A Framework for Efficient Fully-Equipped UC Commitments

Eiichiro Fujisaki

NTT Secure Platform Laboratories
fujisaki.eiichiro@lab.ntt.co.jp

Abstract

We present a general framework for constructing non-interactive universally composable (UC) commitment schemes that are secure against adaptive adversaries in the non-erasure setting under a single re-usable common reference string. Previously, such "fully-equipped" UC commitment schemes are only known in [7, 8], with an unavoidable overhead of $O(\kappa)$; meaning that to commit λ bits, the communication and computational costs are $O(\lambda\kappa)$, where κ denotes the security parameter. Efficient construction of a fully-equipped UC commitment scheme was a long-standing open problem. We introduce a new cryptographic primitive, called all-but-many encryptions (ABMEs), and prove that it is a translation of fully-equipped UC commitment in the algorithmic level. We implement ABMEs from two primitives, called probabilistic pseudo random functions and extractable sigma protocols, where the former is a probabilistic version of pseudo random function and the latter is a special kind of sigma (i.e., canonical 3-round public-coin HVSZK) protocols with some extractability. We provide efficient fully-equipped UC commitment schemes from ABMEs under DDH and DCR-based assumptions, respectively. The former is 3 times faster than arguably the most efficient adaptively secure UC commitment scheme [25] under the DDH assumption (that requires 5-round interaction and the secure-erasure assumption) in the reasonable security parameters. The latter is the first fully-equipped UC commitment scheme with *optimal expansion factor* $O(1)$. We also construct a fully-equipped UC commitment scheme from a general assumption (that trap-door permutations exist), which is far more efficient than the previous construction [8], because our construction does not require non-interactive zero-knowledge proof systems.

1 Introduction

Universal composability (UC) framework [6] guarantees that if a protocol is being proven secure in the UC framework, it remains secure even if it is run concurrently with arbitrary (even insecure) protocols. This composable property gives a designer a fundamental benefit, compared to the classic definitions, which only guarantee that a protocol is secure if it is run in the stand-alone setting. In this work, we focus on universally composable (UC) commitment schemes. As in the classic setting, UC commitments are an essential building block to consturct high level UC-secure protocols. UC commitments imply UC zero-knowledge protocols [7, 12], which play an essential role to construct UC-secure two-party and multi-party computations [8]. Unfortunately, it is known that UC commitments cannot be realized without an additional set-up assumption [7]. The common reference string (CRS) model is most widely used as a set-up assumption when considering the UC framework. So, we also consider schemes in the common reference model.

A commitment scheme is a two-phase protocol between two parties, a committer and a receiver. The basic idea behind the notion of commitment is as follows: In the first phase (or the commitment phase), a committer gives a receiver the digital equivalent of a *sealed envelope* containing value x, and, in the second phase (or the opening phase), the committer reveals x in a way that the receiver can verify it.

From the original concept, it is required that a committer cannot change the value inside the envelope (the binding property), whereas the receiver can learn nothing about x (the hiding property) unless the committer does not help the receiver opens the envelope [1].

Informally, a UC commitment scheme maintains the above binding and hiding properties under any concurrent composition with arbitrary protocols. To achieve this, a UC commitment scheme requires *equivocability* and *extractability*. Roughly, equivocability of a UC commitment scheme in the CRS model can be interpreted as follows: An algorithm (called the simulator) that takes the secret behind the CRS string can generate an *equivocable* commitment that can be opened correctly to any value. On the other hand, extractability can be interpreted as follows: The simulator given the secret can correctly extract the contents of any new *valid* commitment generated by any adversarial algorithm without given the secret, even after it has given the adversary many equivocable commitments, where a commitment is said valid if it can be opened correctly.

Several factors feature UC commitments such as non-interactivity, CRS re-usability, adaptive security, and non-erasure.

Interaction. If an execution of a UC commitment scheme is completed simply by sending each one message from the committer to the receiver in the commitment and opening phases, then it is called *non-interactive*; Otherwise, interactive.

CRS Reusability. The CRS model assumes that CRS strings are generated in a trusted way and given to every party. From the practical point of view, an important question in the CRS model is whether a single CRS string can be fixed beforehand and it can be *re-usable* in unbounded times of executions of cryptographic protocols. Otherwise, a new CRS string must be set up in a trusted way at some point when a new execution of a protocol is invoked.

Adaptive Security. A typical question asked about UC-secure protocols is whether security of the protocols is proven under static or adaptive adversaries. A static adversary can decide to corrupt parities only before the protocols start, whereas an adaptive adversary can decide to corrupt the parties at any point. If a protocol is proven UC-secure against adaptive adversaries, it is called *adaptive* UC-secure.

Non-Erasure. When a party is corrupted, the whole inner state of the party is revealed, including the randomness being used. Some protocols are only proven UC-secure under the assumption that the parties can securely erase their inner state. If such an assumption is unnecessary, the protocol is called *non-erasure*.

Canetti and Fischlin [7] presented the first UC secure commitment scheme in the common reference string model, which is "fully-equipped" – non-interactive, adaptively secure, and non-erasure under a single re-usable common reference string. In [7], two independent public-keys, pk_1 and pk_2, of an IND-CCA secure public-key encryption scheme and a claw-free trap-door permutation pair, (f_0, f_1), are put in the common reference string. The committer sends the receiver a commitment $(e, \mathbf{E}^{\mathsf{cca}}_{pk_0}(x_0), \mathbf{E}^{\mathsf{cca}}_{pk_1}(x_1))$, where $\mathbf{E}^{\mathsf{cca}}$ denotes the IND-CCA secure public-key encryption algorithm. The committed secret is one bit b such that $e = f_b(x_b)$. An honest committer generates a commitment to bit b by picking up random x_b to compute $e = f_b(x_b)$ and $\mathbf{E}_{pk_b}(x_b)$, and sampling E_{1-b} from the image of $\mathbf{E}^{\mathsf{cca}}_{pk_{1-b}}$ [2]. When

[1]There are two different favors in hiding and binding, statistical and computational ones. In the statistically-binding commitment schemes, the binding property holds against unbounded adversaries, whereas in the statistically-hiding commitment schemes, the hiding property holds against unbounded adversaries. By construction, a commitment scheme in the plain model satisfies at most either statistically-binding or statistically-hiding, not both.

[2]The public-key encryption scheme is assumed to be obliviously samplable [7].

he open the commitment or is corrupted by the adversary, the committer reveals x_b and E_{1-b}, along with the randomness behind $\mathbf{E}^{\mathsf{cca}}_{pk_b}(x_b)$. To be equivocable, the simulator computes both x_0, x_1 such that $e = f_0(x_0) = f_1(x_1)$ using the trap-door secret of the claw-free pair, and encrypts both of them. On the contrary, extractability is guaranteed so that the adversary cannot generate x_0, x_1 such that $e = f_0(x_0) = f_1(x_1)$, hence, the simulator can extract bit b by decrypting both ciphertexts and finding b such that $e = f_b(x_b)$. By construction, it is clear that the above scheme requires $O(\kappa)$ overhead, meaning that, to commit to λ-bit secret, it requires $O(\lambda\kappa)$ bits.

Canetti et al. [8] proposed another fully-equipped UC commitment scheme only from trap-door permutation. However, it is constructed in the same framework as in [7] and hence, expansion factor $O(\kappa)$ is unavoidable.

Damgård and Nielsen [12] presented UC commitment schemes with expansion factor $O(1)$. However, their schemes require 3-move interaction between the committer and the verifier in the commitment phase. In addition, their schemes require a common reference string that grows linearly with the number of parties. Damgård and Groth [10, 20] later removed the long CRS problem by using so-called simulation-sound commitments [17, 26, 18]. However, the simulation-sound commitment requires one-time signatures. Therefore, the secure-erasure assumption might be possibly necessary in the adaptive corruption. Nevertheless, it still requires 3-move interaction in the commitment phase.

Recently, Lindell [25] proposed an efficient adaptively secure UC commitment scheme with expansion factor $O(1)$, which is arguably the fastest scheme (by counting the computational time only, not including time loss of interaction), when it is implemented on an appropriate elliptic curve cryptosystem [3]. The scheme requires 5-move interaction in the commitment phase and assumes that the committer may securely erase his inner state. Very informally and loosely, the scheme is constructed as follows: In the commitment phase, the committer first encrypts his secret x by using a CCA seucre public key encryption (where the public-key is put in the CRS). Then the committer and receiver run a kind of non-malleable zero-knowledge protocol (using dual-system encryption schemes [30]), but abort just before the committer finally sends the receiver the last message proving that he knows the secrets committed to in the encryption. The committer then erases his random coins of the encryption. In the decommit phase, the committer reveals the final message in the above protocol. In this scheme, the committer does not reveal the *real* witness of the UC commitment, while it only reveals a zero-knowledge proof that enables the simulator to generate a simulated proof on any message. Therefore, the secure-erasure assumption is crucial for this scheme.

Fischlin, Libert, and Manulis [14] replaced the interactive part with a non-interactive one, by using the GS-proof techniques [21]. By construction, however, the secure-erasure assumption is still unavoidable.

Nishimaki, Fujisaki, and Tanaka [28] presented non-interactive, adaptively secure and non-erasure UC commitment schemes. However, their schemes are simply one-time secure– they consume a new CRS in each execution of commitment and hence, do not have CRS re-usability.

1.1 Our Contributions

We propose a general framework for constructing "fully-equipped" UC commitment schemes as mentioned above. The essential component in the framework is a new cryptographic primitive that we call all-but-many encryption (ABME), which is a translation of fully-equipped UC commitment in the algorithmic level. We construct ABMEs from a unified view of combining two cryptographic primitives, called probabilistic pseudo random functions and extractable sigma protocols, where the former is a probabilistic version of pseudo random functions and the latter is a special kind of sigma (i.e., canonical 3-round public-coin HVZK) protocols with some extractability.

[3]Although the current scheme has a flow, it is claimed that it can be fixed [25].

We propose a fully-equipped UC commitment scheme only from the Decisional Diffie-Hellman (DDH) assumption. From a practitioners' point of view, it is 3 times faster than arguably the previous most efficient adaptively secure UC commitment scheme [25] in the reasonable security parameters. See Table 1.

We also provide a fully-equipped UC commitment scheme with constant expansion factor $O(1)$; meaning that to commit $O(\kappa)$ bits, the communication and computational costs are $O(\kappa)$, where κ denotes the security parameter. This is the first fully-equipped UC commitment scheme with the optimal expansion factor of communication and computation. Only the CRS size is not optimal, $O(\kappa^2)$, which remains open. To prove security of this scheme, we assume that Damgård-Jurik homomorphic encryption scheme is *not multiplicatively hommorphic*, which is similar to the assumption used in [22].

We also present a weak version of ABME, which can be constructed from a general assumption (that trap-door permutations exist). We do not show that every weak ABME can be converted to a fully-equipped UC commitment scheme, but prove that at least our concrete construction from the general assumption is successfully converted so. Since it does not require non-interactive zero-knowledge proof systems, it is far more efficient than the previous scheme [8]. See Table 2. This construction is given in Appendix D.

1.1.1 Basic Idea

UC commitment schemes require equivocability and extractability. Therefore, a public key encryption scheme with the following properties is very useful: For a person who does not know the secret key, it looks a standard public key encryption scheme – If he encrypts a message under a public key properly, the corresponding secret key holder can decrypt the valid ciphertext correctly. However, the secret-key holder can generate a *fake* ciphertext under the public-key, which can be opened to any message along with the consistent randomness. It should be difficult for a user who does not own the secret key to distinguish a fake ciphertext from a real ciphertext even after the message and the randomness used there are revealed. We also require that the encryption scheme is tag-based to fit the UC framework and that the secret key holder can produce fake ciphertexts a-prior unbounded polynomially many times, but nobody without given the secret key can produce a fake ciphertext on a fresh tag even after he has seen an a-prior unbounded polynomially many number of fake ciphertexts. We call such encryptions all-but-many encryptions (ABMEs).

To construct all-but-many encryptions, as the first step idea, we call instance-dependent commitments [1, 23] to mind. An instance-dependent commitment scheme is an "instance-based" commitment scheme that additionally takes x as input to commit to a message and behaves differently depending on whether instance x belongs to NP language L or not: When $x \in L$, a honest committer always generates statistically-hiding commitments, whereas when $x \notin L$, he always generates statistically-binding commitments.

A non-interactive instance-dependent commitment scheme can be constructed if there exists a canonical three-move public-coin statistically zero-knowledge protocol, called the sigma protocol [9] [4], for an NP language L and if the decision problem on L is hard: Let (a, e, z) be the transcript of the sigma protocol on instance x. Let w be the witness of x (if it exists). When a honest committer wants to commit to e, he runs the *simulation* algorithm of the sigma protocol on x with challenge e (regardless of whether $x \in L$ or not) and sends the receiver the first message a. To open the commitment, the committer reveals (e, z). The receiver accepts it if (a, e, z) is an accepted conversation on x in the sigma protocol. By (special) honest verifier statistical zero-knowledgeness, for every $x \in L$ and every e, the transcript on (x, e), i.e., (a, e, z), generated by the simulation algorithm of the sigma protocol is statistically indistinguishable from the transcript on the same (x, e) generated by the real sigma protocol using witness w. This implies

[4] Precisely speaking, we require a slightly stronger variant of sigma protocols as described in Sec. 4.

that when $x \in L$, a honest committer generates statistically hiding commitments. The computational binding holds because it is difficult to find w from x. (Opening a commitment in two ways reveals w due to special soundness.) On the contrary, when $x \notin L$, the first message a, generated by any (possibly dishonest) committer, is statistically binding to e, as long as there exists an accepted conversation for a. This immediately follows from special soundness of sigma protocols. The (computational) hiding property holds because it is hard to decide whether $x \in L$ or not. Therefore, when $x \notin L$, a committer generates statistically-binding commitments.

When $x \in L$, it is obvious that we can construct a simulator that generates equivocable commitments that are statistically indistinguishable from commitments generated by a honest committer. The simulator runs the real sigma protocol with witness w and outputs the first message a in the commitment phase. Since the real sigma protocol can produce answer z for any challenge e, using witness w along with the randomness used when generating a, the simulator can open a into any value e in the opening phase. Therefore, this instance-dependent commitment scheme is equivocable when $x \in L$. We note that to commit to e, a honest committer runs *the simulation algorithm* of the sigma protocol on (x, e), whereas the simulator runs *the real sigma protocol* on x with w.

On the contrary, can we extract e from a when $x \notin L$ (without randomness behind a)? For special languages, it is possible. Fujisaki [16] has recently introduced a special kind of sigma protocol, in which, letting L_{pk} be an NP language indexed by (a series of) pk, the simulator can efficiently check that "a commits to e", for given (x, a, e), where $x \notin L_{pk}$, by using secret key sk behind pk (but no randomness under a is required). Such a sigma protocol is called a weak extractable sigma protocol [16]. In this paper, we require a strong variant, in which, given (x, a), the simulator can extract e by using sk. We call this variant the extractable sigma protocols. Therefore, if there is an extractable sigma protocol for L_{pk}, we have a non-interactive instance-dependent commitment scheme, which is equivocable when $x \in L_{pk}$ and extractable when $x \notin L_{pk}$.

