

All-But-Many Encryption

A New Framework for 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 model under a re-usable common reference string. Previously, such “fully-equipped” UC commitment schemes have been known only in [11, 12], with expansion factor $\Omega(\kappa)$; meaning that to commit λ bits, communication strictly requires $\Omega(\lambda\kappa)$ bits, where κ denotes the security parameter. We introduce new abstraction, called *all-but-many encryption* (ABME), and prove that it can be transformed to a fully-equipped UC commitment scheme. We construct ABME from a variety of assumptions. In particular, we present an ABME scheme related to the DCR assumption, which implies the first fully-equipped UC commitment scheme with *optimal expansion factor* $O(1)$. In addition, it turns out an all-but-many lossy trapdoor function (ABM-LTF) [31] with a better lossy rate than [31].

Keywords: Public-key encryption, universally composable commitments, non-interactive, re-usable, non-erasure and adaptive security, and all-but-many lossy trapdoor functions.

1 Introduction

1.1 Motivating Application: Fully-Equipped UC Commitments

Universal composability (UC) framework [10] guarantees that if a protocol is 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 standalone setting. UC commitments is an essential ingredient to construct high level UC-secure protocols, which imply UC zero-knowledge protocols [11, 20] and UC oblivious transfer [12]. Therefore, any UC-secure two-party and multi-party computations can be realized in the presence of UC commitments. Since UC commitments cannot be realized without an additional set-up assumption [11], the common reference string (CRS) model is widely used.

A commitment scheme consists of a two-phase protocol between two parties, a committer and a receiver. In the commitment phase, a committer gives a receiver the digital equivalent of a *sealed envelope* containing value x , and, in 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 (*binding property*), whereas the receiver can learn nothing about x (*hiding property*) unless the committer helps the receiver open the envelope.

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* at the same time. Informally, equivocability of UC commitments 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

to any value. On the other hand, extractability can be interpreted as the ability of the simulator extracting the contents of a commitment generated by any adversarial algorithm, even after the adversary sees many equivocal commitments generated by the simulator.

Several factors as shown below feature UC commitments:

Interactivity. If an execution of a commitment scheme is completed, simply by sending each one message from the committer to the receiver both in the commitment and opening phases, then it is called *non-interactive*; otherwise, interactive. From a practical viewpoint, non-interactivity is definitely favorable – non-interactive protocols are much easier to implement and more resilient to real threats such as denial of service attacks. Even from a theoretical viewpoint, non-interactive protocols generally make security proofs simpler.

CRS Re-Usability. The CRS model assumes that CRS strings are generated in a trusted way and given to every party. For practical use, it is very important that a global single CRS string can be fixed beforehand and it can be *re-usable* in an unbounded number 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 decides to corrupt parties only before a protocol starts, it is called a static adversary. On the other hand, if an adversary can decide to corrupt parties at any point in the executions of protocols, it is called an *adaptive* adversary. The attacks of adaptive adversaries are more realistic in the real world. So, adaptive UC security is more desirable.

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. However, reliable erasure is a difficult task on a real system. So, it is desirable that a *non-erasure* protocol is proven secure.

1.2 Previous Works

Canetti and Fischlin [11] presented the first UC secure commitment schemes. One of their proposals is “fully-equipped”, i.e., *non-interactive, adaptively UC secure in the non-erasure model under a re-usable common reference string*. By construction, this scheme requires $\Omega(\lambda\kappa)$ bits when committing to λ -bit secret, where κ denotes the security parameter. Canetti et al. [12] constructed its generalized version from (enhanced) trapdoor permutations, which is simply inefficient. Damgård and Nielsen [20] proposed the first adaptively UC secure commitment schemes in the non-erasure model with expansion factor $O(1)$, meaning that to commit to λ -bit secret, communication requires only $O(\lambda)$ bits. However, the commitment phase must take three round interactions between a committer and a receiver. In addition, the CRS size grows linearly in the number of the parties. Soon after, Damgård and Groth [18] removed the dependency of the CRS size, using the simulation sound trap-door commitments, but the improved proposal is still interactive.

The subsequent commitment schemes such as [39, 36, 23, 5] are adaptively UC secure with expansion factor $O(1)$ under a constant size CRS string, but still *sacrifice at least* one or two requirements (See Table 1). Nisimaki, Fujisaki, and Tanaka [39] proposed non-interactive adaptively UC secure

commitments, but the CRS is just one-time, i.e., one needs a new common-reference string for each execution of the commitment protocol. Lindell [36] presented efficient static and adaptively UC secure commitment schemes based on the DDH assumption. Later, Blazy et al.[5] improved Lindell’s UC commitment schemes. However, those constructions require interaction and secure erasure. Fischlin, Libert, and Manulis [23] transformed Lindell’s static UC secure commitment scheme and Camenish and Shoup verifiable encryption scheme [9], into non-interactive adaptively UC secure commitment schemes, by removing the interaction in the sigma protocol using non-interactive Groth-Sahai proofs [29]. The resulting protocols still require secure erasure.

Recently, there is a series of the works [26, 17, 13, 24], in which efficient UC commitments have been proposed in the UC oblivious transfer (OT) hybrid model. The constructions are fast and compact except for the overhead of UC OT protocols. Using the OT extension technique [1], one can make the number of the execution of commitments independent of the number of the execution of OT protocols. However, those schemes are only *static* UC-secure.

To the best of our knowledge, there is no “fully-equipped” UC commitment that breaks the barrier of expansion factor $\Omega(\kappa)$. So far, efficient construction of a fully-equipped UC commitment scheme is a long-standing open problem (even with strong assumptions).

Table 1. Comparison among UC Commitments

Scheme	Expansion Rate of Communication.	Non-Interactiveness	Adaptive Security	Non-Erasure	Re-Usable CRS	Assumption(s)
[11]	$O(\kappa)$	✓	✓	✓	✓	DDH +CFP
[12]	$\omega(\kappa^3 \log \kappa)$	✓	✓	✓	✓	eTDP
[20]	$O(1)$		✓	✓	✓*	DCR
[18]	$O(1)$		✓		✓	DCR+SRSA+sOTS
[39]	$O(1)$	✓	✓	✓		DCR
[36]	$O(1)$		✓		✓	DDH+CRHF
[23]	$O(1)$	✓	✓		✓	DLIN+CRHF+Pairing
[5]	$O(1)$		✓		✓	DDH+CRHF
[26]	$O(1)$			-	✓	UC OT
[17]	$1 + o(1)$	✓**		-	✓	UC OT+PRG
[13]	$1 + o(1)$			-	✓	UC OT+PRG
[24]	$1 + o(1)$			-	✓	UC OT+PRG
Sec. 7	$O(1)$	✓	✓	✓	✓	DCR+Assump. 6 and 7
Sec. 9	$O(\kappa/\log \kappa)$	✓	✓	✓	✓	DDH+CRHF
Sec. D	$O(\kappa/\log \kappa)$	✓	✓	✓	✓	DDH
Sec. 10	$O(\kappa^3)$	✓	✓	✓	✓	eTDP

DDH: Decisional DH assumption. CFP: Claw-Free Permutations. eTDP: Enhanced Trap-Door Permutations. DCR: Decisional Composite Residuosity assumption. SRSA: Strong RSA assumption. sOTS: Strong One-Time Signatures. CRHF: Collision-Resistant Hash Family. DLIN: Decisional Linear assumption. Pairing: Pairing Groups. UC OT: UC Oblivious Transfer Oracle. PRG: Pseudo Random Generator. *: A CRS grows linearly with the number of parties. **: After the UC OT set-up phase.

1.3 Our Contribution

We introduce a new primitive, called **all-but-many encryption (ABME)**, which implies “fully-equipped” UC-secure commitments. There are a lot of obstacles to study the UC framework, due to compli-

cated definitions and proofs with many subtleties. Therefore, we believe that it is desirable to translate the essence of basic UC secure protocols into simple cryptographic primitives. To instantiate ABME, we divide the functionality into two primitives. Then, we consider the case that the primitives are suitably combined. We believe that this construction strategy would be helpful for finding more constructions in the future. We remark that our construction of ABME is inspired by that of all-but-many lossy trapdoor function (ABM-LTF) given by Hofheinz [31]. We will expose the relation in Sections, 1.10 and E.

We present a *compact* ABME scheme related to the DCR assumption, i.e., the first fully-equipped UC commitment scheme with expansion factor $O(1)$, meaning that to commit to λ -bit secret, it requires $O(\kappa)$ bits with a constant number of modular exponentiations. Our DCR-based ABME scheme can be transformed into an ABM-LTF scheme with a better lossy rate than [31] under the same assumption. We also provide ABME from the DDH assumption with overhead $O(\kappa/\log \kappa)$, which is slightly better than prior works with $O(\kappa)$. We also present a fully-equipped UC commitment scheme from *weak* ABME under a general assumption in which (enhanced) trapdoor permutations exist, which is far more efficient than the previous work [12] under the same assumption.

In the following sections, we describe more details.

1.4 All-But-Many Encryption

All-but-many encryption (ABME) enables a party with a secret-key (i.e., the simulator in the UC framework) to generate a *fake* ciphertext and to open it to any message with consistent randomness. In the case that a party is not given the secret-key (i.e., the adversary in the UC framework), it cannot distinguish a fake ciphertext from a *real* (honestly generated) ciphertext even after the message and randomness are revealed. In addition, it cannot produce a fake ciphertext (on a fresh tag) even after seeing many fake ciphertexts and their openings. We construct ABME from two new primitives, denoted probabilistic pseudo random functions and extractable sigma protocols. The former is a probabilistic version of a pseudo random function. The latter is a special type of a sigma protocol [15] with some *extractability*.

1.5 Probabilistic Pseudo-Random Function.

A pPRF = (KG, Spl) is a probabilistic pseudo-random function associated with a key-generation algorithm KG. Let $L_{pk}(t) := \{u | \exists (sk, v) : u = \text{Spl}(pk, sk, t; v)\}$, where (pk, sk) is generated by KG and v denotes random coins of Spl. Additionally, pPRF should be unforgeable – it is difficult to sample $u \in L_{pk}(t)$ for fresh t , if sk is not given. Sometimes, it should be unforgeable even on some superset $\hat{L}_{pk}(t)$.

1.6 Extractable Sigma Protocols

A sigma protocol Σ [15] on NP language L is a canonical 3-round public coin interactive proof system, so that a prover can convince a verifier that he knows witness w behind common input $x \in L$, where the prover first sends commitment a ; the verifier sends back challenge (public-coin) e ; the prover responds with z ; and the verifier finally accepts or rejects conversation (a, e, z) on x . A sigma protocol is associated with simulation algorithm $\text{simP}_{\Sigma}^{\text{com}}$ that takes x (regardless of whether $x \in L$ or not) and challenge e , and produces an accepting conversation $(a, e, z) \leftarrow \text{simP}_{\Sigma}^{\text{com}}(x, e)$

without witness w . If $x \in L$, the distribution of (a, e, z) produced by $\text{simP}_{\Sigma}^{\text{com}}(x, e)$ on random e is statistically indistinguishable from the transcript generated between two honest parties, called *honest-verifier statistically zero knowledge* (HVSZK). If $x \notin L$, for every a there is at most one e such that (a, e, z) can be an accepting conversation on x , called *special soundness*.

An extractable sigma protocol $\Sigma^{\text{ext}} = (\Sigma, \text{Ext})$ on L_{pk} is a special type of a sigma protocol, associated with a DPT algorithm Ext , with the following properties:

- Σ is a sigma protocol on L_{pk} .
- There is a disjoint set L_{pk}^{co} such that $L_{pk} \cap L_{pk}^{\text{co}} = \emptyset$ and for all pk , there is sk such that $\text{Ext}(sk, x, a) = e$ for all $x \in L_{pk}^{\text{co}}$ and all $a \in \text{simP}_{\Sigma}^{\text{com}}(x, e)_1$, where $\text{simP}_{\Sigma}^{\text{com}}(x, e)_1$ is the first output of $\text{simP}_{\Sigma}^{\text{com}}(x, e)$.

Due to special soundness, for all (x, a) with $x \notin L_{pk}$, e is uniquely determined (if it exists). So, the extraction algorithm is well defined. We will show how to construct extractable sigma protocols in Sec. 1.8.

1.7 pPRF + Σ^{ext} \rightarrow ABME

We say that pPRF = (KG, Spl) and $\Sigma^{\text{ext}} = (\Sigma, \text{Ext})$ are well-combined if:

- KG(1^κ) outputs $(pk, sk^{\text{spl}}, sk^{\text{ext}})$, where sk^{spl} is used as a secret key of Spl and sk^{ext} is a secret key of Ext.
- For each pk , there is a set L_{pk}^{co} such that Σ^{ext} is an extractable sigma protocol on $L_{pk} = \{(t, u) | \exists (sk^{\text{spl}}, v) : u = \text{Spl}(pk, sk^{\text{spl}}, t; v)\}$, and has extractability on set L_{pk}^{co} with sk^{ext} .
- pPRF is unforgeable on $\hat{L}_{pk} := U'_{pk} \setminus L_{pk}^{\text{co}}$, where U'_{pk} is a universe.

For well-combined pPRF and Σ^{ext} , one can construct an ABME scheme, by taking the similar method to convert an ordinary sigma protocol to an instance-dependent commitment scheme [3, 32]. Here is the strategy.

- To encrypt message e on tag t , a sender picks random u , runs $\text{simP}_{\Sigma}^{\text{com}}$ on instance (t, u) with challenge e with random z , to compute $(a, e, z) = \text{simP}_{\Sigma}^{\text{com}}(pk, (t, u), e; z)$, and finally outputs (u, a) as a ciphertext. Here z is regard as the random coins of the ciphertext. Due to the unforgeability condition of pPRF, it holds that $(t, u) \in U'_{pk} \setminus \hat{L}_{pk} (= L_{pk}^{\text{co}})$ with an overwhelming probability. Then, e is uniquely determined given $((t, u), a)$. By our precondition, we can decrypt (t, u, a) using sk^{ext} , as $e = \text{Ext}(sk^{\text{ext}}, (t, u), a)$ because $(t, u) \in L_{pk}^{\text{co}}$.
- To make a fake (equivocal) ciphertext on tag t , one picks up random v and compute $u = \text{Spl}(pk, sk^{\text{spl}}, t; v)$ using sk^{spl} . Then he computes a , as same as an honest prover computes the first message on common input (t, u) with witness (sk^{spl}, v) . To open a to arbitrary e , he produces the response z in the sigma protocol. By construction, he can open a to any e because $(t, u) \in L_{pk}$.

We note that an adversary cannot distinguish a real ciphertext produced by a honest sender from a fake ciphertext produced by a simulator, due to pseudo-randomness of pPRF. In addition, an adversary cannot produce a fake ciphertext even after seeing many fake ciphertexts, due to unforgeability (on \hat{L}_{pk}) of pPRF.

1.8 Realizing Extractable Sigma Protocols.

Although sigma protocols (with HVSZK) exist on *many* NP languages, it is not known how to extract the challenge as discussed above. Here we observe that sigma protocols are often implemented on Abelian groups associated with homomorphic maps, in which the first message of such sigma protocols implies a system of linear equations with e and z . Hence, there is a matrix derived from the linear systems. Due to completeness and special soundness, there is an invertible (sub) matrix if and only if $x \notin L_{pk}$ (provided that the linear system is defined in a finite field). Therefore, if one knows the contents of the matrix, one can solve the linear systems when $x \notin L_{pk}$ and obtain e if its length is logarithmic. Suppose for instance that L_{pk} is the DDH language – it does not form a pPRF, but a good toy example to explain how to extract the challenge. Let $x = (g_1, g_2, h_1, h_2) \notin L_{pk}$, 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_{pk} 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 $(g_1, g_2, h_1, h_2) \notin L_{pk}$. We note that e is expressed as a linear combination of a_1 and a_2 , i.e., $(\beta_1(\det A)^{-1})a_1 + (\beta_2(\det A)^{-1})a_2$, where the coefficients are determined by the matrix. Therefore, if the decryption algorithm takes (α, x_1, x_2) and the length of e is logarithmic, it can find e by checking whether $(g_1^{\det A})^e = A_1^{\beta_1} A_2^{\beta_2}$ or not. In a general case where a partial information on the values of the matrix is given, the decryption algorithm can still find logarithmic-length e if the matrix is made so that e can be expressed as a *linear* combination of *unknown values* – the unknown values do not appear with a quadratic form or a more degree of forms in the equations.

In a good case, the decryption algorithm can invert homomorphic map $f(a) = g^a$, using trapdoor f^{-1} . Then, one can obtain (a_1, a_2) as well as the entire values of the matrix and hence extract even *polynomial-length* e . This corresponds to the case of our DCR based implementation, where the corresponding linear system is defined on a finite *ring*, such as \mathbb{Z}_{n^d} . The matrix (say A) derived from the linear system is invertible if and only if $(\det A)^{-1} \bmod n^d$ exists, which corresponds to the condition $x \notin \widehat{L}_{pk}$ for some superset \widehat{L}_{pk} . We note that although $x \notin L_{pk}$ iff $\det A \neq 0 \pmod{n^d}$, it does not suffice for the above because of the divisors. We require unforgeability not on L_{pk} but on \widehat{L}_{pk} , so that the output produced by an adversary can be forced in $L_{pk}^{\text{co}} = U'_{pk} \setminus \widehat{L}_{pk}$.

1.9 Instantiations.

We present ABME schemes from three different types of pPRFs. We first propose a pPRF from Waters signature scheme [45] defined over a ring *equipped with no bilinear map*. As the associated homomorphic map, we employ Damgård-Jurik (DJ) PKE [19]. The output of the Waters based probabilistic pseudo-random function looks pseudo random due to semantic security of DJ PKE. The construction inherits unforgeability from Waters signature scheme under an analogue of the DH assumption in the additive homomorphic encryption. Precisely, we require one more assumption related to DJ PKE, because we require unforgeability on some superset of the language derived from the Waters-like pPRF. we construct an extractable sigma protocol on it. Since the homomorphic map is invertible using the secret key of DJ PKE, we can obtain a compact ABME scheme and hence a fully-equipped UC commitment scheme with expansion factor $O(1)$ with a constant number of computational complexity.

In Appendix D, we simply use as pPRF the Waters signature scheme on the prime order group without bilinear map and provide the DDH version of the ABME scheme above. Although its expansion factor is just $O(\kappa/\log \kappa)$, it is better than the prior work [11] (with $O(\kappa)$). This scheme is helpful to understand our main proposal, because of the simpler construction. So, we recommend the reader to read that section first, if the main proposal looks complicate.

In another construction, pPRF is given by combining a semantic secure PKE scheme with an IND-CCA secure Tag-PKE scheme. We combine ElGamal PKE with tag-based Twin Cramer-Shoup PKE [14] and construct an ABME scheme from pPRF under the DDH assumption. The expansion factor of this scheme is also $O(\kappa/\log \kappa)$. One advantage of this scheme is that it has a short public key (of a constant number of group elements), unlike the proposed schemes above.

We also provide a generic construction of pPRF from a pseudo random function family and a semantically secure PKE scheme. We employ this type of pPRFs to construct a UC commitment scheme from general assumptions.

1.10 Other Related Works

Fehr et al. [22] proposed a PKE scheme secure against simulation-based selective opening chosen-ciphertext attack (SIM-SO-CCA). In general, the notion of SIM-SO-CCA secure PKE is related to that of ABME, but both are incomparable. Indeed, Fehr et al. scheme [22] does not satisfy the requirements of ABME, while ABME does not satisfy SIM-SO-CCA PKE in general, because it does not support CCA security. Although [22] could be tailored to a fully-equipped UC commitment scheme, it cannot overcome the barrier of expansion factor $\Omega(\kappa)$, because it strictly costs $\Omega(\lambda\kappa)$ bits to encrypt λ bit.

Hofheinz presented the notion of all-but-many lossy trapdoor function (ABM-LTF) [31], mainly to construct indistinguishability-based selective opening CCA (IND-SO-CCA) secure PKE. ABM-LTF is a lossy trapdoor function (LTF) [42] with (unbounded) *many* lossy tags. The relation between ABM-LTF and ABME is a generalized analogue of LTF and lossy encryption [41, 2] with unbounded many loss tags. However, unlike the lossy encryption, ABME always requires an *efficient* opening algorithm that can open a ciphertext on a lossy tag to any message with consistent randomness. As mentioned earlier, our construction idea of ABME is strongly inspired by that of ABM-LTF [31]. Hofheinz provided a matrix-based function $\mathbf{Y} = \mathbf{A}\mathbf{X}$, where \mathbf{A} denotes a square matrix and \mathbf{Y}, \mathbf{X} denote column vectors. The algorithm to produce lossy tags is pPRF in our definition. The lossy tags are carefully embedded in matrix \mathbf{X} so that the matrix can be non-invertible if tags are lossy; otherwise invertible. Hofheinz proposed two instantiations. In the DCR-based ABM-LTF, the lossy tags are an analogue of Waters signatures defined in DJ PKE, which is the same as our DCR-based pPRF. Therefore, it is not surprising that our DCR-based ABME scheme requires the same assumptions as the Hofheinz’s ABM-LTF counterpart does. In the latest e-print version [31], Hofheinz has shown that the DCR-based ABM-LTF can be converted to SIM-SO-CCA PKE. To realize this, *an opening algorithm* for ABM-LTF is essentially needed. So, he gave it by sacrificing efficiency. We remark that ABM-LTF *equipped with an opening algorithm* meets the notion of ABME. However, compared to our DCR-based ABME scheme in Sec. 7, Hofheinz’s ABM-LTF-based ABME scheme is less efficient for practical use. Indeed, its expansion rate of ciphertext length per message length is ≥ 31 . In addition, you must use a modulus of $\geq n^6$. On the other hand, our DCR-based ABME scheme has a small expansion rate of $(5 + 1/d)$ and you can use modulus of n^{d+1} for any $d \geq 1$. On the contrary, our DCR-based ABME can be converted to ABM-LTF, and is more efficient than Hofheinz’s ABM-LTF scheme. We compare them in Sec. 8.

