Generic Hardness of Inversion on Ring and Its Relation to Self-Bilinear Map

Takashi Yamakawa[1], Shota Yamada[2], Goichiro Hanaoka[2], Noboru Kunihiro[3]

[1]NTT Secure Platform Laboratories, Tokyo, Japan
takashi.yamakawa.ga@hco.ntt.co.jp
[2]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
{yamada-shota,hanaoka-goichiro}@aist.go.jp
[3]University of Tsukuba, Tsukuba, Japan
kunihiro@cs.tsukuba.ac.jp

May 20, 2019

Abstract

In this paper, we study the generic hardness of the *inversion problem* on a ring, which is a problem to compute the inverse of a given prime c by just using additions, subtractions and multiplications on the ring. If the characteristic of an underlying ring is public and coprime to c, then it is easy to compute the inverse of c by using the extended Euclidean algorithm. On the other hand, if the characteristic is hidden, it seems difficult to compute it. For discussing the generic hardness of the inversion problem, we first extend existing generic ring models to capture a ring of an unknown characteristic. Then we prove that there is no generic algorithm to solve the inversion problem in our model when the underlying ring is isomorphic to $\mathbb{Z}_p$ for a randomly chosen prime p assuming the hardness of factorization of an unbalanced modulus. We also study a relation between the inversion problem on a ring and a self-bilinear map. Namely, we give a construction of a self-bilinear map based on a ring on which the inversion problem is hard, and prove that natural complexity assumptions including the multilinear computational Diffie-Hellman (MCDH) assumption hold w.r.t the resulting sef-bilinear map.

Keywords: generic ring, straight-line program, self-bilinear map, multilinear map

1 Introduction

1.1 Background

Cyclic groups on which the discrete logarithm (DL) is hard to solve have been widely used in cryptography. Though the hardness of the DL is a widely used assumption, we still do not know how to prove that there does not exist a polynomial time algorithm that solves the DL problem. On the other hand, for reliability of the assumption, we want to argue a lower bound for the DL problem in some sense. For this purpose, Shoup [Sho97] introduced an idealized model called the *generic group model* (GGM) building on an earlier works by Babai and Szemerédi [BS84] and Nechaev [Nec94]. In the GGM, group elements are represented by a random string, and group operations are done only through oracle accesses. Intuitively, the GGM captures

adversaries that do not depend on actual representations of group elements. Shoup proved that the DL problem and its related problems are hard to solve with polynomial number of group operations in the GGM.

Though Shoup's GGM can capture most DL-related problems, that does not capture the RSA problem [RSA78], which is another commonly used problem in cryptography. The RSA problem is the problem to compute $g^{1/c}$ given $g \in \mathbb{Z}_N^*$ and a prime c where $N = PQ$ is a product of two primes P and Q. It is well-known that the order of $\mathbb{Z}_N^*$ is hard to compute as long as one cannot factorize N. On the other hand, Shoup's GGM model assumes that an order of a group is public, and this is the reason why his model does not capture the RSA problem. To address the issue, Damgård and Koprowski [DK02] extended the GGM to capture unknown order groups. Intuitively, they proved that the RSA problem is hard to solve for generic algorithms if an entropy of an order of an underlying group is large enough.[1]

There is another extension of the GGM called the generic ring model (GRM) [LR06, AM09, JS09, AMS11]. The GRM is defined similarly to the GGM by simply replacing a group with a ring. The motivation of considering the GRM is more precise validation of the RSA and its related problems. Specifically, since $\mathbb{Z}_N$ is a ring, an adversary may utilize additions on $\mathbb{Z}_N$ in addition to multiplications. Thus GRM is supposed to capture $\mathbb{Z}_N$ as an underlying ring for an RSA modulus N. Here, we remark the characteristic N of a ring $\mathbb{Z}_N$ is public and known by an adversary.

While these two types of extensions have been studied, almost no research has been done for the mix of them, i.e., the GRM in an unknown characteristic setting.[2] We suppose that this is because currently we have no candidate of a ring whose ring operations can be done efficiently while keeping its characteristic hidden. On the other hand, Kim et al. [KKS15] observed that if such a ring exists, then we can construct a self-bilinear map [CL09, YYHK17], which is a bilinear map whose source and target groups are identical. As observed in some works [CL09, YYHK17], a self-bilinear map is quite a powerful primitive for constructing various cryptographic schemes. For example, a self-bilinear map gives an unbounded level multilinear map, which immediately gives an improvement for some multilinear-map-based cryptographic schemes such as multi-party non-interactive key exchange and broadcast encryption [BS02]. Considering the extreme usefulness of a self-bilinear map, even with the lack of an instantiation, it is still meaningful to study more on a ring of unknown characteristic since a candidate of such a ring may be found in the future.

In addition, though Kim et al. [KKS15] gave an idea to construct a self-bilinear map based on a ring of an unknown characteristic, they did not give any formal reduction between assumptions of a resulting self-bilinear map and an underlying ring. For further understanding ring-based construction of a self-bilinear map, it would be helpful to study their relations more.

1.2 Our Contribution

In this work, we consider the *inversion problem* on a ring R that is isomorphic to $\mathbb{Z}_p$ where p is a "hidden" prime. Namely, we assume that additions, subtractions and multiplications on R can be done efficiently, and consider if the multiplicative inverse of a public prime c is efficiently computable. If p is public, then it is easy to compute $c^{-1} \mod p$ by using the extended Euclidean algorithm. However, if p is unknown, then we do not know how to compute $c^{-1} \mod p$ and it seems difficult to compute it.

[1] In the real world, the order of $\mathbb{Z}_N^*$ has no entropy given N though that is hard to compute. They consider an idealized setting where the order has an information theoretical entropy.

[2] To the best of our knowledge, the only work that is relevant to this setting is the work by Kim et al. [KKS15] that considered a "restricted" variant of the model. See Sec 1.4 for the details.

To give a generic lower bound for this problem, we extend the GRM to an unknown characteristic setting. Our model is defined almost similarly to an existing GRM [JS09] except that the ring is chosen according to a certain distribution and its characteristic is hidden to an adversary. We prove that the inverse of a prime c is hard to compute in our model when p is randomly chosen from the set of all primes smaller than 2^ℓ where $\ell = \Omega(\lambda)$ if the factoring of an unbalanced modulus is hard. Moreover, we give a partial evidence that the factoring assumption (or some assumption of similar nature) is needed for proving the generic hardness of the inversion by relating it to straight-line complexity of factorials.

Then we study a relation between an inversion-hard ring and a self-bilinear map. Based on the idea of Kim et al. [KKS15], we consider a ring-based construction of a self-bilinear map. We prove that if the c-inversion problem is hard on the underlying ring for some prime c that is coprime to the characteristic of the ring, then the resulting self-bilinear map satisfies several natural complexity assumptions including the multilinear computational Diffie-Hellman (MCDH) assumption. By combining with the generic hardness result for the inversion problem, we can conclude that these natural complexity assumptions for self-bilinear maps hold against generic adversaries if the factoring of an unbalanced modulus is hard.

Finally, we show that some existing multilinear-map-based constructions gain some advantages if they are instantiated with an unbounded-level multilinear map obtained from a self-bilinear map by showing examples of the non-interactive key exchange (NIKE) and broadcast encryption schemes proposed by Boneh and Silverberg [BS02] and the homomorphic signatures proposed by Catalano et al. [CFW14]. We believe that self-bilinear maps have further applications.

1.3 Discussion

Interpretation. Here, we discuss how we should interpret our results. First, we remark that hardness in the GRM (resp. GGM) *does not* mean hardness with respect to a specific way of representing ring (resp. group) elements. For example, Jager and Schwenk [JS09] showed that computing the Jacobi symbol on $\mathbb{Z}_n$ for a composite number n is as hard as factorizing n in the GRM though that can be done efficiently if group elements are represented in a standard way. Nonetheless, a hardness result in the GRM (resp. GGM) rules out a large class of attacks that do not depend on representations of ring (resp. group) elements. Especially, our result rules out generic attacks against natural assumptions on a prime-order cyclic group with efficiently computable self-bilinear map that do not depend on representations of group elements (under hardness of factoring). This encourages further study for finding a way of representing group elements with which these assumptions survive. We also believe that the complexity of computing inverses by just using generic ring operations (except for division) is of independent interest from the view of broader theoretical computer science as it treats a fundamental problem on computations on rings.

Candidate instantiation. In this paper, we prove that an inversion-hard ring gives a cryprographically useful self-bilinear map. On the other hand, we are currently unaware of any ring with efficiently computable group operations that may satisfy the c-inversion assumption. Here, we discuss a heuristic candidate instantiation of an inversion-hard ring based on an obfuscation [BGI+01]. Intuitively, an obfuscation is to make a program "unintelligible" while keeping its functionality. Our idea is to consider an "encrypted ring" where all ring elements are encrypted and the program that computes ring operations over ciphertexts is obfuscated. Since all ring elements are encrypted, any algorithm cannot obtain any useful information from the representation of ring elements. Therefore the ring can be treated as if that is a "generic ring", and

we can expect that the inversion assumption holds for this ring. Indeed, if we assume that an underlying obfuscation satisfies the virtual black-box (VBB) security [BGI+01], then the resulting scheme satisfies the inversion assumption as shown in Appendix A.

One may think that VBB obfuscation is a too strong assumption, and we can trivially obtain whatever we want based on that. However, we remark that the existence of an inversion-hard ring is not trivial even if we assume VBB obfuscation. What we can say based on VBB obfuscation is that an "encrypted ring" as described above is as secure as a "generic ring" where an adversary can perform ring operations only through operation oracles. Therefore if there exists a generic attack against the c-inversion problem, then the attack also works in the VBB obfuscation-based instantiation. Thus even if we assume VBB obfuscation, it is unclear if the above construction gives an inversion-hard ring without additionally assuming the factoring assumption.

Comparison to existing works. Here, we discuss the relation between our technique and ones used in works by Altmann et al. [AJR08] and Aggarwal and Maurer [AM09]. Their techniques are similar to ours in the sense that they also related generic hardness of some problems on a ring to factoring. Namely, Altmann et al. [AJR08] reduced the ring extraction problem to factoring, and Aggarwal and Maurer [AM09] reduced the RSA problem to factoring in their models. However, a significant difference of this work from theirs is that we consider a ring of an *unknown prime characteristic* p whereas they consider a ring of a *known composite characteristic* N whose factorization is unknown. It looks rather natural that the factoring assumption appears in their setting because they consider a ring of characteristic N, which is a composite number to factorize. On the other hand, in our model, we only consider a ring of a prime characteristic, and no composite number appears in the model. Thus we believe that the relation between the inversion problem in our model and factoring is quite surprising and interesting.

Brown [Bro16, Appendix B] considered the hardness of computing the inverse of a randomly chosen element $x \in \mathbb{Z}_n$ given an encoding of x, where n is the product of randomly chosen primes p and q. He proved that this problem is hard to solve by using straight-line programs if the factorization of n is unknown. Though this seems quite related to our result, there is a significant difference. Namely, he considered a ring with a fixed characteristic n (that is supposed to be public) whereas we consider a ring with a randomly chosen characteristic (that is supposed to be hidden). In the setting of the fixed characteristic, the problem of computing the inverse of a fixed prime c is not hard at all since we can hardwire $(c^{-1} \mod n)$ into the description of a straight-line program. So he considered a problem of computing the inverse of a *randomly chosen* ring element instead of a fixed one. On the other hand, in our setting, what is randomly chosen is the characteristic of the ring, and not the ring element to compute the inverse. Also, as a less significant difference, he considered composite characteristics whereas we consider prime characteristics. By these differences, our result is incomparable to his result.

Damgård and Koprowski [DK02] considered the generic hardness of computing the inverse of a fixed prime in a cyclic group of *u*nknown order. Their generic group model is conceptually similar to our GRM in the sense that the order of the group is *r*andomly chosen by the model to capture the setting where the order is hidden from an adversary. On the other hand, an apparent difference from our work is that they considered groups whereas we consider rings.

Future direction. An obvious open problem is a construction of an inversion-hard ring, which implies a self-bilinear map, in the standard model. A possible approach is to use indistinguishability obfuscation (iO) instead of VBB obfuscation based on a similar idea to the works by Albrecht et al. [AFH+16] and Farshim et al. [FHHL18], which constructed multilinear maps

and graded encodings based on iO. However, a simple adaptation of their techniques would not work since we at least need that the uniform factorial conjecture is true as discussed in Sec. 5.4, whereas iO along with some other assumptions that do not imply the hardness of factoring seem not to imply that the uniform factorial conjecture is true. Thus, to construct an inversion-hard ring, it seems that we have to assume hardness of factoring or some assumption of similar nature in addition to iO, which would require new techniques.

1.4 Related Work

Groups of unknown order. In this paper, we consider a ring of an unknown characteristic. If we consider an additive group of a ring, then that is a group of unknown order, which have been studied well. A typical example of an unknown order group is $\mathbb{Z}_N^*$ for an RSA modulus $N = pq$. If we consider a setting where the Diffie-Hellman oracle on $\mathbb{Z}_N^*$ (with a fixed generator $g \in \mathbb{Z}_N^*$) is available, then that can be seen as a ring of an unknown characteristic, and the inversion problem roughly corresponds to the RSA problem. We remark that in such a setting, the characteristic $\phi(N)$ of the ring is not prime, and our results cannot be directly applied to this setting, though we believe that our results also extend to this setting.

Groups of infeasible inversion. Hohenberger [Hoh03] and Molnar [Mol03] considered *groups of infeasible inversion*, on which it is hard to compute the inverse of a random group element, and showed that such a group is useful for constructing directed transitive signature. Irrer et al. [ILOP04] showed that such a group is also useful for constructing broadcast encryption. However, there have been no known instantiations for such a group until very recently Altuğ and Chen [AC18] proposed a candidate instantiation based on a new computational problem related to isogenies of elliptic curves over unknown characteristic rings.

A group of infeasible inversion can be seen as a special case of our notion of inversion-hard ring where we only consider multiplications and do not consider additions.[3] There is still no candidate construction of an inversion-hard ring (except for the obfuscation-based one given in Appendix A).

"Restricted" GRM in [KKS15]. Sec. 3.3 of Kim et al.'s work [KKS15] can be seen as analyses of certain assumptions in a restricted variant of the GRM of an unknown characteristic. Here the term "'restricted"' means that they restrict a degree of the computation on a ring. More specifically, on a ring $\mathbb{Z}_p$, they restrict the degree of allowed computations to be much smaller than p. This is essential in their analyses because they use the Schwartz-Zippel lemma to argue that for a function f with n variables of degree d, the probability that $f(x_1, \ldots, x_n) = 0 \bmod p$ is negligible for randomly chosen $x_1, \ldots, x_n \in \mathbb{Z}_p$. If d is larger than p, this argument completely does not work. Thus their result cannot be extended to the case where there is no restriction for computations on a ring.

Self-bilinear map. The work of Lee [Lee04] was the first to consider self-bilinear maps in a cryptographic context. However, Cheon and Lee [CL09] observed that Lee's self-bilinear map is essentially not a self-bilinear map. They also showed a strong negative result on the existence of a self-bilinear map of a known order. Namely, they showed that if there exists an efficiently computable self-bilinear map on a known order group, then the computational Diffie-Hellman (CDH) assumption does not hold on the group.

[3] Actually, there is a slight difference between them that in groups of infeasible inversion, we consider a problem of computing an encoding of x^{-1} given an encoding of a randomly chosen element x whereas in inversion-hard rings, we consider a problem of computing an encoding of c^{-1} for a fixed and public prime c. However, we can show that if the latter is hard, then the former is also hard. See Remark 1 for details.

Yamakawa et al. [YYHK17] gave the first positive result on a self-bilinear map. They defined a weaker variant of a self-bilinear map which they call a self-bilinear map with auxiliary information. Then they constructed it based on the factoring assumption and the existence of iO. A self-bilinear map with auxiliary information is defined as a self-bilinear map that can be computed efficiently when some auxiliary information is given. Unfortunately, a self-bilinear map with auxiliary information is not as useful as an "ideal" self-bilinear map. For example, that cannot be used for constructing fully homomorphic signatures as done in this work. This is becuase the size of auxiliary information grows at least by double whenever we apply a group operation.

Kim et al. [KKS15] gave a candidate construction of a cryptographic self-bilinear map. However, it turns out that their construction is not secure [Seo17].

Multilinear map from obfuscation. There are some works that shows the relation between multilinear maps and iO. Paneth and Sahai [PS15] constructed a polynomial jigsaw puzzle, which is a variant of a multilinear map, solely based on iO. However, they did not provide any application of polynomial jigsaw puzzles except iO itself and thus it is unclear how that is useful in constructions of other cryptographic primitives.

Albrecht et al. [AFH+16] constructed a multilinear map based on iO. The construction was recently extended to a graded encoding scheme by Farshim et al. [FHHL18]. However, the levels of multilinearity in their constructions are bounded, and they did not give unbounded-level multilinear map based on iO.

Subsequent work. Subsequent to the posting of our work online, Bartusek, Ma, and Zhandry [BMZ19] used our result to give a separation between the CDH assumption w.r.t. a *fixed generator* and the CDH assumption w.r.t. a *random generator*.

2 Technical Overview

In this section, we give more detailed overview of our results.

2.1 Generic Ring with Unknown Characteristic

To formally discuss generic hardness of the inversion problem on rings of unknown characteristic (specifically, $\mathbb{Z}_p$ for a hidden prime p), we extend existing GRMs to treat such cases. Our model is formalized by using a stright-line program (SLP) similarly to the one in [JS09]. Especially, since we only consider the inversion problem on a ring R in which an adversary is only given a multiplicative identity $[1]_R$ of the ring as an input, we formalize our model by using a no-input stright-line program (NI-SLP), which is a simpler variant of a SLP that do not take any input other than $[1]_R$.

No-input straight-line program. An NI-SLP with length L is described by a sequence of instructions $\mathrm{P} = (i_1, j_1, \circ_1), \ldots, (i_L, j_L, \circ_L)$ where $i_k, j_k \in \{0, 1, \ldots, k-1\}$ and $\circ_k \in \{+, -, \cdot\}$ for all $k \in [L]$. On a ring R, a program P does the following recursive computation.

1. Let $x_0 := [1]_R$.
2. For $1 \leq k \leq L$, let $x_k := x_{i_k} \circ_k x_{j_k}$.
3. Output x_L.

We denote the output of P on a ring R by $\mathrm{P}(R)$. That is, an NI-SLP expresses a computation on a ring that consists of additions, subtractions and multiplications starting from the multiplicative identity $[1]_R$. This is essentially the same as an arithmetic circuit with fixed input $[1]_R$. We use the notion of NI-SLP because this is suitable for stating our model.

Our generic ring model. Here, we explain our extended GRM that captures an unknown characteristic setting. Our model is similarly defined to the GRM given by Jager and Schwenk [JS09] except that we consider a certain distribution $\mathcal{D}_R$ of a ring instead of considering a fixed ring. To clarify the distribution explicitly, we often denote our model by $\mathcal{D}_R$-GRM. In the $\mathcal{D}_R$-GRM, a ring R is chosen according to the distribution $\mathcal{D}_R$, whose description (especially, its characteristic) is not given to an adversary. Let NI-SLP P be an SLP that is initialized to be an empty sequence (which implicitly sets $x_0 := [1]_R$). Then a generic algorithm $\mathcal{A}$ in the model can make the following two types of queries: operation and equal queries.

operation **query:** When $\mathcal{A}$ makes an operation query $(i, j, \circ) \in \{0, \ldots, |\mathrm{P}|\} \times \{0, \ldots, |\mathrm{P}|\} \times \{+, -, \cdot\}$, then $\mathcal{O}_{\mathsf{ring}}$ append $(i, j, \circ)$ to P.

equal **query:** When $\mathcal{A}$ makes an equal query $(i, j) \in \{0, \ldots, |\mathrm{P}|\} \times \{0, \ldots, |\mathrm{P}|\}$, then $\mathcal{O}$ returns true if $\mathrm{P}_i(R) = \mathrm{P}_j(R)$ and false otherwise.

That is, $\mathcal{A}$ makes an operation query $(i, j, \circ)$ to compute $x_i \circ x_j$ and append the value to the sequence $x_0, ..., x_{|\mathrm{P}|}$ of ring elements, and it makes an equal query (i, j) to check the equality of x_i and x_j. Intuitively, this captures everything one can generically perform on a ring. Finally, $\mathcal{A}$ outputs an index i^*, which is interpreted to output x_{i^*}. In this model, for a prime c, we say that the c-inversion assumption holds, if for any PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ outputs $[c]_R^{-1}$ (i.e., $x_{i^*} = [c]_R^{-1}$) is negligible where $[c]_R$ denotes $c \cdot [1]_R$ and $^{-1}$ denotes a multiplicative inverse.

2.2 Generic Hardness of Inversion

Here, we describe how we give a lower bound for the c-inversion problem on the $\mathcal{D}_R$-GRM in the case where $\mathcal{D}_R$ is $\mathbb{Z}_p$ for randomly chosen ℓ-bit prime p. For obtaining the lower bound, we introduce two assumptions on NI-SLPs. Interestingly, both of these assumptions can be reduced to the factoring assumption w.r.t. unvalanced moduli, which is seemingly unrelevant to NI-SLPs. Finally, we give a partial evidence that the factoring assumption is necessary to prove the generic hardness of the c-inversion problem.

New assumptions on NI-SLP. Here, we describe our new assumptions on NI-SLPs, which we call the SLP assumption 1 and 2, respectively. We prove that they hold under the factoring assumption w.r.t. unbalanced moduli. These assumptions are defined as follows, where $\mathcal{P}_\ell$ denotes the set of all primes smaller than 2^ℓ.

SLP assumption 1: We say that the *SLP assumption 1* holds if for any PPT algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{SLP\text{-}1}}_{\mathcal{A}}(\lambda) := \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p \xleftarrow{\$} \mathcal{P}_\lambda; \mathrm{P}(\mathbb{Z}) \neq 0 \wedge \mathrm{P}(\mathbb{Z}_p) = 0]$ is negligible.

SLP assumption 2: We say that the SLP assumption 2 holds if for any PPT algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{A}}(\lambda) := \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; (\mathrm{P}(\mathbb{Z}_{p_1}) \stackrel{?}{=} 0) \neq (\mathrm{P}(\mathbb{Z}_{p_2}) \stackrel{?}{=} 0)]$ is negligible.

Intuitively, the SLP assumption 1 claims that for any efficiently generated NI-SLP P, if $\mathrm{P}(\mathbb{Z}) \neq 0$, then we also have $\mathrm{P}(\mathbb{Z}_p) \neq 0$ with overwhelming probability when $p \xleftarrow{\$} \mathcal{P}_\ell$ where $\mathcal{P}_\ell$ denotes the set of all primes smaller than 2^ℓ, and the SLP assumption 2 claims that for any efficiently

generated NI-SLP P, we have $(\mathrm{P}(\mathbb{Z}_{p_1}) \stackrel{?}{=} 0) = (\mathrm{P}(\mathbb{Z}_{p_2}) \stackrel{?}{=} 0)$ with overwhelming probability where $p_1, p_2 \xleftarrow{\$} \mathcal{P}_\ell$ and $(\mathrm{P}(\mathbb{Z}_{p_i}) \stackrel{?}{=} 0)$ is a variable in $\{0,1\}$ that is 1 if and only if $\mathrm{P}(\mathbb{Z}_{p_i}) = 0$ holds for $i = 1, 2$.

