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 [CF01,CLOS02], with *strict* expansion factor $O(\kappa)$; meaning that to commit λ bits, communication strictly requires $O(\lambda\kappa)$ bits, where κ denotes the security parameter. Efficient construction of a fully-equipped UC commitment scheme is a long-standing open problem. We introduce new abstraction, called *all-but-many encryption* (ABME), and prove that it captures fully-equipped UC commitment schemes. We propose the first fully-equipped UC commitment scheme with *optimal expansion factor* $O(1)$ from our ABME scheme related to the DCR assumption. We also provide an all-but-many lossy trapdoor function (ABM-LTF) [Hof12] from our DCR-based ABME scheme, with a better lossy rate than [Hof12].

Keywords: All-but-many encryptions; non-interactive, non-erasure, and adaptively UC secure commitment schemes in a single, global, and re-usable common reference string; and all-but-many lossy trapdoor functions.

1 Introduction

1.1 Motivating Application: Fully-Equipped UC Commitment

Universal composability (UC) framework [Can01] 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 are an essential ingredient to construct high level UC-secure protocols, which imply UC zero-knowledge protocols [CF01,DN02] and UC oblivious transfer [CLOS02], thereby meaning that 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 [CF01], 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 opens 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 saw many equivocal commitments generated by the simulator.

Several factors as shown below feature UC commitments:

Non-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 parities 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 [CF01] presented the first UC secure commitment schemes. One of their proposals is "fully-equipped" – *non-interactive and adaptively secure in the non-erasure model under a global re-usable common reference string.* By construction, however, the proposal strictly requires, to commit to λ-bit secret, $O(\lambda\kappa)$ communication bits and $O(\lambda)$ modular exponentiations. Canetti et al. [CLOS02] also proposed another fully-equipped UC commitment scheme only from (enhanced) trapdoor permutations, which is constructed in the similar framework as in [CF01] with general non-interactive zero-knowledge proofs. Hence, it is simply inefficient.

So far, these two have been the only known fully-equipped UC commitment schemes. The known subsequent constructions of UC commitments [DN02,DG03,CS03,NFT12,Lin11,FLM11] have improved efficiency, but *sacrifice* at least one or two requirements [1]. Efficient construction of a fully-equipped UC commitment scheme is a long-standing open problem.

1.3 Our Contribution

We introduce special tag-based public key encryption (Tag-PKE) that we call *all-but-many encryption* (ABME), and prove that it implies "fully-equipped" UC commitments. There are a lot of obstacles to study the UC framework, due to complicated definitions and proofs with many subtleties. Therefore, it is desirable to translate the essence of basic UC secure protocols into simple cryptographic primitives.

We propose a compact ABME scheme related to the DCR assumption and thereby the first fully-equipped UC commitment scheme with optimal expansion factor $O(1)$ – to commit λ bit, it requires $O(\kappa)$ bits and a constant number of modular exponentiations. We also present an ABME scheme from the DDH assumption with overhead $O(\kappa/\log\kappa)$, which is slightly better than the prior works (with $O(\kappa)$). We further present a fully-equipped UC commitment scheme from a *weak* ABME scheme under the general assumption (where (enhanced) trapdoor permutations exist), which is far more efficient than the previous work [CLOS02] under the same assumption.

We also present an all-but-many lossy trapdoor function (ABM-LTF) [Hof12] from our DCR-based ABME scheme, with a better lossy rate than [Hof12].

[1] Only [NFT12] and [FLM11] satisfy all but one requirement. [NFT12] does not satisfy CRS re-usability, whereas [FLM11] does not support the non-erasure model.

Our Approach: All-But-Many Encryption In an ABME scheme, a secret-key holding user (i.e., the simulator in the UC framework) can generate a *fake* ciphertext, which can be opened to any message with consistent randomness. On the other hand, it must be infeasible for a secret-key non-holding user (i.e., the adversary in the UC framework) (1) to distinguish a fake ciphertext from a *real* (honestly generated) ciphertext, even after the message and randomness are revealed, and (2) to produce a fake ciphertext (on a fresh tag) even given many fake ciphertexts.

To realize such a scheme, we divide its functionality into two primitives, called *probabilistic pseudo random functions* (PPRF) and *extractable sigma protocols* ($\mathsf{ext}\Sigma$). The former is a kind of a probabilistic version of a pseudo random function (family) in the public parameter model. The latter is special sigma (i.e., canonical 3-round public-coin HVZK) protocols [CDS94] with some extractability. The concept of extractable sigma protocols is not completely new. A weaker notion, called *weak* extractable sigma protocols, appears in [Fuj12] to construct a few (interactive) simulation sound trapdoor commitment (SSTC) schemes. See also [GPY03,MY04,Gen04] for SSTC. This paper requires a stronger notion and its realization, which employed in a different framework. If two primitives are successfully combined, an ABME scheme can be constructed. We discuss more in the following.

Probabilistic Pseudo-Random Function (PPRF). A $\mathsf{PPRF} = (\mathsf{Gen}^{\mathsf{spl}}, \mathsf{Spl})$ is a probabilistic version of a pseudo random function family in the public parameter model. $\mathsf{Gen}_{\mathsf{spl}}(1^\kappa)$ generates a pair of public-key/seed (pk, w), and A PPT algorithm Spl takes (pk, w, t) and outputs (or *samples*) $u \leftarrow \mathsf{Spl}(pk, w, t)$. Let $L_{pk}(t) = \{u | \exists (w, v) : u = \mathsf{Spl}(pk, w, t, v)\}$. Informally, a PPRF requires that (a) u looks pseudo-random on any t (*pseudo randomness*) and (b) it is infeasible for any adversary to find u^* in *some super set*, $\widehat{L}_{pk}(t^*)$, of $L_{pk}(t^*)$ on any fresh t^*, even after it has access to oracle $\mathsf{Spl}(pk, w, \cdot)$ (*unforgeability on* $\widehat{L}_{pk}$), where $\widehat{L}_{pk} := \{(t, u) \,|\, u \in \widehat{L}_{pk}(t)\}$. The purpose of thinking a super set $\widehat{L}_{pk}$ will be clear later.

Extractable Sigma Protocols. An extractable sigma protocol is a special sigma protocol *associated with a language-generation algorithm and a decryption algorithm.* Recall the sigma protocols [CDS94]. A sigma protocol Σ on NP language L is a canonical 3-round public coin interactive proof system such that the prover can convince the verifier that he knows the 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 the conversation (a, e, z) on x. A sigma protocol is associated with a simulation algorithm $\mathsf{sim}\Sigma$ that takes x (regardless of whether $x \in L$ or not) and challenge e, and produces an accepting conversation $(a, e, z) \leftarrow \mathsf{sim}\Sigma(x, e)$ *without witness* w. It is guaranteed that, if $x \in L$, the distributions (a, e, z) produced by $\mathsf{sim}\Sigma(x, e)$ 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 unique e if there is an accepting conversation (a, e, z), which is called *special soundness.*

An extractable sigma protocol $\mathsf{ext}\Sigma = (\mathsf{Gen}^{\mathsf{ext}}, \Sigma, \mathsf{Dec})$ uses two more algorithms than an ordinary sigma protocol: The language-generation algorithm $\mathsf{Gen}^{\mathsf{ext}}$ outputs a pair of public/secret keys, (pk, sk), where pk determines two disjoint sets L_{pk} and L^{ext}_{pk}. Here the sigma protocol Σ works on L_{pk} and the decryption algorithm Dec works on L^{ext}_{pk}, meaning that $\mathsf{Dec}(sk, x, a)$ outputs challenge e if $x \in L^{\mathsf{ext}}_{pk}$ and if an accepting conversation (a, e, z) exists on x. Due to special soundness, e is uniquely determined if $x \notin L_{pk}$. Therefore, the decryption algorithm is well defined.

Combining them. Suppose that an extractable sigma protocol and a PPRF are so well combined that, for $(L_{pk}, L^{\mathsf{ext}}_{pk})$ generated by $\mathsf{Gen}^{\mathsf{ext}}$, L_{pk} is the language derived from the relation of PPRF, and PPRF is unforgeable on $\widehat{L}_{pk}$ $(:= U'_{pk} \backslash L^{\mathsf{ext}}_{pk})$, where U'_{pk} denotes the entire set with respects to pk. We can then convert the extractable sigma protocol on L_{pk} into an ABME scheme in the similar way that a sigma protocol can be converted into an instance-dependent commitment scheme [BMO90,IOS94]. To encrypt message e on tag t, a sender picks random u, runs $\mathsf{sim}\Sigma$ on instance (t, u) with challenge e, to get $(a, e, z) \leftarrow \mathsf{sim}\Sigma(pk, (t, u), e)$, and finally outputs (t, u, a). Due to unforgeability of PPRF, it holds that $(t, u) \in U'_{pk} \backslash \widehat{L}_{pk}$ with an overwhelming probability. Then, e is uniquely determined given $((t, u), a)$, as long as an accepting conversation (a, e, z) exists on (t, u). By our precondition, we can decrypt (t, u, a) using sk, as $e = \mathsf{Dec}(sk, (t, u), a)$ because $(t, u) \in L^{\mathsf{ext}}_{pk}$. On the other hand, a fake ciphertext on tag t is produced using (w, v) as follows: one sets $u := \mathsf{Spl}(pk, w, t; v)$, with random v, where $(t, u) \in L_{pk}$, and computes a, as similarly as an honest prover computes the first message on common input (t, u) with witness (w, v). To open a to e, he produces the third message z in the sigma protocol. It is obvious by construction that he can open a to any e because $(t, u) \in L_{pk}$.

