An extended abstract of this paper appears in the Proceedings of the 14th Information Security Conference (ISC 2011). This version is the full paper.

The n-Diffie-Hellman Problem and its Applications

Liqun Chen* and Yu Chen†,‡

* Hewlett-Packard Laboratories, Bristol, UK
liqun.chen@hp.com
† School of Computer Science, Peking University, Beijing, China
‡ Institute of Information Engineering, Chinese Academy of Sciences
cycosmic@gmail.com

Abstract. The main contributions of this paper are twofold. On the one hand, the twin Diffie-Hellman (twin DH) problem proposed by Cash, Kiltz and Shoup is extended to the n-Diffie-Hellman (n-DH) problem for an arbitrary integer n, and this new problem is shown to be at least as hard as the ordinary DH problem. Like the twin DH problem, the n-DH problem remains hard even in the presence of a decision oracle that recognizes solution to the problem. On the other hand, observe that the double-size key in the Cash et al. twin DH based encryption scheme can be replaced by two separated keys each for one entity, that results in a 2-party encryption scheme which holds the same security feature as the original scheme but removes the key redundancy. This idea is further extended to an n-party case, which is also known as n-out-of-n encryption. As examples, a variant of ElGamal encryption and a variant of Boneh-Franklin IBE have been presented; both of them have proved to be CCA secure under the computational DH assumption and the computational bilinear Diffie-Hellman (BDH) assumption respectively, in the random oracle model. The two schemes are efficient, due partially to the size of their ciphertext, which is independent to the value n.

Keywords: the (strong) n-DH assumption, the (strong) n-BDH assumption, multiple public key encryption, multiple identity-based encryption.

1 Introduction

In EUROCRYPT 2008 [6], Cash, Kiltz and Shoup proposed a new computational problem and named it the *twin Diffie-Hellman* (twin DH) problem with the meaning that given a random triple of the form $(X_1, X_2, Y) \in \mathbb{G}^3$ for a cyclic group $\mathbb{G}$, compute $\mathrm{dh}(X_1, Y)$ and $\mathrm{dh}(X_2, Y)$, where dh is the DH function. They also proposed the *strong twin DH* problem, which is the twin DH problem under

the condition that an adversary is given access to a corresponding decision twin DH oracle. They proved that the strong twin DH problem is as hard as the (ordinary) DH problem, i.e., given a random pair of the form $(X, Y) \in \mathbb{G}^2$, compute $\mathrm{dh}(X, Y)$.

The motivation of their introducing the (strong) twin DH problem is the following: it is well-known that there exist many cryptographic constructions (e.g., the Diffie-Hellman non-interactive key exchange protocol [16] and the Cramer-Shoup encryption scheme [12]) which are based on the DH problem, but security of these constructions can only be proved under the strong DH problem, i.e., the adversary is given access to a decision DH oracle. The reason is that in the security proof, the simulator need the help of the decision oracle to keep the simulation coherent throughout the game. By employing the strong twin DH problem in these constructions, they can successfully prove that the modified constructions are secure under the DH problem, since the strong twin DH problem implies the DH problem. This is a clever trick.

However, their method is not cost free. In order to employ the twin DH problem, their modified construction is "a bit less efficient" than the original one; specifically, the modified construction doubles the key of the original one. For example, in their twin Identity-Based Encryption (IBE) scheme, a master key of a Key Generation Center (KGC) is twin private/public key pairs, written as $((x_1, X_1), (x_2, X_2))$, instead of one (x, X) in the original IBE scheme, and accordingly, an user's secret key associated with this user's identity id (served as a public key of the user) is also two secret values written as (S_1, S_2), each of which is computed under one master key pair. Therefore, a key redundancy is the cost of tighter security reduction.

Can we use this key redundancy to achieve some extra useful function without imposing an efficiency penalty? Observe that in their twin IBE scheme, the identity value id in computing S_1 does not have to be the same as in computing S_2; the two private/public master key pairs (x_1, X_1) and (x_2, X_2) can each belong to an individual KGC. With this slight modification, a user can have two independent identities each associated with one KGC. For example, Alice has her working email address associated with her employer as one KGC and her passport number associated with the government of her country as another KGC. These two KGCs are independent authorities, and do not necessarily have any trust relation or communication between them. Furthermore, the number of the identities and KGCs in the IBE scheme does not have to be restricted to two[1].

This observation leads to the main contributions of our paper that the twin DH problem can be extended to the n-DH problem for an arbitrary number n, which enables us to build an efficient encryption scheme with multiple public keys and an efficient IBE scheme with multiple KGCs and identities. This type of encryption is also known as n-out-of-n encryption, in which a given

[1] The multi-KGC IBE is not an unsolved problem and could be implemented from extending an existing IBE scheme, but we want to show how we can do it *efficiently* using n-out-of-n encryption.

message is encrypted under a set of n individual public keys, and the associated decryption operation makes use of the n corresponding secret keys. It is relevant to other well-known encryption primitives with multi-receivers, such as broadcast encryption [5, 15] (known as 1-out-of-n encryption) and threshold cryptosystem [14] (known as t-out-of-n encryption). The latter has an attractive application, namely attribute-based encryption (ABE) [3, 19]. Compared with the well-explored t-out-of-n threshold encryption or ABE schemes, e.g. using a secret sharing technique [23], an n-out-of-n encryption scheme seems a naive solution. But we think it is worthy studying this type of schemes properly since it has the advantage of simplicity in both algorithm implementation and security analysis.

More specifically, there are a number of contributions in this paper. Here we describe a brief overview of each contribution individually.

The n-DH problem. We present a modification of the twin DH problem [6] by extending the number of the (ordinary) DH instances from 2 to an arbitrary integer n, and name it the *n-DH problem*. Intuitively, the n-DH problem is that given a random $n+1$ tuple of the form $(X_1, \ldots, X_n, Y) \in \mathbb{G}^{n+1}$ for a cyclic group $\mathbb{G}$, compute $(\mathrm{dh}(X_1, Y), \ldots, \mathrm{dh}(X_n, Y))$ where dh is the DH function. We also present the *strong n-DH problem* which is the n-DH problem under the condition that an adversary is given access to a corresponding decision n-DH oracle. We prove that the strong n-DH problem is just as hard as the DH problem.

The n-BDH problem. We present a modification of the twin Bilinear-DH (twin BDH) problem [6, 11]. by extending the number of the (ordinary) BDH instances from 2 to an arbitrary integer n, and name it the *n-BDH problem*. Intuitively, the n-BDH problem is that given a random $2n+1$ tuple of the form $(X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n) \in \mathbb{G}^{2n+1}$ for a cyclic group $\mathbb{G}$, compute $(\mathrm{bdh}(X_1, Y, Z_1), \ldots, \mathrm{bdh}(X_n, Y, Z_n))$ where bdh is the BDH function. We also present the *strong n-BDH problem* which is the n-BDH problem under the condition that an adversary is given access to a corresponding decision n-BDH oracle. We prove that the strong n-BDH problem is just as hard as the BDH problem.

Formalized description of a MPKE scheme. We formalize the concept of an n-out-of-n public key encryption scheme and call it a Multiple Public Key Encryption (MPKE) scheme. We present a formal definition of such a MPKE scheme and a security model with the meaning that a MPKE scheme is secure against adaptive chosen ciphertext attacks. This definition is a simple modification of the usual definition of chosen ciphertext security for a public key encryption scheme [21]. MPKE schemes can be used in those applications, which requires that either a decryptor must be in the possession of n private keys (e.g., each can be bound with an particular attribute) or that n decryptors (each with an individual key) must work together, in order to decrypt a given ciphertext.

Formalized description of a MIBE scheme. We formalize the concept of a Multiple Identity-Based Encryption (MIBE) scheme, in which one encrypts a given message under a set of n (an arbitrary integer) individual public keys, where, unlike a MPKE scheme, each public key is presented as an identity of

someone who holds the corresponding private key. The decryption operation makes use of the n associated secret keys, each of which is generated by a KGC (the n KGCs can be independent to each other). We give a formal definition of such a MIBE scheme and a security model with the meaning that a MIBE scheme is secure against adaptive chosen ciphertext attacks. This definition is a simple modification of the usual definition of chosen ciphertext security for an identity-based encryption scheme [4]. This type of IBE schemes has already been introduced in the literature, e.g. [7, 9, 10]. To the best of our knowledge, the security of the schemes in [7, 9, 10] have not been rigorously analyzed.

A CONCRETE MPKE SCHEME AND A RIGOROUS SECURITY ANALYSIS. We propose a new modification of the hashed ElGamal encryption scheme [1], and name it the n-ElGamal encryption scheme. Here is a summary of this scheme. Let H be a hash function and $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be a symmetric cipher. Let $(X_1, \ldots, X_n)$ be n random group elements served as n public keys and $(x_1, \ldots, x_n)$ be n random integers served as n secret keys, where $X_i = g^{x_i}$ for $i = 1, \ldots, n$ and g is a group generator. To encrypt a message m, one chooses a random integer y and computes

$$Y := g^y, Z_i := X_i^y \text{ for each } i,\ k := \mathsf{H}(Y, Z_1, \ldots, Z_n),\ c := \mathsf{E}(k, m).$$

To decrypt the ciphertext (Y, c), one computes

$$Z_i := Y^{x_i} \text{ for each } i,\ k := \mathsf{H}(Y, Z_1, \ldots, Z_n),\ m := \mathsf{D}(k, c).$$

Observe that the ciphertext for this scheme is extremely compact, and its size is independent to the number n. Based on the strong n-DH assumption (that implies based on the ordinary DH assumption), we prove that the n-ElGamal encryption scheme has chosen ciphertext security, provided that SE is secure against chosen ciphertext attacks and that H is modeled as a random oracle [2].

