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 [8, 9], with an unavoidable overhead of $O(\kappa)$ in the sense of communication and computational complexities; meaning that to commit λ bits, the communication and computational costs require $O(\lambda\kappa)$, where κ denotes the security parameter. Efficient construction of a fully-equipped UC commitment scheme was a long-standing open problem. We introduce a cryptographic primitive, called all-but-many encryptions (ABMEs), and prove that it is a translation of fully-equipped UC commitment in the primitive level. We then construct ABMEs from cryptographic primitives that we call a probabilistic pseudo random function family and extractable sigma protocols – the former is a probabilistic version of a pseudo random function family and the latter is a special kind of sigma (i.e., canonical 3-round public-coin HVSZK) protocols with some extractability. We provide fully-equipped UC commitment schemes from ABMEs under DDH and DCR-based assumptions, respectively. In particular, the DCR-based scheme is the first fully-equipped UC commitment scheme *with optimal expansion factor* $\Omega(1)$; to commit κ bits, the communication and computational costs are $\Omega(\kappa)$. We further construct a fully-equipped UC commitment scheme from a general assumption (in which trap-door permutations exist), which is far more efficient than the previous construction [9], because, unlike [9], our construction does not require non-interactive zero-knowledge proof systems.

1 Introduction

Universal composability (UC) framework [7] 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 construct high level UC-secure protocols. UC commitments imply UC zero-knowledge protocols [8, 13], which play an essential role to construct UC-secure two-party and multi-party computations [9]. Unfortunately, it is known that UC commitments cannot be realized without an additional set-up assumption [8]. The common reference string (CRS) model is most widely used as a set-up assumption when considering the UC framework. So, we also concentrate our 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].

[1] There are two different flavor 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.

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 *equivocal* commitment that can be opened correctly to any value. On the other hand, extractability can be interpreted as the simulator can correctly extract the contents of any *valid* commitment generated by any adversarial algorithm, even after it has given the adversary many equivocal 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.

Non-Interactivity. 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. From the practical point of view, non-interactivity is much more favorable – non-interactive protocols are much easier to be implemented and more resilient to denial of service attacks than interactive ones. Even from the theoretical viewpoint, non-interactive protocols generally make security proofs simpler when considering adaptive UC-security.

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, it is very important that 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 every time when a new execution of a protocol is invoked.

Adaptive Security. If an adversary must decide to corrupt parities only before the protocols start, it is called a static adversary. On the other hand, if an adversary can decide to corrupt the parties at any point in the executions of protocols, even revealing all their secrets, it is called an *adaptive* adversary. If a protocol is proven UC-secure against adaptive (resp. static) adversaries, it is called *adaptive* (resp. static) UC-secure. Adaptive UC security provides a very strong security guarantee.

Non-Erasure Model. When a party is corrupted, its complete inner state 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 states at any point of an execution. If such an assumption is unnecessary, we say that the protocol is defined in the *non-erasure* model. Since reliable erasure is difficult on a real system, it is desirable that a protocol is proven secure in the non-erasure model.

Canetti and Fischlin [8] presented the first UC secure commitment schemes, one of which is "fully-equipped" – non-interactive, adaptively secure, and non-erasure under a single re-usable common reference string. By construction, however, the proposal essentially requires $O(\kappa)$ overhead, meaning that, to commit to λ-bit secret, the UC commitment scheme requires $O(\lambda\kappa)$ communication bit and $O(\lambda\kappa)$ computational cost. Canetti et al. [9] also proposed another fully-equipped UC commitment scheme only from trap-door permutation. However, it is constructed in the same framework as in [8] and hence, expansion factor $O(\kappa)$ is unavoidable.

So far, the above two are the only known fully-equipped UC commitments. All known subsequent constructions of UC commitments [13, 11, 6, 25, 23, 15] have improved efficiency, but do not support at least one or two of the above requirements. Efficiency of UC commitment can be measured by round complexity (i.e., the number of interaction in the commitment and opening phases), communication complexity (i.e., the total amount of communication bits per secret length), and computational complexity (i.e., the total amount of work performed by participants). In addition, a CRS length also contributes to the efficiency of a UC commitment scheme.