2 Preliminaries

For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \dots, n\}$. We denote by O, Ω , and ω the standard notations to classify the growth of functions. We let $\text{negl}(\kappa)$ 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 $\text{Dist}(X_\kappa, Y_\kappa) \triangleq \frac{1}{2} \cdot |\Pr_{s \in \{0, 1\}^\kappa}[X = s] - \Pr_{s \in \{0, 1\}^\kappa}[Y = s]|$. We say that two probability ensembles, X and Y , are statistically indistinguishable (in κ), denoted $X \stackrel{s}{\approx} Y$, if $\text{Dist}(X_\kappa, Y_\kappa) = \text{negl}(\kappa)$. We say that X and Y are computationally indistinguishable (in κ), denoted $X \stackrel{c}{\approx} Y$, if for every (non-uniform) PPT D (ranging over $\{0, 1\}$), it holds that $\{D(1^\kappa, X_\kappa)\}_{\kappa \in \mathbb{N}} \stackrel{s}{\approx} \{D(1^\kappa, Y_\kappa)\}_{\kappa \in \mathbb{N}}$. Let A and B be PPT algorithms that both take $x \in S(\kappa)$, where $S(\kappa)$ is a set associated with each $\kappa \in \mathbb{N}$. We write $\{A(x)\}_{\kappa \in \mathbb{N}, x \in S(\kappa)} \stackrel{s}{\approx} \{B(x)\}_{\kappa \in \mathbb{N}, x \in S(\kappa)}$ to denote $\{A(x_\kappa)\}_{\kappa \in \mathbb{N}} \stackrel{s}{\approx} \{B(x_\kappa)\}_{\kappa \in \mathbb{N}}$ for every sequence $\{x_\kappa\}_{\kappa \in \mathbb{N}}$ such that $x_\kappa \in S(\kappa)$.

3 Building Blocks: Definitions

We now formally define probabilistic pseudo random functions and extractable sigma protocols.

3.1 Probabilistic Pseudo Random Function (pPRF)

pPRF = (KG, Spl) consists of the following two algorithms:

- KG, the key generation algorithm, is a PPT algorithm that takes 1^κ as input, creates (pk, sk) .
- Spl, the sampling algorithm, is a PPT algorithm that takes (pk, sk) and $t \in \{0, 1\}^\kappa$, picks up inner random coins $v \leftarrow \text{COIN}^{\text{spl}}$, and outputs $u = \text{Spl}(pk, sk, t; v)$. We often omit to write pk and instead write this experiment as $u \leftarrow \text{Spl}_{sk}(t)$.

Let $L_{pk}(t) = \{u \mid \exists sk, \exists v : u = \text{Spl}(pk, sk, t; v)\}$, and let $L_{pk} = \{(t, u) \mid t \in \{0, 1\}^\kappa \text{ and } u \in L_{pk}(t)\}$. We assume that pk defines set U_{pk} such that $L_{pk}(t) \subset U_{pk}$ for all $t \in \{0, 1\}^\kappa$. Let $U'_{pk} = \{(t, u) \mid t \in \{0, 1\}^\kappa \text{ and } u \in U_{pk}\}$. We are interested in the case that $L_{pk}(t)$ is so small in U_{pk} , that no one can sample an element from $L_{pk}(t)$ by chance. We require that pPRFs satisfy the following security requirements:

Efficiently samplable and explainable domain: For all pk given by KG and all $t \in \{0, 1\}^\kappa$, U_{pk} is efficiently samplable and explainable [22], that is, there is an PPT sampling algorithm U that takes (pk, t) , picks up random coins R , and outputs u that is uniformly distributed in domain U_{pk} . In addition, for every pk , every $t \in \{0, 1\}^\kappa$, and every $u \in U_{pk}$, there is an efficient explaining algorithm that takes (pk, t) and outputs random coins R behind u , where R is uniformly distributed subject to $U(pk, t; R) = u$.

Pseudo randomness: No adversary A , given pk , can distinguish whether it has access to $\text{Spl}(pk, sk, \cdot)$ or $U(pk, \cdot)$. Here $U(pk, \cdot)$ denotes the uniform sampling algorithm mentioned above. We say that pPRF is pseudo-random if, for all PPT A ,

$$\text{Adv}_{\text{pPRF}, A}^{\text{prf}}(\kappa) = \left| \Pr[\text{Expt}_{\text{pPRF}, A}^{\text{prf}}(\kappa) = 1] - \Pr[\text{Expt}_{U, A}^{\text{prf}}(\kappa) = 1] \right|$$

is negligible in κ , where

$\begin{aligned} &\text{Expt}_{\text{pPRF}, A}^{\text{prf}}(\kappa): \\ &(pk, sk) \leftarrow \text{KG}(1^\kappa) \\ &b \leftarrow A^{\text{Spl}(pk, sk, \cdot)}(pk) \\ &\text{return } b. \end{aligned}$	$\begin{aligned} &\text{Expt}_{U, A}^{\text{prf}}(\kappa): \\ &(pk, sk) \leftarrow \text{KG}(1^\kappa) \\ &b \leftarrow A^{U(pk, \cdot)}(pk) \\ &\text{return } b. \end{aligned}$
--	---

We note that if $\text{Spl}(pk, sk, \cdot)$ is deterministic, we change oracle $U(pk, \cdot)$ as follows: Given fresh t as input, it picks up random R and computes $u = U(pk, t; R)$. It returns u and register (t, u) . Given the same query t , it outputs the same u .

Unforgeability: Let $\widehat{L}_{pk}(t)$ be some superset of $L_{pk}(t)$. Let $\widehat{L}_{pk} = \{(t, u) \mid t \in \{0, 1\}^\kappa \text{ and } u \in \widehat{L}_{pk}(t)\}$. We define the game of unforgeability on \widehat{L}_{pk} as follows: An adversary A takes pk generated by $\text{KG}(1^\kappa)$ and may have access to $\text{Spl}(pk, sk, \cdot)$. The aim of the adversary is to output $(t^*, u^*) \in \widehat{L}_{pk}$ such that t^* has not been queried. We say that pPRF is unforgeable on \widehat{L}_{pk} if, for all PPT A , $\text{Adv}_{\text{pPRF}, A}^{\text{euf-}\widehat{L}}(\kappa) = \Pr[\text{Expt}_{\text{pPRF}, A}^{\text{euf-}\widehat{L}}(\kappa) = 1]$ (where $\text{Expt}_{\text{pPRF}, A}^{\text{euf-}\widehat{L}}$ is defined in Fig. 1) is negligible in κ .

In some application, we require a stronger requirement, where in the same experiment above, it is difficult for the adversary to output (t^*, u^*) in \widehat{L}_{pk} , which did not appear in the query/answer list \mathcal{QA} . We say that pPRF is strongly unforgeable on \widehat{L}_{pk} if, for all PPT A , $\text{Adv}_{\text{pPRF}, A}^{\text{seuf-}\widehat{L}}(\kappa) = \Pr[\text{Expt}_{\text{pPRF}, A}^{\text{seuf-}\widehat{L}}(\kappa) = 1]$ (where $\text{Expt}_{\text{pPRF}, A}^{\text{seuf-}\widehat{L}}$ is defined in Fig. 1) is negligible in κ .

$\begin{aligned} &\text{Expt}_{\text{pPRF}, A}^{\text{euf-}\widehat{L}}(\kappa): \\ &(pk, w) \leftarrow \text{KG}^{\text{pprf}}(1^\kappa) \\ &(t^*, u^*) \leftarrow A^{\text{Spl}(pk, w, \cdot)}(pk) \\ &\text{If } t^* \text{ has not been queried} \\ &\quad \text{and } u^* \in \widehat{L}_{pk}(t^*), \\ &\text{return } 1; \text{ otherwise } 0. \end{aligned}$	$\begin{aligned} &\text{Expt}_{\text{pPRF}, A}^{\text{seuf-}\widehat{L}}(\kappa): \\ &(pk, w) \leftarrow \text{KG}^{\text{pprf}}(1^\kappa) \\ &(t^*, u^*) \leftarrow A^{\text{Spl}(pk, w, \cdot)}(pk) \\ &(t^*, u^*) \notin \mathcal{QA} \\ &\quad \text{and } u^* \in \widehat{L}_{pk}(t^*), \\ &\text{return } 1; \text{ otherwise } 0. \end{aligned}$
--	--

Fig. 1. The experiments of unforgeability (in the left) and strong unforgeability (in the right).

We remark that if Spl is a DPT algorithm and $\widehat{L}_{pk} = L_{pk}$, unforgeability is implied by pseudo randomness.

3.2 Extractable Sigma Protocol

We define extractable sigma protocols. Let $L = \{L_{pk}\}_{pk}$ be an NP language consisting of a collection of set L_{pk} indexed by $pk \in \mathcal{PK}$, where \mathcal{PK} is an infinite sequence of pk . Let R_{pk} be the relation

derived from L_{pk} . Let $\Sigma^{\text{ext}} = (\mathbf{P}_{\Sigma}^{\text{com}}, \mathbf{P}_{\Sigma}^{\text{ans}}, \mathbf{V}_{\Sigma}^{\text{vrfy}}, \text{sim}\mathbf{P}_{\Sigma}^{\text{com}}, \text{Ext})$ be a tuple of algorithms (associated with L) as follows:

- $\mathbf{P}_{\Sigma}^{\text{com}}$ is a PPT algorithm that takes $(x, w) \in R_{pk}$, picks up inner coins r_a , and outputs $a = \mathbf{P}_{\Sigma}^{\text{com}}(x, w; r_a)$.
- $\mathbf{P}_{\Sigma}^{\text{ans}}$ is a DPT algorithm that takes (x, w, r_a, e) and outputs $z = \mathbf{P}_{\Sigma}^{\text{ans}}(x, w, r_a, e)$, where e is an element in a specific domain determined by pk .
- $\mathbf{V}_{\Sigma}^{\text{vrfy}}$ is a DPT algorithm that accepts or rejects (x, a, e, z) .
- $\text{sim}\mathbf{P}_{\Sigma}^{\text{com}}$ is a PPT algorithm that takes (x, e) and outputs $(a, e, z) = \text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; r_z)$, where r_z is inner coins. For our purpose, we additionally require that $r_z = z$, i.e., $(a, e, r_z) = \text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; r_z)$. We note that many sigma protocols satisfy this property.
- Ext is a DPT algorithm that takes (sk, x, a) and outputs e or \perp , where sk is a string with respects to pk .

We say that Σ^{ext} is an extractable sigma protocol on $L = \{L_{pk}\}_{pk}$, if for all pk , there is a set L_{pk}^{co} such that $L_{pk} \cap L_{pk}^{\text{co}} = \emptyset$, and it satisfies the following properties:

Completeness: For every $(x, w) \in R_{pk}$ and every r_a, e (in appropriate specified domains, respectively), it always holds that $\mathbf{V}_{\Sigma}^{\text{vrfy}}(x, \mathbf{P}_{\Sigma}^{\text{com}}(x, w; r_a), e, \mathbf{P}_{\Sigma}^{\text{ans}}(x, w, r_a, e)) = 1$.

Special Soundness: For every $x \notin L$ and every a , there is at most one e such that $\mathbf{V}_{\Sigma}^{\text{vrfy}}(x, a, e, z) = 1$. This implies that if there are two different accepting conversations for the same a on x , i.e., (a, e, z) and (a, e', z') , with $e \neq e'$, it must hold that $x \in L$. We say that such a pair is a *collision* on x . We require for our purpose that there is some superset U' such that $L \subset U'$, and for every $x \in U' \setminus L$ and every e , there is an accepting conversation (a, e, z) on x .

Extractability: We write $(pk, sk^{\text{ext}}) \in R^{\text{ext}}$ if it holds that $\mathbf{V}_{\Sigma}^{\text{vrfy}}(x, a, e', z) = 1$ for all $x \in L_{pk}^{\text{co}}$ and all a so that there are (e, z) such that $\mathbf{V}_{\Sigma}^{\text{vrfy}}(x, a, e, z) = 1$, where $e' = \text{Ext}(sk^{\text{ext}}, x, a)$. We call that Σ^{ext} has extractability on $\{L_{pk}^{\text{co}}\}_{pk}$ if for all $pk \in \mathcal{PK}$, there exists sk^{ext} such that $(pk, sk^{\text{ext}}) \in R^{\text{ext}}$.

We note that, combining with special soundness, we can say that for all $x \in L_{pk}^{\text{co}}$, all e , and all z , it always holds that $e = \text{Ext}(sk, x, \text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; z)_1)$, where $\text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; z)_1$ denotes the first output of $\text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; z)$.

Enhanced Honest-Verifier Statistical Zero-Knowledgeness (eHVSZK): For all $(pk, sk^{\text{ext}}) \in R^{\text{ext}}$, all $(x, w) \in R_{pk}$, and all e in a specific domain, the following ensembles are statistically indistinguishable in κ :

$$\begin{aligned} & \{\text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; r_z)\}_{\kappa \in \mathbb{N}, pk, (x, w) \in R_{pk}, e} \\ & \stackrel{s}{\approx} \{(\mathbf{P}_{\Sigma}^{\text{com}}(x, w; r_a), e, \mathbf{P}_{\Sigma}^{\text{ans}}(x, w, r_a, e))\}_{\kappa \in \mathbb{N}, pk, (x, w) \in R_{pk}, e} \end{aligned}$$

Here the probability of the left-hand side is taken over random variable r_z and the right-hand side is taken over random variable r_a . We remark that since $(a, e, r_z) = \text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; r_z)$ (by our precondition), we have $\mathbf{V}_{\Sigma}^{\text{vrfy}}(x, a, e, z) = 1$ if and only if $(a, e, z) = \text{sim}\mathbf{P}_{\Sigma}^{\text{com}}(x, e; z)$. Therefore, one can instead use $\text{sim}\mathbf{P}_{\Sigma}^{\text{com}}$ to verify (a, e, z) on x .

We note that the concept of the extractable sigma protocol is not entirely new. A weaker notion, called *weak* extractable sigma protocol, appears in [25] to construct (interactive) simulation sound trapdoor commitment (SSTC) schemes (see [27, 38, 28] for SSTC). This paper requires a stronger notion, which is used in a different way.

4 ABM Encryption

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

- ABM.gen is a PPT algorithm that takes 1^κ and outputs $(pk, sk^{\text{spl}}, sk^{\text{ext}})$, where pk defines a universe $U'_{pk} = \{0, 1\}^\kappa \times U_{pk}$, which contains two disjoint sets (as defined below), L_{pk}^{td} and L_{pk}^{ext} , i.e., $L_{pk}^{\text{td}} \cap L_{pk}^{\text{ext}} = \emptyset$ and $L_{pk}^{\text{td}} \cup L_{pk}^{\text{ext}} \subset U'_{pk}$.
- ABM.spl is a PPT algorithm that takes (pk, sk^{spl}, t) , where $t \in \{0, 1\}^\kappa$, picks up inner random coins $v \leftarrow \text{COIN}^{\text{spl}}$, and computes $u \in U_{pk}$. We let

$$L_{pk}^{\text{td}}(t) = \{u \in U_{pk} \mid \exists sk^{\text{spl}}, \exists v : u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)\}.$$

We let $L_{pk}^{\text{td}} = \{(t, u) \mid t \in \{0, 1\}^\kappa \text{ and } u \in L_{pk}^{\text{td}}(t)\}$. Define $\widehat{L}_{pk}^{\text{td}} = U'_{pk} \setminus L_{pk}^{\text{ext}}$. Since $L_{pk}^{\text{td}} \cap L_{pk}^{\text{ext}} = \emptyset$, we have $L_{pk}^{\text{td}} \subseteq \widehat{L}_{pk}^{\text{td}} \subset U'_{pk}$.

- ABM.enc is a PPT algorithm that takes pk , $(t, u) \in U'_{pk}$, and message $x \in \text{MSP}$, picks up inner random coins $r \leftarrow \text{COIN}^{\text{enc}}$, and computes $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$.
- ABM.dec is a DPT algorithm that takes sk^{ext} , (t, u) , and ciphertext c , and outputs $x = \text{ABM.dec}^{(t,u)}(sk^{\text{ext}}, c)$.
- $\text{ABM.col} = (\text{ABM.col}_1, \text{ABM.col}_2)$ is a pair of PPT and DPT algorithms, respectively, such that
 - ABM.col_1 takes $(pk, (t, u), sk^{\text{spl}}, v)$ and outputs $(c, \xi) \leftarrow \text{ABM.col}_1^{(t,u)}(pk, sk^{\text{spl}}, v)$, where $v \in \text{COIN}^{\text{spl}}$.
 - ABM.col_2 takes $((t, u), \xi, x)$, with $x \in \text{MSP}$, and outputs $r \in \text{COIN}^{\text{enc}}$.

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

1. **Adaptive All-but-many property:** $(\text{ABM.gen}, \text{ABM.spl})$ is a probabilistic pseudo random function (pPRF) as defined in Sec. 3.1 with unforgeability on $\widehat{L}_{pk}^{\text{td}} (= U'_{pk} \setminus L_{pk}^{\text{ext}})$.
2. **Dual mode property:**
 - **(Decryption mode)** For all $\kappa \in \mathbb{N}$, all $(pk, sk^{\text{ext}}) \in \text{ABM.gen}(1^\kappa)$, all $(t, u) \in L_{pk}^{\text{ext}}$, and every $x \in \text{MSP}$, it always holds that

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

- **(Trapdoor mode)** Define the following random variables:
 - $\text{dist}^{\text{enc}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)$ denotes random variable (pk, t, u, c, r) defined as follows: $v \leftarrow \text{COIN}^{\text{spl}}$; $u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)$; $r \leftarrow \text{COIN}^{\text{enc}}$; $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$.
 - $\text{dist}^{\text{col}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)$ denotes random variable (pk, t, u, c, r) defined as follows: $v \leftarrow \text{COIN}^{\text{spl}}$; $u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)$; $(c, \xi) \leftarrow \text{ABM.col}_1^{(t,u)}(pk, sk^{\text{spl}}, v)$; $r = \text{ABM.col}_2^{(t,u)}(\xi, x)$.

Then, for all $(pk, sk^{\text{spl}}, sk^{\text{ext}}) \in \text{ABM.gen}(1^\kappa)$, all $t \in \{0, 1\}^\kappa$, all $x \in \text{MSP}$, the following ensembles are statistically indistinguishable in κ :

$$\left\{ \text{dist}^{\text{enc}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x) \right\}_{\kappa \in \mathbb{N}, (pk, sk^{\text{spl}}, sk^{\text{ext}}) \in \text{ABM.gen}(1^\kappa), t \in \{0, 1\}^\kappa, x \in \text{MSP}}$$

$$\stackrel{s}{\approx} \left\{ \text{dist}^{\text{col}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x) \right\}_{\kappa \in \mathbb{N}, (pk, sk^{\text{spl}}, sk^{\text{ext}}) \in \text{ABM.gen}(1^\kappa), t \in \{0, 1\}^\kappa, x \in \text{MSP}}$$

We say that a ciphertext c on (t, u) under pk is *valid* if there exist $x \in \text{MSP}$ and $r \in \text{COIN}^{\text{enc}}$ such that $c = \text{ABM.enc}^{(t, u)}(pk, x; r)$. We say that a valid ciphertext c on (t, u) under pk is *real* if $(t, u) \in L_{pk}^{\text{ext}}$, otherwise *fake*. We remark that as long as c is a real ciphertext, regardless of how it is generated, there is only one consistent x in MSP and it is equivalent to $\text{ABM.dec}^{(t, u)}(sk, c)$.

To suit actual instantiations, we assume that COIN^{spl} and MSP are defined by pk . We further allow COIN^{enc} to depend on message x to be encrypted as well as pk , in order to be consistent with our weak ABM encryption scheme from general assumption in Sec. 10.

5 ABME from Σ^{ext} on Language derived from pPRF

Suppose a probabilistic pseudo random function pPRF = (KG, Spl) and an extractable sigma protocol $\Sigma^{\text{ext}} = (\Sigma, \text{Ext})$ are well-combined. That is,

- KG(1^κ) outputs $(pk, sk^{\text{spl}}, sk^{\text{ext}})$. Later, sk^{spl} is used as a secret key of Spl and sk^{ext} is used as a secret key of Ext.
- For all pk , there is a set L_{pk}^{co} such that $L_{pk} \cap L_{pk}^{\text{co}} = \emptyset$, where $L_{pk} = \{(t, u) \mid \exists (sk^{\text{spl}}, v) : u = \text{Spl}(pk, sk^{\text{spl}}, t; v)\}$.
- Σ^{ext} is an extractable sigma protocol on L_{pk} and has extractability on L_{pk}^{co} where sk^{ext} is the extractable key.
- pPRF is unforgeable on $\widehat{L}_{pk} := U'_{pk} \setminus L_{pk}^{\text{co}}$, where U'_{pk} is a universe (with respects to pk).