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. The following is our key observation to realize the functionality. 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) square 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 case to explain how to extract the challenge. Let $(g_1, g_2, h_1, h_2) \notin L_{pk}$, meaning that $x_1 \neq x_2$ where $x_1 := \log_{g_1}(h_1)$ and $x_2 := \log_{g_2}(h_2)$. The first message (A_1, A_2) of a canonical sigma protocol on L_{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., $\beta_1 a_1 + \beta_2 a_2$, 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 out e by checking whether $g_1^e = A_1^{\beta_1} A_2^{\beta_2}$ or not. In a more realistic case when 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 some case, the decryption algorithm might be luckily capable of inverting homomorphic maps such as $f(a) = g^a$, using trapdoor f^{-1}. It can then obtain (a_1, a_2) as well as the entire values of the matrix and hence extract the entire (polynomial-length) e. This corresponds to our DCR based implementation. However, the corresponding linear system is defined not on a finite field but on a finite ring, such as $\mathbb{Z}_{n^d}$. In that case, there is a super set of L_{pk}, say $\widehat{L}_{pk}$, such that the underlying matrix is invertible if and only if $x \notin \widehat{L}_{pk}$. We then require unforgeability not on L_{pk} but on super set $\widehat{L}_{pk}$, so that we can force an adversary to output $x = (t, u)$ in $L_{pk}^{\mathsf{ext}} = U'_{pk} \backslash \widehat{L}_{pk}$.

Acutual Instantiations. We present ABME schemes from three different types of PPRFs. We propose a PPRF from Waters signature scheme [Wat05] defined over a ring *equipped with no bilinear map.* As the associated homomorphic map, we employ Damgård-Jurik (DJ) PKE [DJ01]. The output of Waters based PPRF looks a pseudo random due to semantic security of DJ PKE. The construction inherits *unforgeability* (i.e., Condition (b) above) 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 super set of the language derived from the Waters-like PPRF. We can construct an extractable sigma protocol on it. Since the homomorphic map is invertible with a secret key of DJ PKE, we can obtain a compact ABME scheme and hence the first fully-equipped UC commitment scheme with optimal expansion factor $O(1)$. An alternative construction of 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 [CKS08] and construct an ABME scheme from the PPRF under the DDH assumption. Although the expansion factor of this scheme is $O(\kappa/\log\kappa)$, it is still better than the prior works (with $O(\kappa)$). This scheme has a short public key. We can construct a 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. The second and third schemes appear in Appendices, D and E.

1.4 Other Related Works

Simulation-based selective opening CCA (SIM-SO-CCA) secure PKE [FHKW10] is related to ABME, but both are incomparable. Indeed, the SIM-SO-CCA secure PKE scheme proposed in [FHKW10] does not satisfy the requirements of ABME. On the other hand, ABME does not satisfy SIM-SO-CCA PKE, because it does not support CCA security. Although the scheme of [FHKW10] could be tailored to a fully-equipped UC commitment scheme, it cannot overcome the barrier of expansion factor $O(\kappa)$, because it strictly costs $O(\lambda\kappa)$ bits to encrypt λ bit.

Hofheinz has presented the notion of all-but-many lossy trapdoor function (ABM-LTF) [Hof12], mainly to construct indistinguishable-based selective opening CCA (IND-SO-CCA) secure PKE. ABM-LTF is lossy trapdoor function (LTF) [PW08] with (unbounded) *many* lossy tags. The relation between ABM-LTF and ABME is a generalized analogue of LTF and lossy encryption [PVW08,BHY09] with unbounded many loss tags. However, unlike the other primitives, ABME always enjoys an *efficient* "opening" algorithm that can open a ciphertext on a "lossy" tag to any message with consistent randomness. Hofheinz has proposed two instantiations. One is related to the DCR assumption and the other is based on pairing groups of a composite order. In the DCR-based ABM-LTF, lossy tags are an analogue of Waters signatures defined in DJ PKE. Such tags are carefully embedded in a matrix so that the matrix can be non-invertible if tags are lossy; otherwise invertible. We were inspired by the lossy tag idea and have generalized it as PPRF. In the latest e-print version [Hof12], Hofheinz has shown that his DCR-based ABM-LTF can be converted to SIM-SO-CCA PKE. To realize this, an opening algorithm for ABM-LTF is needed, and he converted his DCR-based ABM-LTF into one with an opening algorithm, by sacrificing efficiency. We note that ABM-LTF with an opening algorithm meets the notion of ABME. We will show in Sec. 8 how Hofheinz's DCR-based ABM-LTF is converted to an ABME scheme. Its expansion factor is $O(1)$. However, compared to our DCR-based ABME scheme in Sec. 7, Hofheinz's ABM-LTF based ABME scheme is rather inefficient 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 $d \geq 1$. We compare them in Sec. 8. We remark that Hofheinz has not shown that his DCR-based ABM-LTF can be converted to a UC commitment scheme.

2 Preliminaries

For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. We denote by O and ω the standard notations to classify the growth of functions. We let $\mathsf{negl}(\kappa)$ to denote an unspecified function $f(\kappa)$ such that $f(\kappa) = \kappa^{-\omega(1)}$, saying that such a function is negligible in κ. We write PPT and DPT algorithms to denote probabilistic polynomial-time and deterministic poly-time algorithms, respectively. For PPT algorithm A, we write $y \leftarrow A(x)$ to denote the experiment of running A for given x, picking inner coins r uniformly from an appropriate domain, and assigning the result of this experiment to the variable y, i.e., $y = A(x;r)$. Let $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$ be probability ensembles such that each X_κ and Y_κ are random variables ranging over $\{0,1\}^\kappa$. The (statistical) distance between X_κ and Y_κ is $\mathsf{Dist}(X_\kappa, Y_\kappa) \triangleq \frac{1}{2} \cdot |\Pr_{s\in\{0,1\}^\kappa}[X = s] - \Pr_{s\in\{0,1\}^\kappa}[Y = s]|$. We say that two probability ensembles, X and Y, are statistically indistinguishable (in κ), denoted $X \stackrel{\mathrm{s}}{\approx} Y$, if $\mathsf{Dist}(X_\kappa, Y_\kappa) = \mathsf{negl}(\kappa)$. We say that X and Y are computationally indistinguishable (in κ), denoted $X \stackrel{\mathrm{c}}{\approx} Y$, if for every non-uniform PPT D (ranging over $\{0,1\}$), $\{D(1^\kappa, X_\kappa)\}_{\kappa\in\mathbb{N}} \stackrel{\mathrm{s}}{\approx} \{D(1^\kappa, Y_\kappa)\}_{\kappa\in\mathbb{N}}$. Let A and B be PPT algorithms that both take $x \in \{0,1\}^*$. We write $\{A(x)\}_{x\in\{0,1\}^\kappa,\kappa\in\mathbb{N}} \stackrel{\mathrm{s}}{\approx} \{B(x)\}_{x\in\{0,1\}^\kappa,\kappa\in\mathbb{N}}$ to denote $\{A(x_n)\}_{n\in\mathbb{N}} \stackrel{\mathrm{s}}{\approx} \{B(x_n)\}_{n\in\mathbb{N}}$ for every sequence $\{x_n\}_{n\in\mathbb{N}}$ such that $|x_n| = n$.

3 Building Blocks: Definitions

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

3.1 Probabilistic Pseudo Random Function (PPRF)

$\mathsf{PPRF} = (\mathsf{Gen}^{\mathsf{spl}}, \mathsf{Spl})$ consists of the following two algorithms:

- $\mathsf{Gen}^{\mathsf{spl}}$, the key generation algorithm, is a PPT algorithm that takes 1^κ as input, creates pk and picks up $w \leftarrow \mathsf{KSP}^{\mathsf{spl}}_{pk}$ to outputs (pk, w), where pk uniquely determines $\mathsf{KSP}^{\mathsf{spl}}_{pk}$.

– Spl, the sampling algorithm, is a PPT algorithm that takes (pk, w) and $t \in \{0,1\}^\kappa$, picks up inner random coins $v \leftarrow \mathsf{COIN}^{\mathsf{spl}}$, and outputs u.

Here we require that pk determines set U_{pk}. Let us define $U'_{pk} = \{0,1\}^\kappa \times U_{pk}$, $L_{pk}(t) = \{u \in U_{pk} \mid \exists\, w, \exists\, v : u = \mathsf{Spl}(pk, w, t; v)\}$, and $L_{pk} = \{(t, u) \mid t \in \{0,1\}^\kappa \text{ and } u \in L_{pk}(t)\}$. We are only interested in the case that L_{pk} is relatively small in U'_{pk}, in order to avoid sampling from U'_{pk} by chance. We require that PPRFs satisfy the following security requirements:

Efficiently samplable and explainable domain: For every pk given by $\mathsf{Gen}^{\mathsf{spl}}$, set U is efficiently samplable and explainable [FHKW10], that is, there is an efficient sampling algorithm on U that takes pk and random coins R and outputs u uniformly from U_{pk}. In addition, for every $u \in U_{pk}$, there is an efficient explaining algorithm that takes pk and u and outputs random coins R behind u, where R is uniformly distributed subject to $sample(U_{pk}; R) = u$.

Pseudo randomness: Any adversary A, given pk generated by $\mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$, cannot distinguish whether it has had access to $\mathsf{Spl}(pk, w, \cdot)$ or $U(\cdot)$. Here U is the following oracle: If $\mathsf{Spl}(pk, w, \cdot)$ is a deterministic algorithm, $U : \{0,1\}^\kappa \to U_{pk}$ is a random oracle. (Namely, it returns the same (random) value on the same input.) If $\mathsf{Spl}(pk, w, \cdot)$ is probabilistic, then $U(\cdot)$ picks up a fresh randomness $u \stackrel{\mathsf{U}}{\leftarrow} U_{pk}$ for each query t. We say that PPRF is **pseudo random** if, for all non-uniform PPT A, $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PPRF},A}(\kappa) = \left|\Pr[\mathsf{Expt}^{\mathsf{prf}}_{\mathsf{PPRF},A}(\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{prf}}_{U,A}(\kappa) = 1]\right|$ is negligible in κ, where

$\mathsf{Expt}^{\mathsf{prf}}_{\mathsf{PPRF},A}(\kappa)$:	$\mathsf{Expt}^{\mathsf{prf}}_{U,A}(\kappa)$:
$(pk, w) \leftarrow \mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$	$(pk, w) \leftarrow \mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$
$b \leftarrow A^{\mathsf{Spl}(pk,w,\cdot)}(pk)$	$b \leftarrow A^{U(\cdot)}(pk)$
return b.	return b.