1.1 Our Contributions

We present a general framework for constructing "fully-equipped" UC commitment schemes as mentioned above. The essence in the framework is in a notion of all-but-many encryption (ABME), which is a translation of fully-equipped UC commitments in the primitive level.

We construct ABMEs from two cryptographic primitives that we call probabilistic pseudo random functions (PPRF) and extractable sigma protocols (extΣ). 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.

We present fully-equipped UC commitment schemes from ABMEs from the Decisional Diffie-Hellman (DDH)

and Decisional Composite Residuosity (DCR) based assumptions, respectively. In particular, the DCR-based scheme is the first fully-equipped UC commitment scheme with expansion factor $O(1)$ in the communication and computational complexities. Namely, to commit $O(\kappa)$ bits, the communication and computational costs are $O(\kappa)$. To prove security of the DCR-based scheme, we assume slightly a stronger assumption than DCR such that Damgård-Jurik (additive) homomorphic encryption scheme is not multiplicatively homomorphic, which is similar to the assumption used in [21].

We also present a weak variant of ABME, which can be constructed only from a general assumption (that trap-door permutations exist) and converted to a fully-equipped UC commitment scheme [2]. Since it does not require non-interactive zero-knowledge proof systems, it is far more efficient than the previous scheme [9]. This construction is given in Appendix D.

1.1.1 Basic Idea – All-But-Many Encryption

We consider a public key encryption scheme with the following special properties: For Alice who does not know Bob's secret key, it works as a standard public key encryption scheme – when she encrypts a message under Bob's public key properly, Bob, the secret key holder, can decrypt the valid ciphertext correctly, However, it is not the case for Bob. He can generate a *fake* ciphertext under his public-key, which can be opened to any message along with the consistent randomness. It should be difficult for Alice to distinguish a fake ciphertext from a real ciphertext even after Bob revealed the message and the randomness used there. We also require that Bob can produce fake ciphertexts a-prior unbounded polynomially many times, but Alice cannot produce a fake ciphertext (on a fresh tag) even after she has received the fake ciphertexts. (To fit the UC framework, we assume that the encryption scheme is tag-based.) We call such encryption schemes all-but-many encryptions (ABMEs).

To construct ABMEs, as the first step idea, we call instance-dependent commitments [1, 22] to mind. An instance-dependent commitment scheme is an "instance-based" commitment scheme that additionally takes instance x as input to commit to a message and behaves differently depending on whether 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.

It is known that 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 [10] [3], 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 is statistically indistinguishable from the transcript on the same (x, e) generated by the real sigma protocol using witness w. This implies 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 equivocate commitments. We first run the real sigma protocol with witness w and outputs the first message a in the commitment phase, which is statistically indistinguishable from a commitment (i.e., the first message) generated by a honest committer (i.e., the simulation algorithm of the sigma protocol). Since the real sigma protocol can produce answer z for any challenge e, using witness w along with the randomness behind a, the simulator can open a into any value e in the opening phase. Therefore, this instance-dependent commitment scheme is equivocal when $x \in L$.

[2] An arbitrary weak ABME is transformed to a fully-equipped UC commitment scheme, by applying it to a different, less efficient framework, but our construction of a weak ABME from the general assumption is applied to the framework in Fig. 1.

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

On the contrary, when $x \notin L$, we need to extract e from the first message a (without randomness behind a) for our purpose. If it is possible, we call the sigma protocol *extractable*. More precisely, we consider a sigma protocol on an NP language L_{pk} indexed by (a series of) pk, in which when $x \notin L_{pk}$, a simulator can efficiently extract the challenge e from the first message a, for given (x, a, e), by using secret key sk behind pk (but no randomness under a is required). We call it an extractable sigma protocol. We insist that many sigma protocols can be actually converted to extractable sigma protocols. Indeed, most of efficient sigma protocols are implemented on Abelian groups associated with homomorphic maps, in which the first message of such sigma protocols implies linear equations of e and z, which also implies that the matrix derived from the linear equations is invertible if and only if $x \notin L_{pk}$. Therefore, if the simulator knows the contents of matrix, it can solve the linear equations when $x \notin L_{pk}$ and obtain e if the length of e is logarithmic. For instance, let L be a language of DDH. Let $(g_1, g_2, h_1, h_2) \notin L_{pk}$, meaning that $x_1 \neq x_2$ where $x_1 := \log_{g_1}(h_1)$ and $x_2 := \log_{g_2}(h_2)$. The first message (A_1, A_2) of a canonical sigma protocol on L implies linear equations