We now want that only the simulator is able to choose $x \in L_{pk}$ while the adversary is forced to choose $x \notin L_{pk}$ even after it has seen many different $\tilde{x}$'s that belong to L_{pk}. For this purpose, we consider "tag-based" NP language $L_{pk} = \{x = (t, u) \mid t \in \{0,1\}^{\kappa} \text{ and } u \in L_{pk}(t)\}$, with the following properties:

- (Pseudo-randomness) There is set U such that $L_{pk}(t) \subset U$ for any t and it is easy to randomly sample from U. It is infeasible to decide whether one may have access to oracle $L_{pk}(\cdot)$ or $U(\cdot)$ in an unbounded polynomially many times, where oracle $L_{pk}(t)$ returns random u from $L_{pk}(t)$ and oracle $U(t)$ returns random $u \in U$ [5].
- (Unforgeability) It is infeasible to produce $x^* = (t^*, u^*)$ in L_{pk} on fresh tag t^* even if one may have access to oracle $L_{pk}(\cdot)$ in an unbounded polynomially many times.

In fact, such a language can be constructed via a pseudo random function family and a public encryption scheme. For instance,

$$L_{pk} = \{x = (t, u) \mid \exists sk = (s, r) : \ u = F_s(t) \text{ and } c = \mathbf{E}_{pk'}(s; r)\}, \text{ where } pk = (pk', c).$$

If there exists an extractable sigma protocol for such L_{pk}, it can be converted to an all-but-many (ABM) encryption scheme as follows: To encrypt e on tag t, a honest encryptor (without knowing sk) chooses random u from U, generates commitment a on $x = (t, u)$ by using the simulation algorithm of the extractable sigma protocol, and finally outputs (x, a). With an overwhelming probability, it holds that $x \notin L_{pk}$. Therefore, the simulator given sk can extract e from a correctly. On the contrary, the simulator (the secret-key holder) can generate $x = (t, u) \in L_{pk}$ and produce a by using the real sigma

[5] If $L_{pk}(\cdot)$ is deterministic, then U returns the same u on t if it was previously queried.

protocol with witness w of x [6], which is a fake ciphertext that can be opened correctly to any e along with consistent z. By pseudo-randomness of L_{pk} and zero-knowledgeness of the sigma protocol, nobody without given the secret-key sk can distinguish a real ciphertext from a fake ciphertext, even if message e and randomness z are revealed. In addition, from unforgeability of L_{pk} and special soundness of the sigma protocol, no dishonest encryptor can produce a fake ciphertext on a fresh t^*, even after he saw an a-prior unbounded polynomially many number of fake encryptions on t, with $t \neq t^*$.

Finally, we simply see the (real) ciphertext generated by an ABM encryption scheme as a UC commitment, by putting the public key in the common reference string beforehand. To open the commitment, the message and randomness used to be encrypted are revealed.

1.2 ABM Lossy Trap-door Functions

Hofheinz has recently proposed all-but-many lossy trap-door functions (ABM-LTDFs) [22], which are lossy trap-door (deterministic) functions with (unbounded) *many* lossy tags. He has proposed two schemes based on the DCR-based and q-strong DH assumptions, respectively. Our idea of viewing signatures equipped with no public verification procedure (namely, the probabilistic pseudo random functions) as equivocable tags is inspired by the idea of seeing encrypted signatures as lossy tags in [22]. Apparently, both schemes seem quite different – ABM-LTDFs are deterministic whereas ABMEs are probabilistic and, by construction, essentially require randomness. However, they are related in some sense: In [22], a tag is a matrix M such that its determinant is an encryption of zero if a valid signature is embed. If $\det(M) = \mathsf{Enc}_{pk}(0)$, it is lossy; otherwise, it is injective. On the other hand, our construction is obtained by simulating the first message of sigma protocols (with some extractability) for the languages of signatures equipped with no public verification procedure, where the first message implies linear equations when using a typical proof of knowledge protocols based on homomorphic functions. The determinant derived from the linear equations turns out zero if and only if tags correspond signatures.

2 Preliminaries

Let $\mathbb{N}$ be the set of natural numbers. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. We denote by O, Ω, and ω the standard notations to classify the growth of functions. We let $\mathrm{poly}(\kappa)$ denote an unspecified function $f(\kappa)= O(\kappa^c)$ for some constant c. We let $\mathsf{negl}(\kappa)$ to denote an unspecified function $f(\kappa)$ such that $f(\kappa) = \kappa^{-\omega(1)}$, saying that such a function is negligible in κ. We write PPT and DPT algorithms to denote probabilistic polynomial-time and deterministic poly-time algorithms, respectively. For PPT algorithm A, we write $y \leftarrow A(x)$ to denote the experiment of running A for given x, picking inner coins r uniformly from an appropriate domain, and assigning the result of this experiment to the variable y, i.e., $y = A(x; r)$. Let $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$ be probability ensembles such that each X_κ and Y_κ are random variables ranging over $\{0, 1\}^\kappa$. The (statistical) distance between X_κ and Y_κ is $\mathsf{Dist}(X_\kappa, Y_\kappa) \triangleq \frac{1}{2} \cdot |\Pr_{s\in\{0,1\}^\kappa}[X = s] - \Pr_{s\in\{0,1\}^\kappa}[Y = s]|$. We say that two probability ensembles, X and Y, are statistically indistinguishable (in κ), denoted $X \overset{\mathrm{s}}{\approx} Y$, if $\mathsf{Dist}(X_\kappa, Y_\kappa) = \mathsf{negl}(\kappa)$. In particular, we denote by $X \equiv Y$ to say that X and Y are identical. We say that X and Y are computationally indistinguishable (in κ), denoted $X \overset{\mathrm{c}}{\approx} Y$, if for every non-uniform PPT D (ranging over $\{0, 1\}$), $\{D(1^\kappa, X_\kappa)\}_{\kappa\in\mathbb{N}} \overset{\mathrm{s}}{\approx} \{D(1^\kappa, Y_\kappa)\}_{\kappa\in\mathbb{N}}$. Let $R = \{(X, W)\}$ be an NP relation, meaning that given (X, W), it can be decided in a polynomial-time in $|X|$ if $(X, W) \in R$. Here X is called a statement and W is called a witness of X. Let us denote by L_R the NP language characterized by R, meaning that $L_R = \{X \mid \exists W : (X, W) \in R\}$.

[6] In the above case, $w = sk$ for any $x \in L_{pk}$.

2.1 The Universal Composability Framework

We work in the standard universal composability (UC) framework of Canetti [6]. We concentrate on the same model in [7] where the network is asynchronous, the communication is public but ideally authenticated, and the adversary is adaptive in corrupting parties and is active in its control over corrupted parties. Any number of parties can be corrupted and parties cannot erase any of their inner state. We provide a brief description of the UC framework and the ideal commitment functionality for multiple commitments in Appendix A.

3 Probabilistic Pseudo Random Functions

A probabilistic pseudo random function Spl is a probabilistic version of pseudo random function mapping from domain $\{0,1\}^\kappa$ to codomain U parameterized by public key pk. It takes message t and outputs u ($= \mathsf{Spl}(sk, t; v)$) under secret key sk with respects to pk. Informally, the requirement of PPRFs is that (a) u looks at least pseudo-random on any t and (b) it is infeasible for any adversary to compute valid u^* on fresh t^* even after it may have access to oracle $\mathsf{Spl}(sk, \cdot)$, where t^* is called fresh if it has not been queried. Now we formally define PPRFs. A PPRF $(\mathsf{Gen}_{\mathsf{spl}}, \mathsf{Spl})$ consists of the following two algorithms:

- $\mathsf{Gen}_{\mathsf{spl}}$ is a PPT algorithm that takes 1^κ as input and outputs (pk, sk). Here pk uniquely determines a set U, the codomain of Spl. For convenience sake, we assume that the description of pk contains κ and the description of sk contains that of pk. W.l.o.g., we assume $\mathsf{Gen}_{\mathsf{spl}}$ is an NP relation [7].

- Spl is a PPT algorithm that takes sk and $t \in \{0,1\}^\kappa$, picks up inner random coins $v \leftarrow \mathsf{COIN}_{\mathsf{spl}}$, and computes $u \in U$, namely $u = \mathsf{Spl}(sk, t; v)$. $\mathsf{COIN}_{\mathsf{spl}}$ denotes the inner coin space uniquely determined by pk.

For our convenience, we define

$$L_{pk} = \{(t, u) \mid \exists\, sk,\ \exists\, v \in \mathsf{COIN}_{\mathsf{spl}} : (pk, sk) \in \mathsf{Gen}_{\mathsf{spl}}(1^\kappa) \text{ and } u = \mathsf{Spl}(sk, t; v)\}.$$

We require that PPRFs satisfy the following security requirements:

- **Easy sampling:** For every pk given by $\mathsf{Gen}_{\mathsf{spl}}$, it is easy to sample random elements from U.

- **Pseudo randomness:** For every non-uniform PPT adversary A, the advantage of A in the following distinguishing game is negligible in κ: $(pk, sk) \leftarrow \mathsf{Gen}_{\mathsf{spl}}(1^\kappa)$; A takes pk; A may submit an a-prior unbounded polynomially many number of arbitrary messages in $\{0,1\}^\kappa$ to either of two oracles, $\mathsf{Spl}(sk, \cdot)$ or $U(\cdot)$, where U is the following oracle: When $\mathsf{Spl}(sk, \cdot)$ is a deterministic function, $U : \{0,1\}^\kappa \to U$ is a random oracle which returns the same value on the same input. When $\mathsf{Spl}(sk, \cdot)$ is probabilistic, then $U(\cdot)$ picks up random $u \leftarrow U$ every time for every query to return, even if it was already queried. A finally distinguishes which oracle it has had access to. The probability is taken over the inner coins of $\mathsf{Gen}_{\mathsf{spl}}$, Spl, A, and random sampling from U.

- **Unforgeability:** For every non-uniform PPT adversary A, the advantage of A in the following forging game is negligible in κ: A takes pk generated by $\mathsf{Gen}_{\mathsf{spl}}(1^\kappa)$; A may submit a series of arbitrary messages in $\{0,1\}^\kappa$ to oracle $\mathsf{Spl}(sk, \cdot)$; A finally outputs (t, u) such that $(t, u) \in L_{pk}$ and message t has not been queried to $\mathsf{Spl}(sk, \cdot)$. The probability is taken over the inner coins of $\mathsf{Gen}_{\mathsf{spl}}$, Spl, and A.

We remark that if $\mathsf{Spl}(sk, \cdot)$ is a deterministic algorithm and sk is uniquely determined by pk, the unforgeability requirement is implied by pseudo randomness and hence, can be removed from the requirements.

[7] Namely, given (pk, sk), one can easily check $(pk, sk) \in \mathsf{Gen}(1^\kappa)$.

3.1 Construction of PPRFs

A PPRF ($\mathsf{Gen_{spl}}, \mathsf{Spl}$) can be constructed in a straight-forward way from a pseudo random function family $\mathcal{F} = \{(F_i)_{i \in I_\kappa}\}_{\kappa \in \mathbb{N}}$ and a semantically secure (or IND-CPA) public-key encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ [19]: $\mathsf{Gen_{spl}}(1^\kappa)$ picks up $(pk, sk) \leftarrow \mathbf{K}(1^\kappa)$ and $i \leftarrow I_\kappa$ (an index of the pseudo-random function family w.r.t. security parameter κ). It outputs $PK = (pk, \mathbf{E}_{pk}(i; r))$ and $SK = (PK, i, r)$ where r is a random string uniformly chosen from the coin space of the encryption scheme. Then, define $\mathsf{Spl}(SK, t) := F_i(t)$. By construction, it is clear that pseudo-randomness holds. In addition, if there is an adversary that breaks unforgeability, it should break pseudo randomnes of $\mathcal{F}$ or semantic security of Π.

We also propose probabilistic schemes. The idea behind our constructions is to use Waters signature [31] as a PPRF in a group *equipped with no bilinear map*. Let g be a generator of a multiplicative group G of prime order q, on which the DDH assumption holds. For $\kappa + 1$ elements in G, let us define $H(t)= h_0 \prod_{i=1}^{\kappa} h^{t_i}$, where $t = (t[1], \ldots, t[\kappa]) \in \{0,1\}^\kappa$ in which $t[i] \in \{0,1\}$ denotes i-th bit representation of string t. $\mathsf{Gen_{spl}}(1^\kappa)$ chooses $g, h_0, \ldots, h_\kappa \leftarrow G$ and $x_1, x_2 \leftarrow \mathbb{Z}/q\mathbb{Z}$ to set $g_1 = g^{x_1}$, $g_2 = g^{x_2}$. outputs $pk = (G, g, q, \lambda, g_1, g_2, h_0, \ldots, h_\kappa)$. and $sk = (pk, x_2)$, where $U = G \times G$. $\mathsf{Spl}(sk, t; r)$ takes $t \in \{0,1\}^\kappa$, picks up random $r \leftarrow \mathbb{Z}/q\mathbb{Z}$, and computes $u_r = g^r$ and $u_t = g_1^{x_2}(H(t))^r$. It then outputs $u = (u_r, u_t)$.

Theorem 3.1 *The above construction is a PPRF under the DDH assumption.*

Proof. Spl is the same as Waters signature scheme when applied for a non-pairing group. So, unforgeability is immediately guaranteed if the computational DH assumption holds true. Pseudo-randomness is shown in a straightforward way: Suppose that (g, g_1, g_2, K) be a tuple of four group elements in G, which is either a DDH instance ($K = g_1^{x_2}$) or a random tuple (K is a random element in G). To break the DDH problem, a simulator picks up $\vec{h} = (h_0, h_1, \ldots, h_\kappa)$ at random. It then runs adversary A on the above parameters, where A is an adversary to break pseudo-randomness. For any query t, the simulator returns (u_r, u_t) such that $u_r = g^r$ and $u_t = K \cdot H(t)^r$. The simulator outputs the same bit that A outputs. The simulator's advantage is the same as that of A. Therefore, under DDH assumption its advantage is bounded in a negligible (in κ) function. Therefore, it also satisfies pseudo-randomness. Hence, the scheme above is an instantiation of PPRFs if the DDH assumption holds true. ■

We further present another variant of PPRFs based on Waters signature, which can be constructed from *additively* homomorphic IND-CPA public-key encryption schemes. We show the construction in appendix C.

4 Extractable Sigma-Protocol

We introduce extractable sigma protocols. We note that in [16] we have introduced a similar primitive. In this paper we require a slightly stronger variant.

First, we recall a sigma protocol [9]. Let $R = \{(X, W)\}$ be an NP relation. Let L_R be the NP language characterized by $R = \{(X, W)\}$, namely, $L_R = \{X \mid \exists W : (X, W) \in R\}$. A sigma protocol for NP relation R, $\Sigma = (\mathsf{com}\Sigma, \mathsf{ch}\Sigma, \mathsf{ans}\Sigma, \mathsf{sim}\Sigma, \mathsf{Vrfy})$, is a canonical 3-round (public coin) interactive proof system between the prover and the verifier. Let $X \in L$ be a statement to be proven and W denotes a witness of X such that $(X, W) \in R$. X is given to both the prover and the verifier as common input and W is given only to the prover in advance. A Σ-protocol on common input X is executed as follows: The prover picks up random coins r_a, computes a using statement X and witness W, denoted $a = \mathsf{com}\Sigma(X, W; r_a)$, and sends it to the verifier. The verifier picks up a random $\mathsf{challenge}$ element $e \leftarrow \mathsf{ch}\Sigma$, where $\mathsf{ch}\Sigma$ is a publicly-samplable prescribed set, and sends it to the prover. The prover responds with $z = \mathsf{ans}\Sigma(X, W, r_a, e)$. The verifier accepts if $\mathsf{Vrfy}(X, a, e, z) = 1$. We say that (a, e, z) is an accepting conversation on X if $\mathsf{Vrfy}(X, a, e, z) = 1$. We require that the sigma protocols satisfy the following properties:

- **Completeness:** For every r_a (in an appropriate specified domain) and every $e \in \mathsf{ch}\Sigma$, it always holds that Vrfy $(X$, $\mathsf{com}\Sigma(X, W; r_a)$, e, $\mathsf{ans}\Sigma(X, W, r_a, e)) = 1$.