Unforgeability: Let $\widehat{L}_{pk}(t)$ be some super set 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 $\mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$ and may have access to $\mathsf{Spl}(pk, w, \cdot)$. 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 non-uniform PPT A, $\mathsf{Adv}^{\mathsf{euf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}(\kappa) = \Pr[\mathsf{Expt}^{\mathsf{euf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}(\kappa) = 1]$ (where $\mathsf{Expt}^{\mathsf{euf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}$ 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 non-uniform PPT A, $\mathsf{Adv}^{\mathsf{seuf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}(\kappa) = \Pr[\mathsf{Expt}^{\mathsf{seuf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}(\kappa) = 1]$ (where $\mathsf{Expt}^{\mathsf{seuf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}$ is defined in Fig. 1) is negligible in κ.

We remark that (strong) unforgeability implies (1) that $\widehat{L}_{pk}$ should be small enough in U'_{pk} to avoid sampling from $\widehat{L}_{pk}$ by chance, and (2) that, if Spl is a DPT algorithm and $\widehat{L}_{pk} = L_{pk}$, it is implied by pseudo randomness.

$\mathsf{Expt}^{\mathsf{euf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}(\kappa)$:	$\mathsf{Expt}^{\mathsf{seuf}\text{-}\widehat{L}}_{\mathsf{PPRF},A}(\kappa)$:
$(pk, w) \leftarrow \mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$	$(pk, w) \leftarrow \mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$
$(t^*, u^*) \leftarrow A^{\mathsf{Spl}(pk,w,\cdot)}(pk)$	$(t^*, u^*) \leftarrow A^{\mathsf{Spl}(pk,w,\cdot)}(pk)$
If t^* has not been queried	$(t^*, u^*) \notin \mathcal{QA}$
and $u^* \in \widehat{L}_{pk}(t^*)$,	and $u^* \in \widehat{L}_{pk}(t^*)$,
return 1; otherwise 0.	return 1; otherwise 0.

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

3.2 Extractable Sigma Protocol

An extractable sigma protocol, $\mathsf{ext}\Sigma = (\mathsf{Gen}^{\mathsf{ext}}, \mathsf{com}\Sigma, \mathsf{ch}\Sigma, \mathsf{ans}\Sigma, \mathsf{sim}\Sigma, \mathsf{Vrfy}, \mathsf{Dec})$ is a sigma protocol, associated with two algorithms, $\mathsf{Gen}^{\mathsf{ext}}$ and Dec, with the following properties.

- $\mathsf{Gen}^{\mathsf{ext}}$ is an PPT algorithm that takes 1^κ and outputs (pk, sk), such that pk defines the entire set U'_{pk}, and two sub disjoint sets, L_{pk} and L^{ext}_{pk}, i.e., $L_{pk} \cup L^{\mathsf{ext}}_{pk} \subset U'_{pk}$ and $L_{pk} \cap L^{\mathsf{ext}}_{pk} = \emptyset$. We also require that L_{pk} determines binary efficiently recognizable set R_{pk} such that $L_{pk} = \{x | \exists w : (x, w) \in R_{pk}\}$.
- $\mathsf{com}\Sigma$ is a PPT algorithm that takes pk and $(x, w) \in R_{pk}$, picks up inner coins r_a, and outputs a.
- $\mathsf{ch}\Sigma(pk)$ is a publicly-samplable set determined by pk.
- $\mathsf{ans}\Sigma$ is a DPT algorithm that takes (pk, x, r_a, e), where $e \in \mathsf{ch}\Sigma(pk)$, and outputs z.
- Vrfy is a DPT algorithm that accepts or rejects (pk, x, a, e, z).
- $\mathsf{sim}\Sigma$ is a PPT algorithm that takes (pk, x, e) and outputs $(a, e, z) = \mathsf{sim}\Sigma(pk, x, e; r_z)$, where $r_z \leftarrow \mathsf{COIN}^{\mathsf{sim}}$. *We additionally require that* $r_z = z$. Namely, $(a, e, r_z) = \mathsf{sim}\Sigma(pk, x, e; r_z)$.
- Dec is a DPT algorithm that takes (sk, x, a) and outputs e or $\perp$.

We require that $\mathsf{ext}\Sigma$ satisfies the following properties:

Completeness: For every $(pk, sk) \in \mathsf{Gen}^{\mathsf{ext}}(1^\kappa)$, every $(x, w) \in R_{pk}$, every r_a (in an appropriate specified domain) and every $e \in \mathsf{ch}\Sigma(pk)$, it always holds that $\mathsf{Vrfy}(x, \mathsf{com}\Sigma(x, w; r_a), e, \mathsf{ans}\Sigma(x, w, r_a, e))$ $= 1$.

Special Soundness: For every $(pk, sk) \in \mathsf{Gen}^{\mathsf{ext}}(1^\kappa)$, every $x \in U'_{pk} \backslash L_{pk}$ and every a, there is *unique* $e \in \mathsf{ch}\Sigma(pk)$ if there is an accepting conversation for a on x. We say that a pair of two different accepting conversations for the same a on x, i.e., (a, e, z) and (a, e', z'), with $e \neq e'$, is a *collision* on x.

Enhanced Honest-Verifier Statistical Zero-Knowledgeness (eHVSZK): For every $(pk, sk) \in \mathsf{Gen}^{\mathsf{ext}}(1^\kappa)$, every $(x, w) \in R_{pk}$, and every $e \in \mathsf{ch}\Sigma(pk)$, the following ensembles are statistically indistinguishable in κ:

$$\{\mathsf{sim}\Sigma(pk, x, e; r_z)\}_{(pk,sk)\in\mathsf{Gen}^{\mathsf{ext}}(1^\kappa),\,(x,w)\in R_{pk},\,e\in\mathsf{ch}\Sigma(pk),\,\kappa\in\mathbb{N}}$$
$$\overset{\mathsf{s}}{\approx}\{(\mathsf{com}\Sigma(pk, x, w; r_a), e, \mathsf{ans}\Sigma(pk, x, w, r_a, e))\}_{(pk,sk)\in\mathsf{Gen}^{\mathsf{ext}}(1^\kappa),\,(x,w)\in R_{pk},\,e\in\mathsf{ch}\Sigma(pk),\,\kappa\in\mathbb{N}}$$

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 $(a, e, r_z) = \mathsf{sim}\Sigma(pk, x, e; r_z)$, we have $\mathsf{Vrfy}(pk, x, a, e, z) =$ 1 if and only if $(a, e, z) = \mathsf{sim}\Sigma(pk, x, e; z)$. *Therefore, one can instead use* $\mathsf{sim}\Sigma$ *to verify* (a, e, z) on x.

Extractability: For every $(pk, sk) \in \mathsf{Gen}^{\mathsf{ext}}(1^\kappa)$, every $x \in L^{\mathsf{ext}}_{pk}$, and every a such that there is an accepting conversation for a on x, Dec always outputs $e = \mathsf{Dec}(sk, x, a)$ such that (a, e, z) is an accepting conversation on x. We note that, when $x \notin L_{pk}$, e is unique given a, due to the special soundness property. Therefore, the extractability is well defined because $L_{pk} \cap L^{\mathsf{ext}}_{pk} = \emptyset$.

4 ABM Encryption

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

- $\mathsf{ABM.gen}$ is a PPT algorithm that takes 1^κ and outputs $(pk, (sk, w))$, where pk defines a set U_{pk}. We let $U'_{pk} = \{0, 1\}^\kappa \times U_{pk}$. pk also determines two disjoint sets, L^{td}_{pk} and L^{ext}_{pk}, such that $L^{\mathsf{td}}_{pk} \cup L^{\mathsf{ext}}_{pk} \subset U'_{pk}$.
- $\mathsf{ABM.spl}$ is a PPT algorithm that takes (pk, w, t), where $t \in \{0, 1\}^\kappa$, picks up inner random coins $v \leftarrow \mathsf{COIN}^{\mathsf{spl}}$, and computes $u \in U_{pk}$. We write $L^{\mathsf{td}}_{pk}(t)$ to denote the image of $\mathsf{ABM.spl}$ on t under pk, i.e.,
$$L^{\mathsf{td}}_{pk}(t) := \{u \in U_{pk} \,|\, \exists\, w,\ \exists\, v : u = \mathsf{ABM.spl}(pk, w, t; v)\}.$$
We require $L^{\mathsf{td}}_{pk} = \{(t, u) \,|\, t \in \{0, 1\}^\kappa \text{ and } u \in L^{\mathsf{td}}_{pk}(t)\}$. We set $\widehat{L}^{\mathsf{td}}_{pk} := U'_{pk} \backslash L^{\mathsf{ext}}_{pk}$. Since $L^{\mathsf{td}}_{pk} \cap L^{\mathsf{ext}}_{pk} = \emptyset$, we have $L^{\mathsf{td}}_{pk} \subseteq \widehat{L}^{\mathsf{td}}_{pk} \subset U'_{pk}$.
- $\mathsf{ABM.enc}$ is a PPT algorithm that takes pk, $(t, u) \in U'_{pk}$, and message $x \in \mathsf{MSP}$, picks up inner random coins $r \leftarrow \mathsf{COIN}^{\mathsf{enc}}$, and computes $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$, where MSP denotes the message space uniquely determined by pk, whereas $\mathsf{COIN}^{\mathsf{enc}}$ denotes the inner coin space uniquely determined by *pk and x* [2].

[2] We allow the inner coin space to depend on messages to be encrypted, in order to be consistent with our weak ABM encryption scheme from general assumption in Sec. E, which requires the coin space to depend on messages.

– $\mathsf{ABM.dec}$ is a DPT algorithm that takes sk, (t, u), and ciphertext c, and outputs $x = \mathsf{ABM.dec}^{(t,u)}(sk, c)$.
– $\mathsf{ABM.col} = (\mathsf{ABM.col}_1, \mathsf{ABM.col}_2)$ is a pair of PPT and DPT algorithms, respectively, such that
  - $\mathsf{ABM.col}_1$ takes $(pk, (t, u), w, v)$ and outputs $(c, \xi) \leftarrow \mathsf{ABM.col}_1^{(t,u)}(pk, w, v)$, where $v \in \mathsf{COIN}^{\mathsf{spl}}$.
  - $\mathsf{ABM.col}_2$ takes $((t, u), \xi, x)$, with $x \in \mathsf{MSP}$, and outputs $r \in \mathsf{COIN}^{\mathsf{enc}}$.