Here, an ABME scheme is constructed as in Fig. 2.

- ABM.gen(1^κ) runs KG(1^κ) to output $(pk, sk^{\text{spl}}, sk^{\text{ext}})$.
- ABM.spl($pk, sk^{\text{spl}}, t; v$) outputs $u = \text{Spl}(pk, sk^{\text{spl}}, t; v)$.
- ABM.enc $^{(t, u)}(pk, m; r)$ outputs a such that $(a, m, r) = \text{simP}_{\Sigma^{\text{com}}}(pk, (t, u), m; r)$.
- ABM.dec $^{(t, u)}(sk^{\text{ext}}, c)$ outputs $m = \text{Ext}(sk^{\text{ext}}, (t, u), c)$.
- ABM.col $_1^{(t, u)}(pk, sk^{\text{spl}}, v; r_a)$ outputs (c, ξ) such that $c = \text{P}_{\Sigma^{\text{com}}}^{\text{com}}(pk, (t, u), (sk^{\text{spl}}, v); r_a)$, and $\xi = (pk, t, u, sk^{\text{spl}}, v, r_a)$.
- ABM.col $_2^{(t, u)}(\xi, m)$ outputs $r = \text{P}_{\Sigma^{\text{ans}}}^{\text{ans}}(pk, (t, u), sk^{\text{spl}}, v, r_a, m)$, where $\xi = (pk, t, u, sk^{\text{spl}}, v, r_a)$.

Fig. 2. ABME from Σ^{ext} on language derived from pPRF

By construction, the adaptive all-but-many property holds. The dual mode property also holds because:

- If $(t, u) \in L_{pk}^{\text{ext}}$, the first output of $\text{simP}_{\Sigma}^{\text{com}}(pk, (t, u), m)$ is perfectly binding to challenge m due to special soundness (because $L_{pk}^{\text{ext}} \subset U'_{pk} \setminus L_{pk}^{\text{td}}$, with $L_{pk}^{\text{td}} := L_{pk}$), and m can be extracted given $(pk, (t, u), a)$ using sk^{ext} due to extractability.
- If $(t, u) \in L_{pk}^{\text{td}}$, ABM.col runs the real sigma protocol with witness (sk^{sp1}, 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 $\text{simP}_{\Sigma}^{\text{com}}$ (that is run by ABM.enc), due to enhanced HVSZK. We note that even given the same (fixed) sk^{ext} to both algorithms, it does not affect the statistical distance, because it is fixed.

Hence, the resulting scheme is ABME.

6 Fully-Equipped UC Commitment from ABME

We show that ABME implies fully-equipped UC commitment.

We work in the standard universal composability (UC) framework of Canetti [10]. We concentrate on the same model in [11] 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, denoted $\mathcal{F}_{\text{MCOM}}$, in Appendix B.1.

To construct fully-equipped UC commitment, we first put public key pk of ABME in the common reference string. A committer P_i takes tag $t = (\text{sid}, \text{ssid}, P_i, P_j)$ and a message x committed to. It then picks up random u from U_{pk} and compute an ABM encryption $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$ to send (t, u, c) to receiver P_j , which outputs $(\text{receipt}, \text{sid}, \text{ssid}, P_i, P_j)$. To open the commitment, P_i sends $(\text{sid}, \text{ssid}, x, r)$ to P_j and P_j accepts if and only if $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$. If P_j accepts, he outputs (open, t, x) , otherwise do nothing. We formally describe our framework for constructing a UC commitment scheme from ABME in Fig. 3.

Theorem 1. *The proposed scheme in Fig.3 UC-securely realizes the $\mathcal{F}_{\text{MCOM}}$ functionality in the \mathcal{F}_{CRS} -hybrid model in the presence of adaptive adversaries in the non-erasure model.*

Proof (Sketch). For simplicity, we remove the injective map $\iota : \{0, 1\}^k \rightarrow \text{MSP}$ from the scheme. The formal proof is given in Appendix B.2. We here sketch the essence. We consider the man-in-the-middle attack, where we will show that the view of environment \mathcal{Z} in the real world (in the CRS model) can be simulated in the ideal world. Let C, R be honest players and let P_a be a corrupted player controlled by adversary \mathcal{A} . In the man-in-the-middle attack, P_a (i.e., \mathcal{A}) is simultaneously participating in the left and right interactions. In the left interaction, \mathcal{A} interacts with C , as playing the role of the receiver. In the right interaction, \mathcal{A} interacts with R , as playing the role of the committer. In the ideal world, simulator \mathcal{S} simulates the task of C and R by interacting with \mathcal{A} .

In the left interaction: In the real world, \mathcal{Z} chooses $(\text{commit}, \text{sid}, \text{ssid}, C, P_a, x)$ and gives it to C to start the commitment protocol with \mathcal{A} . However, in the ideal world \mathcal{S} cannot receive x until the decommit phase, but must start the commitment protocol only with $t = (\text{sid}, \text{ssid}, C, P_a)$. At the decommit phase, \mathcal{S} receives x for the first time and needs to open to x correctly.

More precisely, in both worlds, \mathcal{Z} sends $(\text{commit}, \text{sid}, \text{ssid}, C, P_a, x)$ to C , but in the ideal world C simply conveys it from \mathcal{Z} to $\mathcal{F}_{\text{MCOM}}$. Then, $\mathcal{F}_{\text{MCOM}}$ sends $(\text{receipt}, \text{sid}, \text{ssid}, C, P_a)$ to \mathcal{S} so that

Common reference string: pk where $(pk, sk) \leftarrow \text{ABM.gen}(1^\kappa)$.

pk uniquely determines $U'_{pk} = \{0, 1\}^\kappa \times U_{pk}$. We implicitly assume that there is injective map $\iota : \{0, 1\}^\kappa \rightarrow \text{MSP}$ such that ι^{-1} is efficiently computable and $\iota^{-1}(y) = \varepsilon$ for every $y \notin \iota(\{0, 1\}^\kappa)$, and also assume that $(\text{sid}, \text{ssid}, P_i, P_j) \in \{0, 1\}^\kappa$.

The commitment phase:

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

The decommitment phase:

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

Fig. 3. Fully-Equipped UC commitment from ABME

\mathcal{S} can start the commit phase with \mathcal{A} (without given x). In both worlds, \mathcal{Z} sends $(\text{open}, \text{sid}, \text{ssid})$ to activate C to start the decommit phase, but in the ideal world C simply sends it to $\mathcal{F}_{\text{MCOM}}$, which sends $(\text{reveal}, \text{sid}, \text{ssid}, C, P_a, x)$ to \mathcal{S} so that \mathcal{S} can start the decommit protocol with x with \mathcal{A} .

In the right interaction: In the real world, \mathcal{Z} receives $(\text{open}, \text{sid}', \text{ssid}', P_a, R, x')$ opened by \mathcal{A} from R at the decommit phase. In the ideal world, \mathcal{S} must correctly extract \tilde{x} from (t', u', c') sent by \mathcal{A} , where $t' = (\text{sid}', \text{ssid}', P_a, R)$, and commit it to the ideal commitment functionality $\mathcal{F}_{\text{MCOM}}$ at the commit phase. At the decommit phase, when \mathcal{A} correctly opens the commitment, \mathcal{S} must let $\mathcal{F}_{\text{MCOM}}$ reveal stored \tilde{x} to \mathcal{Z} , instead of the value that \mathcal{A} actually opened to.

More precisely, in the ideal world, when receiving $(\text{open}, \text{sid}', \text{ssid}')$ (from \mathcal{S}), $\mathcal{F}_{\text{MCOM}}$ sends $(\text{reveal}, \text{sid}', \text{ssid}', P_a, R, \tilde{x})$ to R , where \tilde{x} is the stored value at the commit phase. R simply conveys it from $\mathcal{F}_{\text{MCOM}}$ to \mathcal{Z} .

Adaptive corruption: In the real world, when C or R is corrupted, \mathcal{A} may read their inner state and start to fully control the parties. In the ideal world, the honest parties do nothing except storing inputs to them. So, \mathcal{S} simulates the inner state of the real-world honest party (after \mathcal{S} read the inner state of the ideal-world honest party when it is corrupted) and gives it to \mathcal{A} as if it comes from the real world. The inner state of the real-world honest party includes randomness it has used. In the non-erasure model, honest parties cannot erase any of their state.

The view of \mathcal{Z} : In the real world, \mathcal{Z} have access to \mathcal{A} to order many tasks, for instance, to execute the right interaction with R with value x' , to corrupt either party, or to send the adversary's entire view in the left and right interactions. In the ideal world, \mathcal{Z} instead have access to (the ideal-world adversary) \mathcal{S} , which tries to simulate the role of \mathcal{A} . The view of \mathcal{Z} consists of each interaction with C , R , and the (real-world or ideal-world) adversary, as well as its inner state.

As usual, we consider a sequence of hybrid games on which the probability spaces are identical, but we change the rules of games step by step. See Table 2 for summary.

Ideal World: In the Ideal world, \mathcal{A} interacts with simulator \mathcal{S} in both interactions, where \mathcal{S} simulates the roles of C and R respectively. In the set-up, \mathcal{S} generates $(pk, sk^{\text{spl}}, sk^{\text{ext}}) \leftarrow \text{ABM.gen}(1^\kappa)$, puts pk in the common reference string, and keeps $(sk^{\text{spl}}, sk^{\text{ext}})$. In the left interaction, \mathcal{S} first receives $(\text{receipt}, \text{sid}, \text{ssid}, C, P_a)$ and starts the commitment phase with adversary \mathcal{A} as the committer without given message x . \mathcal{S} computes $u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)$ and $(c, \xi) \leftarrow \text{ABM.col}_1^{(t,u)}(pk, sk^{\text{spl}}, v)$, to send $(\text{commit}, t, (u, c))$ to adversary \mathcal{A} , where $t = (\text{sid}, \text{ssid}, C, P_a)$. At the decommit phase, \mathcal{S} receives $(\text{reveal}, \text{sid}, \text{ssid}, C, P_a, x)$ and then computes $r = \text{ABM.col}_2^{(t,u)}(\xi, x)$ to send (t, x, r) to \mathcal{A} . In the right interaction, \mathcal{S} receives $(\text{commit}, t', u', c')$ from \mathcal{A} where $t' = (\text{sid}', \text{ssid}', P_a, R)$. \mathcal{S} then extracts $\tilde{x} = \text{ABM.dec}^{(t',u')}(sk, c')$ and sends $(\text{commit}, t', \tilde{x})$ to $\mathcal{F}_{\text{MCOM}}$. At the decommit phase when \mathcal{A} opens (t', u', c') correctly with (x', r') , \mathcal{S} sends $(\text{open}, \text{sid}, \text{ssid})$ to $\mathcal{F}_{\text{MCOM}}$; otherwise, do nothing. Upon receiving $(\text{open}, \text{sid}, \text{ssid})$, if the same $(\text{sid}, \text{ssid}, \dots)$ was previously recorded, $\mathcal{F}_{\text{MCOM}}$ reveals *stored* \tilde{x} to environment \mathcal{Z} ; otherwise, do nothing.

In case of adaptive corruption of C after the commit phase but before the decommit phase, \mathcal{S} read x from the inner state of C and computes r as in the case of the decommit phase and compute R such that $U_{pk}(t; R) = u$, which can be efficiently computable because U_{pk} is an explainable domain. Finally it reveals (x, r, R) .

Hybrid Game 1: In this game, the left interaction is modified so that \mathcal{S} instead receives (commit, t, x) where $t = (\text{sid}, \text{ssid}, C, P_a)$. \mathcal{S} then computes $u \leftarrow \text{ABM.spl}(pk, sk^{\text{spl}}, t)$ and $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$ where $r \leftarrow \text{COIN}^{\text{enc}}$, to send (commit, t, u, c) to adversary \mathcal{A} . In the decommit phase when \mathcal{S} receives (open, t) , it sends (t, x, r) to \mathcal{A} .

In case of adaptive corruption of C after the commit phase but before the decommit phase, \mathcal{S} outputs (t, u, x, r, R) after computing R such that $U_{pk}(t; R) = u$.

The view of \mathcal{Z} in this game is statistically close to that in the ideal world, because

$$\{\text{dist}^{\text{col}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)\}_{\kappa \in \mathbb{N}}$$

and

$$\{\text{dist}^{\text{enc}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)\}_{\kappa \in \mathbb{N}},$$

defined in Sec. 4, are statistically indistinguishable in κ .

Hybrid Game 2: In this game, the right interaction is changed as follows. After receiving (t', u', c') , where $t' = (\text{sid}', \text{ssid}', P_a, R)$, \mathcal{S} sends $(\text{commit}, t', \varepsilon)$ to the ideal functionality. In the decommit phase when \mathcal{A} opens (t', u', c') correctly with (x', r') , \mathcal{S} sends $(\text{open}, \text{sid}', \text{ssid}', x')$ to the ideal functionality. Then, the ideal functionality reveals x' (instead of ε) to \mathcal{Z} .

In case of corruption of R before the decommit phase, \mathcal{S} simply outputs (t', u', c') . We note that R has no secret.

The difference of the views of \mathcal{Z} between this game and the previous game is bounded by the following event. Let BD denote the event that \mathcal{S} receives a *fake* ciphertext (t', u', c') from \mathcal{A} in the right intersection. Remember that ciphertext c is called fake if $(t, u) \in L_{pk}^{\text{td}}$ and c is a *valid* ciphertext (which means that there is a pair of message/randomness consistent with c). If this event does not occur, the views of \mathcal{Z} in both games are identical. Hence, the difference of the views of \mathcal{Z} between the two games is bounded by $\Pr[\text{BD}]$. Event BD occurs (in Hybrid Game 2) if and only if \mathcal{A} breaks unforgeability of $(\text{ABM.gen}, \text{ABM.spl})$ on $\widehat{L}_{pk}^{\text{td}}$. Therefore, $\Pr[\text{BD}]$ is negligible in κ .

Hybrid Game 3: In this game, the left interaction is modified again. At the commit phase, when receiving input (commit, t, x) where $t = (\text{sid}, \text{ssid}, C, P_a)$, \mathcal{S} chooses random $u = U_{pk}(t; R)$ with random R and computes $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$, to send (t, u, c) to \mathcal{A} . At the decommit phase, upon receiving input $(\text{open}, \text{sid}, \text{ssid})$, \mathcal{S} plays the same as in the previous game.

In case of corruption of C before the decommit phase, \mathcal{S} simply reveals (x, r, R) (where $u = U_{pk}(t; R)$).

By construction, the difference of the two views of \mathcal{Z} between this game and the previous game is bounded by the advantage of pseudo-randomness of $\text{pPRF} = (\text{ABM.gen}, \text{ABM.spl})$.

Hybrid $^{\mathcal{F}_{\text{crs}}}$ Game: It corresponds to the real world in the CRS model, where \mathcal{A} interacts with honest C and R respectively, and executes the man-in-the-middle attack. In the left interaction, environment \mathcal{Z} activates C to start the commit phase by sending (commit, t, x) to C where $t = (\text{sid}, \text{ssid}, C, P_a)$. \mathcal{Z} activates C to start the decommit phase by sending $(\text{open}, \text{sid}, \text{ssid})$ to C . In the right interaction, at the commit phase when R receives (t', u', c') from \mathcal{A} , it outputs $(\text{receipt}, t')$ to \mathcal{Z} where $t' = (\text{sid}', \text{ssid}', P_a, R)$. At the decommit phase, upon receiving $(\text{sid}', \text{ssid}', x', r')$ from \mathcal{A} , R checks its consistency with (t', u', c') . If the opening is correct, it outputs (reveal, t', x') to \mathcal{Z} .

By construction, the two views of \mathcal{Z} between this game and the previous game are identical.

■

Table 2. The man-in-the-middle attack in the hybrid games

Games	$C(\mathcal{S})$	$P_a(\mathcal{A})$	$R(\mathcal{S})$	$\mathcal{F}_{\text{MCOM}}$
IDEAL	$u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)$ $(c, \xi) \leftarrow \text{ABM.col}_1^{(t,u)}(pk, sk^{\text{spl}}, v)$ $r = \text{ABM.col}_2^{(t,u)}(\xi, x)$		Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{commit}, t', \tilde{x})$ s.t. $\tilde{x} = \text{ABM.dec}^{(t',u')}(sk^{\text{ext}}, c')$. Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{open}, \text{sid}', \text{ssid}')$ if $c' = \text{ABM.enc}^{(t',u')}(pk, x'; r')$.	\tilde{x}
HYBRID ¹	$u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)$ $r \leftarrow \text{COIN}^{\text{enc}}$ $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$		Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{commit}, t', \tilde{x})$ s.t. $\tilde{x} = \text{ABM.dec}^{(t',u')}(sk^{\text{ext}}, c')$ Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{open}, \text{sid}', \text{ssid}')$ if $c' = \text{ABM.enc}^{(t',u')}(pk, x'; r')$.	\tilde{x}
HYBRID ²	$u \leftarrow \text{ABM.spl}(pk, sk^{\text{spl}}, t)$ $r \leftarrow \text{COIN}^{\text{enc}}$ $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$		Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{commit}, t', \varepsilon)$. Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{open}, \text{sid}', \text{ssid}', x')$ if $c' = \text{ABM.enc}^{(t',u')}(pk, x'; r')$.	x'
HYBRID ³	$u \leftarrow U_{pk}(t)$ $r \leftarrow \text{COIN}^{\text{enc}}$ $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$		Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{commit}, t', \varepsilon)$. Send to $\mathcal{F}_{\text{MCOM}}$ $(\text{open}, \text{sid}', \text{ssid}', x')$ if $c' = \text{ABM.enc}^{(t',u')}(pk, x'; r')$.	x'
HYBRID $^{\mathcal{F}_{\text{crs}}}$ (Real world)	$u \leftarrow U_{pk}(t)$ $r \leftarrow \text{COIN}^{\text{enc}}$ $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$		Output x' to \mathcal{Z} if $c' = \text{ABM.enc}^{(t',u')}(pk, x'; r')$.	-

In the commit phase (of the left interaction of \mathcal{A}), committer C sends (t, u, c) to corrupted party $P_a(\mathcal{A})$, where $t = (\text{sid}, \text{ssid}, C, P_a)$. In the decommit phase, C opens (x, r) such that $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$. In the commit phase (of the right interaction of \mathcal{A}), corrupted $P_a(\mathcal{A})$ sends (t', u', c') to receiver R , where $t' = (\text{sid}', \text{ssid}', P_a, R)$. In the decommit phase, it opens (x', r') . Upon receiving $(\text{open}, \text{sid}', \text{ssid}')$, $\mathcal{F}_{\text{MCOM}}$ reveals the value in the entry to environment \mathcal{Z} .

7 Compact ABME from Damgård-Jurik PKE

We present a DCR-based ABME scheme with compact ciphertexts. This is the first fully-equipped UC commitment scheme with expansion factor $O(1)$. We start by recalling Damgård-Jurik public-key encryption scheme (DJ PKE) [19].

Damgård-Jurik PKE. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a tuple of algorithms of Damgård-Jurik (DJ) PKE [19]. A public key of DJ PKE is $pk_{\text{dj}} = (n, d)$ and the corresponding secret-key is $sk_{\text{dj}} = (p, q)$ where $n = pq$ is a composite number of distinct odd primes, p and q , and $1 \leq d < p, q$ is a positive integer (when $d = 1$ it is Paillier PKE [40]). We often write $\Pi^{(d)}$ to clarify parameter d . We let $g := (1+n)$ throughout this paper. To encrypt message $x \in \mathbb{Z}_{n^d}$, one computes $\mathbf{E}_{pk_{\text{dj}}}(x; R) = g^x R^{n^d} \pmod{n^{d+1}}$ where $R \leftarrow \mathbb{Z}_n^{\times -1}$. For simplicity, we write $\mathbf{E}(x)$ instead of $\mathbf{E}_{pk_{\text{dj}}}(x)$, if it is clear. DJ PKE is enhanced additively homomorphic as defined in Appendix C.3. Namely, for every $x_1, x_2 \in \mathbb{Z}_{n^d}$ and every $R_1, R_2 \in \mathbb{Z}_n^{\times}$, one can efficiently compute R such that $\mathbf{E}(x_1 + x_2; R) = \mathbf{E}(x_1; R_1) \cdot \mathbf{E}(x_2; R_2)$. Actually it can be done by computing $R = g^\gamma R_1 R_2 \pmod{n}$, where γ is an integer such that $x_1 + x_2 = \gamma n^d + (x_1 + x_2) \pmod{n^d}$. It is known that $\mathbb{Z}_{n^{d+1}}^{\times}$ is isomorphic to $\mathbb{Z}_{n^d} \times \mathbb{Z}_n^{\times}$ (the product of a cyclic group of order n^d and a group of order $\phi(n)$), and, for any $d < p, q$, element $g = (1+n)$ has order n^d in $\mathbb{Z}_{n^{d+1}}^{\times}$ [19]. Therefore, $\mathbb{Z}_{n^{d+1}}^{\times}$ is the image of $\mathbf{E}(\cdot; \cdot)$. We note that it is known that $\mathbb{Z}_{n^{d+1}}^{\times}$ is efficiently samplable and explainable [20, 22]. It is also known that DJ PKE is IND-CPA if the DCR assumption holds true [19].