- **Special Soundness:** For every $X \notin L_R$ and every a, there is the the unique e in $\mathsf{ch}\Sigma$ if there is an accepted conversation for a; that is, there is z such that $\mathsf{Vrfy}(X, a, e, z) = 1$. In addition, one can always efficiently compute witness W, given X and two different accepted conversations for a on X, (a, e, z) and (a, e', z'), with $c \neq c'$. A pair of accepted two different conversations for the same a on X, i.e., (a, e, z) and (a, e', z'), with $e \neq e'$, is called a collision on X. We insist that a collision on X exists if and only if $X \in L_R$.

- **Enhanced Special Honest-Verifier Statistical Zero-Knowledge:** $\mathsf{sim}\Sigma$ is a PPT algorithm that takes X and $e \in \mathsf{ch}\Sigma$ as input and, picking up $r_z \leftarrow \mathsf{COIN}_{\mathsf{sim}}$, outputs $(a, e, z) = \mathsf{sim}\Sigma(X, e; r_z)$. Given every $(X, W) \in R$ and every $e \in \mathsf{ch}\Sigma$,

$$\{\mathsf{sim}\Sigma(X, e; r_z)\} \overset{\mathrm{s}}{\approx} \{(\mathsf{com}\Sigma(X, W; r_a), e, \mathsf{ans}\Sigma(X, W, r_a, e))\},$$

  where the probability of the left hand is taken over random variable r_z and the right hand is taken over random variable r_a. In this paper, we require slightly more for our sigma protocol. We say that Σ is enhanced special HVSZK if $r_z = z$. Namely, $(a, e, z) = \mathsf{sim}\Sigma(X, e; z)$. Then, we note that $\mathsf{Vrfy}(X, a, e, z) = 1$ if and only if $(a, e, z) = \mathsf{sim}\Sigma(X, e; z)$, which means that one can instead use $\mathsf{sim}\Sigma$ to verify.

4.1 Extractable Sigma Protocol

We now introduce extractable sigma protocols. Let $\mathsf{Gen}_{\mathsf{ext}} = \{(pk, sk)\}$ be an NP relation. We denote by $R_{pk} = \{(X, (sk, W))\}$ an NP relation indexed by pk [8] such that if $(X, (sk, W)) \in R_{pk}$, then $(pk, sk) \in \mathsf{Gen}_{\mathsf{ext}}$. Let us denote by L_{pk} the NP languages characterized by R_{pk}, i.e., $L_{pk} = \{X \mid \exists(sk, W) : (X, (sk, W)) \in R_{pk}\}$.

A extractable sigma-protocol $\mathsf{ext}\Sigma = (\Sigma, \mathsf{Dec})$ for NP relation R_{pk} w.r.t. $\mathsf{Gen}_{\mathsf{ext}}$ consists of the following algorithms:

- $\Sigma(pk) = (\mathsf{com}\Sigma, \mathsf{ch}\Sigma, \mathsf{ans}\Sigma, \mathsf{sim}\Sigma)$ is a sigma protocol for R_{pk} (for every sequence of $\{pk\}_{\kappa\in\mathbb{N}}$) with the enhanced special honest-verifier statistical zero-knowledge mentioned above. We remove Vrfy from Σ, because we can instead use $\mathsf{sim}\Sigma$ for verification.

- Dec, the extract algorithm, is a DPT algorithm that takes sk, X, and a (presumably the first output generated by $\mathsf{sim}\Sigma(pk)(X, e)$) and outputs e or $\perp$.

We require that $\mathsf{ext}\Sigma$-protocols additionally satisfy the following property:

(Extractability) For every $(pk, sk) \in \mathsf{Gen}_{\mathsf{ext}}$, every $X \notin L_{pk}$, every $e \in \mathsf{ch}\Sigma(pk)$, and every a such that there is an accepted conversation (a, e, z) for a on X, it always hold that $\mathsf{Dec}(sk, X, a) = e$.

Here, we note that if there is an accepted conversation (a, e, z) on $X \notin L_R$, e is unique for a, due to the special soundness of the sigma protocols. Therefore, extractability is well defined, because e is uniquely determined by a when $X \notin L_{pk}$. Extractability implies that even if a is generated **in an adversarial way**, there is a unique e consistent with a and it can be extracted from a using sk, as long as $X \notin L_{pk}$ and a has an accepted conversation on X.

[8] Precisely speaking, we consider R_{pk} as an ensemble indexed by a sequence of public keys, $\{pk\}_{\kappa\in\mathbb{N}}$, where there is one pk for every $\kappa \in \mathbb{N}$.

5 ABM Encryptions

All-but-many encryption scheme $\mathsf{ABM.Enc} = (\mathsf{ABM.gen}, \mathsf{ABM.spl}, \mathsf{ABM.enc}, \mathsf{ABM.dec}, \mathsf{ABM.col})$ consists of the following algorithms:

- $\mathsf{ABM.gen}$ is a PPT algorithm on input 1^κ outputs (pk, sk), where pk defines an efficiently samplable set U, the codomain of $\mathsf{ABM.spl}$. We let $S = \{0,1\}^\kappa \times U$. For convenience' sake, we assume that the description of pk contains κ and the description of sk contains that of pk. W.l.o.g., we assume $\mathsf{ABM.gen}$ is an NP relation; that is, given (pk, sk), one can easily check $(pk, sk) \in \mathsf{Gen}(1^\kappa)$.

- $\mathsf{ABM.spl}$ is a PPT algorithm that takes sk and tag $t \in \{0,1\}^\kappa$, picks up inner random coins $v \leftarrow \mathsf{COIN_{spl}}$, and computes $u \in U$. $\mathsf{COIN_{spl}}$ denotes the inner coin space uniquely determined by pk. We define

$$L_{pk}(t) = \{u \in U \mid \exists\, sk,\ \exists\, v \in \mathsf{COIN_{spl}} : (pk, sk) \in \mathsf{ABM.gen}(1^\kappa) \text{ and } u = \mathsf{ABM.spl}(sk, t; v)\}.$$

  We also define $L_{pk} = \{(t, u) \mid t \in \{0,1\}^\kappa \text{ and } u \in L_{pk}(t)\}$.

- $\mathsf{ABM.enc}$ is a PPT algorithm that takes pk, $(t, u) \in S$, and message $x \in \mathsf{MSP}$, picks up inner random coins $r \leftarrow \mathsf{COIN_{enc}}$, and computes c such that $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$, where MSP denotes the message space uniquely determined by pk, whereas $\mathsf{COIN_{enc}}$ denotes the inner coin space uniquely determined by pk *and* x [9].

- $\mathsf{ABM.dec}$ is a DPT algorithm that takes sk, $(t, u) \in S$, and ciphertext c, and computes $x = \mathsf{ABM.dec}^{(t,u)}(sk, c)$.

- $\mathsf{ABM.col}$= $(\mathsf{ABM.col}_1, \mathsf{ABM.col}_2)$ is a pair of PPT and DPT algorithms, respectively, such that
  - $\mathsf{ABM.col}_1$ takes sk, (t, u), and $v \in \mathsf{COIN_{spl}}$ such that $t \in \{0,1\}^\kappa$ and $u = \mathsf{ABM.spl}(sk, t; v)$, and outputs $(c, \xi) \leftarrow \mathsf{ABM.col}_1^{(t,u)}(sk, v)$
  - $\mathsf{ABM.col}_2$ takes ξ and $x \in \mathsf{MSP}$, and outputs $r \in \mathsf{COIN_{enc}}$ such that $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$.

We require that all-but-many encryption schemes satisfy the following properties:

1. **Adaptive All-but-many property:** $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$ is a probabilistic pseudo random function (PPRF). We note that for every pk, $|L_{pk}| = O(\kappa)$ and $\frac{|L_{pk}|}{|S|} = \mathsf{negl}(\kappa)$.

2. **Dual mode property:** For every $\kappa \in \mathbb{N}$ and every $(pk, sk) \in \mathsf{ABM.gen}(1^\kappa)$,

   - **(Decryption mode)** For every $(t, u) \in S \backslash L_{pk}$, and every $x \in \mathsf{MSP}$, it always holds that

   $$\mathsf{ABM.dec}^{(t,u)}(sk, \mathsf{ABM.enc}^{(t,u)}(pk, x)) = x.$$

   - **(Trap-door mode)** For every $(t, u) \in L_{pk}$, every $v \in \mathsf{COIN_{spl}}$ such that $u = \mathsf{ABM.spl}(sk, t; v)$, every $(c, \xi) \in \mathsf{ABM.col}_1^{(t,u)}(sk, v)$, and every $x \in \mathsf{MSP}$, it always holds that

   $$c = \mathsf{ABM.enc}^{(t,u)}(pk, x; \mathsf{ABM.col}_2(\xi, x)).$$

[9] We allow the inner coin space to depend on messages to be encrypted, because our concrete construction of weak ABM encryption from general assumption in Sec. D requires the coin space to depend on messages.

In addition,

$$\left\{ \left(\mathsf{ABM.col}_1^{(t,u)}(sk,v)[1], \quad \mathsf{ABM.col}_2 \left(\mathsf{ABM.col}_1^{(t,u)}(sk,v)[2],\, x \right) \right) \right\} \overset{\mathrm{s}}{\approx} \left\{ \left(\mathsf{ABM.enc}^{(t,u)}(pk,x;r), \quad r \right) \right\}$$

for every $x \in \mathsf{MSP}$, every $(t,u) \in L_{pk}$, and every witness (sk,v) of $(t,u) \in L_{pk}$. Here $\mathsf{ABM.col}_1^{(t,u)}(sk,v)[1]$ denotes the first output of $\mathsf{ABM.col}_1^{(t,u)}(sk,v)$, and $\mathsf{ABM.col}_1^{(t,u)}(sk,v)[2]$ denotes the second output of $\mathsf{ABM.col}_1^{(t,u)}(sk,v)$. The probability of the light-hand side random variable is taken over the random choice of $r \in \mathsf{COIN}_{\mathsf{enc}}$.

We say that a ciphertext c on (t,u) $(\in S)$ under public key pk is valid if there exist $x \in \mathsf{MSP}$ and $r \in \mathsf{COIN}_{\mathsf{enc}}$ such that $c = \mathsf{ABM.enc}^{(t,u)}(pk,x;r)$. We say that a valid ciphertext c on (t,u) $(\in S)$ under public key pk is real if $(t,u) \in S \backslash L_{pk}$, otherwise fake if $(t,u) \in L_{pk}$.

We remark that as long as c is a real ciphertext, there is only one consistent x in MSP and it is equivalent to $\mathsf{ABM.dec}^{(t,u)}(sk,c)$, due to the correctness condition of the decryption mode. *This means that even if a ciphertext is generated by an adversary, it can be decrypted correctly as long as there exists a pair of a message and randomness consistent with the ciphertext and* $(t,u) \in S \backslash L_{pk}$.

6 Construction of ABME from Extractable Sigma Protocol for PPRF

Suppose there is an extractable sigma protocol such that it can prove the possession of witness behind the input and output relation of a pseudo-random function. Then, we can construct an all-but-many encryption scheme. Let $(\mathsf{Gen}_{\mathsf{spl}}, \mathsf{Spl})$ be a probabilistic pseudo random function (PPRF) defined above. Let us define $R_{pk} = \{((t,u),(sk,v)) \,|\, u = \mathsf{Spl}(sk,t;v)\}$, which is an NP relation indexed by (a sequence of) $\{pk\}_{\kappa \in \mathbb{N}}$. For an extractable sigma protocol $\mathsf{ext}\Sigma$ for R_{pk}, an ABM encryption scheme $\mathsf{ABM.Enc}$ is constructed as follows:

- $\mathsf{ABM.gen}(1^\kappa) = \mathsf{Gen}_{\mathsf{spl}}(1^\kappa)$. Let (pk,sk) be generated by $\mathsf{ABM.gen}$. Let U be the codomain of Spl determined by pk. Let $S = \{0,1\}^\kappa \times U$.
- $\mathsf{ABM.spl}(sk,t;v) = \mathsf{Spl}(sk,t;v)$, where $t \in \{0,1\}^\kappa$ and $v \in \mathsf{COIN}_{\mathsf{spl}}$.
- $\mathsf{ABM.enc}^{(t,u)}(pk,x;r) = \mathsf{sim}\Sigma(pk)(X,x;r)[1]$, where $X = (t,u) \in S$, $x \in \mathsf{MSP}$ $(= \mathsf{ch}\Sigma(pk))$, and $r \in \mathsf{COIN}_{\mathsf{enc}}$ $(= \mathsf{COIN}_{\mathsf{sim}})$.

  Here $\mathsf{sim}\Sigma(pk)(X,x;r)[1]$ denotes the first output of $\mathsf{sim}\Sigma(pk)(X,x;r)$.
- $\mathsf{ABM.dec}^{(t,u)}(sk,c) = \mathsf{Dec}(sk,X,c)$, where $X = (t,u)$, and $c = \mathsf{ABM.enc}^{(t,u)}(pk,x;r)$.
- $\mathsf{ABM.col}_1^{(t,u)}(sk,v;r_a) = (c,\xi)$, such that $c = \mathsf{com}\Sigma(pk)(X,W;r_a)$ and $\xi = (sk,t,u,v,r_a)$, where $X = (t,u)$, $W = (sk,v)$ and $u = \mathsf{Spl}(sk,t;v)$.
- $\mathsf{ABM.col}_2(\xi,x) = \mathsf{ans}\Sigma(pk)(X,W,r_a,x)$, where $\xi = (sk,t,u,v,r_a)$, $X = (t,u)$, $W = (sk,v)$, and $x \in \mathsf{MSP}$.

Here, $L_{pk} = \{(t,u) \,|\, \exists (sk,v) : (pk,sk) \in \mathsf{ABM.gen}(1^\kappa) \text{ and } u = \mathsf{Spl}(sk,t;v)\}$. By construction, it is obvious that $\mathsf{ABM.Enc}$ satisfies the adaptive all-but-many property. The dual mode property also holds because: (a) If $X = (t,u) \in S \backslash L_{pk}$, $a \in \mathsf{sim}\Sigma^1(pk)(X,x)$ is perfectly binding to x, due to special soundness and x is extracted from (X,a) only using sk, due to extractability of extractable sigma protocols. (b) If

$X = (t, u) \in L_{pk}$, $\mathsf{ABM.col}$ runs the real (extractable) sigma protocol $(\mathsf{com}\Sigma, \mathsf{ans}\Sigma)$ with witness (sk, v). Therefore, it can produce a fake commitment that can be opened in any way, while it is statistically indistinguishable from that of the simulation algorithm $\mathsf{sim}\Sigma$ (that is run by $\mathsf{ABM.enc}$), due to enhanced honest statistical zero-knowledgeness. Therefore, the resulting scheme is an all-but-many encryption scheme.

7 UC Commitments from ABM Encryptions

We see an ABM encryption as an UC commitment by putting the publickey message in the common reference string. To open the commitment, the commiter sends the message to be encrypted and the randomness used there. We formally describe our UC commitment scheme in Fig. 1.