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

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

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

   – **(Trapdoor mode)** Define the following random variables: $\mathsf{dist}^{\mathsf{enc}}(t, pk, sk, w, x)$ denotes random variable (u, c, r) defined as follows: $v \leftarrow \mathsf{COIN}^{\mathsf{spl}}$; $u = \mathsf{ABM.spl}(pk, w, t; v)$; $r \leftarrow \mathsf{COIN}^{\mathsf{enc}}$; $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$. $\mathsf{dist}^{\mathsf{col}}(t, pk, sk, w, x)$ denotes random variable (u, c, r) defined as follows: $v \leftarrow \mathsf{COIN}^{\mathsf{spl}}$; $u = \mathsf{ABM.spl}(pk, w, t; v)$; $(c, \xi) = \mathsf{ABM.col}_1^{(t,u)}(pk, w, v)$; $r = \mathsf{ABM.col}_2^{(t,u)}(\xi, x)$. Then, the following ensembles are statistically indistinguishable in κ:

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

We say that a ciphertext c on (t, u) under pk is *valid* if there exist $x \in \mathsf{MSP}$ and $r \in \mathsf{COIN}^{\mathsf{enc}}$ such that $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$. We say that a valid ciphertext c on (t, u) under pk is *real* if $(t, u) \in L^{\mathsf{ext}}_{pk}$, 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 $\mathsf{ABM.dec}^{(t,u)}(sk, c)$.

5 ABME from extΣ on Language derived from PPRF

Let $\mathsf{PPRF} = (\mathsf{Gen}^{\mathsf{spl}}, \mathsf{Spl})$ be a PPRF and let $\mathsf{ext}\Sigma = (\mathsf{Gen}^{\mathsf{ext}}, \Sigma, \mathsf{Dec})$ be an extractable sigma protocol. Assume the following conditions hold.

– The first output of $\mathsf{Gen}^{\mathsf{ext}}(1^\kappa)$ is distributed identically to the first output of $\mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$.
– For every L_{pk} generated by $\mathsf{Gen}^{\mathsf{ext}}$, L_{pk} is the language derived from PPRF; namely, $L_{pk} = \{(t, u) \mid \exists (w, v) : t \in \{0, 1\}^\kappa,\ u = \mathsf{Spl}(pk, w, t; v)\}$.
– For $(L_{pk}, L^{\mathsf{ext}}_{pk}, U'_{pk})$ generated by $\mathsf{Gen}^{\mathsf{ext}}$, PPRF is unforgeable on $\widehat{L}_{pk}$, where $\widehat{L}_{pk} := U'_{pk} \backslash L^{\mathsf{ext}}_{pk}$.

Then, we can construct an ABME scheme as described in Fig. 2.

– $\mathsf{ABM.gen}(1^\kappa)$ runs $\mathsf{Gen}^{\mathsf{ext}}(1^\kappa)$ to output (pk, sk). It chooses $w \leftarrow \mathsf{KSP}^{\mathsf{spl}}_{pk}$ and finally outputs $(pk, (sk, w))$. We note that by a precondition the distribution of pk from $\mathsf{Gen}^{\mathsf{ext}}(1^\kappa)$ is identical to that of $\mathsf{Gen}^{\mathsf{spl}}(1^\kappa)$.
– $\mathsf{ABM.spl}(pk, w, t; v)$ outputs $u := \mathsf{Spl}(pk, w, t; v)$ where $v \stackrel{\mathsf{U}}{\leftarrow} \mathsf{COIN}^{\mathsf{spl}}$.
– $\mathsf{ABM.enc}^{(t,u)}(pk, m; r)$ runs $(a, m, r) \leftarrow \mathsf{sim}\Sigma(pk, (t, u), m; r)$ to return the first output a, where $r \stackrel{\mathsf{U}}{\leftarrow} \mathsf{COIN}^{\mathsf{enc}}_{pk}$ $(:= \mathsf{COIN}^{\mathsf{sim}}_{pk})$.
– $\mathsf{ABM.dec}^{(t,u)}(sk, c)$ outputs $m = \mathsf{Dec}(sk, (t, u), c)$.
– $\mathsf{ABM.col}_1^{(t,u)}(pk, w, v; r_a)$ outputs (c, ξ) such that $c := \mathsf{com}\Sigma(pk, (t, u), (w, v); r_a)$, and $\xi := (pk, t, u, w, v, r_a)$.
– $\mathsf{ABM.col}_2^{(t,u)}(\xi, m)$ outputs $r := \mathsf{ans}\Sigma(pk, (t, u), w, v, r_a, m)$, where $\xi = (pk, t, u, w, v, r_a)$.

Fig. 2. ABME from $\mathsf{ext}\Sigma$ on language derived from PPRF

By construction, the adaptive all-but-many property holds in the resulting scheme. The dual mode property also holds because: (a) If $(t,u) \in L^{\mathsf{ext}}_{pk}$, the first output of $\mathsf{sim}\Sigma(pk,(t,u),m)$ is perfectly binding to challenge m due to special soundness (because $L^{\mathsf{ext}}_{pk} \subset U'_{pk} \backslash L^{\mathsf{td}}_{pk}$, with $L^{\mathsf{td}}_{pk} := L_{pk}$), and m can be extracted given $(pk,(t,u),a)$ using sk due to extractability. (b) If $(t,u) \in L^{\mathsf{td}}_{pk}$, $\mathsf{ABM.col}$ runs the real sigma protocol with witness (w,v). Therefore, it can produce a fake commitment that can be opened in any way, while it is statistically indistinguishable from that of the simulation algorithm $\mathsf{sim}\Sigma$ (that is run by $\mathsf{ABM.enc}$), due to enhanced HVSZK. Therefore, 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 [Can01]. We concentrate on the same model in [CF01] where the network is asynchronous, the communication is public but ideally authenticated, and the adversary is adaptive in corrupting parties and is active in its control over corrupted parties. Any number of parties can be corrupted and parties cannot erase any of their inner state. We provide a brief description of the UC framework and the ideal commitment functionality for multiple commitments in Appendix B.1.

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

Common reference string: pk where $(pk,sk) \leftarrow \mathsf{ABM.gen}(1^\kappa)$.
pk uniquely determines $U'_{pk} = \{0,1\}^\kappa \times U_{pk}$. We implicitly assume that there is injective map $\iota :\{0,1\}^\kappa \rightarrow \mathsf{MSP}$ such that ι^{-1} is efficiently computable and $\iota^{-1}(y) = \varepsilon$ for every $y \notin \iota(\{0,1\}^\kappa)$, and also assume that $(\mathtt{sid},\mathtt{ssid},P_i,P_j) \in \{0,1\}^\kappa$.
The commitment phase:

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

The decommitment phase:

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

Fig. 3. Fully-Equipped UC commitment from ABME

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

Proof (Sketch). The formal proof is given in Appendix B.2. We here sketch the essence. First, see Sec. B.1 about the basic UC framework and the ideal commitment functionality $\mathcal{F}_{\mathsf{MCOM}}$. We consider the

Table 1. The man-in-the-midle attack in the hybrid games

Games	$P_i(\mathcal{S}) \xrightarrow{(t,u,c)}$	Corr. $P_{i'}(\mathcal{A}) \xrightarrow{(t',u',c')}$	$P_j(\mathcal{S}) \xrightarrow{(t',\tilde{x})}$	$\mathcal{F}_{\mathsf{MCOM}}$
Ideal	$u = \mathsf{ABM.spl}(pk, w, t; v)$ $(c, \xi) = \mathsf{ABM.col}_1^{(t,u)}(pk, w, v)$ $\mathtt{open}$: x, $r = \mathsf{ABM.col}_2^{(t,u)}(\xi, x)$	(t', u', c') $\mathtt{open}$: (x', r')	$\tilde{x} = \mathsf{ABM.dec}^{(t',u')}(sk, c')$	$\tilde{x}$
Hybrid^1	$u \leftarrow \mathsf{ABM.spl}(pk, w, t)$ $c = \mathsf{ABM.enc}^{(t,u)}(pk, x, r)$ $\mathtt{open}$: x, r	(t', u', c') $\mathtt{open}$: (x', r')	$\tilde{x} = \mathsf{ABM.dec}^{(t',u')}(sk, c')$	$\tilde{x}$
Hybrid^2	$u \leftarrow \mathsf{ABM.spl}(pk, w, t)$ $c = \mathsf{ABM.enc}^{(t,u)}(pk, x, r)$ $\mathtt{open}$: x, r	(t', u', c') $\mathtt{open}$: (x', r')	$\tilde{x} = \epsilon$	x'
Hybrid^3	$u \leftarrow U_{pk}$ $c = \mathsf{ABM.enc}^{(t,u)}(pk, x, r)$ $\mathtt{open}$: x, r	(t', u', c') $\mathtt{open}$: (x', r')	$\tilde{x} = \epsilon$	x'
	$P_i \xrightarrow{(t,u,c)}$	Corr. $P_{i'}(\mathcal{A}) \xrightarrow{(t',u',c')}$	P_j	P_j
$\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}$	$u \leftarrow U_{pk}$ $c = \mathsf{ABM.enc}^{(t,u)}(pk, x, r)$ $\mathtt{open}$: x, r	(t', u', c') $\mathtt{open}$: (x', r')		x'

Here $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_{i'})$ and $t' = (\mathtt{sid}', \mathtt{ssid}', P_{i'}, P_j)$. The view of $\mathcal{Z}$ consists of the view of $\mathcal{A}$ plus the contents in the rightest column.

man-in-the-middle attack, where we show that the view of environment $\mathcal{Z}$ in the real world (in the CRS model) can be simulated in the ideal world. Let P_i, P_j be honest players and let $P_{i'}$ be a corrupted player controlled by adversary $\mathcal{A}$. In the man-in-the-middle attack, $P_{i'}$ (i.e., $\mathcal{A}$) is simultaneously participating in the left and right interactions. In the left interactions, $\mathcal{A}$ interacts with P_i, as playing the role of the receiver. In the right interactions, $\mathcal{A}$ interacts with P_j, as playing the role of the committer.

The following sketch corresponds to the security proof in the (static) man-in-the-middle attack above. Apparently, it seems restrictive, but it is not difficult to handle any adaptive case of the scheme if this case is proven secure.