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ \alpha & \alpha x_2 \end{pmatrix} \begin{pmatrix} z \\ e \end{pmatrix}$$

where $A_1 = g_1^{a_1}$, $A_2 = g_2^{a_2}$, and $g_2 = g_1^{\alpha}$. The above matrix is invertible if and only if $x \notin L_{pk}$. We note that e is expressed as a linear combination of a_1 and a_2, i.e., $\beta_1 a_1 + \beta_2 a_2$, where the coefficients are determined by the matrix. Therefore, if the simulator knows the contents of the matrix and the length of e is logarithmic, it can search e by computing $g_1^e = A_1^{\beta_1} A_2^{\beta_2}$. In our actual constructions, the simulator does not always know the entire values of the matrix. However, if the matrix is carefully made so that e can be expressed as a *linear* combination of "unknown" values, that is, unknown values do not appear with a quadratic form or more degree of forms in the equation, we can still solve logarithmic e. In some case, on the contrary, we can invert homomorphic maps, for instance $f(a) = g^a$, using a trap-door information. Then the simulator can obtain a_1, a_2 as well as the entire values of the matrix, and extract e without the length restriction!

Finally, to construct ABMEs from (extractable) sigma protocols, we additionally require a language L_{pk} has some sort of "unforgeability" – The simulator can choose $x \in L_{pk}$, whereas the adversary cannot choose $x \notin L_{pk}$ (for a fresh tag t) even after it has been given many different $\tilde{x}$'s that belong to L_{pk}. Such a language can be constructed from probabilistic pseudo-random function families, which can be constructed by using a pseudo random function family with a public key encryption scheme, but an efficient construction may be actually easier than pseudo random function family. Later, we provide PPRF families, equipped with extractable sigma protocols.

By combining the above ideas together, we construct ABMEs. We first find a hard-decision language L_{pk} with "unforgeability" (defined over Abelian groups associated with homomorphic maps). We construct a sigma protocol for L_{pk} such that the first message of the sigma protocol (on instance x) implies linear equations of (e, z). Then it implies that the matrix derived from the linear equations is invertible if and only if $x \notin L_{pk}$. We carefully construct the sigma protocol so that e can be at least expressed as a linear combination of "unknown values", which makes the sigma protocol extractable (for logarithmic-size e). If the associated homomorphic map is invertible (with trap-doors), the extractable sigma protocol can extract e without length restriction.

To construct a "fully-equipped" UC commitment scheme from an ABME, we simply see the ciphertext generated by an ABM encryption scheme on random instance x as a UC commitment, where the public key is put 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) [21], which are lossy trap-door (deterministic) functions with (unbounded) *many* lossy tags. He has proposed two schemes based on 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 described later) as equivocal tags is inspired by the constructions of ABM-LTDF appeared in [21].

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 $L_R = \{X \mid \exists W : (X, W) \in R\}$.

2.1 The Universal Composability Framework

We work in the standard universal composability (UC) framework of Canetti [7]. We concentrate on the same model in [8] 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 [4].
- 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

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

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.

3.1 Construction of PPRFs

A PPRF $(\mathsf{Gen}_{\mathsf{spl}}, \mathsf{Spl})$ 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]: $\mathsf{Gen}_{\mathsf{spl}}(1^\kappa)$ 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 [27] 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. $\mathsf{Gen}_{\mathsf{spl}}(1^\kappa)$ 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 $\boldsymbol{h} = (h_0,h_1,\ldots,h_\kappa)$ 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 sigma protocols [10]. 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 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 $e \in \mathsf{ch}\Sigma$,

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

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 (a, e, z).

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 [5] 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 special soundness of the sigma protocols. Therefore, extractability is well defined. In other words, extractability guarantees 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.

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

[5] Precisly 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 $\kappa \in \mathbb{N}$.

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 [6].

- $\mathsf{ABM.dec}$ 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, $\frac{\#L_{pk}}{\#S} = \mathsf{negl}(\kappa)$.