A CONCRETE MIBE SCHEME AND A RIGOROUS SECURITY ANALYSIS. We propose a new modification of the Boneh-Franklin IBE scheme [4] and name it the n-IBE scheme. The following is a summary of this scheme. Let H and G be two hash functions, $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ be a symmetric cipher, and e be a bilinear map function. Let $(X_1, \ldots, X_n)$ be n random group elements served as n master public keys, and $(x_1, \ldots, x_n)$ be n random integers served as n master secret keys, where $X_i = g^{x_i}$ for each i and g is a group generator. Let $(id_1, \ldots, id_n)$ be n arbitrary data strings served as n identities, and $(S_1, \ldots, S_n)$ be n group elements served as n secret keys, where $S_i = \mathsf{G}(X_i, id_i)^{x_i}$ for each i. To encrypt a message m, one chooses a random integer y and computes

$$Y := g^y, W_i := e(\mathsf{G}(X_i, id_i), X_i)^y \text{ for each } i,$$

$$k := \mathsf{H}(id_1, \ldots, id_n, Y, W_1, \ldots, W_n),\ c := \mathsf{E}(k, m).$$

To decrypt the ciphertext (Y, c), one computes

$$W_i := e(S_i, Y) \text{ for each } i,\ k := \mathsf{H}(id_1, \ldots, id_n, Y, W_1, \ldots, W_n),\ m := \mathsf{D}(k, c).$$

Note that the length of the ciphertext in this scheme is also very short, and it is independent to the value n. Based on the strong n-BDH assumption (that implies based on the ordinary BDH assumption), we prove that the n-IBE scheme has chosen ciphertext security, provided that SE is secure against chosen ciphertext attacks and that G and H are modeled as random oracles.

The rest of this paper is organized as follows. We describe definitions of the (strong) n-BDH assumption in Section 2 and of the (strong) n-BDH assumption in Section 3. After that, we present definitions of security models for MPKE schemes and MIBE schemes in Section 4, followed by a concrete MPKE scheme with a rigorous security analysis in Section 5, and a concrete MIBE scheme with a rigorous security analysis in Section 6. We end the paper with conclusions and some open questions for future work in Section 7.

2 The n-DH Assumption

Let $\mathbb{G}$ be a cyclic group of prime order p and with generator g, and let dh be the DH function defined as

$$\mathrm{dh}(X, Y) := Z, \text{ where } X = g^x, Y = g^y \text{ and } Z = g^{xy}.$$

Recall that the DH assumption states it is hard to compute $\mathrm{dh}(X, Y)$ given random $X, Y \in \mathbb{G}$. We define the function

$$\begin{aligned} \mathrm{ndh} : \mathbb{G}^{n+1} &\to \mathbb{G}^n \\ (X_1, \ldots, X_n, Y) &\mapsto (\mathrm{dh}(X_1, Y), \ldots, \mathrm{dh}(X_n, Y)), \end{aligned}$$

and call this the *n-DH function*. We also define a corresponding *n-DH predicate* by

$$\mathrm{ndhp}(X_1, \ldots, X_n, \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n) := \mathrm{ndh}(X_1, \ldots, X_n, \hat{Y}) \stackrel{?}{=} (\hat{Z}_1, \ldots, \hat{Z}_n).$$

The n-DH assumption states that it is hard to compute $\mathrm{ndh}(X_1, \ldots, X_n, Y)$ given random $X_1, \ldots, X_n, Y \in \mathbb{G}$. Accordingly, the *strong n-DH assumption* states that it is hard to compute $\mathrm{ndh}(X_1, \ldots, X_n, Y)$ given random $X_1, \ldots, X_n, Y \in \mathbb{G}$ along with access to the predicate $\mathrm{ndhp}(X_1, \ldots, X_n, \cdot, \cdot, \ldots, \cdot)$, which returns $\mathrm{ndhp}(X_1, \ldots, X_n, \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n)$ on input $(\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n)$. We have the following theorem to address the relation between the DH assumption and the (strong) n-DH assumption:

Theorem 2.1 (DH via strong n-DH) *The (ordinary) DH assumption holds if and only if the strong n-DH assumption holds.*

It is clear that the DH assumption implies the n-DH assumption. We now prove that the DH assumption implies the strong n-DH assumption. To do this, by following the trapdoor test technique of [6], we first create a trapdoor test.

Theorem 2.2 (Trapdoor Test for n-DH) *Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. Let $I = \{2, \ldots, n\}$, and suppose X_1, r_i, s_i for all $i \in I$ are mutually independent random variables, where X_1 is randomly taken in $\mathbb{G}$, and each of r_i and s_i is uniformly distributed over $\mathbb{Z}_p$, and define the random variables $X_i := g^{s_i}/X_1^{r_i}$. Further suppose that $\hat{Y}, \hat{Z}_1, \cdots, \hat{Z}_n$ are random variables taking values in $\mathbb{G}$, each of which is defined as some function of X_i for all $i \in \{1\} \cup I$. Then we have:*

1. *Each X_i for $i \in I$ is uniformly distributed over $\mathbb{G}$;*
2. *All X_i for $i \in \{1\} \cup I$ are mutually independent;*
3. *If $X_i = g^{x_i}$ for $i \in \{1\} \cup I$, then the probability that the truth value of*

$$\hat{Z}_1^{\,r_2}\hat{Z}_2 = \hat{Y}^{s_2} \wedge \cdots \wedge \hat{Z}_1^{\,r_i}\hat{Z}_i = \hat{Y}^{s_i} \wedge \cdots \wedge \hat{Z}_1^{\,r_n}\hat{Z}_n = \hat{Y}^{s_n} \tag{1}$$

does not agree with the truth value of

$$\hat{Z}_1 = \hat{Y}^{x_1} \wedge \cdots \wedge \hat{Z}_i = \hat{Y}^{x_i} \wedge \cdots \wedge \hat{Z}_n = \hat{Y}^{x_n} \tag{2}$$

is at most $(1/p)^{n-1}$; moreover if (2) holds, then (1) certainly holds.

Proof. Observe that $s_i = r_i x_1 + x_i$ for $i \in I$ where $I = \{2, \ldots, n\}$. It is not difficult to verify that each X_i for $i \in I$ is uniformly distributed over $\mathbb{G}$, and that all X_i for $i \in \{1\} \cup I$ and r_i for $i \in I$ are mutually independent, from which the items 1 and 2 follow. To prove the item 3, condition on fixed values of X_i for $i \in \{1\} \cup I$. In the resulting conditional probability space, each r_i for $i \in I$ is uniformly distributed over $\mathbb{Z}_p$, while all x_i, $\hat{Y}$, $\hat{Z}_i$ for $i \in \{1\} \cup I$ are fixed. If (2) holds, (1) certainly holds, because $s_i = r_i x_1 + x_i$ for $i \in I$. Conversely, if (2) does not hold, we show that (1) holds with probability at most $(1/p)^{n-1}$. We take the $n-1$ equations of (1) separately. Each of them uses the same argument as in the proof of the trapdoor test of [6]. Observe that (1) is equivalent to

$$(\hat{Z}_1/\hat{Y}^{x_1})^{r_2} = \hat{Y}^{x_2}/\hat{Z}_2 \wedge \cdots \wedge (\hat{Z}_1/\hat{Y}^{x_1})^{r_i} = \hat{Y}^{x_i}/\hat{Z}_i \wedge \cdots \wedge (\hat{Z}_1/\hat{Y}^{x_1})^{r_n} = \hat{Y}^{x_n}/\hat{Z}_n. \tag{3}$$

Let us take a look at the $(i-1)^{th}$ equation of (3). We can see that if $\hat{Z}_1 = \hat{Y}^{x_1}$ and $\hat{Z}_i \neq \hat{Y}^{x_i}$ no matter whether the other equations of (2) holds or not, then this equation certainly does not hold. This leaves us with the case $\hat{Z}_1 \neq \hat{Y}^{x_1}$. In this case, the left hand side of the equation is a random element of $\mathbb{G}$ (since r_i is uniformly distributed over $\mathbb{Z}_p$), but the right hand side is a fixed element of $\mathbb{G}$. So this equation holds with probability $1/p$. (3) holds if and only if $n-1$ different equations all hold. Now, we argue that these $n-1$ equations are mutually independent, because each r_i for $i \in I$ is uniformly distributed over $\mathbb{Z}_p$, therefore, the probability that (3) holds is at most $(1/p)^{n-1}$. □

Using this trapdoor test as a tool, we can prove Theorem 2.1. Let $\mathcal{B}$ be a DH adversary. Denote its advantage by $\mathsf{AdvDH}_{\mathcal{B},\mathbb{G}}$ with the meaning of the probability that $\mathcal{B}$ computes $\mathrm{dh}(X, Y)$, given random $X, Y \in \mathbb{G}$. Let $\mathcal{A}$ be a strong n-DH adversary. Denote its advantage by $\mathsf{AdvnDH}_{\mathcal{A},\mathbb{G}}$ with the meaning of the

probability that $\mathcal{A}$ computes $\mathrm{ndh}(X_1, \ldots, X_n, Y)$, given random $X_i, Y \in \mathbb{G}$ for $i \in \{1, \ldots, n\}$, along with access to the predicate $\mathrm{ndhp}(X_1, \ldots, X_n, \cdot, \cdot, \ldots, \cdot)$, which on input $(\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n)$, returns $\mathrm{ndhp}(X_1, \ldots, X_n, \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n)$. Theorem 2.1 is a special case of the following:

Theorem 2.3 *Suppose $\mathcal{A}$ is a strong n-DH adversary that makes at most Q_d queries to its decision oracle, and runs in time at most τ. Then there exists a DH adversary $\mathcal{B}$ with the following properties: $\mathcal{B}$ runs in time at most τ, plus the time to perform $O(Q_d \log q)$ group operations and some minor bookkeeping; moreover,*

$$\left(1 - \frac{Q_d}{p^{n-1}}\right) \mathsf{AdvnDH}_{\mathcal{A},\mathbb{G}} \leq \mathsf{AdvDH}_{\mathcal{B},\mathbb{G}}.$$

In addition, if $\mathcal{B}$ does not output "failure", then its output is correct with probability at least $1 - Q_d/p^{n-1}$.