In the ideal world, $\mathcal{A}$ actually interacts with simulator $\mathcal{S}$ in both interactions, where $\mathcal{S}$ pretends to be P_i and P_j respectively. In the left interactions, environment $\mathcal{Z}$ sends $(\mathtt{commit}, \mathtt{sid}, \mathtt{ssid}, P_i, P_{i'}, x)$ to the ideal commitment functionality $\mathcal{F}_{\mathsf{MCOM}}$ (via honest P_i). After receiving $(\mathtt{receipt}, \mathtt{sid}, \mathtt{ssid}, P_i, P_{i'})$ from $\mathcal{F}_{\mathsf{MCOM}}$, $\mathcal{S}$ starts the commitment protocol as the committer without given message x. It sends to $\mathcal{A}$ (u, c) on $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_{i'})$ as computed in Table 1. In the decommitment phase when $\mathcal{Z}$ sends $(\mathtt{open}, \mathtt{sid}, \mathtt{ssid})$ to $\mathcal{F}_{\mathsf{MCOM}}$ (via honest P_i), $\mathcal{S}$ receives x from $\mathcal{F}_{\mathsf{MCOM}}$ and then computes $r = \mathsf{ABM.col}_2^{(t,u)}(\xi, x)$ to send (t, x, r) to $\mathcal{A}$. In the right interactions, $\mathcal{S}$ receives (t', u', c') from $\mathcal{A}$ where $t' = (\mathtt{sid}', \mathtt{ssid}', P_{i'}, P_j)$. It then extracts $\tilde{x} = \mathsf{ABM.dec}^{(t',u')}(sk, c')$ to send to $\mathcal{F}_{\mathsf{MCOM}}$. $\mathcal{F}_{\mathsf{MCOM}}$ then sends $(\mathtt{receipt}, \mathtt{sid}, \mathtt{ssid}, P_{i'}, P_j)$ to environment $\mathcal{Z}$ (via honest P_j). In the decommitment phase when $\mathcal{A}$ opens (t', u', c') correctly with (x', r'), $\mathcal{S}$ sends $(\mathtt{open}, \mathtt{sid}, \mathtt{ssid})$ to $\mathcal{F}_{\mathsf{MCOM}}$; otherwise, do nothing. Upon receiving $(\mathtt{open}, \mathtt{sid}, \mathtt{ssid})$, if the same $(\mathtt{sid}, \mathtt{ssid}, ..)$ was previously recorded, $\mathcal{F}_{\mathsf{MCOM}}$ sends **stored** $\tilde{x}$ to environment $\mathcal{Z}$ (via honest P_j); otherwise, do nothing. We note that in the ideal world, honest parties convey inputs from $\mathcal{Z}$ to the ideal functionalities and vice versa. The view of $\mathcal{Z}$ consists of the view of $\mathcal{A}$ plus the value sent by $\mathcal{F}_{\mathsf{MCOM}}$.

In $\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}$ **(the real world in the CRS model)**, $\mathcal{A}$ interacts with real (committer) P_i and (receiver) P_j. In the right interactions, at the end of the decommitment phase, P_j sends x' to $\mathcal{Z}$ if $\mathcal{A}$ has opened (t', u', c') correctly with (x', r'). The view of $\mathcal{Z}$ consists of the view of $\mathcal{A}$ plus the value sent by P_j.

The goal is to prove that the two views of $\mathcal{Z}$ above are computationally indistinguishable.

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 1 for summary.

Hybrid Game 1 is identical to the ideal world except that in the left interactions, at the beginning of the commitment phase, $\mathcal{S}$ (as P_i) is given message x on tag $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_{i'})$ by $\mathcal{F}_{\mathsf{MCOM}}$. $\mathcal{S}$ computes $u \leftarrow \mathsf{ABM.spl}(pk, w, t)$, and $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$, picking up random r, to send (t, u, c) to adversary $\mathcal{A}$. In the decommitment phase, $\mathcal{S}$ sends (t, x, r) to $\mathcal{A}$.

Hybrid Game 2 is identical to Hybrid Game 1 except that in the right interactions, after receiving (t', u', c'), $\mathcal{S}_2$ sends ϵ to $\mathcal{F}_{\mathsf{MCOM}}$. In the decommitment phase when $\mathcal{A}$ opens (t', u', c') correctly with (x, r), $\mathcal{S}$ sends $(\mathtt{open}, \mathtt{sid}, \mathtt{ssid}, x')$ to $\mathcal{F}_{\mathsf{MCOM}}$. $\mathcal{F}_{\mathsf{MCOM}}$ sends x' to environment $\mathcal{Z}$ (via ideal $\tilde{P}_j$), instead of sending ϵ.

Hybrid Game 3 is identical to Hybrid Game 2 except that in the left interactions, $\mathcal{S}$ instead picks up random $u \leftarrow U_{pk}$ and computes $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$, to send (t, u, c) to $\mathcal{A}$.

[$\mathsf{Ideal} \Rightarrow \mathsf{Hybrid}^1$] The two views of $\mathcal{Z}$ between the ideal world and Hybrid^1 are statistically close, due to the trapdoor mode property.

[$\mathsf{Hybrid}^1 \Rightarrow \mathsf{Hybrid}^2$] We note that the distance of the two views of $\mathcal{Z}$ between Hybrid^1 and Hybrid^2 is bounded by the following event. Let BD_I denote the event in Hybrid Game I ($I \in \{1, 2\}$) that $\mathcal{S}$ receives a fake ciphertext (t', u', c') from $\mathcal{A}$, i.e., $(t', u') \in L^{\mathsf{td}}_{pk}$, in the right intersections. If this event does not occur, the view of $\mathcal{Z}$ in both games are identical, which means $\neg\mathsf{BD}_1 = \neg\mathsf{BD}_2$. Hence, the distance of the views of $\mathcal{Z}$ in the two games is bounded by $\Pr[\mathsf{BD}]$, where $\mathsf{BD} := \mathsf{BD}_1 = \mathsf{BD}_2$. We then evaluate $\Pr[\mathsf{BD}]$ in Hybrid Game 2. (We note that we might not generally evaluate the probability in Hybrid Game 1, because $\mathcal{S}$ must decrypt (t', u', c'), which seems that it needs sk, but knowing sk implies some information on w.) We want to suppress $\Pr[\mathsf{BD}]$ by using the assumption that ($\mathsf{ABM.gen}$, $\mathsf{ABM.spl}$) is unforgeable on $\widehat{L}^{\mathsf{td}}_{pk}$. In Hybrid Game 2, we can construct an adversary B that breaks unforgeability of ($\mathsf{ABM.gen}, \mathsf{ABM.spl}$) on $\widehat{L}^{\mathsf{td}}_{pk}$ as follows. In the left and right interactions, B simulates the role of $\mathcal{S}$ and interacts with $\mathcal{A}$. B uses $\mathsf{ABM.spl}(pk, w, \cdot)$ as oracle to play the role of $\mathcal{S}$ in the left interaction. After $\mathcal{A}$ halts, B outputs (t', u') at random from the communication with $\mathcal{A}$ in the right interactions. We note that, since the communication channel is fully authenticated, it holds that $t' \neq t$ for all t, t', because $t = (\star, \star, P_i, P_{i'})$ and $t' = (\star, \star, P_{i'}, P_j)$. If $(t', u') \in \widehat{L}^{\mathsf{td}}_{pk}$, B succeeds in breaking unforgeability on $\widehat{L}^{\mathsf{td}}_{pk}$, which is upper-bounded by some negligible function. Since event BD occurs at most with the success probability of B. Hence, its probability is negligible, too.

[$\mathsf{Hybrid}^2 \Rightarrow \mathsf{Hybrid}^3$] It is obvious by construction that the distance of the two views of $\mathcal{Z}$ between Hybrid^2 and Hybrid^3 is bounded by the advantage of pseudo-randomness of ($\mathsf{ABM.gen}$, $\mathsf{ABM.spl}$).

[$\mathsf{Hybrid}^3 \Rightarrow \mathsf{Hybrid}^{\mathcal{F}_{\mathsf{MCOM}}}$] By construction, the two views of $\mathcal{Z}$ between Hybrid^3 and $\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{MCOM}}}$ are identical.

Therefore, the two views of $\mathcal{Z}$ between the ideal world and $\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{MCOM}}}$ are computationally close. ■

7 Compact ABME from Damgård-Jurik PKE

We present a DCR-based ABME scheme with compact ciphertexts and hence the first fully-equipped UC commitment scheme with optimal expansion factor $O(1)$. We start by recalling Damgård-Jurik public-key encryption scheme (DJ PKE) [DJ01].

Damgård-Jurik PKE. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a tuple of algorithms of Damgård-Jurik (DJ) PKE [DJ01]. A public key of DJ PKE is $pk_{\mathsf{dj}} = (n, d)$ and the corresponding secret-key is $sk_{\mathsf{dj}} = (p, q)$ 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 [Pai99]). 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 $\mathbf{E}_{pk_{\mathsf{dj}}}(x; R) = g^x R^{n^d} \pmod{n^{d+1}}$ where $R \leftarrow \mathbb{Z}_n^\times$ [3]. For simplicity, we write $\mathbf{E}(x)$ instead of $\mathbf{E}_{pk_{\mathsf{dj}}}(x)$, 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 $x_1 + x_2 = \gamma n^d + ((x_1 + x_2) \bmod n^d)$. It is known that $\mathbb{Z}^\times_{n^{d+1}}$ 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 $\mathbb{Z}^\times_{n^{d+1}}$ [DJ01].

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

Therefore, $\mathbb{Z}^{\times}_{n^{d+1}}$ is the image of $\mathbf{E}(\cdot;\cdot)$. We note that it is known that $\mathbb{Z}^{\times}_{n^{d+1}}$ is *efficiently samplable and explainable* [DN02,FHKW10]. It is also known that DJ PKE is IND-CPA if the DCR assumption holds true [DJ01].

