A Note on Subgroup Security in Pairing-Based Cryptography

Tadanori Teruya
National Institute of Advanced Industrial Science and Technology
Koto-ku, Tokyo, Japan
tadanori.teruya@aist.go.jp

ABSTRACT

Barreto et al. (LATINCRYPT 2015) proposed a security notion, called *subgroup security*, for elliptic curves in pairing-based cryptography. They also claimed that, in some schemes, if an elliptic curve is subgroup-secure, the membership check, called full membership check, can be replaced by a cheaper membership check, called light membership check, which results in faster schemes than the original ones. However, they also noticed that some schemes will not maintain security if this replacement is done. It is unclear what schemes allow a secure replacement of the membership check. In this paper, we show a concrete example of insecurity in the sense of subgroup security in order to help developers understand what subgroup security is and what properties are actually preserved. In our conclusion, we recommend the developers to use the full membership check because it is a simple and general technique to securely implement schemes. If the developers use the light membership check for performance reasons, it is critical to carefully check that security is preserved.

CCS CONCEPTS

• **Security and privacy → Cryptanalysis and other attacks**;

KEYWORDS

Pairing-based cryptography; Membership check; Subgroup security

ACM Reference Format:
Tadanori Teruya. 2018. A Note on Subgroup Security in Pairing-Based Cryptography. In *APKC'18: 5th ACM ASIA Public-Key Cryptography Workshop , June 4, 2018, Incheon, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3197507.3197514

1 INTRODUCTION

1.1 Background

Pairing-based cryptography is an attractive research area which has enabled the construction of many schemes with helpful properties to manage sensitive data with practical efficiency [18]. In pairing-based cryptography, bilinear maps, which are called *pairings*, are defined over elliptic curves. Thanks to many results for high-speed implementations [2, 9, 12, 13, 20], it is well-known how to implement secure and efficiently computable pairings.

Pairings suitable for cryptographic purposes are defined as functions which take as inputs two elements of two cyclic groups of prime order r, and output an element of a cyclic group of the same prime order r. Namely, each pairing is defined over three cyclic groups of the same prime order r which are called *bilinear groups*. Plaintexts, ciphertexts, public-keys, secret-keys, and digital signatures, etc. are encoded as the elements of the bilinear groups in pairing-based cryptographic schemes.

To implement every cryptographic scheme, it is important remark that *the validity of every input must be checked.* If one missed the validity check, then the *invalid curve attack* [1, 14, 19] or the *small subgroup attack* [11, 17] can be applied and the security is lost. In order to check the validity of the bilinear groups, we use the *membership check*. The membership check consists of several operations defined over finite fields and elliptic curves. At least, there is one scalar multiplication of a point on the elliptic curve or one exponentiation of an element of a finite field. These are costly operations in pairing-based cryptography.

1.2 Related Work

Recently, Barreto et al. [3] proposed a security notion, called *subgroup security*, for elliptic curves used in pairing-based cryptography. According to [3], if an elliptic curve is subgroup-secure, then, in some schemes, the calculations of a scalar multiplication and an exponentiation for the membership check can be omitted. Thanks to this property, the membership check can be replaced by a cheaper membership check, and the resulting schemes are faster than the originals. In this paper, we call such cheaper membership check a *light membership check*, and we also call the original membership check *full membership check*. Furthermore, in order to show the existence and practicality of subgroup-secure elliptic curves, Barreto et al. [3] presented several carefully chosen subgroup-secure elliptic curves which have significantly efficient computable pairings. They also presented an application scenario of the subgroup security: a blinded BLS signature scheme [5]. The authors of [3] argue that, subgroup-secure elliptic curves enable us to implement faster and secure schemes. Since the publication of [3] , several subgroup-secure elliptic curves have been defined in an IETF Internet-Draft [15]. (Note that this is a work-in-progress document, not standardized yet.)

However, not all schemes allow replacing the full membership check with the light membership check even if the schemes use subgroup-secure elliptic curves. This issue has already been noticed by Barreto et al. [3]. However, it is unclear what schemes allow a secure replacement of the membership check.

APKC'18, June 4, 2018, Incheon, Republic of Korea

ACM ISBN 978-1-4503-5756-2/18/06…$15.00
https://doi.org/10.1145/3197507.3197514

1.3 Our Contribution

In this paper, we show a concrete example of insecurity in the sense of subgroup security. Our example demonstrates how to lose security when the full membership check is replaced by the light membership check. Concretely, we introduce two simple cryptographic schemes. The first scheme is a slightly modified ElGamal encryption scheme, and the second scheme is a simplified re-encryption mix network scheme. We also introduce several security notions in order to demonstrate our concrete example. Our example is shown as an attack procedure when these two schemes use the light membership check.

Note that our example does not use pairings; thus it seems to be irrelevant to pairing-based cryptography. However, the ElGamal encryption and simplified re-encryption mix network based on bilinear groups defined over subgroup-secure elliptic curves should be secure. This reason for this is as follows. Since these schemes are simple, some parts of these schemes are implicitly used as building blocks in many pairing-based cryptographic schemes. Hence, if these simple schemes using subgroup-secure elliptic curves are insecure, many pairing-based cryptographic schemes based on them could be also insecure. We believe that observation of our example is helpful to design, construct, and implement secure pairing-based cryptographic schemes.

By our example and consideration, the basic problem is that the subgroup security and the light membership check could not preserve the hardness of DDH problem [16] or the circuit privacy [6]. Because preservation of these notions seems to be important to construct advanced cryptographic schemes, in many cases, it is probably difficult to replace the full membership check by the light membership check.

Organization. Preliminaries are described in Section 2. A conventional technique of subgroup handling is briefly explained in Section 3, and a brief introduction to subgroup security is given in Section 4. We introduce encryption schemes in Section 5 in order to show our concrete example in Section 6. Then we conclude in Section 7.

2 PRELIMINARIES

In this section, we introduce mathematical backgrounds of pairing-based cryptography and subgroup security.

2.1 Elliptic Curve

Let $\mathbb{F}_p$ be a finite field of field order p, where p is a prime number greater than 3. An elliptic curve E defined over $\mathbb{F}_p$ is given by the Weierstrass equation $Y^2 = X^3 + aX + b$, where X, Y are two variables, $a, b \in \mathbb{F}_p$, and $4a^3 + 27b^2 \neq 0$. For $\ell = 1, 2, \ldots$, let $\mathbb{F}_{p^\ell}$ be an ℓth extension field of $\mathbb{F}_p$. We define $\mathbb{F}_{p^\ell}$-rational points group $E(\mathbb{F}_{p^\ell})$ as follows:

$$E(\mathbb{F}_{p^\ell}) = \{(x, y) \in \mathbb{F}_{p^\ell} \times \mathbb{F}_{p^\ell} \mid y^2 = x^3 + ax + b\} \cup \{O\}, \quad (1)$$

where O is the point at infinity, and we call elements of $E(\mathbb{F}_{p^\ell}) \setminus \{O\}$ finite points. In this paper, every point of E is represented by the affine coordinates. Hence, all the finite points are uniquely represented by a tuple of two elements of the finite fields. However, the point at infinity is not, and its representation depends on the implementation. In practice, one can efficiently implement the representation of the point at infinity. Hereafter we suppose that one can efficiently encode the point at infinity and efficiently distinguish it from finite points.