Proof. The DH adversary $\mathcal{B}$ works as follows, given a challenge instance (X, Y) of the DH problem. First, $\mathcal{B}$ chooses r_i, $s_i \in \mathbb{Z}_p$ for $i \in I$ and $I = \{2, ..., n\}$ at random, sets $X_1 := X$ and $X_i := g^s / X_1^{r_i}$, and gives $\mathcal{A}$ the challenge instance $(X_1, \ldots, X_n, Y)$. Second, $\mathcal{B}$ processes each decision query $(\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n)$ by testing if

$$\hat{Z}_1^{\,r_2} \hat{Z}_2 = \hat{Y}^{s_2} \wedge \cdots \wedge \hat{Z}_1^{\,r_i} \hat{Z}_i = \hat{Y}^{s_i} \wedge \cdots \wedge \hat{Z}_1^{\,r_n} \hat{Z}_n = \hat{Y}^{s_n}$$

holds. Finally, if and when $\mathcal{A}$ outputs $(Z_1, \ldots, Z_n)$, $\mathcal{B}$ tests if this output is correct by testing whether

$$Z_1^{r_2} Z_2 = Y^{s_2} \wedge \cdots \wedge Z_1^{r_i} Z_i = Y^{s_i} \wedge \cdots \wedge Z_1^{r_n} Z_n = Y^{s_n}$$

holds; if this does not hold, then $\mathcal{B}$ outputs "failure", and otherwise, $\mathcal{B}$ outputs Z_1.

Provide the oracle simulation is perfect, adversary $\mathcal{A}$'s view is identical to its view in the real environment. It remains to calculate the accuracy of the trapdoor test. Note that the probability of the trapdoor test returning a wrong decision result for a query is at most $(1/p)^{n-1}$, and this happens at most Q_d times. Therefore the trapdoor test can simulate the decision oracle perfectly with probability at least $1 - Q_d/p^{n-1}$. Theorem 2.3 follows immediately. □

3 The n-BDH Assumption

In groups equipped with a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p and $\mathbb{G}$ is with generate g, we recall that the BDH function is defined as

$$\mathrm{bdh}(X, Y, Z) := W, \text{ where } X = g^x, Y = g^y, Z = g^z, \text{ and } W = e(g, g)^{xyz}.$$

The BDH assumption states that computing $\mathrm{bdh}(X, Y, Z)$ for random $X, Y, Z \in \mathbb{G}$ is a hard problem. The strong BDH assumption [20] states that the BDH problem remains hard even with the help of a corresponding decision oracle.

Note that for the purpose of describing our main results as simply as possible, without loss of the generality, we make use of symmetric pairings (also called Type-1 pairings). It does not mean that our proposed assumptions and schemes only work with symmetric pairings. Without changing the main results of this paper, this symmetric pairing representation can be modified to the asymmetric pairing one (i.e., $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p). More specifically, one may use Type-2 pairings, where there is an efficiently computable group isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ mapping $g_2 \in \mathbb{G}_2$ to $g_1 \in \mathbb{G}_1$, or Type-3 pairings, where there is no known efficiently computable group isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_2$ mapping g_2 to g_1. We refer readers to [18] for the details of these three types of pairings.

We define the function

$$\begin{aligned} \mathrm{nbdh} : \mathbb{G}^{2n+1} &\to \mathbb{G}_T^n \\ (X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n) &\mapsto (\mathrm{bdh}(X_1, Y, Z_1), \ldots, \mathrm{bdh}(X_n, Y, Z_n)), \end{aligned}$$

and call this the *n-BDH function*. We also define a corresponding *n-BDH predicate* by

$$\begin{aligned} &\mathrm{nbdhp}(X_1, \ldots, X_n, \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n, \hat{W}_1, \ldots, \hat{W}_n) := \\ &\mathrm{nbdh}(X_1, \ldots, X_n, \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n) \stackrel{?}{=} (\hat{W}_1, \ldots, \hat{W}_n). \end{aligned}$$

The *n-BDH assumption* states that it is hard to compute $\mathrm{nbdh}(X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n)$ given random $X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n \in \mathbb{G}$. The *strong n-BDH assumption* states that it is hard to compute $\mathrm{nbdh}(X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n)$, given random $X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n \in \mathbb{G}$, along with the access to the predicate $\mathrm{nbdh}(X_1, \ldots, X_n, \cdot, \cdot, \ldots, \cdot, \cdot, \ldots, \cdot)$, which on input $(\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n, \hat{W}_1, \ldots, \hat{W}_n)$, returns $\mathrm{nbdhp}(X_1, \ldots, X_n, \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n, \hat{W}_1, \ldots, \hat{W}_n)$.

We have the following result to address the relation between the BDH assumption and the (strong) n-BDH assumption:

Theorem 3.1 (BDH via strong n-BDH) *The (ordinary) BDH assumption holds if and only if the strong n-BDH assumption holds.*

It is clear that the BDH assumption implies the n-BDH assumption. We prove that the BDH assumption implies the strong n-BDH assumption. Again, by following the technique developed in [6], we first create a trapdoor test.

Theorem 3.2 (Trapdoor Test for n-BDH) *Let $\mathbb{G}$ be a cyclic group of prime order p with a generator g and a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}_T$ is another cyclic group of order p. Let $I = \{2, \ldots, n\}$, and suppose X_1, r_i, s_i for $i \in I$ are all mutually independent random variables, where X_1 is randomly taken in $\mathbb{G}$, and each of r_i and s_i is uniformly distributed over $\mathbb{Z}_p$, and define the random variables $X_i := g^{s_i}/X_1^{r_i}$ for $i \in I$. Further suppose that $(\hat{Y}_1, \ldots, \hat{Y}_n, \hat{Z}, \hat{W}_i, \ldots, \hat{W}_n)$ are random variables taking values in $\mathbb{G}$, each of which is defined as some function of X_i for all $i \in \{1\} \cup I$. Then we have:*

1. *Each X_i for $i \in I$ is uniformly distributed over $\mathbb{G}$;*
2. *All X_i for $i \in \{1\} \cup I$ are mutually independent;*
3. *If $X_i = g^{x_i}$ for $i \in \{1\} \cup I$, the probability that the truth value of*

$$\hat{W_1}^{r_2}\hat{W_2} = e(\hat{Y_2}, \hat{Z})^{s_2} \wedge \cdots \wedge \hat{W_1}^{r_i}\hat{W_i} = e(\hat{Y_i}, \hat{Z})^{s_i} \wedge \cdots \wedge \hat{W_1}^{r_n}\hat{W_n} = e(\hat{Y_n}, \hat{Z})^{s_n} \tag{4}$$

does not agree with the truth value of

$$\hat{W_1} = e(\hat{Y_1}, \hat{Z})^{x_1} \wedge \cdots \wedge \hat{W_i} = e(\hat{Y_i}, \hat{Z})^{x_i} \wedge \cdots \wedge \hat{W_n} = e(\hat{Y_n}, \hat{Z})^{x_n} \tag{5}$$

is at most $(1/p)^{n-1}$; moreover if (5) holds, then (4) certainly holds.

Proof. Observe that $s_i = r_i x_1 + x_i$ for $i \in I$ where $I = \{2, \ldots, n\}$. It is not difficult to verify that each X_i for $i \in I$ is uniformly distributed over $\mathbb{G}$, and that all X_i for $i \in \{1\} \cup I$ and r_i for $i \in I$ are mutually independent, from which the items 1 and 2 follow. To prove the item 3, condition on fixed values of X_i for $i \in \{1\} \cup I$. In the resulting conditional probability space, each r_i for $i \in I$ is uniformly distributed over $\mathbb{Z}_p$, while all x_i, $\hat{Y_i}$, $\hat{Z}$ and $\hat{W_i}$ for $i \in \{1\} \cup I$ are fixed. If (5) holds, (4) certainly holds, because $s_i = r_i x_1 + x_i$ for $i \in I$. Conversely, if (5) does not hold, we show that (4) holds with probability at most $(1/p)^{n-1}$. We take the $n-1$ equations of (4) separately. Each of them uses the same argument as in the proof of the trapdoor test [6]. Observe (4) is equivalent to

$$(\hat{W_1}/e(\hat{Y_1}, \hat{Z})^{x_1})^{r_2} = e(\hat{Y_2}, \hat{Z})^{x_2}/\hat{W_2} \wedge \cdots \wedge (\hat{W_1}/e(\hat{Y_1}, \hat{Z})^{x_1})^{r_n} = e(\hat{Y_n}, \hat{Z})^{x_n}/\hat{W_n}. \tag{6}$$