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

   - **(Decryption mode)** For every $(t,u) \in S \setminus L_{pk}$, 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

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

     In addition,

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

     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 $r \in \mathsf{COIN_{enc}}$.

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 $r \in \mathsf{COIN_{enc}}$ 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 $(t,u) \in S \setminus L_{pk}$, 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* $(t,u) \in S \setminus L_{pk}$.

[6] 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.

6 ABME from $\mathsf{ext}\Sigma$ for Language derived from 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 PPRF. 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 $R_{pk} = \{((t,u),(sk,v)) \mid u = \mathsf{Spl}(sk,t;v)\}$, 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, $L_{pk} = \{(t,u) \mid \exists (sk,v) : (pk,sk) \in \mathsf{ABM.gen}(1^\kappa) \text{ and } u = \mathsf{Spl}(sk,t;v)\}$. 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 $X = (t,u) \in S \setminus L_{pk}$, $a \in \mathsf{sim}\Sigma^1(pk)(X,x)$ 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

The conversion from an ABME scheme to a fully-equipped UC commitment scheme is straight forward. We first put a public key pk of ABME in the common reference string. A committer P_i takes tag $t = (sid, ssid, P_i, P_j)$ and a message x committed to. It then picks up random u from U and compute an ABM encryption $c = \mathsf{ABM.enc}_{pk}^{(t,u)}(x;r)$ to send (u,c) to receiver P_j. To open the commitment, P_i sends (x,r) to P_j and P_j accepts if and only if $c = \mathsf{ABM.enc}_{pk}^{(t,u)}(x;r)$. We formally describe our UC commitment scheme in Fig. 1.

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.*

We provide a complete proof in Appendix B. Here we specially explain an essence to prove the following claim that two views of the environment $\mathcal{Z}$ are computationally indistinguishable between Hybrid Game1 and Hybrid Game2. Since an ABME scheme uses the *same sk* both to decrypt ciphertext c and to sample u^* such that $(t^*,u^*) \in L$, the reader might be confused about how the security proof goes through. So, we briefly describe the essence of the proof of the above statement, which is to show that $\mathcal{Z}$'s views in these games are computationally indistinguishable in the following man-in-the-middle attack.

In Hybrid Game1, the man-in-the-middle attack is as follows: $\mathcal{Z}$ gives to simulator $\mathcal{S}_1$ message x^* along with tag t^*. $\mathcal{S}_1$ computes $u^* \leftarrow \mathsf{ABM.spl}(sk,t^*)$ using sk, where $(t^*,u^*) \in L$, and $c^* \leftarrow \mathsf{ABM.enc}_{pk}^{(t^*,u^*)}(x^*)$ to send (u^*,c^*)

UC-commitment protocol from 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 $(\mathtt{commit}, sid, ssid, P_i, P_j, x)$ where $x \in \{0,1\}^\kappa$, party P_i proceed as follows: If a tuple $(\mathtt{commit}, sid, ssid, P_i, P_j, x)$ 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 $(\mathtt{receipt}, sid, ssid, P_i, P_j)$.

The decommitment phase:

- Upon receiving input $(\mathsf{open}, sid, ssid)$, 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 $(\mathsf{open}, sid, ssid, P_i, P_j, x)$ 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

to adversary $\mathcal{A}$. Then $\mathcal{A}$ sends to $\mathcal{S}_1$ (u,c) along with t, with $t \neq t^*$. $\mathcal{S}_1$ computes $\tilde{x} = \mathsf{ABM.dec}^{(t,u)}(sk, c)$ using sk and sends message $\tilde{x}$ to functionality $\mathcal{F}^1_{\mathsf{MCOM}}$. Then if Adv opens (u,c) correctly, for example, with (x,r). $\mathcal{S}_1$ sends (t, open) to $\mathcal{F}^1_{\mathsf{MCOM}}$. $\mathcal{F}^1_{\mathsf{MCOM}}$ sends **stored** $\tilde{x}$ to environment $\mathcal{Z}$.