There exists an operation "+" over $E(\mathbb{F}_{p^\ell})$ such that $E(\mathbb{F}_{p^\ell})$ and + form a group. The unit element of this group is O. The inverse element of $P = (x, y) \in E(\mathbb{F}_{p^\ell})$ is defined as $-P = (x, -y)$. The scalar multiplication by an integer a of $P \in E$ is denoted by $[a]P$ and defined as

$$[a]P = \begin{cases} \overbrace{P + P + \cdots + P}^{a \text{ repetitions}} & \text{if } a > 0, \\ O & \text{if } a = 0, \\ \underbrace{(-P) + (-P) + \cdots + (-P)}_{-a \text{ repetitions}} & \text{otherwise.} \end{cases} \quad (2)$$

The group order of $E(\mathbb{F}_{p^\ell})$ is denoted by $\#E(\mathbb{F}_{p^\ell})$. For $\ell = 1, 2, \ldots$, we define the p^ℓth-power Frobenius map $\pi_{p^\ell} = (x^{p^\ell}, y^{p^\ell})$. We call an integer $t = p + 1 - \#E(\mathbb{F}_p)$ by the trace of Frobenius.

Let n be a positive integer. We define n-torsion group $E[n]$ of E as $E[n] = \{P \in E \mid [n]P = O\} = \ker([n])$. For $\ell = 1, 2, \ldots$, we say that E is supersingular if $E[p^\ell] \simeq 0$, ordinary if $E[p^\ell] \simeq \mathbb{Z}_{p^\ell}$.

We say that a group $\mathbb{G}$ is a cyclic if, for all $h \in \mathbb{G}$, h is written as a multiple of (or a power of) an only one element $g \in \mathbb{G}$ (i.e., $\mathbb{G} = \{h \mid \exists x \in \mathbb{Z}, h = g^x\}$). This element g is called a generator of $\mathbb{G}$.

Let r be a prime number different from p. We say that a positive integer k is the embedding degree of E with respect to r if k is the smallest positive integer such that $r \mid (p^k - 1)$. For technical reason, we assume the same assumption introduced by [13] that is $r^2 \nmid (p^k - 1)$. Note that the embedding degree ensures $E[r] \subseteq E(\mathbb{F}_{p^k})$.

2.2 Pairing

Typically, a pairing is defined as a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of prime order r, and e satisfies the following three properties:

- Non-degenerate, i.e., $\forall P \in \mathbb{G}_1$ and $\exists Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$, and $\forall Q \in \mathbb{G}_2$ and $\exists P \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$.
- Bilinear, i.e., $\forall P, Q \in \mathbb{G}_1$ and $\forall R, S \in \mathbb{G}_2$, $e(P + Q, R) = e(P, R) \cdot e(Q, R)$ and $e(P, R + S) = e(P, R) \cdot e(P, S)$.
- Efficiently computable, i.e., the computational time complexity of e is in a polynomial of $\log r$.

By tradition, e is called a bilinear map, and its input and output groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are called by bilinear groups.

There are several types of pairings, called by Types 1, 2, and 3 [18]. We say that e is a Type 3 pairing if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficient homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. In this paper, we focus on Type 3 because several researchers reported that Type 3 pairings are faster than the others [2, 18, 20].

Suppose that an ordinary elliptic curve E defined over a finite field $\mathbb{F}_p$ has a subgroup of order r in $E(\mathbb{F}_p)$ and the embedding degree k greater than 1 with respect to r. In practice, faster bilinear groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of E are defined as $\mathbb{G}_1 = E[r] \cap \ker(\pi_p - [1])$, $\mathbb{G}_2 = E[r] \cap \ker(\pi_p - [p])$, and $\mathbb{G}_T = \mu_r \subseteq \mathbb{F}_{p^k}^\times$, where $\mathbb{F}_{p^k}^\times$ is the

multiplicative group over $\mathbb{F}_{p^k}$, and μ_r is the rth roots of unity μ_r over $\mathbb{F}_{p^k}^{\times}$ which is defined as $\mu_r = \{x \in \mathbb{F}_{p^k}^{\times} \mid x^r = 1\}$. More specifically, μ_r is a subgroup of the cyclotomic subgroup $\mathbb{G}_{\Phi_k(p)}$ over $\mathbb{F}_{p^k}^{\times}$, where Φ_k is the kth cyclotomic polynomial, concretely, $\mathbb{G}_{\Phi_k(p)}$ is defined as $\mathbb{G}_{\Phi_k(p)} = \{x \in \mathbb{F}_{p^k}^{\times} \mid x^{\Phi_k(p)} = 1\}$. Note that $r \mid \Phi_k(p)$.

Furthermore, considerably faster pairings are defined over ordinary elliptic curves whose CM discriminant $D = 1$ or 3 with additional assumptions. In the case of $D = 1$, namely, the Weierstrass equation of an elliptic curve E is written as $Y^2 = X^3 + aX$, suppose that $4 \mid k$ and let $m = k/4$, then E has 4th degree twist E' whose Weierstrass equation defined over $\mathbb{F}_{p^m}$ is written as $Y^2 = X^3 + a'X$, and there exists an isomorphism $\phi_d : E'(\mathbb{F}_{p^k}) \to E(\mathbb{F}_{p^k})$. In the case of $D = 3$, namely, the Weierstrass equation of E is written as $Y^2 = X^3 + b$, suppose that $d \mid k$ and let $m = k/d$, where $d = 3$ or 6, then E has dth degree twist E' whose Weierstrass equation defined over $\mathbb{F}_{p^m}$ is written as $Y^2 = X^3 + b'$, and there also exists an isomorphism $\phi_d : E'(\mathbb{F}_{p^k}) \to E(\mathbb{F}_{p^k})$. In the both $D = 1$ and 3 cases with assumptions described above, the definition of $\mathbb{G}_2$ is replaced by $\mathbb{G}_2 = E'[r] \cap \ker\left([\xi_d]\pi_{p^m} - [1]\right)$, where ξ_d is a dth primitive root of unity (see [13] for more details).

Hereafter, we assume that the CM discriminant of E is 1 or 3, and the assumptions described above are satisfied.

For notational convenience, we denote three groups $\mathcal{G}_1 = E(\mathbb{F}_p)$, $\mathcal{G}_2 = E'(\mathbb{F}_{p^m})$, and $\mathcal{G}_{\mathrm{T}} = \mathbb{G}_{\Phi_k(p)}$, three cofactors $h_1 = \#\mathcal{G}_1/r$, $h_2 = \#\mathcal{G}_2/r$, and $h_{\mathrm{T}} = \#\mathcal{G}_{\mathrm{T}}/r = \Phi_k(p)/r$, respectively, and three orders of groups $n_1 = \#\mathcal{G}_1$, $n_2 = \#\mathcal{G}_2$, and $n_{\mathrm{T}} = \#\mathcal{G}_{\mathrm{T}}$, respectively. Note that, under the assumptions described above, $r^2 \nmid \#\mathcal{G}_1$, $r^2 \nmid \#\mathcal{G}_2$, and $r^2 \nmid \#\mathcal{G}_{\mathrm{T}}$, respectively. We also define bilinear cosubgroups $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_{\mathrm{T}}$ such that $\mathcal{G}_1 \simeq \mathbb{G}_1 \times \mathcal{H}_1$, $\mathcal{G}_2 \simeq \mathbb{G}_2 \times \mathcal{H}_2$, and $\mathcal{G}_{\mathrm{T}} \simeq \mathbb{G}_{\mathrm{T}} \times \mathcal{H}_{\mathrm{T}}$, respectively.

2.3 Pairing-Friendly Family

Almost all elliptic curves which have efficiently computable pairings are specified by five integers D, k, p, r, t. We call this quintuple a pairing parameter. We introduce a subclass of elliptic curves which have significantly efficient computable pairings.

Definition 2.1 (Pairing-friendly elliptic curve [9]). Suppose E is an elliptic curve defined over a finite field $\mathbb{F}_p$, there exists a prime number r such that $r \mid \#E(\mathbb{F}_p)$. Let k be the embedding degree of E with respect to r. We say that E is *pairing-friendly* if $r \geq \sqrt{p}$ and $k < \log_2(r)/8$.