Let us take a look at the $(i-1)^{th}$ equation of (6). We can see that if $\hat{W_1} = e(\hat{Y_1}, \hat{Z})^{x_1}$ and $\hat{W_i} \neq e(\hat{Y_i}, \hat{Z})^{x_i}$ no matter whether the other equations of (5) holds or not, then this equation certainly does not hold. This leaves us with the case $\hat{W_1} \neq e(\hat{Y_1}, \hat{Z})^{x_1}$. In this case, the left hand side of the equation is a random element of $\mathbb{G}_T$ (since r_i is uniformly distributed over $\mathbb{Z}_p$), but the right hand side is a fixed element of $\mathbb{G}_T$. So this equation holds with probability $1/p$. (6) holds if and only if $n-1$ different equations all hold. Now, we argue that these $n-1$ equations are mutually independent, because each r_i for $i \in I$ is uniformly distributed over $\mathbb{Z}_p$, therefore, the probability that (6) holds is at most $(1/p)^{n-1}$. □

Using this trapdoor test as a tool, we can prove Theorem 3.1. Let $\mathcal{B}$ be a BDH adversary. Denote its BDH advantage by $\mathsf{AdvBDH}_{\mathcal{B},\mathbb{G}}$ with the meaning of the probability that $\mathcal{B}$ computes $\mathrm{bdh}(X, Y, Z)$, given random $X, Y, Z \in \mathbb{G}$. Let $\mathcal{A}$ be a strong nbdh adversary. Denote its advantage by $\mathsf{AdvnBDH}_{\mathcal{A},\mathbb{G}}$ with the meaning of the probability that $\mathcal{A}$ computes $\mathrm{ndh}(X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n)$, given random $X_i, Y, Z_i \in \mathbb{G}$ for $i \in \{1, \ldots, n\}$, along with access to a decision oracle for the predicate $\mathrm{nbdhp}(X_1, \ldots, X_n, \cdot, \cdot, \ldots, \cdot, \cdot, \ldots, \cdot)$, which on input $(\hat{Y}, \hat{Z_1}, \ldots, \hat{Z_n}, \hat{W_1}, \ldots, \hat{W_n})$, returns $\mathrm{nbdhp}(X_1, \ldots, X_n, \hat{Y}, \hat{Z_1}, \ldots, \hat{Z_n}, \hat{W_1}, \ldots, \hat{W_n})$. Theorem 3.1 is a special case of the following:

Theorem 3.3 *Suppose $\mathcal{A}$ is a strong n-BDH adversary that makes at most Q_d queries to its decision oracle, and runs in time at most τ. Then there exists a BDH adversary $\mathcal{B}$ with the following properties: $\mathcal{B}$ runs in time at most τ, plus the time to perform $O(Q_d \log q)$ group operations and some minor bookkeeping; moreover,*

$$\left(1 - \frac{Q_d}{p^{n-1}}\right) \mathsf{AdvnBDH}_{\mathcal{A},\mathbb{G}} \leq \mathsf{AdvBDH}_{\mathcal{B},\mathbb{G}}.$$

In addition, if $\mathcal{B}$ does not output "failure", then its output is correct with probability at least $1 - Q_d/p^{n-1}$.

Proof. The BDH adversary $\mathcal{B}$ works as follows, given a challenge instance (X, Y, Z) of the BDH problem. First, $\mathcal{B}$ chooses r_i, s_i, $t_i \in \mathbb{Z}_q$ for $i \in I$ and $I = \{2, \ldots, n\}$ at random, sets $X_1 := X$ and $X_i := g^s / X_1^{r_i}$, $Y = Y$ and $Z_i = Z^{t_i}$, and gives $\mathcal{A}$ the challenge instance $(X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n)$. Second, $\mathcal{B}$ processes each decision query $(\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n, \hat{W}_1, \ldots, \hat{W}_n)$ by testing if

$$\hat{W}_1^{r_2} \hat{W}_2 = e(\hat{Y}, \hat{Z}_2)^{s_2} \wedge \cdots \wedge \hat{W}_1^{r_i} \hat{W}_i = e(\hat{Y}, \hat{Z}_i)^{s_i} \wedge \cdots \wedge \hat{W}_1^{r_n} \hat{W}_n = e(\hat{Y}, \hat{Z}_n)^{s_n}$$

holds. Finally, if and when $\mathcal{A}$ outputs $(W_1, \ldots, W_n)$, $\mathcal{B}$ tests if this output is correct by testing if

$$W_1^{r_2} W_2 = e(Y, Z_2)^{s_2} \wedge \cdots \wedge W_1^{r_i} W_i = e(Y, Z_i)^{s_i} \wedge \cdots \wedge W_1^{r_n} W_n = e(Y, Z_n)^{s_n} \quad (7)$$

holds; if this does not hold, then $\mathcal{B}$ outputs "failure", and otherwise, $\mathcal{B}$ outputs W_1. The proof is completed using the trapdoor test in Theorem 3.2. □

4 Definitions of MPKE and MIBE

In this section we present formal definitions of a Multiple Public Key Encryption (MPKE) scheme and of a Multiple Indentity-Based Encryption (MIBE) scheme, including their security notion: chosen ciphertext security, which are based on the usual definitions of chosen ciphertext security for a public key encryption scheme [21] and an identity-based encryption scheme [4]. Recall that these two types of encryption schemes are n-out-of-n encryption schemes. In the security model an adversary is not allowed to corrupt any decryption key from the entirely n set of the keys.

4.1 Multiple Public Key Encryption

A Multiple Public Key Encryption scheme (say MPKE), with a security parameter 1^κ and associated system parameters params (include a description of a finite key space $\mathcal{K}$, a description of a finite message space $\mathcal{M}$, and a description of a finite ciphertext space $\mathcal{C}$), is specified by three algorithms: KeyGen, Encrypt, and Decrypt:

KeyGen: takes 1^κ and params as input, and generates a set n of public and secret key pairs, written as $(pk_i, sk_i) \in \mathcal{K}$ for $i = 1, \ldots, n$. We also denote the n public keys by $\mathbf{pk} = (pk_1, \ldots, pk_n)$ and the n secret keys by $\mathbf{sk} = (sk_1, \ldots, sk_n)$.

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

Decrypt: takes as input params, a ciphertext $C \in \mathcal{C}$ and $\mathbf{sk}$, and returns M.

These algorithms must satisfy the standard consistency constraint, namely when $(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa, \mathsf{params})$, then

$$\forall M \in \mathcal{M} : \mathsf{Decrypt}(\mathsf{params}, C, \mathbf{sk}) = M \text{ where } C = \mathsf{Encrypt}(\mathsf{params}, \mathbf{pk}, M).$$

Chosen ciphertext security of the scheme MPKE is defined by the following chosen ciphertext attack game, played between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$:

Setup. The challenger takes a security parameter 1^κ and associated params, and runs the KeyGen algorithm. It gives the resulting $\mathbf{pk}$ together with params to $\mathcal{A}$, and keeps the corresponding $\mathbf{sk}$ to itself.

Phase 1. $\mathcal{A}$ makes a number of decryption queries to the challenger, where the input to each query is a ciphertext, say $\hat{C}$. To answer such a query, the challenger decrypts $\hat{C}$ and sends the result to $\mathcal{A}$. These queries may be asked adaptively, that is, each query may depend on the replies to previous queries.

Challenge. Once the adversary decides that Phase 1 is over, it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. The challenger picks a random bit $\beta \in \{0, 1\}$, encrypts M_β, and sends the resulting ciphertext C^* as the challenge to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues more decryption queries as in Phase 1, but with the restriction that $\hat{C} \neq C^*$. These queries may be asked adaptively as in Phase 1.

Guess. Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$ and wins the game if $\beta = \beta'$.

We refer to such an adversary $\mathcal{A}$ as an IND-CCA adversary. We define adversary $\mathcal{A}$'s advantage over the scheme MPKE by $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{MPKE}}(\kappa) = \left|\Pr[\beta = \beta'] - \frac{1}{2}\right|$. The probability is over the random bits used by the challenger and the adversary.

Definition 4.1 *We say that a multiple public key encryption scheme* MPKE *is* IND-CCA *secure if for any probabilistic polynomial time* IND-CCA *adversary* $\mathcal{A}$ *the advantage* $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{MPKE}}(\kappa)$ *is negligible*[2].

When we analyze the scheme MPKE in the random oracle model, then hash functions are modeled as random oracles, and both the challenger and adversary are given access to the random oracles in the above attack game. In that case, we write $\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{MPKE}}(\kappa)$ for the corresponding advantage.

[2] We say that a function $f(\kappa)$ is negligible if for every $c > 0$ there exists a value κ_c such that $f(\kappa) < 1/\kappa^c$ for all $\kappa < \kappa_c$.

4.2 Multiple Identity-Based Encryption

A Multiple Identity-Based Encryption scheme, denoted by MIBE, is specified by four algorithms: Setup, Extract, Encrypt and Decrypt:

Setup: takes a security parameter 1^κ, and returns system parameters params and a set n of master public and secret key pairs, written as (mpk_i, msk_i) for $i = 1, \ldots, n$; without loss of generality, each key pair (mpk_i, msk_i) is associated with the i-th of a set n KGCs. We denote the n master public keys by $\mathbf{mpk} = (mpk_1, \ldots, mpk_n)$ and the n master secret keys by $\mathbf{msk} = (msk_1, \ldots, msk_n)$. The parameters params include a description of a finite message space $\mathcal{M}$, and a description of a finite ciphertext space $\mathcal{C}$.