Construction Idea. (ABM.gen, ABM.sp) below forms Waters-like signature scheme based on DJ PKE, where there is no verification algorithm and the signatures look pseudo random assuming that DJ PKE is IND-CPA. We then construct an extractable sigma protocol on the language derived from (ABM.gen, ABM.sp), as discussed in Sec. 1.4. Here, the decryption algorithm works only when the matrix below in (2) is invertible, which is equivalent to that $(t, (u_r, u_t)) \in L_{pk}^{\text{ext}}$, where

$$L_{pk}^{\text{ext}} = \{(t, (u_r, u_t)) \mid \mathbf{D}(u_t) \not\equiv x_1 x_2 + y(t) \mathbf{D}(u_r) \pmod{p} \quad \wedge \quad \mathbf{D}(u_t) \not\equiv x_1 x_2 + y(t) \mathbf{D}(u_r) \pmod{q}\}.$$

Therefore, we require that (ABM.gen, ABM.sp) should be unforgeable on $\widehat{L}_{pk}^{\text{td}} (= U'_{pk} \setminus L_{pk}^{\text{ext}})$. To prove this statement, we additionally require two more assumptions on DJ PKE, called the **non-multiplication assumption** and the **non-trivial divisor assumption**. The first one is an analogue of the DH assumption in an additively homomorphic encryption. If we consider unforgeability on L_{pk}^{td} , this assumption suffices, but we require unforgeability on $\widehat{L}_{pk}^{\text{td}}$. Then we need one more assumption. We define these assumptions in Appendix C. We note that these assumptions are originally introduced in [31] to obtain the DCR-based ABM-LTF scheme.

7.1 ABME from Damgård-Jurik with Optimal Expansion Factor $O(1)$

- **ABM.gen(1^κ):** It gets $(pk_{\text{dj}}, sk_{\text{dj}}) \leftarrow \mathbf{K}(1^\kappa)$ (the key generation algorithm for DJ PKE), where $pk_{\text{dj}} = (n, d)$ and $sk_{\text{dj}} = (p, q)$. It computes $g_1 = \mathbf{E}(x_1; R_1)$ and $g_2 = \mathbf{E}(x_2; R_2)$ by picking up randomly $x_1, x_2 \leftarrow \mathbb{Z}_{n^d}$ and $R_1, R_2 \leftarrow \mathbb{Z}_{n^{d+1}}^{\times}$. It chooses $\tilde{h} \leftarrow \mathbf{E}(1)$ and $\mathbf{y} = (y_0, \dots, y_\kappa)$

¹ In the original scheme, R is chosen from $\mathbb{Z}_{n^{d+1}}^{\times}$. However, since \mathbb{Z}_n^{\times} is isomorphic to the cyclic group of order n^d in $\mathbb{Z}_{n^{d+1}}^{\times}$ by mapping $R \in \mathbb{Z}_n^{\times}$ to $R^{n^d} \in \mathbb{Z}_{n^{d+1}}^{\times}$, we can instead choose R from \mathbb{Z}_n^{\times} .

where $y_j \leftarrow \mathbb{Z}_{n^{d+1}}$ for $j = 0, 1, \dots, \kappa$. It then computes $\mathbf{h} = (h_0, \dots, h_\kappa)$ such that $h_j := \tilde{h}^{y_j}$. Let $H(t) = h_0 \prod_{i=1}^{\kappa} h_i^{t_i} \pmod{n^{d+1}}$ and let $y(t) = y_0 + \sum_{i=1}^{\kappa} y_i t_i \pmod{n^d}$, where (t_0, \dots, t_κ) is the bit representation of t . We note that $H(t) = \tilde{h}^{y(t)}$. It outputs $(pk, sk^{\text{spl}}, sk^{\text{ext}})$ where $pk := (n, d, g_1, g_2, \mathbf{h})$, $sk^{\text{spl}} := x_2$, and $sk^{\text{ext}} := (p, q, y_0, \mathbf{y})$, where $U'_{pk} := \{0, 1\}^\kappa \times (\mathbb{Z}_{n^{d+1}}^\times)^2$ that contains the disjoint sets of L_{pk}^{td} and L_{pk}^{ext} as described below.

- **ABM.spl** $(pk, sk^{\text{spl}}, t; (r, R_r, R_t))$ where $sk^{\text{spl}} = x_2$: It chooses $r \leftarrow \mathbb{Z}_{n^d}$ and outputs $u := (u_r, u_t)$ such that $u_r := \mathbf{E}(r; R_r)$ and $u_t := g_1^{x_2} \mathbf{E}(0; R_t) \cdot H(t)^r$ where $R_r, R_t \leftarrow \mathbb{Z}_{n^{d+1}}^\times$. We let

$$L_{pk}^{\text{td}} = \{(t, (u_r, u_t)) \mid \exists (x_2, (r, R_r, R_t)) : u_r = \mathbf{E}(r; R_r) \text{ and } u_t = g_1^{x_2} \mathbf{E}(0; R_t) H(t)^r\}.$$

We then define

$$L_{pk}^{\text{ext}} = \{(t, (u_r, u_t)) \mid \mathbf{D}(u_t) \not\equiv x_1 x_2 + y(t) \mathbf{D}(u_r) \pmod{p} \wedge \mathbf{D}(u_t) \not\equiv x_1 x_2 + y(t) \mathbf{D}(u_r) \pmod{q}\}.$$

Since $(t, (u_r, u_t)) \in L_{pk}^{\text{td}}$ holds if and only if $\mathbf{D}(u_t) \equiv x_1 x_2 + y(t) \mathbf{D}(u_r) \pmod{n^d}$, it implies that $\mathbf{D}(u_t) \equiv x_1 x_2 + y(t) \mathbf{D}(u_r) \pmod{n}$. Hence, $L_{pk}^{\text{td}} \cap L_{pk}^{\text{ext}} = \emptyset$.

- **ABM.enc** $^{(t, (u_r, u_t))}(pk, m; (z, s, R_A, R_a, R_b))$: To encrypt message $m \in \mathbb{Z}_{n^d}$, it chooses $z, s \xleftarrow{\cup} \mathbb{Z}_{n^d}$ and computes $A := g_1^z H(t)^s u_t^m R_A^{n^d} \pmod{n^{d+1}}$, $a := \mathbf{E}(z; R_a) \cdot g_2^m \pmod{n^{d+1}}$ and $b := \mathbf{E}(s; R_b) \cdot u_r^m \pmod{n^{d+1}}$, where $R_A, R_a, R_b \xleftarrow{\cup} \mathbb{Z}_{n^{d+1}}^\times$. It outputs $c := (A, a, b)$ as the ciphertext of m on $(t, (u_r, u_t))$.
- **ABM.dec** $^{(t, (u_r, u_t))}(sk^{\text{ext}}, c)$ where $sk^{\text{ext}} = (p, q, y_0, \dots, y_\kappa)$: To decrypt $c = (A, a, b)$, it outputs

$$m := \frac{x_1 \mathbf{D}(a) + y(t) \mathbf{D}(b) - \mathbf{D}(A)}{x_1 x_2 - (\mathbf{D}(u_t) - y(t) \mathbf{D}(u_r))} \pmod{n^d}. \quad (1)$$

- **ABM.col** $_1^{(t, (u_r, u_t))}(pk, sk^{\text{spl}}, (r, R_r, R_t))$ where $sk^{\text{spl}} = x_2$: It picks up $\omega, \eta \xleftarrow{\cup} \mathbb{Z}_{n^d}$, $R'_A, R'_a, R'_b \xleftarrow{\cup} \mathbb{Z}_{n^{d+1}}^\times$. It then computes $A := g_1^\omega \cdot H(t)^\eta \cdot R'_A^{n^d} \pmod{n^{d+1}}$, $a := g^\omega R'_a^{n^d} \pmod{n^{d+1}}$, and $b := g^\eta R'_b^{n^d} \pmod{n^{d+1}}$. It outputs $c := (A, a, b)$ and $\xi := (x_2, (r, R_r, R_t), (u_r, u_t), \omega, \eta, R'_A, R'_a, R'_b)$.
- **ABM.col** $_2(\xi, m)$: To open c to m , it computes $z = \omega - mx_2 \pmod{n^d}$, $s = \eta - mr \pmod{n^d}$, $\alpha = \lfloor (\omega - mx_2 - z)/n^d \rfloor$, and $\beta = \lfloor (\eta - mr - s)/n^d \rfloor$. It then sets $R_A := R'_A \cdot R_t^{-m} \cdot g_1^\alpha \cdot H(t)^\beta \pmod{n^{d+1}}$, $R_a := R'_a \cdot R_2^{-m} \cdot g^\alpha \pmod{n^{d+1}}$, and $R_b := R'_b \cdot R_r^{-m} \cdot g^\beta \pmod{n^{d+1}}$. It outputs (z, s, R_A, R_a, R_b) , where $A = g_1^z H(t)^s u_t^m R_A^{n^d} \pmod{n^{d+1}}$, $a = \mathbf{E}(z; R_a) \cdot g_2^m \pmod{n^{d+1}}$, and $b = \mathbf{E}(s; R_b) \cdot u_r^m \pmod{n^{d+1}}$.

We note that **ABM.col** runs a canonical sigma protocol on L_{pk}^{td} to prove that the prover knows $(x_2, (r, R_r, R_t))$ such that $u_r = \mathbf{E}_{pk}(r; R_r)$ and $u_t = g_1^{x_2} \mathbf{E}_{pk}(0; R_t) H(t)^r$. Hence, the trapdoor mode works correctly when $(t, (u_r, u_t)) \in L_{pk}^{\text{td}}$. On the contrary, **ABM.enc** runs a simulation algorithm of the sigma protocol with message (challenge) m . Notice that (A, a, b) implies the following linear system on \mathbb{Z}_{n^d} ,

$$\begin{pmatrix} \mathbf{D}(A) \\ \mathbf{D}(a) \\ \mathbf{D}(b) \end{pmatrix} = \begin{pmatrix} x_1 & y(t) & \mathbf{D}(u_t) \\ 1 & 0 & x_2 \\ 0 & 1 & \mathbf{D}(u_r) \end{pmatrix} \begin{pmatrix} z \\ s \\ m \end{pmatrix} \quad (2)$$

The matrix is invertible if

$$\mathbf{D}(u_t) \not\equiv (x_1 x_2 + y(t) \mathbf{D}(u_r)) \pmod{p} \text{ and } \mathbf{D}(u_t) \not\equiv (x_1 x_2 + y(t) \mathbf{D}(u_r)) \pmod{q},$$

which means that $(t, (u_r, u_t)) \in L_{pk}^{\text{ext}}$. Hence, the decryption mode works correctly.

Lemma 1 (Implicit in [31]). $(\text{ABM.gen}, \text{ABM.spl})$ is pPRF with unforgeability on $\widehat{L}_{pk}^{\text{td}} (= U'_{pk} \setminus L_{pk}^{\text{ext}})$, under the assumptions, 5, 6 and 7.

The proof is given in Sec. C.4. By this lemma, we have:

Theorem 2. *The scheme constructed as above is an ABME scheme if the DCR assumption (Assumption 5), the non-trivial divisor assumption (Assumption 6), and the non-multiplication assumption (Assumption 7) hold true.*

This scheme has a ciphertext consisting of only 5 group elements (including (u_r, u_t)) and optimal expansion factor $O(1)$. This scheme requires a public-key consisting of $\kappa + 3$ group elements along with some structure parameters.

8 ABM-LTF from DCR-based ABME and Vice Versa

Hofheinz [31] has presented the notion of all-but-many lossy trapdoor function (ABM-LTF). We provide the definition in Appendix E. We remark that ABM-LTF requires that, in our words, $(\text{ABM.gen}, \text{ABM.spl})$ be *strongly* unforgeable, whereas ABME only requires it be unforgeable. However, as shown in [31], unforgeable PPRF can be converted into strongly unforgeable PPRF via a chameleon commitment scheme. Therefore, this difference is not important. We note that we can regard Hofheinz’s DCR-based ABM-LTF (with only unforgeability) as a special case of our DCR-based ABME scheme by fixing a part of the coin space as $(R_A, R_a, R_b) = (1, 1, 1)$. Although the involved matrix of his original scheme is slightly different from ours, the difference is not essential. In the end, we can regard Hofheinz’s DCR-based ABM-LTF as

$$\text{ABM.eval}^{(t, (u_r, u_t))}(pk, (m, z, s)) := \text{ABM.enc}^{(t, (u_r, u_t))}(pk, m; (z, s, 1, 1, 1)),$$

where (m, z, s) denotes a message. This ABM-LTF has $((d-3) \log n)$ -lossyness. In the latest e-print version [31], Hofheinz has shown that his DCR-based ABM-LTF can be converted to SIM-SO-CCA PKE. To construct it, Hofheinz implicitly considered the following PKE scheme such that

$$\text{ABM.enc}^{(t, (u_r, u_t))}(pk, M; (m, z, s)) := (\text{ABM.eval}^{(t, (u_r, u_t))}(pk, (m, z, s)), M \oplus H(m, z, s)),$$

where H is a suitable 2-universal hash function from $(\mathbb{Z}_{n^d})^3$ to $\{0, 1\}^\kappa$ (or $\mathbb{Z}/n\mathbb{Z}$). According to his analysis in Sec. 7.2 in [31], if $d \geq 5$, it can open a ciphertext arbitrarily using Barvinok’s algorithm, when $(t, (u_r, u_t)) \in L^{\text{loss}}$. Then it turns out ABME in our words. For practical use, it is rather inefficient, because its expansion rate of ciphertext length per message length is ≥ 31 , and the modulus of $\geq n^6$ is required. The opening algorithm is also costly. Table 3 shows comparison.

On the contrary, our DCR-based ABME (strengthened with strong unforgeability) can be converted to ABM-LTF². Remember that $(A, a, b) = \text{ABM.enc}^{(t, (u_r, u_t))}(pk, m; (z, s, R_A, R_a, R_b))$. It is obvious that we can extract not only message m but (z, s) by inverting the corresponding matrix, but we point out that we can further retrieve (R_A, R_a, R_b) , too. This means that our DCR based ABME turns out ABM-LTF. Indeed, after extracting (m, z, s) from (A, a, b) , we have $(R_A)^{n^d}, (R_a)^{n^d}, (R_b)^{n^d}$

² Our approach is specific to our DCR-based ABME scheme. On one hand, Hemenway and Ostrovsky [30] have shown that if the message space of lossy encryption is one bit longer than the coin space, the lossy encryption can be converted to a lossy trapdoor function (LTF). Although their method can be applied to our DCR based ABME scheme, the resulting ABM-LTF is less efficient than ours.

Table 3. Comparison among ABMEs

ABME	expansion factor	ciphertext-length	message-length	pk-length
ABME from [31]	$\geq 31^*$	$(5(d+1)+1)\log n$	$\log n$	$(\kappa+3)d\log n$
Sec. 7.1 ($d \geq 1$)	$5+1/d$	$5(d+1)\log n$	$d\log n$	$(\kappa+3)d\log n$
Sec. 9	$5\kappa/\log \kappa$	$(5\ell+5)\log q$	$\ell\log \kappa$	$7\log q$

* : $d \geq 5$ is needed.

in $\mathbb{Z}_{n^{d+1}}^\times$. We remark that R_A, R_a, R_b lie not in $\mathbb{Z}_{n^{d+1}}^\times$ but in $(\mathbb{Z}/n\mathbb{Z})^\times$. So, letting $\alpha = r^{n^d} \bmod n^{d+1}$ where $r \in (\mathbb{Z}/n\mathbb{Z})^\times$, $r = \alpha^{(n^d)^{-1}} \bmod n$ is efficiently solved by $\phi(n)$. Thus, our DCR based ABME turns out ABM-LTF with $(d\log n)$ -lossyness for any $d \geq 1$, whereas Hofheinz's DCR based ABM-LTF is $((d-3)\log n)$ -lossy for any $d \geq 4$.

Table 4. Comparison among ABM-LTFs

ABM-LTF	expansion factor	output-length	input-length	lossyness	notes
[31]	$5/3$	$(5(d+1)+1)\log n$	$3d\log n$	$(d-3)\log n$	$d \geq 4$
ABM-LTF (Sec. 7)	$5/3$	$(5(d+1)+1)\log n$	$3(d+1)\log n$	$d\log n$	$d \geq 1$

9 ABME from Twin Cramer-Shoup

We construct an ABME scheme from the DDH assumption. The expansion factor of this scheme is not optimal but $O(\kappa/\log \kappa)$. However, this expansion rate is still better than the previous work [11] (with $O(\kappa)$). We note that we provide an alternative ABME scheme with the same expansion factor from the DDH assumption in Appendix D, which is the DDH version of the scheme in Sec. 7. So, its public key includes $O(\kappa)$ group elements. On the other hand, this scheme has a short public key only with a constant number of group elements.

We consider the following pPRF. Let Π^{cpa} be an IND-CPA (or even one-way) PKE scheme and let Π^{cca} be an IND-CCA Tag-based PKE scheme. Let pk^{cpa} and pk^{cca} be public keys of both schemes, respectively. Then, see $pk = (pk^{\text{cpa}}, pk^{\text{cca}}, \mathbf{E}^{\text{cpa}}(\xi))$ as the public key of pPRF, where ξ is a random message. Then, we see $\mathbf{E}^{\text{cca}}(t, \xi)$ as the output of Spl on tag t , where $sk^{\text{spl}} = \xi$. This indeed forms pPRF. We now describe a concrete construction by using El Gmal PKE and a tag-based version of Twin-Cramer-Shoup PKE [16, 14] as ingredients, with a slight optimization.

Let $\mathcal{CH} = (\text{CHGen}, \text{CHEval}, \text{CHColl})$ be a chameleon hash commitment scheme. Let g be a generator of a multiplicative group G of prime order q , where we assume that G is efficiently samplable and the DDH assumption holds on the group. Let $\text{TwinCS} = (\text{CS.gen}, \text{CS.enc}, \text{CS.dec})$. be a tag-based version of Twin-Cramer-Shoup PKE [16, 14], where

- **CS.gen(1^κ)**: Via $(pk_{\text{cs}}, sk_{\text{cs}}) \leftarrow \text{CS.gen}(1^\kappa)$, it picks up hash $(pk_{\mathcal{CH}}, sk_{\mathcal{CH}}) \leftarrow \text{CHGen}(1^\kappa)$, generator $g \leftarrow G^\times$, and sets $X = g^x$, $\hat{X} = g^{\hat{x}}$, $Y = g^y$, and $\hat{Y} = g^{\hat{y}}$, where $x, \hat{x}, y, \hat{y} \leftarrow \mathbb{Z}/q\mathbb{Z}$, and finally outputs $pk_{\text{cs}} := (pk_{\mathcal{CH}}, g, X, \hat{X}, Y, \hat{Y})$ and $sk_{\text{cs}} := (pk_{\text{cs}}, x, \hat{x}, y, \hat{y})$.

- $\text{CS.enc}(pk_{\text{cs}}, t, m)$: Via $c \leftarrow \text{CS.enc}(pk_{\text{cs}}, t, m)$, where message $m \in G$, and tag $t \in \{0, 1\}^\kappa$, it outputs $c = (r, d, e, \pi_x, \pi_y)$, by picking up $r \xleftarrow{\text{U}} \text{COIN}_{\mathcal{CH}}$, and computing $d := g^v$, $e := m \cdot X^v$, $\tau := \text{CHEval}(pk_{\mathcal{CH}}, (t, d, e); r)$, $\pi_x := (X^\tau \hat{X})^v$, and $\pi_y := (Y^\tau \hat{Y})^v$, where $v \xleftarrow{\text{U}} \mathbb{Z}/q\mathbb{Z}$.
- $\text{CS.dec}(sk_{\text{cs}}, t, c)$: Via $m = \text{CS.dec}(sk_{\text{cs}}, t, c)$, where $c := (r, d, e, \pi_x, \pi_y)$, it checks if $\pi_x \stackrel{?}{=} d^{\tau x + \hat{x}}$ and $\pi_y \stackrel{?}{=} d^{\tau y + \hat{y}}$, where $\tau = \text{CHEval}(pk_{\mathcal{CH}}, (t, d, e); r)$ and outputs $m := e \cdot d^{-x}$ if the above equations both hold, otherwise $m := \perp$.