UC-commitment protocol from $\mathsf{ABM.Enc}$
Common reference string: pk where $(pk, sk) \leftarrow \mathsf{ABM.gen}(1^\kappa)$.
We implicitly assume that there is injective map ι from $\{0,1\}^\kappa$ to MSP such that ι^{-1} is efficiently computable and $\iota^{-1}(y) = \varepsilon$ for every $y \notin \iota(\{0,1\}^\kappa)$, and also assume that $(sid, ssid, P_i, P_j) \in \{0,1\}^\kappa$.
The commit phase:

- Upon input $(\texttt{commit}, sid, ssid, P_i, P_j, x)$ where $x \in \{0,1\}^\kappa$, party P_i proceed as follows: If a tuple $(\texttt{commit}, sid, ssid, P_i, P_j, x)$ with the same $(sid, ssid)$ was previously recorded, P_i does nothing. Otherwise, P_i sets $t = (sid, ssid, P_i, P_j) \in \{0,1\}^\kappa$. It picks up $u \leftarrow U$ and $r \leftarrow \mathsf{COIN}$, and encrypts message $\iota(x)$ to compute $c = \mathsf{ABM.enc}^{(t,u)}(pk, \iota(x); r)$. P_i sends $((t, u), c)$ to party P_j, and stores $(sid, ssid, P_i, P_j, (t, u), x, r)$.
- P_j ignores the commitment if $t \neq (sid, ssid, P_i, P_j)$, $u \notin U$, or a tuple $(sid, ssid, \dots)$ with the same $(sid, ssid)$ was previously recorded. Otherwise, P_j stores $(sid, ssid, P_i, P_j, (t, u), c)$ and outputs $(\texttt{receipt}, sid, ssid, P_i, P_j)$.

The decommitment phase:

- Upon receiving input $(\texttt{open}, sid, ssid)$, P_i proceeds as follows: If a tuple $(sid, ssid, P_i, P_j, x, r)$ was previously recorded, then P_i sends $(sid, ssid, x, r)$ to P_j. Otherwise, P_i does nothing.
- Upon receiving input $(sid, ssid, x, r)$, P_j proceeds as follows: P_j outputs $(\texttt{open}, sid, ssid, P_i, P_j, x)$ if a tuple $(sid, ssid, P_i, P_j, (t, u), c)$ with the same $(sid, ssid, P_i, P_j)$ was previously recorded and it holds that $x \in \{0,1\}^\kappa$, $r \in \mathsf{COIN}$, and $c = \mathsf{ABM.enc}^{(t,u)}(pk, \iota(x); r)$. Otherwise, P_j does nothing.

Figure 1: Framework for consturing UC commitment from ABM encryption

Theorem 7.1 *The proposed scheme in Fig.1 UC-securely realizes the $\mathcal{F}_{\mathsf{MCOM}}$ functionality in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model in the presence of adaptive adversaries in the non-erasure setting.*

Due to the space limitation, we provide the proof in Appendix B

8 Instantiations of ABME

8.1 ABME from DDH Assumption

We first consider Waters signature [31] in a cyclic group *equipped with no bilinear map* and the DDH assumption holds on the group. Let g be a generator of a multiplicative group G of prime order q, where we assume that G is efficiently samplable. We let $g_i = g^{x_i}$ $(i = 1, 2)$ and $h_j = g^{y_j}$ $(j = 0, 1, \dots, \kappa)$, where $x_1, x_2,\ y_0, y_1, \dots, y_\kappa \in \mathbb{Z}/q\mathbb{Z}$. We write $t = (t_1, \dots, t_\kappa) \in \{0,1\}^\kappa$ where $t_i \in \{0,1\}$ $(i \in [\kappa])$. We let $y(t) = y_0 + \sum_{i=1}^{\kappa} t_i y_i \pmod q$ and define $H(t) = h_0 \prod_{i=1}^{\kappa} h^{t_i}$, that is, $H(t) = g^{y(t)}$. We let $S = \{0,1\}^\kappa \times G^2$. Then we define the set of Waters signature under $pk = (g_1, g_2, H(\cdot))$ as $L = \{(t, u) \mid (t, u) \in \{0,1\}^\kappa \times L_u(t)\}$ such that $L_u(t) = \{(u_v, u_t) \mid \exists v :\ u_v = g^v \text{ and } u_t = g_1^{x_2} H(t)^v\}$. We note that as mentioned above, the Waters signature defined on a cyclic group on which the DDH assumption holds constructs a PPRF. Then we construct a extractable sigma protocol on L_{pk}, which turns out to be an ABME.

- $\mathsf{ABM.gen}(1^\kappa)$**:** It generates g, (x_1, x_2), and $(y_0, \dots, y_\kappa)$ independently and uniformly from the above domains. It then computes $g_1, g_2,\ h_0, \dots, h_\kappa$, and sets (S, L) as above. It outputs $pk = (G, g, q, \lambda, (S, L), g_1, g_2, h_0, \dots, h_\kappa)$. and $sk = (pk, x_1, x_2, y_0, y_1, \dots, y_\kappa)$.

- $\mathsf{ABM.spl}(sk, t; v)$**:** It picks up at random $v \leftarrow \mathbb{Z}/q\mathbb{Z}$, and computes $u_v = g^v$ and $u_t = g_1^{x_2}(H(t))^v$. It then outputs $u = (u_v, u_t)$.

- $\mathsf{ABM.enc}^{((t,u)}(pk, x; (z, s))$**:** To encrypt message $x \in \{0,1\}^\lambda$, where $\lambda = c' \log \kappa$ for some constant $c' > 0$, it picks up $z, s \leftarrow \mathbb{Z}/q\mathbb{Z}$ independently, and then computes $A = g_1^z H(t)^s u_t^x$, $a = g^z g_2^x$, and $b = g^s u_v^x$. It outputs $c = (A, a, b)$ as ciphertext.

- $\mathsf{ABM.dec}^{(t,u)}(sk, c)$**:** To decrypt $c = (A, a, b)$, it searches $x \in \{0,1\}^\lambda$ such that

$$\frac{a^{x_1}}{Ab^{-y(t)}} \cdot \frac{u_t u_v^{-y(t)}}{g_2^{x_1}} = g_1^x.$$

  It aborts if it cannot find such x in a-priori bounded time $T = \Omega(2^\lambda)$.

- $\mathsf{ABM.col}_1^{(t,u)}(sk, v)$**:** It picks up at random $\omega, \eta \leftarrow \mathbb{Z}/q\mathbb{Z}$ and computes $A = g_1^\omega H(t)^\eta$, $a = g^\omega$, and $b = g^\eta$. It outputs $c = (A, a, b)$ and $\xi = (sk, t, u, v, \omega, \eta)$.

- $\mathsf{ABM.col}_2(\xi, x)$**:** To open c to $x \in \{0,1\}^\lambda$, it computes $z = \omega - x x_2 \bmod q$ and $s = \eta - xv \bmod q$ and outputs (z, s).

Roughly speaking, $\mathsf{ABM.enc}$ runs the simulation algorithm of a cannonical sigma protocol on L with message (challenge) x and $\mathsf{ABM.col}$ runs the real protocol of the sigma protocol on L with witness (sk, v).

In the trap-door mode when $(t, u) \in L$, we can consider a canonical sigma protocol so that the prover knows (x_2, v) such that $u_t = g_1^{x_2} H(t)^v$, $g_2 = g^{x_2}$, and $u_r = g^v$. Then, the first message of the canonical sigma protocol is (A, a, b), where $A = g_1^\omega H(t)^\eta$, $a = g^\omega$, and $b = g^\eta$ over randomly chosen $\omega, \eta \in \mathbb{Z}/q\mathbb{Z}$. For any challenge $x \in \{0,1\}^\kappa$, the answer can be computed by $z = \omega - x x_2$ and $s = \eta - xv$. It is verified as $A = g_1^z H(t)^s u_t^x$, $a = g^z g_2^x$, and $b = g^s u_v^x$.

In the decryption mode when $(t, u) \notin L$, the first message (A, a, b) from the simulator for the above canonical sigma protocol commits to x in the perfect binding manner. We now define ω, η, r as $a = g^\omega$, $b = g^\eta$, and $u_r = g^r$. Then, ω' and x_2' are uniquely defined as $A = g_1^{\omega'} H(t)^\eta$ and $u_t = g_1^{x_2'} H(t)^r$. Since

$(t, u) \notin L$, $x_2' \neq x_2$. By special soundness, there is only one x, such that the relation $\omega - xx_2 = \omega' - xx_2'$ holds. Therefore, $x = (\omega - \omega')(x_2 - x_2')^{-1}$. Notice that $Ab^{-y(t)} = g_1^{\omega'}$ and $u_t {u_r}^{-y(t)} = g_1^{x_2'}$. Hence,

$$\frac{a^{x_1}}{Ab^{-y(t)}} \cdot \frac{u_t {u_v}^{-y(t)}}{g_2^{x_1}} = g_1^{\frac{\omega-\omega'}{x_2-x_2'}}.$$

Therefore, the decryptor can find short secret $x \in \{0,1\}^\lambda$ from $g_1^x (= g_1^{\frac{\omega-\omega'}{x_2-x_2'}})$ in $\Omega(2^\lambda)$ steps.

Since $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$ composes a PPRF (under the DDH assumption), the proposed scheme is an instantiation of ABMEs.

Theorem 8.1 *The scheme as above is an ABME if the DDH assumption holds true.*

In the above construction, the message length is restricted to $c' \log \kappa$ for some constant c'. To encrypt a long message $x \in \{0,1\}^\lambda$, where $\lambda = O(\kappa)$, one can simply divide it into ρ messages $x = (x_1, \ldots, x_\rho)$, where each $x_i \in \{0,1\}^{\lambda_0}$ with $\lambda = \rho \cdot \lambda_0$, and send parallelized ciphertexts as a ciphertext of x, i.e.,

$$c = \mathsf{ABM.enc}^{(t,u)}(pk, x_1; (z_1, s_1)) \parallel \cdots \parallel \mathsf{ABM.enc}^{(t,u)}(pk, x_\rho; (z_\rho, s_\rho)).$$

Here each inner randomness, (z_i, s_i) are chosen uniformly and independently. This instantiation is also an all-but-many encryption. From a practitioners' point of view, this paralleized instantiation is a promising candidate of UC commitment. Only the disadvantage of this scheme is that the decryption algorithm is slow if λ_0 is large. However, when it is used as UC commitment, the decryption algorithm is not used among users. (Nobody knows the decryption key in the real world!) Only the simulator in the ideal world needs to decrypt ciphertexts. Therefore, we can expect relatively large λ_0. In fact, we can apply the baby-step giant-step to solving the discrete logarithm problem with short exponents, which is run in time $\Omega(\sqrt{2^{\lambda_0}})$. So, we set $\lambda_0 = 80$, where the task of solving the discrete log of 80-bit secret is almost to compute 2^{40} modular exponentiations, which is at most one-day task using a standard personal computer. We compare our DDH-based scheme with other promising candidates in Table 1, in which we allow simulator $\mathcal{S}$ to solve a discrete logarithm problem over G with 80-bit secret. Then our DDH-based scheme is 3 times faster than arguably the most efficient UC commitment scheme [25] (which is interactive and not non-erasure) when $\kappa = 160$ (80 bit security) and when $\kappa = 320$ (160 bit security).

8.2 ABME from Damgård-Jurik with expantion factor $O(1)$

We propose an efficient ABM encryption scheme based on Damgård-Jurik public-key encryption scheme [11] (a generization of Paillier public-key encryption scheme [29]).

Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be Damgård-Jurik (DJ) public-key encryption scheme, in which (N, v) is a public-key and (P, Q) is a secret-key where let $N = PQ$ be a composit number of large odd primes, P and Q, and $v \geq 1$ be a positive integer (where when $v = 1$ it is equivalent to Paillier). Let $g = (1 + N)$. To encrypt message $x \in \mathbb{Z}_{N^v}$, one computes $\mathbf{E}_{pk}(x; R) = g^x R^{N^v} \pmod{N^{v+1}}$ where $R \leftarrow \mathbb{Z}_{N^{v+1}}$. DJ scheme has the enhanced additively homomorphic property as defined in Appendix C. Namely, for $x_1, x_2 \in \mathbb{Z}_{N^v}$ and $R_1, R_2 \in \mathbb{Z}_{N^{v+1}}$, one can compute R such that $\mathbf{E}_{pk}(x_1 + x_2; R) = \mathbf{E}_{pk}(x_1; R_1) \cdot \mathbf{E}_{pk}(x_2; R_2)$. Acutually it can be done by computing $R = g^\gamma R_1 R_2 \pmod{N^{v+1}}$ where γ, $0 \leq \gamma < N$, is an integer such that $x_1 + x_2 = ((x_1 + x_2) \bmod N^v) + \gamma N^v$.

Let $g_1 = \mathbf{E}(x_1; R_1)$, $g_2 = \mathbf{E}(x_2; R_2)$, and $\vec{h} = (h_0, \ldots, h_\kappa)$ where $h_j \in \mathbb{Z}_{N^{v+1}}$ with $j = 0, 1, \ldots, \kappa$. Let us define $H(t) = h_0 \prod_{i=1}^{\kappa} h^{t_i} \pmod{N^{v+1}}$. Let us set $S = \{0,1\}^\kappa \times (\mathbb{Z}_{N^{v+1}})^2$ and $L = \{(t, (u_r, u_t)) \mid t \in \{0,1\}^\kappa \text{ and } (u_r, u_t) \in L_u(t)\}$, where $L_u(t) = \{(u_r, u_t) \mid \exists (r, R_r, R_t) : u_r = \mathbf{E}_{pk}(r; R_r) \text{ and } u_t = \mathbf{E}_{pk}((x_1 \cdot x_2); R_t) \cdot (H(t))^r\}$. We now provide the description of our ABME construction:

- ABM.gen(1^κ): It gets (pk, sk) generated by the key generator of the DJ encryption scheme on 1^κ, where $pk = (N, v)$ and $sk = (pk, P, Q)$. It generates $x_1, x_2 \leftarrow \mathbb{Z}_{N^v}$ to choose $g_1 \leftarrow \mathbf{E}_{pk}(x_1)$ and $g_2 \leftarrow \mathbf{E}_{pk}(x_2)$. It chooses $\vec{h}$ from the above domains. It sets (S, L) as above. It outputs $PK = (N, v, (S, L), g_1, g_2, \vec{h})$ and $SK = (PK, (x_1, x_2))$.

- ABM.spl($SK, t; (r, R_r)$): It chooses random $r \leftarrow \mathbb{Z}_{N^s}$, and computes $u_r = \mathbf{E}_{pk}(r; R_r)$ and $u_t = \mathbf{E}_{pk}((x_1 \cdot x_2); R_t)(H(t))^r$. It then outputs $u = (u_r, u_t)$.

- ABM.enc$^{(t,(u_r,u_t))}(x; (z, s, R_A, R_a, R_b))$: To encrypt message $x \in \mathbb{Z}_{N^v}$, it chooses $z, s \leftarrow \mathbb{Z}_{N^v}$, $R_A, R_a, R_b \leftarrow \mathbb{Z}_{N^{v+1}}$. It then computes $A = g_1^z H(t)^s u_t^x R_A^{N^v} \pmod{N^{v+1}}$, $a = \mathbf{E}(z; R_a) \cdot g_2^x$, and $b = \mathbf{E}(s; R_b) \cdot u_r^x$. It outputs $c = (A, a, b)$ as the ciphertext of x on $(t, (u_r, u_t))$.