First, it is easy to see that the SLP assumption 2 is implied by the SLP assumption 1 because $\mathrm{P}(\mathbb{Z}_{p_1}) \neq 0$ or $\mathrm{P}(\mathbb{Z}_{p_2}) \neq 0$ implies $\mathrm{P}(\mathbb{Z}) \neq 0$. Moreover, we prove that the SLP assumption 1 holds assuming the hardness of factorizing an unbalanced modulus, i.e., $N = pq$ for primes p and q such that q is much larger than p. The proof outline is as follows. Suppose that there exists an adversary $\mathcal{A}$ that breaks the SLP assumption 1. Based on $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ that factorizes an unbalanced modulus N as follows. $\mathcal{B}$, given $N = pq$, runs $\mathcal{A}(1^\lambda)$ to obtain P. Since $\mathcal{A}$ breaks the SLP assumption 1, the probability that $\mathrm{P}(\mathbb{Z}) \neq 0$ and $\mathrm{P}(\mathbb{Z}_p) = 0$ hold is non-negligible. We can show that if q is chosen from a large enough set of primes, then $\mathrm{P}(\mathbb{Z}) \neq 0$ implies $\mathrm{P}(\mathbb{Z}_q) \neq 0$ with overwhelming probability. (See Lemma 2 for details.) In this case, we have $\gcd(\mathrm{P}(\mathbb{Z}_N), N) = p$ since $\mathrm{P}(\mathbb{Z}_p) = 0$ implies $p | \mathrm{P}(\mathbb{Z}_N)$ and $\mathrm{P}(\mathbb{Z}_q) \neq 0$ implies $q \not| \mathrm{P}(\mathbb{Z}_N)$. Thus $\mathcal{B}$ can factorize N by computing $\gcd(\mathrm{P}(\mathbb{Z}_N), N)$.

Generic hardness of inversion. Our main result can be stated as follows.

Theorem 1. *Let $\mathcal{D}_R$ be the distribution of $\mathbb{Z}_p$ for a randomly chosen prime p from $\mathcal{P}_\ell$. If the SLP assumption 1 holds, then the c-inversion assumption holds in the $\mathcal{D}_R$-GRM. Especially, if the factoring assumption w.r.t. unbalanced moduli holds, then the c-inversion assumption holds in the $\mathcal{D}_R$-GRM.*

In the following we give a proof sketch for the theorem. We consdier the following game hops.

Game 1 : This game is the original game that defines the c-inversion assumption. At the beginning of the game, a prime $p \xleftarrow{\$} \mathcal{P}_\ell$ is chosen and P is initialized to be an empty sequence. When $\mathcal{A}$ makes an operation query $(i, j, \circ)$, then $(i, j, \circ)$ is appended to P. When $\mathcal{A}$ makes an equal query (i, j), then the oracle returns true if $\mathrm{P}_i(\mathbb{Z}_p) = \mathrm{P}_j(\mathbb{Z}_p)$ and false otherwise. When $\mathcal{A}$ outputs i^*, we say that $\mathcal{A}$ wins if $\mathrm{P}_{i^*}(\mathbb{Z}_p) = [c]_{\mathbb{Z}_p}^{-1}$ holds.

Game 2: This game is the same as the above game except that the equality oracle uses a independently random prime p_k instead of p for the k-th equal query for each k. That is, when $\mathcal{A}$ makes the k-th equal query (i, j), the oracle picks an independently random prime $p_k \xleftarrow{\$} \mathcal{P}_\ell$ and returns true if $\mathrm{P}_i(\mathbb{Z}_{p_k}) = \mathrm{P}_j(\mathbb{Z}_{p_k})$ and false otherwise. We note that the winning condition of $\mathcal{A}$ remains unchanged, and it is decided by using p.

What we want to prove is that the $\mathcal{A}$'s winning probability in Game 1 is negligible. First, we show that the difference between its winning probabilities in Game 1 and Game 2 is negligible. In Game 2, responses by the oracle do not differ from the original game unless we have $(\mathrm{P}(\mathbb{Z}_{p_1}) \stackrel{?}{=} 0) \neq (\mathrm{P}(\mathbb{Z}_{p_2}) \stackrel{?}{=} 0)$. The probability that such an event occurs is negligible under the SLP assumption 2, which can be reduced to the SLP assumption 1 as mentioned in the previous paragraph. Thus this modification causes a negligible difference on $\mathcal{A}$'s winning probability. Next, we move on to proving $\mathcal{A}$'s winning probability in Game 2 is negligible. In Game 2, a prime p is not used at all except for deciding if $\mathcal{A}$ wins the game (i.e., $\mathrm{P}_{i^*}(\mathbb{Z}_p) = [c]_R^{-1}$). If we let $\mathrm{P}' := c \cdot \mathrm{P}_{i^*} - 1$ where $c \cdot \mathrm{P}_{i^*} - 1$ denotes the NI-SLP that computes $c \cdot \mathrm{P}_{i^*}(R) - [1]_R$ on any ring R, then the winning condition is equivalent to $\mathrm{P}'(\mathbb{Z}_p) = 0$. Here, it is easy to see that $\mathrm{P}'(\mathbb{Z}) \neq 0$ holds since 1 is not a multiple of c. Therefore under the SLP assumption 1, $\mathcal{A}$'s winning probability (i.e., $\Pr[\mathrm{P}'(\mathbb{Z}_p) = 0]$) is negligible. Thus, the c-inversion assumption holds in the $\mathcal{D}_R$-GRM. Finally, since the SLP assumptions 1 and 2 hold under the factoring assumption w.r.t. unbalanced moduli as mentioned in the previous paragraph, the generic hardness of the

c-inversion problem is proven under the same assumption.

Is factoring assumption necessary? For proving the generic hardness of the c-inversion assumption, we assumed the factoring assumption (or the SLP assumption 1, which is implied by the factoring assumption). One may wonder if this can be done without assuming any unproven assumption. We give a partial evidence that the factoring assumption (or some assumption of similar nature) is necessary. Specifically, we relate the hardness of the problem to the straight-line complexity [SS95] of factorials. For an integer N, the straight-line complexity $\tau(N)$ of N is defined as the minimum length of a NI-SLP P such that $\mathrm{P}(\mathbb{Z}) = N$. That is, the straight-line complexity is the smallest number of additions or multiplications to compute N starting from 1. Though it is conjectured that $\tau(n!)$ is super polynomial in $\log(n)$ [SS95, Che04, Mar13], that has been open for a long time. Here, we consider the uniform version of this conjecture.

Uniform Factorial Conjecture. . *There does not exist a PPT algorithm $\mathcal{A}$ that is given n as input and outputs an NI-SLP P^n such that $\mathrm{P}^n(\mathbb{Z}) = n!$ for all $n \in \mathbb{N}$.*

Though this conjecture is plausible, it still seems difficult to prove that this conjecture is true since this is just a uniform variant of the above mentioned long standing open problem. If the uniform factorial conjecture is false, then we can compute the characteristic $\mathrm{char}(R)$ of any ring R as long as $\mathrm{char}(R)$ is a prime by a simple binary search algorithm (See 5.4 for the detail). If $\mathrm{char}(R)$ is known, then the c-inversion assumption is trivially broken by the extended Eucledian algorithm. Therefore to prove the generic hardness of the c-inversion problem or even ensuring that hardness of computing a characteristic of a ring of a prime characteristic, we have to assume at least that the uniforml factorial conjecture is true, which seems difficult to unconditionally prove.

It has been widely recognized that there is a close relationship between the complexity of computing factorials and interger factorization (e.g., [Sha79, Che04]). Indeed, the uniform factorial conjecture can be easily proven true under the factoring assumption by a binary search algorithm similar to the one described above. On the other hand, we are unaware of any other standard assumption that would imply the uniform factorial conjecture. For example, even if we assume the existence of iO, it is not clear if the uniform factorial conjecture is true or not. Thus, if we want to reduce the generic hardness of the c-inversion problem to a standard assumption, it is somehow unavoidable to assume the factoring assumption (unless we find any other standard assumption that implies the uniform factorial conjecture).

2.3 Self-Bilinear Map from Inversion-Hard Ring

Here, we explain how to obtain a self-bilinear map based on an inversion-hard ring. Conceptually, the idea is quite simple: we consider the multiplication on the underlying ring to be a self-bilinear map. Actually, this idea was already presented by Kim et al. [KKS15], but they did not give a specific condition for a ring so that natural complexity assumptions for a resulting self-bilinear map hold. In this paper, we prove that the hardness of c-inversion problem for a prime c is sufficient to obtain useful assumptions for a resulting self-bilinear map. Specifically, we prove that several complexity assumptions for self-bilinear maps including the MCDH assumption hold if the c-inversion problem for some prime c is hard on an underlying ring. The proof is based on the very similar idea to classical results that reduce the CDH assumption to the factoring assumption [Shm85, McC88] and its extensions [HK09, Seu13, YYHK17]. We give a rough sketch of our construction and proof below.

We first describe the construction of a group G with a self-bilinear map e based on a ring R and prepare some notations. We let a group G be an additive subgroup of R generated

by $[1]_R$ and set the generator $[1]_G$ to be $[c]_R$. Then we define e by $e(X,Y) := X \cdot Y$ where $\cdot$ denotes the multiplication on R. For an integer $n \geq 2$, we let e_n denote a n-multilinear map obtained from a self-bilinear map e. Namely, e_n is defined by $e_2 := e$ and $e_k(X_1, \ldots, X_k) := e(e_{k-1}(X_1, \ldots, X_{k-1}), X_k)$ for $k \geq 3$. For $x \in \mathbb{Z}$, we let $[x]_G := x \cdot [1]_G$ and for $x \in \mathbb{Z}$ and $k \in \mathbb{N}$, $[x]_k$ denotes $x \cdot e_k([1]_G, \ldots, [1]_G)$. (If $k = 1$, then $[x]_1$ denotes $[x]_G$.) By using this notation, we have $e([x]_G, [y]_G) = [cxy]_G$, $[x]_k = [c^k x]_R$, $[x]_k + [y]_k = [x+y]_k$ and $e([x]_{k_1}, [x]_{k_2}) = [x \cdot y]_{k_1+k_2}$ for any $x, y \in \mathbb{Z}$ and $k, k_1, k_2 \in \mathbb{N}$.

For $n \in \mathbb{N}$, we say that the n-MCDH assumption holds if any PPT adversary given $[x_1]_1, \ldots, [x_{n+1}]_1$ cannot compute $[\prod_{i=1}^{n+1} x_i]_n$ with non-negligible probability where $x_1, \ldots, x_{n+1} \xleftarrow{\$} [2^{\ell+\lambda}]$ and 2^ℓ is an upper bound of $\mathrm{ord}(G)$ [4]. Then we prove the following theorem.

Theorem 2. *If the c-inversion assumption holds for the underlying ring R ,then for any integer $n \geq 2$, the n-MCDH assumption holds for (G, e).*

In this overview, we give a proof sketch for the case of $n = 2$ since the basic idea is the same for the general case. Suppose that there exists a PPT algorithm $\mathcal{A}$ that breaks the 2-MCDH assumption. Then we construct a PPT algorithm $\mathcal{B}$ that solves the c-inversion assumption. $\mathcal{B}$ is given $[1]_R$ and its goal is to compute $[c]_R^{-1}$. $\mathcal{B}$ sets $[1]_G := [c]_R$, picks $x'_i \xleftarrow{\$} [2^{\ell+\lambda}]$ and sets $[x_i]_G := [cx'_i + 1]_R$ for $i = 1, 2, 3$ where ℓ is an integer such that $\mathrm{ord}(G) < 2^\ell$. This implicitly defines $x_i = x'_i + c^{-1} \mod \mathrm{ord}(G)$ for $i = 1, 2, 3$. Since $(x'_i \mod \mathrm{ord}(G))$ is distributed almost uniformly and we have $\gcd(c, \mathrm{ord}(G)) = 1$, $(x'_i \mod \mathrm{ord}(G))$ is also almost uniformly distributed for $i = 1, 2, 3$. Thus if $\mathcal{B}$ gives $([1]_G, [x_1]_G, [x_2]_G, [x_3]_G)$ generated as above to $\mathcal{A}$ as input, then $\mathcal{A}$ outputs $T = [x_1 x_2 x_3]_2$ with non-negligible probability. Here, we have

$$\begin{aligned} T &= [x_1 x_2 x_3]_2 \\ &= [c^2(x'_1 + c^{-1})(x'_2 + c^{-1})(x'_3 + c^{-1})]_R \\ &= [c^2 x'_1 x'_2 x'_3 + c(x'_1 x'_2 + x'_2 x'_3 + x'_3 x'_1) + (x'_1 + x'_2 + x'_3) + c^{-1}]_R. \end{aligned}$$

Therefore if $\mathcal{B}$ subtracts $[c^2 x'_1 x'_2 x'_3 + c(x'_1 x'_2 + x'_2 x'_3 + x'_3 x'_1) + (x'_1 + x'_2 + x'_3)]_R$ from T, then it obtains $[c^{-1}]_R$ and succeeds in solving the c-inversion problem. Thus the 2-MCDH assumption holds under the c-inversion assumption. In Sec. 6 we show that we can apply the similar technique for more general assumptions.

2.4 Applications of Self-Bilinear Maps

Here, we give some applications of self-bilinear maps. As already observed in previous works [CL09, YYHK17], self-bilinear maps imply multilinear maps (or more generally, graded encodings [GGH13]) with unbounded multilinearity. By this feature, if we instantiate existing multi-linear map-based scheme by using self-bilinear maps, then the resulting scheme gains certain advantages. Here we give example of multiparty non-interactive key exchange (NIKE), broadcast encryption, and homomorphic signatures. We believe that self-bilinear maps have further applications.

Multiparty NIKE and broadcast encryption. Boneh and Silverberg [BS02] gave constructions of non-interactive key exchange (NIKE) and broadcast encryption schemes based on a multilinear map. If we instantiate these schemes by self-bilinear maps, the resulting schemes gain the following two advantages. First, we need not bound the maximum number of users at the setup phase. Second, the public key size of each user does not depend on the number of

[4] In our definition of the MCDH assumption, $x_1, \ldots, x_{n+1}$ are chosen from $[2^{\ell+\lambda}]$ instead of $[\mathrm{ord}(G)]$ because $\mathrm{ord}(G)$ is unknown in our setting. However, this causes a negligible difference (See Remark 4)

users. Boneh and Silverberg [BS02] also gave a construction of a multilinear-map-based broadcast encryption scheme based on the similar idea to their NIKE scheme. Similarly to NIKE, if we replace a multilinear map with a self-bilinear map in their construction, we obtain a broadcast encryption such that we need not bound the maximum number of users at the setup phase, and the sizes of ciphertext and public key of each user does not depend on the number of users. The formal descriptions of our schemes are given in Sec 7.1 and 7.2.

Homomorphic signatures. Here, we give a construction of homomorphic signatures based on a self-bilinear map. Our construction is based on the scheme proposed by Catalano et al. [CFW14] based on a multilinear map. Their scheme supports any polynomial of degree $p(\lambda)$ for a priori fixed polynomial p, and the security is proven under the $2p(\lambda)$-APMDH assumption. Our idea is to replace a multilinear map with a self-bilinear map. The reason the scheme in [CFW14] only supports polynomials of bounded polynomial degree is that existing multilinear maps only have a polynomially bounded level of multilinearity. On the other hand, a self-bilinear map gives an exponentially unbounded level multilinear map. Therefore if we instantiate the scheme in [CFW14] with such a multilinear map, the resulting scheme supports polynomials of exponentially unbounded degree, which is equivalent to polynomial size circuits of unbounded depth. As a result, we obtain fully homomorphic signatures that supports all polynomial size circuits of unbounded depth based on a self-bilinear map. We prove that our scheme is selectively secure under the ℓ-APMDH assumption for all ℓ such that $\log(\ell) = \mathsf{poly}(\lambda)$. Since we give a candidate construction of a self-bilinear map that satisfies the above assumption in Sec. 6, we can instantiate our scheme with our self-bilinear map. To the best of our knowledge, this is the first construction of a fully homomorphic signatures for unbounded depth circuits without relying on succinct non-interactive argument of knowledge SNARK [BCCT12]. The definition of fully homomorphic signatures and the full description of our scheme can be found in Sec 7.3.

3 Preliminaries

3.1 Notations

For any natural number n, $[n]$ denotes the set $\{1, \ldots, n\}$. $x \xleftarrow{\$} S$ denotes x is randomly chosen from a finite set S, and $x \xleftarrow{\$} \mathcal{A}(y)$ denotes that x is an output of a randomized algorithm $\mathcal{A}$ with input y. We say that a function $f(\cdot) : \mathbb{N} \to [0, 1]$ is negligible if for all positive polynomials $p(\cdot)$ and all sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$. We say that an algorithm $\mathcal{A}$ is probabilistic polynomial time (PPT) if there exists a polynomial p such that a running time of $\mathcal{A}$ with input length λ is less than $p(\lambda)$. We often use poly to denote an unspecified polynomial. For a polynomial p, $\deg(p)$ denotes the degree of p. $\mathcal{P}_\ell$ denotes the set of all primes smaller than 2^ℓ. For a ring R, $[1]_R$ denotes its multiplicative identity, for any integer n, $[n]_R$ denotes $n \cdot [1]_R$, and $\mathrm{char}(R)$ denotes the characteristic of R, i.e., the smallest positive integer n such that $[n]_R = [0]_R$. We often denote $[0]_R$ by 0. Similarly, for an additive cyclic group G with a generator $[1]_G$, for any integer n, $[n]_G$ denotes $n \cdot [1]_G$, and $\mathrm{ord}(G)$ denotes the order of G, i.e., the smallest positive integer n such that $[n]_G = 0$. We often denote $[0]_G$ by 0. We note that this bracket notation for G depends on the value of $[1]_G$, and thus it is well-defined only when $[1]_G$ is fixed.

3.2 Factoring Assumption

Here we define the factoring assumption. In this paper, unlike common style of the factoring assumption in cryptography, we assume the hardness of the factoring w.r.t. unbalanced moduli,

which is written as $N = pq$ for primes p and q such that p is much smaller than q. We note that the similar assumption was used in [Lip94]. The formal definition of our assumption is as follows.

Definition 1. *We say that the factoring assumption w.r.t. unbalanced moduli holds if for any PPT algorithm $\mathcal{A}$ and any polynomial* poly*,*

$$\Pr[p \xleftarrow{\$} \mathcal{P}_\lambda, q \xleftarrow{\$} \mathcal{P}_{\mathsf{poly}(\lambda)}; p \leftarrow \mathcal{A}(\lambda, pq)]$$

is negligible.

The best known factoring algorithm for RSA moduli of the form $N = pq$ where p is much smaller than q is the elliptic curve method [Len87], which works in sub-exponential time in $\log p \approx \lambda$. Therefore the above assumption is plausible.

3.3 Self-bilinear Map

Here, we define a self-bilinear map. We note that we use the additive notation rather than the multiplicative notation as in [CL09] for compatibility with a ring. Looking ahead, we construct a group with a self-bilinear map based on a ring and the group operation of the group is defined as the addition of the underlying ring.

Definition 2. *(Self-bilinear Map [CL09]) For a cyclic additive group G, a self-bilinear map $e : G \times G \to G$ has the following properties.*

- *For all $X, Y \in G$ and $\alpha \in \mathbb{Z}$, it holds that*

$$e(\alpha \cdot X, Y) = e(X, \alpha \cdot Y) = \alpha \cdot e(X, Y).$$

- *The map e is non-degenerate, i.e, if $X, Y \in G$ are generators of G, then $e(X, Y)$ is also a generator of G.*

A self bilinear map naturally extends to an n-multilinear map for any integer $n \geq 2$. This is done in the following recursive way: suppose that an n-multilinear map e_n can be constructed from a self-bilinear map e, then we can construct an $(n+1)$-multilinear map e_{n+1} by defining

$$e_{n+1}(X_1, \ldots, X_n, X_{n+1}) := e(e_n(X_1, \ldots, X_n), X_{n+1}).$$

We note that for any $X \in G$ and $n \in \mathbb{Z}$, $e_n(X, \ldots, X)$ can be computed by $O(\log(n))$ times evaluations of e in the similar way to the square-and-multiply algorithm for exponentiations. Especially, even if $n = 2^{\mathsf{poly}(\lambda)}$, $e_n(X, \ldots, X)$ can be computed in polynomial time in λ as long as e is computable in polynomial time in λ. Let $[1]_G$ be a generator of G. For $x \in \mathbb{Z}$ and $k \in \mathbb{N}$, $[x]_k$ denotes $x \cdot e_k([1]_G, \ldots, [1]_G)$. (If $k = 1$, then $[x]_1$ denotes $[x]_G = x \cdot [1]_G$.) By using this notation, for any $x, y \in \mathbb{Z}$ and $k, k_1, k_2 \in \mathbb{N}$, we have

$$[x]_k + [y]_k = [x + y]_k$$

and

$$e([x]_{k_1}, [y]_{k_2}) = [x \cdot y]_{k_1 + k_2}.$$

In this way, a self-bilinear map can be seen as a graded encoding system [GGH13] with unique encodings and unbounded levels. We note that values in a bracket can be reduced modulo $\mathrm{ord}(G)$, i.e., for any $x \in \mathbb{Z}$ and $k \in \mathbb{N}$, we have $[x]_k = [x \mod \mathrm{ord}(G)]_k$. We remark that this bracket notation depends on a generator $[1]_G$. Therefore when we consider a self-bilinear map on a group, we always clarify which element is set to be $[1]_G$ as a part of public parameters. In the above way, a self-bilinear map can be seen as a graded encoding system [GGH13] with unique encodings and exponentially unbounded levels.

4 Straight-line Program

Here, we define a straight-line program (SLP) which is a computation model over ring. Actually, we define a simplified variant of an SLP which we call a no-input SLP (NI-SLP). We define two complexity assumptions about NI-SLPs. We prove that these assumptions hold under the factoring assumption.

4.1 Definition

Here, we define an NI-SLP. An NI-SLP is defined as a usual SLP except that no ring element is given to a program as input. Actually, for an NI-SLP, a ring on which the computation is done itself is considered to be an "input" of the program. The following definition of an NI-SLP is the same as the definition of an SLP in [Bro16] except that no ring element is given to the program as input.

Definition 3. *(straight-line program with no-input) A no-input straight-line program (NI-SLP)* P *of length* L *is written as a sequence of instructions* $\mathrm{P} = (i_1, j_1, \circ_1), \ldots, (i_L, j_L, \circ_L)$ *where* $i_k, j_k \in \{0, 1, \ldots, k-1\}$ *and* $\circ_k \in \{+, -, \cdot\}$ *for all* $k \in [L]$*. On a ring* R*,* P *computes as follows.*

1. *Let* $x_0 := [1]_R$.
2. *For* $1 \leq k \leq L$*, let* $x_k := x_{i_k} \circ_k x_{j_k}$.
3. *Output* x_L.

We denote the output of P *on a ring* R *by* $\mathrm{P}(R)$.

Notations. For an NI-SLP P, $|\mathrm{P}|$ denotes the length of P, and P_k denotes the NI-SLP given by the sequence of the first k instructions of P. For any NI-SLPs $\mathrm{P}^1, \mathrm{P}^2$ and $\circ \in \{+, -, \cdot\}$, $\mathrm{P}^1 \circ \mathrm{P}^2$ denotes an NI-SLP that computes $\mathrm{P}^1(R) \circ \mathrm{P}^2(R)$ over any ring R. For any NI-SLP P and an integer $z \in \mathbb{Z}$, $z \cdot \mathrm{P}$ denotes an NI-SLP that computes $[z]_R \circ \mathrm{P}(R)$ for any ring R. It is easy to see that we can define them so that we have $|\mathrm{P}^1 \circ \mathrm{P}^2| = |\mathrm{P}^1| + |\mathrm{P}^2| + 1$ and $|z \cdot \mathrm{P}| \leq 2\log(z) + |\mathrm{P}|$ for any NI-SLPs $\mathrm{P}^1, \mathrm{P}^2, \mathrm{P}$ and integer z.