Construction Idea. ($\mathsf{ABM.gen}, \mathsf{ABM.spl}$) 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 ($\mathsf{ABM.gen}, \mathsf{ABM.spl}$), as discussed in Sec. 1.3. Here, the decryption algorithm works only when the matrix below in (2) is invertible, which is equivalent to that $(t,(u_r,u_t)) \in L^{\mathsf{ext}}_{pk}$, where

$$L^{\mathsf{ext}}_{pk} = \{(t,(u_r,u_t))|\, \mathbf{D}(u_t) \not\equiv x_1x_2 + y(t)\mathbf{D}(u_r) \bmod p \,\wedge\, \mathbf{D}(u_t) \not\equiv x_1x_2 + y(t)\mathbf{D}(u_r) \bmod q\}.$$

Therefore, we require that ($\mathsf{ABM.gen}, \mathsf{ABM.spl}$) should be unforgeable on $\widehat{L}^{\mathsf{td}}_{pk} (= U'_{pk} \backslash L^{\mathsf{ext}}_{pk})$. To prove this statement, we additionally require two more assumptions on DJ PKE, called $\mathsf{the\ non\text{-}multiplication\ assumption}$ and $\mathsf{the\ non\text{-}trivial\ divisor\ assumption}$. The first one is an analogue of the DH assumption in an additively homomorphic encryption. If we consider unforgeability on L^{td}_{pk}, this assumption suffices, but we require unforgeability on $\widehat{L}^{\mathsf{td}}_{pk}$. Then we need one more assumption. We define these assumptions in Appendix C due to the space limitation. We note that these assumptions are originally introduced in [Hof12] to obtain a DCR-based ABM-LTF.

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

- $\mathsf{ABM.gen}(1^\kappa)$**:** It gets $(pk_{\mathsf{dj}}, sk_{\mathsf{dj}}) \leftarrow \mathbf{K}(1^\kappa)$ (the key generation algorithm for DJ PKE), where $pk_{\mathsf{dj}} = (n,d)$ and $sk_{\mathsf{dj}} = (p,q)$. It then picks up $x_1, x_2 \stackrel{\cup}{\leftarrow} \mathbb{Z}_{n^d}$, $R_1, R_2 \stackrel{\cup}{\leftarrow} \mathbb{Z}^{\times}_{n^{d+1}}$, and computes $g_1 = \mathbf{E}(x_1;R_1)$ and $g_2 = \mathbf{E}(x_2;R_2)$. It then picks up $\tilde{h} \leftarrow \mathbf{E}(1)$ and computes $\boldsymbol{h} = (h_0,\ldots,h_\kappa)$ such that $h_j := \tilde{h}^{y_j}$ where $y_j \stackrel{\cup}{\leftarrow} \mathbb{Z}_{n^{d+1}}$ for $j = 0,1,\ldots,\kappa$. 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 $(t_0,\ldots,t_\kappa)$ represents the bit string of t. We note that $H(t) = \tilde{h}^{y(t)}$. It outputs $(pk,(sk,w))$ where $pk := (n,d,g_1,g_2,\boldsymbol{h})$, $sk := (p,q)$ and $w := x_2$, where we define $U'_{pk} := \{0,1\}^\kappa \times (\mathbb{Z}^{\times}_{n^{d+1}})^2$ that contains the disjoint sets of L^{td}_{pk} and L^{ext}_{pk} as described below.
- $\mathsf{ABM.spl}(pk, x_2, t; (r, R_r, R_t))$**:** 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 $R_r, R_t \leftarrow \mathbb{Z}^{\times}_{n^{d+1}}$. We let

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

  We then define

  $$L^{\mathsf{ext}}_{pk} = \{(t,(u_r,u_t))|\, \mathbf{D}(u_t) \not\equiv x_1x_2 + y(t)\mathbf{D}(u_r) \bmod p \,\wedge\, \mathbf{D}(u_t) \not\equiv x_1x_2 + y(t)\mathbf{D}(u_r) \bmod q\}.$$

  Since $(t,(u_r,u_t)) \in L^{\mathsf{td}}_{pk}$ holds if and only if $\mathbf{D}(u_t) \equiv x_1x_2 + y(t)\mathbf{D}(u_r) \pmod{n^d}$, it implies that $\mathbf{D}(u_t) \equiv x_1x_2 + y(t)\mathbf{D}(u_r) \pmod{n}$. Hence, $L^{\mathsf{td}}_{pk} \cap L^{\mathsf{ext}}_{pk} = \emptyset$.
- $\mathsf{ABM.enc}^{(t,(u_r,u_t))}(pk, m; (z,s,R_A,R_a,R_b))$**:** To encrypt message $m \in \mathbb{Z}_{n^d}$, it chooses $z, s \stackrel{\cup}{\leftarrow} \mathbb{Z}_{n^d}$ 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 $R_A, R_a, R_b \stackrel{\cup}{\leftarrow} \mathbb{Z}^{\times}_{n^{d+1}}$. It outputs $c := (A,a,b)$ as the ciphertext of m on $(t,(u_r,u_t))$.
- $\mathsf{ABM.dec}^{(t,(u_r,u_t))}(sk, c)$**:** To decrypt $c = (A,a,b)$, it outputs

  $$m := \frac{x_1\mathbf{D}(a) + y(t)\mathbf{D}(b) - \mathbf{D}(A)}{x_1x_2 - (\mathbf{D}(u_t) - y(t)\mathbf{D}(u_r))} \bmod n^d. \tag{1}$$

- $\mathsf{ABM.col}_1^{(t,(u_r,u_t))}(pk, x_2, (r,R_r,R_t))$**:** It picks up $\omega, \eta \stackrel{\cup}{\leftarrow} \mathbb{Z}_{n^d}$, $R'_A, R'_a, R'_b \stackrel{\cup}{\leftarrow} \mathbb{Z}^{\times}_{n^{d+1}}$. 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 $\xi := (x_2,(r,R_r,R_t),(u_r,u_t),\omega,\eta,R'_A,R'_a,R'_b)$.
- $\mathsf{ABM.col}_2(\xi, m)$**:** To open c to m, it computes $z = \omega - mx_2 \bmod n^d$, $s = \eta - mr \bmod n^d$, $\alpha = \lfloor(\omega - mx_2 - z)/n^d\rfloor$, 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 $\mathsf{ABM.col}$ runs a canonical sigma protocol on L^{td}_{pk} 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 $(t, (u_r, u_t)) \in L^{\mathsf{td}}_{pk}$. On the contrary, $\mathsf{ABM.enc}$ runs a simulation algorithm of the sigma protocol with message (challenge) x. Notice that (A, a, b) implies the following linear system on $\mathbb{Z}_{n^d}$,

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

The matrix is invertible if

$$\mathbf{D}(u_t) \neq (x_1x_2 + y(t)\mathbf{D}(u_r)) \pmod{p} \text{ and } \mathbf{D}(u_t) \neq (x_1x_2 + y(t)\mathbf{D}(u_r)) \pmod{q},$$

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

Lemma 1 (Implicit in [Hof12]). $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$ *is PPRF with unforgeability on* $\widehat{L}^{\mathsf{td}}_{pk} (= U'_{pk} \backslash L^{\mathsf{ext}}_{pk})$*, under the assumptions, 3, 4 and 5.*

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 3), the non-tirvial divisor assmuption (Assumption 4), and the non-multiplication assumption (Assumption 5) 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 based ABME and Vice Versa

Hofheinz [Hof12] has presented the notion of all-but-many lossy trapdoor function (ABM-LTF). We provide the definition in Appendix F. We remark that ABM-LTF requires that, in our words, ($\mathsf{ABM.gen}$, $\mathsf{ABM.spl}$) be *strongly* unforgeable, whereas ABME only requires it be unforgeable. However, as shown in [Hof12], 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

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

where (m, z, s) denotes a message. This ABM-LTF has $((d-3)\log n)$-lossyness. In the latest e-print version [Hof12], 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

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

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 [Hof12], if $d \geq 5$, it can open an ciphertext arbitrarily using Barvinok's alogorithm, when $(t, (u_r, u_t)) \in L^{\mathsf{loss}}$. 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 2 shows comparison.

On the contrary, our DCR-based ABME (strengthened with strong unforgeability) can be converted to ABM-LTF [4]. Remember that $(A, a, b) = \mathsf{ABM.enc}^{(t,(u_r,u_t))}(pk, m; (z, s, R_A, R_a, R_b))$. It is obvious that

[4] Our approach is specific to our DCR-based ABME scheme. On one hand, Hemenway and Ostrovsky [HO13] 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 2. Comparison among ABMEs

ABME	expansion factor	ciphertext-length	message-length	pk-length
ABME from [Hof12]	$\geq 31^*$	$(5(d+1)+1)\log n$	$\log n$	$(\kappa+3)d\log n$
Sec. 7.1 ($d \geq 1$)	$5+1/d$	$5(d+1)\log n$	$d\log n$	$(\kappa+3)d\log n$
Sec. D	$5\kappa/\log\kappa$	$(5\ell+5)\log q$	$\ell\log\kappa$	$7\log q$

$*: d \geq 5$ is needed.

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 mean that our DCR based ABME turns out ABM-LTF. Indeed, after extracting (m, z, s) from (A, a, b), we have $(R_A)^{n^d}, (R_a)^{n^d}, (R_b)^{n^d}$ in $\mathbb{Z}^\times_{n^{d+1}}$. We remark that R_A, R_a, R_b lie not in $\mathbb{Z}^\times_{n^{d+1}}$ 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.

Table 3. Comparison among ABM-LTFs

ABM-LTF	expansion factor	output-length	input-length	lossyness
[Hof12]	$5/3$	$(5(d+1)+1)\log n$	$3d\log n$	$(d-3)\log n$
ABM-LTF (Sec. 7)	$5/3$	$(5(d+1)+1)\log n$	$3(d+1)\log n$	$d\log n$

9 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

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 1–35. Springer-Verlag, 2009.

[BMO90] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Perfect zero-knowledge in constant rounds. In *STOC '90*, pages 482–493. ACM, 1990.

[BR09] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 407–424. Springer-Verlag, 2009.

[BG85] M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In George Robert Blakley and David Chaum, editors, *CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–299. Springer-Verlag, 1985.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 2004.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer-Verlag, 2003.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001)*, pages 136–145. IEEE Computer Society, 2001. The full version available at at Cryptology ePrint Archive `http://eprint.iacr.org/2000/067`.

[CF01] R. Canetti and M. Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit 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`.

[CKS08] D. Cash, E. Kiltz, and V. Shoup. The twin diffie-hellman problem and applications. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145. Springer-Verlag, 2008.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1994.