Extract: takes as input params, a master key msk_i and an arbitrary identity $id_i \in \{0,1\}^*$ for $i \in \{1, \ldots, n\}$. It returns a secret key sk_i. By repeating the Extract algorithm n times with different i values, one can obtain $\mathbf{sk} = (sk_1, \ldots, sk_n)$ associated with $\mathbf{id} = (id_1, \ldots, id_n)$. Note that msk_i and id_i do not have to uniquely match to each other. Theoretically speaking, any arbitrary identity can bind with any master key, and therefore, the case $id_i = id_j$ for $i \neq j$ is allowed.

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

Decrypt: takes as input params, a ciphertext $C \in \mathcal{C}$ and $\mathbf{sk}$, and returns M.

These algorithms must satisfy the standard consistency constraint, namely when $(\mathbf{mpk}, \mathbf{msk}, \mathsf{params}) \leftarrow \mathsf{Setup}(1^\kappa)$ and $\mathbf{sk} \leftarrow \mathsf{Extract}(\mathsf{params}, \mathbf{msk}, \mathbf{id})$, then

$$\forall m \in \mathcal{M} : \mathsf{Decrypt}(\mathsf{params}, C, \mathbf{sk}) = M \text{ where } C = \mathsf{Encrypt}(\mathsf{params}, \mathbf{mpk}, \mathbf{id}, M).$$

Chosen ciphertext security of scheme MIBE is defined by the following chosen ciphertext attack game, played between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$:

Setup. The challenger takes the security parameter 1^κ and runs the Setup algorithm. It gives the adversary the resulting params and $\mathbf{mpk}$, and keeps the associated $\mathbf{msk}$ to itself.

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

- Extraction query $\langle i, \hat{id}_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $\hat{sk}_i$ associated with $\hat{id}_i$ and msk_i. It sends $\hat{sk}_i$ to $\mathcal{A}$.
- Decryption query $\langle \hat{\mathbf{id}}, \hat{C} \rangle$. The challenger responds by running algorithm Extract n times to generate the private key $\hat{\mathbf{sk}}$ corresponding to $\hat{\mathbf{id}}$. It then runs algorithm Decrypt to decrypt the ciphertext $\hat{C}$. It sends the resulting plaintext to $\mathcal{A}$.

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

Challenge. Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and a set of identities $\hat{\mathbf{id}}^*$ on which it wishes to be challenged. The only constraint is that each element id_i^* of $\hat{\mathbf{id}}^*$ did not appear in

any private key extraction query associated with msk_i in Phase 1. The challenger picks a random bit $\beta \in \{0,1\}$ and set $C^* = \mathsf{Encrypt}(\mathsf{params}, \mathbf{mpk}, \hat{\mathbf{id}}^*, M_\beta)$. It sends C^* as the challenge to the adversary.

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

- Extraction query $\langle i, \hat{id}_i \rangle$, where $\hat{id}_i \neq$ the i-th element of $\hat{\mathbf{id}}^*$. Challenger responds as in Phase 1.
- Decryption query $\langle \hat{\mathbf{id}}, \hat{C} \rangle \neq \langle \hat{\mathbf{id}}^*, C^* \rangle$. Challenger responds as in Phase 1.

These queries may be asked adaptively as in Phase 1.

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

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. We define $\mathcal{A}$'s advantage over the scheme MIBE by $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{MIBE}}(\kappa) = |\Pr[\beta = \beta'] - \frac{1}{2}|$. The probability is over the random bits used by the challenger and the adversary.

Definition 4.2 *We say that a Multiple IBE scheme* MIBE *is* IND-ID-CCA *secure if for any probabilistic polynomial time* IND-ID-CCA *adversary* $\mathcal{A}$ *the advantage* $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{MIBE}}(\kappa)$ *is negligible.*

When we analyze such a scheme MIBE in the random oracle model, we write $\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{MIBE}}(\kappa)$ for the corresponding advantage.

5 The n-ElGamal Encryption Scheme

In this section, we present details of the n-ElGamal encryption scheme. The scheme makes use of a hash function H and a symmetric cipher $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$. Let $\mathbb{G}$ be a cyclic group of prime order p and with generator g. A set of public keys for this scheme is denoted by a n-tuple of random group elements $\mathbf{pk} = (X_1, \ldots, X_n)$, with a set of corresponding secret keys denoted by $\mathbf{sk} = (x_1, \ldots, x_n)$, where $X_i = g^{x_i}$ for $i \in I$ and $I = (1, \ldots, n)$. To encrypt a message $m \in \mathcal{M}$, one chooses a random $y \in \mathbb{Z}_p$, and computes

$$Y := g^y, Z_i := X_i^y \text{ for } i \in I, k := \mathsf{H}(Y, Z_1, \ldots, Z_n), C := \mathsf{E}(k, M).$$

The ciphertext is (Y, c). Decryption works accordingly: given (Y, c) and secret key $\mathbf{sk}$, one computes

$$Z_i := Y^{x_i} \text{ for } i \in I, k := \mathsf{H}(Y, Z_1, \ldots, Z_n), M := \mathsf{D}(k, C).$$

As mentioned earlier, the size of the ciphertext in this scheme is independent to the number of public and secret keys n. Like the twin ElGamal encryption scheme [6], the scheme does not add redundancy in the ciphertext in order to achieve CCA security, as in the Fujisaki-Okamoto transformation [17].

Following the arguments in [1, 6, 13], we now show that the n-ElGamal encryption scheme is secure against chosen ciphertext attack, under the strong

n-DH assumption, and the assumption that SE is secure against chosen ciphertext attack, if H is modeled as a random oracle. By Theorem 2.1, the same holds under the (ordinary) DH assumption. Formally speaking, we denote the n-ElGamal encryption scheme $\mathsf{MPKE}_{\mathrm{ndh}}$, and analyze security of this scheme with the following theorem, under the security model previously defined in Section 4.1.

Theorem 5.1 *Suppose* H *is modeled as a random oracle,* SE *is secure against chosen ciphertext attack, and the DH assumption holds in* $\mathbb{G}$*. The* $\mathsf{MPKE}_{\mathrm{ndh}}$ *is secure against chosen ciphertext attack. In particular, suppose* $\mathcal{A}$ *is an adversary that carries out a chosen ciphertext attack against* $\mathsf{MPKE}_{\mathrm{ndh}}$ *in the random oracle model, and* $\mathcal{A}$ *runs in time* τ*, and makes at most* Q_h *hash queries and* Q_d *decryption queries. Then there exists an adversary* $\mathcal{B}_{\mathrm{dh}}$ *against the DH problem and an adversary* $\mathcal{B}_{\mathrm{sym}}$ *against the chosen ciphertext security of* SE*, such that both* $\mathcal{B}_{\mathrm{dh}}$ *and* $\mathcal{B}_{\mathrm{sym}}$ *run in time at most* τ*, plus the time to perform* $O((Q_h + Q_d)\log p)$ *group operations; moreover,*

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{MPKE}_{\mathrm{ndh}}} \leq \left(\frac{p^{n-1}}{p^{n-1}-Q_h}\right)\mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh}},\mathbb{G}} + \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}}.$$

Proof. We proceed with a sequence of games.

Game 0. Let Game 0 be the original chosen ciphertext attack game for a MPKE scheme as defined in Section 4.1, and let S_0 be the event that $\beta' = \beta$ in this game.

Setup: To start the game, the challenger generates the secret key set $\mathbf{sk} = (x_1, \ldots, x_n)$ and their corresponding public key set $\mathbf{pk} = (X_1, \ldots, X_n)$. The challenger gives $\mathbf{pk}$ to the adversary.

Hash oracle query $\langle \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n \rangle$: The challenger maintains a list of tuples $(Y, Z_1, \ldots, Z_n, k)$ as explained below. We refer to this list as the L list, which is initially empty and indexed by elements of $\mathbb{G}^{n+1}$. Whenever the adversary makes a query $\langle \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n \rangle$, if there is already a tuple on the L list indexed by it then the challenger responds with $L[\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n] = \hat{k}$. Otherwise, the challenger picks a random symmetric key $\hat{k}$, adds the tuple $\langle \hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n, \hat{k} \rangle$ to the L list and responds the adversary with $\hat{k}$.

Phase 1 - Decryption query $\langle \hat{Y}, \hat{C} \rangle$: The challenger answers the decryption queries using its secret key $\mathbf{sk}$. The challenger need to call the H query in this operation.

Challenge: Once the adversary decides that Phase 1 is over it outputs two messages M_0, M_1 on which it wishes to be challenged. The challenger chooses a random $y \in \mathbb{Z}_p$, sets $Y := g^y$, $Z_i = X_i^y$ for $i = 1, \ldots, n$, then fetches the symmetric key k by querying H with $\langle Y, Z_1, \ldots Z_n \rangle$, and computes $c := \mathsf{E}_k(M_\beta)$, and returns the ciphertext (Y, C) to $\mathcal{A}$.

Phase 2. The decryption queries in Phase 2 are processed just as in Phase 1.

Guess: The adversary $\mathcal{A}$ outputs its guess β' for β.

That finishes the description of Game 0. Despite the syntactic difference, it is clear that

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{MPKE}_{\mathrm{ndh}}} = |\Pr[S_0] - 1/2|. \tag{8}$$

Game 1. We now describe Game 1, which is the same as Game 0, but with the following difference: the challenger will abort the game if the adversary query H at $\langle Y, Z_1, \ldots, Z_n \rangle$ either in Phase 1 or Phase 2. Everything else remains exactly the same as Game 0. Let S_1 be the event that $\beta' = \beta$ in Game 1 and F be the event that the adversary queries the random oracle at $\langle Y, Z_1, \ldots Z_n \rangle$ in Game 1. Since Game 0 and Game 1 proceed identically unless F occurs, we have

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[F]. \tag{9}$$

We claim that

$$\Pr[F] \leq \mathsf{AdvnDH}_{\mathcal{B}_{\mathrm{ndh}},\mathbb{G}}, \tag{10}$$

where $\mathcal{B}_{\mathrm{ndh}}$ is an efficient strong n-DH adversary that makes at most Q_h decison oracle queries. Next we detail how $\mathcal{B}_{\mathrm{ndh}}$ plays the role of the challenger in Game 1 to gain the advantage as claimed.