4.2 Complexity Assumptions

Here, we define complexity assumptions about an NI-SLP, which is used in the proof of our result in Sec. 5. Then we prove that these assumptions hold under the factoring assumption w.r.t. unbalanced moduli.

Intuitively, the first assumption claims that for any efficiently generated NI-SLP P, if $\mathrm{P}(\mathbb{Z}) \neq 0$, then $\mathrm{P}(\mathbb{Z}_p) \neq 0$ also holds with overwhelming probability when p is uniformly chosen from $\mathcal{P}_\lambda$.

Definition 4. *We say that the SLP assumption 1 holds if for any PPT algorithm* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{SLP\text{-}1}}_{\mathcal{A}}(\lambda) := \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p \xleftarrow{\$} \mathcal{P}_\lambda; \mathrm{P}(\mathbb{Z}) \neq 0 \wedge \mathrm{P}(\mathbb{Z}_p) = 0]$ *is negligible.*

The second assumption claims that for any efficiently generated NI-SLP P, we have $(\mathrm{P}(\mathbb{Z}_{p_1}) \stackrel{?}{=} 0) = (\mathrm{P}(\mathbb{Z}_{p_2}) \stackrel{?}{=} 0)$ with overwhelming probability where p_1 and p_2 are uniformly chosen from $\mathcal{P}_\lambda$.

Definition 5. *We say that the SLP assumption 2 holds if for any PPT algorithm* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{A}}(\lambda) := \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; (\mathrm{P}(\mathbb{Z}_{p_1}) \stackrel{?}{=} 0) \neq (\mathrm{P}(\mathbb{Z}_{p_2}) \stackrel{?}{=} 0)]$ *is negligible.*

We prove that the above assumptions hold under the factoring assumption w.r.t. unbalanced moduli.

Theorem 3. *If the factoring assumption w.r.t. unbalanced moduli holds, then the SLP assumption 1 and 2 hold.*

Before proving the above theorem, we prepare some lemmas.

Lemma 1. *For any NI-SLP* P *of length* L*, we have* $|\mathrm{P}(\mathbb{Z})| \leq 2^{2^{L-1}}$.

Proof. We prove the lemma by the induction. When $L = 1$ clearly we have $\mathrm{P}(\mathbb{Z}) \leq 2$. (The equation holds when $\mathrm{P} = (1, 1, +)$.) We assume that $|\mathrm{P}(\mathbb{Z})| \leq 2^{2^{L-1}}$ holds for all NI-SLP P of length smaller than L. Suppose that P is written as $\mathrm{P} = (i_1, j_1, \circ_1), \ldots, (i_{L+1}, j_{L+1}, \circ_{L+1})$. Then we let $\mathrm{P}_1 = (i_1, j_1, \circ_1), \ldots, (i_{i_{L+1}}, j_{i_{L+1}}, \circ_{i_{L+1}})$ and $\mathrm{P}_2 = (i_1, j_1, \circ_1), \ldots, (i_{j_{L+1}}, j_{j_{L+1}}, \circ_{j_{L+1}})$. Then clearly we have $\mathrm{P}(\mathbb{Z}) = \mathrm{P}_1(\mathbb{Z}) \circ_{L+1} \mathrm{P}_2(\mathbb{Z})$. By the assumption, we have $|\mathrm{P}_i(\mathbb{Z})| \leq 2^{2^{L-1}}$ for $i = 1, 2$. Therefore we have $|\mathrm{P}(\mathbb{Z})| = |\mathrm{P}_1(\mathbb{Z})| \circ_{L+1} |\mathrm{P}_2(\mathbb{Z})| \leq 2^{2^{L-1}} \circ_{L+1} 2^{2^{L-1}} \leq 2^{2^L}$. Therefore the lemma also holds for an NI-SLP with length $L + 1$ and thus the lemma is proven. □
□

Lemma 2. *For any NI-SLP* P *of length* L*,* $\Pr[q \xleftarrow{\$} \mathcal{P}_{2L}; \mathrm{P}(\mathbb{Z}) \neq 0 \wedge \mathrm{P}(\mathbb{Z}_q) = 0]$ *is negligible in* L.

Proof. By Lemma 1, we have $\mathrm{P}(\mathbb{Z}) \leq 2^{2^{L-1}}$. Therefore the number of primes that divides $\mathrm{P}(\mathbb{Z})$ is at most 2^{L-1}. On the other hand, by the prime number theorem, the number of primes smaller than 2^{2L} is $\Omega(2^{2L}/L)$. Therefore when q is uniformly chosen from primes smaller than 2^{2L}, the probability that q divides $\mathrm{P}(\mathbb{Z})$ is $O(L/2^L)$, which is negligible. Clearly, $q|\mathrm{P}(\mathbb{Z})$ and $\mathrm{P}(\mathbb{Z}_q) = 0$ are equivalent, and thus the probability that $\mathrm{P}(\mathbb{Z}_q) = 0$ holds is negligible. □
□

Lemma 3. *If the SLP assumption 1 holds, then the SLP assumption 2 holds.*

Proof. Let $\mathcal{A}$ be any algorithm. Then we have

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{A}}(\lambda) \\
&= \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; (\mathrm{P}(\mathbb{Z}_{p_1}) \stackrel{?}{=} 0) \neq (\mathrm{P}(\mathbb{Z}_{p_2}) \stackrel{?}{=} 0)] \\
&= \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; (\mathrm{P}(\mathbb{Z}_{p_1}) = 0 \wedge \mathrm{P}(\mathbb{Z}_{p_2}) \neq 0) \vee (\mathrm{P}(\mathbb{Z}_{p_1}) \neq 0 \wedge \mathrm{P}(\mathbb{Z}_{p_2}) = 0)] \\
&\leq \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; (\mathrm{P}(\mathbb{Z}_{p_1}) = 0 \wedge \mathrm{P}(\mathbb{Z}_{p_2}) \neq 0)] \\
&\quad + \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; (\mathrm{P}(\mathbb{Z}_{p_1}) \neq 0 \wedge \mathrm{P}(\mathbb{Z}_{p_2}) = 0)] \\
&= 2 \cdot \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; \mathrm{P}(\mathbb{Z}_{p_1}) = 0 \wedge \mathrm{P}(\mathbb{Z}_{p_2}) \neq 0] \\
&\leq 2 \cdot \Pr[\mathrm{P} \leftarrow \mathcal{A}(1^\lambda), p_1, p_2 \xleftarrow{\$} \mathcal{P}_\lambda; \mathrm{P}(\mathbb{Z}_{p_1}) = 0 \wedge \mathrm{P}(\mathbb{Z}) \neq 0] \\
&= 2 \cdot \mathsf{Adv}^{\mathsf{SLP\text{-}1}}_{\mathcal{A}}(\lambda).
\end{aligned}
$$

Therefore if the SLP assumption 1 holds, then the SLP assumption 2 holds. □

Then we prove Theorem 3.

Proof. (of Theorem 3.) By Lemma 3, what we have to prove is that the SLP assumption 1 holds under the factoring assumption w.r.t. unbalanced moduli. Let $\mathcal{A}$ be a PPT adversary that breaks the SLP assumption 1 whose running time is at most $T(\lambda)$. Then we construct an algorithm $\mathcal{B}$ that factorizes $N = pq$ where $p \xleftarrow{\$} \mathcal{P}_\lambda$ and $q \xleftarrow{\$} \mathcal{P}_{\mathsf{poly}(\lambda)}$ for $\mathsf{poly} = 2T$. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}(\ell, N)$**:** It computes $\mathrm{P} \leftarrow \mathcal{A}(1^\ell)$ and $s = \mathrm{P}(\mathbb{Z}_N)$, and output $\gcd(s, N)$.

We show the correctness of the above algorithm. First, we have

$$\begin{aligned}
&\Pr[\mathrm{P}(\mathbb{Z}_p) = 0 \wedge \mathrm{P}(\mathbb{Z}_q) \neq 0] \\
&= \Pr[\mathrm{P}(\mathbb{Z}_p) = 0 \wedge \mathrm{P}(\mathbb{Z}) \neq 0] - \Pr[\mathrm{P}(\mathbb{Z}_p) = 0 \wedge \mathrm{P}(\mathbb{Z}_q) = 0 \wedge \mathrm{P}(\mathbb{Z}) \neq 0] \\
&> \mathsf{Adv}^{\mathsf{SLP\text{-}1}}_{\mathcal{A}}(\lambda) - \Pr[\mathrm{P}(\mathbb{Z}_q) = 0 \wedge \mathrm{P}(\mathbb{Z}) \neq 0].
\end{aligned}$$

Since P is generated by $\mathcal{A}$ whose running time is at most $T(\lambda)$, the length of P is at most $T(\lambda)$. Therefore by Lemma 2, $\Pr[\mathrm{P}(\mathbb{Z}_q) = 0 \wedge \mathrm{P}(\mathbb{Z}) \neq 0]$ is negligible. Therefore, if $\mathsf{Adv}^{\mathsf{SLP\text{-}1}}_{\mathcal{A}}(\lambda)$ is non-negligible, then $\Pr[\mathrm{P}(\mathbb{Z}_p) = 0 \wedge \mathrm{P}(\mathbb{Z}_q) \neq 0]$ is also non-negligible. Therefore with non-negligible probability, we have $\gcd(s, N) = p$, and $\mathcal{B}$ succeeds in factoring N. □

5 Inversion-hard Ring

In this section, we first formalize a ring with efficient operations as a *ring scheme*, and for any prime c, define the c-inversion assumption for it. Next, we define the GRM that can capture unknown characteristic rings. Then we prove that for any prime c, the c-inversion assumption holds in the GRM under the factoring assumption for a certain ring distribution. Finally, we show an evidence that a certain complexity assumption is necessary for proving the generic hardness of the c-inversion assumption.

5.1 Ring Scheme

Here, we define a ring scheme and the c-inversion assumption for it. A ring scheme consists of PPT algorithms $(\mathsf{RGen}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Mult})$.

$\mathsf{RGen}(1^\lambda) \to PP$: This algorithm takes the security parameter 1^λ as input and outputs the public parameter PP, which specifies a ring R whose characteristic is at most 2^ℓ and its multiplicative identity $[1]_R$. We assume that $[1]_R$ can be computed efficiently from PP. Though PP is input to all other algorithms, we omit them for simplicity.

$\mathsf{Add}(X, Y) \to X + Y$: This algorithm takes two elements $X, Y \in R$ as input and outputs $X + Y \in R$.

$\mathsf{Sub}(X, Y) \to X - Y$: This algorithm takes two elements $X, Y \in R$ as input and outputs $X - Y \in R$.

$\mathsf{Mult}(X, Y) \to X \cdot Y$: This algorithm takes two elements $X, Y \in R$ as input and outputs $X \cdot Y \in R$.

c-inversion assumption. Here, for a prime c, we define the c-inversion assumption for a ring scheme. We say that the c-inversion assumption holds for a ring scheme if no PPT algorithm can compute $[c]_R^{-1}$ where $^{-1}$ denotes the multiplicative inverse.

Definition 6. *Let c be a prime (that may depend on λ). The c-inversion assumption holds for a ring scheme* $(\mathsf{RGen}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Mult})$ *if for any PPT algorithm $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathsf{inv}}_{\mathcal{A},c}(\lambda) := \Pr[PP \xleftarrow{\$} \mathsf{GGen}(1^\lambda) : [c]_R^{-1} \xleftarrow{\$} \mathcal{A}(PP, c)]$$

is negligible where R denotes the ring specified by PP.

Remark 1. *We can also define a problem to compute* $[x]_R^{-1}$ *given* $[x]_R$ *for a randomly chosen ring element* x *similarly to the case for groups with infeasible inversion [Hoh03, Mol03, ILOP04, AC18] (We call the problem random-inversion problem.) Actually, if* c *is coprime to the characteristic of* R *and almost all elements of* R *is invertible, then the* c*-inversion assumption on the ring implies that the random-inversion problem is hard. We briefly describe the reduction below. A reduction algorithm randomly chooses* $[x']_R \xleftarrow{\$} R$*, set* $[x]_R := [c]_R \cdot [x']_R$ *and gives* $[x]_R$ *to an algorithm that solves the random-inversion assumption with non-negligible probability. Since we assume that* c *is coprime to the characteristic of* R *and almost all elements of* R *are invertible, the distribution of* $[x]_R$ *is almost uniform on* R*, and thus the algorithm returns* $[x]_R^{-1} = [c]_R^{-1} \cdot [x']_R^{-1}$ *with non-negligible probability. Then the reduction algorithm multiply this by* $[x']_R$ *to obtain* $[c^{-1}]_R$*, and breaks the* c*-inversion assumption. This completes the reduction.*

5.2 Generic Ring Model

Here, we extend the GRM to capture rings of unknown characteristics. Our GRM is defined by modifying the model given by Jager and Schwenk [JS09] in the following three aspects. First, our model is parametrized by a distribution $\mathcal{D}_R$ of a ring, and ring is chosen according to $\mathcal{D}_R$ whereas a ring is fixed in the model in [JS09]. Second, while their model allowed a generic algorithm to compute a multiplicative inverse, our model does not allow this because that cannot be computed on a ring of unknown characteristic. Third, while a random ring element is chosen as a "secret value" at the beginning of the game in their model, there is no secret value in our model. This is because the only problem we consider in our model is the c-inversion problem that does not have any secret value. (In our setting, the secret is a ring itself.) The formal definition of our generic ring model is as follows.

Definition of $\mathcal{D}_R$-generic ring model. Let $\mathcal{D}_R = \{\mathcal{D}_R(\lambda)\}_{\lambda\in\mathbb{N}}$ be a sequence of a distribution $\mathcal{D}_R(\lambda)$ of a ring R. In the $\mathcal{D}_R$-generic ring model, a generic algorithm $\mathcal{A}$ is given 1^λ as input and given access to an oracle $\mathcal{O}_{\mathsf{ring}}$ described below.

$\mathcal{O}_{\mathsf{ring}}$ picks a ring R according to the distribution $\mathcal{D}_R(\lambda)$. It maintains an NI-SLP P, which is set to be empty sequence at the beginning. $\mathcal{A}$ is allowed to make the following two kinds of query, the $\mathsf{operation}$ and equal queries.

$\mathsf{operation}$ **query:** When $\mathcal{A}$ makes an $\mathsf{operation}$ query $(i, j, \circ) \in \{0, \ldots, |\mathrm{P}|\} \times \{0, \ldots, |\mathrm{P}|\} \times \{+, -, \cdot\}$, then $\mathcal{O}_{\mathsf{ring}}$ append $(i, j, \circ)$ to P.

equal **query:** When $\mathcal{A}$ makes an equal query $(i, j) \in \{0, \ldots, |\mathrm{P}|\} \times \{0, \ldots, |\mathrm{P}|\}$, then $\mathcal{O}$ returns true if $\mathrm{P}_i(R) = \mathrm{P}_j(R)$ and false otherwise.

We define the c-inversion assumption in this model.

Definition 7. *We say that the* c*-inversion assumption holds in the* $\mathcal{D}_R$*-generic model if for any PPT generic algorithm* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{gen\text{-}inv}}_{\mathcal{A},\mathcal{D}_R,c}(\lambda) := \Pr[i^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{ring}}}(1^\lambda, c) : \mathrm{P}_{i^*}(R) = [c]_R^{-1}]$$

is negligible where $\mathcal{O}_{\mathsf{ring}}$ *denotes the oracle in the* $\mathcal{D}_R$*-generic model,* R *denotes a ring chosen by* $\mathcal{O}_{\mathsf{ring}}$ *and* P *denotes an NI-SLP maintained by* $\mathcal{O}_{\mathsf{ring}}$ *at the end of the game.*

5.3 Generic Hardness of the c-inversion Assumption

Here, we prove that the c-inversion assumption holds for any prime c in the $\mathcal{D}_R$-generic ring model if the factoring assumption holds where $\mathcal{D}_R$ is the distribution of $R = \mathbb{Z}_p$ for randomly chosen prime $p \xleftarrow{\$} \mathcal{P}_\ell$.

Theorem 4. *(Parametrized version of Theorem 1.) Let $\mathcal{D}_R$ be the distribution of $R = \mathbb{Z}_p$ for randomly chosen prime $p \xleftarrow{\$} \mathcal{P}_\ell$. Then for any prime c and PPT generic algorithm $\mathcal{A}$, there exist PPT algorithms $\mathcal{B}$ and $\mathcal{C}$ such that the following holds:*

$$\mathsf{Adv}^{\mathsf{gen-inv}}_{\mathcal{A},\mathcal{D}_R,c}(\lambda) \leq Q \cdot \mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{B}}(\ell) + \mathsf{Adv}^{\mathsf{SLP\text{-}1}}_{\mathcal{C}}(\ell)$$

where Q is the maximum number of $\mathcal{A}$'s equal *query. Especially, if $\ell = \Omega(\lambda)$ and the factoring assumption w.r.t. unbalanced moduli holds, then the c-inversion assumption holds in the $\mathcal{D}_R$-generic ring model.*

Remark 2. *In an asymptotic sense, we can set $\ell = \lambda$. However, since we only give reductions for SLP assumptions 1 and 2 from the factoring, for which there exists a sub-exponential time attack, there may exist sub-exponential attack against these assumptions. In that case, we have to set ℓ larger than λ (say, λ^3) for obtaining λ-bit security.*

Proof. (of Theorem 4.) The latter part easily follows from the former part and Theorem 3. We prove the former part by considering the following sequence of games for a generic algorithm $\mathcal{A}$.

Game 1**:** This game is the original game that defines the c-inversion assumption as described in Sec. 5.2. For clarity, we describe the game again. In this game, $\mathcal{O}_{\mathsf{ring}}$ picks $p \xleftarrow{\$} \mathcal{P}_\ell$ and initialize P to be an empty sequence. When $\mathcal{A}$ makes an operation query $(i, j, \circ)$, then $\mathcal{O}_{\mathsf{ring}}$ append $(i, j, \circ)$ to P. When $\mathcal{A}$ makes an equal query (i, j), then $\mathcal{O}_{\mathsf{ring}}$ returns true if $\mathrm{P}_i(\mathbb{Z}_p) = \mathrm{P}_j(\mathbb{Z}_p)$ and false otherwise. When $\mathcal{A}$ outputs i^*, we say that $\mathcal{A}$ wins if $\mathrm{P}_{i^*}(\mathbb{Z}_p) = [c]^{-1}_{\mathbb{Z}_p}$ holds.

Game 2**:** This game is the same as the above game except that $\mathcal{O}_{\mathsf{ring}}$ uses a independently random prime p_k instead of p for the k-th equal query for each k. That is, when $\mathcal{A}$ makes the k-th equal query (i, j), the oracle $\mathcal{O}_{\mathsf{ring}}$ picks an independently random prime $p_k \xleftarrow{\$} \mathcal{P}_\ell$ and returns true if $\mathrm{P}_i(\mathbb{Z}_{p_k}) = \mathrm{P}_j(\mathbb{Z}_{p_k})$ and false otherwise. We note that the winning condition of $\mathcal{A}$ remains unchanged, and it is decided by using p.

Let T_k be the event that $\mathcal{A}$ wins in Game k for $k = 1, 2$. Clearly, we have $\mathsf{Adv}^{\mathsf{gen-inv}}_{\mathcal{A},\mathcal{D}_R,c}(\lambda) = \Pr[T_1]$. We prove that $\Pr[T_1]$ is negligible by the following lemmas. Let Q be the maximum number of $\mathcal{A}$'s equal query.

Lemma 4. *There exists a PPT algorithm $\mathcal{B}$ such that $|\Pr[T_2] - \Pr[T_1]| \leq Q \cdot \mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{B}}(\ell)$ holds.*

Proof. For $0 \leq k \leq Q$, we consider hybrids H_k. where the oracle $\mathcal{O}_{\mathsf{ring}}$ works as in Game 2 until the k-th query, and as in Game 1 from the $(k+1)$-th query. Then it is clear that H_0 is equivalent to Game 1 and H_Q is equivalent to Game 2. Let S_k be the event that $\mathcal{A}$ wins in H_k. By the triangle inequality, we have $|\Pr[T_2] - \Pr[T_1]| \leq \sum_{k=1}^{Q} |\Pr[S_k] - \Pr[S_{k-1}]|$. In the following, we prove that for any $1 \leq k \leq Q$, $|\Pr[S_k] - \Pr[S_{k-1}]| \leq \mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{B}}$ holds. To do so, we construct an adversary $\mathcal{B}$ against the SLP assumption 2 by using $\mathcal{A}$. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}(1^\lambda)$**:** $\mathcal{B}$ gives 1^λ to $\mathcal{A}$ as input. Then it simulates the oracle $\mathcal{O}_{\mathsf{ring}}$ in H_{k-1} and H_k until $\mathcal{A}$ makes k-th equal query (i, j) without choosing p. (We note that H_{k-1} and H_k are identical until this point. We also note that p is not needed until this point, and thus $\mathcal{B}$ can simulate it without choosing p.) Let P be an NI-SLP maintained by $\mathcal{O}_{\mathsf{ring}}$ at this point. $\mathcal{B}$ appends $(i, j, -)$ to P to generate P′ and outputs P′.

Then we prove that $\mathcal{B}$ works correctly. Let p_k be a prime randomly chosen from $\mathcal{P}_\ell$ that is supposed to be chosen by $\mathcal{O}_{\mathsf{ring}}$ when responding the k-th equal query in H_k. We remark that $\mathcal{B}$ halts before simulating the oracle for the k-th equal query, and thus it works independently

of p_k. Let F_k be the event that $(\mathrm{P}_i(\mathbb{Z}_p) = \mathrm{P}_j(\mathbb{Z}_p) \wedge \mathrm{P}_i(\mathbb{Z}_{p_k}) \neq \mathrm{P}_j(\mathbb{Z}_{p_k}))$ or $(\mathrm{P}_i(\mathbb{Z}_p) \neq \mathrm{P}_j(\mathbb{Z}_p) \wedge \mathrm{P}_i(\mathbb{Z}_{p_k}) = \mathrm{P}_j(\mathbb{Z}_{p_k}))$ holds where P denotes an NI-SLP maintained by $\mathcal{O}_{\mathsf{ring}}$ at the point of $\mathcal{A}$'s k-th equal query. H_{k-1} and H_k are completely the same unless F_k occurs. Therefore we have $|\Pr[S_k] - \Pr[S_{k-1}]| \leq \Pr[F_k]$. If F_k occurs, then we have $(\mathrm{P}'(\mathbb{Z}_p) = 0 \wedge \mathrm{P}'(\mathbb{Z}_{p_k}) \neq 0)$ or $(\mathrm{P}'(\mathbb{Z}_p) \neq 0 \wedge \mathrm{P}'(\mathbb{Z}_{p_k}) = 0)$. By the definition, the probability this event occurs is $\mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{B}}(\ell)$. (We remark again that $\mathcal{B}$ works independently of p and p_k.) Then we have $|\Pr[S_k] - \Pr[S_{k-1}]| \leq \Pr[F_k] \leq \mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{B}}(\ell)$ and thus the lemma follows. □

□

Lemma 5. *There exists a PPT algorithm $\mathcal{C}$ such that $\Pr[T_2] \leq \mathsf{Adv}^{\mathsf{SLP\text{-}1}}_{\mathcal{C}}(\ell)$ holds.*

Proof. We construct an adversary $\mathcal{C}$ against the SLP assumption 1 by using $\mathcal{A}$. the description of $\mathcal{C}$ is as follows.