[CS04] 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.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 125–140. Springer-Verlag, 2001.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *STOC 2003*, pages 426–437. ACM, 2003.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer-Verlag, 2002. The full version is available at `http://www.brics.dk/RS/01/41/`.

[FHKW10] Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 381–402. Springer-Verlag, 2010.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 468–485. Springer-Verlag, 2011.

[Fuj12] Eiichiro Fujisaki. New constructions of efficient simulation-sound commitments using encryption and their applications. In Orr Dunkelman, editor, *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 136–155. Springer-Verlag, 2012.

[GPY03] Juan A Garay, Philip P.Mackenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 177–194. Springer-Verlag, 2003.

[Gen04] R. Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 220–236. Springer-Verlag, 2004. The full version available at at Cryptology ePrint Archive `http://eprint.iacr.org/2003/214`.

[HO13] B. Hemenway and R. Ostrovsky. Building lossy trapdoor functions from lossy encryption. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 241–260. Springer-Verlag, 2013.

[Hof12] Dennis Hofheinz. All-but-many lossy trapdoor functions. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 209–227. Springer-Verlag, 2012. (last revised 18 Mar 2013 at http://eprint.iacr.org/2011/230).

[IOS94] Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. Language dependent secure bit commitment. In Yvo G. Desmedt, editor, *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 188–201. Springer-Verlag, 1994.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer-Verlag, 2006.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 446–466. Springer-Verlag, 2011. The full version available at at Cryptology ePrint Archive `http://eprint.iacr.org/2011/180`.

[MRY04] Philip MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 171–190. Springer-Verlag, 2004.

[MY04] Philip MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 382–400. Springer-Verlag, 2004.

[NFT12] Ryo Nishimaki, Eiichiro Fujisaki, and Keisuke Tanaka. An efficient non-interactive universally composable string-commitment scheme. *IEICE Transactions*, 95-A(1):167–175, 2012.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer-Verlag, 2008.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *STOC 2008*, pages 187–196. ACM, 2008.

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, December 2001.

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer-Verlag, 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 $H_\iota : \{0,1\}^* \to \{0,1\}^\kappa$ 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, $\Pr[\iota \leftarrow \mathcal{I}_\kappa;\ (x,y) \leftarrow C_\kappa(H_\iota) : x \neq y \ \wedge\ H_\iota(x) = H_\iota(y)] = \mathsf{negl}(\kappa)$.

A.2 Chameleon Commitment

A chameleon commitment $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{CHEval}, \mathsf{CHColl})$ 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 $c = \mathsf{CHEval}(pk, x; r)$. 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 $r \in \mathsf{COIN}_{pk}$, and outputs $r' \in \mathsf{COIN}_{pk}$ such that $\mathsf{CHEval}(pk, x; r) = \mathsf{CHEval}(pk, x'; r')$. We require that for every (pk, tk) generated by $\mathsf{CHGen}(1^\kappa)$, every $x, x' \in \{0,1\}^\kappa$, and every $r \in \mathsf{COIN}_{pk}$, there exists a *unique* $r' \in \mathsf{COIN}_{pk}$ such that $\mathsf{CHEval}(pk, x; r) = \mathsf{CHEval}(pk, x'; r')$, and $\mathsf{CHColl}(pk, tk, x, x', r)$ always computes r' in time $poly(\kappa + |x| + |x'|)$. 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,

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

A.3 Tag-Based PKEs

A Tag-PKE $\Pi = (\mathsf{Tag.Gen}, \mathsf{Tag.Enc}, \mathsf{Tag.Dec})$ is a tag-based PKE [Sho01,MRY04,Kil06] that consists of three polynomial-time algorithms: $\mathsf{Tag.Gen}$, the key-generation algorithm, is a PPT algorithm which on input 1^n outputs a pair of the public and secret keys, (pk, sk). $\mathsf{Tag.Enc}$, 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 $m \in \mathsf{MSP}$, and produces $c \leftarrow \mathsf{Tag.Enc}(pk, t, m; r)$, picking up $r \stackrel{\cup}{\leftarrow} \mathsf{COIN}$, where MSP and COIN denote the message space and the coin space determined by pk, respectively. $\mathsf{Tag.Dec}$, 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 $\mathsf{Tag.Dec}(sk, t, c)$. We require that for (sufficiently large) every $k \in \mathbb{N}$, every $t \in \{0,1\}^{p(\kappa)}$ every (pk, sk) generated by $\mathsf{Tag.Gen}(1^k)$, and every message $m \in \mathsf{MSP}$, it always holds $\mathsf{Tag.Dec}(sk, t, \mathsf{Tag.Enc}(pk, t, m)) = m$.

IND-CCA Security We recall CCA security for Tag-PKEs [MRY04], called weak CCA security [Kil06]. 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 $\mathsf{Expt}^{\mathsf{ind\text{-}cca}}_{\Pi,A,b}(\kappa)$.

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

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

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

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

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

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

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

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

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 [Can01].

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

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

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

Fig. 4. The ideal multi-commitment functionality

We consider UC commitment schemes that can be used repeatedly under a single common reference string (**re-usable common reference string**). The multi-commitment ideal functionality $\mathcal{F}_{\mathsf{MCOM}}$ from [CLOS02] is the ideal functionality of such commitments, which is given in Figure 4.

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

B.2 Proof of Theorem 1

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

For simplicity, we assume $\{0,1\}^\kappa \subset \mathsf{MSP}$, without loss of generality, which enables us to remove the injective map $\iota : \{0,1\}^\kappa \to \mathsf{MSP}$ 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 $(pk, sk) \leftarrow \mathsf{ABM.gen}(1^\kappa)$ and sets CRS to be pk (along with U_{pk} and $U' = \{0,1\}^\kappa \times U_{pk}$).
- **Simulating ideal functionality $\mathcal{F}_{\mathsf{CRS}}$:** Since $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{CRS}}$, every request (even from a honest party) to achieve a common reference string comes to $\mathcal{S}$, it returns the above-chosen CRS to the requested party.
- **Simulating the communication with $\mathcal{Z}$:** Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.
- **Simulating the commit phase when P_i is honest:** Upon receiving from $\mathcal{F}_{\mathsf{MCOM}}$ the receipt message $(\mathtt{receipt}, \mathtt{sid}, \mathtt{ssid}, P_i, P_j)$, $\mathcal{S}$ generates $u = \mathsf{ABM.spl}(pk, w, t; v)$ so that $(t, u) \in L_{pk}^{\mathsf{td}}$, where $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_j)$, and computes $(c, \xi) = \mathsf{ABM.col}_1^{(t,u)}(pk, w, v)$, namely, c is a fake ciphertext on (t, u). $\mathcal{S}$ sends $(\mathtt{sid}, \mathtt{ssid}, (t, u, c))$ to adversary $\mathcal{A}$, as it expects to receive from P_i. $\mathcal{S}$ stores $(\mathtt{sid}, \mathtt{ssid}, P_i, P_j, (t, u, c), \xi)$.
- **Simulating the decommit phase when P_i is honest:** Upon receiving from $\mathcal{F}_{\mathsf{MCOM}}$ the message $(\mathtt{open}, \mathtt{sid}, \mathtt{ssid}, P_i, P_j, x)$, $\mathcal{S}$ computes $r = \mathsf{ABM.col}_2^{(t,u)}(\xi, x)$ and sends $(\mathtt{sid}, \mathtt{ssid}, x, r)$ 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 $(\mathtt{sid}, \mathtt{ssid}, P_i, P_j, x)$, which value previously came from $\mathcal{Z}$ and was sent to $\mathcal{F}_{\mathsf{MCOM}}$, and then computes $r = \mathsf{ABM.col}_2^{(t,u)}(\xi, x)$ and reveals $(\mathtt{sid}, \mathtt{ssid}, P_i, P_j, x, r)$ to $\mathcal{A}$.
- **Simulating the commit phase when the committer P_i is corrupted and the receiver P_j is honest:** Upon receiving $(\mathtt{sid}, \mathtt{ssid}, (t, u), c)$ from $\mathcal{A}$, $\mathcal{S}$ decrypts $x = \mathsf{ABM.dec}^{(t,u)}(sk, c)$. If the decryption is invalid, then $\mathcal{S}$ sends a dummy commitment $(\mathtt{commit}, \mathtt{sid}, \mathtt{ssid}, P_i, P_j, \varepsilon)$ to $\mathcal{F}_{\mathsf{MCOM}}$. Otherwise, $\mathcal{S}$ sends $(\mathtt{commit}, \mathtt{sid}, \mathtt{ssid}, P_i, P_j, x)$ to $\mathcal{F}_{\mathsf{MCOM}}$.
- **Simulating the decommit stage when the committer P_i is corrupted and the receiver P_j is honest:** Upon receiving $(\mathtt{sid}, \mathtt{ssid}, x', r')$ from $\mathcal{A}$, as it expects to send to P_j, $\mathcal{S}$ sends

$(\texttt{open}, \texttt{sid}, \texttt{ssid})$ to $\mathcal{F}_{\mathsf{MCOM}}$. ($\mathcal{F}_{\mathsf{MCOM}}$ follows its codes: If a tuple $(\texttt{sid}, \texttt{ssid}, P_i, P_j, x)$ with the same $(\texttt{sid}, \texttt{ssid})$ was previously stored by $\mathcal{F}_{\mathsf{MCOM}}$, $\mathcal{F}_{\mathsf{MCOM}}$ sends $(\texttt{sid}, \texttt{ssid}, P_i, P_j, x)$ 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 $(\texttt{sid}, \texttt{ssid}, (t, u, c))$ 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 $\mathcal{F}_{\mathsf{MCOM}}$ 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 $p(\cdot)$,

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

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 $\mathcal{F}^1_{\mathsf{MCOM}}$, and the simulator, denoted $\mathcal{S}_1$, work exactly in the same way as $\mathcal{F}_{\mathsf{MCOM}}$ and $\mathcal{S}$ do respectively, except for **the case that P_i is honest**: In Hybrid Game 1, at the beginning of the commitment phase, $\mathcal{F}^1_{\mathsf{MCOM}}$ gives simulator $\mathcal{S}_1$ the committed value x via a honest party P_i together with $(\texttt{receipt}, \texttt{sid}, \texttt{ssid}, P_i, P_j)$. $\mathcal{S}_1$ then sets up $(t, u) \in L^{\mathsf{td}}_{pk}$ in the same way as $\mathcal{S}$ does (using w), but $\mathcal{S}_1$ instead computes c as $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$, by picking up $r \overset{\mathsf{U}}{\leftarrow} \mathsf{COIN}^{\mathsf{enc}}$. When simulating the decommitment phase or simulating adaptive corruption of P_i before the decommit phase, $\mathcal{S}_1$ simply sends $(\texttt{sid}, \texttt{ssid}, x, r)$ to adversary $\mathcal{A}$.