Setup: $\mathcal{B}_{\mathrm{ndh}}$ is given $(X_1, \ldots, X_n, Y)$ as the n-DH challenge instance. $\mathcal{B}_{\mathrm{ndh}}$ gives the adversary $\mathbf{pk} = (X_1, \ldots, X_n)$. Note that the only difference between $\mathcal{B}_{\mathrm{ndh}}$ and the challenger in Game 1 is that the former does not know the $\mathbf{sk} = (x_1, \ldots, x_n)$.

Hash oracle queries: Except processes the queries the same way as the challenger does in Game 1, for every random oracle query $(\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_n)$, $\mathcal{B}_{\mathrm{ndh}}$ sends this tuple to its own decision oracle, and marks it "good" or "bad" accordingly.

Phase 1 - Decryption queries: $\mathcal{B}_{\mathrm{ndh}}$ can process the decryption queries without using the secret key: given a ciphertext $(\hat{Y}, \hat{c})$, it checks if it has already seen a "good" tuple of the form $(\hat{Y}, \cdot, \ldots, \cdot)$ in L; if so, it uses the key associated with that tuple; if not, it generates a random key, and it will stay on the lookout for a "good" tuple of the form $(\hat{Y}, \cdot, \ldots, \cdot)$ in future random oracle queries, associating this key with that tuple to keep things consistent.

Challenge: Once the adversary decides that Phase 1 is over it outputs two messages M_0, M_1 on which it wishes to be challenged. $\mathcal{B}_{\mathrm{ndh}}$ checks if there is a "good" tuple of the form $(Y, \cdot, \ldots, \cdot)$, if so, it aborts; if not, it generates a random key k (it will stay on the lookout for a "good" tuple of the form $(\hat{Y}, \cdot, \ldots, \cdot)$ in future random oracle queries, associating this key with that tuple to keep things consistent), and computes $c := \mathsf{E}_k(M_\beta)$, and returns the ciphertext (Y, c) to $\mathcal{A}$.

Phase 2 - Decryption queries: The decryption queries in Phase 2 are processed just as in Phase 1. If the adversary issues a "good" tuple of the form $(Y, \cdot, \ldots, \cdot)$, $\mathcal{B}_{\mathrm{ndh}}$ aborts.

Guess: The adversary $\mathcal{A}$ outputs its guess β' for β.

At the end of the game, $\mathcal{B}_{\mathrm{ndh}}$ checks if it has seen a "good" tuple of the form $(Y, \cdot, \ldots, \cdot)$; if so, it outputs the last n components. According to the definition

of event F, Equation (10) follows immediately. Theorem 2.3 gives us an efficient DH adversary $\mathcal{B}_{\mathrm{dh}}$ with

$$\mathsf{AdvnDH}_{\mathcal{B}_{\mathrm{ndh},\mathbb{G}}} \leq \frac{p^{n-1}}{p^{n-1}-Q_h}\mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh},\mathbb{G}}}.$$

Finally, it is easy to see that in Game 1, the adversary is essentially playing the chosen ciphertext attack game against SE. Thus, there is an efficient adversary $\mathcal{B}_{\mathrm{sym}}$ such that

$$|\Pr[S_1] - 1/2| = \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}}. \tag{11}$$

Theorem 5.1 now follows by combining (8)-(11). □

6 The n-IBE scheme

We now present details of the n-IBE scheme. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of prime order p and $\mathbb{G}$ with generator g, and further let the two groups be equipped with a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. A master public key set is a tuple of n group elements $\mathbf{mpk} = (X_1, \ldots, X_n)$, where $X_i = g^{x_i}$ for $i \in I$ and $I = \{1, \ldots, n\}$. The corresponding master private key set is a tuple $\mathbf{msk} = (x_1, \ldots, x_n)$, which are selected at random from $\mathbb{Z}_p$. We treat the secret/public master key set $(\mathbf{msk}, \mathbf{mpk})$ as n separate key pairs $(x_1, X_1), \ldots, (x_n, X_n)$, which belong to n Key Generation Centers (KGCs) respectively. This scheme uses a symmetric cipher $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ and two hash functions H and G, where the hash function G is defined as $\mathbb{G} \times \{0,1\}^* \to \mathbb{G}$, and the hash function H is defined as $(\{0,1\}^*)^n \times \mathbb{G} \times \mathbb{G}_T^n \times \to \{0,1\}^\lambda$ (λ is the length of a symmetric key in algorithm SE).

A private key set associated with n individual identities, denoted by $\mathbf{id} = (id_1, \ldots, id_n)$ for $id_i \in \{0,1\}^*$ and $i \in I$, is a tuple of n group elements $\mathbf{sk} = (S_1, \ldots, S_n)$. The i-th element of $\mathbf{sk}$ is $S_i = \mathsf{G}(X_i, id_i)^{x_i}$. To encrypt a message $M \in \mathcal{M}$ for $\mathbf{id}$, one chooses $y \in \mathbb{Z}_p$ at random and sets

$$Y := g^y, W_i := e(\mathsf{G}(X_i, id_i), X_i)^y \text{ for } i \in I,$$

$$k := \mathsf{H}(id_1, \ldots, id_n, Y, W_1, \ldots, W_n), C := \mathsf{E}(k, M).$$

The ciphertext is (Y, C). To decrypt using $\mathbf{sk}$ for $\mathbf{id}$, one computes

$$W_i := e(S_i, Y) \text{ for } i \in I, k := \mathsf{H}(id_1, \ldots, id_n, Y, W_1, \ldots, W_n), M := \mathsf{D}(k, C).$$

Similar to the n-ElGamal encryption scheme in Section 5, the length of the ciphertext in the n-IBE scheme is independent to the number of KGCs and identities n. Like the twin IBE scheme of [6], the n-IBE scheme does not add redundancy to the ciphertext as in the Fujisaki-Okamoto transformation [17], which, e.g., is used in the Boneh-Franklin IBE scheme [4] and the Sakai-Kasahara IBE scheme [8, 22]. Now we essentially follow the security analysis approach for

the twin IBE scheme in [6] (the approach was originally proposed in [20]), to rigorously analyze security of the n-IBE scheme. We denote our n-IBE scheme by $\mathsf{MIBE}_{\mathrm{nbdh}}$. We prove that $\mathsf{MIBE}_{\mathrm{nbdh}}$ is secure against the chosen ciphertext attack under the strong n-BDH assumption. By Theorem 3.1, $\mathsf{MIBE}_{\mathrm{nbdh}}$ have been eventually proved to be IND-ID-CCA secure under the BDH assumption if the symmetric cipher is chosen ciphertext secure and the hash functions are treated as random oracles.

Theorem 6.1 *Suppose* H *and* G *are modeled as random oracles. Further, suppose the BDH assumption holds with* $(\mathbb{G}, \mathbb{G}_T, e)$*, and that the symmetric cipher* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *is secure against chosen ciphertext attack. Then* $\mathsf{MIBE}_{\mathrm{nbdh}}$ *is secure against the chosen ciphertext attack.*
In particular, suppose $\mathcal{A}$ *is an adversary that carries out a chosen ciphertext attack against* $\mathsf{MIBE}_{\mathrm{nbdh}}$*, and that* $\mathcal{A}$ *runs in time* τ*, and makes at most* Q_h *hash* H *queries,* Q_g *hash* G *queries,* Q_d *decryption queries, and* Q_e *secret key* sk_i *extraction queries associated with* id_i*, where* sk_i *(*id_i*) is an element of* $\mathbf{id}$ *(*$\mathbf{sk}$*). Then there exist a BDH adversary* $\mathcal{B}_{\mathrm{bdh}}$ *and an adversary* $\mathcal{B}_{\mathrm{sym}}$ *against the chosen ciphertext security of* SE*, such that both* $\mathcal{B}_{\mathrm{bdh}}$ *and* $\mathcal{B}_{\mathrm{sym}}$ *run in time at most* τ*, plus that time to perform* $O((Q_e + Q_h + Q_g + Q_d)\log p)$ *group operations; moreover*[3]

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{MIBE}_{\mathrm{nbdh}}} \leq \left(\frac{eQ_e}{n}\right)^n \cdot \left(\frac{q^{n-1}}{q^{n-1}-Q_h} \cdot \mathsf{AdvBDH}_{\mathcal{B}_{\mathrm{bdh}},\mathbb{G}} + \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}}\right).$$

Proof. We proceed with a sequence of games.
Game 0. Let Game 0 be the original MIBE chosen ciphertext attack game as defined in Section 4.2, and let S_0 be the event that $\beta' = \beta$ in this game.

Setup. The challenger keeps the master secret key set $\mathbf{msk} = (x_1, \ldots, x_n)$ to itself and gives the adversary the corresponding public key set $\mathbf{mpk} = (X_1, \ldots, X_n)$.