$\mathcal{C}(1^\lambda)$**:** $\mathcal{C}$ simulates Game 2 for $\mathcal{A}$ until $\mathcal{A}$ outputs i^* without choosing p. (We note that Game 2 can be simulated without using p until deciding if $\mathcal{A}$ wins.) Let P be an NI-SLP maintained by $\mathcal{O}_{\mathsf{ring}}$ at this point. $\mathcal{B}$ sets $\mathrm{P}' := c \cdot \mathrm{P}_{i^*} - 1$ and outputs P'.

We prove that $\mathcal{C}$ works correctly. If $\mathcal{A}$ wins in Game 2, $\mathrm{P}_{i^*}(\mathbb{Z}_p) = [c]^{-1}_{\mathbb{Z}_p}$ holds, and equivalently, $\mathrm{P}'(\mathbb{Z}_p) = 0$ holds for P' output by $\mathcal{C}$. By the definition of P', we have $\mathrm{P}'(\mathbb{Z}) \neq 0$. Therefore, the probability that $\mathrm{P}'(\mathbb{Z}_p) = 0$ holds is $\mathsf{Adv}^{\mathsf{SLP\text{-}2}}_{\mathcal{C}}(\ell)$. (We remark that $\mathcal{C}$ works independently of p.) Thus the lemma follows. □

□

The theorem is proven by combining the above lemmas. □

□

Remark 3. *We can also rule out generic ring algorithms based on essentially the same proof that compute $b \cdot [c^{-1}]_R$ for integers b, c adaptively chosen by the algorithm such that c does not divide b. We gave the proof for the simpler case where $b = 1$ and c is a fixed prime because this is sufficient for the application for self-bilinear maps given in Sec. 6.*

5.4 Is Factoring Assumption Necessary?

For proving the generic hardness of the c-inversion assumption, we assume the factoring assumption. One may wonder if this can be done without assuming any unproven assumption. Here, we give a partial evidence that the factoring assumption (or some assumption of similar nature) is needed. Specifically, we relate the hardness of the problem to the straight-line complexity [SS95] of factorials. For an integer N, the straight-line complexity $\tau(N)$ of N is defined as the minimum length of a NI-SLP P such that $\mathrm{P}(\mathbb{Z}) = N$. That is, the straight-line complexity is the smallest number of additions or multiplications to compute N starting from 1. Though it is conjectured that $\tau(n!)$ is super polynomial in $\log(n)$ [SS95, Che04, Mar13], this has been open for a long time. Here, we consider the uniform version of this conjecture [5].

Uniform Factorial Conjecture. . *There does not exist a PPT algorithm $\mathcal{A}$ that is given n as input and outputs an NI-SLP P^n such that $\mathrm{P}^n(\mathbb{Z}) = n!$ for all $n \in \mathbb{N}$.*

Though this conjecture is plausible, it still seems difficult to prove that this conjecture is unconditionally true since this is just a uniform variant of the above mentioned long standing open problem . Here, we prove that if the above conjecture is false, i.e., if there exists a PPT algorithm $\mathcal{A}$ that is given n as input and outputs an NI-SLP P^n such that $\mathrm{P}^n(\mathbb{Z}) = n!$ for all

[5] A related discussion can be found in [Che04, Sec. 5]

$n \in \mathbb{N}$, then we can compute the characteristic $\mathrm{char}(R)$ of any ring R in polynomial time in $\log(\mathrm{char}(R))$ as long as $\mathrm{char}(R)$ is a prime. If $\mathrm{char}(R)$ is known, then the c-inversion assumption is trivially broken by the extended Eucledian algorithm. The algorithm to compute $\mathrm{char}(R)$ is given as follows. For any integer n, let P^n be an NI-SLP that computes $n!$ on $\mathbb{Z}$ such that $|\mathrm{P}^n| = \mathsf{poly}(\log n)$. Let R be a ring whose characteristic is a prime smaller than 2^ℓ. First, we consider the following recursively defined procedure.

$\mathsf{procedure}(A, B)$**:** If $A = B$, it outputs A. Else it computes $C := \frac{A+B}{2}$ and $\mathrm{P}^C \xleftarrow{\$} \mathcal{A}(C)$. If $\mathrm{P}^C(R) = 0$, then do $\mathsf{procedure}(A, C)$. Else do $\mathsf{procedure}(C, B)$.

By using this procedure, $\mathrm{char}(R)$ can be computed by executing $\mathsf{procedure}(0, 2^\ell)$. This can be seen by the fact that $\mathrm{P}^C(R) = 0$ if and only if $C \geq \mathrm{char}(R)$ and thus this algorithm gives the binary search for $\mathrm{char}(R)$. Next, we verify that this algorithm runs in polynomial time in ℓ. It is easy to see that the recursion depth of $\mathsf{procedure}$ is ℓ. In each execution of $\mathsf{procedure}$, $\mathrm{P}^C(R)$ is computed once for some $0 \leq C \leq 2^\ell$. Since we have $|\mathrm{P}^C| = \mathsf{poly}(\log(C)) \leq \mathsf{poly}(\ell)$ by the assumption, the whole algorithm runs in polynomial time in ℓ.

Therefore to prove the generic hardness of the c-inversion problem or even ensuring that hardness of computing a characteristic of a ring of a prime characteristic, we have to assume at least that the uniform factorial conjecture is true, which seems difficult to unconditionally prove.

It has been widely recognized that there is a close relationship between the complexity of computing factorials and interger factorization (e.g., [Sha79, Che04]). Indeed, the uniform factorial conjecture can be easily reduced to the factoring assumption by a binary search algorithm similar to the one described above. On the other hand, we are unaware of any other standard assumption that would imply the uniform factorial conjecture. For example, even if we assume the existence of iO, it is not clear if the uniform factorial conjecture is true or not. Thus, if we want to reduce the generic hardness of the c-inversion problem to a standard assumption, it is somehow unavoidable to assume the factoring assumption (unless we find any other standard assumption that implies the uniform factorial conjecture).

6 Self-bilinear Map

In this section, we formalize a group with efficiently computable self-bilinear map as a *self-bilinear group scheme*. We define several complexity assumptions including the MCDH assumption for a self-bilinear map group scheme. We construct a self-bilinear group scheme that satisfies those assumptions based on a ring scheme that satisfies the c-inversion assumption for a prime c. We also give a definition of the *generic self-bilinear map model* (GSBM). By combining with the result in Sec. 5, we show that several assumptions holds in the model under the factoring assumption w.r.t. unbalanced moduli.

6.1 Self-bilinear group scheme.

Here, define a self-bilinear group scheme. A self-bilinear map group scheme consists of PPT algorithms $(\mathsf{GGen}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Map})$.

$\mathsf{GGen}(1^\lambda) \to PP$: This algorithm takes the security parameter 1^λ as input and outputs public parameters PP which specifies a cyclic group G whose order is at most 2^ℓ, a generator $[1]_1$ of G and a self-bilinear map e on G. e is required to satisfy the properties given in Def. 2. PP is input to all other algorithms below, but we omit them for simplicity.

$\mathsf{Add}(X, Y) \to X + Y$: This algorithm takes $X, Y \in G$ as input and outputs $X+Y$. Equivalently, if $X = [x]_k$ and $[y]_k$ for $x, y, k \in \mathbb{Z}$, then it outputs $[x + y]_k$.

$\mathsf{Sub}(X, Y) \to -X$: This algorithm takes two elements $X, Y \in G$ as input and outputs $X + Y$. Equivalently, if $X = [x]_k$ and $[y]_k$ for $x, y, k \in \mathbb{Z}$, then it outputs $[x - y]_k$.

$\mathsf{Map}(X, Y) \to e(X, Y)$: This algorithm takes two elements $X, Y \in G$ as input and outputs $e(X, Y)$. Equivalently, if $X = [x]_{k_1}$ and $[y]_{k_2}$ for $x, y, k_1, k_2 \in \mathbb{Z}$, then it outputs $[x \cdot y]_{k_1+k_2}$.

Hardness assumptions. Here, we define hardness assumptions for a self-bilinear group. First, we define the multilinear computational Diffie-Hellman (MCDH) assumption, which was originally defined for multilinear maps [GGH13]. We extend the definition to the one for self-blinear maps.

Definition 8. *n-MCDH assumption. For $n \in \mathbb{N}$, we say that the n-multilinear computational Diffie-Hellman (n-MCDH) assumption holds if for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[[\prod_{i=1}^{n+1} x_i]_n \leftarrow \mathcal{A}(PP, \{[x_i]_1\}_{i\in[n+1]})\right]$$

is negligible where $PP \leftarrow \mathsf{GGen}(1^\lambda)$ and $x_1, \ldots, x_{n+1} \xleftarrow{\$} [2^{\ell+\lambda}]$.

Remark 4. *The assumption is equivalent if we choose $x_1, \ldots, x_{n+1} \xleftarrow{\$} [\mathrm{ord}[G]]$ since the statistical distance of these distributions are negligible. Since we cannot compute $\mathrm{ord}(G)$ efficiently from PP, we adopt this definition. The same remark is applied to the all other assumptions w.r.t. self-bilinear maps defined in this paper.*

Next, we define the augmented power multilinear Diffie-Hellman (APMDH) assumption, which was originally defined for multilinear maps [CFW14]. We extend the definition to the one for self-blinear maps.

Definition 9. *ℓ-APMDH assumption. For $\ell \in \mathbb{N}$, we say that the ℓ-augmented power multilinear Diffie-Hellman (ℓ-APMDH) assumption holds if for any PPT adversary $\mathcal{A}$,*

$$\Pr[[x_1^{\ell-1}(x_2x_3)^\ell]_\ell \xleftarrow{\$} \mathcal{A}(PP, [x_1]_1, [x_2]_1, [x_3]_1, [x_1x_2]_1, [x_1x_3]_1, [x_1x_2x_3]_1)]$$

is negligible where $PP \leftarrow \mathsf{GGen}(1^\lambda)$ and $x_1, x_2, x_3 \xleftarrow{\$} [2^{\ell+\lambda}]$.

Next, we define more general parametrized assumption which we call the multilinear generalized Diffie-Hellman (MGDH) assumption. This can be seen as a family of Diffie-Hellman type computational assumptions including the MCDH and APMDH assumptions. This is an analogue of the multilinear generalized Diffie-Hellman with auxiliary information (AI-MGDH) assumption defined for self-bilinear maps with auxiliary information [YHK16].

Definition 10. *$(\mathcal{F}, f^*, \ell^*)$-MGDH assumption. Let $\mathcal{F}$ be a set of monic monomials with n variables, f^* be a monic monomial with n variables, and ℓ^* be a natural number. We say that the $(\mathcal{F}, f^*, \ell^*)$-multilinear generalized Diffie-Hellman ($(\mathcal{F}, f^*, \ell^*)$-MGDH) assumption holds if for any PPT adversary $\mathcal{A}$,*

$$\Pr[[f^*(x_1, \ldots, x_n)]_{\ell^*} \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_\mathcal{F}}(PP)]$$

is negligible where $PP \leftarrow \mathsf{GGen}(1^\lambda)$, $x_1, \ldots, x_n \xleftarrow{\$} [2^{\ell+\lambda}]$ and $\mathcal{O}_\mathcal{F}$ is an oracle which is given $f \in \mathcal{F}$ and returns $[f(x_1, \ldots, x_n)]_1$.

6.2 Construction based on Inversion-hard Ring.

Here, we give a construction of a self-bilinear group scheme based on a ring scheme. Then we show that if the c-inversion assumption for the underlying ring scheme holds, then various hardness assumptions including the MCDH assumption and APMDH assumption hold for the resulting self-bilinear group scheme.

Let $(\mathsf{RGen}_{\mathsf{ring}}, \mathsf{Add}_{\mathsf{ring}}, \mathsf{Sub}_{\mathsf{ring}}, \mathsf{Mult}_{\mathsf{ring}})$ be a ring scheme and c be a prime such that $\gcd(c, \mathrm{char}(R)) = 1$ with overwhelming probability where R is a ring specified by $PP \xleftarrow{\$} \mathsf{RGen}_{\mathsf{ring}}(1^\lambda)$. The construction is as follows.

$\mathsf{GGen}_{\mathsf{sbm}}(1^\lambda)$: This algorithm computes $PP \xleftarrow{\$} \mathsf{RGen}_{\mathsf{ring}}(1^\lambda)$ and simply outputs PP. Let R and $[1]_R$ be the ring and the multiplicative identity specified by PP. As public parameters of self-bilinear group scheme, PP specifies a group G, a generator $[1]_G \in G$ and a self-bilinear map e as follows. G is defined as the additive subgroup of R generated by $[1]_R$ and let $[1]_G := [c]_R$. The self-bilinear map e is defined by $e(X, Y) = X \cdot Y$ where $\cdot$ denotes the multiplication on R. It is easy to verify that e is actually a self-bilinear map on G.

$\mathsf{Add}_{\mathsf{sbm}}(X, Y)$: This algorithm computes $\mathsf{Add}_{\mathsf{ring}}(X, Y)$ and outputs its output.

$\mathsf{Sub}_{\mathsf{sbm}}(X, Y)$: This algorithm computes $\mathsf{Sub}_{\mathsf{ring}}(X, Y)$ and outputs its output.

$\mathsf{Map}_{\mathsf{sbm}}(X, Y)$: This algorithm computes $\mathsf{Mult}_{\mathsf{ring}}(X, Y)$ and outputs its output.

The scheme is correct if the underlying ring scheme is correct. For a self-bilinear map e defined as above, we have $e([x]_1, [y]_1) = [cxy]_1$ because we have $e([x]_1, [y]_1) = e([cx]_R, [cy]_R) = [cx]_R \cdot [cy]_R = [c^2xy]_R = [cxy]_1$. We remark that we have $[x]_n = [c^n x]_R$ because we have $[x]_n = x \cdot e_n([1]_1, \ldots, [1]_1) = x \cdot [c^{n-1}]_1 = x \cdot [c^n]_R = [c^n x]_R$.

Hardness assumptions. Here, we prove that the complexity assumptions defined in Sec 6.1 hold for the above scheme if the c-inversion assumption holds for the underlying ring scheme. Specifically, for any integers n, ℓ, we prove that the n-MCDH holds under the c-inversion assumption and the ℓ-APMDH assumption holds under the c-inversion assumption. As an extension of these results, we give a sufficient condition of parameters $(\mathcal{F}, f^*, \ell^*)$ for which the $(\mathcal{F}, f^*, \ell^*)$-MGDH assumption holds under the c-inversion assumption.

First, we prove that the MCDH assumptions holds under the c-inversion assumption.

Theorem 5. *(Formal version of Theorem 2.) If the c-inversion assumption holds for the underlying ring scheme, then for any integer n, the n-MCDH assumption holds for the above self-bilinear group scheme.*

Proof. We assume that there exists a PPT adversary $\mathcal{A}$ that breaks the n-MCDH assumption and construct a PPT adversary $\mathcal{B}$ that breaks the c-inversion assumption. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}(PP)$**:** $\mathcal{B}$ picks $x'_1, \ldots, x'_{n+1} \xleftarrow{\$} [2^{\ell+\lambda}]$ and sets $[x_i]_1 := [cx'_i + 1]_R$ for $i \in [n+1]$. Then $\mathcal{B}$ runs $\mathcal{A}$ with an input $(PP, \{[x_i]_1\}_{i\in[n+1]})$ to obtain T, and computes $A := \prod_{i=1}^{n}(cx'_i + 1)$ and $S := T - [x'_{n+1} \cdot A]_R - [c^{-1} \cdot (A-1)]_R$. Here, we remark that since $A \equiv 1 \mod c$, $c^{-1} \cdot (A-1)$ is an integer and thus $\mathcal{B}$ can compute $[c^{-1} \cdot (A-1)]_R$. Then it outputs S.

We show that $\mathcal{B}$ works correctly. First, we remark that in the simulation by $\mathcal{B}$, for $i \in [n+1]$, x_i is defined as $x_i = x'_i + c^{-1} \mod \mathrm{ord}(G)$ since it sets $[x_i]_1 := [cx'_{n+1} + 1]_R$ and we defined $[1]_1 = [c]_R$. Since $(x'_i \mod \mathrm{ord}(G))$ is distributed almost uniformly on $\mathbb{Z}_{\mathrm{ord}(G)}$ and c is coprime to $\mathrm{ord}(G)$, $(x_i \mod \mathrm{ord}(G))$ is also distributed almost uniformly on $\mathbb{Z}_{\mathrm{ord}(G)}$ and especially $[x_i]_1$ is

almost uniformly distributed on G as in a problem instance of the MCDH assumption. Thus if $\mathcal{A}$ breaks the n-MCDH assumption, then $\mathcal{A}$ outputs $[\prod_{i=1}^{n+1} x_i]_n$ with non-negligible probability also in the simulation by $\mathcal{B}$. If $T = [\prod_{i=1}^{n+1} x_i]_n$, then we have $T = [\prod_{i=1}^{n+1} x_i]_n = [\prod_{i=1}^{n+1}(x'_i + c^{-1})]_n = [c^n \prod_{i=1}^{n+1}(x'_i + c^{-1})]_R = [(\prod_{i=1}^{n}(cx'_i + 1))(x'_{n+1} + c^{-1})]_R = [x'_{n+1} \cdot A + c^{-1} \cdot (A-1) + c^{-1}]_R$. Thus we have $S = T - x'_{n+1} \cdot A - c^{-1} \cdot (A-1) = [c^{-1}]_R$ and thus $\mathcal{B}$ succeeds in solving the c-inversion problem. □

The following theorem state that the APMDH assumption holds under the c-inversion assumption.

Theorem 6. *If the c-inversion assumption holds for the underlying ring scheme, then for any integer ℓ such that $\log \ell = \mathsf{poly}(\lambda)$, the ℓ-APMDH assumption holds for the above self-bilinear group scheme.*

We omit the proof because this theorem follows as a corollary of Theorem 7. (See Example 1).

Next, we prove a more general theorem which gives a sufficient condition of parameters $(\mathcal{F}, f^*, \ell^*)$ such that the $(\mathcal{F}, f^*, \ell^*)$-MGDH assumption holds under the c-inversion assumption. Before stating our theorem, we prepare some definitions.

Definition 11. *For a monic monomial f defined by $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i^{e_i}$, we define its corresponding polynomial $\bar{f}$ by $\bar{f}(x_1, \ldots, x_n) := \sum_{i=1}^{n} e_i \cdot x_i$.*

Definition 12. *For a family $\mathcal{F}$ of monic monomials with n variables, a monic monomial f^* with n variables and an integer ℓ^*, we say that $(\mathcal{F}, f^*, \ell^*)$ is MGDH-compatible if we have $n = \mathsf{poly}(\lambda)$, $\log(\deg(f)) = \mathsf{poly}(\lambda)$ for all $f \in \mathcal{F}$, $\log(\deg(f^*)) = \mathsf{poly}(\lambda)$ and $\log \ell^* = \mathsf{poly}(\lambda)$, and there exists a tuple of constant $(a_1, \ldots, a_n) \in \mathbb{Z}^n$ (that does not depend on λ) such that $\bar{f}(a_1 \ldots, a_n) + 1 \geq 0$ for all $f \in \mathcal{F}$, $\bar{f}^*(a_1, \ldots, a_n) + \ell^* + 1 \leq 0$.*

Then our theorem is stated as follows. Actually, a similar theorem for a self-bilinear map with auxiliary information was given in [YHK16].

Theorem 7. *If the c-inversion assumption holds for the underlying ring scheme and $(\mathcal{F}, f^*, \ell^*)$ is MGDH-compatible, then the $(\mathcal{F}, f^*, \ell^*)$-MGDH assumption holds for the above self-bilinear group scheme.*

Proof. We assume that there exists an adversary $\mathcal{A}$ that breaks the $(\mathcal{F}, f^*, \ell^*)$-MGDH assumption. We construct an adversary $\mathcal{B}$ that breaks the c-inversion assumption based on $\mathcal{A}$. Let $(a_1, \ldots, a_n) \in \mathbb{Z}^n$ be a tuple of integers such that $\bar{f}(a_1 \ldots, a_n) + 1 \geq 0$ holds for all $f \in \mathcal{F}$ and $\bar{f}^*(a_1, \ldots, a_n) + \ell^* + 1 \leq 0$. (Such a tuple exists because we assume that $(\mathcal{F}, f^*, \ell^*)$ is MGDH-compatible.) The description of $\mathcal{B}$ is as follows.

$\mathcal{B}(PP)$**:** $\mathcal{B}$ picks $x'_1, \ldots, x'_n \xleftarrow{\$} [2^{\ell+\lambda}]$ and gives PP to $\mathcal{A}$. For each $\mathcal{A}$'s query $f \in \mathcal{F}$ defined by $f(x_1 \ldots x_n) = \prod_{i=1}^{n} x_i^{e_i}$, $\mathcal{B}$ computes $[c^{(\bar{f}(a_1,\ldots,a_n)+1)} \prod_{i=1}^{n}(cx'_i + 1)^{e_i}]_R$ and returns it to $\mathcal{A}$. (Here, $x_1, \ldots, x_n$ are implicitly defined by $x_i := c^{a_i}(cx'_i + 1) \mod \mathrm{ord}(G)$.) Let T be $\mathcal{A}$'s output. Suppose that f^* is defined by $f^*(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i^{e^*_i}$. It computes $S := [\frac{\prod_{i=1}^{n}(cx'_i+1)^{e^*_i} - 1}{c}]_R$. Here, we remark that it is non-trivial for $\mathcal{B}$ to compute S because a ring scheme only provides addition, subtraction and multiplication algorithms and does not provide a division algorithm. However, in fact, S can be computed in polynomial time without using a division. We prove it later. Then $\mathcal{B}$ computes $T' := c^{-(f^*(a_1,\ldots,a_n)+\ell^*+1)} \cdot T - S$ and outputs T'.

The above completes the description of $\mathcal{B}$. We show that $\mathcal{B}$ works correctly. First, we remark that in the simulation by $\mathcal{B}$, for $i \in [n]$, x_i is defined as $x_i = c^{a_i}(cx'_i + 1) \mod \mathrm{ord}(G)$. Since $(x'_i \mod \mathrm{ord}(G))$ is distributed almost uniformly on $\mathbb{Z}_{\mathrm{ord}(G)}$ and c is coprime to $\mathrm{ord}(G)$, $(x_i \mod \mathrm{ord}(G))$ is also almost uniformly distributed on $\mathbb{Z}_{\mathrm{ord}(G)}$ as in the problem instance of the MGDH assumption. Then we show that $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathcal{F}}$ correctly for $\mathcal{A}$. When $\mathcal{A}$ queries $f \in \mathcal{F}$ defined by $f(x_1 \dots x_n) = \prod_{i=1}^{n} x_i^{e_i}$, the oracle is supposed to return $[f(x_1, \dots, x_n)]_1$. Here, we have

$$\begin{aligned}
[f(x_1, \dots, x_n)]_1 &= [c \cdot \prod_{i=1}^{n} (c^{a_i}(cx'_i + 1))^{e_i}]_R \\
&= [c^{(\bar{f}(a_1,\dots,a_n)+1)} \prod_{i=1}^{n} (cx'_i + 1)^{e_i}]_R.
\end{aligned}$$

Therefore the simulation by $\mathcal{B}$ is correct. Hence if $\mathcal{A}$ breaks the MGDH assumption, then it outputs $T = [f^*(x_1, \dots, x_n)]_{\ell^*}$ with non-negligible probability in the simulation by $\mathcal{B}$. Then we have

$$\begin{aligned}
T &= [f^*(x_1, \dots, x_n)]_{\ell^*} \\
&= [c^{\ell^*} \cdot \prod_{i=1}^{n} x_i^{e_i^*}]_R \\
&= [c^{\ell^*} \cdot \prod_{i=1}^{n} (c^{a_i}(cx'_i + 1))^{e_i^*}]_R \\
&= [c^{\bar{f}^*(a_1,\dots,a_n)+\ell^*} \cdot \prod_{i=1}^{n} (cx'_i + 1)^{e_i^*}]_R \\
&= c^{\bar{f}^*(a_1,\dots,a_n)+\ell^*+1} \cdot (S + [c^{-1}]_R)
\end{aligned}$$

Therefore we have $T' := c^{-(f^*(a_1,\dots,a_n)+\ell^*+1)} \cdot T - S = [c^{-1}]_R$. Therefore $\mathcal{B}$ succeeds in breaking the c-inversion assumption.

