0,1\}^*$, the encryption algorithm does as follows:

1. Compute $h(\mathsf{ID}) \in \{0,1\}^\ell$ and pick a random $s \in \mathbb{Z}_n$.

2. Compute $C_0 = g^s \pmod n$, $C_1 = (g_1 g^{h(\mathsf{ID})})^s \pmod n$.

3. Compute $C_2 = H(C_0) \oplus M \in \{0,1\}^\delta$.

4. Output the ciphertext $\mathsf{CT} = (C_1, C_2) \in \mathbb{Z}_n \times \{0,1\}^\delta$.

Decrypt(CT, $\mathsf{sk}_{\mathsf{ID}}$)**:** To decrypt a ciphertext $\mathsf{CT} = (C_1, C_2)$ using a private key $\mathsf{sk}_{\mathsf{ID}}$ for an identity ID, the decryption algorithm does as follows:

1. Compute $C_0 = C_1^{\mathsf{sk}_{\mathsf{ID}}} \pmod n$ and compute $M = H(C_0) \oplus C_2$.

2. Output the message $M \in \{0,1\}^\delta$.

Correctness. We can use the fact that $g^{\mathrm{ord}_n g} = 1 \pmod n$ and $(x+h(\mathsf{ID}))\cdot \mathsf{sk}_{\mathsf{ID}} = 1 + k \cdot \mathrm{ord}_n g$ for some positive integer $k \in \mathbb{Z}$. Then the correctness of the decryption algorithm can be verified as follows:

$$\begin{aligned} C_1^{\mathsf{sk}_{\mathsf{ID}}} \pmod n &= \left((g_1 g^{h(\mathsf{ID})})^s\right)^{\mathsf{sk}_{\mathsf{ID}}} \pmod n = \left(g^{(x+h(\mathsf{ID}))s}\right)^{\mathsf{sk}_{\mathsf{ID}}} \pmod n \\ &= \left(g^{(x+h(\mathsf{ID}))\cdot \mathsf{sk}_{\mathsf{ID}}}\right)^s \pmod n = g^{(1+k\cdot \mathrm{ord}_n g)s} \pmod n \\ &= g^s \left(g^{\mathrm{ord}_n g}\right)^{ks} \pmod n = g^s \pmod n. \end{aligned}$$

Remark 1. The problem of finding $\mathrm{ord}_n g = p_1 q_1$ from the public parameters (n, g, g_1) is equivalent to that of factoring n. This can be shown as follows: if $\mathrm{ord}_n g = p_1 q_1$ of g modulo n is found, an adversary can also obtain a value $p_1 + q_1$ from the equation $n = 4p_1 q_1 + 2(p_1 + q_1) + 1$. From the two values $\mathrm{ord}_n g$ and $p_1 + q_1$, the adversary can have p_1 and q_1 and easily factor n. The converse is trivial.

Remark 2. Given at least one private key $\mathsf{sk}_{\mathsf{ID}}$ for ID, the problem of computing the discrete logarithm $x \in \mathbb{Z}_{\mathrm{ord}_n g}$ such that $g^x = g_1 \pmod n$ is also equivalent to that of factoring n. This can be shown as follows: if x is found, an adversary can have the equation $(x+h(\mathsf{ID}))\cdot \mathsf{sk}_{\mathsf{ID}} = 1 + k \cdot \mathrm{ord}_n g$ for some integer $k \in \mathbb{Z}$. Therefore, $g^{(x+h(\mathsf{ID}))\cdot \mathsf{sk}_{\mathsf{ID}} - 1} = 1 \pmod n$. Let $(x+h(\mathsf{ID}))\cdot \mathsf{sk}_{\mathsf{ID}} - 1 = 2^s t$, where t is an odd integer. Then, the order of $g^{2^{s-1}t}$ modulo n is 2, and then $\gcd(g^{2^{s-1}t} - 1, n)$ or $\gcd(g^{2^{s-1}t} + 1, n)$ is a non-trivial factor of n. Conversely, if n is factored, then $\mathrm{ord}_n g$ is easily obtained. Then, an adversary can obtain the inverse of $\mathsf{sk}_{\mathsf{ID}}$ (mod $\mathrm{ord}_n g$) and the discrete logarithm x from extracting $h(\mathsf{ID})$.

Remark 3. A number theory shows that there exist $3(p_1-1)(q_1-1)$ elements[2] of the order $2p_1q_1$ modulo n and $(p_1-1)(q_1-1)$ elements of the order p_1q_1 modulo n, which are almost all portion of $\mathbb{Z}_n^*$. If the setup algorithm picks an element g of the order $2p_1q_1$, then it simply has to compute $g^2 \pmod n$ to obtain an element of the order p_1q_1, which is from the equation $\mathrm{ord}_n(g^u) = \mathrm{ord}_n g / \gcd(u, \mathrm{ord}_n g)$.

Remark 4. The probability that the key generation algorithm aborts in the Step 2 is at most $\frac{1}{p_1} + \frac{1}{q_1} - \frac{1}{p_1 q_1}$ for the prefixed $x \in \mathbb{Z}_{\mathrm{ord}_n g}$, and if the order is determined by two large primes (e.g., 1023 bits at the current

[2]More precisely, in $\mathbb{Z}_n^*$ with n understood, there exist $3(p_1-1)(q_1-1)$ elements of the order $2p_1q_1$, $(p_1-1)(q_1-1)$ elements of the order p_1q_1, $3(p_1-1)$ elements of the order $2p_1$, p_1-1 elements of the order p_1, $3(q_1-1)$ elements of the order $2q_1$, q_1-1 elements of the order q_1, 3 elements of the order 2, and 1 element of the order 1.

112-bit security level), the probability is negligible. One way to completely avoid such a negligible case is that x and ℓ are selected in such a way that the bitlength of x and the output length ℓ of h are less than $\log(\min\{p_1, q_1\}) - 1$ bits. In this case, the addition value $x + h(\mathsf{ID})$ for any identity ID remains always less than p_1 or q_1, and thus $\gcd(x + h(\mathsf{ID}), \mathrm{ord}_n g) = 1$ with probability 1.

Remark 5. As in RSA, it is required to avoid a small decryption exponent $\mathsf{sk}_{\mathsf{ID}} \in \mathbb{Z}_{\mathrm{ord}_n g}$. The exponent (i.e., a private key) is not directly chosen at random by the key generation algorithm, and determined by the equation:

$$(x + h(\mathsf{ID})) \cdot \mathsf{sk}_{\mathsf{ID}} \equiv 1 \pmod{\mathrm{ord}_n g}. \tag{1}$$

Therefore, the small decryption exponent problem can happen if $\mathsf{sk}_{\mathsf{ID}}$ unavoidably becomes a 'short' size (e.g., 30 bits) with respect to some identities. In addition, an adversary gets the largest exponent L among a set $\{\mathsf{sk}_{\mathsf{ID}}\}$ of acquired private keys, and next tries to find $\mathrm{ord}_n g$ by beginning with exponentiation $g^R \pmod n$ such as $R > L$. If the largest value L gets close to the (unknown) order $\mathrm{ord}_n g$, the adversary can gain computational advantage in finding the order of g by simply excluding considerable amount of exponents less than L.

Solving these small or large exponent problems simultaneously is to select an appropriate discrete logarithm x in $\mathbb{Z}_{\mathrm{ord}_n g}$. The equation (1) shows that, roughly speaking, $\log f_1 < \log(\mathsf{sk}_{\mathsf{ID}})$, where f_1 is the quotient $\lfloor \mathrm{ord}_n g / (x + h(\mathsf{ID})) \rfloor$. Such the lower bound means that if we choose x as an appropriate size as long as the trapdoor subgroup discrete-logarithm problem (DLP) is hard in the subgroup of $\mathbb{Z}_n^*$, then we can make the size of private key always larger than $\log f_1$ bits. Also, we can show that $\log(\mathsf{sk}_{\mathsf{ID}}) \leq \log(\mathrm{ord}_n g - f_1)$ as the upper bound[3]. Such the upper bound implies that a private key $\mathsf{sk}_{\mathsf{ID}}$ is at least $\log f_1$ *bits distance* away from the (unknown) order $\mathrm{ord}_n g$, and thus even with the largest decryption exponent L the adversary has to add at least $\log f_1$ bits into L and do exponentiations from it. It seems that what is worse to the adversary is that the value f_1 is unknown and changes per an identity ID. As an instance at the current 112-bit security level, if $\mathrm{ord}_n g = p_1 q_1$ is chosen as the product of two 1023-bit primes and the bitlength of x as about 1021 bits and the output length ℓ of h as 256 bits, then the size of a private key for any identity is at least 1025 bits and the largest private key is at least 1025 bits away from the 2046-bit order $\mathrm{ord}_n g$. In such a case when the bitlength of x is artificially reduced from $\log(\mathrm{ord}_n g)$ to about $[\log(\mathrm{ord}_n g)]/2$, we have to ensure that it should be computationally infeasible to find the discrete logarithm x such that $g_1 = g^x \pmod n$ on the trapdoor subgroup of $\mathbb{Z}_n^*$.

Remark 6. The ciphertext size is one element in $\mathbb{Z}_n$ plus δ-bit string. The cost of generating a private key for an identity is just a modular inversion in $\mathbb{Z}_{\mathrm{ord}_n g}$, and the decryption cost is one exponentiation in modulus n by rasing an about $\log n$-bit exponent. The encryption cost is three exponentiations g^s, $g^{h(\mathsf{ID})s}$, and g_1^s in modulus n for an about $\log n$-bit randomly chosen exponent s, but they can be calculated in fixed bases g and g_1. The size of the exponent $h(\mathsf{ID})s$ in encryption becomes about $\ell + \log n$ bits and thus $h(\mathsf{ID})s$ can get larger than $\mathrm{ord}_n g$ (as a number), but $h(\mathsf{ID})s$ cannot be reduced modulo $\mathrm{ord}_n g$ without knowing the order $\mathrm{ord}_n g$ of g modulo n. Thus, in case of using fixed-base exponentiations, the values of $g^j \pmod n$ should be precomputed for $j = 1, \ldots, \ell + \log n$.

Remark 7. It is worth noting that our scheme can be constructed under a smaller size of subgroups as long as the $\widetilde{q}$-TSDH assumption holds in the smaller trapdoor subgroups of $\mathbb{Z}_n^*$ for the $\widetilde{q}$ number of adversarial queries. For instance, if $n = pq = (2p_1p_2 + 1)(2q_1q_2 + 1)$ for shorter size of primes p_1, p_2, q_1 and q_2, there are four possible combinations for group orders such as p_1q_1, p_1q_2, p_2q_1, and p_2q_2. In such a smaller subgroup, all of the efficiency factors (except the ciphertext size) can be improved further. For instance, if the bitlength of $\mathrm{ord}_n g$ becomes about 448 bits, the size of private key is reduced into at least a quarter of 2046

[3] We give the analysis of this upper (and lower) bound in Appendix A.

bits, and all the exponentiations in encryption (by choosing a randomness s of about $(\log n)/4$ bitlength) and decryption can be performed under exponents whose bitlengths are roughly from 448-704 bits (instead of about 2046-2302 bits).

3.2 Chosen Plaintext Security

Theorem 1. *Let h and H be modeled as random oracles and $q_h < \widetilde{q}$. Suppose the $(t', \varepsilon', \widetilde{q})$-TSDH assumption holds in the trapdoor subgroup of $\mathbb{Z}_n^*$. Then our IBE system is (t, ε, q_K)-IND-ID-CPA secure, where*

$$\varepsilon \cdot \frac{2}{q_h q_H} \leq \varepsilon', \qquad t \approx t'.$$

Here, $\{q_h, q_H\}$ is the number of $\{h, H\}$ queries, respectively.

Proof. Suppose that there exists an adversary $\mathcal{A}$ which can break the CPA security of our IBE system. We show how to build an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve a $\widetilde{q}$-TSDH problem in the trapdoor subgroup of $\mathbb{Z}_n^*$, with n understood in the scheme description. On input $\big(n, g, g^x, g^{(x+r^*)y}, r^*, \{1/(x+r_i), r_i\}_{i=1}^{\widetilde{q}}\big)$, $\mathcal{B}$ tries to output g^y in $\mathbb{Z}_n^*$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup $\mathcal{B}$ sets $g_1 = g^x$ and gives $\mathsf{PP} = (n, g, g_1, h, H)$ to $\mathcal{A}$.

Query Phase 1 $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries as follows:

h queries: $\mathcal{B}$ picks a random i^* from $\{1, \ldots, q_h\}$. $\mathcal{B}$ maintains a list of tuples $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$, referred to as the h^{list}. Given ID_i, $\mathcal{B}$ scans through the h^{list} to see if ID_i appears in a tuple $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$. If it does, $\mathcal{B}$ responds with $h(\mathsf{ID}_i)$. Otherwise, $\mathcal{B}$ selects a value $r_i \in \{0,1\}^\ell$ among $\{r_j\}_{j=1}^{\widetilde{q}}$ and sets $h(\mathsf{ID}_i) = r_i$. $\mathcal{B}$ adds the new tuple $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$ to the h^{list} and responds with $h(\mathsf{ID}_i)$. If $i = i^*$, $\mathcal{B}$ uses r^* instead of r_i.