Hash oracle queries: To track random oracle responses, the challenger uses two lists L and K to process G and H oracle queries, respectively. Both lists are initially empty. L list is used to process G queries, whose tuples are of the form (X_i, id, Q_i), where $Q_i = \mathsf{G}(X_i, id)$ can be viewed as the the public key for identity id associated with the i-th KGC. The tuples in L are indexed by (X_i, id). K list is used to process H queries, whose tuples are of the form $(id_1, \ldots, id_n, Y, W_1, \ldots, W_n, k)$, $k = \mathsf{H}(id_1, \ldots, id_n, Y, W_1, \ldots, W_n)$ is the corresponding symmetric key. The tuples in K are indexed by $(id_1, \ldots, id_n, Y)$. When processing a G query or H query, the challenger returns the corresponding entry if it is defined, and otherwise initialize it with an appropriate random value and return that value.

Phase 1 - Private key queries: The challenger answers the private key queries using the master secret key set **msk**.

[3] Here $e \approx 2.71$ is the base of the natural logarithm.

Phase 1 - Decryption queries: The challenger uses the master secret key set **msk** to extract the private key, then answers the decryption query with the corresponding private key. In this operation, the challenger may need to call the hash oracles.

Challenge: Once the adversary decides that Phase 1 is over it outputs two equal length plaintext $M_0, M_1 \in \mathcal{M}$ and a set of identities $\mathbf{id}^*$ on which it wishes to be challenged. The challenger randomly picks $y \in \mathbb{Z}_p$ and sets $Y = g^y$, and randomly picks $\beta \in \{0, 1\}$, then encrypts M_β and sends it to $\mathcal{A}$.

Phase 2: The challenger proceeds in the same way it did in Phase 1.

Guess: $\mathcal{A}$ outputs its guess β' for β.

Accord to the definition, we have

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{MIBE}_{\mathsf{nbdh}}} = |\Pr[S_0] - 1/2|. \tag{12}$$

Game 1: Game 1 resembles Game 0, but now we change how the challenger processes queries to G, private key queries, and challenge query.

Setup: Same as in Game 0.

Hash oracle queries: In addition to inserting oracle responses into L as in Game 0, the challenger also "marks" some entries in the L list used to store G responses. On querying $\mathsf{G}(X_i, id)$, in addition to the normal processing, with probability δ the challenger marks $L[X_i, id]$ (the value of δ will be determined later). The challenger completely hides the marks from the adversary.

Phase 1 - Private key query: If the corresponding tuple $L[X_i, id]$ is marked, the challenger aborts the game. Otherwise, the challenger responds the same way as in Game 0.

Phase 2 - Decryption query: The challenger processes decryption query in the same way as in Game 0.

Challenge: Once the adversary decides that Phase 1 is over it outputs two equal length plaintext $M_0, M_1 \in \mathcal{M}$ and a set of identities $\mathbf{id}^* = (id_1^*, \ldots, id_n^*)$ on which it wishes to be challenged. If one or more $L[X_i, id_i^*]$ are not marked, then the challenger aborts. Otherwise it proceeds normally (i.e., it picks $y \in \mathbb{Z}_q$, sets $Y = g^y$, then encrypts M_β and sends the result to the adversary).

Phase 2: The challenger proceeds the private key queries and the decryption queries the same way as it did in Phase 1.

Guess: The adversary outputs its guess β' for β.

Let S_1 be the event that $\beta' = \beta$ and F_1 be the event that the challenger aborts in Game 1. Since the coins that determine F_1 are independent of the rest of the game, it follows that

$$\Pr[S_1]/\Pr[S_0] = \Pr[\neg F_1] = \delta^n \cdot (1 - \delta)^{Q_e}. \tag{13}$$

Here $\delta^n \cdot (1 - \delta)^{Q_e}$ features the reduction tightness, we denote it by reduction factor r. For sufficiently large n and relatively small x, we know

$$\left(1 + \frac{x}{n}\right)^n \approx e^x.$$

By applying the Inequality of Arithmetic and Geometric Means, we have

$$\begin{aligned}\delta^n \cdot (1-\delta)^{Q_e} &= \left(\frac{n}{Q_e}\right)^n \left(\frac{Q_e}{n}\delta\right)^n \cdot (1-\delta)^{Q_e} \\ &\leq \left(\frac{n}{Q_e}\right)^n \left(1 - \frac{n}{n+Q_e}\right)^{n+Q_e} \\ &\approx \left(\frac{n}{Q_e}\right)^n e^{-n}.\end{aligned} \tag{14}$$

The equality of (14) holds when $\delta = n/(n+Q_e)$. Thus the lower bound of the reduction factor is $r_{\min} = \left(\frac{n}{Q_e}\right)^n e^{-n}$. If the adversary makes less queries the reduction factor r will only be greater. Thus we have

$$r_{\min} \cdot \Pr[S_0] \leq r \cdot \Pr[S_0] = \Pr[S_1], \tag{15}$$

that gives us

$$\Pr[S_0] \leq \frac{1}{r_{\min}} \Pr[S_1] = e^n \cdot \left(\frac{Q_e}{n}\right)^n \Pr[S_1]. \tag{16}$$

Game 2. Game 2 will be like Game 1, except that the challenger will aborts for some hash H oracle queries.

Hash oracle query: The same as in Game 1.

Phase 1 - Private key queries: The same as in Game 1.

Phase 1 - Decryption queries: The same as in Game 1.

Challenge: Once the adversary decides that Phase 1 is over it outputs two equal length plaintext $M_0, M_1 \in \mathcal{M}$ and a set identities $\mathbf{id}^* = (id_1^*, \ldots, id_n^*)$ on which it wishes to be challenged. If one or more $L[X_i, id_i^*]$ are not marked, then the challenger aborts. Otherwise, the challenger computes $W_i := e(\mathsf{G}(X_i, id_i^*), X_i)^y$, then it checks whether $(id_1^*, \ldots, id_n^*, Y, W_1, \ldots, W_n)$ has been asked in Phase 1. If so, the challenger aborts. Otherwise the challenger proceeds normally (i.e., setting $\mathsf{H}(id_1^*, \ldots, id_n^*, Y, W_1, \ldots, W_n) := k$, computes $C := \mathsf{E}(k, M_\beta)$, and returns (Y, C)).

Phase 2. The challenger proceeds in the same way as in Phase 1, except that the challenger will aborts if the adversary queries H at $(id_1^*, \ldots, id_n^*, Y, W_1, \ldots, W_n)$.

Guess. The adversary outputs its guess β' for β.

Let S_2 be the event that $\beta' = \beta$ in Game 2. Let F_{nbdh} be the event that the adversary queries H at $\mathsf{H}(id_1^*, \ldots, id_n^*, Y, Z_1, \ldots, Z_n)$ either in Phase 1 or Phase 2. Since Game 1 and Game 2 are exactly the same when F_{nbdh} does not occur, it follows that

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[F_{\text{nbdh}}]. \tag{17}$$

We claim that

$$\Pr[F_{\text{nbdh}}] \leq \mathsf{AdvnBDH}_{\mathcal{B}_{\text{nbdh},\mathbb{G}}}, \tag{18}$$

for an efficient strong n-BDH adversary $\mathcal{B}_{\text{nbdh}}$ that makes Q_h decision oracle queries.

Next we details how $\mathcal{B}_{\text{nbdh}}$ plays the role of the challenger in Game 2 to gain the claimed advantage.

Setup: $\mathcal{B}_{\text{nbdh}}$ is given a n-BDH instance $(X_1, \ldots, X_n, Y, Z_1, \ldots, Z_n)$ as input and begins to run Game 2, acting as the challenger for the adversary. It sets the public key set $(X_1, \ldots, X_n)$.

Hash oracle queries: When the adversary requests $\mathsf{G}(X_i, id)$, if $L[X_i, id]$ gets marked, $\mathcal{B}_{\text{nbdh}}$ chooses a random $r \in \mathbb{Z}_p$, sets $Q_i := Z_i^r$, stores it at $L[X_i, id]$ and gives it to the adversary. If the entry does not get marked, $\mathcal{B}_{\text{nbdh}}$ stores and returns $Q_i := g^r$ instead. In either case, r is remembered for later use. For every query $\mathsf{H}(\hat{id}_1, \ldots, \hat{id}_n, \hat{Y}, \hat{W}_1, \ldots, \hat{W}_n)$, $\mathcal{B}_{\text{nbdh}}$ looks up each $\hat{Q}_i$ stored at $L[X_i, \hat{id}_i]$ (if some $\hat{Q}_i$ has not been created yet, the challenger queries the G oracle with $(X_i, \hat{id}_i)$ by itself), and then queries its n-BDH decision oracle with $(\hat{X}_1, \ldots \hat{X}_n, \hat{Y}, \hat{Q}_1, \ldots, \hat{Q}_n, \hat{W}_1, \ldots, \hat{W}_n)$, and marks the tuple as "good" or "bad" depending the answer.

Phase 1 - Private key queries: When the adversary requests private key for an identity $\hat{id}$ associated with the i-th KGC, if $L[X_i, \hat{id}]$ is not marked, $\mathcal{B}_{\text{nbdh}}$ retrieves the r used to generate the entry g^r in $L[X_i, \hat{id}]$, and returns $\hat{S}_i = X_i^r$. If $L[X_i, \hat{id}]$ is marked, $\mathcal{B}_{\text{nbdh}}$ immediately aborts.