What is left is to prove that $S = [\frac{\prod_{i=1}^{n}(cx'_i+1)^{e_i^*}-1}{c}]_R$ can be computed in polynomial time by $\mathcal{B}$. We fix $x_1, \dots, x_n$ and define a function $g : \mathbb{Z}^n \to R$ as

$$g(e_1, \dots, e_n) := \left[\frac{\prod_{i=1}^{n}(cx_i + 1)^{e_i} - 1}{c}\right]_R.$$

Then it is easy to see that we have

$$\begin{aligned}
g(e_1, \dots, e_k, 0, \dots, 0) &= (cx_k + 2) \cdot g(e_1, \dots, e_k - 1, 0, \dots, 0) \\
&\quad -(cx_k + 1) \cdot g(e_1, \dots, e_k - 2, 0, \dots, 0)
\end{aligned}$$

for any $k \in [n]$, $e_1, \dots, e_{k-1} \in \mathbb{Z}_{\geq 0}$ and $e_k \in \mathbb{Z}_{\geq 2}$. We also have

$$g(e_1, \dots, e_{k-1}, 1, 0, \dots, 0) = (cx_k + 1) \cdot g(e_1, \dots, e_{k-1}, 0, 0, \dots, 0) + [x_k]_R$$

Therefore if we let

$$A = \begin{pmatrix} [cx_i + 2]_R & [-(cx_i + 1)]_R \\ [1]_R & [0]_R \end{pmatrix}$$

then we have

$$\begin{aligned}
&\begin{pmatrix} g(e_1,\ldots,e_{k-1},e_k,0,\ldots,0) \\ g(e_1,\ldots,e_{k-1},e_k-1,0,\ldots,0) \end{pmatrix} \\
&= A \begin{pmatrix} g(e_1,\ldots,e_{k-1},e_k-1,0,\ldots,0) \\ g(e_1,\ldots,e_{k-1},e_k-2,0,\ldots,0) \end{pmatrix} \\
&= A^{e_k-1} \begin{pmatrix} g(e_1,\ldots,e_{k-1},1,0,\ldots,0) \\ g(e_1,\ldots,e_{k-1},0,0,\ldots,0) \end{pmatrix} \\
&= A^{e_k-1} \begin{pmatrix} (cx_k+1)\cdot g(e_1,\ldots,e_{k-1},0,0,\ldots,0) + [x_k]_R \\ g(e_1,\ldots,e_{k-1},0,0,\ldots,0) \end{pmatrix}
\end{aligned}$$

By using the standard square-and-multiply algorithm, A^{e_k-1} can be computed by $O(\log(e_k))$ matrix multiplications, each of which can be computed by constant number of additions and multiplications on R. Therefore if $g(e_1,\ldots,e_{k-1},0,0,\ldots,0)$ is given, then $g(e_1,\ldots,e_{k-1},e_k,0,\ldots,0)$ can be computed by polynomial number of additions and multiplications if $\log c$, $\log x_1,\ldots$, $\log x_n$ and $\log e_1$, $\ldots$, $\log e_n$ are polynomial in λ. Since we have $g(0,\ldots,0) = [0]_R$, one can compute $S = g(e_1^*,\ldots,e_n^*)$ in polynomial time by computing $g(e_1^*,0,\ldots,0)$, $g(e_1^*,e_2^*,0,\ldots,0)$, $\ldots$, $g(e_1^*,\ldots,e_n^*)$ in order by using the above algorithm. □

We give an example of implication of the above theorem. Namely, we show that Theorem 6 can be reduced to Theorem 7.

Example 1. *If we set* $n = 3$, $\mathcal{F} = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ *and* $f^*(x_1,x_2,x_3) := x_1^{\ell-1}(x_2x_3)^\ell$ *where* $f_1(x_1,x_2,x_3) := x_1$, $f_2(x_1,x_2,x_3) := x_2$, $f_3(x_1,x_2,x_3) := x_3$, $f_4(x_1,x_2,x_3) := x_1x_2$, $f_5(x_1,x_2,x_3) := x_1x_3$, *and* $f_6(x_1,x_2,x_3) := x_1x_2x_3$, *then the* $(\mathcal{F},f^*,\ell)$*-MGDH assumption is equivalent to the* ℓ*-APMDH assumption. If we set* $a_1 = 1$, $a_2 = -1$ *and* $a_3 = -1$, *then we have* $\bar{f}_1(a_1,a_2,a_3)+1 = 2$, $\bar{f}_2(a_1,a_2,a_3)+1 = 0$, $\bar{f}_3(a_1,a_2,a_3)+1 = 0$, $\bar{f}_4(a_1,a_2,a_3)+1 = 1$, $\bar{f}_1(a_1,a_2,a_3)+1 = 1$, $\bar{f}_1(a_1,a_2,a_3)+1 = 0$, *and* $\bar{f}^*(a_1,a_2,a_3)+\ell+1 = (\ell-1)-\ell-\ell+\ell+1 = 0$. *Therefore* $(\mathcal{F},f^*,\ell)$ *is MGDH-compatible, and thus the the* ℓ*-APMDH assumption holds under the generalized c-inversion assumption.*

6.3 Generic Self-bilinear Map Model

Here, we introduce the *generic self-bilinear map model* (GSBM) where an algorithm is only allowed to compute group operations and evaluations of a self-bilinear map independently of actual representations of group elements. We remark that as shown in [CL09], if an order of a group with efficiently computable self-bilinear map is known, then even the CDH assumption does not hold on the group. Therefore in our model, a group is chosen from a distribution $\mathcal{D}_{SB}$ and a generic algorithm cannot know which group was chosen. Actually, our GSBM is very similar to the GRM defined in Sec 5.2. We just replace additions and multiplications in the generic ring model by additions on a group and evaluations of a self-bilinear map respectively.

Straight-line Program on Self-bilinear Group. Before describing our model, we define a straight-line program (SLP) on a self-bilinear map group. Intuitively, that is defined similarly to an SLP on a ring except that multiplications are replaced by evaluations of a self-bilinear map. The formal definition is as follows.

Definition 13. *(straight-line program for self-bilinear group) A straight-line program (SLP)* P *of length* L *with* n *inputs is written as a sequence*

$$\mathrm{P} = (i_1,j_1,\circ_1),\ldots,(i_L,j_L,\circ_L)$$

where $i_k, j_k \in \{\mathsf{in}_1, \ldots, \mathsf{in}_n, 0, 1, \ldots, k-1\}$ *and* $\circ_k \in \{+, -, \cdot\}$ *for all* $k \in [L]$. *For* $(G, [1]_G, e)$ *where* G *is an additive cyclic group,* $[1]_G$ *is a generator of* G *and* e *is a self-bilinear map on* G, P *computes as follows.*

1. *Let* $x_0 := [1]_R$.
2. *For* $1 \leq k \leq n$ *let* $x_{\mathsf{in}_k} := \mathsf{input}_k$.
3. *For* $1 \leq k \leq L$, *let*
$$x_k := \begin{cases} x_{i_k} \circ_k x_{j_k} & \text{if } \circ_k \in \{+, -\} \\ e(x_{i_k}, x_{j_k}) & \text{if } \circ_k = \cdot \end{cases}$$
4. *Output* x_L.

We denote the output of P *with input* $(\mathsf{input}_1, \ldots, \mathsf{input}_n)$ *on a self-bilinear group* $(G, [1]_G, e)$ *by* $\mathrm{P}(\mathsf{input}_1, \ldots, \mathsf{input}_n; (G, [1]_G, e))$. *For an NI-SLP* P, $|\mathrm{P}|$ *denotes the length of* P, *and* P_k *denotes the NI-SLP given by the sequence of the first* k *elements of* P.

Generic self-bilinear map model. The definition of our GSBM is as follows.

Let $\mathcal{D}_{SB} = \{\mathcal{D}_{SB}(\lambda)\}_{\lambda \in \mathbb{N}}$ be a sequence of a distribution $\mathcal{D}_{SB}(\lambda)$ of $(G, [1]_G, e)$ where G is an additive cyclic group such that $|G| \leq 2^\ell$, $[1]_G$ is a generator of G and e is a self-bilinear map on G. In the $\mathcal{D}_{SB}$-GSBM, a generic algorithm $\mathcal{A}$ is given 1^λ as input and allowed to access an oracle $\mathcal{O}_{\mathsf{sbm}}$ described below.

$\mathcal{O}_{\mathsf{sbm}}$ picks a ring R according to the distribution $\mathcal{D}_{SB}$, and chooses secret values $X_1, \ldots, X_n$ according to a certain distribution, which depends on a problem to consider. It maintains an SLP P, which is set to be empty sequence at the beginning. $\mathcal{A}$ is allowed to make the following two kinds of query, the $\mathsf{operation}$ and equal queries.

$\mathsf{operation}$ **query:** When $\mathcal{A}$ makes an $\mathsf{operation}$ query $(i, j, \circ) \in \{\mathsf{in}_1, \ldots, \mathsf{in}_n, 0, \ldots, |P|\} \times \{\mathsf{in}_1, \ldots, \mathsf{in}_n, 0, \ldots, |P|\} \times \{+, -, \cdot\}$, then $\mathcal{O}_{\mathsf{sbm}}$ append $(i, j, \circ)$ to P.

equal **query:** When $\mathcal{A}$ makes an equal query $(i, j) \in \{\mathsf{in}_1, \ldots, \mathsf{in}_n, 0, \ldots, |P|\} \times \{\mathsf{in}_1, \ldots, \mathsf{in}_n, 0, \ldots, |P|\}$, then $\mathcal{O}_{\mathsf{sbm}}$ returns true if $P_i(X_1, \ldots, X_n; (G, [1]_G, e)) = P_j(X_1, \ldots, X_n; (G, [1]_G, e))$ and false otherwise.

Complexity assumptions. Then we define complexity assumptions in this model.

Definition 14. *We say that* n*-MCDH assumption holds in the* $\mathcal{D}_{SB}$*-GSBM if for any PPT generic algorithm* $\mathcal{A}$ *in the model,*

$$\Pr\left[i^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sbm}}}(1^\lambda); \mathrm{P}_{i^*}(X_1, \ldots, X_{n+1}; (G, [1]_G, e)) = [\prod_{i=1}^{n+1} x_i]_n\right]$$

is negligible where the secret values are chosen as $X_i := [x_i]_1$ *for* $x_i \xleftarrow{\$} [2^{\ell+\lambda}]$ *for* $i \in [n+1]$.

Definition 15. *We say that* ℓ*-APMDH assumption holds in the* $\mathcal{D}_{SB}$*-GSBM if for any PPT generic algorithm* $\mathcal{A}$ *in the model,*

$$\Pr[i^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sbm}}}(1^\lambda); \mathrm{P}_{i^*}(X_1, \ldots, X_6; (G, [1]_G, e)) = [x_1^{\ell-1}(x_2 x_3)^\ell]_\ell]$$

is negligible where the secret values are chosen as $X_1 := [x_1]_1$, $X_2 := [x_2]_1$, $X_3 := [x_3]_1$, $X_4 := [x_1 x_2]_1$, $X_5 := [x_1 x_3]_1$ *and* $X_6 := [x_1 x_2 x_3]_1$ *for* $x_1, \ldots, x_3 \xleftarrow{\$} [2^{\ell+\lambda}]$.

Definition 16. *For a set $\mathcal{F} := \{f_1, \ldots, f_m\}$ of monic monomials of n-input, a monic monomial f^* of n-input and an integer ℓ^*, we say that the $(\mathcal{F}, f^*, \ell^*)$-MGDH assumption holds in the $\mathcal{D}_{SB}$-GSBM if for any PPT generic algorithm $\mathcal{A}$ in the model,*

$$\Pr[i^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sbm}}}(1^\lambda); \mathrm{P}_{i^*}(X_1, \ldots, X_n; (G, [1]_G, e)) = [f^*(x_1, \ldots, x_n)]_\ell$$

is negligible where the secret values are chosen as $X_j := [f_j(x_1, \ldots, x_n)]_1$ for $j \in [m]$ where $x_i \xleftarrow{\$} [2^{\ell+\lambda}]$ for $i \in [n]$.

Generic hardness of complexity assumptions. In the following, we let $\mathcal{D}_{SB}$ be a distribution of $(G, [1]_G, e)$ where G is a additive group $\mathbb{Z}_p$, $[1]_G := c$, and a self-bilinear map e is defined by $e([x]_G, [y]_G) = [cxy]_G$ for a prime number $c \geq 2^\ell$ where p is uniformly chosen from $\mathcal{P}_\ell$.

We can reduce the MCDH, APMDH and MGDH assumptions in $\mathcal{D}_{SB}$-generic self-bilinear group model to the c-inversion assumption in the $\mathcal{D}_R$-generic ring model based on similar arguments to proofs of Theorem 5, 6 and 7 where $\mathcal{D}_R$ is a distribution of $R = \mathbb{Z}_p$ for $p \xleftarrow{\$} \mathcal{P}_\ell$. Since the c-inversion assumption holds in the $\mathcal{D}_R$-generic ring model under the factoring assumption w.r.t. unbalanced moduli as shown in Theorem 4, the above assumptions also hold under the same assumption. Therefore we obtain the following theorems.

Theorem 8. *If the factoring assumption w.r.t. unbalanced moduli holds, then for any natural number n, then the n-MCDH assumption holds in the $\mathcal{D}_{SB}$-GSBM.*

Theorem 9. *If the factoring assumption w.r.t. unbalanced moduli holds, then for any natural number ℓ, the ℓ-APMDH assumption holds in the $\mathcal{D}_{SB}$-GSBM.*

Theorem 10. *If the factoring assumption w.r.t. unbalanced moduli holds and $(\mathcal{F}, f^*, \ell^*)$ is MGDH-compatible, then $(\mathcal{F}, f^*, \ell^*)$-MGDH assumption holds in the $\mathcal{D}_{SB}$-GSBM.*

7 Applications of Self-bilinear Map

Here we discuss applications of a self-bilinear map.

7.1 Multiparty NIKE

Here, we give a definition of a multiparty NIKE. Then we give a construction of a multiparty NIKE scheme based on a self-bilinear map that satisfies the MCDH assumption. The construction can be seen as an instantiation of the scheme proposed by Boneh and Silverberg [BS02] with an unbounded level multilinear map obtained by a self-bilinear map.

Definition. First, we formally define multiparty non-interactive key exchange (NIKE) and its security following [BZ14]. A multiparty NIKE scheme consists of three algorithms (Setup,Publish,KeyGen).

$\mathsf{Setup}(1^\lambda)$**:** This algorithm takes a security parameter 1^λ as input[6]. It outputs public parameters params.

$\mathsf{Publish}(\mathsf{params})$**:** This algorithm takes public parameters params as input. It outputs a public key pk and a secret key sk.

$\mathsf{KeyGen}(\mathsf{params}, sk, pk_1, \ldots, pk_{n-1})$**:** This algorithm takes public parameter params, a secret key sk and public keys $pk_1, \ldots, pk_{n-1}$. It outputs a shared key k.

[6]In our definition, the setup algorithm does not take the number of maximum users as input. This means that our scheme admits unbounded number of users.

As correctness, we require that for any n, $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk_i, sk_i) \leftarrow \mathsf{Publish}(\mathsf{params})$ for $i \in [n]$, for any $i_1, i_2 \in [n]$, we have

$$\begin{aligned} & \mathsf{KeyGen}(\mathsf{params}, sk_{i_1}, pk_1, \ldots, pk_{i_1-1}, pk_{i_1+1}, \ldots pk_n) \\ = \; & \mathsf{KeyGen}(\mathsf{params}, sk_{i_2}, pk_1, \ldots, pk_{i_2-1}, pk_{i_2+1}, \ldots pk_n). \end{aligned}$$

We define the security notion for NIKE. In this paper, we consider the minimum security notion against a passive adversary.

We say that a multiparty NIKE scheme is statically secure if for any integer n which is polynomial in the security parameter, for any efficient adversary $\mathcal{A}$, $|\Pr[b \xleftarrow{\$} \mathcal{A}(\mathsf{params}, pk_1, \ldots, pk_n, K_b)] - 1/2|$ is negligible, where $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk_i, sk_i) \leftarrow \mathsf{Publish}(\mathsf{params})$ for $i = 1, \ldots, n$, $K_1 := \mathsf{KeyGen}(\mathsf{params}, sk_1, pk_2, \ldots, pk_n)$, $K_0 \xleftarrow{\$} \{0,1\}^{\ell_K}$ and $b \xleftarrow{\$} \{0,1\}$.

Remark 5. *In some existing works [BZ14, KRS15], they consider stronger security notion that considers adversaries actively generate malformed public keys. In this paper, we do not consider such an adversary, and we assume that an adversary only observe honestly generated public keys.*

Construction. We describe the construction of multiparty NIKE scheme based on a self-bilinear map. Let $(\mathsf{GGen}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Map})$ be a self-bilinear group scheme. Let ℓ be the maximum size of a representation of an element of a group generated by GGen. For $r \in \{0,1\}^\ell$ $\mathsf{GL}_r : \{0,1\}^\ell \to \{0,1\}$ denotes the Goldreich-Levin hardcore bit function [GL89], i.e., for $x \in \{0,1\}^\ell$, $\mathsf{GL}_r(x) = \bigoplus_{i=1}^{\ell}(x_i \cdot r_i)$ where x_i and r_i denotes i-th bit of x and r respectively. The description of our scheme is as follows.

$\mathsf{Setup}(1^\lambda)$**:** This algorithm computes $PP \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, picks $r \xleftarrow{\$} \{0,1\}^\ell$ and outputs $\mathsf{params} = (PP, r)$. We let $(G, [1]_G, e)$ be a group, generator and self-bilinear map specified by PP.

$\mathsf{Publish}(\mathsf{params})$**:** This algorithm randomly picks $x \xleftarrow{\$} [2^{\ell+\lambda}]$, computes $[x]_G$, sets $sk := x$ and $pk := [x]_G$ and outputs (pk, sk).

$\mathsf{KeyGen}(\mathsf{params} = (PP, r), sk, pk_1, \ldots, pk_{n-1})$**:** This algorithm computes $K := \mathsf{GL}_r(sk \cdot e_{n-1}(pk_1, \ldots, pk_{n-1}))$ and outputs K.

For any n, $\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk_j = [x_j]_G, sk_j = x_j) \leftarrow \mathsf{Publish}(\mathsf{params})$ for $j \in [n]$, for any $i \in [n]$, we have $\mathsf{KeyGen}(\mathsf{params}, sk_i, pk_1, \ldots, pk_{i-1}, pk_{i+1}, \ldots pk_n) = \mathsf{GL}_r([\prod_{i=1}^n x_i]_{n-1})$. Therefore the correctness follows.

Security. The security of the above scheme can be stated as follows.

Theorem 11. *If the n-MCDH assumption holds for all $n \in \mathbb{N}$ for $(\mathsf{GGen}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Map})$, then the above scheme is statically secure.*

Proof. (sketch) If the $(n-1)$-MCDH assumption holds, any PPT algorithm given $[x_1]_G, \ldots, [x_n]_G$ cannot compute $[\prod_{i=1}^n x_i]_{n-1}$ with non-negligible probability where $x_1, \ldots, x_n \xleftarrow{\$} [2^{\ell+\lambda}]$. Therefore by using the Goldreich-Levin theorem [GL89], any PPT algorithm given $[x_1]_G, \ldots, [x_n]_G$ cannot distinguish $\mathsf{GL}_r([\prod_{i=1}^n x_i]_{n-1})$ from random bit for $r \xleftarrow{\$} \{0,1\}^\ell$. □

7.2 Broadcast Encryption

Here, we define broadcast encryption. Actually, we define distributed broadcast encryption [BZ14] in which any users can join a system without any secret information. Then we show that a distributed encryption scheme can be obtained by any multiparty NIKE scheme. The conversion is similar to the one presented in [BZ14]. As a result, we obtain a distributed broadcast encryption based on a self-bilinear map secure under the MCDH assumption.

Definition. A distributed broadcast encryption scheme consists of four algorithms $(\mathsf{Setup}, \mathsf{Join}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{Setup}(1^\lambda)$**:** It takes the security parameter 1^λ as input and outputs public parameters PP.

$\mathsf{Join}(PP)$**:** It takes public parameters PP as input and outputs a public key pk and a secret key sk.

$\mathsf{Enc}(PP, pk_1, \ldots, pk_n, msg)$**:** It takes public parameters PP, a message msg, and a public keys $pk_1, \ldots, pk_n$ of designated receivers and outputs a ciphertext CT.

$\mathsf{Dec}(PP, sk, pk_1, \ldots, pk_n, CT)$**:** It takes public parameters PP, a secret key sk, public keys of designated receivers and a ciphertext CT and outputs a message msg.

As correctness, we require that for any security parameter λ, an integer n and a message msg, we have $\mathsf{Dec}(PP, sk, pk_1, \ldots, pk_n, CT) = msg$, where $PP \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk_1, sk_1), \ldots, (pk_n, sk_n) \leftarrow \mathsf{Join}(PP)$, $CT \leftarrow \mathsf{Enc}(PP, pk_1, \ldots, pk_n, msg)$.

We define the security notion. We consider the following experiment. A challenger runs $\mathsf{Setup}(1^\lambda)$ to generate public parameters PP and runs $\mathsf{Join}(PP)$ n times to generate $(pk_1, sk_1), \ldots, (pk_n, sk_n)$. It gives $(PP, pk_1, \ldots, pk_n)$ to $\mathcal{A}$. $\mathcal{A}$ chooses two messages msg_0 and msg_1 to submit them to $\mathcal{C}$. $\mathcal{C}$ uniformly choose $b \xleftarrow{\$} \{0,1\}$ and runs $\mathsf{Enc}(PP, pk_1, \ldots, pk_n, msg_b)$ to generate CT, and gives CT to $\mathcal{A}$. Finally $\mathcal{A}$ outputs b'. We say that $\mathcal{A}$ wins if $b = b'$ holds.

We say that a distributed broadcast encryption scheme is statically secure if for any n (polynomially bounded in λ) and any PPT adversary $\mathcal{A}$, $|\Pr[\mathcal{A} \text{ wins}] - 1/2|$ is negligible.