H queries: $\mathcal{B}$ maintains a list of tuples $< A_i, w_i >$, referred to as the H^{list}. Given $A_i \in \mathbb{Z}_n$, $\mathcal{B}$ scans through the H^{list} to see if A_i appears in a tuple $< A_i, w_i >$. If it does, $\mathcal{B}$ responds with $H(A_i) = w_i$. Otherwise, $\mathcal{B}$ picks a random $w_i \in \{0,1\}^\delta$ and sets $H(A_i) = w_i$. $\mathcal{B}$ adds the new tuple $< A_i, w_i >$ to the H^{list} and responds with $H(A_i)$.

Key queries: Given $\mathsf{ID}_i \in \mathcal{ID}$, $\mathcal{B}$ scans through the h^{list} to find a tuple $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$. Let $h(\mathsf{ID}_i) = r_i$. If $\mathsf{ID}_{i^*} = \mathsf{ID}_i$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ selects the corresponding $1/(x+r_i) \in \mathbb{Z}_{\mathrm{ord}_{ng}}$ among $\{1/(x+r_j)\}_{j=1}^{\widetilde{q}}$ as $\mathsf{sk}_{\mathsf{ID}_i}$. $\mathcal{B}$ gives $1/(x+r_i)$ to $\mathcal{A}$.

Challenge $\mathcal{A}$ outputs $M_0, M_1 \in \{0,1\}^\delta$ and ID^*. If $\mathsf{ID}_{i^*} \neq \mathsf{ID}^*$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ uses the r^* as $h(\mathsf{ID}^*)$ and sets $C_1^* = g^{(x+r^*)y} \in \mathbb{Z}_n$. Notice that $C_1 = g^{(x+r^*)y} = (g_1 g^{h(\mathsf{ID}^*)})^y$ under $s = y$. Next, $\mathcal{B}$ picks a random $R \in \{0,1\}^\delta$ and sets $C_2^* = R$. $\mathcal{B}$ gives $\mathsf{CT}^* = (C_1^*, C_2^*)$ to $\mathcal{A}$.

Query Phase 2 $\mathcal{A}$ issues more h, H, and key queries on the constraint that the private key query for ID^* is not allowed. $\mathcal{B}$ responds as in Query Phase 1.

Guess $\mathcal{A}$ outputs a guess $b \in \{0,1\}$. At this point, $\mathcal{B}$ picks a random tuple $< A_i, w_i >$ from the H^{list} and outputs w_i as the solution to the given $\widetilde{q}$-TSDH problem.

Analysis. We see that the computation that $\mathcal{B}$ requires is almost the same as $\mathcal{A}$'s computation that needs to break the CPA security of the IBE scheme. Next, to analyze $\mathcal{B}$'s advantage, we prove the following claims.

Claim 1: The probability that $\mathcal{B}$ dost not abort is at least $1/q_h$.

Proof. Let $\overline{\mathsf{abort}}$ be the event that $\mathcal{B}$ does not abort during the simulation. Since the selection of i^* is independent of $\mathcal{A}$'s view, the probability that $\mathsf{ID}^* = \mathsf{ID}_{i^*}$ is at least $1/q_h$. Thus, the probability $\Pr[\overline{\mathsf{abort}}]$ is at least $1/q_h$. $\square$

Unless $\mathcal{B}$ does not abort, $\mathcal{B}$ is simulating a real attack environment for $\mathcal{A}$. Let $\mathcal{H}$ be the event that $\mathcal{A}$ issues g^y as an H query at some point, which implies that at the end of the simulation g^y (as one of values $\{A_i\}_{i=1}^{q_H}$) appears in the H^{list}. We show that $\Pr[\mathcal{H}] \geq 2\varepsilon$, by using the same proof strategy as in [9].

Claim 2: (from [9]) $\Pr[\mathcal{H}] \geq 2\varepsilon$.

As long as $\mathcal{B}$ does not abort, $\mathcal{B}$ can use the $\mathcal{A}$'s advantage to solve the $\widetilde{q}$-TSDH problem. With the probability at least 2ε from Claim 2, the correct answer g^y appears in some tuple on the H^{list}. At the end of simulation, $\mathcal{B}$ picks a random value from $\{A_i\}_{i=1}^{q_H}$ in the H^{list}, so that $\mathcal{B}$ produces the answer with probability at least $2\varepsilon/q_H$. This is done with the probability at least $1/q_h$ from Claim 1, implying that $\mathcal{B}$ does not abort. Hence, by putting them all together, we can get the probability $\mathcal{B}$ produces the answer for the $\widetilde{q}$-TSDH problem at least $2\varepsilon/q_H q_h$. This concludes the proof of Theorem 1. ■

Anonymity. We can also prove that our IBE scheme is anonymous under the same $\widetilde{q}$-TSDH assumption. To prove it, we create a sequence of hybrid games which differ by the challenge ciphertext (CT^*) given to the adversary: $\mathsf{Game}_0 : \mathsf{CT}^* = (C_1 = (g_1 g^{h(\mathsf{ID}^*)})^s, C_2)$, $\mathsf{Game}_1 : \mathsf{CT}^* = (C_1 = (g_1 g^{h(\mathsf{ID}^*)})^s, R_2)$, $\mathsf{Game}_2 : \mathsf{CT}^* = (C_1 = (g_1 g^{h(R)})^{\widetilde{s}}, R_2)$, where s, $\widetilde{s}$, R and R_2 are randomly chosen. An adversary's ability to distinguish between Game_0 and Game_1 can be transformed to solve the $\widetilde{q}$-TSDH problem as in the proof of Theorem 1 (under the equivalent conversion from the left-or-right game to the real-or-random game). Next, it is easy to see that Game_1 and Game_2 are statistically identical. By rewinding the sequence of the hybrid games in the reverse order, we can show that, under the $\widetilde{q}$-TSDH assumption, it is infeasible for the adversary to distinguish between two games where CT^* is generated under (ID_0, M) and (ID_1, M), respectively.

3.3 CCA-Secure Construction

We extend our IBE scheme to achieve chosen-ciphertext security by applying the variant of the Fujisaki-Okamoto transform [17]. The variant [24] (denoted by $\mathsf{FO}_{\mathsf{ID}}$, hereafter) allows for achieving CCA security from any CPA-secure IBE scheme in the random oracle model, while preserving security reduction tightly.

Setup(k)**:** As in the CPA-secure scheme, except for selecting hash functions. The setup algorithm picks three hash functions $h : \{0,1\}^* \to \{0,1\}^\ell$, where $\ell < \log(\mathrm{ord}_n g)$, $H : \mathbb{Z}_n \to \{0,1\}^{\delta+\theta}$, and $\widetilde{H} : \{0,1\}^* \to \{0,1\}^{\lceil \log n \rceil}$. The algorithm outputs $\mathsf{PP} = (n, g, g_1, h, H, \widetilde{H})$ and $\mathsf{msk} = (x, \mathrm{ord}_n g)$.

KeyGen(msk, ID)**:** As in the CPA-secure scheme.

Encrypt(PP, ID, M)**:** To encrypt a message $M \in \{0,1\}^\delta$ under an identity $\mathsf{ID} \in \mathcal{ID} = \{0,1\}^*$, the encryption algorithm does as follows:

1. Pick a random $\rho \in \{0,1\}^\theta$.
2. Compute $s = \widetilde{H}(M, \mathsf{ID}, \rho) \in \{0,1\}^{\lceil \log n \rceil}$ and $h(\mathsf{ID}) \in \{0,1\}^\ell$.
3. Compute $C_0 = g^s \pmod n$, $C_1 = (g_1 g^{h(\mathsf{ID})})^s \pmod n$.
4. Compute $C_2 = H(C_0) \oplus (M || \rho) \in \{0,1\}^{\delta+\theta}$.
5. Output the ciphertext $\mathsf{CT} = (C_1, C_2) \in \mathbb{Z}_n \times \{0,1\}^{\delta+\theta}$.

Decrypt(CT, PP, $\mathsf{sk}_{\mathsf{ID}}$)**:** To decrypt a ciphertext $\mathsf{CT} = (C_1, C_2)$ using a private key $\mathsf{sk}_{\mathsf{ID}}$ for an identity ID, the decryption algorithm does as follows:

1. Compute $C_0 = C_1^{\mathsf{sk}_{\mathsf{ID}}} \pmod n$.

2. Compute $M||\rho = H(C_0) \oplus C_2$, where $M \in \{0,1\}^{\delta}$ and $\rho \in \{0,1\}^{\theta}$.

3. Compute $s = \widetilde{H}(M, \mathsf{ID}, \rho) \in \{0,1\}^{\lceil \log n \rceil}$ and $h(\mathsf{ID}) \in \{0,1\}^{\ell}$.

4. Check if $C_1 \stackrel{?}{=} (g_1 g^{h(\mathsf{ID})})^s \pmod{n}$ holds. If not, output reject.

5. Otherwise, output the message $M \in \{0,1\}^{\delta}$.

3.4 Chosen Ciphertext Security

Theorem 2. *Let h, H, and $\widetilde{H}$ be modeled as random oracles and $q_h < \widetilde{q}$. Suppose the $(t', \varepsilon', \widetilde{q})$-TSDH assumption holds in the trapdoor subgroup of $\mathbb{Z}_n^*$. Then our IBE system is $(t, \varepsilon, q_K, q_D)$-IND-ID-CCA secure, where*

$$\varepsilon \cdot \frac{2\varepsilon}{q_H q_h}\left(1 - \frac{q_{\widetilde{H}}}{2^{\theta}}\right)\left(1 - \frac{q_D}{\mathrm{ord}_n g}\right) \leq \varepsilon', \qquad t' \approx t + \mathcal{O}(t_e \cdot q_{\widetilde{H}}).$$

Here, $\{q_h, q_H, q_{\widetilde{H}}\}$ is the number of $\{h, H, \widetilde{H}\}$ oracle queries, respectively, and θ is the bitlength of a randomness used in encryption, and $\mathrm{ord}_n g$ *is the order of g modulo n, and t_e is the exponentiation time in $\mathbb{Z}_n$.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ which can break the CCA security of our IBE system. We show how to build an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve a $\widetilde{q}$-TSDH problem in the trapdoor subgroup of $\mathbb{Z}_n^*$, with n understood in the scheme description. On input $\left(n, g, g^x, g^{(x+r^*)y}, r^*, \left\{1/(x+r_i), r_i\right\}_{i=1}^{\widetilde{q}}\right)$, $\mathcal{B}$ tries to output g^y in $\mathbb{Z}_n^*$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup $\mathcal{B}$ sets $g_1 = g^x$ and gives $\mathsf{PP} = (n, g, g_1, h, H, \widetilde{H})$ to $\mathcal{A}$.

Query Phase 1 $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries as follows:

h queries: As before, $\mathcal{B}$ picks a value i^* from $\{1, \ldots, q_h\}$. $\mathcal{B}$ maintains a list of tuples $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$, referred to as the h^{list}. Given ID_i, $\mathcal{B}$ scans through the h^{list} to see if ID_i appears in a tuple $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$. If it does, $\mathcal{B}$ responds with $h(\mathsf{ID}_i)$. Otherwise, $\mathcal{B}$ selects a value r_i among $\{r_j\}_{j=1}^{\widetilde{q}}$ and sets $h(\mathsf{ID}_i) = r_i$. $\mathcal{B}$ adds the new tuple $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$ to the h^{list} and responds with $h(\mathsf{ID}_i)$. If $i = i^*$, $\mathcal{B}$ uses r^* instead of r_i.

H queries: $\mathcal{B}$ maintains a list of tuples $< A_i, w_i >$, referred to as the H^{list}. Given $A_i \in \mathbb{Z}_n$, $\mathcal{B}$ scans through the H^{list} to see if A_i appears in a tuple $< A_i, w_i >$. If it does, $\mathcal{B}$ responds with $H(A_i) = w_i$. Otherwise, $\mathcal{B}$ picks a random $w_i \in \{0,1\}^{\delta+\theta}$ and sets $H(A_i) = w_i$. $\mathcal{B}$ adds the new tuple $< A_i, w_i >$ to the H^{list} and responds with $H(A_i)$.