TwinCS is an IND-CCA secure Tag-PKE scheme if the DDH assumption holds true and \mathcal{CH} is a chameleon commitment scheme. The proof is omitted.

pPRF = (Gen^{spl}, Spl) from TwinCS is constructed as follows:

- Gen^{spl}(1^κ): It picks up $(pk_{\text{cs}}, sk_{\text{cs}}) \leftarrow \text{CS.gen}(1^\kappa)$, where $pk_{\text{cs}} = (pk_{\mathcal{CH}}, g, X, \hat{X}, Y, \hat{Y})$ and $sk_{\text{cs}} = (x, \hat{x}, y, \hat{y})$. It picks up $\zeta \xleftarrow{\text{U}} G^\times$, $v_0 \xleftarrow{\text{U}} \mathbb{Z}/q\mathbb{Z}$, and computes $(d_0, e_0) = (g^{v_0}, \zeta^{-1} X^{v_0})$. It finally outputs $pk := (pk_{\text{cs}}, d_0, e_0)$ and $sk^{\text{spl}} := \zeta$.
- Spl(pk, sk^{spl}, t): It takes (pk, sk^{spl}, t) and outputs $u = (r, d, e, \pi_x, \pi_y) = \text{CS.enc}(pk_{\text{cs}}, t, \zeta; v)$ where $v \xleftarrow{\text{U}} \mathbb{Z}/q\mathbb{Z}$.

We let $L_{pk}^{\text{td}} :=$

$$\{(t, (r, d, e, \pi_x, \pi_y)) \mid \exists (\zeta, v_0, v) : (d_0, e_0) = (g^{v_0}, \zeta^{-1} X^{v_0}) \text{ and } (r, d, e, \pi_x, \pi_y) = \text{CS.enc}(pk_{\text{cs}}, t, \zeta; v)\}.$$

and $\hat{L}_{pk}^{\text{td}} :=$

$$\{(t, (r, d, e, \pi_x, \pi_y)) \mid \exists (\tilde{v}, v) : (d_0 d, e_0 e) = (g^{\tilde{v}}, X^{\tilde{v}}) \text{ and } (d, \pi_x, \pi_y) = (g^v, (X^\tau \hat{X})^v, (Y^\tau \hat{Y})^v)\},$$

where $\tau = \text{CHEval}(pk_{\mathcal{CH}}, (t, d, e); r)$. We note that $L_{pk}^{\text{td}} = \hat{L}_{pk}^{\text{td}}$. Hence, $L_{pk}^{\text{ext}} = U'_{pk} \setminus L_{pk}^{\text{td}}$, where $U'_{pk} := \{0, 1\}^\kappa \times \text{COIN}_{\mathcal{CH}} \times G^4$.

Lemma 2. *The scheme obtained above is a PPRF with unforgeability on \hat{L}_{pk}^{td} if the DDH assumption holds true and \mathcal{CH} is a chameleon commitment scheme.*

Proof. By construction, it is obvious that the above scheme satisfies pseudo randomness. The unforgeability follows from the following analysis.

Let us define G_0 as the original unforgeability game, in which the challenger sets up all secrets and public parameter $pk = (pk_{\text{cs}}, d_0, e_0)$. The challenger returns $(d, e, \pi_x, \pi_y) \leftarrow \text{CS.enc}(pk_{\text{cs}}, t, \zeta)$ for every query t that the adversary A submits as query. Let ϵ_0 be the advantage of A in game G_0 , i.e., the probability that it outputs $(d', e', \pi'_x, \pi'_y) \in \text{CS.enc}(pk_{\text{cs}}, t', \zeta)$ where t' is not queried.

We consider a sequence of $q + 1$ games, $G_{1,0}, \dots, G_{1,q}$, where q denotes the number of queries that A submits. We define Game $G_{1,0}$ as G_0 . Let t_1, \dots, t_q be a sequence of queries from A . In game $G_{1,i}$, where $i \in \{0, \dots, q\}$, the challenger returns $(d, e, \pi_x, \pi_y) \leftarrow \text{CS.enc}(pk_{\text{cs}}, t_j, 0^{|\zeta|})$ for $j \leq i$, whereas returns $(d, e, \pi_x, \pi_y) \leftarrow \text{CS.enc}(pk_{\text{cs}}, t_j, \zeta)$ for $j > i$. Let $\epsilon_{1,i}$ be the advantage of A in game $G_{1,i}$, i.e., the probability that it outputs $(d', e', \pi'_x, \pi'_y) \in \text{CS.enc}(pk_{\text{cs}}, t', \zeta)$ where t' is not queried.

The difference of the adversary's advantage, $\epsilon_{1,i} - \epsilon_{1,i+1}$, between each two games, $G_{1,i}$ and $G_{1,i+1}$, for every $i \in \{0, \dots, q-1\}$, is evaluated by the advantage of IND-CCA security for TwinCS. Namely, we construct an algorithm B using A as oracle that breaks IND-CCA security for TwinCS.

B takes pk_{cs} and chooses $\zeta \xleftarrow{u} G^\times$ and sets $(d_0, e_0) := (g^{v_0}, \zeta^{-1} X^{v_0})$ where $v_0 \xleftarrow{u} \mathbb{Z}/q\mathbb{Z}$. For the first j queries of A , with $j \leq i$, B returns $\text{CS.enc}(pk_{cs}, t_j, 0^{|\zeta|})$. When A submits the $i+1$ -th query t_{i+1} , B submits $(0^{|\zeta|}, \zeta)$ to the encryption oracle, and receives the challenge ciphertext $(d^*, e^*, \pi_x^*, \pi_y^*)$. For the remaining queries, B returns $\text{CS.enc}(pk_{cs}, t_j, \zeta)$ where $i+1 < j$.

When A outputs $c' = (d', e', \pi_x', \pi_y')$ for a fresh tag t' , B queries c' to the decryption oracle. If the decryption oracle returns ζ , B outputs bit 0; otherwise 1. By construction, we have $\epsilon_{1,i}(\kappa) - \epsilon_{1,i+1}(\kappa) \leq \text{Adv}_{\text{TwinCS}, A}^{\text{ind-cca}}(\kappa)$, for every $i \in \{0, \dots, q-1\}$, which is negligible in κ if the DDH assumption holds on G and \mathcal{CH} is a chameleon hash commitment scheme. We note that B needs the decryption oracle only once, to check that c' is a ciphertext of ζ .

In Game G_2 , the challenger behaves as follows: It is given pk_{cs} and $|\zeta|$ as input, chooses a random tag t , and obtains ciphertext (d, e, π_x, π_y) of a random message ζ^{-1} on tag t . It then sets $(d_0, e_0) := (d, e)$. Here, the challenger is not given ζ . For every query t_i of A , $1 \leq i \leq q$, the challenger returns $\text{CS.enc}(pk_{cs}, t_i, 0^{|\zeta|})$. Let ϵ_2 be the advantage of A in game G_2 . Since this change is conceptual from $G_{1,q}$ $\epsilon_{1,q} = \epsilon_2$.

Game G_3 is the same game as G_2 except that when A finally outputs $c' = (d', e', \pi_x', \pi_y')$ on a fresh tag t' , the challenger submits it to the decryption oracle and outputs its reply. We note that the challenger did not reveal any information on t to A , because it feeds only (d_0, e_0) to A . Hence, it holds that $t' \neq t$ with probability $1 - \frac{q}{2^\kappa}$. If c' is a ciphertext of ζ , the challenger results in decrypting $c = (d, e, \pi_x, \pi_y)$ on tag t , which is bounded by the advantage of an adversary that breaks one-wayness of TwinCS in the chosen-ciphertext attack. The advantage is bounded by twice of that of IND-CCA security of TwinCS.

Hence, we have $\epsilon_0(\kappa) \leq (q+2)\text{Adv}_{\text{TwinCS}, B}^{\text{ind-cca}}(\kappa) + \frac{q}{2^\kappa}$. ■

We now construct an ABME scheme from the Twin-Cramer-Shoup based pPRF scheme .

- **ABM.gen**(1^κ): It gets $(pk_{cs}, sk_{cs}) \leftarrow \text{CS.gen}(1^\kappa)$ (the key generation algorithm of Twin Cramer-Shoup), where $pk_{cs} = (pk_{\mathcal{CH}}, g, X, \hat{X}, Y, \hat{Y})$ and $sk_{cs} = (x, \hat{x}, y, \hat{y})$. It chooses $\xi \xleftarrow{u} G^\times$, $v_0 \xleftarrow{u} \mathbb{Z}/q\mathbb{Z}$, and computes $d_0 := g^{v_0}$, and $e_0 := \xi^{-1} X^{v_0}$. It sets $\lambda = O(\log \kappa)$. It finally outputs $pk, sk^{\text{spl}}, sk^{\text{ext}}$, where $pk := (pk_{cs}, d_0, e_0, \lambda)$, $sk^{\text{ext}} := sk_{cs}$, and $sk^{\text{spl}} := \zeta$. We let $U'_{pk} := \{0, 1\}^\kappa \times \text{COIN}_{\mathcal{CH}} \times G^4$ that contains the disjoint sets, L_{pk}^{td} and L_{pk}^{ext} , as defined below.
- **ABM.spl**($pk, sk^{\text{spl}}, t; v$): It takes (pk, sk^{spl}, t) where $sk^{\text{spl}} = \zeta$, picks up $v \xleftarrow{u} \mathbb{Z}/q\mathbb{Z}$, and outputs $u := (r, d, e, \pi_x, \pi_y) = \text{CS.enc}(pk_{cs}, \zeta; v)$, where $\tau := \text{CHEval}(pk_{\mathcal{CH}}, (t, d, e); r)$. Here we define $L_{pk}^{\text{td}} = \hat{L}_{pk}^{\text{td}} =$

$$\{(t, (r, d, e, \pi_x, \pi_y)) \mid \exists (\tilde{v}, v) : d_0 d = g^{\tilde{v}}, e_0 e = h^{\tilde{v}}, d = g^v, \pi_x = (X^\tau \hat{X})^v, \text{ and } \pi_y = (Y^\tau \hat{Y})^v\}.$$

We note that $\tilde{v} = v_0 + v$. We define $L_{pk}^{\text{ext}} = U'_{pk} \setminus \hat{L}_{pk}^{\text{td}}$.

- **ABM.enc**($^{(t,u)}$)($pk, m; (\hat{z}, z)$): To encrypt message $m \in \{0, 1\}^n$, it parses m as (m_1, \dots, m_ℓ) where $\ell = n/\lambda$ and $m_i \in \{0, 1\}^\lambda$. It picks up vectors, $\tilde{z}, z \xleftarrow{u} G^\ell$, where $\tilde{z} = (\tilde{z}_1, \dots, \tilde{z}_\ell)$ and $z = (z_1, \dots, z_\ell)$, and computes 2-by- ℓ matrix A 3-by- ℓ matrix B such that

$$A = \begin{pmatrix} g & d_0 d \\ X & e_0 e \end{pmatrix} \begin{pmatrix} \tilde{z}_1 & \dots & \tilde{z}_\ell \\ m_1 & \dots & m_\ell \end{pmatrix}, \text{ and } B = \begin{pmatrix} g & d \\ X^\tau \hat{X} & \pi_x \\ Y^\tau \hat{Y} & \pi_y \end{pmatrix} \begin{pmatrix} z_1 & \dots & z_\ell \\ m_1 & \dots & m_\ell \end{pmatrix}. \quad (3)$$

It finally outputs $c = (A, B)$.

- $\text{ABM.dec}^{(t,u)}(sk^{\text{ext}}, c)$: Let $A = (\mathbf{a}_1, \dots, \mathbf{a}_\ell)$ and $B = (\mathbf{b}_1, \dots, \mathbf{b}_\ell)$, where $\mathbf{a}_i = (a_{1,i}, a_{2,i})^\top$ and $\mathbf{b}_i = (b_{1,i}, b_{2,i}, b_{3,i})^\top$. For all $i \in [\ell]$, it searches “consistent” $m_i \in \{0, 1\}^\lambda$ such that

$$\frac{(a_{1,i})^x}{a_{2,i}} = \left(\frac{(d_0d)^x}{e_0e}\right)^{m_i} \text{ if } e_0e \neq (d_0d)^x, \quad \frac{(b_{1,i})^{\tau x + \hat{x}}}{b_{2,i}} = \left(\frac{d^{\tau x + \hat{x}}}{\pi_x}\right)^{m_i} \text{ if } \pi_x \neq d^{\tau x + \hat{x}},$$

$$\text{and } \frac{(b_{1,i})^{\tau y + \hat{y}}}{b_{3,i}} = \left(\frac{d^{\tau y + \hat{y}}}{\pi_y}\right)^{m_i} \text{ if } \pi_y \neq d^{\tau y + \hat{y}}, \quad \text{where } \tau = H(t, d, e). \quad (4)$$

It aborts if it find no m_i or “inconsistent” one for some $i \in [\ell]$; otherwise outputs $m = (m_1, \dots, m_\ell) \in \{0, 1\}^n$.

- $\text{ABM.col}_1^{(t,u)}(pk, t, sk^{\text{spl}}, v; (\tilde{\mathbf{w}}, \mathbf{w}))$: It picks up $\tilde{w}_i, w_i \xleftarrow{\text{U}} \mathbb{Z}/q\mathbb{Z}$ for $i \in [\ell]$. It sets $a_{1,i} := g^{\tilde{w}_i}$, $a_{2,i} := X^{\tilde{w}_i}$, $b_{1,i} := d^{w_i}$, $b_{2,i} := (X^\tau \hat{X})^{w_i}$, and $b_{3,i} := (Y^\tau \hat{Y})^{w_i}$, where $\tau = H(t, u, e)$. It finally outputs $c = (A, B)$ and $\xi = (v_0, v, \tilde{\mathbf{w}}, \mathbf{w})$, where $\tilde{\mathbf{w}} = (\tilde{w}_1, \dots, \tilde{w}_\ell)$ and $\mathbf{w} = (w_1, \dots, w_\ell)$.
- $\text{ABM.col}_2^{(t,u)}(\xi, m)$: To open $c = (A, B)$ to m , it parses m as (m_1, \dots, m_ℓ) and computes, for all $i \in [\ell]$, $\tilde{z}_i := \tilde{w}_i - m_i \cdot \tilde{v} \bmod q$ and $z_i := w_i - m_i \cdot v \bmod q$, where $\tilde{v} = v_0 + v$. It finally outputs $(\tilde{\mathbf{z}}, \mathbf{z})$, consistent with m in Equation (3).

Suppose that $(t, (r, d, e, \pi_x, \pi_y)) \in L_{pk}^{\text{td}}$. Each column vector $\mathbf{a}_i = (a_{1,i}, a_{2,i})^\top$ in A from ABM.col_1 can be seen as the first message in a canonical sigma protocol on common input (d_0d, e_0e) to prove that $\log_g(d_0d) = \log_X(e_0e)$, and \tilde{z}_i from ABM.col_2 corresponds to the response on challenge m_i . Hence, $(A, \mathbf{m}, \tilde{\mathbf{z}})$ is the accepting conversation of the parallel execution of the sigma protocol with parallel challenge $\mathbf{m} = (m_1, \dots, m_\ell)$, where $m_i \in \{0, 1\}^\lambda$. Similarly, $(B, \mathbf{m}, \mathbf{z})$ is the accepting conversation of the parallel execution of a sigma protocol on common input (d, π_x, π_y) with parallel challenges \mathbf{m} to prove that $\log_g(d) = \log_{X^\tau \hat{X}}(\pi_x) = \log_{Y^\tau \hat{Y}}(\pi_y)$. By construction, the trapdoor mode works correctly.

The decryption mode works as follows: We note that $(t, (r, d, e, \pi_x, \pi_y)) \in L_{pk}^{\text{td}}$ if and only if $\text{rank}(A(t, u)) = 1$ and $\text{rank}(B(t, u)) = 1$, where $A(t, u) := \begin{pmatrix} g & d_0d \\ X & e_0e \end{pmatrix}$ and $B(t, u) := \begin{pmatrix} g & d \\ X^\tau \hat{X} & \pi_x \\ Y^\tau \hat{Y} & \pi_y \end{pmatrix}$.

So, when $(t, (r, d, e, \pi_x, \pi_y)) \in L_{pk}^{\text{ext}} (= U'_{pk} \setminus L_{pk}^{\text{td}})$, $\text{rank}(A(t, u)) = 2$ or $\text{rank}(B(t, u)) = 2$. Hence, each m_i can be retrieved by checking either of equations in (4). We note that if $\text{rank}(A(t, u)) = \text{rank}(B(t, u)) = 2$, the linear system (3) is overdetermined. Then, one should check if \mathbf{m} is inconsistent to the system (that is, there is no solution in the system), using the other equations. If so, the decryption is rejected.

We note, however, that the “consistency check” is unnecessary for our motivating application (fully-equipped UC commitments), because it suffices that the simulator can decrypt *valid* ciphertexts correctly, because an adversary cannot correctly open an invalid ciphertext on $(t, u) \in L_{pk}^{\text{ext}}$.

Theorem 3. *The scheme constructed as above is an ABME scheme if the DDH assumption on G holds true and \mathcal{CH} is a chameleon hash commitment scheme.*

This scheme has a ciphertext consisting of $5\ell + 4$ group elements plus $|\text{COIN}_{\mathcal{CH}}|$ -bit string (including $u = (r, d, e, \pi_x, \pi_y)$), for encrypting message $m \in \{0, 1\}^{\ell\lambda}$, with a public-key consisting of 7 group elements along with structure parameters. Therefore, the expansion factor of this scheme is $5\frac{\kappa}{\lambda} = O(\frac{\kappa}{\log \kappa})$. Since the UC commitment from [11] consists of two Cramer-Shoup encryptions plus the output of a claw-free permutation per one-bit message, its expansion factor is 8κ plus the length of the trap door commitment. This expansion factor in [11] is strict, by construction, which cannot be improved.

10 Fully-Equipped UC Commitment from Trapdoor Permutations

If we can construct an ABME scheme from trapdoor permutation (family), it is done, but we have no idea how to construct it. We instead construct a *weak* ABME scheme. The only difference of weak ABME from standard ABME is that in the trapdoor mode, $\text{dist}^{\text{enc}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)$ is not statistically but *computationally* indistinguishable from $\text{dist}^{\text{col}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)$. Namely,

$$\left\{ \left(\text{ABM.spl}(pk, sk^{\text{spl}}, t; v), \quad c, \quad \text{ABM.col}_2^{(t,u)}(\xi, x) \right) \right\} \stackrel{c}{\approx} \left\{ \left(\text{ABM.spl}(pk, sk^{\text{spl}}, t; v), \quad \text{ABM.enc}^{(t,u)}(pk, x; r), \quad r \right) \right\}$$

for every $(pk, (sk, w)) \in \text{ABM.gen}(1^\kappa)$, every $x \in \text{MSP}$, every $t \in \{0, 1\}^\kappa$, where $v \leftarrow \text{COIN}^{\text{spl}}$, $(c, \xi) \leftarrow \text{ABM.col}_1^{(t,u)}(pk, sk^{\text{spl}}, v)$, and $r \leftarrow \text{COIN}^{\text{enc}}$. We construct a weak ABME scheme from two independent trapdoor permutations as follows.

Let $\mathcal{F} = \{(f, f^{-1}) \mid f : \{0, 1\}^\kappa \rightarrow \{0, 1\}^\kappa\}_{\kappa \in \mathbb{N}}$ be a trapdoor permutation family and let $b : \{0, 1\}^\kappa \rightarrow \{0, 1\}$ be a hard-core predicate for a trapdoor permutation f . Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be the Blum-Goldwasser cryptosystem [6] that is a semantic secure public key encryption scheme, derived from the following encryption algorithm $\mathbf{E}_f(x; r) = f^{(k+1)}(r) \parallel (x_1 \oplus b(r)) \parallel \dots \parallel (x_k \oplus b(f^{(k)}(r)))$, where (x_1, \dots, x_κ) , $x_i \in \{0, 1\}$, 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 has efficiently samplable and explainable presumable ciphertext space $\{0, 1\}^{\kappa+k}$ [11, 22]. Let us denote by $F : \{0, 1\}^\kappa \times \{0, 1\}^\kappa \rightarrow \{0, 1\}^\kappa$ a pseudo-random function (constructed from f in a standard way).

- $\text{ABM.gen}(1^\kappa)$: It draws two trapdoor permutations, (f, f^{-1}) and (f', f'^{-1}) , over $\{0, 1\}^\kappa$ uniformly and independently from \mathcal{F} . Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be the Blum-Goldwasser cryptosystem mentioned above. Let F be a pseudo random function derived from f' . It then picks up random $s \leftarrow \{0, 1\}^\kappa$ and encrypt it to $e' = \mathbf{E}_{f'}(s; r)$. It outputs $(pk, sk^{\text{spl}}, sk^{\text{ext}})$, where $pk = (F, f, f', e')$, $sk^{\text{spl}} = (s, r)$, and $sk^{\text{ext}} = f^{-1}$. We define $U'_{pk} = \{0, 1\}^\kappa \times \{0, 1\}^k$.
- $\text{ABM.spl}(pk, sk^{\text{spl}}, t)$: It takes tag $t \in \{0, 1\}^\kappa$ and outputs $u = F_s(t)$ where $sk^{\text{spl}} = (s, r)$. We define

$$L_{pk}^{\text{td}} = \widehat{L}_{pk}^{\text{td}} = \{(t, u) \mid \exists (s, r) \text{ such that } e' = \mathbf{E}_{f'}(s; r) \text{ and } u = F_s(t)\}.$$

- $\text{ABM.enc}^{(t,u)}(pk, x)$: 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 $e' = \mathbf{E}_{f'}(s; r)$, by using the NP-reduction. We note that one can find such G without knowing (s, r) . In addition, if such (s, r) does not exist for given (t, u) , G so obtained does not have a Hamiltonian cycle.
 - To encrypt 0, it picks a random permutation $\pi = (\pi_1, \dots, \pi_q)$ 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 permuted 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 ($i \in [q]$). 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. 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.