- ABM.dec$^{(t,(u_r,u_t))}(sk, c)$: To decrypt $c = (A, a, b)$, it computes $\omega = \mathbf{D}_{sk}(a)$, $\eta = \mathbf{D}_{sk}(b)$, $r = \mathbf{D}_{sk}(u_r)$, $\omega' = \frac{\mathbf{D}_{sk}(A \cdot H(t)^{-\eta})}{x_1} \bmod N^v$, and $x_2' = \frac{\mathbf{D}_{sk}(u_t \cdot (H(t))^{-r})}{x_1} \bmod N^v$. It outputs $x = (\omega - \omega')(x_2 - x_2')^{-1} \bmod N^v$.

- ABM.col$_1^{(t,(u_r,u_t))}(sk, (r, R_r))$: It picks up at random $\omega, \eta \leftarrow \mathbb{Z}_{N^v}$ and $R_A', R_a', R_b' \leftarrow \mathbb{Z}_{N^{v+1}}$. It then computes $A = g_1^\omega H(t)^\eta (R_A')^{N^v}$, $a = g^\omega (R_a')^{N^v}$, and $b = g^\eta (R_b')^{N^v}$. It outputs $c = (A, a, b)$ and $\xi = (sk, t, (u_r, u_t), r, \omega, \eta, R_A', R_a', R_b')$.

- ABM.col$_2(\xi, x)$: To open c to x, it computes $z = \omega - x x_2 \bmod N^v$ and $s = \eta - xr \bmod N^v$. Then, it computes $\alpha = (\omega - x x_2 - z)/N^v$ and $\beta = (\eta - xr - s)/N^v$. It computes R_A, R_a, and R_b as $R_A' R_t^{-x} g_1^\alpha H(t)^\beta$, $R_a' R_2^{-x} g^\alpha$, and $R_b' R_r^{-x} g^\beta$, respectively. It outputs (z, s, R_A, R_a, R_b), which satisfy $A = g_1^z H(t)^s u_t^x R_A^{N^v} \pmod{N^{v+1}}$, $a = \mathbf{E}(z; R_a) \cdot g_2^x$, and $b = \mathbf{E}(s; R_b) \cdot u_r^x$.

ABM.col runs the real sigma protocol on L with witness $(sk, (r, R_r))$. By construction, the trap-door mode works correctly. On the contrary, ABM.enc runs the simulation algortithm of a canonical sigma protocol on language L with message (challenge) x. It is known that $\mathbb{Z}_{N^{v+1}}^\times$ is isomorphic to $\mathbb{Z}_{N^v} \times \mathbb{Z}_N^\times$ (the product of a cyclic group of order N^v and a group of order $\phi(N)$), and, for any $v < P, Q$, element $g = (1 + N)$, where $N = PQ$, has order N^v in $\mathbb{Z}_{N^{v+1}}^\times$ [11]. By this, $\mathbf{D}_{sk}(\alpha) \neq \varepsilon$ for every $\alpha \in \mathbb{Z}_{N^{v+1}}^\times$, meaning that every u_r, u_t in $\mathbb{Z}_{N^{v+1}}^\times$ can be decrypted to messages in $\mathbb{Z}_{N^v}$. Therefore, the decryption mode works correctly.

We assume DJ scheme is IND-CPA [10] and the non-multiplication assumption (defined in Appendix C) holds true. In addition, the image of $\mathbf{E}_{pk}$ is $\mathbb{Z}_{N^{v+1}}^\times$ and hence efficiently samplable. Therefore, (ABM.gen, ABM.spl) is a PPRF (See Theorem C.2). We have the following theorem.

Theorem 8.2 *The scheme constructed as above is an instantiation of ABMEs if Damgård-Jurik public-key encryption scheme is IND-CPA and the non-multiplication assumption defined in Appendix C holds.*

The message size is $v|N|$ and the ciphertext size is $(v+1)|N|$. The expansion factor is then $O((1+1/v))$ $= O(1)$ for constant $v \geq 1$ in the sense of both communication and computation. The public-key size (i.e., the common reference string size) is $O(\kappa^2)$.

References

[1] M. Bellare, S. Micali, and R. Ostrovsky. Perfect zero-knowledge in constant rounds. In *STOC '90*, pages 482–493. ACM, 1990.

[10] It is known that DJ scheme is IND-CPA if the decision composite residue (DCR) assumption holds true [11].

[2] M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 407–424. Springer-Verlag, 2009.

[3] M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G. R. Blakley and D. Chaum, editors, *CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–299. Springer-Verlag, 1985.

[4] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Cachin and Camenisch [5], pages 223–238.

[5] C. Cachin and J. Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[6] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001)*, pages 136–145. IEEE Computer Society, 2001. The full version available at at Cryptology ePrint Archive `http://eprint.iacr.org/2000/067`.

[7] R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 2001.

[8] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC 2002*, pages 494–503. ACM, 2002. The full version available at `http://eprint.iacr.org/2002/140`.

[9] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Desmedt [13], pages 174–187.

[10] I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *STOC 2003*, pages 426–437. ACM, 2003.

[11] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 125–140. Springer-Verlag, 2001.

[12] I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer-Verlag, 2002. The full version is available at `http://www.brics.dk/RS/01/41/`.

[13] Y. G. Desmedt, editor. *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[14] M. Fischlin, B. Libert, and M. Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Lee and Wang [24], pages 468–485.

[15] M. K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[16] E. Fujisaki. New constructions of efficient simulation-sound commitments using encryption and their applications. In O. Dunkelman, editor, *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 136–155. Springer, 2012.

[17] J. A. Garay, P. P.Mackenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 177–194. Springer-Verlag, 2003.

[18] R. Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In Franklin [15], pages 220–236. The full version available at at Cryptology ePrint Archive `http://eprint.iacr.org/2003/214`.

[19] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[20] J. Groth. *Honest Verifier Zero-Knowledge Arguments Applied*. PhD thesis, Basic Resarch in Computer Science, University of Aarhus, 2004.

[21] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 2008.

[22] D. Hofheinz. All-but-many lossy trapdoor functions. *IACR Cryptology ePrint Archive*, 2011:230, 2011. To Appear in Eurocrypt 2012.

[23] T. Itoh, Y. Ohta, and H. Shizuya. Language dependent secure bit commitment. In Desmedt [13], pages 188–201.

[24] D. H. Lee and X. Wang, editors. *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*. Springer-Verlag, 2011.

[25] Y. Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*. Springer-Verlag, 2011. The full version available at at Cryptology ePrint Archive `http://eprint.iacr.org/2011/180`.

[26] P. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In Cachin and Camenisch [5], pages 382–400.

[27] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[28] R. Nishimaki, E. Fujisaki, and K. Tanaka. An efficient non-interactive universally composable string-commitment scheme. *IEICE Transactions*, 95-A(1):167–175, 2012.

[29] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[30] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer-Verlag, 2008.

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

A UC framework and Ideal Commitment Functionality

The UC framework defines a probabilistic poly-time (PPT) environment machine $\mathcal{Z}$ that oversees the execution of a protocol in one of two worlds. In both worlds, there are an adversary and honest parties (some of whom may be corrupted by the adversary). In the *ideal world*, there additionally exists a trusted party (characterized by *ideal functionality* $\mathcal{F}$) that carries out the computation of the protocol, instead of honest parties. In the *real world*, the real protocol is run among the parties. The environment adaptively chooses the inputs for the honest parties, interacts with the adversary throughout the computation, and receives the honest parties' outputs. Security is formulated by requiring the existence of an ideal-world adversary (simulator) $\mathcal{S}$ so that no environment $\mathcal{Z}$ can distinguish the real world where it runs with the real adversary $\mathcal{A}$ from the ideal world where it runs with the ideal-model simulator $\mathcal{S}$.

In slightly more detail, the task of honest parties in the ideal world is only to convey inputs from the environment to the ideal functionality and vice verca (the honest parties communicate only with the environment and ideal functionalities). The environment may order the adversary to corrupt any honest party in any timing during the execution of the protocol (**addaptive corruption**), and it may receive the inner state of the honest party from the adversary. Therefore, the ideal-world simulator must simulate the inner state of the honest party as if it comes from the real world, because the honest parties in the ideal world do nothing except storing inputs to them). The inner state of the honest party includes randomness it has used. We insist that honest parties may not erase any of its state (**non-erasure setting**).

We denote by $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\kappa, z)$ the output of the environment $\mathcal{Z}$ with input z after an ideal execution with the ideal adversary (simulator) $\mathcal{S}$ and functionality $\mathcal{F}$, with security parameter κ. We will only consider black-box simulators $\mathcal{S}$, and so we denote the simulator by $\mathcal{S}^{\mathcal{A}}$ that means that it works with the adversary $\mathcal{A}$ attacking the real protocol. Furthermore, we denote by $\mathsf{Real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ the output of environment $\mathcal{Z}$ with input z after a real execution of the protocol π with adversary $\mathcal{A}$, with security parameter κ.

Our protocols are executed in the common reference string (CRS). model. This means that the protocol π is run in a hybrid model where the parties have access to an ideal functionality $\mathcal{F}_{\mathsf{crs}}$ that chooses a CRS according to the prescribed distribution and hands it to any party that requests it. We denote an execution of π in such a model by $\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$. Informally, a protocol π **UC-realizes a functionality** $\mathcal{F}$ in the $\mathcal{F}_{\mathsf{crs}}$ hybrid model if there exists a PPT simulator $\mathcal{S}$ such that for every non-uniform PPT environment $\mathcal{Z}$ and every PPT adversary $\mathcal{A}$, it holds that

$$\{\mathsf{Ideal}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*} \stackrel{\mathrm{c}}{\approx} \{\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*}.$$

The importance of the universal composability framework is that it satisfies a composition theorem that states that any protocol that is universally composable is secure when it runs concurrently with many other arbitrary protocols. For more details, see [6].

We consider UC commitment schemes that can be used repeatedly under a single common reference string (**re-usable common reference string**). The multi-commitment ideal functionality $\mathcal{F}_{\mathsf{MCOM}}$ from [8] is the ideal functionality of such commitments, which is given in Figure 2.

As in many previous works, the UC framework we use assumes authenticated communication. If it is not assumed, our protocols is executed in $\mathcal{F}_{\mathsf{crs}}$ and $\mathcal{F}_{\mathsf{auth}}$ hybrid models. For simplicity and conciseness, we simply assume communictation between parties are authenticated.

Functionality $\mathcal{F}_{\mathsf{MCOM}}$

$\mathcal{F}_{\mathsf{MCOM}}$ proceeds as follows, running with parties, $P_1, \dots, P_n$, and an adversary $\mathcal{S}$:

- **Commit phase:** Upon receiving input $(\texttt{commit}, sid, ssid, P_i, P_j, x)$ from P_i, proceed as follows: If a tuple $(\texttt{commit}, sid, ssid, \dots)$ with the same $(sid, ssid)$ was previously recorded, does nothing. Otherwise, record the tuple $(sid, ssid, P_i, P_j, x)$ and send $(\texttt{receipt}, sid, ssid, P_i, P_j)$ to P_j and $\mathcal{S}$.
- **Reveal phase:** Upon receiving input $(\texttt{open}, sid, ssid)$ from P_i, proceed as follows: If a tuple $(sid, ssid, P_i, P_j, x)$ was previously recorded, then send $(\texttt{reveal}, sid, ssid, P_i, P_j, x)$ to P_j and $\mathcal{S}$. Otherwise, does nothing.

Figure 2: The ideal multi-commitment functionality

B Proof of Theorem 7.1

For simplicity, we assume $\{0,1\}^\kappa \subset \mathsf{MSP}$, without loss of generality, which enables us to remove the injective map ι: $\{0,1\}^\kappa \to \mathsf{MSP}$ from the scheme. In addition, we define $L := L_{pk}$ for simplicity. The description of the simulator's task is described as follows:

The ideal-world adversary (simulator) $\mathcal{S}$:

- **Initialization step:** $\mathcal{S}$ chooses $(pk, sk) \leftarrow \mathsf{ABM.gen}(1^\kappa)$ and sets CRS to be pk (along with (U, S)).
- **Simulating ideal functionality $\mathcal{F}_{\mathsf{CRS}}$:** Since $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{CRS}}$, every request (even from a honest party) to achieve a common reference string comes to $\mathcal{S}$, it returns the above-chosen CRS to the requested party.
- **Simulating the communication with $\mathcal{Z}$:** Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.
- **Simulating the commit phase when P_i is honest:** Upon receiving from $\mathcal{F}_{\mathsf{MCOM}}$ the receipt message $(\texttt{receipt}, sid, ssid, P_i, P_j)$, $\mathcal{S}$ generates $u = \mathsf{ABM.spl}(sk, t; v)$ so that $(t, u) \in L$, where $t = (sid, ssid, P_i, P_j)$, and computes $(c, \xi) \leftarrow \mathsf{ABM.col}_1^{(t,u)}(sk, v)$, namely, c is a fake ciphertext on (t, u). $\mathcal{S}$ sends $(sid, ssid, (t, u), c)$ to adversary $\mathcal{A}$, as it expects to receive from P_i. $\mathcal{S}$ stores $(sid, ssid, P_i, P_j, t, c, \xi)$. If P_j is uncorrupted and adversary $\mathcal{A}$ sends $(sid, ssid, (t, u), c)$ to $\mathcal{S}$, as it expects to send to P_j, $\mathcal{S}$ runs the honest strategy of P_j.
- **Simulating the decommit phase when P_i is honest:** Upon receiving from $\mathcal{F}_{\mathsf{MCOM}}$ the message $(\texttt{open}, sid, ssid, P_i, P_j, x)$, $\mathcal{S}$ computes $r = \mathsf{ABM.col}_2(\xi, x)$ and sends $(sid, ssid, x, r)$ to adversary $\mathcal{A}$. If P_j is uncorrupted and adversary $\mathcal{A}$ sends $(sid, ssid, x, r)$ to $\mathcal{S}$, as it expects to send to P_j, $\mathcal{S}$ runs the honest strategy of P_j.
- **Simulating adaptive corruption of P_i after the commit phase but before the decommit phase:** When P_i is corrupted, $\mathcal{S}$ immediately read P_i's stored value $(sid, ssid, P_i, P_j, x)$, which value previously came from $\mathcal{Z}$ and was sent to $\mathcal{F}_{\mathsf{MCOM}}$, and then runs exactly the same as it does after it has received $(\texttt{open}, sid, ssid, P_i, P_j, x)$ in the decommit phase for honest P_i.
- **Simulating the commit phase when the committer P_i is corrupted and the receiver P_j is honest:** Upon receiving $(sid, ssid, (t, u), c)$ from $\mathcal{A}$, $\mathcal{S}$ decrypts $x = \mathsf{ABM.dec}^{(t,u)}(sk, c)$. If the

decryption is invalid, then $\mathcal{S}$ sends a dummy commitment $(\texttt{commit}, sid, ssid, P_i, P_j, \varepsilon)$ to $\mathcal{F}_{\mathsf{MCOM}}$. Otherwise, $\mathcal{S}$ sends $(\texttt{commit}, sid, ssid, P_i, P_j, x)$ to $\mathcal{F}_{\mathsf{MCOM}}$.

- **Simulating the decommit stage when the committer P_i is corrupted and the receiver P_j is honest:** $\mathcal{S}$ runs the honest strategy of P_j with $\mathcal{A}$ controlling P_i.
- **Simulating adaptive corruption of P_j after the commit phase but before the decommit phase:** When P_j has been corrupted, $\mathcal{S}$ simply sends $(sid, ssid, (t, u), c)$ to adversary $\mathcal{A}$ as if it comes from P_j.