$\widetilde{H}$ queries: $\mathcal{B}$ maintains a list of tuples $< B_i, t_i, \mathsf{CT}_i, \mathsf{ID}_i >$, referred to as the $\widetilde{H}^{list}$. Given $B_i \in \{0,1\}^*$, $\mathcal{B}$ scans through the $\widetilde{H}^{list}$ to see if B_i appears in a tuple $< B_i, t_i, \mathsf{CT}_i, \mathsf{ID}_i >$. If it does, $\mathcal{B}$ responds with $H(B_i) = t_i$. Otherwise, $\mathcal{B}$ picks a random $t_i \in \{0,1\}^{\lceil \log n \rceil}$ and sets $\widetilde{H}(B_i) = t_i$. Additionally, $\mathcal{B}$ constructs CT_i under a message M' and ID_i, where M' is the first δ bits of B_i, and ρ' is the last θ bits of B_i, and the rest of B_i becomes an identity ID_i. $\mathcal{B}$ adds the new tuple $< B_i, t_i, \mathsf{CT}_i, \mathsf{ID}_i >$ to the $\widetilde{H}^{list}$ and responds with $\widetilde{H}(B_i)$.

Key queries: Given $\mathsf{ID}_i \in \mathcal{ID}$, $\mathcal{B}$ scans through the h^{list} to find a tuple $< \mathsf{ID}_i, h(\mathsf{ID}_i) >$. Let $h(\mathsf{ID}_i) = r_i$. If $\mathsf{ID}_{i^*} = \mathsf{ID}_i$, $\mathcal{B}$ aborts. (We refer to this event as abort1.) Otherwise, $\mathcal{B}$ selects $1/(x+r_i) \in \mathbb{Z}_{\mathrm{ord}_n g}$ among $\{1/(x+r_j)\}_{j=1}^{\widetilde{q}}$ as $\mathsf{sk}_{\mathsf{ID}_i}$. $\mathcal{B}$ gives $1/(x+r_i)$ to $\mathcal{A}$.

Decryption queries: Given a pair $(\mathsf{CT}_i, \mathsf{ID}_i)$, $\mathcal{B}$ scans through the $\widetilde{H}^{list}$ to find a tuple $< B_j, t_j, \mathsf{CT}_j, \mathsf{ID}_j >$ such that $\mathsf{ID}_i = \mathsf{ID}_j$ and $\mathsf{CT}_i = \mathsf{CT}_j$. If there exists such a tuple, $\mathcal{B}$ outputs the corresponding message M' that is the first δ bits of B_j. Otherwise, $\mathcal{B}$ outputs reject.

Challenge $\mathcal{A}$ outputs $M_0, M_1 \in \{0,1\}^\delta$ and ID^*. If $\mathsf{ID}_{i^*} \neq \mathsf{ID}^*$, $\mathcal{B}$ aborts. (We refer to this event as abort2.) Otherwise, $\mathcal{B}$ uses r^* as $h(\mathsf{ID}^*)$, selects a random $\rho^* \in \{0,1\}^\theta$, and sets $C_1^* = g^{(x+r^*)y} \in \mathbb{Z}_n$. Notice that $C_1 = g^{(x+r^*)y} = (g_1 g^{h(\mathsf{ID}^*)})^y$ under $s = y$. If $(M_0, \mathsf{ID}_{i^*}, \rho^*)$ or $(M_1, \mathsf{ID}_{i^*}, \rho^*)$ was queried to $\widetilde{H}$ oracle, then $\mathcal{B}$ aborts. (We refer to this event as abort3.) Next, $\mathcal{B}$ picks a random $R \in \{0,1\}^{\delta+\theta}$ and sets $C_2^* = R$. $\mathcal{B}$ gives $\mathsf{CT}^* = (C_1^*, C_2^*)$ to $\mathcal{A}$.

Query Phase 2 $\mathcal{A}$ issues more h, H, $\widetilde{H}$, private key, and decryption queries on the constraint that the private key query for ID^* and also the decryption query for $(\mathsf{CT}^*, \mathsf{ID}^*)$ are not allowed. $\mathcal{B}$ responds as in Query Phase 1.

Guess $\mathcal{A}$ outputs a guess $b \in \{0,1\}$. At this point, $\mathcal{B}$ picks a random $< A_i, w_i >$ from the H^{list} and outputs w_i as the solution to the given $\widetilde{q}$-TSDH problem.

Analysis. We easily see that the computation that $\mathcal{B}$ requires is dominated by at most $q_{\widetilde{H}}$ encryptions plus $\mathcal{A}$'s computation that needs to break the CCA security of the IBE scheme.

As observed in [24], there could be a troublesome decryption query in the case where $\mathcal{A}$ issues $(\mathsf{CT}_i = (C_{i,1}, C_{i,2}), \mathsf{ID}_i)$ as a decryption query without making the relevant $\widetilde{H}$ query in advance. That is, let $C_{i,1} = (g_1 g^{h(\mathsf{ID}_i)})^{s_i}$ and $C_{i,2} = H(g^{s_i}) \oplus (M_i || \rho_i)$ for some $s_i \in \{0,1\}^{\lceil \log n \rceil}$, $M_i \in \{0,1\}^\delta$ and $\rho_i \in \{0,1\}^\theta$. Obviously, the answer to such a decryption query is reject, since there is no relevant tuple $< B_i, t_i, \mathsf{CT}_i, \mathsf{ID}_i >$. After such a decryption query, $\mathcal{A}$ issues a tuple $(M_i, \mathsf{ID}_i, \rho_i)$ to $\widetilde{H}$ query. $\mathcal{B}$ then has to pick a random $\widetilde{s} \in \{0,1\}^{\lceil \log n \rceil}$ (if there does not exist such a tuple). If $C_{i,1} = (g_1 g^{h(\mathsf{ID}_i)})^{s_i} = (g_1 g^{h(\mathsf{ID}_i)})^{\widetilde{s}} \pmod n$, then $\mathcal{B}$ aborts. (We refer to this event as abort4.) This is because, in that case, $\mathcal{B}$ should have output the correctly decrypted message M_i instead of the previous output reject. Recall that s_i is chosen by $\mathcal{A}$ (since $\mathcal{A}$ did not issue $(M_i, \mathsf{ID}_i, \rho_i)$ to $\widetilde{H}$ query) and $\widetilde{s}$ is chosen by $\mathcal{B}$.

If $(g_1 g^{h(\mathsf{ID}_i)})^{s_i} = (g_1 g^{h(\mathsf{ID}_i)})^{\widetilde{s}} \pmod n$, Lemma 1 shows that $(x + h(\mathsf{ID}_i))s_i \equiv (x + h(\mathsf{ID}_i))\widetilde{s} \pmod{\mathrm{ord}_n g}$. We can assume that $\gcd(x + h(\mathsf{ID}_i), \mathrm{ord}_n g) = 1$[4] for two large primes p_1 and q_1 such that $\mathrm{ord}_n g = p_1 q_1$. Under the assumption, if $(g_1 g^{h(\mathsf{ID}_i)})^{s_i} = (g_1 g^{h(\mathsf{ID}_i)})^{\widetilde{s}} \pmod n$, then it implies that $\widetilde{s} \equiv s_i \pmod{\mathrm{ord}_n g}$.

Claim 3: The probability that $\mathcal{B}$ dost not abort is at least $\frac{1}{q_h}(1 - \frac{q_{\widetilde{H}}}{2^\theta})(1 - \frac{q_D}{\mathrm{ord}_n g})$.

Proof. Let $\overline{\mathsf{abort}}$ be the event that $\mathcal{B}$ does not abort during the simulation. For $\overline{\mathsf{abort}}$ to happen, the equality $\overline{\mathsf{abort}} = \overline{\mathsf{abort1}} \wedge \overline{\mathsf{abort2}} \wedge \overline{\mathsf{abort3}} \wedge \overline{\mathsf{abort4}}$ should hold. The events abort1 and abort2 are complementary. Thus, as in the proof of Theorem 1, we know that $\Pr[\overline{\mathsf{abort1}} \wedge \overline{\mathsf{abort2}}]$ is at least $1/q_h$.

The event abort3 occurs if either $(M_0, \mathsf{ID}_{i^*}, \rho^*)$ or $(M_1, \mathsf{ID}_{i^*}, \rho^*)$ was queried to $\widetilde{H}$ oracle by $\mathcal{A}$. This is when $\mathcal{B}$'s work to map $\widetilde{H}(M_0, \mathsf{ID}_{i^*}, \rho^*)$ or $\widetilde{H}(M_1, \mathsf{ID}_{i^*}, \rho^*)$ into the exponent $s = y$ is hindered by the $\mathcal{A}$'s previous query. Note that M_0, M_1 and ID_{i^*} are chosen by $\mathcal{A}$ and ρ^* is chosen by $\mathcal{B}$. abort3 then happens when $\mathcal{B}$ selects a random $\rho^* \in \{0,1\}^\theta$ such that either $(M_0, \mathsf{ID}_{i^*}, \rho^*)$ or $(M_1, \mathsf{ID}_{i^*}, \rho^*)$ becomes one of the input values issued by $\mathcal{A}$. For fixed values (M_0, ID_{i^*}) and (M_1, ID_{i^*}), abort3 occurs if the θ-bit random ρ^* (chosen by $\mathcal{B}$) is equal to a last θ-bit value $\star$ (chosen by $\mathcal{A}$) among $\{(M_0, \mathsf{ID}_{i^*}, \star)\}$ and $\{(M_1, \mathsf{ID}_{i^*}, \star)\}$. For one challenge query, the probability that abort3 occurs becomes at most $q_{\widetilde{H}}/2^\theta$, and thus $\Pr[\overline{\mathsf{abort3}}]$ becomes at least $(1 - \frac{q_{\widetilde{H}}}{2^\theta})$.

The event abort4 occurs if $\widetilde{s} \equiv s_i \pmod{\mathrm{ord}_n g}$, where s_i is chosen by $\mathcal{A}$ and $\widetilde{s}$ is chosen by $\mathcal{B}$. This is when $\mathcal{B}$'s work to correctly decrypt a ciphertext is hindered by the $\mathcal{B}$'s previous output reject. For one decryption query (without asking a relevant $\widetilde{H}$ oracle beforehand), abort4 happens if s_i (chosen by $\mathcal{A}$) is equal to $\widetilde{s}$ (chosen by $\mathcal{B}$) in $\mathbb{Z}_{\mathrm{ord}_n g}$ and thus the probability that abort4 occurs becomes at most $1/\mathrm{ord}_n g$. Since $\mathcal{A}$ can make q_D decryption queries, the probability that abort4 occurs becomes at most $q_D/\mathrm{ord}_n g$, and thus $\Pr[\overline{\mathsf{abort4}}]$ becomes at least $(1 - \frac{q_D}{\mathrm{ord}_n g})$.

The events $\overline{\mathsf{abort1}} \wedge \overline{\mathsf{abort2}}$ and $\overline{\mathsf{abort3}}$ and $\overline{\mathsf{abort4}}$ are relatively independent. Hence, the probability

[4] See Remark 4 in Section 3.1.

Table 1: Efficiency comparison between the pairing-based IBE schemes and ours

	PP	CT	$\mathsf{sk}_{\mathsf{ID}}$	KeyGen	Encryption	Decryption
BF	$2\mathbb{G}_2$	$1\mathbb{G}_2, 1\mathtt{h}$	$1\mathbb{G}_1$	$1\mathtt{H}_1, 1\mathtt{E}_1$	$1\mathtt{H}_1, 1\mathtt{E}_2^{\mathtt{f}}, 1\mathtt{E}_{\mathtt{T}}, 1\mathtt{P}$	$1\mathtt{E}_2^{\mathtt{f}}, 1\mathtt{P}$
BB_1	$3\mathbb{G}_1, 1\mathbb{G}_{\mathtt{T}}$	$2\mathbb{G}_1, 1\mathtt{h}$	$2\mathbb{G}_2$	$2\mathtt{E}_2^{\mathtt{f}}$	$3\mathtt{E}_1^{\mathtt{f}}, 1\mathtt{E}_{\mathtt{T}}^{\mathtt{f}}$	$3\mathtt{E}_1^{\mathtt{f}}, 2\mathtt{P}^{\sharp}$
SK	$2\mathbb{G}_1, 1\mathbb{G}_{\mathtt{T}}$	$1\mathbb{G}_1, 1\mathtt{h}$	$1\mathbb{G}_2$	$1\mathtt{E}_2^{\mathtt{f}}$	$2\mathtt{E}_1^{\mathtt{f}}, 1\mathtt{E}_{\mathtt{T}}^{\mathtt{f}}$	$2\mathtt{E}_1^{\mathtt{f}}, 1\mathtt{P}$
Ours	n, $2\mathbb{Z}_n$	$1\mathbb{Z}_n, 1\mathtt{h}$	$1\mathbb{Z}_{\mathrm{ord}_{ng}}$	$1\mathtt{I}_{\mathrm{ord}_{ng}}$	$2\mathtt{E}_n^{\mathtt{f}}$, $1\widehat{\mathtt{E}}_n^{\mathtt{f}}$	$1\mathtt{E}_n$, $1\mathtt{E}_n^{\mathtt{f}}$, $1\widehat{\mathtt{E}}_n^{\mathtt{f}}$

$\mathtt{h}$: output size of hash function; $\mathtt{H}_1$: map-to-point hash into $\mathbb{G}_1$; $\{\mathtt{E}_1, \mathtt{E}_2, \mathtt{E}_{\mathtt{T}}, \mathtt{E}_n\}$: a general exponentiation in $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbb{Z}_n\}$, respectively; $\widehat{\mathtt{E}}_n$; an exponentiation with a $(\log n + \ell)$-bit exponent for ℓ-bit hash output; "$\mathtt{f}$": means that fixed-based exponentiations can be performed; $\mathtt{I}_{\mathrm{ord}_{ng}}$: a modular inversion in $\mathbb{Z}_{\mathrm{ord}_{ng}}$; $\mathtt{P}$: pairing; $\sharp$: two parings can be optimized into about 1.2 pairing.