In general, it is quite hard to find appropriate pairing parameters whose corresponding elliptic curve is pairing-friendly. To overcome this problem, several researchers investigated systematic methods to generate appropriate pairing parameters. According to a survey presented in [9], we can efficiently obtain pairing parameters with pairing-friendly elliptic curves under some restrictions for various security levels and efficient implementations. Precisely, pairing parameters specified by polynomials which are defined as follows.

Definition 2.2 (Family of elliptic curves [9]). Let $p(x)$, $r(x)$, and $t(x)$ be three nonzero polynomials defined over $\mathbb{Q}$. For a given positive integer k and positive square-free integer D, the triple $\big(p(x), r(x), t(x)\big)$ *parameterizes a family of elliptic curves with embedding degree k and CM discriminant D* if the following conditions are satisfied:

- $p(x) = f(x)^d$ for some $d \geq 1$ and $f(x)$ that represents primes (informally, $f(x)$ is a polynomial defined over $\mathbb{Q}$ and there are infinitely many prime numbers $f(x)$ for $x \in \mathbb{Z}$).
- $r(x)$ is not constant, irreducible, and integer-values and has positive leading coefficient.
- $r(x)$ divides $p(x) + 1 - t(x)$.
- $r(x)$ divides $\Phi_k\big(t(x) - 1\big)$, where Φ_k is the kth cyclotomic polynomial.
- The equation $Dy^2 = 4p(x) - t(x)^2$ has infinitely many integer solutions (x, y).

Several families which enable us to obtain pairing-friendly elliptic curves suitable for implementation of practical pairing-based cryptographic schemes have been proposed [9]. We refer the readers to [9] for more details.

Algorithm 1: FullCheck: The full membership check

Input: $(\mathbb{G}, G)$: A group $\mathbb{G}$, and an element G of $\mathbb{G}$ or not.
Output: 1 or 0, where 1 means $G \in \mathbb{G}$, and 0 does not.

if $\mathbb{G} = \mathbb{G}_1$ *or* $\mathbb{G} = \mathbb{G}_2$ **then**
  Parse G as a point $P = (x, y)$;
  if *all conditions in below are satisfied:*
    - $P \neq O$.
    - $x \in \mathbb{F}_p$ *and* $y \in \mathbb{F}_p$ *if* $\mathbb{G} = \mathbb{G}_1$, *or* $x \in \mathbb{F}_{p^m}$ *and* $y \in \mathbb{F}_{p^m}$ *if* $\mathbb{G} = \mathbb{G}_2$.
    - $y^2 = x^3 + ax + b$ *if* $\mathbb{G} = \mathbb{G}_1$, *or* $y^2 = x^3 + a'x + b'$ *if* $\mathbb{G} = \mathbb{G}_2$.
    - $[r]P = O$.

  then return 1;
  else return 0;
else if $\mathbb{G} = \mathbb{G}_{\mathrm{T}}$ **then**
  Parse G as a one value x;
  if *all conditions in below are satisfied:*
    - $x \neq 1$.
    - $x \in \mathbb{F}_{p^k}^{\times}$.
    - $x^r = 1$.

  then return 1;
  else return 0;
else return 0;

3 CONVENTIONAL TECHNIQUE OF SUBGROUP HANDLING

3.1 Membership Check

For notational convenience, we define a procedure of full membership check as a function.

Definition 3.1 (Full membership check). We define an algorithm of *full membership check for* $\mathbb{G}_1$, $\mathbb{G}_2$, *and* $\mathbb{G}_{\mathrm{T}}$ as Algorithm 1, and we denote this algorithm as a function FullCheck.

Note that if FullCheck outputs 1, the input is valid, otherwise, invalid. Every pairing-based cryptographic scheme requires that

solving the discrete logarithm problems defined over $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are impractical. In conventional choices of elliptic curves for pairing-based cryptography, the hardness of the discrete logarithm problems defined over other groups (e.g., $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_T$) are not guaranteed. This is the reason why every pairing-based cryptographic scheme requires every input to be represented in valid way. Hence, the full membership check for all the inputs is mandatory before calculating the main procedure in schemes. Expensive operations in the full membership check are the scalar multiplication and the exponentiation by r.

3.2 Subgroup Elimination

The full membership check could be replaced by eliminating subgroups. For examples, for an input P of the elliptic curve E (resp. an input Q of the elliptic curve E', and an input x of the finite field). Then:

(1) Compute $P' \leftarrow [h_1]P$ (resp. $Q' \leftarrow [h_2]Q$, and $x' \leftarrow x^{h_T}$).
(2) Abort if $P' = O$ (resp. $Q' = O$, and $x' = 1$).
(3) Use P' (resp. Q', and x') instead of P (resp. Q, and x).

As seen above, each procedure eliminates elements of bilinear cosubgroup in P (resp. Q, and x). In this paper, we call this method *cosubgroup elimination*. This method is also applicable for every scheme. In almost all of pairing-friendly elliptic curves, the bit length of h_1 is much smaller than that of r, hence, the scalar multiplication by h_1 is not expensive. In contrast, the bit lengths of h_2 and h_T are much larger than that of r, and the scalar multiplications by h_2 and h_T are expensive. Fuentes-Castañeda et al. [10] proposed how to compute the scalar multiplication by h_2 faster than r over $\mathcal{G}_2$. The computation of exponentiation by h_T over $\mathcal{G}_T$ is almost same as the (hard part of) final exponentiation. A faster final exponentiation is also proposed by [10].

However, there are some concerns when the cosubgroup elimination is used instead of the full membership check. Every input element is multiplied by h_1 (resp. multiplied by h_2, and powered by h_T) even if the element is already contained in $\mathbb{G}_1$ (resp. $\mathbb{G}_2$, and $\mathbb{G}_T$). In other words, the cosubgroup elimination ignores validity and changes every input. Developers must revise every scheme carefully for the correctness and the compatibility. It seems to be difficult for non-experts to notice errors if this revision is wrong.

4 BRIEF INTRODUCTION OF SUBGROUP SECURITY

First of all, we describe *subgroup security* [3].

Definition 4.1 (Subgroup security [3]). Let $p(u), t(u), r(u) \in \mathbb{Q}[u]$ parameterize a family of ordinary pairing-friendly elliptic curves, and for any particular $u_0 \in \mathbb{Z}$ such that $p = p(u_0)$ and $r = r(u_0)$ are prime, let E be the resulting pairing-friendly elliptic curve defined over $\mathbb{F}_p$ such that $\#E(\mathbb{F}_p)$ is divisible by r. We say that E is *subgroup-secure* if all $\mathbb{Q}[u]$-irreducible factors of $h_1(u)$, $h_2(u)$, and $h_T(u)$ that can represent primes and that have degree at least that of $r(u)$, contain no prime factors smaller than $r(u_0) \in \mathbb{Z}$ when evaluated at $u = u_0$.

Then we also define a procedure of light membership check as a function.

Algorithm 2: LightCheck: The light membership check

Input: $(\mathbb{G}, G)$: A group $\mathbb{G}$, and an element G of $\mathbb{G}$ or not.
Output: 1 or 0, where 1 means $G \in \mathcal{G}_2$ or $\mathcal{G}_T$, and 0 does not.

1 **if** $\mathbb{G} = \mathbb{G}_2$ **then**
2 Parse G as a point $P = (x, y)$;
3 **if** *all conditions in below are satisfied:*
  - $P \neq O$.
  - $x \in \mathbb{F}_{p^m}$ *and* $y \in \mathbb{F}_{p^m}$.
  - $y^2 = x^3 + a'x + b'$.
4 **then return** 1;
5 **else return** 0;
6 **else if** $\mathbb{G} = \mathbb{G}_T$ **then**
7 Parse G as a one value x;
8 **if** *all conditions in below are satisfied:*
  - $x \neq 1$.
  - $x \in \mathbb{F}^{\times}_{p^k}$.
  - $x^{\Phi_k(p)} = 1$.
9 **then return** 1;
10 **else return** 0;

Definition 4.2 (Light membership check). We define an algorithm of *light membership check for* $\mathbb{G}_2$ *and* $\mathbb{G}_T$ as Algorithm 2, and we denote this algorithm as a function LightCheck.