We need to prove that the simulator described above satisfies that for every $\mathcal{Z}$ and every $\mathcal{A}$,

$$\{\mathsf{Ideal}_{\mathcal{F}_{\mathsf{MCOM}},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*} \stackrel{\mathrm{c}}{\approx} \{\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*}.$$

We now consider a sequence of the following games on which the probability spaces are identical, but we change the rules of games step by step.

Hybrid Game 1: In this game, the ideal commitment functionality, denoted $\mathcal{F}^1_{\mathsf{MCOM}}$, and the simulator, denoted $\mathcal{S}_1$, work exactly in the same way as $\mathcal{F}_{\mathsf{MCOM}}$ and $\mathcal{S}$ do respectively, except for **the case that P_i is honest**: In the commitment phase in Hybrid Game 1, $\mathcal{F}^1_{\mathsf{MCOM}}$ gives simulator $\mathcal{S}_1$ *the committed value x* by a honest party P_i together with $(\texttt{receipt}, sid, ssid, P_i, P_j)$. $\mathcal{S}_1$ then sets up $(t, u) \in L$ in the same way as $\mathcal{S}$ does (using sk), *but $\mathcal{S}_1$ computes c (without using sk) as $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$, by picking up $r \leftarrow \mathsf{COIN}$.* When simulating the decommit phase or simulating adaptive corruption of P_i before the decommit phase, $\mathcal{S}_1$ simply sends $(sid, ssid, x, r)$ to adversary $\mathcal{A}$.

Since $(t, u) \in L$, $\mathsf{ABM.enc}$ is in the trap-door mode, which means that for every v such that $u = \mathsf{ABM.spl}(sk, t; v)$ and every $x \in \mathsf{MSP}$, the first output of $\mathsf{ABM.col}^{(t,u)}_1(sk, v)$ and $\mathsf{ABM.enc}^{(t,u)}(pk, x)$ are statistically indistinguishable even if the consistent randomness is revealed. Therefore,

$$\{\mathsf{Ideal}_{\mathcal{F}_{\mathsf{MCOM}},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*} \stackrel{\mathrm{s}}{\approx} \{\mathsf{Hybrid}^1{}_{\mathcal{F}^1_{\mathsf{MCOM}},\mathcal{S}_1^{\mathcal{A}},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*}.$$

Hybrid Game 2: In this game, the ideal commitment functionality $\mathcal{F}^2_{\mathsf{MCOM}}$ and the simulator $\mathcal{S}_2$ work exactly in the same way as the counterparts do in Hybrid Game 1, except for **the case that P_i is corrupted and P_j is honest in the commitment phase:** In the commitment phase in Hybrid Game 2, when $\mathcal{S}_2$ receives $((t, u), c)$ from P_i controlled by adversary $\mathcal{A}$, where $t = (sid, ssid, P_i, P_j)$ and $u \in U$, then $\mathcal{S}_2$ *sends a dummy commitment* $(\texttt{commit}, sid, ssid, P_i, P_j, \varepsilon)$ to $\mathcal{F}^2_{\mathsf{MCOM}}$. In the decommit phase, when $\mathcal{S}_2$ receives $(sid, ssid, x', r)$ from P_i controlled by adversary $\mathcal{A}$, $\mathcal{S}_2$ ignores if $c \neq \mathsf{ABM.enc}^{(t,u)}(pk, x'; r)$; otherwise, it sends $(\texttt{open}, sid, ssid, x')$ to $\mathcal{F}^2_{\mathsf{MCOM}}$. Then, $\mathcal{F}^2_{\mathsf{MCOM}}$ *replaces the stored value ε with value x'* and sends $(\texttt{reveal}, sid, ssid, P_i, P_j, x')$ to P_j and $\mathcal{S}_2$.

Let us define BD_I as the event that the simmulator receives a *fake* ciphertext c on (t, u) from P_i controlled by adversary $\mathcal{A}$ in Hybrid Game I, where $I = 1, 2$. Remember that a ciphertext c is called fake if c is a valid ciphertext (i,e, there exist a pair of a message and randomness consistent with c) and $(t, u) \in L$.

The rules of the hybrid games, 1 and 2, may change only when BD_1 and BD_2 occur in each game, which means that $\neg\mathsf{BD}_1 = \neg\mathsf{BD}_2$ and thus, $\mathsf{BD}_1 = \mathsf{BD}_2$. So, we use the same notation BD to denote the event such that the simulator receives a fake ciphertext from the adversary in the hybrid games, 1 and 2, namely, $\mathsf{BD} := \mathsf{BD}_1 = \mathsf{BD}_2$.

By a simple evaluation such that $\Pr[A] - \Pr[C] \leq \Pr[B]$ if $\Pr[A \wedge \neg B] = \Pr[C \wedge \neg B]$, we have for fixed κ and z,

$$\mathsf{Dist}\Big(\mathsf{Hybrid}^1{}_{\mathcal{F}^1_{\mathsf{MCOM}},\mathcal{S}_1^{\mathcal{A}},\mathcal{Z}}(\kappa, z), \mathsf{Hybrid}^2{}_{\mathcal{F}^2_{\mathsf{MCOM}},\mathcal{S}_2^{\mathcal{A}},\mathcal{Z}}(\kappa, z)\Big) \leq \Pr[\mathsf{BD}],$$

where the output of $\mathcal{Z}$ is (assumed to be) a bit.

We now show that $\Pr[\mathsf{BD}]$ is negligible in κ.

Lemma B.1 *Event* BD *occurs in Hybrid game 2 at most with probability* $q_A\epsilon^{\mathsf{uf}}$*, where* q_A *denotes the total number of* $\mathcal{A}$ *sending the commitments to honest parties and* ϵ^{uf} *denotes the maximum advantage of an adversary breaking unforgeability of PPRF* $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$*.*

Proof. We construct the following algorithm B_0 that takes pk from $\mathsf{ABM.gen}$ and simulates the roles of $\mathcal{S}_2$ and $\mathcal{F}^2_{\mathsf{MCOM}}$ perfectly, interacting $\mathcal{Z}$ and $\mathcal{A}$, by having access to $\mathsf{ABM.spl}(sk,\cdot)$ as follows: **In the case when P_i is honest:** In the commitment phase when $\mathcal{Z}$ sends $(\mathtt{commit}.sid, ssid, P_i, P_j, x)$ to $\mathcal{F}^2_{\mathsf{MCOM}}$ (via honest P_i), B_0 submits $t = (sid, ssid, P_i, P_j)$ to $\mathsf{ABM.spl}(sk,\cdot)$ to obtain u such that $(t,u) \in L$. Then B_0 computes fake ciphertext $c \leftarrow \mathsf{ABM.enc}^{(t,u)}(pk, x)$ as commitment in the same way as $\mathcal{S}_2$ $(= \mathcal{S}_1)$ does. We note that c can be computed without sk as long as (t,u) is given. **In the case where P_i is corrupted and P_j is honest:** In the commitment phase when corruptted P_i controlled by $\mathcal{A}$ sends a commitment $((t,u),c)$ to $\mathcal{S}_2$ as it expects to send to honest P_j, B_0 simply plays the roles of $\mathcal{S}_2$ and $\mathcal{F}^2_{\mathsf{MCOM}}$. Later, in the opening phase when corruptted P_i controlled by $\mathcal{A}$ sends $(sid, ssid, x', r)$ to $\mathcal{S}_2$ as it expects to send to honest P_j, B_0 simply plays the role of $\mathcal{F}^2_{\mathsf{MCOM}}$.

We note that $\mathcal{S}_2$ uses sk only when it computes $u \leftarrow \mathsf{ABM.spl}(sk, t)$. in the commitment phase when P_i is honest. Since B_0 may have access to oracle $\mathsf{ABM.spl}(sk,\cdot)$, B_0 play the roles of $\mathcal{S}_2$ and $\mathcal{F}^2_{\mathsf{MCOM}}$ identically, interacting with $\mathcal{Z}$ and $\mathcal{A}$.

We now construct an algorithm B_χ, where $\chi \in [q_A]$, that is the same as B_0 except that it aborts and outputs (t,u) when $\mathcal{A}$ generates χ-th (in total) commitment $((t,u),c)$ to a honest party. Here, $q_{\mathcal{A}}$ denotes the total number of $\mathcal{A}$ sending the commitments to honest parties. We note that

$$\Pr[\mathsf{BD}] \leq \sum_{i=1}^{q_{\mathcal{A}}} \Pr[(t,u) \leftarrow B_i(pk)^{\mathsf{ABM.spl}(sk,\cdot),\mathcal{Z},\mathcal{A}} : (t,u) \in L]$$

The probability of B_i outputting $(t,u) \in L$ is bounded by ϵ^{uf}. Therefore, we have $\Pr[\mathsf{BD}] \leq q_{\mathcal{A}}\epsilon^{\mathsf{uf}}$. $\square$

By this, we have

$$\{\mathsf{Hybrid}^1{}_{\mathcal{F}^1_{\mathsf{MCOM}},\mathcal{S}_1^{\mathcal{A}},\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*} \stackrel{\mathrm{c}}{\approx} \{\mathsf{Hybrid}^2{}_{\mathcal{F}^2_{\mathsf{MCOM}},\mathcal{S}_2^{\mathcal{A}},\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*}.$$

Hybrid Game 3: In this game, $\mathcal{F}^3_{\mathsf{MCOM}}$ works exactly in the same way as $\mathcal{F}^2_{\mathsf{MCOM}}$. $\mathcal{S}_3$ works exactly in the same way as $\mathcal{S}_2$ except for **the case that P_i is honest in the commitment phase:** In the commitment phase when receiving $(\mathtt{receipt}, sid, ssid, P_i, P_j, x)$ from $\mathcal{F}^3_{\mathsf{MCOM}}$, $\mathcal{S}_3$ *picks up* $u \leftarrow U$ *at random*, instead of generating $u \leftarrow \mathsf{ABM.spl}(sk, t)$ so that $(t,u) \in L$, where $t = (sid, ssid, P_i, P_j)$. It then computes $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$. Note that x is given from the ideal commitment functionality. We note that in Hybrid Game 2, $\mathcal{S}_2$ makes use of sk only when it computes $u \leftarrow \mathsf{ABM.spl}(sk, t)$, whereas in Hybrid Game 3, $\mathcal{S}_3$ does not use sk any more. With an overwhelming probability, $(t,u) \in S \backslash L$.

The computational difference of the views of environment $\mathcal{Z}$ between these two games is bounded by pseudo-randomness of $\mathsf{ABM.spl}$, because we can construct a distinguisher D, using $\mathcal{Z}$ and $\mathcal{A}$ as oracle with having access to either of $\mathsf{ABM.spl}(sk,\cdot)$ or $U(\cdot)$, where oracle $U(t)$ returns random $u \in U$ on query t, but if $\mathsf{ABM.spl}(sk,\cdot)$ is deterministic, then $U(\cdot)$ returns the same u on t if it was previously queried. When D have access to $\mathsf{ABM.spl}(sk,\cdot)$, it simulates Hybrid Game 2; otherwise, it simulates Hybrid Game 3. Therfore, we have:

$$\{\mathsf{Hybrid}^2{}_{\mathcal{F}^2_{\mathsf{MCOM}},\mathcal{S}_2^{\mathcal{A}},\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*} \stackrel{\mathrm{c}}{\approx} \{\mathsf{Hybrid}^3{}_{\mathcal{F}^3_{\mathsf{MCOM}},\mathcal{S}_3^{\mathcal{A}},\mathcal{Z}}(\kappa,z)\}_{\kappa\in\mathbb{N};z\in\{0,1\}^*}.$$

Game $\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}$: The common reference string functionality $\mathcal{F}_{\mathsf{CRS}}$ parameterized by $\mathsf{ABM.gen}$ is given in Figure 3. The ideal CRS functionality $\mathcal{F}_{\mathsf{CRS}}$ is replaced with by $\mathcal{S}_3$'s task simulating $\mathcal{F}_{\mathsf{CRS}}$, which

Functionality $\mathcal{F}_{\mathsf{CRS}}$

$\mathcal{F}_{\mathsf{CRS}}$ parameterized by $\mathsf{ABM.gen}$ proceeds as follows:

- $\mathcal{F}_{\mathsf{CRS}}$ runs $(pk, sk) \leftarrow \mathsf{ABM.gen}(1^\kappa)$; and sets CRS to be pk. Upon receiving message $(\texttt{common-reference-string}, sid)$ with any sid, $\mathcal{F}_{\mathsf{CRS}}$ returns the same CRS to the activating party.

Figure 3: The common reference string functionality

is identical to the task of the ideal functionality. Other tasks made by $\mathcal{S}_3$ is replaced with those by the corresponding parties in the real world in the $\mathcal{F}_{\mathsf{CRS}}$ model. It is obvious from construction that both corresponding tasks between two worlds are identical. We further observe that $\mathcal{F}^3_{\mathsf{MCOM}}$ simply convey their input from a party to a party. Therefore, we can remove the ideal commitment functionality. Hence, we have

$$\{\mathsf{Hybrid}^3{}_{\mathcal{F}^3_{\mathsf{MCOM}}, \mathcal{S}_3^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N}; z\in\{0,1\}^*} \equiv \{\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N}; z\in\{0,1\}^*}.$$

Therefore; in the end, we have

$$\{\mathsf{Ideal}_{\mathcal{F}_{\mathsf{MCOM}}, \mathcal{S}^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N}; z\in\{0,1\}^*} \overset{\mathrm{c}}{\approx} \{\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)\}_{\kappa\in\mathbb{N}; z\in\{0,1\}^*}.$$

■

C PPRFs from Additive Hommorphic Encryption

Very recently in [22], Hofheinz has introduced a new assumption called the non-multiplication assumption for Damgård-Jurik public key encryption [11]. We propose a generalization of this assumption applied to any additive homomorphic public key encryption scheme.

Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a public-key encryption scheme in the standard sense. For given (pk, sk) generated by $\mathbf{K}(1^\kappa)$, let X be the message space and R be the coin space, with respects to pk. Let Y be the image of $\mathbf{E}_{pk}$, i.e., $Y = \mathbf{E}_{pk}(X; R)$. Here we assume that X is a commutative finite ring equipped with an additive operation $+$ and an multiplication operation $\times$. We also assume Y is a finite Abelian group with $\star$ operation.

We say that Π is an additively homomorphic public key encryption scheme if for every pk generated by $\mathbf{K}$, every $x_1, x_2 \in X$, and every $r_1, r_2 \in R$, there exists $r \in R$ such that

$$\mathbf{E}_{pk}(x_1; r_1) \star \mathbf{E}_{pk}(x_2; r_2) = \mathbf{E}_{pk}(x_1 + x_2; r).$$

In particular, we say that that Π is *enhanced* additively homomorphic if Π is additively homomorphic and $r \in R$ must be efficiently computable, given pk, and (x_1, x_2, r_1, r_2).

The mapping above is homomorphic in the mathematical sense – Namely, $\mathbf{E}_{pk}(x_1) \star \cdots \star \mathbf{E}_{pk}(x_n) \in Y$ for every $n \in \mathbb{Z}$ and every $x_1, \ldots, x_n \in X$. We write $c^z \in Y$, for $c \in Y$ and $z \in \mathbb{Z}$, to denote $\overbrace{c \star \cdots \star c}^{z}$.