$\Pr[\overline{\mathsf{abort}}]$ is then at least $\frac{1}{q_h}(1-\frac{q_{\tilde{H}}}{2^\theta})(1-\frac{q_D}{\mathrm{ord}_{ng}})$. $\square$

Unless $\mathcal{B}$ does not abort, $\mathcal{B}$ is simulating a real attack environment for $\mathcal{A}$. Let $\mathcal{H}$ be the event that $\mathcal{A}$ issues g^y as an H query at some point, which implies that at the end of the simulation g^y (as one of values $\{A_i\}_{i=1}^{q_H}$) appears in the H^{list}. We show that $\Pr[\mathcal{H}] \geq 2\varepsilon$, by using the same proof strategy as in [9].

Claim 4: (from [9]) $\Pr[\mathcal{H}] \geq 2\varepsilon$.

As long as $\mathcal{B}$ does not abort, $\mathcal{B}$ can use the $\mathcal{A}$'s advantage to solve the $\widetilde{q}$-TSDH problem. With the probability at least 2ε from Claim 4, the correct answer g^y appears in some tuple on the H^{list}. At the end of simulation, $\mathcal{B}$ picks a random element from $\{A_i\}_{i=1}^{q_H}$ in the H^{list}, so that $\mathcal{B}$ produces the answer with probability at least $2\varepsilon/q_H$. This is done with the probability at least $\frac{1}{q_h}(1-\frac{q_{\tilde{H}}}{2^\theta})(1-\frac{q_D}{\mathrm{ord}_{ng}})$ from Claim 3, implying that $\mathcal{B}$ does not abort. Hence, by putting them all together, we can get the probability $\mathcal{B}$ produces the answer for the $\widetilde{q}$-TSDH problem at least $\frac{2\varepsilon}{q_H q_h}(1-\frac{q_{\tilde{H}}}{2^\theta})(1-\frac{q_D}{\mathrm{ord}_{ng}})$. This concludes the proof of Theorem 2. ■

3.5 Efficiency comparison to the previous practical IBE schemes

We compare our IBE scheme with the previous pairing-based IBE schemes such as BF [9], SK [34], and BB_1 [5], which are now in the process of standardization. For simplicity, we assume that all IBE schemes are CCA secure by applying $\mathsf{FO}_{\mathsf{ID}}$ to each of them, and ideally the efficiency comparison is not affected by the difficulty of solving a computational hardness problem on which the security of each IBE scheme is based. We consider Supersingular (SS) curves of embedding degree 2 over large prime fields for the pairing-based IBE schemes, because especially we want to compare the efficiency in case of the SS curves at the 80-bit security level that are believed to give the fastest pairing operation. In both encryption and decryption of our scheme, we consider one of exponentiations modulo n as being performed with $(\log n + \ell)$-bit exponent for ℓ-bit hash output. We also assume that any exponentiation is performed with fixed-base when a base is one of public parameters.

Table 1 presents the efficiency comparison between the pairing-based IBE schemes and ours, and Table 2 shows the representation sizes of group elements and estimated calculation timings of various operations. We equivalently consider SS-512 curves and 1024-bit modulus n as the same 80-bit security level, and SS-1024 curves and 2048-bit modulus n as the same 112-bit security level. The operation timings are estimated when running PBC and Integer libraries on Intel Core i5-4590 @ 3.30GHz. By putting the results of Table 1 and 2 all together, we have the efficiency comparison result shown in Table 3 with respect to actual overheads

Table 2: Representation sizes and estimated calculation timings for IBE

Curves / Integers	Sizes (bits)					Timings (ms: 10^{-3} seconds)							
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	$\mathbb{Z}_n$	$\mathbb{Z}_{\mathrm{ord}_n g}$	$\mathtt{E}_1$	$\mathtt{E}_2$	$\mathtt{H}_1$	$\mathtt{E}_\mathtt{T}$	$\mathtt{P}$	$\mathtt{I}_{\mathrm{ord}_n g}$	$\mathtt{E}_n$	$\widehat{\mathtt{E}}_n$
SS-512 / n-1024	512	512	1024	1024	1022	2.65	2.65	6.13	0.74	3.01	0.05	1.79	2.36
SS-1024 / n-2048	1024	1024	2048	2048	2046	41.33	41.33	2.16	5.26	65.88	0.05	12.44	13.41

Table 3: Overheads and estimated calculation timings at the 80- and 112-bit security levels

	SS-512 / n-1024						SS-1024 / n-2048					
	Overheads (bits)			Timings (ms)			Overheads (bits)			Timings (ms)		
	PP	$\mathsf{CT}^\flat$	$\mathsf{SK}_{\mathsf{ID}}$	KGen	Enc	Dec	PP	$\mathsf{CT}^\flat$	$\mathsf{SK}_{\mathsf{ID}}$	KGen	Enc	Dec
BF	1024	768	512	8.78	10.41	3.54	2048	1280	1024	43.49	81.56	74.14
BB_1	2560	1280	1024	1.06	1.73	$5.20^\sharp$	5120	2304	2048	16.53	25.85	$103.85^\sharp$
SK	2048	768	512	0.53	1.20	4.07	4096	1280	1024	8.26	17.58	82.41
Ours	3072	1280	1022	0.05	1.19	2.62	6144	2304	2046	0.05	7.66	17.61

$\flat$: $\mathtt{h} = 256$ (bits); $\sharp$: a ratio of two pairings is calculated as 1.2 pairing.

and estimated calculation times at the 80- and 112-bit security levels. Table 3 shows that ours gives a longer size of ciphertexts, but is more efficient than the other pairing-based schemes in terms of key generation, encryption, and decryption.[5] This computational advantage becomes remarkable when IBE schemes are implemented on the SS-1024 curves and 2048-bit modulus n, respectively. We can also expect that, if our IBE scheme can be constructed on a smaller size of subgroups of $\mathbb{Z}_n^*$ (mentioned in Remark 7 in Section 3.1), the computational advantage stands out much more noticeably at the 112-bit security level.

4 New Public-Key Signature Scheme

We present a new PKS scheme whose security relies on a new assumption, called $\widetilde{q}$-Trapdoor Subgroup Exponent Inversion (TSEI) assumption for $\widetilde{q}$ adversarial signature queries.

4.1 Construction

Setup(k): As in the IBE scheme, except for selecting a hash function. The setup algorithm picks one hash function $h : \{0,1\}^* \to \{0,1\}^\ell$, where $\ell < \log(\mathrm{ord}_n g)$. The algorithm outputs $\mathsf{PK} = (n, g, g_1, h)$ and $\mathsf{sk} = (x, \mathrm{ord}_n g)$.

Sign(m, sk): To sign a message $m \in \{0,1\}^*$, the signing algorithm does the following:

1. Compute $h(m) \in \{0,1\}^\ell$, where $\ell < \log(\mathrm{ord}_n g)$ and thus $h(m) \in \mathbb{Z}_{\mathrm{ord}_n g}$.
2. Check if $\gcd(x + h(m), \mathrm{ord}_n g) \neq 1$. If so, abort.

[5] If MNT curves are used, encryption of the pairing-based schemes could be slightly faster than ours, but decryption of the others would be much slower than ours.

3. Otherwise, compute σ such that $(x+h(m)) \cdot \sigma \equiv 1 \pmod{\mathrm{ord}_n g}$.

4. Output the signature $\sigma = 1/(x+h(m)) \in \mathbb{Z}_{\mathrm{ord}_n g}$.

Verify(PK, m, σ): To verify a signature σ on a message m, the verification algorithm does the following:

1. Compute $h(m) \in \{0,1\}^\ell$.

2. Output accept if $(g_1 g^{h(m)})^\sigma \stackrel{?}{=} g \pmod{n}$ holds. If not, output reject.

Correctness. As before, we use the fact that $g^{\mathrm{ord}_n g} = 1 \pmod{n}$ and $(x+h(m)) \cdot \sigma = 1 + k \cdot \mathrm{ord}_n g$ for some integer $k \in \mathbb{Z}$. Then the correctness of the verification algorithm can be verified as follows:

$$\begin{aligned}(g_1 g^{h(m)})^\sigma \pmod{n} &= g^{(x+h(m))\cdot\sigma} \pmod{n} = g^{1+k\cdot\mathrm{ord}_n g} \pmod{n} \\ &= g\left(g^{\mathrm{ord}_n g}\right)^k \pmod{n} = g \pmod{n}.\end{aligned}$$

Remark 8. The signing cost is one modular inversion and the verification cost is two (fixed-base) exponentiations. The signature size is one element in $\mathbb{Z}_{\mathrm{ord}_n g}$. As in the previous IBE scheme, the size of the exponent $h(m)\sigma$ in verification becomes $\ell + \log(\mathrm{ord}_n g)$ bits and $h(m)\sigma$ cannot be reduced modulo $\mathrm{ord}_n g$ without knowing $\mathrm{ord}_n g$. Thus, in case of using fixed-base exponentiations, the values of $g^j \pmod{n}$ should be prepared for $j = 1, \ldots, \ell + \log(\mathrm{ord}_n g)$.

Remark 9. As in IBE, the discrete logarithm $x \in \mathbb{Z}_{\mathrm{ord}_n g}$ should be carefully chosen to ensure that (1) the trapdoor subgroup DLP is believed to be hard in the subgroup of $\mathbb{Z}_n^*$ and (2) the small or large *signature* exponent attack is thwarted. Also, our PKS scheme can be constructed under a smaller size of subgroups (as mentioned in Section 3.1) if the $\widetilde{q}$-TSEI assumption holds in the smaller trapdoor subgroups of $\mathbb{Z}_n^*$.

Remark 10. The signature generation can be viewed as a function $F_x : \{0,1\}^\ell \to \mathbb{Z}_{\mathrm{ord}_n g}$ such that $F_x(w) = 1/(x+w)$, which is deterministic for the fixed $x \in \mathbb{Z}_{\mathrm{ord}_n g}$. F_x is injective: if $F_x(w_1) = F_x(w_2)$ for two $w_1, w_2 \in \{0,1\}^\ell$, then $1/(x+w_1) \equiv 1/(x+w_2)$ and thus $w_1 \equiv w_2 \pmod{\mathrm{ord}_n g}$. Since $w_1 < \mathrm{ord}_n g$ and $w_2 < \mathrm{ord}_n g$ (notice that $\ell < \log(\mathrm{ord}_n g)$), it follows that $w_1 = w_2$. Therefore, if h is a collision-resistant hash function, then the signing algorithm generates a (deterministic) signature $F_x(h(m))$ on an arbitrary length message $m \in \{0,1\}^*$. Using the Coron's Technique [16], we can modify the signing algorithm into a randomized one where a signature on m consists of two elements $\sigma = (1/(x+h(m||R)), R)$ for a random salt R. The variant makes the signature length longer, but allows for achieving a tight security reduction to the $\widetilde{q}$-TSEI problem.

4.2 Security

Theorem 3. *Let h be modeled as a random oracle and $q_S \le q_h < \widetilde{q}$. Suppose the $(t', \varepsilon', \widetilde{q})$-TSEI assumption holds in the trapdoor subgroup of $\mathbb{Z}_n^*$. Then our signature scheme is (t, ε, q_S)-strongly unforgeable against chosen message attacks, where*

$$\varepsilon \cdot \frac{1}{q_h}\left(1 - \frac{\widetilde{q}}{2^\ell}\right) \le \varepsilon', \qquad t \approx t'.$$

Here, q_h is the number of h queries and ℓ is the output length of h.

Proof. Suppose that there exists an adversary $\mathcal{A}$ which can break the strong unforgeability of our signature scheme. We show how to build an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve a $\widetilde{q}$-TSEI problem in the trapdoor

subgroup of $\mathbb{Z}_n^*$. On input $\left(n,\ g,\ g^x,\ \left\{1/(x+r_i), r_i\right\}_{i=1}^{\widetilde{q}}\right)$, $\mathcal{B}$ tries to output a new pair $\left(1/(x+r^*), r^*\right) \in \mathbb{Z}_{\mathrm{ord}_n g} \times \{0,1\}^\ell$, where $r^* \notin \{r_i\}_{i=1}^{\widetilde{q}}$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup $\mathcal{B}$ sets $g_1 = g^x$ and gives $\mathsf{PK} = (n, g, g_1, h)$ to $\mathcal{A}$.