Remark 6. *At first glance, the above security notion seems weaker than the usual static security of broadcast encryption because we do not allow an adversary to corrupt receivers who are out of the target set. However, in distributed setting, secret and public keys of such receivers can be simulated efficiently by using the public parameters. Therefore we still capture the setting where the adversary may corrupt some receivers (as long as the set of corrupted users is determined at the beginning of the experiment).*

Remark 7. *There is stronger security notion for broadcast encryption called adaptive security, where an adversary can determine which receiver to corrupt adaptively. In this paper we only consider the static security and does not consider the adaptive security.*

Construction. Here, we give a construction of distributed broadcast encryption scheme based on a multiparty NIKE scheme. This is based on the conversion proposed in [BZ14]. Let $(\mathsf{Setup}_{\mathsf{NIKE}}, \mathsf{Publish}_{\mathsf{NIKE}}, \mathsf{KeyGen}_{\mathsf{NIKE}})$ be a multiparty NIKE scheme. Then we construct a distributed broadcast encryption scheme $(\mathsf{Setup}_{\mathsf{BE}}, \mathsf{Join}_{\mathsf{BE}}, \mathsf{Enc}_{\mathsf{BE}}, \mathsf{Dec}_{\mathsf{BE}})$ as follows.

$\mathsf{Setup}_{\mathsf{BE}}(1^\lambda)$**:** It runs $\mathsf{Setup}_{\mathsf{NIKE}}(1^\lambda)$ to obtain public parameters PP and outputs PP as its own public parameters.

$\mathsf{Join}_{\mathsf{BE}}(PP)$**:** It runs $\mathsf{Publish}_{\mathsf{NIKE}}(PP)$ to obtain a public key pk and a secret key sk, and outputs (pk, sk).

$\mathsf{Enc}(PP, pk_1, \ldots, pk_n, msg)$**:** It runs $\mathsf{Publish}_{\mathsf{NIKE}}(PP)$ to obtain (pk^*, sk^*). Then it runs $\mathsf{KeyGen}_{\mathsf{NIKE}}(\mathsf{PP}, sk^*, \{pk$ to obtain K. It computes $\Psi := K \oplus M$ and outputs a ciphertext $CT = (pk^*, \Psi)$.

$\mathsf{Dec}(PP, sk, pk_1, \ldots, pk_n, CT)$**:** It parses CT as pk^*, Ψ. It finds i such that sk is a corresponding secret key of pk_i. Then it runs $K \leftarrow \mathsf{KeyGen}_{\mathsf{NIKE}}(\mathsf{PP}, sk, pk^*, pk_1, \ldots, pk_{i-1}, pk_{i+1}, \ldots, pk_n)$ and outputs $M := K \oplus \Psi$.

The security of the above scheme is immediate from the security of the underlying multiparty NIKE scheme.

Theorem 12. *If the multiparty NIKE scheme* ($\mathsf{Setup}_{\mathsf{NIKE}}$,$\mathsf{Publish}_{\mathsf{NIKE}}$ *is statically secure, then the distributed broadcast encryption scheme* ($\mathsf{Setup}_{\mathsf{BE}}$,$\mathsf{Join}_{\mathsf{BE}}$,$\mathsf{Enc}_{\mathsf{BE}}$, $\mathsf{Dec}_{\mathsf{BE}}$) *is statically secure.*

7.3 Fully Homomorphic Signatures

Here, we construct a selectively secure fully homomorphic signatures based on a self-bilinear map that satisfies the ℓ-APMDH assumption for all ℓ such that $\log \ell = \mathsf{poly}(\lambda)$. Our scheme can be seen as an instantiation of the scheme proposed by Catalano et al. [CFW14] with an unbounded level multilinear map obtained by a self-bilinear map. To the best of our knowledge, this is the first fully homomorphic signatures for unbounded depth circuits without relying on SNARKs [BCCT12].

Definition. Here, we only consider the single-data case because there is a generic conversion from single-data homomorphic signatures to multi-data homomorphic signatures [GVW15]. Moreover, we let the message space to be $\{0, 1\}$. We note that this does not lose generality because if one wants to sign a longer message, one can simply sign bitwise. A single-data fully homomorphic signature scheme consists of PPT algorithms ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval}$).

$\mathsf{KeyGen}(1^\lambda, 1^n) \to (vk, sk)$**:** This algorithm takes the security parameter 1^λ and a data size 1^n, and outputs a pair $(vk.sk)$ of a verification key and a signing key.

$\mathsf{Sign}(sk, i, m) \to \sigma$**:** This algorithm takes a signing key sk, an index $i \in [n]$ and a message $m \in \{0, 1\}$, and outputs a signature σ.

$\mathsf{Eval}(f, (m_1, \sigma_1), \ldots, (m_n, \sigma_n)) \to \sigma^*$**:** This algorithm takes a function f (described by a polynomial size circuit) and pairs $(m_1, \sigma_1), \ldots, (m_n, \sigma_n)$ of a message and a signature, and outputs a signature σ^* for the message $f(m_1, \ldots, m_n)$.

$\mathsf{Verify}(vk, f, m, \sigma) \to 1/0$**:** This algorithm takes a verification key vk, a function f, a message m and a signature σ, and outputs 1 if accepts and 0 else

Remark 8. *In the above, we only consider the verification of evaluated signature output by* Eval*, but one can also verify "fresh" signatures generated by* Sign *by setting f to be the identity function.*

We require a fully homomorphic signature scheme to satisfy the following properties.

Correctness For any $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, 1^n)$, $(m_1, \ldots, m_n) \in \{0, 1\}^N$ and a function f (described by a polynomial size circuit), if we let $\sigma_i \leftarrow \mathsf{Sign}(sk, i, m_i)$, $m^* := f(m_1, \ldots, m_n)$,

and $\sigma^* := \mathsf{Eval}(f, (m_1, \sigma_1), \ldots, (m_n, \sigma_n))$, then we have $\mathsf{Verify}(vk, f, m^*, \sigma^*) = 1$.

Security. In this paper, we only consider the selective security. To define the selective security, we consider the following game between an adversary $\mathcal{A}$ and a challenger.

$\mathcal{A}$ sends $(m_1, \ldots, m_n) \in \{0,1\}^n$ to the challenger. The challenger generates $(vk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n)$, computes $\sigma_i \leftarrow \mathsf{Sign}(sk, i, m_i)$ for $i \in [n]$ and gives $vk, \sigma_1, \ldots, \sigma_n$ to $\mathcal{A}$. $\mathcal{A}$ outputs (f^*, m^*, σ^*). We say that $\mathcal{A}$ wins if $\mathsf{Verify}(vk, f^*, m^*, \sigma^*)$ and $m^* \neq f^*(m_1, \ldots, m_n)$ hold. We say that the a homomorphic signature scheme is selectively secure if for any $n = \mathsf{poly}(\lambda)$ and any PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible.

Construction. Here, we construct a selectively secure single data fully homomorphic signature scheme. Our scheme is based on the idea of [CFW14]. Namely, our scheme is almost automatically obtained by replacing a multilinear map by a self-bilinear map in the scheme of [CFW14].

Before giving our construction, we prepare some notations and remarks on (arithmetic) circuits. We can convert any boolean circuit $f : \{0,1\}^n \to \{0,1\}$ to an equivalent arithmetic circuit $f' : \mathbb{Z}^n \to \mathbb{Z}$ such that $f(x_1, \ldots, x_n) = f'(x_1, \ldots, x_n)$ for any $(x_1, \ldots, x_n) \in \{0,1\}^n$. For an arithmetic circuit f and a wire w of f, f_w denotes an arithmetic circuit that outputs the value assigned to w. More precisely, if w is an input gate corresponding to i-th input, we define $f_w(x_1, \ldots, x_n) := x_i$ and if w is an output gate of the gate that computes $\circ \in \{+, -, \cdot\}$ for input wires w_1 and w_2, we define $f_w(x_1, \ldots, x_n) := f_{w_1}(x_1 \ldots, x_n) \circ f_{w_2}(x_1, \ldots x_n)$. We define w's degree as the degree of f_w. As discussed in [CFW14], without loss of generality, we can assume that incoming wires of an addition gate is the same degree. Therefore in our construction, we assume that f is given as an arithmetic circuit with this property.

The construction of our scheme is as follows. We let message space to be $\{0,1\}$.

$\mathsf{KeyGen}(1^\lambda, 1^n) \to (vk, sk)$**:** This algorithm generates $PP \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, chooses $r_i \xleftarrow{\$} [2^{\ell+\lambda}]$ $(i = 1, \ldots, n)$ and $x, y, z \xleftarrow{\$} [2^{\ell+\lambda}]$, and sets $R_i := [r_i]_1$ $(i = 1, \ldots, n)$, $X := [x]_1$, $Y := [y]_1$, $Z = [z]_1$, $S := [xy]_1$ $T := [yz]_1$ and $U := [xyz]_1$. Then it sets $vk := (PP, \{R_i\}_{i \in [n+1]}, X, Y, Z, S, T, U)$, $sk := (x, y, vk)$
and outputs $(vk.sk)$.

$\mathsf{Sign}(sk, i, m) \to \sigma$: This algorithm computes $\Lambda := y \cdot (R_i - m \cdot Z)$, $\Gamma := x \cdot \Lambda$, and outputs $\sigma := (\Lambda, \Gamma)$.

$\mathsf{Eval}(pk, f, (m_1, \sigma_1), \ldots, (m_n, \sigma_n)) \to \sigma^*$**:** Parse as $(\Lambda_i, \Gamma_i) \leftarrow \sigma_i$ for all $i \in [n]$. For all wires w of f, this algorithm computes Λ_w and Γ_w as follows. If w is an input wire of the circuit corresponding to i-th input, then it sets $\Lambda_w := \Lambda_i$ and $\Gamma_w := \Gamma_i$. Then it computes as follows gate by gate.

Addition Gate: Let w be an output wire of an addition gate whose input gates are w_1 and w_2. It computes

$$\Lambda_w := \Lambda_{w_1} + \Lambda_{w_2}, \ \Gamma_w := \Gamma_{w_1} + \Gamma_{w_2}.$$

Multiplication Gate: Let w be an output wire of an addition gate whose input gates are w_1 and w_2. Let d_1, d_2, d be degrees of w_1, w_2, w respectively. (Then we have $d = d_1 + d_2$ by the definition of degree.) It computes

$$\Lambda_w := e(\Lambda_{w_1}, \Gamma_{w_2}) + e(\Lambda_{w_1}, f_{w_2}(m) \cdot U_{d_2}) + e(f_{w_1}(m) \cdot U_{d_1}, \Lambda_{w_2}),$$

$$\Gamma_w := e(\Gamma_{w_1}, \Gamma_{w_2}) + e(\Gamma_{w_1}, f_{w_2}(m) \cdot U_{d_2}) + e(f_{w_1}(m) \cdot U_{d_1}, \Gamma_{w_2})$$

where U_{d_i} denotes $e_{d_i}(U, \ldots, U)$ and $f_{w_i}(m)$ denotes $f_{w_i}(m_1, \ldots, m_n)$ for $i = 1, 2$.

Let w_{out} be the output wire of f. Then it outputs $\sigma^* = (\Lambda_{w_{\mathsf{out}}}, \Gamma_{w_{\mathsf{out}}})$.

$\mathsf{Verify}(pk, f, m, \sigma) \to 1/0$**:** Parse $(\Lambda, \Gamma) \leftarrow \sigma$. Let d be the degree of f. It computes $R = [f(r_1, \ldots, r_n)]_d$, $Z_d = e_d(Z, \ldots, Z)$ and $W_d = [x^{d-1}y^d]_d = e(e_d(S, \ldots, S), X)$. We remark that though this algorithm does not take $r_1, \ldots, r_n$ as input, R can be computed efficiently by "evaluating f on $R_1, \ldots, R_n$" where an addition gate is interpreted as an addition on G and a multiplication gate is interpreted as an evaluation of e. If

$$e(R - m \cdot Z_d, W_d) = e(\Lambda, [1]_d)$$

and

$$e(\Gamma, [1]_1) = e(\Lambda, X)$$

hold, then it outputs 1 and otherwise outputs 0.

This complete the description of the scheme.

We check the correctness of the scheme. In the following for any wire w, we denote $f_w(r_1, \ldots, r_n)$ and $f_w(m_1, \ldots, m_n)$ by $f_w(r)$ and $f_w(m)$ respectively for simplicity. For proving the correctness, we show the following claim.

Claim 1. *Let $\sigma_i = (\Lambda_i, \Gamma_i)$ be a signature generated by $\mathsf{Sign}(sk, i, m_i)$. Then in the computation of $\mathsf{Eval}(pk, f, \{(m_i\sigma_i)\}_{i\in[n]})$, for any wire w of f with degree d, we have*

$$\Lambda_w = [x^{d-1}y^d(f_w(r) - f_w(m)z^d)]_d$$

and

$$\Gamma_w = [x^d y^d(f_w(r) - f_w(m)z^d)]_d.$$

Proof. (of Claim 1.) We prove this claim by induction. First, for an input wire w, the claim holds since if w is an output wire of the i-th input gate, then we have $[x^0y^1(f_w(r) - f_w(m)z^1)]_1 = y[r_i - m_i z]_1 = y \cdot (R - m_i \cdot Z)$ and $[x^1y^1(f_w(r) - f_w(m)z^1)]_1 = xy \cdot (R - m_i \cdot Z)$. We assume that the claim holds for all wires whose depth is smaller than k and prove that the claim holds for any wire w with depth $k+1$. Let w_1 and w_2 be incoming wires of the gate from which w comes, and d_1 and d_2 be degrees of w_1 and w_2 respectively. Then we have $\Lambda_{w_i} = [x^{d_i-1}y^{d_i}(f_{w_i}(r) - f_{w_i}(m)z^{d_i})]_{d_i}$ and $\Gamma_{w_i} = [x^{d_i}y^{d_i}(f_{w_i}(r) - f_{w_i}(m)z^{d_i})]_{d_i}$. We consider the following cases depending on the type of the gate.

Addition: This is the case that w is an output of an addition gate. In this case, we have

$$\Lambda_w = \Lambda_{w_1} + \Lambda_{w_2} = [x^{d-1}y^d(f_w(r) - f_w(m)z^d)]_d$$

and

$$\Gamma_w = \Lambda_{w_1} + \Lambda_{w_2} = [x^d y^d(f_w(r) - f_w(m)z^d)]_d$$

where we used $f_{w_1}(r) + f_{w_2}(r) = f_w(r)$, $f_{w_1}(m) + f_{w_2}(m) = f_w(m)$ and $d_1 = d_2 = d$.

Multiplication: This is the case that w is an output of a multiplication gate. Let w_1 and w_2 be incoming wires to the gate, and d_1 and d_2 be degrees of w_1 and w_2 respectively. Then we have

$$\begin{aligned}
\Lambda_w \quad &= e(\Lambda_{w_1}, \Gamma_{w_2}) + e(\Lambda_{w_1}, f_{w_2}(m) \cdot U_{d_2}) + e(f_{w_1}(m) \cdot U_{d_1}, \Lambda_{w_2}) \\
&= [x^{d_1+d_2-1}y^{d_1+d_2}(f_{w_1}(r)f_{w_2}(r) - z^{d_2}f_{w_1}(r)f_{w_2}(m) \\
&\quad - z^{d_1}f_{w_2}(r)f_{w_1}(m) + f_{w_1}(m)f_{w_2}(m)z^{d_1+d_2})]_{d_1+d_2} \\
&+ [x^{d_1+d_2-1}y^{d_1+d_2}z^{d_2}(f_{w_1}(r)f_{w_2}(m) - f_{w_1}(m)f_{w_2}(m)z^{d_1})]_{d_1+d_2} \\
&+ [x^{d_1+d_2-1}y^{d_1+d_2}z^{d_1}(f_{w_2}(r)f_{w_1}(m) - f_{w_1}(m)f_{w_2}(m)z^{d_2})]_{d_1+d_2} \\
&= [x^{d-1}y^d(f_w(r) - f_w(m)z^d)]_d
\end{aligned}$$

and

$$\begin{aligned}\Gamma_w &= e(\Gamma_{w_1}, \Gamma_{w_2}) + e(\Gamma_{w_1}, f_{w_2}(m) \cdot U_{d_2}) + e(f_{w_1}(m) \cdot U_{d_1}, \Gamma_{w_2}) \\ &= x \cdot \Lambda_w \\ &= [x^d y^d (f_w(r) - f_w(m) z^d)]_d\end{aligned}$$

where we used $f_{w_1}(r) f_{w_2}(r) = f_w(r)$, $f_{w_1}(m) f_{w_2}(m) = f_w(m)$ and $d = d_1 + d_2$.

Therefore, in any case, the claim also holds for w. Therefore the claim holds. □

□

By the claim, if we let $(\Lambda, \Gamma) \xleftarrow{\$} \mathsf{Eval}(pk, f, \{(m_i, \sigma_i)\}_{i \in [n]})$, then we have

$$\Lambda = [x^{d-1} y^d (f(r) - f(m) z^d)]_d$$

and

$$\Gamma = [x^d y^d (f(r) - f(m) z^d)]_d.$$

where $f(r)$ and $f(m)$ denotes $f(r_1, \ldots, r_n)$ and $f(m_1, \ldots, m_n)$ respectively and d is the degree of f. If $m^* = f(m)$, then we have

$$e(R - m^* \cdot Z_d, W_d) = e([f(r) - f(m) \cdot z^d]_d, [x^{d-1} y^d]_d) = e(\Lambda, [1]_d)$$

and

$$e(\Gamma, [1]_1) = e(x \cdot \Lambda, [1]_1) = e(\Lambda, X).$$

Therefore the correctness holds.

The security of the scheme can be stated as follows.

Theorem 13. *If the ℓ-APSBDH assumption holds for all ℓ such that $\log(\ell) = \mathsf{poly}(\lambda)$ for the underlying self-bilinear map, then the above scheme is selectively secure.*

Proof. We assume that there exists an adversary $\mathcal{A}$ that breaks the selective security of the scheme. Let d_{max} be the maximum degree of f output by $\mathcal{A}$ as a part of a forgery. Then we construct an algorithm $\mathcal{B}$ that breaks the d_{max}-APMDH assumption by using $\mathcal{A}$.

$\mathcal{B}(PP, [x_1]_1, [x_2]_1, [x_3]_1, [x_1 x_2]_1, [x_1 x_3]_1, [x_1 x_2 x_3]_1)$**:** First, runs $\mathcal{A}$ to obtain a signature query $(m_1, \ldots, m_n)$. Then $\mathcal{B}$ chooses $y_i \xleftarrow{\$} [2^{\ell+\lambda}]$ and sets $R_i := [y_i]_1 + m_i \cdot [x_3]_1$ for $i \in [n]$. It sets $X := [x_1]_1$, $Y := [x_2]_1$, $Z := [x_3]_1$, $S := [x_1 x_2]_1$, $T := [x_1 x_3]_1$, $U := [x_1 x_2 x_3]_1$ and $pk := (PP, \{R_i\}_{i \in [n]}, X, Y, Z, S, T, U)$. For $i = 1, 2, \ldots, n$ it sets $\Lambda_i := y_i \cdot [x_2]_1$, $\Gamma_i := y_i \cdot [x_1 x_2]_1$ and $\sigma_i := (\Lambda_i, \Gamma_i)$. Then $\mathcal{B}$ gives pk and $\{\sigma_i\}_{i \in [n]}$ to $\mathcal{A}$. Let $(f^*, m^*, \sigma^* = (\Lambda^*, \Gamma^*))$ be the forgery output by $\mathcal{A}$. Let d^* be the degree of f^*. Then $\mathcal{B}$ computes $\sigma' = (\Lambda', \Gamma') \xleftarrow{\$} \mathsf{Eval}(pk, f^*, \{(m_i, \sigma_i)\}_{i \in [n]})$, $\alpha := (f^*(m_1, \ldots, m_n) - m^*)(\Lambda^* - \Lambda')$ and outputs $e(\alpha, U_{d_{\mathsf{max}} - d^*})$ where $U_{d_{\mathsf{max}} - d^*}$ denotes $e_{d_{\mathsf{max}} - d^*}(U, \ldots, U)$. (If $d_{\mathsf{max}} = d^*$, then it outputs α.)

We show that $\mathcal{B}$ works correctly. It is easy to verify that the statistical distance between pk simulated by $\mathcal{B}$ and pk in the real scheme is negligible. We note that x, y and z are implicitly set to be x_1, x_2 and x_3 respectively and r_i is set to be $y_1 + m_i \cdot x_3$ for $i \in [n]$. We have $x_2 \cdot (R_i - m_i \cdot Z) = x_2 \cdot [y_i]_1 = y_i \cdot [x_2]_1$ and $x_1 x_2 \cdot (R_i - m_i \cdot Z) = y_i \cdot [x_1 x_2]_1$. Therefore signatures for m_i are simulated correctly. Therefore if $\mathcal{A}$ breaks the selective security of the scheme, then it outputs $(f^*, m^*, \sigma^* = (\Lambda^*, \Gamma^*))$ such that

$$e(R - m^* \cdot Z_{d^*}, W_{d^*}) = e(\Lambda^*, [1]_{d^*}),$$

$$e(\Gamma, [1]_1) = e(\Lambda^*, X)$$

and $m^* \neq f^*(m_1, \ldots, m_n)$ hold with non-negligible probability where we have $R = [f(r_1, \ldots, r_n)]_{d^*}$, $Z_d = [x_3]_{d^*}$ and $W_d = [x_1^{d^*-1} x_2^{d^*}]_{d^*}$. On the other hand, by the correctness of the scheme, we have

$$e(R - f^*(m_1, \ldots, m_n) \cdot Z_{d^*}, W_{d^*}) = e(\Lambda', [1]_{d^*}),$$
$$e(\Gamma', [1]_1) = e(\Lambda', X).$$

Therefore we have

$$e((f^*(m_1, \ldots, m_n) - m^*) \cdot Z_{d^*}, W_{d^*}) = e(\Lambda^* - \Lambda', [1]_{d^*}).$$

This is equivalent to $\Lambda^* - \Lambda' = (f^*(m_1, \ldots, m_n) - m^*) \cdot [x_1^{d^*-1}(x_2 x_3)^{d^*}]_{d^*}$. Since $f^*(m_1, \ldots, m_n) - m^* \in \{1, -1\}$, we have $\alpha = (f^*(m_1, \ldots, m_n) - m^*) \cdot (\Lambda^* - \Lambda') = (f^*(m_1, \ldots, m_n) - m^*)^2 \cdot [x_1^{d^*-1}(x_2 x_3)^{d^*}]_d = [x_1^{d^*-1}(x_2 x_3)^{d^*}]_{d^*}$ and thus $e(\alpha, U_{d_{\mathsf{max}}-d^*}) = [x_1^{d_{\mathsf{max}}-1}(x_2 x_3)^{d_{\mathsf{max}}}]_{d_{\mathsf{max}}}$. Therefore $\mathcal{B}$ succeeds in breaking the d_{max}-APMDH assumption.

□

Acknowledgment

We thank anonymous reviewers for helpful comments and Dan Brown for pointing out [Bro16, Appendix B]. The second author is supported by JSPS KAKENHI Grant Number 16K16068, the second and third authors are supported by JST CREST Grant Number JPMJCR19F6, and the fourth author is supported by JST CREST Grant Number JPMJCR14D6 and JSPS KAKENHI Grant Number JP16H02780.