What we want to assume is that Π is additively homomorphic, but not equipped with any efficient multiplicative operation $\diamond$ such that $\mathbf{E}_{pk}(x_1) \diamond \mathbf{E}_{pk}(x_2) = \mathbf{E}_{pk}(x_1 \times x_2)$ for any given $\mathbf{E}_{pk}(x_1)$ and $\mathbf{E}_{pk}(x_2)$. Formally, we define this property as follows:

Assumption C.1 (Non-Mult Assumption) *Let* Π *be an additively hommorphic public key encryption scheme along with a ring* $(X,+,\times)$ *as the message space w.r.t. pk and a group* $(Y,\star)$ *as the image of* $\mathbf{E}_{pk}$*. We say that the non-multiplication assumption holds on* Π *if for every non-uniform PPT algorithm* A*,* $\mathsf{Adv}_A^{\mathsf{mult}}(\kappa) = \mathsf{negl}(\kappa)$*, where* $\mathsf{Adv}_A^{\mathsf{mult}}(\kappa) \triangleq$

$$\Pr[(pk,sk) \leftarrow \mathbf{K}(1^\kappa);\, c_1,c_2 \leftarrow Y;\, c^* \leftarrow A(pk,c_1,c_2) : \mathbf{D}_{sk}(c^*) = \mathbf{D}_{sk}(c_1)\cdot \mathbf{D}_{sk}(c_2)].$$

We now construct a PPRF $(\mathsf{Gen}_{\mathsf{spl}}, \mathsf{Spl})$. Let $\Pi = (\mathbf{K},\mathbf{E},\mathbf{D})$ be an enhanced additively homomorphic public-key encryption scheme. Let X, R, and Y be the same as mentioned above. In addition, let group $(X,+)$ be cyclic, i.e., $(X,+) \simeq \mathbb{Z}/n\mathbb{Z}$ for some integer n. Let $x_1,x_2 \in X$. Let $g_1 \in \mathbf{E}_{pk}(x_1)$ and $g_2 \in \mathbf{E}_{pk}(x_2)$. Let $h_0,h_1,\ldots,h_\kappa \in Y$. Let us define $H(t)= h_0 \star \prod_{i=1}^{\kappa} h^{t[i]} \in Y$, where $t = (t[1],\ldots,t[\kappa]) \in \{0,1\}^\kappa$ is the bit representation of t. Let us define $L_u(t)$ such that

$$L_u(t) = \{(u_r,u_t) \in Y^2 \mid r = \mathbf{D}_{sk}(u_r) \text{ and } x_1 \times x_2 = \mathbf{D}_{sk}(u_t \star H(t)^{-r})\}.$$

We let $S = \{0,1\}^\kappa \times Y^2$ and $L = \{(t,(u_r,u_t)) \mid t \in \{0,1\}^\kappa \text{ and } (u_r,u_t) \in L_u(t)\}$.

A PPRF $(\mathsf{Gen}_{\mathsf{spl}}, \mathsf{Spl})$ is constructed as follows:

- $\mathsf{Gen}(1^\kappa)$**:** It runs $\mathbf{K}(1^\kappa)$ and obtain (pk,sk). It generates $x_1,x_2 \leftarrow X$ and $h_0,h_1,\ldots,h_\kappa \leftarrow Y$ uniformly. Set $d = x_1 \times x_2 \in X$. It generates $g_1 \leftarrow \mathbf{E}_{pk}(x_1)$ and $g_2 \leftarrow \mathbf{E}_{pk}(x_2)$. It outputs $PK= (pk,g_1,g_2,h_0,\ldots,h_\kappa)$ and $SK = (PK,d)$.
- $\mathsf{Spl}(SK,t;r)$**:** It picks up $r \leftarrow X$, generates $u_r \leftarrow \mathbf{E}_{pk}(r)$ and $u_t \leftarrow \mathbf{E}_{pk}(d)\star H(t)^r$, and then outputs $u = (u_r,u_t)$.

Theorem C.2 *Let* Π *be an enhanced additively homomorphic public-key encryption scheme mentioned above. Suppose that* Π *is IND-CPA and the non-multiplication assumption holds on* Π*. Then, the above* $(\mathsf{Gen}_{\mathsf{spl}}, \mathsf{Spl})$ *is a PPRF.*

Proof. The proof of pseudo randomness is almost straight-forward: Suppose that pk is generated by $\mathbf{K}(1^\kappa)$. Let S be a simulator such that it breaks IND-CPA of Π using A, where A is an adversary to output 1 if it determined that it has had access to a PPRF. We run S on pk. It picks up at random $x_1,x_2,x \leftarrow X$, $h_0,h_1,\ldots,h_\kappa \leftarrow Y$, and sets $g_1 \leftarrow \mathbf{E}_{pk}(x_1)$ and $g_2 \leftarrow \mathbf{E}_{pk}(x_2)$. It sends (m_0,m_1) to the challenger, where $m_0 = x$, and $m_1 = x_1 \times x_2 \in X$. It then receives $\mathbf{E}_{pk}(m_b)$, where b is a random bit chosen by the challenger. It then runs adversary A on $PK = (pk,g_1,g_2,\vec{h})$, where $\vec{h} = (h_0,h_1,\ldots,h_\kappa)$. For any query t, the simulator picks up random $r \leftarrow X$ and returns (u_r,u_t) such that $u_r = g^r$ and $u_t = \mathbf{E}_{pk}(m_b)\star(H(t))^r$. $u_t = \mathbf{E}_{pk}(x_1 \times x_2)\star(H(t))^r$. Finally, the simulator outputs the same bit that A outputs.

Note that when $b = 0$, (u_r,u_t) is distributed uniformly over Y^2. On the other hand, when $b = 1$. Since S outputs the same bit that A outputs, $\mathsf{Adv}_\Pi^{\mathsf{ind\text{-}cpa}} S(\kappa) = \Pr[S = 1 \mid b = 1] - \Pr[S = 1 \mid b = 0]$ $= \Pr[A = 1 \mid b = 1] - \Pr[A = 1 \mid b = 0] = \mathsf{Adv}_{\mathsf{pprf}} A(\kappa)$. Therefore, $\mathsf{Adv}_{\mathsf{pprf}} A(\kappa) = \mathsf{Adv}_\Pi^{\mathsf{ind\text{-}cpa}} S(\kappa) = \mathsf{negl}(\kappa)$.

The proof of unforgeability on this scheme is substantially similar to that in [4, 31, 2]. We provide a sketch of the proof.

Let G_0 be the original unforgeability game, in which $PK = (pk,g_1,g_2,\vec{h}) \leftarrow \mathsf{Gen}(1^\kappa)$; A takes PK, queries, $m_1,\ldots,m_{q_s}$, to $\mathsf{Spl}(sk,\cdot)$, and tries to output m_0 along with $u \in L_u(m_0)$ and $m_0 \notin \{m_1,\ldots,m_{q_s}\}$. Let us denote by ε_0 the advantage of A in G_0.

In game G_1, we modify the choice of $\vec{h}$ as follows: Recall now that $(X,+,\times)$ is a finite commutative ring such that $(X,+) \simeq \mathbb{Z}/n\mathbb{Z}$ for some integer n. Let Gen_1 be the generator in game G_1. Let $\theta = O(\frac{q_s}{\varepsilon_0})$, where q_s denotes the maximum number of queries A submits to Spl. Gen_1 picks up (pk,g_1,g_2) as Gen does.

It then picks up $a_0, a_1, \ldots, a_\kappa \leftarrow \mathbb{Z}/n\mathbb{Z}$. It picks up $y_1, \ldots, y_\kappa \leftarrow [0, \cdots, (\theta-1)]$ and $y_0 \in [0, \ldots, \kappa(\theta-1)]$. It finally outputs $PK = (pk, g_1, g_2, \vec{h})$, by setting $h_i = g^{a_i} g_2^{y_i}$ for $i \in [0, \cdots, \kappa]$. Since $(X, +) \simeq \mathbb{Z}/n\mathbb{Z}$ and $\mathbf{E}_{pk}$ is additively homomorphic, $Y \subset \mathbb{Z}/n\mathbb{Z}$. Hence, the distribution of $\vec{h}$ is identical to that in the previous game, and this change is conceptual. Therefore, the advantage of A in G_1, ε, is equal to ε_0.

For $t \in \{0,1\}^\kappa$, let $a(t) = a_0 + \sum t[i] \cdot a_i \pmod n$ and $y(t) = y_0 + \sum t[i] \cdot y_i \in \mathbb{Z}$. Then we have $H(t) = g^{a(t)} g_2^{y(t)}$.

Let $\gamma_{\vec{y}} : (\{0,1\}^\kappa)^{q_s+1} \to \{0,1\}$ be a predicate such that $\gamma_{\vec{y}}(\vec{t}) = 1$ if and only if $y(t_0) = 0$ and $\wedge_{i=1}^{q_s} y(t_i) \neq 0$, where $\vec{t} = (t_0, \ldots, t_{q_s}) \in (\{0,1\}^\kappa)^{q_s+1}$. Let $Q(\vec{t})$ be the event that at the end of game G_1, adversary A queries, $t_1, \ldots, t_{q_s}$ and outputs t_0 as the target message, on which A tries to generate the output of $\mathsf{Spl}(sk, t_0)$.

We now borrow the following lemmas due to [2].

Lemma C.3 *[2]. Let $Q(\vec{t})$ be the event in game G_1 mentioned above. Then,*

$$\Pr[Q(\vec{t}) \wedge (\gamma_{\vec{y}}(\vec{t}) = 1)] = \Pr[Q(\vec{t})] \Pr[\gamma_{\vec{y}}(\vec{t}) = 1].$$

Here the probability is taken over A, Gen_1, and Spl.

Lemma C.4 *[2]. Let n, θ, κ be positive integers, such that $\kappa\theta < n$. Let $y_0, y_1, \ldots, y_\kappa$ be elements in the domains mentioned above and let $y(t) = y_0 + \sum t_i \cdot y_i \in \mathbb{Z}$. Then, for every $t_0, \ldots, t_\kappa \in \{0,1\}^\kappa$, we have*

$$\frac{1}{\kappa(\theta-1)+1}\left(1 - \frac{q_s}{\theta}\right) \leq \Pr_{\vec{y}}[\gamma_{\vec{y}}(\vec{t}) = 1] \leq \frac{1}{\kappa(\theta-1)+1},$$

where the probability is taken over random variable $\vec{y} = (y_0, y_1, \ldots, y_\kappa)$ uniformly distributed over the specified domain mentioned above.

Now, in game G_2 we modify the challenger as follows: When the event that $\gamma_{\vec{y}}(\vec{t}) \neq 1$ occurs in game G_2, the challenger aborts the game. Let ε_2 be the advantage of A in game G_2. It immediately follows from the above lemmas that $\varepsilon_1 \cdot \min_{\vec{t}}\{\Pr_{\vec{y}}[\gamma_{\vec{y}}(\vec{t}) = 1]\} \leq \varepsilon_2$.

In game G_3, the challenger is given (pk, g_1, g_2) where $pk \leftarrow \mathbf{K}(1^\kappa)$ and $g_1, g_2 \leftarrow Y$. It picks up $\vec{a}$ and $\vec{y}$ as in game G_2. When A queries t, it picks up $r' \leftarrow X$ $(\simeq \mathbb{Z}/n\mathbb{Z})$ and selects $u_r \leftarrow g_1^{-\frac{1}{y(t)}} \star \mathbf{E}_{pk}(r')$ and $u_t \leftarrow g_1^{-\frac{a(t)}{y(t)}} \star \mathbf{E}_{pk}(0) \star (H(t))^{r'}$.

Let $r = \mathbf{D}_{sk}(u_r) = -\frac{x_1}{y(t)} + r'$. Then, it holds that for $y(t) \neq 0$, there is $v \in R$ such that $u_t = \mathbf{E}_{pk}(x_1 \times x_2; v) \star (H(t))^r$, because the decryption of the righthand side under sk is

$$x_1 x_2 + (a(t) + y(t)x_2)r = x_1 x_2 + (a(t) + y(t)x_2) \cdot \left(-\frac{x_1}{y(t)} + r'\right) = -\frac{a(t)}{y(t)} \cdot x_1 + (a(t) + y(t)x_2) \cdot r'.$$

Therefore, the righthand side is $g_1^{-\frac{a(t)}{y(t)}} \star \mathbf{E}_{pk}(0; v) \star (H(t))^{r'}$ for some $v \in R$. This is substantially equivalent to the technique of all-but-one simulation technique in [4]. As in game G_2, the simulator always abort if $\gamma_{\vec{y}}(\vec{t}) = 1$ holds. Hence, the advantage of A in this game, denoted ε_3, is equivalent to ε_2.

In the final game, we construct a simulator S that breaks the non-multiplication assumption. Let $(pk, sk) \leftarrow \mathbf{K}(1^\kappa)$ and $c_1, c_2 \leftarrow Y$. S takes (pk, c_1, c_2) as input. Then, it sets $g_1 := c_1$ and $g_2 := c_2$ and runs the challenger and adversary A in game G_3 on (pk, g_1, g_2).

We note that when A outputs $(u_r(t_0), u_t(t_0) \in L_u(t_0)$ in this game, it holds that $\mathbf{D}_{sk}(u_t(t_0)) = x_1 \times x_2 + r \cdot (a(t_0) + y(t_0)x_2) \cdot r$ where $r = \mathbf{D}_{sk}(u_r(t_0)) \in \mathbb{Z}/n\mathbb{Z}$ and $r \cdot (a(t_0) + y(t_0)x_2)$ denotes $\sum_{i=1}^{r}(a(t_0) + y(t_0)x_2)$. Since $y(t_0) = 0$, S now have

$$u_t(t_0) = \mathbf{E}_{pk}(x_1 \times x_2) \star (u_r)^{a(t_0)}.$$

Finally, S outputs $\mathbf{E}_{pk}(x_1 \times x_2)$ by computing $\frac{u_t(t_0)}{u_r^{a(t_0)}}$. By construction, it is obvious that the advantage of S is equivalent to ε_3.

■

D Fully-Equipped UC Commitment from Trap-Door Permutations

If we can construct an ABME from trap-door permutation (family), it is done, but we have no idea how to construct it. We instead construct a *weak* ABME from the same starting point. The only difference of weak ABME from standard ABME is that when $(t,u) \in L$, the distibution of $\mathsf{ABM.col}$ on (t,u) is not statistically but *computationally* indistinguishable from that of $\mathsf{ABM.enc}$. More precisely,

$$\left\{ \left(\mathsf{ABM.col}_1^{(t,u)}(sk,v)[1], \quad \mathsf{ABM.col}_2 \left(\mathsf{ABM.col}_1^{(t,u)}(sk,v)[2],\, x \right) \right) \right\} \stackrel{\mathrm{c}}{\approx} \left\{ \left(\mathsf{ABM.enc}^{(t,u)}(pk,x;r), \quad r \right) \right\}$$

for $x \in \mathsf{MSP}$, $(t,u) \in L$, and witness (sk,v) of $(t,u) \in L$.

We construct a weak ABM encryption scheme from trap-door permutations as follows.

Let $\mathcal{F} = \{(f, f^{-1}) \,|\, f : \{0,1\}^\kappa \to \{0,1\}^\kappa\}_{\kappa \in \mathbb{N}}$ be a trap-door permutation family and let b: $\{0,1\}^\kappa \to \{0,1\}$ be a hard-core predicate for a trap-door permutation f. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be the Blum-Goldwasser cryptosystem [3] that is a semantic secure public key encryption scheme, derived from the following encryption algorithm $\mathbf{E}_f(x;r) = f^{(k+1)}(r) \,||\, (x_1 \oplus b(r)) \,||\, \ldots \,||\, (x_k \oplus b(f^{(k)}(r)))$, where $(x_1, \ldots, x_\kappa)$, $x_i \in \{0,1\}$, denotes the bit representation of x. $r \in \{0,1\}^\kappa$ denotes inner randomness of this encryption and $f^{(k)}$ denotes k times iteration of f. We note that this public key encryption scheme is *oblivious samplable* with respects to pseudo-ciphertext space $\{0,1\}^{\kappa+k}$ [7], namely, $\{\mathbf{E}_f(x)\} \stackrel{\mathrm{c}}{\approx} \{U_{\kappa+k}\}$ for every message $x \in \{0,1\}^\kappa$, where $U_{\kappa+k}$ denotes a uniform distribution over $\{0,1\}^{\kappa+k}$. Let us denote by F: $\{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ a pseudo-random function (constructed from f in the standard way).