Query Phase $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries as follows:

h queries: $\mathcal{B}$ picks a random i^* from $\{1, \ldots, q_h\}$. $\mathcal{B}$ maintains a list of tuples $< m_i, h(m_i) >$, referred to as the h^{list}. Given $m_i \in \{0,1\}^*$, $\mathcal{B}$ scans through the h^{list} to see if m_i appears in a tuple $< m_i, h(m_i) >$. If it does, $\mathcal{B}$ responds with $h(m_i)$. Otherwise, $\mathcal{B}$ uses an unused random $r_i \in \{0,1\}^\ell$ from $\{r_i\}_{i=1}^{\widetilde{q}}$ and sets $h(m_i) = r_i$. $\mathcal{B}$ adds the new $< m_i, h(m_i) >$ to the h^{list} and responds with $h(m_i)$. If $i = i^*$, then $\mathcal{B}$ selects a random $r^* \in \{0,1\}^\ell$. If $r^* \in \{r_i\}_{i=1}^{\widetilde{q}}$, then $\mathcal{B}$ aborts. (We refer to this event as $\mathsf{abort1}$) Otherwise, i.e., if $r^* \notin \{r_i\}_{i=1}^{\widetilde{q}}$, $\mathcal{B}$ sets $h(m_{i^*}) = r^*$.

Signature queries: Given $m_i \in \{0,1\}^*$, $\mathcal{B}$ scans through the h^{list} to find m_i in the h^{list}. If $i = i^*$, then $\mathcal{B}$ aborts. (We refer to this event as $\mathsf{abort2}$) Otherwise, i.e., if $i \neq i^*$, $\mathcal{B}$ responds with $1/(x+r_i)$ (from $\{1/(x+r_i)\}_{i=1}^{\widetilde{q}}$) as the signature on m_i, where $h(m_i) = r_i$.

Output $\mathcal{A}$ outputs a valid signature forgery (m^*, σ^*). Depending on whether or not m^* was queried beforehand, there are two possible cases:

[*Case 1.*] m^* is not queried during the signature queries. In that case, if $m^* \neq m_{i^*}$, then $\mathcal{B}$ aborts. (We refer to this event as $\mathsf{abort3}$) Otherwise, in that case, $h(m^*) = r^* \notin \{r_i\}_{i=1}^{\widetilde{q}}$, so that $\mathcal{B}$ can obtain $\sigma^* = 1/(x+r^*) \in \mathbb{Z}_{\mathrm{ord}_n g}$ from the following verification equation:

$$\begin{aligned}(g_1 g^{h(m^*)})^{\sigma^*} = g \pmod n &\Longleftrightarrow g^{(x+h(m^*))\cdot\sigma^*} = g \pmod n \\ &\Longleftrightarrow (x+r^*)\cdot\sigma^* \equiv 1 \pmod{\mathrm{ord}_n g} \ \text{(by Lemma 1)}.\end{aligned}$$

$\mathcal{B}$ can then obtain the pair $(1/(x+r^*), r^*)$ as the solution of the $\widetilde{q}$-TSEI problem.

[*Case 2.*] m^* is one of the queried messages. Say $m^* = m_t$ for some $t \in \{1, \ldots, q_S\}$ and the corresponding signature (that was generated by one of signature queries) is σ_t. Again, $\mathcal{B}$ refers to the h^{list} and finds the tuple $< m^*, h(m^*) >$ such that $h(m^*) = r_i$ for some $r_i \in \{r_i\}_{i=1}^{\widetilde{q}}$. There exists such a tuple, because $m^* = m_t$ was issued as a signature query and $h(m^*)$ was mapped to some r_i. In this case, the two signatures σ^* and σ_t on the same message $m^* = m_t$ should be different, i.e., $\sigma^* \neq \sigma_t$ in $\mathbb{Z}_{\mathrm{ord}_n g}$. However, from the verification equation, we know that $(g_1 g^{h(m^*)})^{\sigma^*} = g \pmod n$ and $(g_1 g^{h(m^*)})^{\sigma_t} = g \pmod n$. Thus, by Lemma 1, $(x+h(m^*))\cdot\sigma^* \equiv 1$ and $(x+h(m^*))\cdot\sigma_t \equiv 1 \pmod{\mathrm{ord}_n g}$. We can assume that $\gcd(x+h(m^*), \mathrm{ord}_n g) = 1$, since otherwise $\mathcal{B}$ cannot generate the previous signature σ_t. Under the assumption, the two equations show that $\sigma^* = \sigma_t$ in $\mathbb{Z}_{\mathrm{ord}_n g}$, which is the contradiction to $\sigma^* \neq \sigma_t$ in $\mathbb{Z}_{\mathrm{ord}_n g}$.

Analysis. We can easily see that the computational time of $\mathcal{B}$ is almost the same as that of $\mathcal{B}$. Next, to analyze the $\mathcal{B}$'s advantage, we prove the following claim.

Claim 5: The probability that $\mathcal{B}$ dost not abort is at least $\frac{1}{q_h}(1 - \frac{\widetilde{q}}{2^\ell})$.

Proof. Let $\overline{\mathsf{abort}}$ be the event that $\mathcal{B}$ does not abort. Obviously, for $\overline{\mathsf{abort}}$ to happen, the equality $\overline{\mathsf{abort}} = \overline{\mathsf{abort1}} \wedge \overline{\mathsf{abort2}} \wedge \overline{\mathsf{abort3}}$ should hold. The event $\mathsf{abort1}$ happens if the ℓ-bit string r^* chosen by $\mathcal{B}$ belongs to $\{r_i\}_{i=1}^{\widetilde{q}}$. The probability that $\mathsf{abort1}$ happens is at most $\widetilde{q}/2^\ell$, and thus $\Pr[\overline{\mathsf{abort1}}]$ becomes at least $(1 - \frac{\widetilde{q}}{2^\ell})$. Next, $\mathsf{abort2}$ and $\mathsf{abort3}$ are complementary. Since the selection of i^* is independent of $\mathcal{A}$' view, it follows that $\Pr[\overline{\mathsf{abort2}} \wedge \overline{\mathsf{abort3}}]$ becomes at least $\frac{1}{q_h}$. We know that the events $\overline{\mathsf{abort1}}$ and $\overline{\mathsf{abort2}} \wedge \overline{\mathsf{abort3}}$ are relatively independent, so that the probability $\Pr[\overline{\mathsf{abort}}]$ is at least $\frac{1}{q_h}(1 - \frac{\widetilde{q}}{2^\ell})$. $\square$

Next, we can see that as long as $\mathcal{B}$ does not abort in the simulation, $\mathcal{B}$ provides $\mathcal{A}$ with a perfect simulation whose distribution is identical to that in a real interaction with a signer. This is because (1) the

Table 4: Efficiency comparison between the practical PKS schemes and ours

	RSA-FDH	RSA-PSS	DSA	ECDSA	Ours
Public key	(n,e)	(n,e)	(p,q,g,g^x)	(E,q,g,g^x)	(n,g,g^x)
Secret key	d	d	x	x	$(x,\mathrm{ord}_n g)$
Signing cost	1 $\mathtt{E}_n$	1 $\mathtt{E}_n$	1 $\mathtt{E}_p$	1 $\mathtt{E}_E$	1 $\mathtt{I}_{\mathrm{ord}_n g}$
Verification cost	1 $\mathtt{E}_n^e$	1 $\mathtt{E}_n^e$	2 $\mathtt{E}_p$	2 $\mathtt{E}_E$	1 $\mathtt{E}_n$ + 1 $\widehat{\mathtt{E}}_n$
Signature length	$\mathbb{Z}_n$	$\mathbb{Z}_n$	2 $\mathbb{Z}_q$	2 $\mathbb{Z}_q$	$\mathbb{Z}_{\mathrm{ord}_n g}$
Signing algorithm	Deterministic	Randomized	Randomized	Randomized	Deterministic

In DSA, q is a prime such that q divides $p-1$, and g is a generator of an order-q subgroup of $\mathbb{Z}_p^*$; In ECDSA, E is an elliptic-curve group based on a prime (or other fields), and g is a generator of an order-q subgroup of E; $\mathtt{E}_n$: a general modular exponentiation in $\mathbb{Z}_n$; $\mathtt{E}_p$: a modular exponentiation with a $\log q$-bit exponent in $\mathbb{Z}_p$; $\mathtt{E}_E$: a general exponentiation (i.e., point multiplication) over an elliptic-curve group E; $\widehat{\mathtt{E}}_n$: a modular exponentiation with a $(\log(\mathrm{ord}_n g)+\ell)$-bit exponent in $\mathbb{Z}_n$, where ℓ is the output bitlength of hash function; $\mathtt{E}_n^e$: a modular exponentiation with a small exponent (e.g., $e=65537$) in $\mathbb{Z}_n$; $\mathtt{I}_{\mathrm{ord}_n g}$: a modular inversion in $\mathbb{Z}_{\mathrm{ord}_n g}$.

Table 5: Representation sizes and estimated calculation timings for PKS

Curves / Integers	Sizes (bits)			Timings (ms: 10^{-3} seconds)					
	$\mathbb{Z}_n$	$\mathbb{Z}_q$	$\mathbb{Z}_{\mathrm{ord}_n g}$	$\mathtt{E}_E$	$\mathtt{E}_p$	$\mathtt{I}_{\mathrm{ord}_n g}$	$\mathtt{E}_n$	$\widehat{\mathtt{E}}_n$	$\mathtt{E}_n^e$
EC-224 / n-2048, p-2048	2048	224	2046	1.03	1.34	0.05	12.44	13.41	0.58

simulation of h oracles is obviously perfect as the output values are chosen at random among given values $\{r_i\}_{i=1}^{\widetilde{q}}$ and a randomly chosen $r^* \notin \{r_i\}_{i=1}^{\widetilde{q}}$, and (2) the simulation of signature oracles is also perfect as each signature on a message is generated with given $\{1/(x+r_i)\}_{i=1}^{\widetilde{q}}$.

It follows that as long as $\mathcal{B}$ does not abort in the simulation, $\mathcal{B}$ can use $\mathcal{A}$'s advantage ε to break the strong unforgeability of our signature scheme. From Claim 5, $\mathcal{B}$'s advantage ε' is then given as $\varepsilon \cdot 1/q_h \cdot (1-\widetilde{q}/2^\ell) \leq \varepsilon'$, as required. This concludes the proof of Theorem 3. ■

Table 6: Overheads and estimated calculation timings for PKS at the 112-bit security level

	RSA-FDH	RSA-PSS	DSA	ECDSA	Ours
Secret key (bits)	2048	2048	224	224	4092
Signing cost (ms)	12.44 (3.11$^\flat$)	12.44 (3.11$^\flat$)	1.34 (0.27†)	1.03 (0.21†)	0.05
Verification cost (ms)	0.58$^\sharp$	0.58$^\sharp$	2.68	2.06	25.85
Signature length (bits)	2048	2048	448	448	2046

$\flat$: CRT is used; $\dagger$: fixed-base exponentiation is performed; $\sharp$: $e=65537$ is raised.

4.3 Efficiency comparison to the previous practical PKS schemes

In this section we compare our signature scheme with the previous practical PKS schemes such as RSA-{FDH, PSS}, DSA, and ECDSA, which are now wildly used in practice. Table 4 presents the efficiency comparison between the previous schemes and ours. Table 5 shows the representation sizes and estimated calculation timings at the 112-bit security level when we consider a NIST-recommended elliptic-curve over a 224-bit prime field as well as 2048-bit modulus n and p. The estimated timings are calculated when running Elliptic-Curve and Integer libraries on Intel Core i5-4590 @ 3.30GHz. Table 6 shows, by putting the results of Table 4 and 5 together, that our scheme provides the *fastest* signing, compared to the other practical schemes: about 62.2 (=3.11/0.05) times faster than RSA-{FDH, PSS} that employ the Chinese remainder theorem (CRT), and about 4.2 (=0.21/0.05) times faster than ECDSA that uses fixed-base exponentiation.

On the other hand, our scheme is less efficient than the others in terms of verification cost. Our verification is about 44.5 (=25.85/0.58) times slower than RSA-{FDH, PSS}, and about 12.5 (=25.85/2.06) times slower than ECDSA. Regarding the signature size, ours is almost the same as RSA-{FDH, PSS}, but about 4.5 (=2046/448) times longer than DSA and ECDSA. For our scheme, however, there is a chance to improve the efficiency if our signature scheme is constructed on a smaller size of subgroups of $\mathbb{Z}_n^*$. An interesting case is when we take an order-p_2q_2 subgroup of $\mathbb{Z}_n^*$ where $n = pq = (2p_1p_2+1)(2q_1q_2+1)$ (as mentioned in Remark 7 in Section 3.1). If n is 2048 bits and $\mathrm{ord}_n g = p_2q_2$ is about 448 bits, the signature length of ours is the same as that of DSA and ECDSA, and apparently verification will be faster than our current construction. Our ongoing work is to investigate the difficulty of solving the $\widetilde{q}$-TSEI problem in the smaller size of trapdoor subgroups, and compare the efficiency with the others when implementing our scheme over the smaller subgroups.