Note that Algorithm 2 comes from a recommendation proposed by Barreto et al. [3]. The light membership check for $\mathbb{G}_1$ is not defined, the reason probably being that the degree of $h_1(u)$ is smaller than that of $r(u)$, and the operations of $\mathcal{G}_1$ are much more efficient than that of $\mathcal{G}_2$ and $\mathcal{G}_T$. Thus, there is no motivation to accelerate the full membership check for $\mathbb{G}_1$. In [3], they recommended that use the full membership check or the cosubgroup elimination for $\mathbb{G}_1$.

Barreto et al. [3] claimed that thanks to the property of the subgroup security, some pairing-based cryptographic schemes preserve their security when the full membership check is replaced by the light membership check. The reason is the hardness of the discrete logarithm problems defined over $\mathcal{H}_2$ and $\mathcal{H}_T$ are guaranteed by the definition of subgroup security. In other words, the small subgroup attack [17] cannot be applied. The resulting scheme is faster than the original scheme, because the light membership check has no expensive operation.

However, Barreto et al. [3] also noticed that every scheme does not allow the above replacement in a secure manner. A method for determining possibility of this replacement has not been proposed yet.

5 CRYPTOGRAPHIC SCHEMES FOR CONCRETE EXAMPLE

Hereafter, we use multiplicative notation for every group for notational convenience (i.e., "+" and O are replaced by "·" and 1, respectively).

5.1 ElGamal Encryption with Membership Check

For our main purpose, we introduce a slightly modified syntax for public key encryption. However, this syntax naturally captures the properties of the ElGamal encryption [8], which is a one of the well-known constructions of the public key encryption defined over cyclic groups.

5.1.1 Syntax. Let λ be a security parameter, and let r be a prime number such that $r = \Theta(2^\lambda)$. Let $\mathbb{G}$ be a cyclic group of order r. The syntax of public key encryption PKE for the plaintext space $\mathbb{G}$ is defined by the following algorithms.

$\mathsf{ParamGen}(1^\lambda) \to \mathsf{pp}$**:** The public parameter generation algorithm ParamGen takes as an input a security parameter λ, and then output a public parameter pp.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$**:** The key generation algorithm KeyGen takes as an input a public parameter pp and outputs a pair (sk, pk) of a secret-key and a public-key.

$\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m) \to c$**:** The encryption algorithm Enc takes as inputs a public parameter pp, a public-key pk, and an element $m \in \mathbb{G}$ as a plaintext. It outputs a ciphertext c.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk}, c) \to m$**:** The decryption algorithm Dec takes as inputs a public parameter pp, a secret-key sk, a public-key pk, and a ciphertext c, and outputs an element $m \in \mathbb{G}$ as a plaintext.

$\mathsf{Check}(\mathsf{pp}, c) \to 0$ **or** 1**:** The membership check algorithm Check takes as inputs a public parameter pp and a ciphertext c. It outputs 0 or 1.

$\mathsf{ReRand}(\mathsf{pp}, \mathsf{pk}, c) \to c'$ **or** $\perp$**:** The re-randomization algorithm ReRand takes as inputs a public parameter pp, a public-key pk, and a ciphertext c. It outputs a ciphertext c' or a special symbol $\perp$.

Hereafter, for simplicity, we assume when a public parameter pp (which contains a security parameter, finite fields, elliptic curves, and etc.) is generated, pp is implicitly shared with all entities. We omit descriptions of pp as input if it is allowed.

5.1.2 Correctness. We say that a scheme PKE for $\mathbb{G}$ is correct if the following properties hold. Suppose $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$ and then $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.

- For any $m \in \mathbb{G}$, if $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, then $\mathsf{Check}(c) = 1$, otherwise $\mathsf{Check}(c) = 0$.
- For any $m \in \mathbb{G}$, suppose $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, then $\mathsf{Dec}(\mathsf{sk}, \mathsf{pk}, c) = m$.
- For any $m \in \mathbb{G}$, if $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, then $\mathsf{Dec}(\mathsf{sk}, \mathsf{pk}, c) = \mathsf{Dec}(\mathsf{sk}, \mathsf{pk}, c')$, where $c' \leftarrow \mathsf{ReRand}(\mathsf{pk}, c)$, otherwise $\perp \leftarrow \mathsf{ReRand}(\mathsf{pk}, c)$.

5.1.3 Security.

Definition 5.1 (Indistinguishability against chosen-plaintext attack). Let λ be a security parameter and let $\mathcal{A}$ be a probabilistic polynomial-time (p.p.t.) adversary. We denote the following game by $\mathsf{Exp}_{\mathrm{IND-CPA}}(\lambda, \mathcal{A})$:

(1) $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$.
(2) $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.
(3) Choose b from $\{0, 1\}$ uniformly at random.
(4) $(m_0, m_1) \leftarrow \mathcal{A}(\mathrm{find}, \mathsf{pk})$.
(5) $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$.
(6) $\hat{b} \leftarrow \mathcal{A}(\mathrm{guess}, \mathsf{pk}, c)$.
(7) Output 1 if $b = \hat{b}$, 0 otherwise.

We also define the advantage $\mathsf{Adv}_{\mathrm{IND-CPA}}(\lambda, \mathcal{A})$ of $\mathsf{Exp}_{\mathrm{IND-CPA}}(\lambda, \mathcal{A})$ as follows:

$$\mathsf{Adv}_{\mathrm{IND-CPA}}(\lambda, \mathcal{A}) = \left| \Pr\left[\mathsf{Exp}_{\mathrm{IND-CPA}}(\lambda, \mathcal{A}) = 1\right] - \frac{1}{2} \right|. \quad (3)$$

We say that a scheme PKE is secure in the sense of *indistinguishability against chosen-plaintext attack (IND-CPA-secure)* if $\mathsf{Adv}_{\mathrm{IND-CPA}}(\lambda, \mathcal{A})$ is negligible in λ for any p.p.t. $\mathcal{A}$.

5.1.4 Construction. If there exists a cyclic group $\mathbb{G}$ of prime order r such that the decisional Diffie–Hellman (DDH) problem [16] is hard, the ElGamal encryption scheme defined over $\mathbb{G}$ [8] is a one of well-known constructions of IND-CPA-secure PKE for $\mathbb{G}$. In this paper, we consider bilinear groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_\mathrm{T}$. So we construct PKE using $\mathbb{G} = \mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_\mathrm{T}$.

$\mathsf{ParamGen}(1^\lambda)$**:** Generate $(\mathbb{G}, g)$, where $\mathbb{G} = \mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_\mathrm{T}$ of prime order r, r is a prime number such that $r = \Theta(2^\lambda)$, and g is a generator of $\mathbb{G}$. Output $\mathsf{pp} = (\mathbb{G}, g)$.

$\mathsf{KeyGen}(\mathsf{pp})$**:** Parse $(\mathbb{G}, g) \leftarrow \mathsf{pp}$. Pick s from $\mathbb{Z}_r^\times$ uniformly at random, then set $h \leftarrow g^s$, and then set $\mathsf{sk} \leftarrow (s)$ and $\mathsf{pk} \leftarrow (h)$. Output (sk, pk).

$\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m)$**:** Parse $(\mathbb{G}, g) \leftarrow \mathsf{pp}$ and parse $(h) \leftarrow \mathsf{pk}$. Pick x from $\mathbb{Z}_r^\times$ uniformly at random, then compute $a \leftarrow g^x$ and $b \leftarrow m \cdot h^x$. Output $c = (a, b)$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk}, c)$**:** Parse $(\mathbb{G}, g) \leftarrow \mathsf{pk}$, $(s) \leftarrow \mathsf{sk}$, $(h) \leftarrow \mathsf{pk}$, and $(a, b) \leftarrow c$. Output $m = b/a^s$.