In Hybrid Game2, the man-in-the-middle attack is as follows: $\mathcal{Z}$ gives to simulator $\mathcal{S}_1$ message x^* along with tag t^*. $\mathcal{S}_2$ computes $u^* \leftarrow \mathsf{ABM.spl}(sk, t^*)$ using sk, where $(t^*, u^*) \in L$, and $c^* \leftarrow \mathsf{ABM.enc}_{pk}^{(t^*,u^*)}(x^*)$ to send (u^*, c^*) to adversary $\mathcal{A}$. Then $\mathcal{A}$ sends to $\mathcal{S}_2$ (u,c) along with t, with $t \neq t^*$. Then $\mathcal{S}_2$ **instead sends** ϵ **to functionality** $\mathcal{F}^2_{\mathsf{MCOM}}$. Then if Adv opens (u,c) correctly, for example, with (x,r). $\mathcal{S}_1$ **instead sends** (t,x) **to** $\mathcal{F}^2_{\mathsf{MCOM}}$. $\mathcal{F}^2_{\mathsf{MCOM}}$ sends x to environment $\mathcal{Z}$.

We should prove that the above two $\mathcal{Z}$'s views are computationally indistinguishable, assuming that PPRF (ABM.gen, ABM.spl) is unforgeable, but as mentioned above, the simulator uses the same sk both to decrypt ciphertext c and to compute u^* and sk cannot be divided into mutually-independent decryption and sampling keys. Nevertheless, we can prove the statement as follows: Let BD_I denote the event in Hybrid Game I ($I \in \{1,2\}$) that the simulator receives a *fake* ciphertext (u,c) on tag t, that is, $(t,u) \in L$. If the event does not happen, $\mathcal{Z}$'s views in both games are identical, which means $\neg\mathsf{BD}_1 = \neg\mathsf{BD}_2$. Hence, the difference of $\mathcal{Z}$'s outputs in both games is bounded by Pr[BD], where $\mathsf{BD} := \mathsf{BD}_1 = \mathsf{BD}_2$. We then evaluate Pr[BD] in Hybrid Game2, not in Hybrid Game1, where the simulator does not decrypt any ciphertext. Hence, the probability is bounded by the advantage of unforgeability of (ABM.gen, ABM.spl).

8 Instantiations of ABME

8.1 ABME from DDH Assumption

We consider Waters signature [27] 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

$t = (t_1, \ldots, t_\kappa) \in \{0,1\}^\kappa$ 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 $H(t) = h_0 \prod_{i=1}^{\kappa} h_i^{t_i}$, 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 $pk = (g, g_1, g_2, H(\cdot))$ as $L = \{(t,u) \,|\, (t,u) \in \{0,1\}^\kappa \times L_u(t)\}$ such that $L_u(t) = \{(u_v, u_t) \,|\, \exists (x_2, v) : u_v = g^v;\ u_t = g_1^{x_2} H(t)^v;\ g_2 = g^{x_2}\}$. We note that as mentioned above, the Waters signature defined on a cyclic group on which the DDH assumption holds constructs a PPRF. We then construct an extractable sigma protocol on L, which turns out to be an ABME.

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

- $\mathsf{ABM.spl}(sk, t; v)$**:** 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 $\lambda = \Omega(\log \kappa)$, 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

$$\frac{a^{x_1} b^{y(t)}}{A} = \left(\frac{g_2^{x_1}}{u_t u_v^{-y(t)}} \right)^x.$$

  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, ABM.enc runs the simulation algorithm of a canonical sigma protocol on L with message (challenge) x and ABM.col runs the real protocol of the sigma protocol on L with witness (x_2, 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 $u_v = g^v$. 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, v as $a = g^\omega$, $b = g^\eta$, and $u_v = g^v$. Then, x_2' is uniquely defined as $u_t = g_1^{x_2'} H(t)^v$. If (A, a, b) can be opened with (z, s, x), it implies that

$$\begin{pmatrix} \log_g A \\ \omega \\ \eta \end{pmatrix} = \begin{pmatrix} x_1 & y(t) & x_1 x_2' + y(t) v \\ 1 & 0 & x_2 \\ 0 & 1 & v \end{pmatrix} \begin{pmatrix} z \\ s \\ x \end{pmatrix}$$

Since $(t,u) \notin L$, $x_2' \neq x_2$ and hence, the determinant of the matrix above is non-zero and (z, s, x) is unique.