References

[AC18] Salim Ali Altug and Yilei Chen. A candidate group with infeasible inversion. Cryptology ePrint Archive, Report 2018/926, 2018. https://eprint.iacr.org/2018/926. 5, 16

[AFH+16] Martin R. Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. Multilinear maps from obfuscation. In *TCC 2016-A Part I*, pages 446–473, 2016. 4, 6

[AJR08] Kristina Altmann, Tibor Jager, and Andy Rupp. On black-box ring extraction and integer factorization. In *ICALP 2008*, pages 437–448, 2008. 4

[AM09] Divesh Aggarwal and Ueli M. Maurer. Breaking RSA generically is equivalent to factoring. In *EUROCRYPT*, pages 36–53, 2009. 2, 4

[AMS11] Divesh Aggarwal, Ueli Maurer, and Igor Shparlinski. The Equivalence of Strong RSA and Factoring in the Generic Ring Model of Computation. In *WCC*, pages 17–26, Paris, France, April 2011. 2

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012. 11, 29

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, pages 224–241, 2009. 40

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001. 3, 4, 37

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In *CRYPTO 2019 (to appear)*, 2019. 6

[Bro16] Daniel R. L. Brown. Breaking RSA may be as difficult as factoring. *J. Cryptology*, 29(1):220–241, 2016. 4, 13, 33

[BS84] László Babai and Endre Szemerédi. On the complexity of matrix group problems I. In *FOCS*, pages 229–240, 1984. 1

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002. 2, 3, 10, 11, 26

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO*, 2014. 26, 27, 28

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *CRYPTO*, pages 371–389, 2014. 3, 11, 20, 29, 30

[Che04] Qi Cheng. On the ultimate complexity of factorials. *Theor. Comput. Sci.*, 326(1-3):419–429, 2004. 9, 18, 19

[CL09] Jung Hee Cheon and Dong Hoon Lee. A note on self-bilinear maps. *Bulletin of the Korean Mathematical Society*, 46(2):303–309, 2009. 2, 5, 10, 12, 24

[DK02] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *EUROCRYPT*, pages 256–271, 2002. 2, 4

[FHHL18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. In *PKC*, 2018. 4, 6

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013. 10, 12, 20

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. 37

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989. 27

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *TCC 2016-B, Part II*, pages 241–268, 2016. 37

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC 2015*, pages 469–477, 2015. 29

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. 37

[HK09] Dennis Hofheinz and Eike Kiltz. The group of signed quadratic residues and applications. In *CRYPTO*, pages 637–653, 2009. 9

[Hoh03] Susan Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, Massachusetts Institute of Technology, 2003. 5, 16

[ILOP04] Jim Irrer, Satyanarayana Lokam, Lukasz Opyrchal, and Atul Prakash. Infeasible group inversion and broadcast encryption. Technical report, University of Michigan Electrical Engineering and Computer Science Tech Note CSE-TR-485-04, 2004. 5, 16

[JS09] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. In *ASIACRYPT*, pages 399–416, 2009. 2, 3, 6, 7, 16

[KKS15] Jinsu Kim, Sungwook Kim, and Jae Hong Seo. Multilinear map via scale-invariant FHE: enhancing security and efficiency. *IACR Cryptology ePrint Archive*, 2015:992, 2015. 2, 3, 5, 6, 9

[KRS15] Dakshita Khurana, Vanishree Rao, and Amit Sahai. Multi-party key exchange for unbounded parties from indistinguishability obfuscation. In *ASIACRYPT 2015 I*, pages 52–75, 2015. 27

[Lee04] Hyang-Sook Lee. A self-pairing map and its applications to cryptography. *Applied Mathematics and Computation*, 151(3):671–678, 2004. 5

[Len87] Jr. Lenstra, H. W. Factoring integers with elliptic curves. *The Annals of Mathematics*, 126(3):pp. 649–673, 1987. 12

[Lip94] Richard J. Lipton. Straight-line complexity and integer factorization. In *ANTS-I*, pages 71–79, 1994. 12

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988. 37

[LR06] Gregor Leander and Andy Rupp. On the equivalence of RSA and factoring regarding generic ring algorithms. In *ASIACRYPT*, pages 241–251, 2006. 2

[Mar13] Klas Markström. The straight line complexity of small factorials and primorials. *CoRR*, abs/1306.3091, 2013. 9, 18

[McC88] Kevin S. McCurley. A key distribution system equivalent to factoring. *J. Cryptology*, 1(2):95–105, 1988. 9

[Mol03] David Molnar. Homomorphic signature schemes. Bachelor's thesis, Harvard College, 2003. 5, 16

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *CRYPTO Part II*, pages 629–658, 2016. 37

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):91–101, 1994. 1

[PS15] Omer Paneth and Amit Sahai. On the equivalence of obfuscation and multilinear maps. *IACR Cryptology ePrint Archive*, 2015:791, 2015. 6

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. 2

[Seo17] Jae Hong Seo. personal communication, 2017. 6

[Seu13] Yannick Seurin. New constructions and applications of trapdoor DDH groups. In *Public Key Cryptography*, pages 443–460, 2013. 9

[Sha79] Adi Shamir. Factoring numbers in o(log n) arithmetic steps. *Inf. Process. Lett.*, 8(1):28–31, 1979. 9, 19

[Shm85] Zahava Shmuely. Composite diffie-hellman public-key generating systems are hard to break. Technical Report 356, Computer Science Department, Technion, Israel, 1985. 9

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997. 1

[SS95] Michael Shub and Steve Smale. On the intractability of hilbertfs nullstellensatz and an algebraic version of g $np \neq p$? h. *Duke Math. J.*, 81(1):47–54, 1995. 9, 18

[YHK16] Takashi Yamakawa, Goichiro Hanaoka, and Noboru Kunihiro. Generalized hardness assumption for self-bilinear map with auxiliary information. In *ACISP*, pages 269–284, 2016. 20, 22

[YYHK17] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications. *Algorithmica*, 79(4):1286–1317, 2017. 2, 6, 9, 10

A Instantiation of Inversion-hard Ring via VBB Obfuscation

Here, we give a concrete instantiation of a ring scheme that satisfies the c-inversion assumption based on a VBB obfuscation. Since it is known that there does not exist a VBB obfuscation, we consider our construction just as a proof of concept.

A.1 Definitions

Here, we give definitions needed for stating our construction.

Pseudorandom permutation. Here, we define a pseudorandom permutation (PRP). A pseudorandom permutation on $\{0,1\}^\ell$ consists of PPT algorithms $(\mathsf{PRPGen}, \mathsf{PRP}, \mathsf{PRP}^{-1})$.

$\mathsf{PRPGen}(1^\lambda) \to K$**:** This algorithm takes the security parameter 1^λ as input and output a key K.

$\mathsf{PRP}(K, x) \to y$**:** This algorithm takes a key K and $x \in \{0,1\}^\ell$ as input, and outputs $y \in \{0,1\}^\ell$.

$\mathsf{PRP}(K, y) \to x$**:** This algorithm takes a key K and $y \in \{0,1\}^\ell$ as input and outputs $x \in \{0,1\}^\ell$

As correctness we require that for every $\lambda \in \mathbb{N}$, every $K \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$ and every $x \in \{0,1\}^\ell$, we have $\mathsf{PRP}^{-1}(K, \mathsf{PRP}(K, x)) = x$.

As security, we require the following. For any PPT distinguisher $\mathcal{D}$,

$$\Pr[1 \xleftarrow{\$} \mathcal{D}^{\mathsf{PRP}(K,\cdot),\mathsf{PRP}^{-1}(K,\cdot)}] - \Pr[1 \xleftarrow{\$} \mathcal{D}^{\sigma(\cdot),\sigma^{-1}(\cdot)}]$$

is negligible where $K \xleftarrow{\$} \mathsf{PRPGen}(1^\lambda)$ and σ is chosen from the set of all permutations over $\{0,1\}^\ell$. It is known that there exists a PRP if there exists a one-way function [LR88, GGM86, HILL99].

Virtual black-box obfuscation We define a virtual black-box (VBB) obfuscation.

Definition 17. *(Virtual black-box obfuscation [BGI+01].) For a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, we say that a PPT oracle machine $\mathcal{O}$ is a virtual black-box (VBB) obfuscator if the following conditions are satisfied.*

Functionality: *For every $\lambda \in \mathbb{N}$, every $C \in \mathcal{C}_\lambda$ and every input x to C, there exists a negligible function* negl *such that*

$$\Pr[\mathcal{O}(C)(x) \neq C(x)] \leq \mathsf{negl}(|C|)$$

where the probability is over the coins of $\mathcal{O}$.

Virtual black-box: *For every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ and a negligible function μ such that for every $\lambda \in \mathbb{N}$ and every $C \in \mathcal{C}_\lambda$,*

$$|\Pr[1 \xleftarrow{\$} \mathcal{A}(\mathcal{O}(C))] - \Pr[1 \xleftarrow{\$} S^C(1^{|C|})]| \leq \mu(|C|)$$

where the probabilities are over the coins of $\mathcal{D}$, $\mathcal{A}$, $\mathcal{S}$ and $\mathcal{O}$.

Definition 18. *(VBB obfuscation for P/poly.) We say that $\mathcal{O}$ is a VBB obfuscator for P/poly if for every circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ such that every circuit in $\mathcal{C}_\lambda$ is of size* $\mathsf{poly}(\lambda)$*, $\mathcal{O}$ is a VBB obfuscator for $\mathcal{C}$.*

Barak et al. proved that a VBB obfuscator for *P/poly* does not exist in the standard model. On the other hand, Garg et al. [GMM+16] proved that there exists a VBB obfuscator for *P/poly* in the weak multilinear map model [MSZ16] if there exists a pseudorandom function (PRF) computable in NC^1.

A.2 Construction

Construction. We construct a ring scheme based on a pseudorandom permutation (PRP) $(\mathsf{PRPGen}, \mathsf{PRP}, \mathsf{PRP}^{-1})$ on $2^{\ell'}$ where we assume $\ell' \geq \ell + \lambda$ and a VBB obfuscator $\mathcal{O}$ in the weak multilinear map model. The definition of a PRP is given in Appendix A.1. For any prime c, we construct a ring scheme as follows.

$\mathsf{GGen}(1^\lambda) \to PP$: This algorithm generates $p \xleftarrow{\$} \mathcal{P}_\ell$ and $K \leftarrow \mathsf{KGen}(1^\lambda)$, and computes $\tilde{C}_{\mathsf{ring}} \xleftarrow{\$} \mathcal{O}(C_{\mathsf{ring}})$ where the description of C_{ring} is given in Fig. 1. This specifies a ring $R := \{\mathsf{PRP}(K, X) : X \in \mathbb{Z}_p\}$ whose multiplicative identity is $[1]_R := \mathsf{PRP}(K, 1)$ and whose ring operations are defined by $\mathsf{PRP}(K, X) \circ \mathsf{PRP}(K, Y) := \mathsf{PRP}(K, (X \circ Y \mod p))$ for $X, Y \in \mathbb{Z}_p$ and $\circ \in \{+, -, \cdot\}$. Intuitively, the circuit C_{ring} makes it possible to compute operations on this ring efficiently. More precisely, it has three modes that are $\mathsf{identity}$,

Circuit $C_{\mathsf{ring}}[p,K](\mathsf{mode},\mathsf{in})$:	Circuit $C'_{\mathsf{ring}}[p,K](\mathsf{mode},\mathsf{in})$:
If $\mathsf{mode} = \mathsf{identity}$	If $\mathsf{mode} = \mathsf{identity}$
Return $\mathsf{PRP}(K,1)$	Return $\mathsf{PRP}(K,1)$
If $\mathsf{mode} = \mathsf{operation}$	If $\mathsf{mode} = \mathsf{operation}$
Parse $(\mathrm{Enc}_1,\mathrm{Enc}_2,\circ) \leftarrow \mathsf{in}$	Parse $(\mathrm{Enc}_1,\mathrm{Enc}_2,\circ) \leftarrow \mathsf{in}$
$X_1 \leftarrow \mathsf{PRP}^{-1}(K,\mathrm{Enc}_1)$	$X_1 \leftarrow \mathsf{PRP}^{-1}(K,\mathrm{Enc}_1)$
$X_2 \leftarrow \mathsf{PRP}^{-1}(K,\mathrm{Enc}_2)$	$X_2 \leftarrow \mathsf{PRP}^{-1}(K,\mathrm{Enc}_2)$
If $X_1 \notin \mathbb{Z}_p$ or $X_2 \notin \mathbb{Z}_p$	If $X_1 \notin \mathbb{Z}_p$ or $X_2 \notin \mathbb{Z}_p$
Return $\perp$	Return $\perp$
Else	Else
$X' := X_1 \circ X_2 \mod p$	$X' := X_1 \circ X_2 \mod p$
$\mathrm{Enc}' \leftarrow \mathsf{PRP}(K,X')$	$\mathrm{Enc}' \leftarrow \mathsf{PRP}(K,X')$
Return Enc'	Return Enc'
If $\mathsf{mode} = \mathsf{test}$	If $\mathsf{mode} = \mathsf{test}$
Parse $\mathrm{Enc}^* \leftarrow \mathsf{in}$	Return $\perp$
$X^* \leftarrow \mathsf{PRP}^{-1}(K,\mathrm{Enc}^*)$	
If $X^* \notin \mathbb{Z}_p$	
Return $\perp$	
Else	
If $X^* = [c]^{-1}_{\mathbb{Z}_p}$	
Return true	
Else	
Return $\perp$	

Figure 1: Description of circuits $C_{\mathsf{ring}}[p,K]$ and $C'_{\mathsf{ring}}[p,K]$.

$\mathsf{operation}$ and test modes depending on the first component of an input. In the $\mathsf{identity}$ mode, it outputs the multiplicative identity of a ring, in the $\mathsf{operation}$ mode, it simulates a ring operation and in the test mode, the circuit tests whether the c-inversion assumption was broken or not. It outputs $PP := \tilde{C}_{\mathsf{ring}}$.

$\mathsf{Add}(\mathrm{Enc}_1,\mathrm{Enc}_2) \to \mathrm{Enc}'$: This algorithm computes $\mathrm{Enc}' \leftarrow \tilde{C}_{\mathsf{ring}}(\mathrm{Enc}_1,\mathrm{Enc}_2,+)$ and output Enc'.

$\mathsf{Sub}(\mathrm{Enc}_1,\mathrm{Enc}_2) \to \mathrm{Enc}'$: This algorithm computes $\mathrm{Enc}' \leftarrow \tilde{C}_{\mathsf{ring}}(\mathrm{Enc}_1,\mathrm{Enc}_2,-)$ and outputs Enc'.

$\mathsf{Mult}(\mathrm{Enc}_1,\mathrm{Enc}_2) \to \mathrm{Enc}'$: This algorithm computes $\mathrm{Enc}' \leftarrow \tilde{C}_{\mathsf{ring}}(\mathrm{Enc}_1,\mathrm{Enc}_2,\cdot)$ and outputs Enc'.

Remark 9. *The* test *mode of* $C_{\mathsf{ring}}[p,K]$ *is not necessary for the correctness of the scheme. This mode is used only in the security proof.*

We prove that the c-inversion assumption holds for the above ring scheme under the factoring assumption w.r.t. unbalanced moduli.

Theorem 14. *If the factoring assumption w.r.t. unbalanced moduli holds, then the c-inversion assumption holds for the above ring scheme.*

Proof. Let $C'_{\mathsf{ring}}[p,K]$ be a circuit that works similarly to $C_{\mathsf{ring}}[p,K]$ except that it always outputs $\perp$ when $\mathsf{mode} = \mathsf{test}$. (The full description is given in Fig. 1.) Let $\mathcal{A}$ be a PPT algorithm against the c-inversion assumption for the scheme. Then we consider another PPT

algorithm $\mathcal{B}$ that distinguish obfuscations of $C_{\mathsf{ring}}[p,K]$ and $C'_{\mathsf{ring}}[p,K]$. The description of $\mathcal{B}$ is as follows.

$\mathcal{B}(\tilde{C})$ $\mathcal{B}$ is given an obfuscated circuit $\tilde{C}$ as input and runs $\mathrm{Enc}^* \xleftarrow{\$} \mathcal{A}(\tilde{C})$. It outputs as $\tilde{C}(\mathsf{test}, \mathrm{Enc}^*)$ outputs.

If $\tilde{C}$ is an obfuscation of $C_{\mathsf{ring}}[p,K]$, $\mathcal{B}$ outputs 1 in the test mode if and only if $\mathcal{A}$ succeeds in breaking the c-inversion assumption, and if $\tilde{C}$ is an obfuscation of $C_{\mathsf{ring}}[p,K]$, $\mathcal{B}$ never outputs 1. Therefore we have

$$\mathsf{Adv}^{\mathsf{gen\text{-}inv}}_{\mathcal{A}}(\lambda) = \Pr[1 \xleftarrow{\$} \mathcal{B}(\tilde{C}) : \tilde{C} \xleftarrow{\$} \mathcal{O}(C_{\mathsf{ring}}[p,K])]$$

and

$$0 = \Pr[1 \xleftarrow{\$} \mathcal{B}(\tilde{C}) : \tilde{C} \xleftarrow{\$} \mathcal{O}(C'_{\mathsf{ring}}[p,K])]$$

where $p \xleftarrow{\$} \mathcal{P}_\ell$ and $K \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$.

On the other hand, by the property of a VBB obfuscation, there exists a PPT algorithm $\mathcal{S}$ such that

$$|\Pr[1 \xleftarrow{\$} \mathcal{B}(\tilde{C}) : \tilde{C} \xleftarrow{\$} \mathcal{O}(C_{\mathsf{ring}}[p,K])] - \Pr[1 \xleftarrow{\$} \mathcal{S}^{C_{\mathsf{ring}}[p,K]}(1^M)]|$$

and

$$|\Pr[1 \xleftarrow{\$} \mathcal{B}(\tilde{C}) : \tilde{C} \xleftarrow{\$} \mathcal{O}(C'_{\mathsf{ring}}[p,K])] - \Pr[1 \xleftarrow{\$} \mathcal{S}^{C'_{\mathsf{ring}}[p,K]}(1^M)]|$$

are negligible where $p \xleftarrow{\$} \mathcal{P}_\ell$ and $K \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$ and M denotes the maximum size of $C_{\mathsf{ring}}[p,K]$ and $C'_{\mathsf{ring}}[p,K]$.

Therefore we have

$$\mathsf{Adv}^{\mathsf{gen\text{-}inv}}_{\mathcal{A}}(\lambda) \leq |\Pr[1 \xleftarrow{\$} \mathcal{S}^{C_{\mathsf{ring}}[p,K]}(1^\lambda, 1^M)] - \Pr[1 \xleftarrow{\$} \mathcal{S}^{C'_{\mathsf{ring}}[p,K]}(1^\lambda, 1^M)]| + \mathsf{negl}(\lambda)$$

where negl is a negligible function, $p \xleftarrow{\$} \mathcal{P}_\ell$ and $K \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$. Here, we prove the following lemma.

Lemma 6. *$|\Pr[1 \xleftarrow{\$} \mathcal{S}^{C_{\mathsf{ring}}[p,K]}(1^\lambda, 1^M)] - \Pr[1 \xleftarrow{\$} \mathcal{S}^{C'_{\mathsf{ring}}[p,K]}(1^\lambda, 1^M)]|$ is negligible where $p \xleftarrow{\$} \mathcal{P}_\ell$ and $K \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$.*

It is clear that Theorem 14 is proven if the above lemma is proven. Intuitively, Lemma 6 can be reduced to the c-inversion assumption in the $\mathcal{D}_R$-generic ring model because $\mathcal{S}$ is only allowed to compute ring operations in a black-box manner through its oracle, and unless $\mathcal{S}$ queries the witness that it breaks the c-inversion assumption in the test mode, $C_{\mathsf{ring}}[p,K]$ and $C'_{\mathsf{ring}}[p,K]$ works completely the same. The full proof of Lemma 6 is given below. □

Proof. (of Lemma 6) We prove it by considering the following sequence of games.

Game 0**:** In this game, $\bar{\mathcal{S}}$ is given 1^M as input and allowed to access the oracle $C_{\mathsf{ring}}[p,K]$ described in Fig 1 where $p \xleftarrow{\$} \mathcal{P}_\ell$ and $K \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.

Game 1**:** In this game, the oracle is replaced by $\mathcal{O}^{(1)}_{\mathsf{ring}}[p,\sigma]$ described in Fig. 2 where $p \xleftarrow{\$} \mathcal{P}_\ell$ and σ is a random permutation on $\{0,1\}^{\ell'}$. $\mathcal{O}^{(1)}_{\mathsf{ring}}[p,\sigma]$ works similarly to $C_{\mathsf{ring}}[p,K]$ except that $\mathsf{PRP}(K,\cdot)$ and $\mathsf{PRP}^{-1}(K,\cdot)$ are replaced by σ and σ^{-1} respectively.

Oracle $\mathcal{O}^{(1)}_{\mathsf{ring}}[p,\sigma](\mathsf{mode},\mathsf{in})$:	Oracle $\mathcal{O}^{(2)}_{\mathsf{ring}}[p,\sigma](\mathsf{mode},\mathsf{in})$:
If mode = identity	If mode = identity
Return $\sigma(1)$	Return $\sigma(1)$
If mode = operation	If mode = operation
Parse $(\mathrm{Enc}_1,\mathrm{Enc}_2,\circ) \leftarrow \mathsf{in}$	Parse $(\mathrm{Enc}_1,\mathrm{Enc}_2,\circ) \leftarrow \mathsf{in}$
$X_1 \leftarrow \sigma^{-1}(\mathrm{Enc}_1)$, $X_2 \leftarrow \sigma^{-1}(\mathrm{Enc}_2)$	If $\mathrm{Enc}_1 \notin L$ or $\mathrm{Enc}_2 \notin L$
If $X_1 \notin \mathbb{Z}_p$ or $X_2 \notin \mathbb{Z}_p$	Return $\perp$
Return $\perp$	Else
Else	$X_1 \leftarrow \sigma^{-1}(\mathrm{Enc}_1)$, $X_2 \leftarrow \sigma^{-1}(\mathrm{Enc}_2)$
$X' := X_1 \circ X_2 \mod p$	$X' := X_1 \circ X_2 \mod p$
$\mathrm{Enc}' \leftarrow \sigma(X')$	$\mathrm{Enc}' \leftarrow \sigma(X')$
Return Enc'	$L \leftarrow L \cup \{\mathrm{Enc}'\}$
If mode = test	Return Enc'
Parse $\mathrm{Enc}^* \leftarrow \mathsf{in}$	If mode = test
$X^* \leftarrow \sigma^{-1}(\mathrm{Enc}^*)$	Parse $\mathrm{Enc}^* \leftarrow \mathsf{in}$
If $X^* \notin \mathbb{Z}_p$	If $\mathrm{Enc}^* \notin L$
Return $\perp$	Return $\perp$
Else	Else
If $X^* = [c]^{-1}_{\mathbb{Z}_p}$	$X^* \leftarrow \sigma^{-1}(\mathrm{Enc}^*)$
Return true	If $X^* = [c]^{-1}_{\mathbb{Z}_p}$
Else	Return true
Return $\perp$	Else
	Return $\perp$

Figure 2: Description of oracles $\mathcal{O}^{(1)}_{\mathsf{ring}}[p,\sigma]$ and $\mathcal{O}^{(2)}_{\mathsf{ring}}[p,\sigma]$.

Game 2: In this game, the oracle is replaced by $\mathcal{O}^{(2)}_{\mathsf{ring}}[p,\sigma]$ described in Fig. 2 where $p \xleftarrow{\$} \mathcal{P}_\ell$, σ is a random permutation on $\{0,1\}^{\ell'}$, and a set $L \subset \{0,1\}^{\ell'}$ is initialized to be $\{\sigma(1)\}$ at the beginning of the game. Intuitively, L records ciphertexts issued by the oracle, and $\mathcal{O}^{(2)}_{\mathsf{ring}}[p,\sigma]$ works similarly to $\mathcal{O}^{(1)}_{\mathsf{ring}}[p,\sigma]$ except that it returns $\perp$ if an operation or test query contains a ciphertext outside of L.