$\mathsf{Check}(\mathsf{pp}, c)$**:** Parse $(\mathbb{G}, g) \leftarrow \mathsf{pp}$ and parse $(a, b) \leftarrow c$. If $\mathsf{FullCheck}(\mathbb{G}, a) = 1 \wedge \mathsf{FullCheck}(\mathbb{G}, b) = 1$, then output 1, otherwise output 0.

$\mathsf{ReRand}(\mathsf{pp}, \mathsf{pk}, c)$**:** Parse $(\mathbb{G}, g) \leftarrow \mathsf{pp}$, $(h) \leftarrow \mathsf{pk}$, $(a, b) \leftarrow c$. If $\mathsf{Check}(\mathsf{pp}, c) = 0$, then output $\perp$. Otherwise pick x from $\mathbb{Z}_r^\times$ uniformly at random, then compute $a' \leftarrow a \cdot g^x$ and $b' \leftarrow b \cdot h^x$. Then output $c' = (a', b')$.

Lemma 5.2. *(Informal.) Let $\mathbb{G}$ be a cyclic group of prime order r. If the DDH problem defined over $\mathbb{G}$ is hard, then the ElGamal encryption scheme defined over $\mathbb{G}$ is an IND-CPA-secure* PKE *for $\mathbb{G}$.*

Intuitively, if an encryption scheme is IND-CPA-secure, then there is no p.p.t. adversary which can distinguish any two ciphertexts of the scheme.

We refer the readers to [16] for the details of the definition of the DDH problem and a proof of the above lemma.

5.2 Simplified Re-Encryption Mix Network

In this section, we define a simplified version of re-encryption mix network for explanation of our concrete example. Typically, a mix network is a kind of multi-party protocol to realize an anonymous (untraceable) communication channel [7]. More precisely, the mix network hides the link between inputs and outputs. For simplicity, we simplify and define a scheme of a re-encryption mix network protocol based on PKE for $\mathbb{G}$.

5.2.1 Syntax. A simplified re-encryption mix network SMixNet based on PKE for $\mathbb{G}$ is defined by functions ParamGen, KeyGen, Enc, Dec, ReRand, and Shuffle. We define the syntax of Shuffle as follows. The definitions of syntax of other functions are the same as the definitions of PKE for $\mathbb{G}$ in Section 5.1.

Shuffle$(\mathsf{pp}, \mathsf{pk}, \sigma, \boldsymbol{v}) \to \boldsymbol{v}'$ **or** $\perp$**:** The shuffle algorithm Shuffle takes as inputs a public parameter pp, a public-key pk, a permutation σ of degree-ℓ, and a vector $\boldsymbol{v}$ of ℓ ciphertexts $\boldsymbol{v} = (c_0, c_1, \ldots, c_{\ell-1})$. It outputs another vector $\boldsymbol{v}' = (c'_0, c'_1, \ldots, c'_{\ell-1})$ or a special symbol $\perp$.

5.2.2 Correctness. We say that a scheme SMixNet based on a scheme PKE for $\mathbb{G}$ is correct if the following property holds. Suppose $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$. For any positive integer $\ell > 1$ such that $\ell = O(\mathrm{poly}(\lambda))$, any $m_i \in \mathbb{G}$ for $i = 0, 1, \ldots, \ell-1$, and any permutation σ of degree-ℓ, if $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ for $i = 0, 1, \ldots, \ell-1$, then $\mathsf{Shuffle}\big(\mathsf{pk}, \sigma, (c_0, c_1, \ldots, c_{\ell-1})\big)$ outputs $(c'_0, c'_1, \ldots, c'_{\ell-1})$ and $\mathsf{Dec}(\mathsf{sk}, \mathsf{pk}, c_i) = \mathsf{Dec}\big(\mathsf{sk}, \mathsf{pk}, c'_{\sigma(i)}\big)$ for $i = 0, 1, \ldots, \ell-1$, otherwise $\mathsf{Shuffle}\big(\mathsf{pk}, \sigma, (c_0, c_1, \ldots, c_{\ell-1})\big) = \perp$.

5.2.3 Security.

Definition 5.3 (Permutation privacy). Let λ be a security parameter, let $\mathcal{A}$ be a p.p.t. adversary which can access the encryption oracle. We define a game $\mathsf{Exp}^{\mathrm{PP}}(\lambda, \mathcal{A})$ by the following procedure:

(1) $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$.
(2) $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.
(3) Choose a positive integer $\ell > 1$ such that $\ell = O(\mathrm{poly}(\lambda))$.
(4) Choose a permutation σ of degree-ℓ uniformly at random.
(5) When $\mathcal{A}(\mathrm{find}, \ell)$ is run:
   (a) $\mathcal{A}$ access the encryption oracle ℓ times with inputs ℓ plaintexts $(m_0, \ldots, m_{\ell-1})$.
   (b) When $\mathcal{A}$ receives ℓ ciphertexts $\boldsymbol{v} = (c_0, \ldots, c_{\ell-1})$, $\mathcal{A}$ outputs a vector $\hat{\boldsymbol{v}}$ of ℓ ciphertexts $\hat{\boldsymbol{v}} = (\hat{c}_0, \ldots, \hat{c}_{\ell-1})$.
(6) $\boldsymbol{v}' = (c'_0, \ldots, c'_{\ell-1}) \leftarrow \mathsf{Shuffle}(\mathsf{pk}, \sigma, \hat{\boldsymbol{v}})$.
(7) If $\boldsymbol{v}' = \perp$, then output 0.
(8) $\hat{\sigma} \leftarrow \mathcal{A}(\mathrm{guess}, \boldsymbol{v}, \boldsymbol{v}')$.
(9) Output 1 if $\sigma = \hat{\sigma}$, 0 otherwise.

Note that $\mathcal{A}$ does not have the public-key pk. We also define the advantage $\mathsf{Adv}^{\mathrm{PP}}(\lambda, \mathcal{A})$ of $\mathsf{Exp}^{\mathrm{PP}}(\lambda, \mathcal{A})$ as follows:

$$\mathsf{Adv}^{\mathrm{PP}}(\lambda, \mathcal{A}) = \left| \Pr\left[\mathsf{Exp}^{\mathrm{PP}}(\lambda, \mathcal{A}) = 1\right] - \frac{1}{\ell!} \right|. \tag{4}$$

We say that a scheme SMixNet is *permutation-private (P-P)* if $\mathsf{Adv}^{\mathrm{PP}}(\lambda, \mathcal{A})$ is negligible in λ for any p.p.t. $\mathcal{A}$.

Note that the P-P notion only guarantees a weak form of security as it does not guarantee any partial information hiding of the permutation. We define a stronger security notion which captures the level of security expected in practice as follows.

Definition 5.4 (Permutation indistinguishability). Let λ be a security parameter, let $\mathcal{A}$ be a p.p.t. adversary. We define a game $\mathsf{Exp}^{\mathrm{PI}}(\lambda, \mathcal{A})$ by the following procedure:

(1) $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$.
(2) $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.
(3) Choose a positive integer $\ell > 1$ such that $\ell = O(\mathrm{poly}(\lambda))$.
(4) Choose b from $\{0, 1\}$ uniformly at random.
(5) $(\hat{\boldsymbol{v}}, \sigma_0, \sigma_1) \leftarrow \mathcal{A}(\mathrm{find}, \mathsf{pk}, \ell)$, where $\hat{\boldsymbol{v}} = (\hat{c}_0, \ldots, \hat{c}_{\ell-1})$.
(6) $\boldsymbol{v}' \leftarrow \mathsf{Shuffle}(\mathsf{pk}, \sigma_b, \hat{\boldsymbol{v}})$.
(7) If $\boldsymbol{v}' = \perp$, then output 0.
(8) $\hat{b} \leftarrow \mathcal{A}(\mathrm{guess}, \mathsf{pk}, \hat{\boldsymbol{v}}, \boldsymbol{v}')$.
(9) Output 1 if $b = \hat{b}$, 0 otherwise.