Notice that $x_1 \omega + y(t)\eta - \log_g A = x_1(x_2 - x_2')x$. Since $g_1^{x_2'} = u_t u_v^{-y(t)}$,

$$\frac{a^{x_1} b^{y(t)}}{A} = \left(\frac{g_2^{x_1}}{u_t u_v^{-y(t)}} \right)^{x_1}$$

Therefore, the decryptor can find secret $x \in \{0,1\}^\lambda$ in $\Omega(2^\lambda)$ steps, where $\lambda = O(\log \kappa)$.

Since (ABM.gen, 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.*

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 [12] (a generization of Paillier public-key encryption scheme [26]).

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 $\mathbf{E}_{pk}(x_1 + x_2; R) = \mathbf{E}_{pk}(x_1; R_1) \cdot \mathbf{E}_{pk}(x_2; R_2)$. Actually 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 $\boldsymbol{h} = (h_0, \ldots, h_\kappa)$ where $h_j \in \mathbb{Z}_{N^{v+1}}$ with $j = 0, 1, \ldots, \kappa$. Let us define $H(t) = h_0 \prod_{i=1}^{\kappa} h_i^{t_i} \pmod{N^{v+1}}$. 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:

- $\mathsf{ABM.gen}(1^\kappa)$**:** 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 $\boldsymbol{h}$ from the above domains. It sets (S, L) as above. It outputs $PK = (N, v, (S, L), g_1, g_2, \boldsymbol{h})$ and $SK = (PK, (x_1, x_2))$.

- $\mathsf{ABM.spl}(SK, t; (r, R_r))$**:** 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 $u = (u_r, u_t)$.

- $\mathsf{ABM.enc}^{(t,(u_r,u_t))}(x; (z, s, R_A, R_a, R_b))$**:** 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 $(t, (u_r, u_t))$.

- $\mathsf{ABM.dec}^{(t,(u_r,u_t))}(sk, c)$**:** To decrypt $c = (A, a, b)$, it outputs

$$x = \frac{x_1 \mathbf{D}(a) + y(t)\mathbf{D}(b) - \mathbf{D}(A)}{x_1 x_2 - (\mathbf{D}(u_t) - y(t)\mathbf{D}(u_r))} \bmod N^v.$$

- $\mathsf{ABM.col}_1^{(t,(u_r,u_t))}(sk, (r, R_r))$**:** 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 $\xi = (sk, t, (u_r, u_t), r, \omega, \eta, R'_A, R'_a, R'_b)$.

- $\mathsf{ABM.col}_2(\xi, x)$**:** 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 $b = \mathbf{E}(s; R_b) \cdot u_r^x$.

$\mathsf{ABM.col}$ runs the real sigma protocol on L with witness $(sk, (r, R_r))$. By construction, the trap-door mode works correctly. On the contrary, $\mathsf{ABM.enc}$ runs the simulation algorithm of a canonical sigma protocol on language L with message (challenge) x. It is known that $\mathbb{Z}^\times_{N^{v+1}}$ is isomorphic to $\mathbb{Z}_{N^v} \times \mathbb{Z}^\times_N$ (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 $\mathbb{Z}^\times_{N^{v+1}}$ [12]. By this, $\mathbf{D}_{sk}(\alpha) \neq \varepsilon$ for every $\alpha \in \mathbb{Z}^\times_{N^{v+1}}$, meaning that every u_r, u_t in $\mathbb{Z}^\times_{N^{v+1}}$ can be decrypted to messages in $\mathbb{Z}_{N^v}$. Notice that (A, a, b) satisfies on $\mathbb{Z}_{N^v}$,

$$\begin{pmatrix} \mathbf{D}(A) \\ \mathbf{D}(a) \\ \mathbf{D}(b) \end{pmatrix} = \begin{pmatrix} x_1 & y(t) & x_1 x'_2 + y(t)\mathbf{D}(u_r) \\ 1 & 0 & x_2 \\ 0 & 1 & \mathbf{D}(u_r) \end{pmatrix} \begin{pmatrix} z \\ s \\ x \end{pmatrix},$$

where $\mathbf{D}(u_t) = x_1 x'_2 + y(t)\mathbf{D}(u_r)$. The determinant of the above matrix, $x_1(x'_2 - x_2)$, is non-zero if and only if $(t, (u_r, u_t)) \notin L$. Therefore, the decryption mode works correctly.