5 Analysis in the Generic Group Model

To provide confidence in the $\widetilde{q}$-TSDH and $\widetilde{q}$-TSEI assumptions, we establish a lower bound on the computational complexity of the $\widetilde{q}$-TSDH and $\widetilde{q}$-TSEI problems for generic groups in the sense of [37, 6], respectively. We assume that factoring n is computationally infeasible, with n understood before.

In the generic-group model, elements in the trapdoor subgroup of $\mathbb{Z}_n^*$ appear to be encoded as unique random strings, so that no property other than equality can be directly tested by the adversary $\mathcal{A}$. $\mathcal{A}$ performs operations on group elements by interacting two oracles: one oracle for computing the group action, and one oracle for computing the exponentiation. For $\mathrm{ord}_n g(= p_1q_1)$ of the subgroup of $\mathbb{Z}_n^*$, the opaque encoding of the group elements is modeled as an injective function $\xi : \mathbb{Z}_{\mathrm{ord}_n g} \to \{0,1\}^{\lceil \log(\mathrm{ord}_n g)\rceil}$, which maps all $a \in \mathbb{Z}_{\mathrm{ord}_n g}$ to the string representation $\xi(a)$ that corresponds to $g^a \in \mathbb{Z}_n^*$ for a generator g of the subgroup. $\mathcal{A}$ communicates with the oracles using the ξ-representations of the group elements. Note that g corresponds to $\xi(1)$ modulo n.

The overall proof strategy of our analysis is similar to that of Shoup's analysis [37], but there exist several differences between them as follows: (1) due to computational hardness problems, $\mathcal{A}$ in [37] takes as input a list of encoding values, whereas $\mathcal{A}$ in ours takes as input a list of encoding values and *domain values*, and (2) depending on oracle queries made by $\mathcal{A}$, the domain values in [37] are maintained by a list of linear polynomials in indeterminates, whereas the domain values in ours are maintained by a list of rational fractions in indeterminates, and (3) to analyze the probability that $\mathcal{A}$ wins, [37] chooses random values corresponding to the indeterminates when $\mathcal{A}$ terminates, whereas ours picks the random values before $\mathcal{A}$ is given the input values. Especially, the reason of (3) in our analysis is because the $\widetilde{q}$-TSDH and $\widetilde{q}$-TSEI problems requires to give $\mathcal{A}$ *domain values* (as well as encoding values) as input, and thus at the initialization

step, the random values (later mapping to indeterminates) should be chosen and embedded into those domain values. Although the order is reversed, the philosophy in ours is the same as in Shoup's analysis.

5.1 The $\widetilde{q}$-TSDH Problem

Theorem 4. *Let $\mathcal{A}$ be an algorithm that solves the $\widetilde{q}$-TSDH problem in the generic group model. Assume that ξ is a random encoding function for the subgroup of $\mathbb{Z}_n^*$. If $\mathcal{A}$ makes a total of at most q_G queries to the oracles computing the group action and exponentiation in the subgroup of $\mathbb{Z}_n^*$, then*

$$\varepsilon = \Pr\Big[\mathcal{A}\Big(n, \xi(1), \xi(x), \xi((x+r^*)y), r^*, \big\{1/(x+r_i), r_i\big\}_{i=1}^{\widetilde{q}}\Big) = \xi(y)\Big] \leq \frac{3(q_G+4)^3}{p_1},$$

where (1) $1/(x+r_i)$ for $i = 1,\ldots,\widetilde{q}$ are in $\mathbb{Z}_{\mathrm{ord}_n g}$, (2) r^* and r_i for $i = 1,\ldots,\widetilde{q}$ are randomly chosen in $\{0,1\}^\ell$ for some $\ell < \log(\mathrm{ord}_n g)$, (3) $r^* \notin \{r_i\}_{i=1}^{\widetilde{q}}$, and (4) x and y are randomly chosen in $\mathbb{Z}_{\mathrm{ord}_n g}$.

Proof. Without loss of generality, assume that p_1 is larger than q_1 when $\mathrm{ord}_n g = p_1 q_1$. Throughout the game, we implicitly use the fact that $\mathbb{Z}_{\mathrm{ord}_n g}$ is isomorphic to $\mathbb{Z}_{p_1} \times \mathbb{Z}_{q_1}$ given by CRT. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$.

$\mathcal{B}$ first picks random $x, y \in \mathbb{Z}_{\mathrm{ord}_n g}$, $r_i \in \{0,1\}^\ell$ for $i = 1,\ldots,\widetilde{q}$, and $r^* \in \{0,1\}^\ell$ such that $r^* \notin \{r_i\}_{i=1}^{\widetilde{q}}$. $\mathcal{B}$ computes $\{1/(x+r_i)\}_{i=1}^{\widetilde{q}}$ in $\mathbb{Z}_{\mathrm{ord}_n g}$. $\mathcal{B}$ maintains a list of pairs, $L = \{(F_i, z_i, \xi_i) : i = 1,\ldots,\tau\}$, such that, at step τ in the game, $\tau \leq 3 + q_G$. Here, the $F_\star \in \mathbb{Z}_{p_1}[X,Y]$ are *rational fractions* in the indeterminates X, Y with coefficients in $\mathbb{Z}_{p_1}$. The $z_\star \in \mathbb{Z}_{q_1}$ are images (to $\mathbb{Z}_{q_1}$) of domain values in $\mathbb{Z}_{\mathrm{ord}_n g}$. The $\xi_\star \in \{0,1\}^{\lceil \log(\mathrm{ord}_n g)\rceil}$ are arbitrary distinct strings given to $\mathcal{A}$.

The list is initialized at step $\tau = 0$ by initializing $\tau \leftarrow 3$ and setting $F_1 = 1$, $F_2 = X$, $F_3 = (X + r^*_{p_1})Y$, where $r^*_{p_1} \equiv r^* \pmod{p_1}$, and $z_1 = 1$, $z_2 = x_{q_1}$, $z_3 = (x_{q_1} + r^*_{q_1}) y_{q_1}$, where $x_{q_1} \equiv x \pmod{q_1}$, $y_{q_1} \equiv y \pmod{q_1}$, and $r^*_{q_1} \equiv r^* \pmod{q_1}$. The corresponding strings ξ_1, ξ_2, and ξ_3 are set to arbitrary distinct strings in $\{0,1\}^{\lceil \log(\mathrm{ord}_n g)\rceil}$.

$\mathcal{B}$ starts the game by providing $\mathcal{A}$ with the encodings ξ_1, ξ_2, ξ_3, and r^*, $\{1/(x+r_i), r_i\}_{i=1}^{\widetilde{q}}$. $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries as follows:

Group action. Given a multiplication/division selection bit and two operands ξ_i and ξ_j with $1 \leq i, j \leq \tau$, compute $F_\tau \leftarrow F_i \pm F_j \in \mathbb{Z}_{p_1}[X,Y]$ and $z_\tau \leftarrow z_i \pm z_j \in \mathbb{Z}_{q_1}$ depending on whether a multiplication or a division is requested. If $F_\tau = F_l$ and $z_\tau = z_l$ for some $l < \tau$, set $\xi_\tau \leftarrow \xi_l$; otherwise, set ξ_τ to a random string in $\{0,1\}^{\lceil \log(\mathrm{ord}_n g)\rceil}$ distinct from $\xi_1,\ldots,\xi_{\tau-1}$. Add $(F_\tau, z_\tau, \xi_\tau)$ to the list L and give ξ_τ to $\mathcal{A}$, then increment τ by one.

Exponentiation. Given two operands ξ_i and $(1/(x+r_j), r_j)$ with $1 \leq i < \tau$ and $1 \leq j \leq \widetilde{q}$, compute $F_\tau \leftarrow F_i \cdot 1/(X + r_{j,p_1}) \in \mathbb{Z}_{p_1}[X,Y]$ and $z_\tau \leftarrow z_i \cdot 1/(x_{q_1} + r_{j,q_1}) \in \mathbb{Z}_{q_1}$, where $r_{j,p_1} \equiv r_j \pmod{p_1}$ and $r_{j,q_1} \equiv r_j \pmod{q_1}$. If $F_\tau = F_l$ and $z_\tau = z_l$ for some $l < \tau$, set $\xi_\tau \leftarrow \xi_l$; otherwise, set ξ_τ to a string in $\{0,1\}^{\lceil \log(\mathrm{ord}_n g)\rceil}$ distinct from $\xi_1,\ldots,\xi_{\tau-1}$. Add $(F_\tau, z_\tau, \xi_\tau)$ to the list L and give ξ_τ to $\mathcal{A}$, then increment τ by one.

After at most q_G queries, $\mathcal{A}$ terminates and returns a value ξ_ℓ and $1 \leq \ell \leq \tau$. Let F_ℓ be the corresponding function in the list L. In order to exhibit the correctness of $\mathcal{A}$'s answer within the simulation framework, $\mathcal{B}$ computes the rational fraction $F_\star = (X + r^*_{p_1})F_\ell \in \mathbb{Z}_{p_1}[X,Y]$. Notice that if $\mathcal{A}$'s answer is correct for particular exponents regarding indeterminates X and Y, we must have the following equality:

$$F_\star = (X + r^*_{p_1})F_\ell = F_3. \tag{2}$$

We know that the rational fraction F_ℓ has the following form:

$$F_\ell = \frac{\{(X + r^*_{p_1})Y\}^a (\sum_{i=1}^{q_G - 1} b_i X^i) + \sum_{i=1}^{q_G} c_i X^i}{\Pi_{i=1}^{\widetilde{q}} (X + r_{i,p_1})^{d_i}}, \tag{3}$$

where $a = 0$ or 1, the coefficients b_i ($i = 1, \ldots, q_G - 1$) and c_i ($i = 1, \ldots, q_G$) are in $\mathbb{Z}_{p_1}$, and $\sum_{i=1}^{\widetilde{q}} d_i \leq q_G$. Then, the equation (2) can be represented by the equation $\widetilde{F}_\star = 0$, where $\widetilde{F}_\star \in \mathbb{Z}_{p_1}[X,Y]$ is a linear polynomial of at most $q_G + 2$ degrees with respect to the indeterminates X and Y.

At this point $\mathcal{B}$ uses the random exponents x_{p_1} and y_{p_1} corresponding to indeterminates X and Y in $\mathbb{Z}_{p_1}$, where $x_{p1} \equiv x \pmod{p_1}$, $y_{p_1} \equiv y \pmod{p_1}$, and evaluates all the polynomials under the assignment. If the assignment causes two non-identical polynomials within the list L to assume the same value, then the simulation provided by $\mathcal{B}$ to $\mathcal{A}$ was flawed since it presented as distinct two group elements that were in fact equal. If it causes the non-trivial equation (2) to be satisfied, then $\mathcal{A}$ has won the game. However, if no non-trivial equality emerged from the assignment, then the simulation provided by $\mathcal{B}$ is perfect and resulted in $\mathcal{A}$'s failure to solve the instance. By the above argument, the success probability of $\mathcal{A}$ in the generic model is bounded by the probability that at least one equality among the following collections is satisfied, for random assigned values x_{p_1}, y_{p_1} in $\mathbb{Z}_{p_1}$ corresponding to indeterminates X and Y:

1. $F_i(x_{p_1}, y_{p_1}) - F_j(x_{p_1}, y_{p_1}) = 0$ in $\mathbb{Z}_{p_1}$, yet $F_i \neq F_j$ for some $i, j \in \{1, \ldots, \tau\}$,
2. $(x_{p_1} + r^*_{p_1}) F_\ell(x_{p_1}, y_{p_1}) = F_3(x_{p_1}, y_{p_1})$ in $\mathbb{Z}_{p_1}$.