We also define the advantage $\mathsf{Adv}^{\mathrm{PI}}(\lambda, \mathcal{A})$ of $\mathsf{Exp}^{\mathrm{PI}}(\lambda, \mathcal{A})$ as follows:

$$\mathsf{Adv}^{\mathrm{PI}}(\lambda, \mathcal{A}) = \left| \Pr\left[\mathsf{Exp}^{\mathrm{PI}}(\lambda, \mathcal{A}) = 1\right] - \frac{1}{2} \right|. \tag{5}$$

We say that a scheme SMixNet is *permutation-indistinguishable (P-IND)* if $\mathsf{Adv}^{\mathrm{PI}}(\lambda, \mathcal{A})$ is negligible in λ for any p.p.t. $\mathcal{A}$.

Lemma 5.5. *If a scheme* SMixNet *is P-IND, then* SMixNet *is also P-P.*

Proof. (Sketch.) We show a proof of Lemma 5.5 by contradiction. Suppose there exists a p.p.t. adversary $\mathcal{A}$ such that it breaks the P-P, i.e., $\mathsf{Adv}^{\mathrm{PP}}(\lambda, \mathcal{A})$ is not negligible in λ. Let us consider the game of $\mathsf{Exp}^{\mathrm{PI}}(\lambda, \mathcal{B})$ between a challenger $\mathcal{C}$, where $\mathcal{B}$ is a p.p.t. adversary using $\mathcal{A}$. $\mathcal{C}$ generates pp, sk, and pk, and then $\mathcal{C}$ sends pk (and pp) to $\mathcal{B}$. Hence, $\mathcal{B}$ can answer any query of the encryption oracle issued by $\mathcal{A}$ when $\mathcal{A}(\mathrm{find}, \ell)$ is run. Then $\mathcal{B}$ sends $(\hat{\boldsymbol{v}}, \sigma_0, \sigma_1)$ to $\mathcal{C}$, where $\hat{\boldsymbol{v}}$ is the result of $\mathcal{A}(\mathrm{find}, \ell)$, σ_0 and σ_1 are two permutations of degree-ℓ. Then $\mathcal{C}$ sends $\boldsymbol{v}'$ to $\mathcal{B}$, where $\boldsymbol{v}'$ is an output of Shuffle. Therefore, $\mathcal{B}$ simulates the game $\mathsf{Exp}^{\mathrm{PP}}(\lambda, \mathcal{A})$. Hence $\mathsf{Adv}^{\mathrm{PI}}(\lambda, \mathcal{B}) = \mathsf{Adv}^{\mathrm{PP}}(\lambda, \mathcal{A})$ is not negligible in λ by the assumption. □

Theorem 5.6. *A scheme* SMixNet *is P-IND if the underlying scheme* PKE *is IND-CPA-secure.*

Proof. (Sketch.) We use a hybrid argument [16] and the triangle inequality to prove Theorem 5.6. Consider the game $\mathsf{Exp}^{\mathrm{PI}}(\lambda, \mathcal{B})$ with a challenger $\mathcal{C}$, where $\mathcal{B}$ is a p.p.t. adversary.

(1) $\mathcal{C}$ computes $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and chooses a positive integer $\ell > 1$ such that $\ell = O(\mathrm{poly}(\lambda))$.
(2) Then $\mathcal{C}$ chooses b from $\{0, 1\}$ uniformly at random.
(3) Then $\mathcal{C}$ sends (pk, ℓ) to $\mathcal{B}$.
(4) $\mathcal{B}$ chooses ℓ plaintexts $m_0, \ldots, m_{\ell-1}$.
(5) Then $\mathcal{B}$ generates $\hat{\boldsymbol{v}}$, σ_0, and σ_1, where $\hat{\boldsymbol{v}} = (\hat{c}_0, \ldots, \hat{c}_{\ell-1})$ and $\hat{c}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ for $i = 0, 1, \ldots, \ell-1$, and σ_0 and σ_1 are two permutations of degree-ℓ.

Firstly, $\mathsf{Check}(\hat{c}_i) = 1$ for all $i = 0, 1, \ldots, \ell-1$. Let us consider the following vectors:

$$(\widetilde{c}_0, \ldots, \widetilde{c}_{\ell-1}) \leftarrow \mathsf{Shuffle}(\mathsf{pk}, \sigma_0, \hat{\boldsymbol{v}}),$$
$$(\overline{c}_0, \ldots, \overline{c}_{\ell-1}) \leftarrow \mathsf{Shuffle}(\mathsf{pk}, \sigma_1, \hat{\boldsymbol{v}}).$$

We also consider two distributions of 2ℓ ciphertexts in the above. Let $S_0 = \{(\widetilde{c}_0, \ldots, \widetilde{c}_{\ell-1})\}$ and let $S_1 = \{(\overline{c}_0, \ldots, \overline{c}_{\ell-1})\}$. We also

Algorithm 3: A construction of Shuffle based on the ElGamal encryption scheme

Input: $(\mathsf{pp}, \mathsf{pk}, \sigma, \boldsymbol{v})$, where $\boldsymbol{v} = (c_0, c_1, \ldots, c_{\ell-1})$.
Output: $\boldsymbol{v}' = (c'_0, c'_1, \ldots, c'_{\ell-1})$ or $\perp$.

```
for j = 0 to ℓ − 1 do
    c'_j ← ReRand(pp, pk, c_{σ^{-1}(j)});
    if c'_j = ⊥ then return ⊥;
end
return v' = (c'_0, c'_1, ..., c'_{ℓ−1});
```

consider the following hybrid distributions:

$$\begin{aligned} H_0 = S_0 &= \{(\widetilde{c}_0, \widetilde{c}_1, \ldots, \widetilde{c}_{\ell-1})\},\\ H_1 &= \{(\overline{c}_0, \widetilde{c}_1, \ldots, \widetilde{c}_{\ell-1})\},\\ H_2 &= \{(\overline{c}_0, \overline{c}_1, \ldots, \widetilde{c}_{\ell-1})\},\\ &\vdots\\ H_{\ell-1} &= \{(\overline{c}_0, \ldots, \overline{c}_{\ell-2}, \widetilde{c}_{\ell-1})\},\\ H_\ell = S_1 &= \{(\overline{c}_0, \ldots, \overline{c}_{\ell-2}, \overline{c}_{\ell-1})\}. \end{aligned}$$

If the underlying scheme PKE is IND-CPA-secure, there is no p.p.t. adversary which can distinguish any two ciphertexts of PKE. Namely, for all $j = 0, 1, \ldots, \ell-1$ and for any p.p.t. adversary $\mathcal{D}$,

$$\epsilon_j = \left|\Pr[\mathcal{D}(\boldsymbol{v}) = 1 \mid \boldsymbol{v} \leftarrow H_j] - \Pr[\mathcal{D}(\boldsymbol{v}) = 1 \mid \boldsymbol{v} \leftarrow H_{j+1}]\right|$$

is negligible in λ. Applying the hybrid argument and the triangle inequality, there is also no p.p.t. adversary $\mathcal{A}$ which can distinguish H_0 and H_ℓ because

$$\begin{aligned} 2 \cdot \mathsf{Adv}_{\mathrm{PI}}(\lambda, \mathcal{A}) &= \left|\Pr[\mathcal{A}(\mathsf{guess}, \boldsymbol{v}, \boldsymbol{v}') = 1 \mid \boldsymbol{v}' \leftarrow H_0]\right.\\ &\qquad \left. - \Pr[\mathcal{A}(\mathsf{guess}, \boldsymbol{v}, \boldsymbol{v}') = 1 \mid \boldsymbol{v}' \leftarrow H_\ell]\right|\\ &\le \sum_{j=0}^{\ell-1} \left|\Pr[\mathcal{D}(\boldsymbol{v}) = 1 \mid \boldsymbol{v} \leftarrow H_j] - \Pr[\mathcal{D}(\boldsymbol{v}) = 1 \mid \boldsymbol{v} \leftarrow H_{j+1}]\right|\\ &\le \ell \cdot \max_j \epsilon_j. \end{aligned}$$