- $\mathsf{ABM.gen}(1^\kappa)$**:** It draws two trap-door permutations, (f, f^{-1}) and (f', f'^{-1}), over $\{0,1\}^\kappa$ uniformly and independently from $\mathcal{F}$. It then consturct the BG encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ with public key f and secret key f^{-1}. It also consturct the BG encryption scheme $\Pi' = (\mathbf{K}', \mathbf{E}', \mathbf{D}')$ with (f', f'^{-1}) and pseudo random function F from f'. It then picks up random $s \leftarrow \{0,1\}^\kappa$ and encrypt it to $e' = \mathbf{E}'(s;r)$. It outputs pk= (F, Π, Π', e') and $sk = (pk, f^{-1}, (s,r))$. We define $S = \{0,1\}^\kappa \times \{0,1\}^\kappa$.
- $\mathsf{ABM.spl}(sk,t)$**:** It takes tag $t \in \{0,1\}^\kappa$ and outputs $u = F_s(t)$. We define
$$L := L_{pk} = \{(t,u) \,|\, \exists (s,r)\, \texttt{s.t.}\; e' = \mathbf{E}'(s;r) \,\texttt{and}\, u = F_s(t)\}.$$
- $\mathsf{ABM.enc}^{(t,u)}(pk,x)$**:** It takes (t,u) and one bit message $x \in \{0,1\}$ along with pk, and first obtains a graph G (of q nodes) so that finding a Hamiltonian cycle in G is equivalent to finding (s,r) such that $u = F_s(t)$ and $e' = \mathbf{E}'(s;r)$, by using the NP-reduction. (If such (s,r) does not exist for given (t,u), G so obtained does not have a Hamiltonian cycle.) This encryption procedure is the same as the commitment described in [8], called the adaptive Hamiltonian commitment, except that in our scheme a commitment is encrypted under a public key f independent of F and Π', and an encrypted permutation or a pseudo ciphertext is also sent to the verifier.
  - To encrypt 0, it picks a random permutation $\pi = (\pi_1, \ldots, \pi_q)$ of q nodes, where $\pi_i \in \{0,1\}^{\log q}$, and encrypts every π_i and all the entries of the adjacency matrix of the permutated graph

$H = \pi(G)$. It outputs $\{A_i\}_{i\in[q]}$ and $\{B_{i,j}\}_{i,j\in[q]}$, such that $A_i = \mathbf{E}_f(\pi_i)$ $(\in \{0,1\}^{\kappa+\log q})$ and $B_{i,j} = \mathbf{E}_f(a_{i,j})$ $(\in \{0,1\}^{\kappa+1})$ where $a_{i,j} \in \{0,1\}$ denotes the (i,j)-entry of the adjacency matrix of H.

  – To encrypt 1, it picks q random $(\kappa + \log q)$-bit string A_i $(i \in [q])$ (corresponding to a pseudo ciphertext of π_i). It then chooses a randomly labeled Hamiltonian cycle, and for all the entries in the adjacency matrix corresponding to edges on the Hamiltonian cycle, it encrypts 1's. For all the other entries, it picks up random $\kappa + 1$-bit strings (corresponding to pseudo ciphertexts of the entries). It outputs $\{A_i\}_{i\in[q]}$ and $\{B_{i,j}\}_{i,j\in[q]}$, where a Hamiltonian cycle is embedded in $\{B_{i,j}\}_{i,j\in[q]}$, but the other strings are merely random strings.

- $\mathsf{ABM.dec}^{(t,u)}(sk, c)$**:** To decrypt $c = (\{A_i\}_{i\in[q]}, \{B_{i,j}\}_{i,j\in[q]})$, it firstly decrypt all elements to retreive π and matrix H. Then it checks that $H = \pi(G)$. If it holds, it outputs 0; otherwise, 1.

- $\mathsf{ABM.col}_1^{(t,u)}(sk)$**:** It first obtains a graph G (of q nodes) so that finding a Hamiltonian cycle in G is equivalent to finding (s, r) such that $u = F_s(t)$ and $e' = \mathbf{E}'(s; r)$, by using the NP-reduction. It picks a random permutation $\pi = (\pi_1, \ldots, \pi_q)$ of q nodes and computes $H = \pi(G)$. It encrypts under f all π_i's and all the entries of the adjacency matrix of the permutated graph $H = \pi(G)$. It outputs $c =(\{A_i\}_{i\in[q]}, \{B_{i,j}\}_{i,j\in[q]})$ and the Hamiltonian cycle of G, denoted ζ, where $\xi = (sk, t, u, \zeta, \pi)$.

- $\mathsf{ABM.col}_2(\xi, x)$**:** If $x = 0$, it open π and every entry of the adjacency matrix, otherwise if $x = 1$, it opens only the entries corresponding to the Hamiltonian cycle in the adjacency matrix.

Then, we apply this weak ABME to our framework (Fig. 1).

Theorem D.1 *The scheme in Fig.1 obtained by applying the above weak ABME UC-securely realizes the $\mathcal{F}_{\mathsf{MCOM}}$ functionality in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model in the presence of adaptive adversaries in the non-erasure setting.*

Proof. The only difference from the proof of Theorem 7.1 is when we compare the game of the ideal world with Hybrid Game 1. In the proof of Theorem 7.1, the outcome from $\mathsf{ABM.col}$ is statistically indistinguishable from the outcome from $\mathsf{ABM.enc}$ in the trap-door mode when $(t, u) \in L$. When using a weak ABME, the difference is computational. Hence, we need to construct a polynomially bounded distinguisher that tries to distinguish the two games where we cannot give sk to the distinguisher because it includes witness of (t, u), while the distinguisher should be able to decrypt valid ciphertexts generated by the adversary. Fortunately, in this construction, sk can be divided into (Π, f^{-1}) and $(\Pi', e', (s, r))$, where the former includes the decryption key and the latter includes the witness of (t, u). In addition, both are independently generated. Therefore, we can give the distinguisher only (Π, f^{-1}), which suffices to decrypt a valid ciphertext, and do not give it $(\Pi', e', (s, r))$ in order to distinguish the outcome from $\mathsf{ABM.col}$ from that of $\mathsf{ABM.enc}$. By this, we can conclude that the views of the environment in both games are computationally indistinguishable. ■

We note that if the common reference string must strictly come from the uniform distribution, we require trap-door permutations with dense public descriptions. This construction does not require non-interactive zero-knowledge proof systems. So, it is far more efficient than the previous fully-equipped UC commitment scheme from trap-door permutation [8].

E Small Remarks on ABMEs

We remark that any ABM encryption scheme can be converted to IND-CPA tag-based public key encryption scheme, by modifying the encryption algorithm to output (u, c) as a ciphertext of message x on tag t, as it picks up random $u \leftarrow U$ and computes $c \leftarrow \mathsf{ABM.enc}^{(t,u)}(pk, x)$. The proof is straightforward from indistinguishability of $(t, u) \in L_{pk}$ from $(t, u') \notin L_{pk}$.

We remark that a parallel encryptions of an ABM encryption scheme on the same (t, u) under the same public key is also an ABM encryption. Let x be a message longer than $|\mathsf{MSP}|$, i.e., $|x| > |\mathsf{MSP}|$. To encrypt x, simply divide x into m pieces, $\vec{x} = (x_1, \ldots, x_m)$, where $x_i \in \mathsf{MSP}$, and compute ciphertexts of each x_i under the same public key pk with the same (t, u) as

$$\mathsf{ABM.enc}^{(t,u)}(pk, \vec{x}; \vec{r}) := \Big(\mathsf{ABM.enc}^{(t,u)}(pk, x_1; r_1), \cdots, \mathsf{ABM.enc}^{(t,u)}(pk, x_m; r_m)\Big),$$

where $\vec{r} = (r_1, \ldots, r_m)$ and each r_i is chosen independently. It is easy to see that this parallel encryptions defines a new all-but-many encryption scheme.

F Comparison

Throughout this section, κ denotes the security parameter. λ denotes the size of the secret committed to. n denotes the number of all possible parties. In the column for # of rounds, a/b denotes the corresponding scheme requires a rounds in the commit phase, and b rounds in the opening phase. $T^{\mathsf{exp}}(\kappa)$ denotes the cost of computing one modular exponentiation of κ bit string. $T_{\mathsf{tdp}}(k)$ denotes the cost of computing one execution of trap-door permutation over $\{0,1\}^k$. CFP means Claw-free permutations. CRHF denotes Collision-Resistant Hash. DDH means the Decisional DH assumption. DLIN means the Decisional Linear Assumption. DCR denotes the Decesional Composit Residuosity assumption and Non-Mult is the non-multiplication assumption defined at Assumption C.1.

Here, when the length of committed secret λ is short, for instance $\lambda = 160, 320$, the schemes implemented in elliptic curve of small size have a great advantage over those based on factoring in terms of communication and computational costs. We compare our scheme in Sec. 8.1 with other candidates based on elliptic curve cryptosystems. In Table 1, G denotes a cyclic group implemented in elliptic curves. $\hat{G}$ denotes a symmetric paring group and G_T denotes the multiplicative group in a finite field defined by pairing operation $e(\cdot, \cdot) : \hat{G} \times \hat{G} \rightarrow G_T$ where the embedding degree is at most 6, i.e., $|G_T| \leq 6|\hat{G}|$. $\rho = \lambda \cdot {\lambda_0}^{-1}$, where $\lambda_0 = 80$. For 80 bit security, $\kappa = 160$, $\lambda = 160$, $|G| = 160$, and $|\hat{G}| = 170$ (so that $|G_T| \approx 1000$). For 160 bit security, $\kappa = 320$, $\lambda = 320$, $|G| = 320$, and $|\hat{G}| = 512$ (so that $|G_T| \approx 3000$). We use the simultaneous modular exponentiation technique appearing in Section 14.6 in [27], in which it is expected the cost of $(1 + \frac{1}{3}(1 - (\frac{1}{2})^{k-1}))$ modular exponentiation to compute $\prod_{i=1}^{k} g_i^{x_i}$ for small k and large $\kappa = |x_i|$. Therefore, it is almost the cost of $1\frac{1}{6}$ exponentiation when $k = 2$ and $1\frac{1}{4}$ when $k = 3$. We note that our proposal in Sec. 8.1 is 3 times faster than [25] and has less communication in the reasonable security parameters.

In Table 2, we compare our construction in Sec. D with the scheme appeared in [8]. Both schemes are fully-equipped UC commitments in the common reference string model and assumes only the existence of trap-door permutations. T_{NP} denotes the cost of one NP reduction from one-way function to a Hamiltonian graph. q denotes the number of the vertices of the Hamiltonian graph. Our scheme is far more efficient than [8], because ours does not require non-interactive zero-knowledge proof systems.

In Table 3, we compare our schemes with the other fully equipped schemes. Expansion factor of communication is evaluated by dividing the total size of communication in the commitment and opening phases by security parameter κ. Expansion factor of complexity is evaluated by dividing the total number of computing a *basic cryptographic function* on input of size κ by the security parameter – When schemes are implemented in elliptic curve or factoring based cryptosystems, we consider that one basic cryptographic computation is a modular exponentiation of size κ. When they are implemented from trap-door permutations, we think that one basic cryptographic computation is one execution of the permutation on $\{0,1\}^{\kappa}$.

Schemes	CRS size	Communication	Complexity	# of rounds	Non- -Erasure?	Assump.
CF01 [7]	$7\|G\|$	$11\lambda\|G\|$	$8\lambda T^{\mathsf{exp}}(\|G\|)$	1/1	Yes	DDH
($\kappa = 160$)	1280 bit	35.2 KB	$1280T^{\mathsf{exp}}(160)$			+CFP
($\kappa = 320$)	2240 bit	140.8 KB	$2560T^{\mathsf{exp}}(320)$			
Lin11 [25]	$8\|G\|$	$13\|G\| + 4\kappa$	$27\frac{1}{3}T^{\mathsf{exp}}(\|G\|)$	5/1	No	DDH
($\kappa = 160$)	1280 bit	2720 bit	$27\frac{1}{3}T^{\mathsf{exp}}(160)$			+CRHF
($\kappa = 320$)	2560 bit	5440 bit	$27\frac{1}{3}T^{\mathsf{exp}}(320)$			
FLM11 [14]	$9\|\hat{G}\|$	$21\|\hat{G}\|$	$\gg 2T^{\mathsf{exp}}(\|G_T\|)$	1/1	No	DLIN
($\kappa = 160$)	1530 bit	3570 bit	$\gg 432T^{\mathsf{exp}}(160)$			+CRHF
($\kappa = 320$)	4608 bit	10752 bit	$\gg 2 * 885T^{\mathsf{exp}}(320)$			
Sec. 8.1	$(\kappa + 4)\|G\|$	$(2 + 3\rho)\|G\|$	$3\frac{7}{12}\rho T^{\mathsf{exp}}(\|G\|)$	1/1	Yes	DDH
($\kappa = 160$, $\rho = 2$)	3.2 KB	1280 bit	$7\frac{1}{6}T^{\mathsf{exp}}(160)$			
($\kappa = 320$, $\rho = 4$)	13 KB	4800 bit	$10\frac{3}{4}T^{\mathsf{exp}}(320)$			

Table 1: Practitioners' Point of View: Comparison with the previous re-usable and adaptive UC commitments (to λ bit secret).

Schemes	CRS size	Communication	Complexity of each user	Assumption
CLOS02 [8]	$\omega(\kappa^3 \log(\kappa))$	$\omega(\lambda \cdot q^2\kappa^3 \log \kappa)$	$\lambda q^2 T_{\mathsf{NP}} + \omega(\lambda q^2 T_{\mathsf{tdp}}(\kappa^3 \log \kappa))$	TDP
Sec. D	$O(\kappa)$	$O(\lambda \cdot q^2\kappa)$	$T_{\mathsf{NP}} + \lambda q^2 T_{\mathsf{tdp}}(\kappa)$	TDP

Table 2: Fully-Equipped UC commitments (to λ bit secret) from general assumptions.

Schemes	Expansion factor of communication	Expansion factor of computation	CRS size	Fully-Equipped?	Assumption
CF01 [7]	$O(\kappa)$	$O(\kappa)$	$O(\kappa)$	Yes	DDH+CRHF
CLOS02 [8]	$\omega(\kappa^5 \log \kappa)$	$\omega(\kappa^2 \log \kappa \frac{T_{\mathsf{tdp}}(\kappa^3)}{T_{\mathsf{tdp}}(\kappa)})$	$\omega(\kappa^3 \log \kappa)$	Yes	TDP
Sec. 8.1	$O(\frac{\kappa}{\log \kappa})$	$O(\frac{\kappa}{\log \kappa})$	$O(\kappa^2)$	Yes	DDH
Sec. 8.2	$O(1)$	$O(1)$	$O(\kappa^2)$	Yes	DCR+Non-Mult
Sec D	$O(\kappa^3)$	$O(\kappa^2)$	$O(\kappa)$	Yes	TDP

Table 3: Comparison among the fully equipped UC commitment schemes.