Phase 1 - Decryption queries: Upon receiving the decryption query $(\hat{Y}, \hat{c})$ with $\hat{\mathbf{id}} = (\hat{id}_1, \ldots, \hat{id}_n)$, $\mathcal{B}_{\text{nbdh}}$ responds as follows:

1. If all $L[X_i, \hat{id}]$ tuples are unmarked, $\mathcal{B}_{\text{nbdh}}$ retrieves the corresponding private key $(\hat{S}_1, \ldots, \hat{S}_n)$ and computes $\hat{W}_i := e(\hat{S}_i, \hat{Y})$. $\mathcal{B}_{\text{nbdh}}$ queries the H oracle at $(\hat{id}_1, \ldots, \hat{id}_n, \hat{Y}, \hat{W}_1, \ldots, \hat{W}_n)$, obtains the symmetric key $\hat{k}$.
2. If one or more $L[X_i, \hat{id}]$ are marked, $\mathcal{B}_{\text{nbdh}}$ check whether there is a "good" tuple on K list indexed by $(\hat{id}_1, \ldots, \hat{id}_n, \hat{Y})$. If so, $\mathcal{B}_{\text{nbdh}}$ retrieves the associated $\hat{k}$. If not, the challenger generates a random key $\hat{k}$, and it will stay on the lookout for a "good" tuple indexed by $(\hat{id}_1, \ldots, \hat{id}_n, \hat{Y})$ in future random oracle queries, associating this key with that tuple to keep things consistent.

In either way, the challenger can get the symmetric key $\hat{k}$ and decrypts the ciphertext.

Challenge: Once the adversary decides that Phase 1 is over it outputs two equal length plaintext $M_0, M_1 \in \mathcal{M}$ and a set of identities $\mathbf{id}^* = (id_1^*, \ldots, id_n^*)$ on which it wishes to be challenged. If one or more $L[X_i, id_i^*]$ are not marked, $\mathcal{B}_{\text{nbdh}}$ aborts. Otherwise, $\mathcal{B}_{\text{nbdh}}$ looks for a "good" tuple indexed by $(id_1, \ldots, id_n, Y)$ on K, if there is one, $\mathcal{B}_{\text{nbdh}}$ aborts. If there is no such a "good" tuple indexed by $(id_1^*, \ldots, id_n^*, Y)$, $\mathcal{B}_{\text{nbdh}}$ generates a random key $\hat{k}$ associated to $(id_1^*, \ldots, id_n^*, Y)$, and it will stay on the lookout for a "good" tuple indexed by $(id_i^*, \ldots, id_n^*, Y)$ in future random oracle queries, associating this key with that tuple to keep things consistent. $\mathcal{B}$ computes $C^* := \mathsf{E}(k, M_\beta)$, and returns (Y, C^*) to $\mathcal{A}$.

Phase 2: $\mathcal{B}_{\text{nbdh}}$ proceeds the same way as it did in Phase 1, except that it will abort if the adversary queries H at $(id_1^*, \ldots, id_n^*, Y, W_1, \ldots, W_n)$, where $W_i = e(\mathsf{G}(X_i, id_i^*), X_i)$. Note that $\mathcal{B}_{\text{nbdh}}$ identify such query with the help of its decision oracle.

Guess. The adversary outputs its guess β' for β.

After the game ends, $\mathcal{B}_{\text{nbdh}}$ examines K and looks for a good entry of the form $(id_1^*, \ldots, id_n^*, Y, W_1, \ldots, W_n)$. If it finds one ($F_{\text{nbdh}}$ happens), it outputs $(W_i)^{1/r}$. According to the definition of event of F_{nbdh}, Equation (18) holds immediately.

Finally, in Game 2 the adversary is essentially playing the chosen ciphertext game against SE. Thus there is an adversary $\mathcal{B}_{\text{sym}}$ such that

$$|\Pr[S_2] - 1/2 \cdot \Pr[\neg F_1]| = \mathsf{AdvCCA}_{\mathcal{B}_{\text{sym}}, \mathsf{SE}}. \tag{19}$$

Theorem 6.1 follows by combining (12)-(19). □

Remark 1. We have two methods to answer the G query with (X_i, id) when $L[X_i, id]$ is marked:

1. Choose a random r, set $\mathsf{G}(X_i, id) = Z_i^r$. The corresponding answer to the underlying problem is $(W_i)^{1/r}$.
2. Choose a random r, set $\mathsf{G}(X_i, id) = Z_i \cdot g^r$. The corresponding answer to the underlying problem is $Z_i / e(X_i, Y)^r$.

7 Conclusions

We have proposed a new computational problem called the n-DH problem, which is an extension of the twin DH problem of [6], and also proposed the associated strong n-DH problem and the (strong) n-BDH problem. We have shown that the strong n-DH (n-BDH) problem is as hard as the ordinary DH (BDH) problem. We have introduced a formal definition of n-out-of-n encryption which has two versions, namely MPKE and MIBE for the conventional public key setting and identity-based key setting respectively. We have also proposed an efficient MPKE (MIBE) scheme and proved it is CCA secure under the DH (BDH) assumption.

In our security model for an MPKE (MIBE) scheme, the adversary is not allowed to corrupt any individual key in the whole set of n keys, which is used in the challenge phase. This security model suits our target applications of multiple key encryption very well, where the decryption process requires that either a decryptor must holds n keys or that n decryptors much work together. However, whether this model can be strengthened and whether there is any practical motivation to any enhancement of the model might be an interesting topic for further investigation. Whether there are other applications which can benefit from the (strong) n-DH/n-BDH problem is another question which could lead to some future research.

References

1. Michel Abdalla, Mihir Bellare, and Phillip Rogaway, *The oracle diffie-hellman assumptions and an analysis of DHIES*, Topics in Cryptology - CT-RSA 2001, LNCS, vol. 2020, 2001, pp. 143–158.
2. Mihir Bellare and Phillip Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, the 1st ACM Conference on Computer and Communications Security, ACM Press, 1993, pp. 62–73.
3. John Bethencourt, Amit Sahai, and Brent Waters, *Ciphertext-Policy Attribute-Based Encryption*, IEEE Symposium on Security and Privacy 2007 (SP' 2007) (2007), 321–334.
4. Dan Boneh and Matthew Franklin, *Identity-based encryption from the weil pairing*, Advances in Cryptology - CRYPTO 2001, LNCS, vol. 2139, 2001, pp. 213–229.
5. Dan Boneh, Craig Gentry, and Brent Waters, *Collusion resistant broadcast encryption with short ciphertexts and private keys*, Advances in Cryptology - CRYPTO 2005, LNCS, vol. 3621, Springer, 2005, pp. 258–275.
6. David Cash, Eike Kiltz, and Victor Shoup, *The twin diffie-hellman problem and applications*, Advances in Cryptology - EUROCRYPT 2008, LNCS, vol. 4965, 2008, pp. 127–145.
7. Liqun Chen, *An interpretation of identity-based cryptography*, Foundations of Security Analysis and Design IV, FOSAD 2006/2007 Tutorial Lectures, LNCS, vol. 4677, 2007, pp. 183–208.
8. Liqun Chen and Zhaohui Cheng, *Security proof of sakai-kasahara's identity-based encryption scheme*, Cryptography and Coding, 10th IMA International Conference, LNCS, vol. 3796, 2005, pp. 442–459.
9. Liqun Chen and Keith Harrison, *Multiple trusted authorities in identifier based cryptography from pairings on elliptic curves*.
10. Liqun Chen, Keith Harrison, David Soldera, and Nigel Smart, *Applications of multiple trust authorities in pairing based cryptosystems*, In Proceedings of Infrastructure Security Conference 2002, LNCS, vol. 2437, 2003, pp. 260–275.
11. Yu Chen and Liqun Chen, *Twin bilinear diffie-hellman inversion problem and its application*, The 13th Annual International Conference on Information Security and Cryptology, ICISC 2010, LNCS, Springer, 2010, p. to appear.
12. Ronald Cramer and Victor Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, Advances in Cryptology - CRYPTO 1998, LNCS, vol. 1462, 1998, pp. 13–25.
13. ———, *Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack*, SIAM Journal on Computing **33** (2001), 167–226.
14. Ivan Damgård and Mads Jurik, *A length-flexible threshold cryptosystem with applications*, Information Security and Privacy, 8th Australasian Conference, ACISP 2003, LNCS, vol. 2727, Springer, 2003, pp. 350–364.
15. Cécile Delerablée, Pascal Paillier, and David Pointcheval, *Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys*, Pairing-Based Cryptography - Pairing 2007, LNCS, vol. 4575, 2007, pp. 39–59.
16. Whitefield Diffie and Martin E. Hellman, *New directions in cryptograpgy*, IEEE Transactions on Infomation Theory **22(6)** (1976), 644–654.
17. Eiichiro Fujisaki and Tatsuaki Okamoto, *Secure integration of asymmetric and symmetric encryption schemes*, Advances in Cryptology - CRYPTO 1999, LNCS, vol. 1666, 1999, pp. 537–554.

18. Steve Galbraith, Kenny Paterson, and Nigel P. Smart, *Pairings for cryptographers*, Discrete Applied Mathematics **156(16)** (2008), 3113–3121.
19. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters, *Attribute-based encryption for fine-grained access control of encrypted data*, ACM Conference on Computer and Communications Security, ACM CCS 2006, ACM, 2006, pp. 89–98.
20. Benoít Libert and Jean-Jacques Quisquater, *Identity based encryption without redundancy*, ACNS 2005, LNCS, vol. 3531, pp. 285–300.
21. Charles Rackoff and Daniel R. Simon, *Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack*, Advances in Cryptology - CRYPTO 1991, LNCS, vol. 576, 1991, pp. 433–444.
22. Ryuichi Sakai and Masao Kasahara, *Id based cryptosystems with pairing on elliptic curve*, Cryptology ePrint Archive, Report 2003/054, 2003, `http://eprint.iacr.org/`.
23. Adi Shamir, *How to share a secret*, Commun. ACM **22** (1979), no. 11, 612–613.