Recall that $\ell = O(\mathrm{poly}(\lambda))$. Hence, $\mathsf{Exp}_{\mathrm{PI}}(\lambda, \mathcal{B})$ is negligible in λ for any p.p.t. adversary $\mathcal{B}$. □

5.2.4 Construction. We show a construction of Shuffle by using the ElGamal encryption scheme in Algorithm 3. Note that, by the definition in Section 5.1.4, ReRand in Algorithm 3 uses the full membership check FullCheck.

The ElGamal encryption scheme is a IND-CPA-secure encryption scheme. By Theorem 5.6, a scheme SMixNet, whose Shuffle is constructed by Algorithm 3, is P-IND.

6 A CONCRETE EXAMPLE OF INSECURITY

In this section, we show a concrete example of insecurity in the sense of subgroup security. Our example is a demonstration of an attack against the permutation-private (P-P) security of a scheme SMixNet whose Shuffle is constructed as shown in Algorithm 3, and underlying group $\mathbb{G}$ used by the ElGamal encryption scheme is subgroup-secure. However, FullCheck used in Check is replaced by LightCheck. We denote this scheme by SMixNet′, and also denote Check by Check′ when FullCheck is replaced by LightCheck.

6.1 Attack against Subgroup Security

First of all, we fix a group $\mathbb{G} = \mathbb{G}_2$ which is defined on a subgroup-secure elliptic curve and underlying group in the ElGamal encryption scheme used by the scheme SMixNet′. (Note that the following discussion is also applicable to $\mathbb{G} = \mathbb{G}_{\mathrm{T}}$.)

We show that SMixNet′ is not P-P by constructing a p.p.t. adversary breaking its security.

Theorem 6.1. *For* SMixNet′ *(i.e., its* Shuffle *is constructed as Algorithm 3 and* Check′ *is used), there exists a p.p.t. adversary* $\hat{\mathcal{A}}$ *such that* $\mathsf{Adv}_{\mathrm{PP}}(\lambda, \hat{\mathcal{A}})$ *is not negligible in* λ.

Proof. Consider the game $\mathsf{Exp}_{\mathrm{PP}}(\lambda, \hat{\mathcal{A}})$ defined in Definition 5.3. When $\hat{\mathcal{A}}(\mathsf{find}, \ell)$ is run:

(1) $\hat{\mathcal{A}}$ chooses ℓ plaintexts $(m_0, \ldots, m_{\ell-1})$.
(2) Then $\hat{\mathcal{A}}$ access to the encryption oracle ℓ times with inputs the plaintexts $(m_0, \ldots, m_{\ell-1})$.
(3) When $\hat{\mathcal{A}}$ receives ℓ ciphertexts $\boldsymbol{v} = (c_0, \ldots, c_{\ell-1})$, $\hat{\mathcal{A}}$ parses $(a_i, b_i) \leftarrow c_i$ for $i = 0, 1, \ldots, \ell-1$.
(4) Then $\hat{\mathcal{A}}$ chooses ℓ distinct points $\hat{h}_0, \hat{h}_1, \ldots, \hat{h}_{\ell-1}$ from $\mathcal{H}_2$. (Note that $\hat{\mathcal{A}}$ knows the descriptions of $\mathcal{H}_2$ and $\mathcal{G}_2$ because $\hat{\mathcal{A}}$ has a public parameter pp in this game.)
(5) Then $\hat{\mathcal{A}}$ computes $\hat{a}_i \leftarrow a_i \cdot \hat{h}_i$ and for $i = 0, 1, \ldots, \ell-1$.
(6) Then $\hat{\mathcal{A}}$ outputs $\hat{\boldsymbol{v}} = (\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_{\ell-1})$, where $\hat{c}_i = (\hat{a}_i, b_i)$ for $i = 0, 1, \ldots, \ell-1$.

When $\hat{\mathcal{A}}(\mathsf{guess}, \boldsymbol{v}, \boldsymbol{v}')$ is run, let $\boldsymbol{v}' = (c'_0, c'_1, \ldots, c'_{\ell-1})$:

(1) $\hat{\mathcal{A}}$ computes $\hat{h}'_i \leftarrow \hat{h}_i^r$, where r is the order of $\mathbb{G}_2$, for $i = 0, 1, \ldots, \ell-1$. (Note that $\hat{h}_0, \ldots, \hat{h}_{\ell-1}$ were generated by $\hat{\mathcal{A}}$ when $\hat{\mathcal{A}}(\mathsf{find}, \ell)$ was run.)
(2) Then $\hat{\mathcal{A}}$ parses $(a'_i, b'_i) \leftarrow c'_i$ and then computes $a''_i \leftarrow (a'_i)^r$ for $i = 0, 1, \ldots, \ell-1$.
(3) Then $\hat{\mathcal{A}}$ outputs a permutation $\hat{\sigma}$ of degree-ℓ such that $\hat{h}'_i = a''_{\hat{\sigma}(i)}$.

Now we consider what $\hat{\mathcal{A}}$ did. Firstly, by the definition of $\mathcal{G}_2$ (explained in Section 2.2), $\mathcal{G}_2 \simeq \mathbb{G}_2 \times \mathcal{H}_2$; thus every element x of $\mathcal{G}_2$ corresponds one-to-one to a pair of two elements $(x_0, x_1) \in \mathbb{G}_2 \times \mathcal{H}_2$ because $\gcd(r, \#\mathcal{H}_2) = 1$. Secondly, the difference between SMixNet and SMixNet′ is Check′, therefore ReRand only randomizes $\mathbb{G}_2$ part and does not modify $\mathcal{H}_2$ part. More precisely, consider the game $\mathsf{Exp}_{\mathrm{PP}}(\lambda, \hat{\mathcal{A}})$ above. For $i = 0, 1, \ldots, \ell-1$, let $(a_i, b_i) = c_i$, and $\hat{\mathcal{A}}$ outputs $\hat{c}_i = (\hat{h}_i \cdot a_i, b_i)$ when $\hat{\mathcal{A}}(\mathsf{find}, \ell)$ is run, then $\hat{\mathcal{A}}$ receives $c'_{\sigma(i)} = \left(\hat{h}_i \cdot a'_{\sigma(i)}, b'_{\sigma(i)}\right)$, and then $\hat{\mathcal{A}}$ obtains $\hat{h}'_i = \hat{h}_i^r \leftarrow \left(\hat{h}_i \cdot a'_{\sigma(i)}\right)^r$. Finally, in the game $\mathsf{Exp}_{\mathrm{PP}}(\lambda, \hat{\mathcal{A}})$, Check′ does not output 0, because Check′ uses LightCheck. Hence, $\hat{\mathcal{A}}$ always outputs $\hat{\sigma} = \sigma$. [1] □

6.2 Discussion

First of all, we consider why the scheme SMixNet′ is insecure. As mentioned above, the difference between SMixNet and SMixNet′ is only Check′. Since the operations in both schemes only act on

[1] We present Sage scripts of our example in https://github.com/tell/note-on-subgroup-security.