We notice that each function F_j for $1 \leq j \leq \tau$ has the form of the equation (3). Then, each non-trivial functions $F_i - F_j$ for fixed i and j can be represented as a polynomial of degree at most $2q_G + 1$ in the indeterminate X and Y, so that it vanishes for random assignment of the indeterminates x_{p_1}, y_{p_1} in $\mathbb{Z}_{p_1}$ with probability at most $(2q_G + 1)/p_1$. In the second case, since the equation (2) can be represented as a polynomial of degree at most $q_G + 2$ in the indeterminate X, it vanishes for random assignment of the indeterminates x_{p_1}, y_{p_1} in $\mathbb{Z}_{p_1}$ with probability at most $(q_G + 2)/p_1$. Summing over all valid pairs in the above cases, we deduce that $\mathcal{A}$ wins the game with probability

$$\varepsilon \leq \left(\binom{\tau}{2} \frac{2q_G + 1}{p_1} + \frac{q_G + 2}{p_1} \right).$$

Since $\tau \leq 3 + q_G$, we have $\varepsilon \leq 3(q_G + 4)^3/p_1$, as required. ■

5.2 The q-TSEI Problem

Theorem 5. *Let $\mathcal{A}$ be an algorithm that solves the $\widetilde{q}$-TSEI problem in the generic group model. Assume that ξ is a random encoding function for the subgroup of $\mathbb{Z}_n^*$. If $\mathcal{A}$ makes a total of at most q_G queries to the oracles computing the group action and exponentiation in the subgroup of $\mathbb{Z}_n^*$, then*

$$\varepsilon = \Pr\left[\mathcal{A}\Big(n, \xi(1), \xi(x), \big\{1/(x + r_i), r_i\big\}_{i=1}^{\widetilde{q}}\Big) = \big(1/(x + r^*), r^*\big) \right] \leq \frac{3(q_G + 3)^3}{p_1},$$

where (1) $1/(x + r_i)$ for $i = 1, \ldots, \widetilde{q}$ are in $\mathbb{Z}_{\mathrm{ord}_n g}$, (2) r_i for $i = 1, \ldots, \widetilde{q}$ are randomly chosen in $\{0,1\}^\ell$ for some $\ell < \log(\mathrm{ord}_n g)$, and (3) x is randomly chosen in $\mathbb{Z}_{\mathrm{ord}_n g}$.

Proof. As before, assume that p_1 is larger than q_1 when $\mathrm{ord}_n g = p_1 q_1$, and we use the fact that $\mathbb{Z}_{\mathrm{ord}_n g}$ is isomorphic to $\mathbb{Z}_{p_1} \times \mathbb{Z}_{q_1}$. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$.

$\mathcal{B}$ first picks random $x \in \mathbb{Z}_{\mathrm{ord}_n g}$ and $r_i \in \{0,1\}^\ell$ for $i = 1, \ldots, \widetilde{q}$. $\mathcal{B}$ computes $\{1/(x + r_i)\}_{i=1}^{\widetilde{q}}$ in $\mathbb{Z}_{\mathrm{ord}_n g}$. $\mathcal{B}$ maintains a list of pairs, $L = \{(F_i, z_i, \xi_i) : i = 1, \ldots, \tau\}$, such that, at step τ in the game, $\tau \leq 2 + q_G$. Here, the $F_\star \in \mathbb{Z}_{p_1}[X]$ are *rational fractions* in the indeterminate X with coefficients in $\mathbb{Z}_{p_1}$. The $z_\star \in \mathbb{Z}_{q_1}$ are images (to $\mathbb{Z}_{q_1}$) of domain values in $\mathbb{Z}_{\mathrm{ord}_n g}$. The $\xi_\star \in \{0,1\}^{\lceil \log(\mathrm{ord}_n g) \rceil}$ are arbitrary distinct strings given out to $\mathcal{A}$.

The list is initialized at step $\tau = 0$ by initializing $\tau \leftarrow 2$ and setting $F_1 = 1$ and $F_2 = X$, and $z_1 = 1$ and $z_2 = x_{q_1}$, where $x_{q_1} \equiv x \pmod{q_1}$. The corresponding strings ξ_1 and ξ_2 are set to arbitrary distinct strings in $\{0,1\}^{\lceil \log(\mathrm{ord}_n g) \rceil}$.

$\mathcal{B}$ starts the game by providing $\mathcal{A}$ with the encodings ξ_1, ξ_2, and $\{1/(x+r_i), r_i\}_{i=1}^{\widetilde{q}}$. $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries as follows:

Group action. Given a multiplication/division selection bit and two operands ξ_i and ξ_j with $1 \le i, j \le \tau$, compute $F_\tau \leftarrow F_i \pm F_j \in \mathbb{Z}_{p_1}[X]$ and $z_\tau \leftarrow z_i \pm z_j \in \mathbb{Z}_{q_1}$ depending on whether a multiplication or a division is requested. If $F_\tau = F_l$ and $z_\tau = z_l$ for some $l < \tau$, set $\xi_\tau \leftarrow \xi_l$; otherwise, set ξ_τ to a random string in $\{0,1\}^{\lceil \log(\mathrm{ord}_n g) \rceil}$ distinct from $\xi_1, \ldots, \xi_{\tau-1}$. Add $(F_\tau, z_\tau, \xi_\tau)$ to the list L and give ξ_τ to $\mathcal{A}$, then increment τ by one.

Exponentiation. Given two operands ξ_i and $(1/(x+r_j), r_j)$ with $1 \le i < \tau$ and $1 \le j \le \widetilde{q}$, compute $F_\tau \leftarrow F_i \cdot 1/(X + r_{j,p_1}) \in \mathbb{Z}_{p_1}[X]$ and $z_\tau \leftarrow z_i \cdot 1/(x_{q_1} + r_{j,q_1}) \in \mathbb{Z}_{q_1}$, where $r_{j,p_1} \equiv r_j \pmod{p_1}$ and $r_{j,q_1} \equiv r_j \pmod{q_1}$. If $F_\tau = F_l$ and $z_\tau = z_l$ for some $l < \tau$, set $\xi_\tau \leftarrow \xi_l$; otherwise, set ξ_τ to a string in $\{0,1\}^{\lceil \log(\mathrm{ord}_n g) \rceil}$ distinct from $\xi_1, \ldots, \xi_{\tau-1}$. Add $(F_\tau, z_\tau, \xi_\tau)$ to the list L and give ξ_τ to $\mathcal{A}$, then increment τ by one.

After at most q_G queries, $\mathcal{A}$ terminates and returns a pair $(c^*, r^*) \in \mathbb{Z}_{\mathrm{ord}_n g} \times \{0,1\}^\ell$. In order to exhibit the correctness of $\mathcal{A}$'s answer within the simulation framework, $\mathcal{B}$ computes the rational fraction $F_\star = (X + r^*_{p_1}) c^*_{p_1}$, where $r^*_{p_1} \equiv r^* \pmod{p_1}$ and $c^*_{p_1} \equiv c^* \pmod{p_1}$. Notice that if $\mathcal{A}$'s answer is correct for particular exponents regarding indeterminate X, we must have the following equality:

$$F_\star = (X + r^*_{p_1}) c^*_{p_1} = 1. \tag{4}$$

We know that each of a rational fraction F_i for $1 \le i \le \tau$ has the following form:

$$F_i = \frac{\sum_{i=1}^{q_G} c_i X^i}{\prod_{i=1}^{\widetilde{q}} (X + r_{i,p_1})^{d_i}}, \tag{5}$$

where the coefficients c_i for $i = 1, \ldots, q_G$ are in $\mathbb{Z}_{p_1}$, $r_{i,p_1} \equiv r_i \pmod{p_1}$ for $i = 1, \ldots, q_G$, and $\sum_{i=1}^{\widetilde{q}} d_i \le q_G$.

At this point $\mathcal{B}$ use the random exponent x_{p_1} corresponding to indeterminate X in $\mathbb{Z}_{p_1}$, where $x_{p_1} \equiv x \pmod{p_1}$, and evaluates all the polynomials under the assignment. By the similar argument as in the proof of Theorem 4, the success probability of $\mathcal{A}$ in the generic model is bounded by the probability that at least one equality among the following collections is satisfied, for random assigned value x_{p_1} in $\mathbb{Z}_{p_1}$ corresponding to indeterminate X:

1. $F_i(x_{p_1}) - F_j(x_{p_1}) = 0$ in $\mathbb{Z}_{p_1}$, yet $F_i \neq F_j$ for some $i, j \in \{1, \ldots, \tau\}$,
2. $(x_{p_1} + r^*_{p_1}) c^*_{p_1} = 1$ in $\mathbb{Z}_{p_1}$.

Each function F_i for $1 \le i \le \tau$ has the form of the equation (5). Then, each non-trivial functions $F_i - F_j$ for fixed i and j can be represented as a polynomial of degree at most $2q_G$ in the indeterminate X, so that it vanishes for random assignment of the indeterminate x in $\mathbb{Z}_{p_1}$ with probability at most $2q_G/p_1$. In the second case, the equation (4) can be represented as a polynomial of degree at most 1 in the indeterminate X, so that it vanishes for random assignment of the indeterminate x in $\mathbb{Z}_{p_1}$ with probability at most $1/p_1$. Summing over all valid pairs in the above cases, we deduce that $\mathcal{A}$ wins the game with probability

$$\varepsilon \le \left(\binom{\tau}{2} \frac{2q_G}{p_1} + \frac{1}{p_1} \right).$$

Since $\tau \le 2 + q_G$, we have $\varepsilon \le 3(q_G + 3)^3/p_1$, as required. ■

References

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 2010.

[2] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2011.

[3] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM-CCS*, pages 62–73. ACM, 1993.

[4] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and rabin. In Ueli M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[5] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[7] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[9] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[10] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. *IACR Cryptology ePrint Archive*, 2007:177, 2007.

[11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[12] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006.

[13] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT*, pages 523–552, 2010.

[14] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO*, volume 8043 of *Lecture Notes in Computer Science*, pages 435–460. Springer, 2013.

[15] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[16] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000.

[17] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[18] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM, 2008.

[19] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

[20] Jens Groth. Cryptography in subgroups of z_n. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2005.

[21] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In Jonathan Katz, editor, *PKC*, volume 9020 of *Lecture Notes in Computer Science*, pages 799–822. Springer, 2015.

[22] Susan Hohenberger. The cryptographic impact of groups with infeasible inversion. *Master's thesis, Massachusetts Institute of Technology, Supervised by: Ronald L. Rivest*, 2003.

[23] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.

[24] Takashi Kitagawa, Peng Yang, Goichiro Hanaoka, Rui Zhang, Hajime Watanabe, Kanta Matsuura, and Hideki Imai. Generic transforms to acquire cca-security for identity based encryption: The cases of fopkc and REACT. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *Lecture Notes in Computer Science*. Springer, 2006.

[25] Hyung Tae Lee, Jung Hee Cheon, and Jin Hong. Accelerating id-based encryption based on trapdoor dl using pre-computation. *IACR Cryptology ePrint Archive*, 2011:187, 2011.

[26] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, 2010.

[27] Wenbo Mao and Chae Hoon Lim. Cryptanalysis in prime order subgroups of Z^*_n. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 214–226. Springer, 1998.

[28] Ueli M. Maurer and Yacov Yacobi. Non-interactive public-key cryptography. In Donald W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 498–507. Springer, 1991.

[29] Chandrashekhar Meshram. An efficient id-based cryptographic encryption based on discrete logarithm problem and integer factorization problem. *Inf. Process. Lett.*, 115(2):351–358, 2015.

[30] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2010.

[31] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptography*, 52(2):219–241, 2009.

[32] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2005.

[33] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[34] Ryuichi Sakai and Masao Kasahara. Id based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54, 2003.

[35] Amitabh Saxena and Ben Soh. A cryptographic primitive based on hidden-order groups. *J. Mathematical Cryptology*, 3(2):89–132, 2009.

[36] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[37] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[38] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[39] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

A Upper and lower bounds for bitlength of a modular inverse

Assume that $\gcd(x+h(\mathsf{ID}), \mathrm{ord}_n g) = 1$ and we find an inverse $\mathsf{sk}_{\mathsf{ID}} \in \mathbb{Z}_{\mathrm{ord}_n g}$ such that $(x+h(\mathsf{ID})) \cdot \mathsf{sk}_{\mathsf{ID}} \equiv 1 \pmod{\mathrm{ord}_n g}$. It is easy to show that $\log q_1 \leq \log(\mathsf{sk}_{\mathsf{ID}})$ as the lower bound, where q_1 is the quotient in the division of $\mathrm{ord}_n g/(x+h(\mathsf{ID}))$. This is because the equation $(x+h(\mathsf{ID})) \cdot \mathsf{sk}_{\mathsf{ID}} \equiv 1 \pmod{\mathrm{ord}_n g}$ means that $(x+h(\mathsf{ID})) \cdot \mathsf{sk}_{\mathsf{ID}} = 1 + k \cdot \mathrm{ord}_n g$ for some positive integer k. Since $x+h(\mathsf{ID})$ and $\mathsf{sk}_{\mathsf{ID}}$ are all less than $\mathrm{ord}_n g$, $\mathsf{sk}_{\mathsf{ID}}$ should be larger than the quotient q_1. This means that $\log q_1 < \log(\mathsf{sk}_{\mathsf{ID}})$ in terms of the bitlength.