Game 3: In this game, the oracle is replaced by $\mathcal{O}^{(3)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]$ described in Fig. 4 where $p \xleftarrow{\$} \mathcal{P}_\ell$, $\mathrm{Enc}_0 \xleftarrow{\$} \{0,1\}^{\ell'}$, a set $L' \subset \mathbb{Z}_p \times \{0,1\}^{\ell'}$ is initialized to be $\{(1,\mathrm{Enc}_0)\}$ at the beginning of the game and L'_2 denotes the set of all $\mathrm{Enc} \in \{0,1\}^{\ell'}$ such that $(X,\mathrm{Enc}) \in L'$ for some $X \in \mathbb{Z}_p$. We note that if there exists X, Y such that $(X,\mathrm{Enc}),(Y,\mathrm{Enc}) \in L'$, then we always have $X = Y$, and thus the step to find X_1, X_2 such that $(X_1,\mathrm{Enc}_1),(X_2,\mathrm{Enc}_2) \in L'$ is well-defined. $\mathcal{O}^{(3)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]$ works similarly to $\mathcal{O}^{(2)}_{\mathsf{ring}}[p,\sigma]$ except that instead of choosing σ at the beginning of the game, the oracle $\mathcal{O}^{(3)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]$ simulates σ by a "lazy sampling" [BCLO09] where a value of σ is assigned whenever that is needed in the game. Intuitively, $(X,\mathrm{Enc}) \in L$ means that Enc is assigned to be $\sigma(X)$.

Game 4: In this game, the oracle is replaced by $\mathcal{O}^{(4)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]$ described in Fig. 4 where $p \xleftarrow{\$} \mathcal{P}_\ell$, $\mathrm{Enc}_0 \xleftarrow{\$} \{0,1\}^{\ell'}$, a set $L'' \subset \mathbb{Z} \times \{0,1\}^{\ell'}$ is initialized to be $\{(0,\mathrm{Enc}_0)\}$, an NI-SLP P is initialized to be an empty string at the beginning of the game, and L''_2 denotes the set of all $\mathrm{Enc} \in \{0,1\}^{\ell'}$ such that $(i,\mathrm{Enc}) \in L''$ for some $i \in \mathbb{Z}$. In the step to find i_1, i_2 such that $(i_1,\mathrm{Enc}_1),(i_2,\mathrm{Enc}_2) \in L''$, if there exist multiple such i_1, i_2, the oracle picks the smallest one. $\mathrm{P}||(i_1,i_2,\circ)$ means to append $(i_1,i_2,\circ)$ to P. $\mathcal{O}^{(4)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]$ works similarly to $\mathcal{O}^{(3)}_{\mathsf{ring}}[p,\sigma]$ except that it maintains assigned values of σ in another way by using L'' and

Oracle $\mathcal{O}^{(3)}_{\mathsf{ring}}[p, \mathrm{Enc}_0](\mathsf{mode}, \mathsf{in})$:	Oracle $\mathcal{O}^{(4)}_{\mathsf{ring}}[p, \mathrm{Enc}_0](\mathsf{mode}, \mathsf{in})$:
If $\mathsf{mode} = \mathsf{identity}$	If $\mathsf{mode} = \mathsf{identity}$
Return Enc_0	Return Enc_0
If $\mathsf{mode} = \mathsf{operation}$	If $\mathsf{mode} = \mathsf{operation}$
Parse $(\mathrm{Enc}_1, \mathrm{Enc}_2, \circ) \leftarrow \mathsf{in}$	Parse $(\mathrm{Enc}_1, \mathrm{Enc}_2, \circ) \leftarrow \mathsf{in}$
If $\mathrm{Enc}_1 \notin L'_2$ or $\mathrm{Enc}_2 \notin L'_2$	If $\mathrm{Enc}_1 \notin L''_2$ or $\mathrm{Enc}_2 \notin L_2$
Return $\perp$	Return $\perp$
Else	Else
Find X_1, X_2	Find i_1, i_2
s.t. $(X_1, \mathrm{Enc}_1), (X_2, \mathrm{Enc}_2) \in L'$	s.t. $(i_1, \mathrm{Enc}_1), (i_2, \mathrm{Enc}_2) \in L''$
$X' := X_1 \circ X_2 \mod p$	$\mathrm{P} \leftarrow \mathrm{P}\|\|(i_1, i_2, \circ)$
If $\exists$ Enc such that $(X, \mathrm{Enc}) \in L'$	If $\exists i$ s.t. $(i, \mathrm{Enc}) \in L''$ and $\mathrm{P}_i(\mathbb{Z}_p) = \mathrm{P}(\mathbb{Z}_p)$
$\mathrm{Enc}' \leftarrow \mathrm{Enc}$	$\mathrm{Enc}' \leftarrow \mathrm{Enc}$
Else	Else
$\mathrm{Enc}' \xleftarrow{\$} \{0,1\}^{\ell'} \setminus \lvert L'_2 \rvert$	$\mathrm{Enc}' \xleftarrow{\$} \{0,1\}^{\ell'} \setminus \lvert L''_2 \rvert$
$L' \leftarrow L' \cup \{(X', \mathrm{Enc}')\}$	$L'' \leftarrow L'' \cup \{(\lvert \mathrm{P} \rvert, \mathrm{Enc}')\}$
Return Enc'	Return Enc'
If $\mathsf{mode} = \mathsf{test}$	If $\mathsf{mode} = \mathsf{test}$
Parse $\mathrm{Enc}^* \leftarrow \mathsf{in}$	Parse $\mathrm{Enc}^* \leftarrow \mathsf{in}$
If $\mathrm{Enc}^* \notin L'_2$	If $\mathrm{Enc}^* \notin L''_2$
Return $\perp$	Return $\perp$
Else	Else
Find X^* s.t. $(X^*, \mathrm{Enc}^*) \in L'$	Find X^* s.t. $(X^*, \mathrm{Enc}^*) \in L''$
If $X^* = [c]^{-1}_{\mathbb{Z}_p}$	If $X^* = [c]^{-1}_{\mathbb{Z}_p}$
Return true	Return true
Else	Else
Return $\perp$	Return $\perp$

Figure 3: Description of oracles $\mathcal{O}^{(3)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ and $\mathcal{O}^{(4)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$.

P. Intuitively, $(i, \mathrm{Enc}) \in L''$ means that Enc is assigned to be $\sigma(\mathrm{P}_i(\mathbb{Z}_p))$.

Game 5: In this game, the oracle is replaced by $\mathcal{O}^{(5)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ described in Fig. 3 where $p \xleftarrow{\$} \mathcal{P}_\ell$, $\mathrm{Enc}_0 \xleftarrow{\$} \{0,1\}^{\ell'}$, $L'' \subset \mathbb{Z} \times \{0,1\}^{\ell'}$ is initialized to be $\{(0, \mathrm{Enc}_0)\}$, an NI-SLP P is initialized to be an empty string at the beginning of the game, and L''_2 denotes the set of all $\mathrm{Enc} \in \{0,1\}^{\ell'}$ such that $(i, \mathrm{Enc}) \in L''$ for some $i \in \mathbb{Z}$. We note that $\mathcal{O}^{(5)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ is the same as $\mathcal{O}^{(4)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ except that it always returns $\perp$ if $\mathsf{mode} = \mathsf{test}$.

Game 6: In this game, the oracle is replaced by $\mathcal{O}^{(6)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ described in Fig. 3 where $p \xleftarrow{\$} \mathcal{P}_\ell$, $\mathrm{Enc}_0 \xleftarrow{\$} \{0,1\}^{\ell'}$, a set $L' \subset \mathbb{Z}_p \times \{0,1\}^{\ell'}$ is initialized to be $\{(1, \mathrm{Enc}_0)\}$ at the beginning of the game and L'_2 denotes the set of all $\mathrm{Enc} \in \{0,1\}^{\ell'}$ such that $(X, \mathrm{Enc}) \in L'$ for some $X \in \mathbb{Z}_p$. We note that $\mathcal{O}^{(6)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ is the same as $\mathcal{O}^{(3)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ except that it always returns $\perp$ if $\mathsf{mode} = \mathsf{test}$.

Game 7: In this game, the oracle is replaced by $\mathcal{O}^{(7)}_{\mathsf{ring}}[p, \sigma]$ described in Fig. 5 where $p \xleftarrow{\$} \mathcal{P}_\ell$, σ is a random permutation on $\{0,1\}^{\ell'}$, and a set $L \subset \{0,1\}^{\ell'}$ is initialized to be $\{\sigma(1)\}$ at the beginning of the game. We note that $\mathcal{O}^{(7)}_{\mathsf{ring}}[p, \sigma]$ is the same as $\mathcal{O}^{(2)}_{\mathsf{ring}}[p, \sigma]$ except that it always returns $\perp$ if $\mathsf{mode} = \mathsf{test}$.

Game 8: In this game, the oracle is replaced by $\mathcal{O}^{(8)}_{\mathsf{ring}}[p, \sigma]$ described in Fig. 5 where $p \xleftarrow{\$} \mathcal{P}_\ell$ and

Oracle $\mathcal{O}^{(5)}_{\mathsf{ring}}[p, \mathrm{Enc}_0](\mathsf{mode}, \mathsf{in})$:	Oracle $\mathcal{O}^{(6)}_{\mathsf{ring}}[p, \mathrm{Enc}_0](\mathsf{mode}, \mathsf{in})$:
If $\mathsf{mode} = \mathsf{identity}$	If $\mathsf{mode} = \mathsf{identity}$
Return Enc_0	Return Enc_0
If $\mathsf{mode} = \mathsf{operation}$	If $\mathsf{mode} = \mathsf{operation}$
Parse $(\mathrm{Enc}_1, \mathrm{Enc}_2, \circ) \leftarrow \mathsf{in}$	Parse $(\mathrm{Enc}_1, \mathrm{Enc}_2, \circ) \leftarrow \mathsf{in}$
If $\mathrm{Enc}_1 \notin L''_2$ or $\mathrm{Enc}_2 \notin L_2$	If $\mathrm{Enc}_1 \notin L'_2$ or $\mathrm{Enc}_2 \notin L'_2$
Return $\perp$	Return $\perp$
Else	Else
Find i_1, i_2	Find X_1, X_2
s.t. $(i_1, \mathrm{Enc}_1), (i_2, \mathrm{Enc}_2) \in L''$	s.t. $(X_1, \mathrm{Enc}_1), (X_2, \mathrm{Enc}_2) \in L'$
$\mathrm{P} \leftarrow \mathrm{P}\|\|(i_1, i_2, \circ)$	$X' := X_1 \circ X_2 \mod p$
If $\exists i$ s.t. $(i, \mathrm{Enc}) \in L''$ and $\mathrm{P}_i(\mathbb{Z}_p) = \mathrm{P}(\mathbb{Z}_p)$	If $\exists$ Enc such that $(X, \mathrm{Enc}) \in L'$
$\mathrm{Enc}' \leftarrow \mathrm{Enc}$	$\mathrm{Enc}' \leftarrow \mathrm{Enc}$
Else	Else
$\mathrm{Enc}' \xleftarrow{\$} \{0,1\}^{\ell'} \setminus \lvert L''_2 \rvert$	$\mathrm{Enc}' \xleftarrow{\$} \{0,1\}^{\ell'} \setminus \lvert L'_2 \rvert$
$L'' \leftarrow L'' \cup \{(\lvert \mathrm{P} \rvert, \mathrm{Enc}')\}$	$L' \leftarrow L' \cup \{(X', \mathrm{Enc}')\}$
Return Enc'	Return Enc'
If $\mathsf{mode} = \mathsf{test}$	If $\mathsf{mode} = \mathsf{test}$
Return $\perp$	Return $\perp$

Figure 4: Description of oracles $\mathcal{O}^{(5)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ and $\mathcal{O}^{(6)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$.

σ is a random permutation on $\{0,1\}^{\ell'}$. We note that $\mathcal{O}^{(8)}_{\mathsf{ring}}[p, \sigma]$ is the same as $\mathcal{O}^{(1)}_{\mathsf{ring}}[p, \sigma]$ except that it always returns $\perp$ if $\mathsf{mode} = \mathsf{test}$.

Game 9**:** In this game, the oracle is replaced by $C'_{\mathsf{ring}}[p, K]$ where $p \xleftarrow{\$} \mathcal{P}_\ell$ and $K \xleftarrow{\$} \mathsf{PRPGen}(1^\lambda)$.

We let T_i be the event that $\mathcal{S}$ outputs 1, and let Q be the maximum number of $\mathcal{S}$'s queries. What we have to prove is that $|\Pr[T_9] - \Pr[T_1]|$ is negligible. We prove this by the following lemmas.

Lemma 7. *If* $(\mathsf{PRPGen}, \mathsf{PRP}, \mathsf{PRP}^{-1})$ *is a secure PRP, then* $\Pr[T_1] - \Pr[T_0]$ *is negligible.*

Proof. Game 1 is the same as Game 0 except that $\mathsf{PRP}(K, \cdot)$ and $\mathsf{PRP}^{-1}(K, \cdot)$ are replaced by σ and σ^{-1}. Since $\mathcal{S}$ only accesses to them as oracles, they are computationally indistinguishable by the security of a PRP. □

□

Lemma 8. *We have* $|\Pr[T_2] - \Pr[T_1]| \leq \frac{Q \cdot 2^\ell}{2^{\ell'} - Q - 1}$. *Especially,* $|\Pr[T_2] - \Pr[T_1]|$ *is negligible.*

Proof. Since at most one element is added to L at each query, we have $|L| \leq Q + 1$ throughout the game. For any $\mathrm{Enc}\{0,1\}^{\ell'} \setminus L$, $\mathcal{S}$ cannot obtain any information about $\sigma^{-1}(\mathrm{Enc})$, and thus the probability that $\sigma^{-1}(\mathrm{Enc}) \in \mathbb{Z}_p$ is $\frac{p - |L|}{2^{\ell'} - |L|} \leq \frac{2^\ell}{2^{\ell'} - Q - 1}$. Therefore the probability that $\mathcal{S}$ makes at least one query that contains $\mathrm{Enc} \in \{0,1\}^{\ell'} \setminus L$ such that $\sigma^{-1}(\mathrm{Enc}) \notin \mathbb{Z}_p$ is at most $\frac{Q \cdot 2^\ell}{2^{\ell'} - Q - 1}$. Unless this occurs, Game 1 and Game 2 are identical. Since we assume $\ell' \geq \ell + \lambda$ and Q is polynomial in λ, $\frac{Q \cdot 2^\ell}{2^{\ell'} - Q - 1}$ is negligible. □

□

The following two lemmas are easy to verify because the modifications from Game 2 to Game 3 and from Game 3 to Game 4 are just conceptual.

Lemma 9. *We have* $\Pr[T_3] = \Pr[T_2]$.

Oracle $\mathcal{O}^{(2)}_{\mathsf{ring}}[p,\sigma](\mathsf{mode},\mathsf{in})$:	Oracle $\mathcal{O}^{(8)}_{\mathsf{ring}}[p,\sigma](\mathsf{mode},\mathsf{in})$:
If $\mathsf{mode} = \mathsf{identity}$	If $\mathsf{mode} = \mathsf{identity}$
Return $\sigma(1)$	Return $\sigma(1)$
If $\mathsf{mode} = \mathsf{operation}$	If $\mathsf{mode} = \mathsf{operation}$
Parse $(\mathrm{Enc}_1, \mathrm{Enc}_2, \circ) \leftarrow \mathsf{in}$	Parse $(\mathrm{Enc}_1, \mathrm{Enc}_2, \circ) \leftarrow \mathsf{in}$
If $\mathrm{Enc}_1 \notin L$ or $\mathrm{Enc}_2 \notin L$	$X_1 \leftarrow \sigma^{-1}(\mathrm{Enc}_1)$, $X_2 \leftarrow \sigma^{-1}(\mathrm{Enc}_2)$
Return $\perp$	If $X_1 \notin \mathbb{Z}_p$ or $X_2 \notin \mathbb{Z}_p$
Else	Return $\perp$
$X_1 \leftarrow \sigma^{-1}(\mathrm{Enc}_1)$, $X_2 \leftarrow \sigma^{-1}(\mathrm{Enc}_2)$	Else
$X' := X_1 \circ X_2 \mod p$	$X' := X_1 \circ X_2 \mod p$
$CT' \leftarrow \sigma(X')$	$\mathrm{Enc}' \leftarrow \sigma(X')$
$L \leftarrow L \cup \{\mathrm{Enc}'\}$	Return Enc'
Return Enc'	If $\mathsf{mode} = \mathsf{test}$
If $\mathsf{mode} = \mathsf{test}$	Return $\perp$
Return $\perp$	

Figure 5: Description of oracles $\mathcal{O}^{(7)}_{\mathsf{ring}}[p,\sigma]$ and $\mathcal{O}^{(8)}_{\mathsf{ring}}[p,\sigma]$.

Lemma 10. *We have* $\Pr[T_4] = \Pr[T_3]$.

Lemma 11. *There exists a PPT generic algorithm* $\bar{\mathcal{S}}$ *in the* $\mathcal{D}_R$*-generic model such that* $|\Pr[T_5] - \Pr[T_4]| \leq Q \cdot \mathsf{Adv}^{\mathsf{gen-inv}}_{\bar{\mathcal{S}},\mathcal{D}_R,c}(\lambda)$ *where* $\mathcal{D}_R$ *is a distribution of a ring* $R = \mathbb{Z}_p$ *for* $p \xleftarrow{\$} \mathcal{P}_\ell$*. Especially, if the factoring assumption w.r.t. unbalanced moduli holds, then* $|\Pr[T_5] - \Pr[T_4]|$ *is negligible.*

Proof. The latter part of the lemma follows from the former part and Theorem 4. We prove the former part in the following. We construct a generic algorithm $\bar{\mathcal{S}}$ against the c-inversion assumption in the $\mathcal{D}_R$-generic ring model. Recall that $\bar{\mathcal{S}}$ has access to the oracle $\mathcal{O}_{\mathsf{ring}}$ defined in Sec.5.2. As a building block, we use an algorithm $\mathcal{S}$ that accesses to an oracle $\mathcal{O}$ (which is supposed to be either of $\mathcal{O}^{(4)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]$ or $\mathcal{O}^{(5)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]$) .

$\bar{\mathcal{S}}^{\mathcal{O}_{\mathsf{ring}}}(1^\lambda)$**:** This algorithm picks $\mathrm{Enc}_0 \xleftarrow{\$} \{0,1\}^{\ell'}$, generates a list $L'' := \{(0,\mathrm{Enc}_0)\}$, sets $ctr := 0$ and gives 1^M to $\mathcal{S}$ as input. When $\mathcal{S}$ makes a query $(\mathsf{mode},\mathsf{in})$, $\bar{\mathcal{S}}$ simulates the oracle as follows. If $\mathsf{mode} = \mathsf{identity}$, then $\bar{\mathcal{S}}$ returns Enc_0. If $\mathsf{mode} = \mathsf{operation}$ and $\mathsf{in} = (\mathrm{Enc}_1, \mathrm{Enc}_2, \circ)$, $\bar{\mathcal{S}}$ finds i_1, i_2 such that $(i_1, \mathrm{Enc}_1), (i_2, \mathrm{Enc}_2) \in L''$. If such i_1 or i_2 does not exist, then $\mathcal{S}'$ returns $\perp$ to $\mathcal{S}$ as a response of $\mathcal{S}$'s oracle $\mathcal{O}$. Otherwise $\bar{\mathcal{S}}$ makes a $\mathsf{operation}$ query $(i_1, i_2, \circ)$ to its own oracle $\mathcal{O}_{\mathsf{ring}}$ and increments $ctr \leftarrow ctr + 1$. Then for $j \in \{0, 1, \ldots, ctr-1\}$, $\bar{\mathcal{S}}$ makes an equal query (ctr, j) to its own oracle $\mathcal{O}_{\mathsf{ring}}$. If there exists $j \in \{0, 1, \ldots, ctr-1\}$ such that $\mathcal{O}_{\mathsf{ring}}$ returns true for an equal query (ctr, j), then $\bar{\mathcal{S}}$ sets $\mathrm{Enc}_{ctr} := \mathrm{Enc}_j$. (If there exist multiple such j, it simply takes the smallest one.) If there does not exist such j, then $\bar{\mathcal{S}}'$ chooses $\mathrm{Enc}_{ctr} \xleftarrow{\$} \{0,1\}^{\ell'} \setminus L''_2$. Then $\bar{\mathcal{S}}$ adds $(ctr, \mathrm{Enc}_{ctr})$ to L'' and returns Enc_{ctr} to $\mathcal{S}$ as a response by the $\mathcal{S}$'s oracle $\mathcal{O}$. If $\mathsf{mode} = \mathsf{test}$, $\bar{\mathcal{S}}$ returns $\perp$ as a response by $\mathcal{O}$. When the execution of $\mathcal{S}$ halts, $\bar{\mathcal{S}}$ picks $k \xleftarrow{\$} [Q]$ where Q denotes the number of $\mathcal{S}$'s test query. Let $\mathsf{in} = \mathrm{Enc}^*$ be the $\mathcal{S}$'s k-th test query. $\bar{\mathcal{S}}$ finds i^* such that $(i^*, \mathrm{Enc}^*) \in L''$ and outputs i^*.

First, we remark that responses from $\mathcal{O}^{(4)}_{\mathsf{ring}[p,\mathrm{Enc}_0]}$ and $\mathcal{O}^{(5)}_{\mathsf{ring}[p,\mathrm{Enc}_0]}$ are identical unless $\mathcal{S}$ queries $(\mathsf{test}, \mathrm{Enc}^*)$ such that $\mathrm{P}_{i^*}(\mathbb{Z}_p) = [c]^{-1}_{\mathbb{Z}_p}$ where i^* is an integer such that $(i^*, \mathrm{Enc}^*) \in L''$ and P is an NI-SLP maintained by an oracle. We denote the event that $\mathcal{S}$ makes such a query by F. Then we have $|\Pr[1 \xleftarrow{\$} \mathcal{S}^{\mathcal{O}_{\mathsf{ring}}(4)[p,\mathrm{Enc}_0]}(1^\lambda, 1^M)] - \Pr[1 \xleftarrow{\$} \mathcal{S}^{\mathcal{O}^{(5)}_{\mathsf{ring}}[p,\mathrm{Enc}_0]}(1^\lambda, 1^M)]| \leq \Pr[F]$. Here, we notice

that unless F occurs, $\bar{\mathcal{S}}$ perfectly simulates the environment for $\mathcal{S}$ that accesses to $\mathcal{O}^{(4)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$ or $\mathcal{O}^{(5)}_{\mathsf{ring}}[p, \mathrm{Enc}_0]$. Thus the probability that the output i^* by $\bar{\mathcal{S}}$ satisfies $\mathrm{P}_{i^*}(\mathbb{Z}_p) = [c]^{-1}_{\mathbb{Z}_p}$ where P is an NI-SLP maintained by the oracle $\mathcal{O}_{\mathsf{ring}}$, is $\Pr[F]/Q$. This probability is defined to be $\mathsf{Adv}^{\mathsf{gen\text{-}inv}}_{\bar{\mathcal{S}},\mathcal{D}_R,c}(\lambda)$. Therefore we have $\mathsf{Adv}^{\mathsf{gen\text{-}inv}}_{\bar{\mathcal{S}},\mathcal{D}_R,c}(\lambda) = \Pr[F]/Q$. Therefore the lemma follows. □
□

The following lemmas can be proven similarly to Lemma 7 to 11.

Lemma 12. *If* $(\mathsf{PRPGen}, \mathsf{PRP}, \mathsf{PRP}^{-1})$ *is a secure PRP, then* $|\Pr[T_9] - \Pr[T_5]|$ *is negligible.*

By combining the above lemmas, Lemma 6 is proven, and thus Theorem 14 is proven. □
□