This encryption procedure is the same as the adaptive Hamiltonian commitment protocol in [12], except that a commitment in our scheme is encrypted under a public key f independent of F .

- $\text{ABM.dec}^{(t,u)}(sk, c)$: To decrypt $c = (\{A_i\}_{i \in [q]}, \{B_{i,j}\}_{i,j \in [q]})$, it firstly decrypts all elements to retrieve π and matrix H , using $sk = f^{-1}$. Then it checks that $H = \pi(G)$. If it holds, it outputs 0; otherwise, 1.
- $\text{ABM.col}_1^{(t,u)}(pk, sk^{\text{spl}}, v)$: It first obtains a graph G (of q nodes) so that finding a Hamiltonian cycle in G is equivalent to finding $sk^{\text{spl}} = (s, r)$ such that $u = F_s(t)$ and $e' = \mathbf{E}_{f'}(s, r)$, by using the NP-reduction. It picks a random permutation $\pi = (\pi_1, \dots, \pi_q)$ 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 permuted graph $H = \pi(G)$. It outputs (c, ξ) where $c = (\{A_i\}_{i \in [q]}, \{B_{i,j}\}_{i,j \in [q]})$ and $\xi = ((t, u), \zeta, \pi)$. Here ζ denotes the Hamiltonian cycle of G .
- $\text{ABM.col}_2(\xi, x)$: If $x = 0$, it opens π and every entry of the adjacency matrix, otherwise if $x = 1$, it opens only the entries corresponding to the Hamiltonian cycle ζ in the adjacency matrix.

Then, we apply this weak ABME scheme to our framework (Fig. 3).

Theorem 4. *The scheme in Fig.3 obtained by applying the above weak ABME UC-securely realizes the $\mathcal{F}_{\text{MCOM}}$ functionality in the \mathcal{F}_{CRS} -hybrid model in the presence of adaptive adversaries in the non-erasure setting.*

Proof. The only difference from the proof of Theorem 1 is when we compare the ideal world with Hybrid Game 1. In the proof of Theorem 1, in the trapdoor mode when $(t, u) \in L_{pk}^{\text{td}}$, the output of ABM.col is statistically indistinguishable from that of ABM.enc . However, this case only guarantees computational difference. To show that the environment views in both games are computationally indistinguishable, we need to construct, for contradiction, a distinguisher that can distinguish the output of ABM.col from the output of ABM.enc without knowing sk^{spl} , while it can extract the values committed to by corrupted parties at the same time. Fortunately, in this construction, the decryption key $sk^{\text{ext}} = f^{-1}$ is independent of the equivocable key $sk^{\text{spl}} = (s, r)$. It is not the case of the rest of our constructions, in which one can obtain sk^{spl} if one knows sk^{ext} . Therefore, we require statistical closeness in there. Hence, we can construct a distinguisher that takes $sk^{\text{ext}} = f^{-1}$ and starts either with the ideal world or Hybrid Game 1. Here, the environment views in both games are bounded by the distinguisher's advantage, which is negligible. ■

We note that if the common reference string must strictly come from the uniform distribution, we require trapdoor permutations with dense public descriptions.

We note that *parallel k executions* of this weak ABME scheme with one bit message space yields a weak ABME scheme with k -bit message space, by sending parallel ciphertexts of the same message on the same tag under the same public key. Then, the scheme is also transformed into a fully-equipped UC-secure commitment scheme with k -bit message space.

This construction does not require non-interactive zero-knowledge proof systems. To the best of our knowledge, the most efficient non-interactive zero-knowledge proofs from trapdoor permutations is given by Kilian and Petrank [34], which requires a CRS size of $\omega(|C|\kappa^2 \log \kappa)$ and a proof size of $\omega(|C|\kappa^2 \log \kappa)$, where $|C|$ is the circuit size of the statement. We compare our construction with the previous result [12] with the most efficient NIZK proof system in Table 5.

Table 5. Fully-Equipped UC commitments (to λ bit secret) from general assumptions (enhanced trapdoor permutations).

schemes	CRS size	communication	complexity of each user
CLOS02 [12]	$\omega(\kappa^3 \log(\kappa))$	$\omega(\lambda \cdot q^2 \kappa^3 \log \kappa)$	$\lambda q^2 T_{\text{NP}} + \omega(\lambda q^2 T_{\text{tdp}}(\kappa^3 \log \kappa))$
Sec. 10	$O(\kappa)$	$O(\lambda \cdot q^2 \kappa)$	$T_{\text{NP}} + \lambda q^2 T_{\text{tdp}}(\kappa)$

T_{NP} denotes the cost of one NP reduction from one-way function to a Hamiltonian graph. $T_{\text{tdp}}(\kappa)$ denotes the cost of computing one execution of trapdoor permutation over $\{0, 1\}^k$. q denotes the number of the vertices of the Hamiltonian graph.

11 Acknowledgments

We thank Kirill Morozov and his students for nice feedback in the early version of this work. We also thank Dennis Hofheinz for valuable discussion.

References

1. D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *STOC '96*, pages 479–488, Philadelphia, Pennsylvania, USA, May 1996. ACM.
2. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Joux* [33], pages 1–35.
3. M. Bellare, S. Micali, and R. Ostrovsky. Perfect zero-knowledge in constant rounds. In *STOC '90*, pages 482–493. ACM, 1990.
4. M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters’ ibe scheme. In *Joux* [33], pages 407–424.
5. O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Analysis and improvement of lindell’s uc-secure commitment schemes. In M. J. Jacobson, Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS 2013*, volume 7954 of *Lecture Notes in Computer Science*, pages 534–551. Springer, Heidelberg, 2013.
6. 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, Heidelberg, 1985.
7. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Cachin and Camenisch [8], pages 223–238.
8. 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, Heidelberg, 2004.
9. 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, Heidelberg, 2003.
10. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS 2001*, pages 136–145. IEEE Computer Society, 2001. The full version available at <http://eprint.iacr.org/2000/067>.
11. 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, Heidelberg, 2001.
12. 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 is available at <http://eprint.iacr.org/2002/140>.
13. I. Cascudo, I. Damgård, B. M. David, I. Giacomelli, J. B. Nielsen, and R. Trifiletti. Additively homomorphic UC commitments with optimal amortized overhead. In J. Katz, editor, *PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 495–515. Springer, Heidelberg, 2015.
14. D. Cash, E. Kiltz, and V. Shoup. The twin diffie-hellman problem and applications. In *Smart* [44], pages 127–145.

15. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Desmedt [21], pages 174–187.
16. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33(1):167–226, 2004. Early version in CRYPTO’98.
17. I. Damgård, B. M. David, I. Giacomelli, and J. B. Nielsen. Compact VSS and efficient homomorphic UC commitments. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014 (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 213–232. Springer, Heidelberg, 2014.
18. I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *STOC 2003*, pages 426–437. ACM, 2003.
19. 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, Heidelberg, 2001.
20. 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, Heidelberg, 2002. The full version is available at <http://www.brics.dk/RS/01/41/>.
21. 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, Heidelberg, 1994.
22. S. Fehr, D. Hofheinz, E. Kiltz, and H. Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 381–402. Springer, Heidelberg, 2010.
23. 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, Heidelberg, 2011.
24. T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, and R. Trifiletti. On the complexity of additively homomorphic UC commitments. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A (1)*, volume 9562 of *Lecture Notes in Computer Science*, pages 542–565. Springer, Heidelberg, 2016.
25. 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, Heidelberg, 2012.
26. J. A. Garay, Y. Ishai, R. Kumaresan, and H. Wee. On the complexity of UC commitments. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 677–694. Springer, Heidelberg, 2014.
27. 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, Heidelberg, 2003.
28. 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, Heidelberg, 2004. The full version available at at Cryptology ePrint Archive <http://eprint.iacr.org/2003/214>.
29. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In Smart [44], pages 415–432.
30. B. Hemenway and R. Ostrovsky. Building lossy trapdoor functions from lossy encryption. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013 (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 241–260. Springer, Heidelberg, 2013.
31. D. Hofheinz. All-but-many lossy trapdoor functions. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 209–227. Springer, Heidelberg, 2012. (last revised 18 Mar 2013 at <http://eprint.iacr.org/2011/230>).
32. T. Itoh, Y. Ohta, and H. Shizuya. Language dependent secure bit commitment. In Desmedt [21], pages 188–201.
33. A. Joux, editor. *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 2009.
34. J. Kilian and E. Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.
35. E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, Heidelberg, 2006.

36. 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, Heidelberg, 2011. The full version available at at Cryptology ePrint Archive <http://eprint.iacr.org/2011/180>.
37. P. MacKenzie, M. K. Reiter, and K. Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In M. Naor, editor, *TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 171–190. Springer, Heidelberg, 2004.
38. P. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In Cachin and Camenisch [8], pages 382–400.
39. R. Nishimaki, E. Fujisaki, and K. Tanaka. An efficient non-interactive universally composable string-commitment scheme. *IEICE Transactions*, 95-A(1):167–175, 2012.
40. 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, Heidelberg, 1999.
41. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, Heidelberg, 2008.
42. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *STOC 2008*, pages 187–196. ACM, 2008.
43. V. Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, December 2001.
44. N. P. Smart, editor. *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*. Springer, Heidelberg, 2008.
45. 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, Heidelberg, 2005.

A Definitions

A.1 Collision-Resistant Hash Function Family

Let $\mathcal{H} = \{H_\iota\}_{\iota \in \mathcal{I}}$ be a keyed hash family of functions $H_\iota : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ indexed by $\iota \in \mathcal{I}_\kappa (= \mathcal{I} \cap \{0, 1\}^\kappa)$. A keyed hash-function family \mathcal{H} is called collision-resistant (CR) if, for every non-uniform PPT adversary C , $\Pr[\iota \leftarrow \mathcal{I}_\kappa; (x, y) \leftarrow C_\kappa(H_\iota) : x \neq y \wedge H_\iota(x) = H_\iota(y)] = \text{negl}(\kappa)$.

A.2 Chameleon Commitment

A chameleon commitment $\mathcal{CH} = (\text{CHGen}, \text{CHEval}, \text{CHColl})$ consists of three algorithms: CHGen is a PPT algorithm that takes as input security parameter 1^κ and outputs a pair of public and trapdoor keys (pk, tk) . CHEval is a PPT algorithm that takes as input pk and message $x \in \{0, 1\}^\kappa$, drawing random r from coin space COIN_{pk} , and outputs chameleon hash value $c = \text{CHEval}(pk, x; r)$. Here COIN_{pk} is uniquely determined by pk . CHColl is a DPT algorithm that takes as input (pk, tk) , $x, x' \in \{0, 1\}^\kappa$ and $r \in \text{COIN}_{pk}$, and outputs $r' \in \text{COIN}_{pk}$ such that $\text{CHEval}(pk, x; r) = \text{CHEval}(pk, x'; r')$. We require that for every (pk, tk) generated by $\text{CHGen}(1^\kappa)$, every $x, x' \in \{0, 1\}^\kappa$, and every $r \in \text{COIN}_{pk}$, there exists a *unique* $r' \in \text{COIN}_{pk}$ such that $\text{CHEval}(pk, x; r) = \text{CHEval}(pk, x'; r')$, and $\text{CHColl}(pk, tk, x, x', r)$ always computes r' in time $\text{poly}(\kappa + |x| + |x'|)$. In addition, for any x, x' , if r is uniformly distributed, then so is r' . We require \mathcal{CH} is collision-resistance in the following sense: For every non-uniform PPT adversary A ,

$$\Pr \left[(pk, tk) \leftarrow \text{CHGen}(1^\kappa); (x_1, x_2, r_1, r_2) \leftarrow A(pk) : \text{CHEval}(pk, x_1; r_1) = \text{CHEval}(pk, x_2; r_2) \wedge (x_1 \neq x_2) \right] = \text{negl}(\kappa).$$

A.3 Tag-Based PKEs

A Tag-PKE $\Pi = (\text{Tag.Gen}, \text{Tag.Enc}, \text{Tag.Dec})$ is a tag-based PKE [43, 37, 35] that consists of three polynomial-time algorithms: Tag.Gen , the key-generation algorithm, is a PPT algorithm which on input 1^n outputs a pair of the public and secret keys, (pk, sk) . Tag.Enc , the encryption algorithm, is a PPT algorithm that takes public key pk , a tag $t \in \{0, 1\}^{p(\kappa)}$ for some fixed polynomial p and message $m \in \text{MSP}$, and produces $c \leftarrow \text{Tag.Enc}(pk, t, m; r)$, picking up $r \xleftarrow{\text{U}} \text{COIN}$, where MSP and COIN denote the message space and the coin space determined by pk , respectively. Tag.Dec , the decryption algorithm, is a deterministic polynomial-time algorithm that takes a secret key sk , t , and a ciphertext $c \in \{0, 1\}^*$, and outputs $\text{Tag.Dec}(sk, t, c)$. We require that for (sufficiently large) every $k \in \mathbb{N}$, every $t \in \{0, 1\}^{p(\kappa)}$ every (pk, sk) generated by $\text{Tag.Gen}(1^k)$, and every message $m \in \text{MSP}$, it always holds $\text{Tag.Dec}(sk, t, \text{Tag.Enc}(pk, t, m)) = m$.

IND-CCA Security We recall CCA security for Tag-PKEs [37], called weak CCA security [35]. We simply call it IND-CCA (for Tag-PKEs), because we only consider tag-PKEs.

We define IND-CCA security for tag-PKEs as follows. To an adversary $A = (A_1, A_2)$ and $b \in \{0, 1\}$, we associate the following experiment $\text{Expt}_{\Pi, A, b}^{\text{ind-cca}}(\kappa)$.

$$\begin{aligned} & \text{Expt}_{\Pi, A, b}^{\text{ind-cca}}(\kappa): \\ & (pk, sk) \leftarrow \text{Tag.Gen}(1^\kappa) \\ & (t^*, m_0, m_1, st) \leftarrow A_1^{\text{D}^{sk}}(pk) \\ & c^* \leftarrow \text{Tag.Enc}(pk, t^*, m_b) \\ & b' \leftarrow A_2^{\text{Tag.Dec}_{sk}}(st, t^*, c^*) \\ & \text{Return } b'. \end{aligned}$$

The adversary A_2 is restricted not to query decryption oracle $\text{Tag.Dec}(sk, \cdot, \cdot)$ with (t^*, \star) . We define the advantage of A in the experiment as

$$\text{Adv}_{\Pi, A}^{\text{ind-cca}}(\kappa) = \Pr[\text{Expt}_{\Pi, A, 1}^{\text{ind-cca}}(\kappa) = 1] - \Pr[\text{Expt}_{\Pi, A, 0}^{\text{ind-cca}}(\kappa) = 1].$$

We say that Π is IND-CCA secure if $\text{Adv}_{\Pi, A}^{\text{ind-cca}}(\kappa) = \text{negl}(\kappa)$ for every PPT A .

B UC Framework and Fully-Equipped UC Commitments from ABME

B.1 UC framework and Ideal Commitment Functionality

The UC framework defines a non-uniform 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 which 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 (i.e., the honest parties in the ideal world 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 real-world 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 real-world honest party includes randomness it has used. We insist that honest parties cannot erase any of its state (non-erasure model).

We denote by $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)$ 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 simulator \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 $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}(\kappa, z)$ 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}_{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 $\text{HYBRID}_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{CRS}}}(\kappa, z)$. Informally, a protocol π UC-realizes a functionality \mathcal{F} in the \mathcal{F}_{CRS} hybrid model if there exists a PPT simulator \mathcal{S} such that for every non-uniform PPT environment \mathcal{Z} every PPT adversary \mathcal{A} , and every polynomial $p(\cdot)$, it holds that

$$\{\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}} \stackrel{c}{\approx} \{\text{HYBRID}_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{CRS}}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}}.$$

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 [10].

We consider UC commitment schemes that can be used repeatedly under a single common reference string (re-usable common reference string). The multi-commitment ideal functionality $\mathcal{F}_{\text{MCOM}}$ from [12] is the ideal functionality of such commitments, which is given in Figure 4.

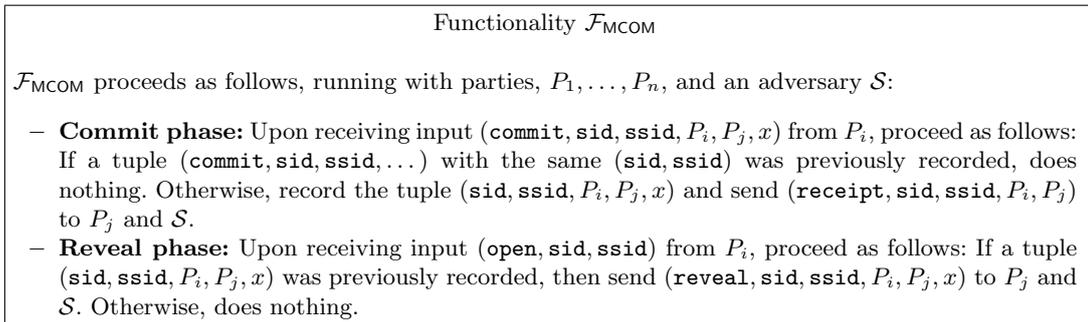


Fig. 4. The ideal multi-commitment functionality

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

B.2 Proof of Theorem 1

Theorem 1 (restated) The proposed scheme in Fig.3 UC-securely realizes the $\mathcal{F}_{\text{MCOM}}$ functionality in the \mathcal{F}_{CRS} -hybrid model in the presence of adaptive adversaries in the non-erasure model.

For simplicity, we assume $\{0, 1\}^\kappa \subset \text{MSP}$, without loss of generality, which enables us to remove the injective map $\iota : \{0, 1\}^\kappa \rightarrow \text{MSP}$ from the scheme. The description of the simulator's task is described as follows:

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

- **Initialization step:** \mathcal{S} chooses $(pk, sk) \leftarrow \text{ABM.gen}(1^\kappa)$ and sets CRS to be pk (along with U_{pk} and $U' = \{0, 1\}^\kappa \times U_{pk}$).
- **Simulating ideal functionality \mathcal{F}_{CRS} :** Since \mathcal{S} simulates \mathcal{F}_{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 $\mathcal{F}_{\text{MCOM}}$ the receipt message $(\text{receipt}, \text{sid}, \text{ssid}, P_i, P_j)$, \mathcal{S} generates $u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)$ so that $(t, u) \in L_{pk}^{\text{td}}$, where $t = (\text{sid}, \text{ssid}, P_i, P_j)$, and computes $(c, \xi) = \text{ABM.col}_1^{(t, u)}(pk, sk^{\text{spl}}, v)$, namely, c is a fake ciphertext on (t, u) . \mathcal{S} sends $(\text{sid}, \text{ssid}, (t, u, c))$ to adversary \mathcal{A} , as it expects to receive from P_i . \mathcal{S} stores $(\text{sid}, \text{ssid}, P_i, P_j, (t, u, c), \xi)$.
- **Simulating the decommit phase when P_i is honest:** Upon receiving from $\mathcal{F}_{\text{MCOM}}$ the message $(\text{open}, \text{sid}, \text{ssid}, P_i, P_j, x)$, \mathcal{S} computes $r = \text{ABM.col}_2^{(t, u)}(\xi, x)$ and sends $(\text{sid}, \text{ssid}, x, r)$ to adversary \mathcal{A} .
- **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 $(\text{sid}, \text{ssid}, P_i, P_j, x)$, whose value previously came from \mathcal{Z} and was sent to $\mathcal{F}_{\text{MCOM}}$, and then computes $r = \text{ABM.col}_2^{(t, u)}(\xi, x)$ and R such that $u = U_{pk}(t; R)$, which can be efficiently computable because U_{pk} is an explainable domain. Finally, it reveals (x, r, R) to \mathcal{A} .
- **Simulating the commit phase when the committer P_i is corrupted and the receiver P_j is honest:** Upon receiving $(\text{sid}, \text{ssid}, (t, u), c)$ from \mathcal{A} , \mathcal{S} decrypts $x = \text{ABM.dec}^{(t, u)}(sk^{\text{ext}}, c)$. If the decryption is invalid, then \mathcal{S} sends a dummy commitment $(\text{commit}, \text{sid}, \text{ssid}, P_i, P_j, \varepsilon)$ to $\mathcal{F}_{\text{MCOM}}$. Otherwise, \mathcal{S} sends $(\text{commit}, \text{sid}, \text{ssid}, P_i, P_j, x)$ to $\mathcal{F}_{\text{MCOM}}$.
- **Simulating the decommit stage when the committer P_i is corrupted and the receiver P_j is honest:** Upon receiving $(\text{sid}, \text{ssid}, x', r')$ from \mathcal{A} , as it expects to send to P_j , \mathcal{S} sends $(\text{open}, \text{sid}, \text{ssid})$ to $\mathcal{F}_{\text{MCOM}}$. ($\mathcal{F}_{\text{MCOM}}$ follows its codes: If a tuple $(\text{sid}, \text{ssid}, P_i, P_j, x)$ with the same $(\text{sid}, \text{ssid})$ was previously stored by $\mathcal{F}_{\text{MCOM}}$, $\mathcal{F}_{\text{MCOM}}$ sends $(\text{sid}, \text{ssid}, P_i, P_j, x)$ to P_j and \mathcal{S} .)
- **Simulating adaptive corruption of P_j after the commit phase but before the decommit phase:** When P_j has been corrupted, \mathcal{S} simply reveals $(\text{sid}, \text{ssid}, (t, u, c))$ to adversary \mathcal{A} as if it comes from P_j .