$\mathbb{G}_2$, adversaries can manipulate $\mathcal{H}_2$ to break the security. In short, *the subgroup security and the light membership check could not preserve the hardness of DDH problem [16] or the circuit privacy [6].*

Now we consider countermeasures against the attack described above. First of all, a simple countermeasure is the full membership check.

Next, let us consider other countermeasures of how to still use the light membership check preserving the security of the scheme SMixNet′. One possibility is replacing $\mathbb{G}_2$ by $\mathcal{G}_2$, and revising the ElGamal encryption scheme for the correctness and the security. In this way, in a revised scheme, since all elements belong to $\mathcal{G}_2$, integers are sampled from $\mathbb{Z}_{n_2}^{\times}$ uniformly at random when calculating the exponentiations (the scalar multiplications) in KeyGen, Enc, and ReRand. By the definition of $\mathcal{G}_2$, the bit length of n_2 is much larger than that of r; therefore this revision causes significantly slower calculation of the exponentiations. (Another possibility is exploiting the mapping of isomorphism $\mathcal{G}_2 \simeq \mathbb{G}_2 \times \mathcal{H}_2$. However, this seems to be equivalent to use the full membership check.) Note that it is unknown whether this modification can be applied for every cryptographic scheme.

7 CONCLUSION

We showed a concrete example of insecurity in the sense of subgroup security, which is a security notion regarding the parameter choice for pairing-based cryptography, proposed by Barreto et al. [3]. More precisely, we showed that there exists an attack against a scheme which uses a subgroup-secure elliptic curve and the light membership check in order to improve performance. We recommend developers to use the full membership check because it is a simple countermeasure and general technique to implement schemes securely.

For practical aspects, the performance of pairing-based cryptographic schemes should be improved. Subgroup security is an interesting security notion in order to improve performance. However, there might be an attack. If the developers use the light membership check for performance improvement, the developers should confirm whether the security is preserved or not. The essential problem of subgroup security is *how to determine whether the full membership check can be replaced in a scheme with the light membership check without loss of security.* To the best of our knowledge, the solution to this problem has not been noticed yet. To approach this problem, it seems to us that security analyses regarding the hardness of DDH problem [16] and the circuit privacy [6] are needed. Furthermore, it also seems to be difficult for non-experts to use subgroup security in an appropriate way. Hence, (automatic or semi-automatic) analysis and determining methodology for subgroup security is demanded, and its investigation is our future work.

Recently, several researchers also proposed a carefully selected elliptic curves in elliptic curve cryptography for security and efficiency, for example, SafeCurves [4]. It is interesting to analyze their choice with respect to the hardness of DDH problem [16] and the circuit privacy [6], which, to the best of our knowledge, has not been done yet, and which we leave as future work.

Last but not the least, using the full membership check is a simple and general technique to implement pairing-based cryptographic schemes. Therefore, investigation of faster full membership check is also our future work.

ACKNOWLEDGMENTS

The author thanks the anonymous reviewers of APKC 2018 for their valuable comments and insightful suggestions. The author also thanks the members of Shin-Akarui-Angou-Benkyou-Kai for their valuable comments. A part of this work is supported by JST CREST grant number JPMJCR1688.

REFERENCES

[1] Adrian Antipa, Daniel R. L. Brown, Alfred Menezes, René Struik, and Scott A. Vanstone. 2003. Validation of Elliptic Curve Public Keys. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings (Lecture Notes in Computer Science)*, Yvo Desmedt (Ed.), Vol. 2567. Springer, 211–223. https://doi.org/10.1007/3-540-36288-6_16

[2] Diego F. Aranha, Paulo S. L. M. Barreto, Patrick Longa, and Jefferson E. Ricardini. 2013. The Realm of the Pairings. In *Selected Areas in Cryptography - SAC 2013*. 3–25. https://doi.org/10.1007/978-3-662-43414-7_1

[3] Paulo S. L. M. Barreto, Craig Costello, Rafael Misoczki, Michael Naehrig, Geovandro C. C. F. Pereira, and Gustavo Zanon. 2015. Subgroup Security in Pairing-Based Cryptography. In *Progress in Cryptology - LATINCRYPT 2015*. 245–265. https://doi.org/10.1007/978-3-319-22174-8_14

[4] Daniel J. Bernstein and Tanja Lange. 2014. SafeCurves: choosing safe curves for elliptic-curve cryptography. (December 2014). https://safecurves.cr.yp.to/ Accessed January 11, 2018.

[5] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *Public Key Cryptography - PKC 2003*. 31–46. https://doi.org/10.1007/3-540-36288-6_3

[6] Dario Catalano and Dario Fiore. 2015. Using Linearly-Homomorphic Encryption to Evaluate Degree-2 Functions on Encrypted Data. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*. 1518–1529. https://doi.org/10.1145/2810103.2813624

[7] David Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* 24, 2 (1981), 84–88. https://doi.org/10.1145/358549.358563

[8] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory* 31, 4 (1985), 469–472. https://doi.org/10.1109/TIT.1985.1057074

[9] David Freeman, Michael Scott, and Edlyn Teske. 2010. A Taxonomy of Pairing-Friendly Elliptic Curves. *J. Cryptology* 23, 2 (2010), 224–280. https://doi.org/10.1007/s00145-009-9048-z

[10] Laura Fuentes-Castañeda, Edward Knapp, and Francisco Rodríguez-Henríquez. 2011. Faster Hashing to $\mathbb{G}_2$. In *Selected Areas in Cryptography - SAC 2011*. 412–430. https://doi.org/10.1007/978-3-642-28496-0_25

[11] Daniel Genkin, Luke Valenta, and Yuval Yarom. 2017. May the Fourth Be With You: A Microarchitectural Side Channel Attack on Several Real-World Applications of Curve25519. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*. 845–858. https://doi.org/10.1145/3133956.3134029

[12] Florian Hess. 2008. Pairing Lattices. In *Pairing-Based Cryptography - Pairing 2008*. 18–38. https://doi.org/10.1007/978-3-540-85538-5_2

[13] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. 2006. The Eta Pairing Revisited. *IEEE Trans. Information Theory* 52, 10 (2006), 4595–4602. https://doi.org/10.1109/TIT.2006.881709

[14] Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. 2015. Practical Invalid Curve Attacks on TLS-ECDH. In *Computer Security - ESORICS 2015*. 407–425. https://doi.org/10.1007/978-3-319-24174-6_21

[15] A. Kato, M. Scott, T. Kobayashi, and Y. Kawahara. 2016. Barreto-Naehrig Curves draft-kasamatsu-bncurves-02. (March 2016). https://tools.ietf.org/html/draft-kasamatsu-bncurves-02

[16] Jonathan Katz and Yehuda Lindell. 2015. *Introduction to Modern Cryptography* (2nd ed.). Chapman & Hall/CRC.

[17] Chae Hoon Lim and Pil Joong Lee. 1997. A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroupp. In *Advances in Cryptology - CRYPTO 1997*. 249–263. https://doi.org/10.1007/BFb0052240

[18] Nadia El Mrabet and Marc Joye (Eds.). 2016. *Guide to Pairing-Based Cryptography*. Chapman and Hall/CRC.

[19] Samuel Neves and Mehdi Tibouchi. 2016. Degenerate Curve Attacks - Extending Invalid Curve Attacks to Edwards Curves and Other Models. In *Public-Key Cryptography - PKC 2016*. 19–35. https://doi.org/10.1007/978-3-662-49387-8_2

[20] Eric Zavattoni, Luis J. Dominguez Perez, Shigeo Mitsunari, Ana H. Sánchez-Ramírez, Tadanori Teruya, and Francisco Rodríguez-Henríquez. 2015. Software Implementation of an Attribute-Based Encryption Scheme. *IEEE Trans. Computers* 64, 5 (2015), 1429–1441. https://doi.org/10.1109/TC.2014.2329681