We assume DJ scheme is IND-CPA [7] and the non-multiplication assumption (defined in Appendix C) holds true. In addition, the image of $\mathbf{E}_{pk}$ is $\mathbb{Z}^{\times}_{N^{v+1}}$ 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 $O(\kappa^2)$. In the forthcoming paper, we provide schemes with $O(\kappa)$ sized public-key.

References

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

[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] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer-Verlag, 2003.

[7] 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`.

[8] 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.

[9] 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`.

[10] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Desmedt [14], pages 174–187.

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

[7] It is known that DJ scheme is IND-CPA if the decision composite residue (DCR) assumption holds true [12].

[12] 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.

[13] 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/`.

[14] 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.

[15] M. Fischlin, B. Libert, and M. Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 468–485. Springer-Verlag, 2011.

[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 M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 220–236. Springer-Verlag, 2004. 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] D. Hofheinz. All-but-many lossy trapdoor functions. *IACR Cryptology ePrint Archive*, 2011:230, 2011. To Appear in Eurocrypt 2012.

[22] T. Itoh, Y. Ohta, and H. Shizuya. Language dependent secure bit commitment. In Desmedt [14], pages 188–201.

[23] 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*, pages 446–466. Springer-Verlag, 2011. The full version available at at Cryptology ePrint Archive `http://eprint.iacr.org/2011/180`.

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

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

[26] 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.

[27] 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 versa (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 (**adaptive 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

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

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 [7].

We consider UC commitment schemes that can be used repeatedly under a single common reference string (**reusable common reference string**). The multi-commitment ideal functionality $\mathcal{F}_{\mathsf{MCOM}}$ from [9] is the ideal functionality of such commitments, which is given in Figure 2.

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

$\mathcal{F}_{\mathsf{MCOM}}$ proceeds as follows, running with parties, $P_1,\ldots,P_n$, 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 $(\texttt{commit}, sid, ssid, \ldots)$ 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

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 communication between parties are authenticated.

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 $\iota\colon \{0,1\}^\kappa \to \mathsf{MSP}$ 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 $(\mathtt{receipt}, sid, ssid, P_i, \mathcal{S}$ 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 $(\mathtt{open}, sid, ssid, P_i, P_j, x)$, $\mathcal{S}$ 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 $(\mathtt{open}, sid, ssid, P_i, P_j, x)$ 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 $(\mathtt{commit}, sid, ssid, P_i, P_j, \varepsilon)$ to $\mathcal{F}_{\mathsf{MCOM}}$. Otherwise, $\mathcal{S}$ sends $(\mathtt{commit}, sid, ssid, P_i, P_j, x)$ 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 $\mathcal{A}$,

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

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 (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 $(t,u) \in L$, 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 $\mathsf{ABM.col}_1^{(t,u)}(sk, v)$ and $\mathsf{ABM.enc}^{(t,u)}(pk, x)$ are statistically indistinguishable even if the consistent randomness is revealed. Therefore,

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

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* $(\mathtt{commit}, sid, ssid, P_i, P_j, \varepsilon)$ 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 $(\mathtt{open}, sid, ssid, x')$ to $\mathcal{F}^2_{\mathsf{MCOM}}$. Then, $\mathcal{F}^2_{\mathsf{MCOM}}$ *replaces the stored value* ε *with value* x' and sends $(\mathtt{reveal}, sid, ssid, P_i, P_j, x')$ to P_j and $\mathcal{S}_2$.

Let us define BD_I as the event that the simulator 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* (ABM.gen, ABM.spl).

Proof. We construct the following algorithm B_0 that takes pk from 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 corrupted 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 corrupted 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 $\Pr[\mathsf{BD}] \leq q_{\mathcal{A}}\epsilon^{\mathsf{uf}}$. □

By this, we have

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

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, $(t,u) \in S \setminus L$.

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:

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

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 is identical to the

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 $(\mathtt{common\text{-}reference\text{-}string}, sid)$ with any sid, $\mathcal{F}_{\mathsf{CRS}}$ returns the same CRS to the activating party.