We remark that in the ideal world, honest parties simply convey inputs from environment \mathcal{Z} to the ideal functionalities and vice versa. Therefore, when $\mathcal{F}_{\text{MCOM}}$ sends something to honest P_j , it is immediately sent to \mathcal{Z} .

We will prove that there is an ideal-world simulator \mathcal{S} such that for every \mathcal{Z} , every \mathcal{A} , and every polynomial $p(\cdot)$,

$$\{\text{IDEAL}_{\mathcal{F}_{\text{MCOM}}, \mathcal{S}^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}} \stackrel{c}{\approx} \{\text{HYBRID}_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{crs}}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}}.$$

To prove this, we then 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 $\mathcal{F}_{\text{MCOM}}^1$, and the simulator, denoted \mathcal{S}_1 , work exactly in the same way as $\mathcal{F}_{\text{MCOM}}$ and \mathcal{S} do respectively, except for the case that P_i is honest: In Hybrid Game 1, at the beginning of the commit phase, $\mathcal{F}_{\text{MCOM}}^1$ gives simulator \mathcal{S}_1 the committed value x together with (receipt, sid, ssid, P_i, P_j). \mathcal{S}_1 then sets up $(t, u) \in L_{pk}^{\text{td}}$ in the same way as \mathcal{S} does (using sk^{spl}), but \mathcal{S}_1 instead computes c as $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$, by picking up $r \stackrel{\text{U}}{\leftarrow} \text{COIN}^{\text{enc}}$. When simulating the decommit phase or simulating adaptive corruption of P_i before the decommit phase, \mathcal{S}_1 reveals (u, x, r, R) after computing R such that $u = U_{pk}(t; R)$.

Consider the simulation that honest P_i opens commitment (t, u, c) in both games. The distribution of (u, c, r) on $t = (\text{sid}, \text{ssid}, P_i, P_j)$ as generated in Hybrid Game 1 is statistically indistinguishable from those on the same t as generated in the ideal world, because the two distribution ensembles, $\{\text{dist}^{\text{col}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)\}_{\kappa \in \mathbb{N}}$ and $\{\text{dist}^{\text{enc}}(pk, t, sk^{\text{spl}}, sk^{\text{ext}}, x)\}_{\kappa \in \mathbb{N}}$, defined in Sec. 4, are statistically indistinguishable in κ . So, we have

$$\{\text{IDEAL}_{\mathcal{F}_{\text{MCOM}}, \mathcal{S}^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}} \stackrel{s}{\approx} \{\text{HYBRID}_{\mathcal{F}_{\text{MCOM}}^1, \mathcal{S}_1^{\mathcal{A}}, \mathcal{Z}}^1(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}}.$$

Hybrid Game 2: In this game, the ideal commitment functionality $\mathcal{F}_{\text{MCOM}}^2$ 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 commit phase: At the commit phase in Hybrid Game 2, when \mathcal{S}_2 receives (t, u, c) from P_i controlled by adversary \mathcal{A} where $t = (\text{sid}, \text{ssid}, P_i, P_j)$, \mathcal{S}_2 sends a dummy commitment (commit, sid, ssid, P_i, P_j, ε) to $\mathcal{F}_{\text{MCOM}}^2$. At 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 $c \neq \text{ABM.enc}^{(t,u)}(pk, x'; r)$; otherwise, it sends (open, sid, ssid, x') to $\mathcal{F}_{\text{MCOM}}^2$. Then, $\mathcal{F}_{\text{MCOM}}^2$ replaces the stored value ε with value x' and sends (reveal, sid, ssid, P_i, P_j, x') to P_j and \mathcal{S}_2 .

Let us define BD_I as each event in Hybrid Game I , where $I = 1, 2$, that the simulator receives a *fake* ciphertext c on (t, u) from P_i controlled by adversary \mathcal{A} . Remember that ciphertext c is called fake if $(t, u) \in L_{pk}^{\text{td}}$ and c is a *valid* ciphertext (which means that there is a pair of message/randomness consistent with c). The hybrid games, 1 and 2, may differ only when BD_1 and BD_2 occur in each game, which means that $\neg \text{BD}_1 = \neg \text{BD}_2$ and thus, $\text{BD}_1 = \text{BD}_2$. 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, $\text{BD} := \text{BD}_1 = \text{BD}_2$.

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

$$\text{Dist}\left(\text{HYBRID}_{\mathcal{F}_{\text{MCOM}}^1, \mathcal{S}_1^{\mathcal{A}}, \mathcal{Z}}^1(\kappa, z), \text{HYBRID}_{\mathcal{F}_{\text{MCOM}}^2, \mathcal{S}_2^{\mathcal{A}}, \mathcal{Z}}^2(\kappa, z)\right) \leq \Pr[\text{BD}],$$

where the output of \mathcal{Z} is a bit.

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

Lemma 3. *Event BD occurs at most with probability $q_A \epsilon^{\text{euf}}$, where q_A denotes the total number of \mathcal{A} sending the commitments to honest parties and ϵ^{euf} denotes the maximum advantage of an adversary breaking unforgeability of $\text{pPRF} = (\text{ABM.gen}, \text{ABM.spl})$ on $\widehat{L}_{pk}^{\text{td}}$.*

Proof. Since BD occurs with the same probability in both games, we consider the probability in Hybrid Game 2. We construct the following algorithm B_0 that takes pk from ABM.gen and simulates the roles of \mathcal{S}_2 and $\mathcal{F}_{\text{MCOM}}^2$ perfectly, interacting \mathcal{Z} and \mathcal{A} , by having access to oracle $\text{ABM.spl}(pk, sk^{\text{spl}}, \cdot)$ as follows:

In the case when P_i is honest: In the commit phase when \mathcal{Z} sends $(\text{commit.sid}, \text{ssid}, P_i, P_j, x)$ to $\mathcal{F}_{\text{MCOM}}^2$ (via honest P_i), B_0 submits $t = (\text{sid}, \text{ssid}, P_i, P_j)$ to $\text{ABM.spl}(pk, sk^{\text{spl}}, \cdot)$ to obtain u such that $(t, u) \in L_{pk}^{\text{td}}$. Then B_0 computes fake ciphertext $c \leftarrow \text{ABM.enc}^{(t,u)}(pk, x)$ as a commitment in the same way as \mathcal{S}_2 ($= \mathcal{S}_1$) does.

In the case where P_i is corrupted and P_j is honest: In the commit 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 $\mathcal{F}_{\text{MCOM}}^2$. Later, in the opening phase when corrupted P_i controlled by \mathcal{A} sends $(\text{sid}, \text{ssid}, x', r)$ to \mathcal{S}_2 as it expects to send to honest P_j , B_0 simply plays the role of $\mathcal{F}_{\text{MCOM}}^2$.

\mathcal{S}_2 uses sk^{spl} only when it computes $u \leftarrow \text{ABM.spl}(pk, sk^{\text{spl}}, t)$ in the commit phase when P_i is honest. B_0 instead may have access to oracle $\text{ABM.spl}(pk, sk^{\text{spl}}, \cdot)$, and simulates the roles of \mathcal{S}_2 and $\mathcal{F}_{\text{MCOM}}^2$ identically without knowing sk^{spl} .

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_A denotes the total number of \mathcal{A} sending the commitments to honest parties. We note that

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

The probability of B_i outputting $(t, u) \in \widehat{L}_{pk}^{\text{td}}$ is bounded by ϵ^{euf} . Therefore, we have $\Pr[\text{BD}] \leq q_A \epsilon^{\text{euf}}$. \square

By this lemma, we have

$$\{\text{HYBRID}^1_{\mathcal{F}_{\text{MCOM}}^1, \mathcal{S}_1^A, \mathcal{Z}}(\kappa, z)\}_{z \in \{0,1\}^{p(\kappa)}, \kappa \in \mathbb{N}} \stackrel{c}{\approx} \{\text{HYBRID}^2_{\mathcal{F}_{\text{MCOM}}^2, \mathcal{S}_2^A, \mathcal{Z}}(\kappa, z)\}_{z \in \{0,1\}^{p(\kappa)}, \kappa \in \mathbb{N}}$$

Hybrid Game 3: In this game, $\mathcal{F}_{\text{MCOM}}^3$ works exactly in the same way as $\mathcal{F}_{\text{MCOM}}^2$ does. \mathcal{S}_3 works exactly in the same way as \mathcal{S}_2 does except for **the case that P_i is honest in the commit phase**: In the commit phase when receiving $(\text{receipt}, \text{sid}, \text{ssid}, P_i, P_j, x)$ from $\mathcal{F}_{\text{MCOM}}^3$, \mathcal{S}_3 picks up $u = U_{pk}(t; R)$ with random R , instead of generating $u \leftarrow \text{ABM.spl}(pk, sk^{\text{spl}}, t)$ where $t = (\text{sid}, \text{ssid}, P_i, P_j)$. With an overwhelming probability, $(t, u) \in L_{pk}^{\text{td}}$. \mathcal{S}_3 then computes $c = \text{ABM.enc}^{(t,u)}(pk, x; r)$.

In case of adaptive corruption of P_i after the commit phase but before the decommit phase, \mathcal{S}_3 simply reveals (x, r, R) to \mathcal{A} .

We note that in Hybrid Game 2, \mathcal{S}_2 makes use of sk^{spl} only when it computes $u \leftarrow \text{ABM.spl}(pk, sk^{\text{spl}}, t)$, whereas in Hybrid Game 3, \mathcal{S}_3 does not use sk^{spl} any more. The difference of the views of \mathcal{Z} between

these two games is bounded by pseudo-randomness of $(\text{ABM.gen}, \text{ABM.spl})$, because we can construct a distinguisher D , using \mathcal{Z} and \mathcal{A} as oracle with having access to either of $\text{ABM.spl}(sk^{\text{spl}}, \cdot)$ or $U_{pk}(\cdot)$. When D has access to $\text{ABM.spl}(sk^{\text{spl}}, \cdot)$, it simulates Hybrid Game 2; otherwise, it simulates Hybrid Game 3. Therefore, we have

$$\{\text{HYBRID}^2_{\mathcal{F}_{\text{MCOM}}^2, \mathcal{S}_2^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}} \stackrel{c}{\approx} \{\text{HYBRID}^3_{\mathcal{F}_{\text{MCOM}}^3, \mathcal{S}_3^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}}.$$

Hybrid $_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{CRS}}}$ Game: This is the real world in the CRS model (or in the CRS hybrid model), where a honest party activated for the commitment functionality follows the code of the protocol in Fig. 3. The common reference string functionality \mathcal{F}_{CRS} parameterized by ABM.gen is given in Figure 5.

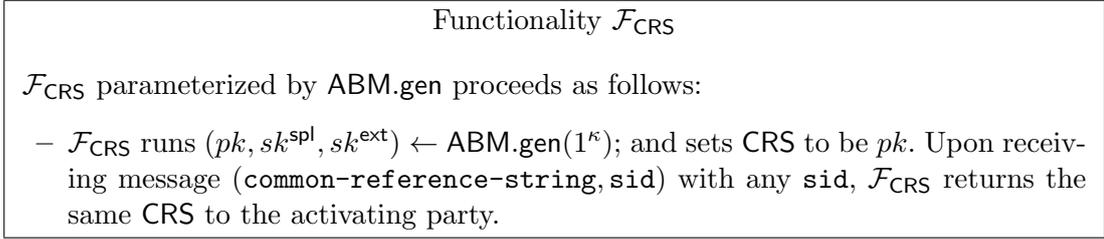


Fig. 5. The common reference string functionality

It is obvious by construction that two worlds are identical.

$$\{\text{HYBRID}^3_{\mathcal{F}_{\text{MCOM}}^3, \mathcal{S}_3^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}} \equiv \{\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}_{\pi, \mathcal{A}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}}.$$

In the end, we have

$$\{\text{IDEAL}_{\mathcal{F}_{\text{MCOM}}^{\mathcal{A}}, \mathcal{S}^{\mathcal{A}}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}} \stackrel{c}{\approx} \{\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}_{\pi, \mathcal{A}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0,1\}^{p(\kappa)}}.$$

■

C PPRF from Damgård-Jurik PKE

In this section, we provide the formal proof of Lemma 1. Although the proof is implicitly shown in [31], we provide it for completeness.

To prove the statement, we require two more assumptions related to DJ PKE, along with the standard DCR assumption, called the **non-multiplication assumption** and the **non-trivial divisor assumption**, which originally appeared in [31]. We first prove that our target scheme is a PPRF with unforgeability on L_{pk}^{td} (not on $\widehat{L}_{pk}^{\text{td}}$) under the DCR assumption and the non-multiplication assumption. We prove this in a generalized case that DJ PKE is replaced with an arbitrary enhanced additive homomorphic encryption scheme. We then prove that the resulting scheme has unforgeability on \widehat{L}_{pk} , additionally assuming the non-divisor assumption.

C.1 Assumptions and Some Useful Lemmas

Let us write $\Pi^{(d)}$ to denote DJ PKE with parameter d .

Assumption 5. We say that the DCR assumption holds if for every PPT A , there exists a key generation algorithm \mathbf{K} such that $\text{Adv}_A^{\text{dcr}}(\kappa) =$

$$\Pr[\text{Expt}_A^{\text{dcr}-0}(\kappa) = 1] - \Pr[\text{Expt}_A^{\text{dcr}-1}(\kappa) = 1]$$

is negligible in κ , where

$$\begin{array}{l|l} \text{Expt}_A^{\text{dcr}-0}(\kappa) : & \text{Expt}_{d,A}^{\text{dcr}-1}(\kappa) : \\ \left. \begin{array}{l} n \leftarrow \mathbf{K}(1^\kappa); R \xleftarrow{\cup} \mathbb{Z}_{n^2}^\times \\ c = R^n \bmod n^2 \\ \text{return } A(n, c). \end{array} \right\} & \left. \begin{array}{l} n \leftarrow \mathbf{K}(1^\kappa); R \xleftarrow{\cup} \mathbb{Z}_{n^2}^\times \\ c = (1+n)R^n \bmod n^2 \\ \text{return } A(n, c). \end{array} \right\} \end{array}$$

Assumption 6 ([31]). We say that the non-trivial divisor assumption holds on $\Pi^{(d)}$ if for every PPT A , $\text{Adv}_{A,\Pi^{(d)}}^{\text{divisor}}(\kappa) = \text{negl}(\kappa)$ where

$$\text{Adv}_{A,\Pi^{(d)}}^{\text{divisor}}(\kappa) = \Pr[(pk, sk) \leftarrow \mathbf{K}(1^\kappa); A(n) = c : 1 < \gcd(\mathbf{D}(c), n) < n].$$

This assumes that an adversary cannot compute an encryption of a non-trivial divisor of n , i.e., $\mathbf{E}(p)$, under given public-key pk_{dj} only. Since the adversary is only given pk_{dj} , the assumption is plausible.

Lemma 4. If A is an adversary against $\Pi^{(d)}$, there is adversary A' against $\Pi^{(1)}$ such that

$$\text{Adv}_{A,\Pi^{(d)}}^{\text{divisor}}(\kappa) \leq \text{Adv}_{A',\Pi^{(1)}}^{\text{divisor}}(\kappa).$$

Assumption 7 ([31]). We say that the non-multiplication assumption holds on DJ PKE $\Pi^{(d)}$ if for every PPT adversary A , the advantage of A , $\text{Adv}_{A,\Pi^{(d)}}^{\text{mult}}(\kappa) = \text{negl}(\kappa)$, where

$$\text{Adv}_{A,\Pi^{(d)}}^{\text{mult}}(\kappa) = \Pr[(pk, sk) \leftarrow \mathbf{K}(1^\kappa); c_1, c_2 \leftarrow \mathbb{Z}_{n^{d+1}}^\times; c^* \leftarrow A(pk, c_1, c_2) : \mathbf{D}_{sk}(c^*) = \mathbf{D}_{sk}(c_1) \cdot \mathbf{D}_{sk}(c_2)].$$

This assumes that an adversary cannot compute $\mathbf{E}(x_1 \cdot x_2)$ for given $(pk_{\text{dj}}, \mathbf{E}(x_1), \mathbf{E}(x_2))$. If the multiplicative operation is easy, DJ PKE turns out a fully-homomorphic encryption (FHE), which is unlikely. Although breaking the non-multiplication assumption does not mean that DJ PKE turns out a FHE, this connection gives us some feeling that this assumption is plausible.

Lemma 5. If A is an adversary against DJ PKE $\Pi^{(d)}$, there is an adversary A' against $\Pi^{(1)}$ such that

$$\text{Adv}_{A,\Pi^{(d)}}^{\text{mult}}(\kappa) \leq \text{Adv}_{A',\Pi^{(1)}}^{\text{mult}}(\kappa).$$

Lifting-Up and Re-Randomization. We give very useful lemmas below, which are implicitly used in [19] to prove that $\Pi^{(d)}$ for any $d \geq 1$ is IND-CPA secure under the DCR assumption. In order to prove Lemmas, 4 and 5, these lemmas are essential.

Lemma 6 (from [19, 31]). Let n be a public key of both DJ PKE $\Pi^{(d)}$, where $d \geq 1$, and DJ PKE $\Pi^{(1)}$. We let $\tau : \mathbb{Z}_{n^2}^\times \rightarrow \mathbb{Z}_{n^{d+1}}^\times$ be the canonical embedding map defined by $\tau(c) = c \bmod n^{d+1}$ where $c \in \mathbb{Z}_{n^2}^\times$ is canonically interpreted as an integer in $\{0, \dots, n^2 - 1\}$. We let $\pi : \mathbb{Z}_{n^{d+1}}^\times \rightarrow \mathbb{Z}_{n^2}^\times$ be the canonical homomorphism defined by $\pi(\hat{c}) = \hat{c} \bmod n^2$ where $\hat{c} \in \mathbb{Z}_{n^{d+1}}^\times$ is canonically interpreted as an integer in $\{0, \dots, n^{d+1} - 1\}$. We then have:

- $\pi \circ \tau$ is the identity map over $\mathbb{Z}_{n^2}^\times$.
- For every $c \in \mathbb{Z}_{n^2}^\times$, $\mathbf{D}^{(1)}(c) \equiv \mathbf{D}^{(d)}(\tau(c)) \pmod{n}$.
- For every $\hat{c} \in \mathbb{Z}_{n^{d+1}}^\times$, $\mathbf{D}^{(1)}(\pi(\hat{c})) \equiv \mathbf{D}^{(d)}(\hat{c}) \pmod{n}$.

Based on Lemma 6, we have the following lemma.

Lemma 7 (from [19, 31]). *There is an algorithm B that takes any public-key $pk = (n, d)$ ($d > 1$) and any ciphertext $c \in \mathbb{Z}_{n^2}^\times$ for $\Pi^{(1)}$, and efficiently samples random $\hat{c} \in \mathbb{Z}_{n^{d+1}}^\times$ conditioned on $\mathbf{D}^{(1)}(\pi(\hat{c})) = \mathbf{D}^{(1)}(c) \pmod{n}$.*

Proof. B is constructed as follows: Given $c \in \mathbb{Z}_{n^2}^\times$, choose random $y \xleftarrow{U} \{0, 1, \dots, n^{d-1} - 1\}$; set $\hat{c} = \tau(c) \cdot \mathbf{E}^{(d)}(yn)$; output \hat{c} .

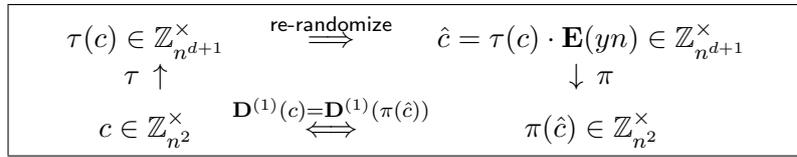


Fig. 6. Diagram of Lifting up and Re-Randomization

C.2 Proof of Lemmas, 4 and 5

By using algorithm B , random instances given to adversary A are converted into proper random instances given to adversary A' . Letting the output of A' be \hat{c} , we output $\pi(\hat{c})$ as the output of A , which obtains the lemmas, 4 and 5.

C.3 PPRF from Waters Signature on Additively Homomorphic Encryptions

We define enhanced additive homomorphic encryptions, which is a generalization of Damgård-Jurik PKE.

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 $x_1, \dots, x_n \in X$. 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 8 (Non-Multiplication 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 , $\text{Adv}_A^{\text{mult}}(\kappa) = \text{negl}(\kappa)$, where $\text{Adv}_A^{\text{mult}}(\kappa) \triangleq$*

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

This assumption is a generalized version of Assumption 7.