Next, we show that $\log(\mathsf{sk}_{\mathsf{ID}}) \leq \log(\mathrm{ord}_n g - q_1)$ as the upper bound. For simplicity, let $a = \mathrm{ord}_n g$ and $b = x + h(\mathsf{ID})$. Using the extended Euclidean algorithm, we can express the process of finding the inverse as follows:

$$\gcd(x + h(\mathsf{ID}), \mathrm{ord}_n g) = s_k a + t_k b,$$

where s_k and t_k are the k-th terms of the sequences defined recursively by $s_0 = 1$, $t_0 = 0$, $s_1 = 1$, $t_1 = 1$, and $s_j = s_{j-2} - q_{j-1}s_{j-1}$, $t_j = t_{j-2} - q_{j-1}t_{j-1}$ for $j = 2, 3, \ldots, k$, where the q_j are the quotients in the divisions of the Euclidean algorithm when it is used to find $\gcd(x + h(\mathsf{ID}), \mathrm{ord}_n g)$. In this case, we has only to consider the sequence values t_j for $j = 2, \ldots, k$, because t_j can be the inverse in the end when we take 'mod $\mathrm{ord}_n g$' operation in the final $(k-1)$-th equation. When k is the last index, we see that $r_k = 0$ and $r_{k-1} = 1$ in the process of the extended Euclidean algorithm and then the inverse comes from t_{k-1}.

A simple calculation shows that t_j for an even number $j(\geq 2)$ is always negative and decreasing such as $t_2 > t_4 > \cdots$ (until k (or $k-1$) appears), whereas t_j for an odd number $j(\geq 3)$ is always positive and increasing such as $t_3 < t_5 < \cdots$ (until k (or $k-1$) appears). We first show that $|t_j| < \mathrm{ord}_n g$ for all even number j even if t_j for an even number j decreases, and $t_j < \mathrm{ord}_n g$ for all odd number j even if t_j for an odd number j increases.

Claim 6: $|t_j| < \mathrm{ord}_n g$ *for all even numbers* j*, and* $t_j < \mathrm{ord}_n g$ *for all odd numbers* j*.*

Proof. The value $a(= \mathrm{ord}_n g)$ can be expressed by $a = |t_j| r_{j-1} + |t_{j-1}| r_j$ for $j = 2, 3 \ldots, k$, where r_j and r_{j-1} are the remainders of the Euclidean algorithm when it is used to find $\gcd(a, b)$. In any case, all the remainders are larger than or equal to 0, and not both 0 at the same time. Thus, Claim 6 holds. □

If k terminates as an even number, t_{k-1} can be the inverse. As $t_{k-1} > 0$ and $t_{k-1} < \mathrm{ord}_n g$ from Claim 6, we know that $\mathsf{sk}_{\mathsf{ID}} = t_{k-1}$ in $\mathbb{Z}_{\mathrm{ord}_n g}$. Similarly, if k terminates as an odd number, t_{k-1} can be the inverse. As $t_{k-1} < 0$ and $|t_{k-1}| < \mathrm{ord}_n g$ from Claim 6, we know that $\mathsf{sk}_{\mathsf{ID}} = \mathrm{ord}_n g + t_k$ in $\mathbb{Z}_{\mathrm{ord}_n g}$. Next, we show that the increasing t_i for odd numbers $i = 3, 5, \ldots, k$ cannot exceed $\mathrm{ord}_n g - q_1$.

Claim 7: $\mathrm{ord}_n g - q_1 \geq t_{k-1}$ *when* k *terminates as an even number.*

Proof. When k terminates as an even number, as mentioned above, we know that $r_k = 0$ and $r_{k-1} = 1$ in the process of the extended Euclidean algorithm. Thus, $a = |t_k|$, and since k is an even number, $t_k < 0$ and thus $a = -t_k$. Also, we can check that for all even numbers j, $t_j = -q_1 - t_{j-1}q_{j-1} - \alpha$ for some $\alpha \geq 0$. Hence, when $j = k$, $a = q_1 + t_{k-1}q_{k-1} + \alpha$, and $a - q_1 = t_{k-1}q_{k-1} + \alpha$. Since $a = \mathrm{ord}_n g$ and $q_{k-1} \geq 1$, then we have $\mathrm{ord}_n g - q_1 \geq t_{k-1}$. □

Claim 6 and 7 tell us that, in $\mathbb{Z}_{\mathrm{ord}_n g}$, t_j for all odd numbers j cannot be larger than $\mathrm{ord}_n g - q_1$ even if they increases. Notice that the largest t_j for an even number j is $t_2 = -q_1$. Thus, we can have the bounds such that, for an odd number k, $t_k \leq \mathrm{ord}_n g - q_1$, and for an even number k, $\mathrm{ord}_n g + t_{k-1} \leq \mathrm{ord}_n g - q_1$. Since either t_k or $\mathrm{ord}_n g + t_{k-1}$ is the modular inverse $\mathsf{sk}_{\mathsf{ID}}$ in $\mathbb{Z}_{\mathrm{ord}_n g}$ when k is the last index in the process of the extended Euclidean algorithm, the two inequalities give the upper bound such that $\mathsf{sk}_{\mathsf{ID}} \leq \mathrm{ord}_n g - q_1$. Finally, in terms of the bitlength, it is obvious that $\log(\mathsf{sk}_{\mathsf{ID}}) \leq \log(\mathrm{ord}_n g - q_1)$, as required.

On the Security of Trapdoor Subgroups of $\mathbb{Z}_N^*$

Marc Joye

marc.joye@gmail.com

In a recent work, Park, Lee and Lee [1] present an identity-based encryption scheme and a companion signature scheme. Interestingly, their schemes work in RSA subgroups. The authors introduce two security assumptions to prove the security of their schemes. We show in this short note that, unfortunately, the assumptions do not hold.

Assumptions. Consider PPT algorithm $\mathsf{TRSAgen}$ which, on input a security parameter 1^κ, generates an RSA modulus $N = pq$ where $p = 2p_1 + 1$ and $q = 2q_1 + 1$, with p, q, p_1, q_1 prime, an element $g \in \mathbb{Z}_N^*$ of order p_1q_1, and a length ℓ for some $\ell < |p_1q_1|$. We write $(N, g, l) \leftarrow \mathsf{TRSAgen}(1^\kappa)$.

Assumption 1 (TSDH Assumption). *With the previous notations, the $\tilde{q}$-TSDH assumption asserts that*

$$\Pr\left[\mathcal{A}\left(N, g, g^x, g^{(x+r_0)y}, r_0, \left\{\left(1/(x+r_i), r_i\right)\right\}_{i=1}^{\tilde{q}}\right) = g^y\right]$$

is negligible for any PPT algorithm $\mathcal{A}$; the probabilities are taken over the experiment of running $(N, g, \ell) \leftarrow \mathsf{TRSAgen}(1^\kappa)$ and choosing at random $x, y \in \mathbb{Z}_{p_1q_1}$ and $r_0, r_1, \ldots, r_{\tilde{q}} \in \{0,1\}^\ell$.

Assumption 2 (TSEI Assumption). *With the previous notations, the $\tilde{q}$-TSEI assumption asserts that*

$$\Pr\left[\mathcal{A}\left(N, g, g^x, \left\{\left(1/(x+r_i), r_i\right)\right\}_{i=1}^{\tilde{q}}\right) = \left(1/(x+r^*), r^*\right) \mid r^* \notin \{r_1, \ldots, r_{\tilde{q}}\}\right]$$

is negligible for any PPT algorithm $\mathcal{A}$; the probabilities are taken over the experiment of running $(N, g, \ell) \leftarrow \mathsf{TRSAgen}(1^\kappa)$ and choosing at random $x \in \mathbb{Z}_{p_1q_1}$ and $r_1, \ldots, r_{\tilde{q}} \in \{0,1\}^\ell$.

Analysis. Define the set $\mathscr{S} = \left\{(\sigma_i, r_i)\right\}_{i=1}^{\tilde{q}}$ where $\sigma_i = 1/(x + r_i) \pmod{p_1q_1}$. From a pair of elements $(\sigma_1, r_1), (\sigma_2, r_2) \in \mathscr{S}$, it is easily checked that

$$\sigma_1\sigma_2(r_1 - r_2) + \sigma_1 - \sigma_2 \equiv 0 \pmod{p_1q_1} \ .$$

We can write $\Lambda := \sigma_1\sigma_2(r_1 - r_2) + \sigma_1 - \sigma_2$ as $\Lambda = 2^t\Lambda_0$ where $2^t \| \Lambda$ and Λ_0 is odd. It is worth observing that $\Lambda_0 \propto p_1q_1$. Next we choose a random element $h \in \mathbb{Z}_N^*$ with Jacobi symbol -1. It follows then that

$$\gcd(h^{\Lambda_0} \pm 1 \bmod N, N)$$

yields the factorization of N. Indeed, since $\left(\frac{h}{N}\right) = -1$, we can assume without loss of generality that $\left(\frac{h}{p}\right) = 1$ and $\left(\frac{h}{q}\right) = -1$. Hence, letting $\alpha = \Lambda_0/p_1$ and $\beta = \Lambda_0/q_1$ in $\mathbb{Z}$, we have $h^{\Lambda_0} \equiv \left(\frac{h}{p}\right)^\alpha \equiv 1 \pmod{p}$ and $h^{\Lambda_0} \equiv \left(\frac{h}{q}\right)^\beta \equiv -1 \pmod{p}$ since β is odd. In turn, this implies that $\gcd(h^{\Lambda_0} - 1 \bmod N, N) = p$ and $\gcd(h^{\Lambda_0} + 1 \bmod N, N) = q$. Hence, we get $p_1q_1 = (p-1)(q-1)/4$.

Once p_1q_1 is known, an attacker against the TSDH assumption can recover x from (σ_1, r_1) as $x = (1/\sigma_1) - r_1 \bmod p_1q_1$ and next $z := g^y \bmod N$ as $z = (g^{(x+r_0)y})^\gamma \bmod N$ where $\gamma := 1/(x + r_0) \bmod p_1q_1$.

Similarly, an attacker against the TSEI assumption can first recover x from (σ_1, r_1) and then compute a new pair $(1/(x + r^*) \bmod p_1q_1, r^*)$ for any chosen $r^* \in \{0,1\}^\ell$.

EXTENSION. The authors of [1] more generally consider the case of RSA moduli of the form $N = pq$ where $p = 2p_1 \cdots p_I + 1$ and $q = 2q_1 \cdots q_J + 1$ with $p_1, \ldots, p_I$ and $q_1, \ldots, q_J$ prime, and p_1q_1-order subgroups of $\mathbb{Z}_N^*$. In this case, an attacker, from two pairs (σ_1, r_1) and (σ_2, r_2) where $\sigma_i = 1/(x + r_i) \bmod p_1q_1$ with $i \in \{1, 2\}$, can compute Λ as

$$\Lambda := \sigma_1\sigma_2(r_1 - r_2) + \sigma_1 - \sigma_2 \ .$$

Again the main observation is that the so-defined Λ is a multiple of p_1q_1.

Given two integers a and b whose prime factorizations are $a = \prod_{i \in \mathscr{I}} {p_i}^{e_i}$ and $b = \prod_{j \in \mathscr{J}} {p_i}^{f_i}$ with $e_i, f_i > 0$, we define the operator $\mathrm{cop}_b(a) = \prod_{i \in \mathscr{I}, i \notin \mathscr{J}} {p_i}^{e_i}$. This can be obtained by successive GCDs:

$\delta \leftarrow \gcd(a, b)$
until $(\delta = 1)$ **do**
    $a \leftarrow a/\delta$; $\delta \leftarrow \gcd(a, b)$
return a

Against the TSDH assumption, the attacker computes

$$x' := (1/\sigma_1) - r_1 \bmod \mathrm{cop}_{\sigma_1}(\Lambda)\,, \quad \gamma := 1/(x' + r_0) \bmod \mathrm{cop}_{(x'+r_0)}(\Lambda)$$

and then recovers $z := g^y \bmod N$ as

$$z = \left(g^{(x+r_0)y}\right)^{\gamma} \bmod N \ .$$

For the TSEI assumption, an attacker needs to produce a pair $(1/(x + r^*), r^*)$ where the first component is computed modulo p_1q_1. The knowledge of a multiple of p_1q_1 (i.e., Λ) is not enough. However, by considering more than two pairs (σ_i, r_i), the attacker can easily obtain p_1q_1; namely

$$\Lambda' := \gcd\big(\sigma_1\sigma_2(r_1 - r_2) + \sigma_1 - \sigma_2, \sigma_1\sigma_3(r_1 - r_3) + \sigma_1 - \sigma_3\big)$$

is likely to be equal to p_1q_1 or a small multiple thereof (that allows the recovery of p_1q_1 by pulling out its small factors).

References

1. Jong Hwan Park, Kwangsu Lee, and Dong Hoon Lee. Efficient identity-based encryption and public-key signature from trapdoor subgroups. Cryptology ePrint Archive, Report 2016/500, 2016. http://eprint.iacr.org/.