Figure 3: The common reference string functionality

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

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

■

C PPRFs from Additive Homomorphic Encryption

Very recently in [21], Hofheinz has introduced a new assumption called the non-multiplication assumption for Damgård-Jurik public key encryption [12]. 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 homomorphic 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, $\mathsf{Adv}^{\mathsf{mult}}_A(\kappa) = \mathsf{negl}(\kappa)$, where $\mathsf{Adv}^{\mathsf{mult}}_A(\kappa) \triangleq$*

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

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 $PK = (pk, g_1, g_2, \boldsymbol{h})$, where $\boldsymbol{h} = (h_0, h_1, \ldots, h_\kappa)$. 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, $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_\Pi S(\kappa) = \Pr[S = 1 \mid b = 1] - \Pr[S = 1 \mid b = 0] = \Pr[A = 1 \mid b = 1] - \Pr[A = 1 \mid b = 0] = \mathsf{Adv}_{\mathsf{pprf}} A(\kappa)$. Therefore, $\mathsf{Adv}_{\mathsf{pprf}} A(\kappa) = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_\Pi S(\kappa) = \mathsf{negl}(\kappa)$.

The proof of unforgeability on this scheme is substantially similar to that in [4, 27, 2]. We provide a sketch of the proof.

Let G_0 be the original unforgeability game, in which $PK = (pk, g_1, g_2, \boldsymbol{h}) \leftarrow \mathsf{Gen}(1^\kappa)$; 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 $\boldsymbol{h}$ 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 $PK = (pk, g_1, g_2, \boldsymbol{h})$, 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 $\boldsymbol{h}$ 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 $\gamma_{\boldsymbol{y}} : (\{0,1\}^\kappa)^{q_s+1} \to \{0,1\}$ be a predicate such that $\gamma_{\boldsymbol{y}}(\boldsymbol{t}) = 1$ if and only if $y(t_0) = 0$ and $\wedge_{i=1}^{q_s} y(t_i) \neq 0$, where $\boldsymbol{t} = (t_0, \ldots, t_{q_s}) \in (\{0,1\}^\kappa)^{q_s+1}$. Let $Q(\boldsymbol{t})$ 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 $Q(\boldsymbol{t})$ be the event in game G_1 mentioned above. Then,*

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

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*

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

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

Now, in game G_2 we modify the challenger as follows: When the event that $\gamma_{\boldsymbol{y}}(\boldsymbol{t}) \neq 1$ 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 $\varepsilon_1 \cdot \min_{\boldsymbol{t}}\{\Pr_{\boldsymbol{y}}[\gamma_{\boldsymbol{y}}(\boldsymbol{t}) = 1]\} \leq \varepsilon_2$.

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 $\boldsymbol{a}$ and $\boldsymbol{y}$ 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

$$x_1x_2 + (a(t) + y(t)x_2)r = x_1x_2 + (a(t) + y(t)x_2) \cdot \Big(-\frac{x_1}{y(t)} + r'\Big) = -\frac{a(t)}{y(t)} \cdot x_1 + (a(t) + y(t)x_2) \cdot r'.$$

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 $\gamma_{\boldsymbol{y}}(\boldsymbol{t}) = 1$ 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 distribution of $\mathsf{ABM.col}$ on (t,u) is not statistically but *computationally* indistinguishable from that of $\mathsf{ABM.enc}$. More precisely,

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

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 $\mathcal{F} = \{(f, f^{-1}) \mid f : \{0,1\}^\kappa \to \{0,1\}^\kappa\}_{\kappa\in\mathbb{N}}$ 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 $(x_1, \ldots, x_k)$, $x_i \in \{0,1\}$, 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 $\{0,1\}^{\kappa+k}$ [8], namely, $\{\mathbf{E}_f(x)\} \overset{c}{\approx} \{U_{\kappa+k}\}$ 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 construct the BG encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ with public key f and secret key f^{-1}. It also construct 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 $pk = (F, \Pi, \Pi', e')$ 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

$$L := L_{pk} = \{(t,u) \mid \exists (s,r) \,\mathtt{s.t.}\, e' = \mathbf{E}'(s;r) \text{ and } u = F_s(t)\}.$$

- $\mathsf{ABM.enc}^{(t,u)}(pk,x)$**:** 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 [9], 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 retrieve 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 ABM.col is statistically indistinguishable from the outcome from 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 ABM.col from that of 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 [9].