We now construct a PPRF $(\text{Gen}_{\text{spl}}, \text{Spl})$. 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 $h_0, h_1, \dots, h_\kappa \in Y$. Let us define $H(t) = h_0 \star \prod_{i=1}^\kappa h^{t[i]} \in Y$, where $t = (t[1], \dots, t[\kappa]) \in \{0, 1\}^\kappa$ is the bit representation of t . Let us define $L_{pk}(t)$ such that

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

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

- **Gen** (1^κ) : It runs $\mathbf{K}(1^\kappa)$ and obtain (pk, sk) . It generates $x_1, x_2 \leftarrow X$ and $h_0, h_1, \dots, h_\kappa \leftarrow Y$ 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 $PK = (pk, g_1, g_2, h_0, \dots, h_\kappa)$ and $SK = (PK, d)$.
- **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 9. *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 $(\text{Gen}_{\text{spl}}, \text{Spl})$ is a PPRF with unforgeability on L_{pk} .*

Proof. The proof of pseudo randomness is almost straightforward: 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 when it decides that it has access to a PPRF. We run S on pk . It picks up at random $x_1, x_2 \leftarrow X$, $h_0, h_1, \dots, h_\kappa \leftarrow Y$, 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 $m_0 = 0$, 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 $PK = (pk, g_1, g_2, \mathbf{h})$, where $\mathbf{h} = (h_0, h_1, \dots, h_\kappa)$. For any query t , the simulator picks up random $r \leftarrow X$ and returns (u_r, u_t) such that $u_r = \mathbf{E}_{pk}(r)$ and $u_t = \mathbf{E}_{pk}(m_b) \star (H(t))^r$. Finally, the simulator outputs the same bit that A outputs.

When $b = 0$, (u_r, u_t) is computationally indistinguishable from a uniform distribution over Y^2 , because $\mathbf{E}_{pk}(0)$ is computationally indistinguishable from a uniform distribution over Y . On

the other hand, when $b = 1$. Since S outputs the same bit that A outputs, $\text{Adv}_{\Pi}^{\text{ind-cpa}} S(\kappa) = \Pr[S = 1 | b = 1] - \Pr[S = 1 | b = 0] = \Pr[A = 1 | b = 1] - \Pr[A = 1 | b = 0] = \text{Adv}_{\text{pprf}} A(\kappa)$. Therefore, $\text{Adv}_{\text{pprf}} A(\kappa) = \text{Adv}_{\Pi}^{\text{ind-cpa}} S(\kappa) = \text{negl}(\kappa)$.

The proof of unforgeability on this scheme is substantially similar to that in [7, 45, 4]. We provide a sketch of the proof.

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

In game G_1 , we modify the choice of \mathbf{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 $a_0, a_1, \dots, a_\kappa \leftarrow \mathbb{Z}/n\mathbb{Z}$. It picks up $y_1, \dots, y_\kappa \leftarrow [0, \dots, (\theta - 1)]$ and $y_0 \in [0, \dots, \kappa(\theta - 1)]$. It finally outputs $PK = (pk, g_1, g_2, \mathbf{h})$, by setting $h_i = g^{a_i} g_2^{y_i}$ for $i \in [0, \dots, \kappa]$. 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 \mathbf{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 $\gamma_{\mathbf{y}} : (\{0, 1\}^\kappa)^{q_s+1} \rightarrow \{0, 1\}$ be a predicate such that $\gamma_{\mathbf{y}}(\mathbf{t}) = 1$ if and only if $y(t_0) = 0$ and $\bigwedge_{i=1}^{q_s} y(t_i) \neq 0$, where $\mathbf{t} = (t_0, \dots, t_{q_s}) \in (\{0, 1\}^\kappa)^{q_s+1}$. Let $Q(\mathbf{t})$ be the event that at the end of game G_1 , adversary A queries, t_1, \dots, t_{q_s} and outputs t_0 as the target message, on which A tries to generate the output of $\text{Spl}(sk, t_0)$.

We now borrow the following lemmas due to [4].

Lemma 8. [4]. *Let $Q(\mathbf{t})$ be the event in game G_1 mentioned above. Then,*

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

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

Lemma 9. [4]. *Let n, θ, κ be positive integers, such that $\kappa\theta < n$. Let $y_0, y_1, \dots, y_\kappa$ 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 $t_0, \dots, t_\kappa \in \{0, 1\}^\kappa$, we have*

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

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

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

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 \mathbf{a} and \mathbf{y} as in game G_2 . When A queries t , it picks up $r' \leftarrow X \simeq \mathbb{Z}/n\mathbb{Z}$ and selects $u_r \leftarrow g_1^{-\frac{1}{y(t)}} \star \mathbf{E}_{pk}(r')$ and $u_t \leftarrow g_1^{-\frac{a(t)}{y(t)}} \star \mathbf{E}_{pk}(0) \star (H(t))^{r'}$.

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

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

Therefore, the righthand side is $g_1^{-\frac{a(t)}{y(t)}} \star \mathbf{E}_{pk}(0; v) \star (H(t))^{r'}$ for some $v \in R$. This is substantially equivalent to the technique of all-but-one simulation technique in [7]. As in game G_2 , the simulator always abort if $\gamma_{\mathbf{y}}(t) = 1$ 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 has now

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

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 . \blacksquare

C.4 Proof of Lemma 1

We now complete the proof of Lemma 1. We note that we have already shown in Theorem 9 that the proposed scheme is unforgeable on L_{pk}^{td} under Assumption 5 and Assumption 7, since Assumption 8 is a generalized version of Assumption 7. We now show the following.

Lemma 1 (restated) $\text{pPRF} = (\text{ABM.gen}, \text{ABM.spl})$ is a PPRF with unforgeability on $\widehat{L}_{pk}^{\text{td}}$ as defined above, under the assumptions, 5, 6 and 7.

Proof. Let $\text{pPRF} = (\text{ABM.gen}, \text{ABM.spl})$ be defined on $\Pi^{(d)}$. For pk generated by ABM.gen and integer $f \geq 1$, we let

$$L_{pk}^{(f)} := \left\{ (t, (u_r, u_t)) \mid \mathbf{D}(u_t) \equiv x_1 x_2 + y(t) \mathbf{D}(u_r) \pmod{n^f} \right\},$$

where \mathbf{D} is the decryption algorithm of $\Pi^{(d)}$. By construction, it is clear that $L_{pk}^{(d)} = L_{pk}^{\text{td}}$. We note that $L_{pk}^{\text{td}} \subset L_{pk}^{(1)}$. We remark that $\widehat{L}_{pk}^{\text{td}}$ is the union of disjoint sets, $L_{pk}^{(1)}$ and L_{divisor} such that

$$L_{\text{divisor}} := \left\{ (t, (u_r, u_t)) \mid 1 < \gcd\left(\mathbf{D}\left(\frac{u_t}{g_1^{x_2} u_r^{y(t)}}\right), n\right) < n \right\}.$$

We first show that our target PPRF has unforgeability on $L_{pk}^{(1)}$. In the proof of Theorem 9, we change the proof as follows: In the final game, the simulator instead takes $(pk_{\text{dj}}, c_1, c_2)$ where $pk_{\text{dj}} = (n, 1)$ is a public key of DJ PKE $\Pi^{(1)}$ and (c_1, c_2) , where $c_i \in \mathbb{Z}_{n^2}^\times$, is an instance of the non-multiplication problem on $\Pi^{(1)}$. The simulator sets $pk'_{\text{dj}} := (n, d)$ and lifts up (c_1, c_2) to $(g_1, g_2) \in (\mathbb{Z}_{n^{d+1}}^\times)^2$ using algorithm B in Lemma 6. Then the simulator start game G_3 with $(pk'_{\text{dj}}, g_1, g_2)$ by playing the role of the challenger. When adversary A outputs $(t_0, (u_r, u_t)) \in L_{pk}^{(1)}$, the simulator can solve the non-multiplication problem on $\Pi^{(1)}$ by computing $\frac{u_t(t_0)}{u_r^{a(t_0)}} \pmod{n}$. Therefore, the probability of A outputting such pairs is negligible; otherwise, it contradicts Assumption 7.

We next prove that our target PPRF has unforgeability on L_{divisor} . We directly construct an algorithm C that breaks the non-trivial divisor assumption on $\Pi^{(d)}$. We let C take pk_{dj} from $\Pi^{(d)}$. Then, C sets up all public parameter consistent with pk_{dj} and the corresponding secret key except sk_{dj} . We note that C can sample (u_r, u_t) on arbitrary t under the public key, because sk_{dj} is not needed to sample (u_r, u_t) . C runs adversary A and finally obtain $(t^*, (u_r^*, u_t^*)) \in L_{\text{divisor}}$. Then, it outputs $c^* := \frac{u_t^*}{g_1^{x_2(u_r^*)^{y(t^*)}}}$. $(t^*, (u_r^*, u_t^*)) \in L_{\text{divisor}}$, means that $1 < \gcd(\mathbf{D}_{sk_{\text{dj}}}(c^*), n) < n$. Therefore, the probability that $(t^*, (u_r^*, u_t^*)) \in L_{\text{divisor}}$ is negligible; otherwise, it contradicts Assumption 6.

D Instantiation of ABME from Waters Signature based on DDH

In this section, we describe an ABME scheme from Waters based pPRF over a prime cyclic group without bilinear map. In the construction, the message length is restricted to $\lambda = \Omega(\log \kappa)$. Therefore, this scheme requires $\Omega(\kappa/\log \kappa)$ expansion factor.

D.1 PPRF from Waters Signature in Non-Bilinear Group of Prime Order

The idea behind the construction is to use Waters signature [45] 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 $t = (t_1, \dots, t_{\kappa}) \in \{0, 1\}^{\kappa}$ in which $t_i \in \{0, 1\}$ denotes i -th bit representation of string t .

- $\text{Gen}_{\text{spl}}(1^{\kappa})$ picks up $g, h_0, \dots, h_{\kappa} \xleftarrow{U} G$ and $x_1, x_2 \xleftarrow{U} \mathbb{Z}/q\mathbb{Z}$ to set $g_1 = g^{x_1}$, $g_2 = g^{x_2}$. It outputs $pk = (G, g, q, \lambda, g_1, g_2, h_0, \dots, h_{\kappa})$, and $sk = x_2$, where $U := G \times G$.
- $\text{Spl}(pk, sk, t; r)$ takes $t \in \{0, 1\}^{\kappa}$ and outputs $u = (u_r, u_t)$, by computing $u_r = g^r$ and $u_t = g_1^{x_2} (H(t))^r$ where $r \xleftarrow{U} \mathbb{Z}/q\mathbb{Z}$.

Theorem 10. *The above construction is pPRF under the DDH assumption.*

Proof. Spl is the same as Waters signature scheme when applied to a non-pairing group. So, unforgeability is immediately guaranteed if the computational DH assumption holds true. Pseudo-randomness also holds under the DDH assumption because $(g^r, H(t)^r)$ is computationally indistinguishable from two independent random elements in G : To explain more details, suppose that (g, \hat{g}, h, \hat{h}) is a tuple of four group elements in G , which is either a DDH instance or a random tuple. To break the DDH problem, a simulator sets $g_1 := g^{x_1}$, $g_2 := g^{x_2}$, $K := g^{x_1 x_2}$, and $h_i := \hat{g}^{a_i}$, where $x_1, x_2, a_0, \dots, a_{\kappa} \leftarrow G$. It then runs adversary A on the above parameters, where A is an adversary to break pseudo-randomness. For any query t , the simulator picks up random $s, v \leftarrow \mathbb{Z}/q\mathbb{Z}$ and returns (u_r, u_t) such that $u_r = g^s h^v$ and $u_t = K \cdot (\hat{g}^{a_0})^s (\hat{h}^{a_0})^v \prod_{i \geq 1} (\hat{g}^{a_i})^{s t_i} (\hat{h}^{a_i})^{v t_i}$. We note that $u_r = g^{s + \log_g(h)v}$ and $u_t = KH(t)^{s + \log_{\hat{g}}(\hat{h})v}$. Hence, (u_r, u_t) is a Waters signature if (g, \hat{g}, h, \hat{h}) is a DDH tuple; otherwise it is a pair of two random elements. The simulator outputs the same bit that A outputs. The simulator's advantage is the same as that of A . Under the DDH assumption, its advantage is bounded by a negligible (in κ) function. Therefore, it also satisfies pseudo-randomness. Hence, the scheme above is an instantiation of pPRF if the DDH assumption holds true. ■

D.2 ABME from Waters Signature without Pairing

We consider Waters signature [45] 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 $\mathbf{h} = (h_0, \dots, h_\kappa)$ with $h_j = g^{y_j}$, where $x_1, x_2, y_0, y_1, \dots, y_\kappa \leftarrow \mathbb{Z}/q\mathbb{Z}$. We write $t = (t_1, \dots, t_\kappa) \in \{0, 1\}^\kappa$ where $t_i \in \{0, 1\}$ ($i \in [\kappa]$). We let $y(t) = y_0 + \sum_{i=1}^\kappa t_i y_i \pmod{q}$ and define $H(t) = h_0 \prod_{i=1}^\kappa h_i^{t_i}$, that is, $H(t) = g^{y(t)}$. We let $U'_{pk} = \{0, 1\}^\kappa \times G^2$. Then we define the set of Waters signature under $pk = (g, g_1, g_2, \mathbf{h})$ as $L_{pk}^{\text{td}} = \{(t, u) \mid (t, u) \in \{0, 1\}^\kappa \times L_{pk}(t)\}$ such that

$$L_{pk}^{\text{td}}(t) = \{(u_v, u_t) \mid \exists(x_2, v) : u_v = g^v, u_t = g_1^{x_2} H(t)^v, \text{ and } g_2 = g^{x_2}\}.$$

We let $\widehat{L}_{pk}^{\text{td}} = L_{pk}^{\text{td}}$ and so, $L^{\text{ext}} = U'_{pk} \setminus L_{pk}^{\text{td}}$. 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 ABME scheme as follows.

- **ABM.gen**(1^κ): It generates $g, (x_1, x_2)$, and $\mathbf{y} = (y_0, \dots, y_\kappa)$ independently and uniformly from the above domains, respectively. It then computes $g_1, g_2, \mathbf{h} = (h_0, \dots, h_\kappa)$ as above. It outputs $pk = (G, g, q, \lambda, g_1, g_2, \mathbf{h})$, $sk^{\text{spl}} = x_2$, and $sk^{\text{ext}} = (x_1, \mathbf{y})$, where $\lambda = \Omega(\log \kappa)$.
- **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 $u = (u_v, u_t)$.
- **ABM.enc**^(t, u)($pk, m; (z, s)$): To encrypt message $m \in \{0, 1\}^\lambda$, where $\lambda = \Omega(\log \kappa)$, it picks up $z, s \leftarrow \mathbb{Z}/q\mathbb{Z}$ independently, and then computes $A = g_1^z H(t)^s u_t^m$, $a = g^z g_2^m$, and $b = g^s u_v^m$. It outputs $c = (A, a, b)$ as ciphertext.
- **ABM.dec**^(t, u)(sk^{ext}, c) where $sk^{\text{ext}} = (x_1, \mathbf{y})$: To decrypt $c = (A, a, b)$, it searches $m \in \{0, 1\}^\lambda$ such that

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

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

- **ABM.col**₁^(t, u)(sk^{spl}, v) where $sk^{\text{spl}} = x_2$: 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 $\xi = (x_2, t, u, v, \omega, \eta)$.
- **ABM.col**₂(ξ, x): To open c to $x \in \{0, 1\}^\lambda$, it computes $z = \omega - mx_2 \pmod{q}$ and $s = \eta - mv \pmod{q}$ and outputs (z, s) .

We note that **ABM.enc** runs the simulation algorithm of a canonical sigma protocol on L_{pk}^{td} with message (challenge) m and **ABM.col** runs the real protocol of the sigma protocol on L_{pk}^{td} with witness (x_2, v) .

In the trap-door mode when $(t, u) \in L_{pk}^{\text{td}}$, 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 $m \in \{0, 1\}^\kappa$, the answer can be computed by $z = \omega - mx_2$ and $s = \eta - mv$. It is verified as $A = g_1^z H(t)^s u_t^m$, $a = g^z g_2^m$, and $b = g^s u_v^m$.

In the decryption mode when $(t, u) \in L_{pk}^{\text{ext}} (= U'_{pk} \setminus L_{pk}^{\text{td}})$, the first message (A, a, b) from the simulator for the above canonical sigma protocol commits to m 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, m) , it implies that

$$\begin{pmatrix} \log_g A \\ \omega \\ \eta \end{pmatrix} = \begin{pmatrix} x_1 & y(t) & x_1 x'_2 + y(t)v \\ 1 & 0 & x_2 \\ 0 & 1 & v \end{pmatrix} \begin{pmatrix} z \\ s \\ m \end{pmatrix}$$

Since $(t, u) \notin L_{pk}^{\text{td}}$, $x'_2 \neq x_2$ and hence, the determinant of the matrix above is non-zero and (z, s, m) is unique.

Notice that $x_1\omega + y(t)\eta - \log_g A = x_1(x_2 - x'_2)m$. Since $g_1^{x'_2} = u_t u_v^{-y(t)}$,

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

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

Since $(\text{ABM.gen}, \text{ABM.spl})$ is pPRF (under the DDH assumption), the proposed scheme is an ABME scheme.

Theorem 11. *The scheme as above is an ABME if the DDH assumption holds true.*

E All-But-Many Lossy Trapdoor Functions

We recall all-but-many lossy trapdoor functions (ABM-LTF) [31], by slightly modifying the notation to fit our purpose.

All-but-many lossy trapdoor function $\text{ABM.LTF} = (\text{ABM.gen}, \text{ABM.spl}, \text{ABM.eval}, \text{ABM.inv})$ consists of the following algorithms:

- **ABM.gen** is a PPT algorithm that takes 1^κ and outputs $(pk, sk^{\text{spl}}, sk^{\text{ext}})$, where pk defines a set U_{pk} . We let $U'_{pk} = \{0, 1\}^\kappa \times U_{pk}$. pk also determines two disjoint sets, L_{pk}^{loss} and L_{pk}^{inj} , such that $L_{pk}^{\text{loss}} \cup L_{pk}^{\text{inj}} \subset U'_{pk}$.
- **ABM.spl** is a PPT algorithm that takes (pk, sk^{spl}, t) , where $t \in \{0, 1\}^\kappa$, picks up inner random coins $v \leftarrow \text{COIN}^{\text{spl}}$, and computes $u \in U_{pk}$. We write $L_{pk}^{\text{loss}}(t)$ to denote the image of **ABM.spl** on t under pk , i.e.,

$$L_{pk}^{\text{loss}}(t) := \{u \in U_{pk} \mid \exists sk^{\text{spl}}, \exists v : u = \text{ABM.spl}(pk, sk^{\text{spl}}, t; v)\}.$$

We require $L_{pk}^{\text{loss}} = \{(t, u) \mid t \in \{0, 1\}^\kappa \text{ and } u \in L_{pk}^{\text{loss}}(t)\}$. We set $\widehat{L}_{pk}^{\text{loss}} := U'_{pk} \setminus L_{pk}^{\text{inj}}$. Since $L_{pk}^{\text{loss}} \cap L_{pk}^{\text{inj}} = \emptyset$, we have $L_{pk}^{\text{loss}} \subseteq \widehat{L}_{pk}^{\text{loss}} \subset U'_{pk}$.

- **ABM.eval** is a DPT algorithm that takes pk , (t, u) , and message $x \in \text{MSP}$ and computes $c = \text{ABM.eval}^{(t,u)}(pk, x)$, where MSP denotes the message space uniquely determined by pk .
- **ABM.inv** is a DPT algorithm that takes sk^{ext} , (t, u) , and c , and computes $x = \text{ABM.inv}^{(t,u)}(sk^{\text{ext}}, c)$.

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

1. **Adaptive All-but-many property:** $(\text{ABM.gen}, \text{ABM.spl})$ is a probabilistic pseudo random function (PPRF), as defined in Sec. 3.1, with *strongly* unforgeability on $\widehat{L}_{pk}^{\text{loss}} = U'_{pk} \setminus L_{pk}^{\text{inj}}$. Strong unforgeability in this paper is called *evasiveness* in [31].
2. **Inversion** For every $\kappa \in \mathbb{N}$, every $(pk, sk^{\text{spl}}, sk^{\text{ext}}) \in \text{ABM.gen}(1^\kappa)$, every $(t, u) \in L_{pk}^{\text{inj}}$, and every $x \in \text{MSP}$, it always holds that

$$\text{ABM.inv}^{(t,u)}(sk^{\text{ext}}, \text{ABM.eval}^{(t,u)}(pk, x)) = x.$$

3. **ℓ -Lossyness** For every $\kappa \in \mathbb{N}$, every $(pk, sk^{\text{spl}}, sk^{\text{ext}}) \in \text{ABM.gen}(1^\kappa)$, and every $(t, u) \in L_{pk}^{\text{loss}}$, the image set $\text{ABM.eval}^{(t,u)}(pk, \text{MSP})$ is of size at most $|\text{MSP}| \cdot 2^{-\ell}$.

Here L_{pk}^{loss} (resp. L_{pk}^{inj}) in ABM-LTFs corresponds to L_{pk}^{td} (resp. L_{pk}^{ext}) in ABMEs. We remark that ABM-LTFs [31] require that $(\text{ABM.gen}, \text{ABM.spl})$ should be *strongly* unforgeable, whereas ABMEs requires that $(\text{ABM.gen}, \text{ABM.spl})$ be only unforgeable.