The distribution of (u, c, r) on $t = (\texttt{sid}, \texttt{ssid}, P_i, P_j)$ as generated in Hybrid Game 1 is statistically indistinguishable to those on the same t as generated in the ideal world, because the two distribution ensembles, $\{\mathsf{dist}^{\mathsf{col}}(t, pk, sk, w, x)\}_{\kappa\in\mathbb{N}}$ and $\{\mathsf{dist}^{\mathsf{enc}}(t, pk, sk, w, x)\}_{\kappa\in\mathbb{N}}$, defined in Sec. 4, are statistically indistinguishable in κ, for every (ensemble) $(pk, (sk, w)) \in \mathsf{ABM.gen}(1^\kappa)$, every (ensemble) $t \in \{0,1\}^\kappa$, and every (ensemble) $x \in \mathsf{MSP}(\kappa)$. Therefore, we have

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

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

Let us define BD_I as the event that the simulator receives a *fake* ciphertext c on (t, u) from P_i controlled by adversary $\mathcal{A}$ in Hybrid Game I, where $I = 1, 2$. Remember that a ciphertext c is called fake if $(t, u) \in L^{\mathsf{td}}_{pk}$ and c is a "valid" ciphertext (which means that there is a pair of message/randomness consistent with c).

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

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

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

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

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

Lemma 2. *Event BD occurs in Hybrid Game 2 at most with probability $q_A\epsilon^{\mathsf{euf}}$, 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 $\mathsf{PPRF} = (\mathsf{ABM.gen}, \mathsf{ABM.spl})$ on $\widehat{L}^{\mathsf{td}}_{pk}$.*

Proof. We construct the following algorithm B_0 that takes pk from $\mathsf{ABM.gen}$ and simulates the roles of $\mathcal{S}_2$ and $\mathcal{F}^2_{\mathsf{MCOM}}$ perfectly, interacting $\mathcal{Z}$ and $\mathcal{A}$, by having access to $\mathsf{ABM.spl}(pk, w, \cdot)$ as follows: **In the case when P_i is honest:** In the commitment phase when $\mathcal{Z}$ sends $(\texttt{commit.sid}, \texttt{ssid}, P_i, P_j, x)$ to $\mathcal{F}^2_{\mathsf{MCOM}}$ (via honest P_i), B_0 submits $t = (\texttt{sid}, \texttt{ssid}, P_i, P_j)$ to $\mathsf{ABM.spl}(pk, w, \cdot)$ to obtain u such that $(t, u) \in L^{\mathsf{td}}_{pk}$. Then B_0 computes fake ciphertext $c \leftarrow \mathsf{ABM.enc}^{(t,u)}(pk, x)$ as commitment in the same way as $\mathcal{S}_2$ $(= \mathcal{S}_1)$ does. **In the case where P_i is corrupted and P_j is honest:** In the commitment phase when corrupted P_i controlled by $\mathcal{A}$ sends a commitment (t, u, c) to $\mathcal{S}_2$ as it expects to send to honest P_j, B_0 simply plays the roles of $\mathcal{S}_2$ and $\mathcal{F}^2_{\mathsf{MCOM}}$. Later, in the opening phase when corrupted P_i controlled by $\mathcal{A}$ sends $(\texttt{sid}, \texttt{ssid}, x', r)$ to $\mathcal{S}_2$ as it expects to send to honest P_j, B_0 simply plays the role of $\mathcal{F}^2_{\mathsf{MCOM}}$.

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

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

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

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

By this, we have

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

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

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

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

Game $\mathsf{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}$: 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 $\mathcal{F}_{\mathsf{CRS}}$ parameterized by $\mathsf{ABM.gen}$ is given in Figure 5. The ideal

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

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

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

Fig. 5. The common reference string functionality

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

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

Therefore; in the end, we have

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

■

C PPRF from Damgård-Jurik PKE

In this section, we provide the formal proof of Lemma 1. Although the proof is implicitly shown in [Hof12], we provide it for completeness.

To prove the statement, we require two more assumptions related to DJ PKE, along with the standard DCR assmption, called the non-multiplication assumption and the non-trivial divisor assumption, which originally appeared in [Hof12]. We first prove that our target scheme is a PPRF with unforgeability on L^{td}_{pk} (not on $\widehat{L}^{\mathsf{td}}_{pk}$) under the DCR assumption and the non-multiplication assumption. We prove this in a more 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 3. *We say that the DCR assumption holds if for every PPT A, there exists a key generation algorithm $\mathbf{K}$ such that* $\mathsf{Adv}^{\mathsf{dcr}}_A(\kappa) =$

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

is negligible in κ, where

$\mathsf{Expt}^{\mathsf{dcr}-0}_A(\kappa)$:	$\mathsf{Expt}^{\mathsf{dcr}-1}_{d,A}(\kappa)$:
$n \leftarrow \mathbf{K}(1^\kappa)$; $R \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}^\times_{n^2}$	$n \leftarrow \mathbf{K}(1^\kappa)$; $R \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}^\times_{n^2}$
$c = R^n \bmod n^2$	$c = (1+n)R^n \bmod n^2$
return $A(n,c)$.	*return* $A(n,c)$.

Assumption 4 ([Hof12]). *We say that the non-trivial divisor assumption holds on $\Pi^{(d)}$ if for every PPT A,* $\mathsf{Adv}^{\mathsf{divisor}}_{A,\Pi^{(d)}}(\kappa) = \mathsf{negl}(\kappa)$ *where*

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

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

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

Assumption 5 ([Hof12]). *We say that the non-multiplication assumption holds on DJ PKE $\Pi^{(d)}$ if for every PPT adversary A, the advantage of A,* $\mathsf{Adv}^{\mathit{mult}}_{A,\Pi^{(d)}}(\kappa) = \mathsf{negl}(\kappa)$*, where*

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

This assumes that an adversary cannot compute $\mathbf{E}(x_1 \cdot x_2)$ for given $(pk_{\mathsf{dj}}, \mathbf{E}(x_1), \mathbf{E}(x_2))$. 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 4. *If A is an adversary against DJ PKE $\Pi^{(d)}$, there is an adversary A' against $\Pi^{(1)}$ such that*

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

Lifting-Up and Re-Randomization. We give very useful lemmas below, which are implicitly used in [DJ01] to prove that $\Pi^{(d)}$ for any $d \geq 1$ is IND-CPA secure under the DCR assumption. In order to prove Lemmas, 3 and 4, these lemmas are essential.

Lemma 5 (from [DJ01,Hof12]). *Let n be a public key of both DJ PKE $\Pi^{(d)}$, where $d \geq 1$, and DJ PKE $\Pi^{(1)}$. We let $\tau : \mathbb{Z}^{\times}{}_{n^2} \to \mathbb{Z}^{\times}_{n^{d+1}}$ be the canonical embedding map defined by $\tau(c) = c \bmod n^{d+1}$ where $c \in \mathbb{Z}^{\times}{}_{n^2}$ is canonically interpreted as an integer in $\{0, \ldots, n^2 - 1\}$. We let $\pi : \mathbb{Z}^{\times}_{n^{d+1}} \to \mathbb{Z}^{\times}_{n^2}$ be the canonical homomorphism defined by $\pi(\hat{c}) = \hat{c} \bmod n^2$ where $\hat{c} \in \mathbb{Z}^{\times}_{n^{d+1}}$ is canonically interpreted as an integer in $\{0, \ldots, n^{d+1} - 1\}$. We then have:*

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

Based on Lemma 5, we have the following lemma.

Lemma 6 (from [DJ01,Hof12]). *There is an algorithm B that takes any public-key $pk = (n, d)$ ($d > 1$) and any ciphertext $c \in \mathbb{Z}^{\times}_{n^2}$ for $\Pi^{(1)}$, and efficiently samples random $\hat{c} \in \mathbb{Z}^{\times}_{n^{d+1}}$ conditioned on $\mathbf{D}^{(1)}(\pi(\hat{c})) = \mathbf{D}^{(1)}(c) \pmod{n}$.*

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

$$\begin{array}{ccc}
\tau(c) \in \mathbb{Z}^{\times}_{n^{d+1}} & \overset{\text{re-randomize}}{\Longrightarrow} & \hat{c} = \tau(c) \cdot \mathbf{E}(yn) \in \mathbb{Z}^{\times}_{n^{d+1}} \\
\tau \uparrow & & \downarrow \pi \\
c \in \mathbb{Z}^{\times}_{n^2} & \overset{\mathbf{D}^{(1)}(c) = \mathbf{D}^{(1)}(\pi(\hat{c}))}{\Longleftrightarrow} & \pi(\hat{c}) \in \mathbb{Z}^{\times}_{n^2}
\end{array}$$

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

C.2 Proof of Lemmas, 3 and 4

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, 3 and 4.

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 $x_1, \ldots, x_n \in X$. We write $c^z \in Y$, for $c \in Y$ and $z \in \mathbb{Z}$, to denote $\overbrace{c \star \cdots \star c}^{z}$.

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

Assumption 6 (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, $\mathsf{Adv}_A^{\mathit{mult}}(\kappa) = \mathsf{negl}(\kappa)$, where $\mathsf{Adv}_A^{\mathit{mult}}(\kappa) \triangleq$*

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

This assumption is a generalized version of Assumption 5.

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

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

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

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

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

Theorem 7. *Let Π be an enhanced additively homomorphic public-key encryption scheme mentioned above. Suppose that Π is IND-CPA and the non-multiplication assumption holds on Π. Then, the above $(\mathsf{Gen}_{\mathsf{spl}}, \mathsf{Spl})$ is a PPRF with unforgeability on L_u.*

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 if it determined that it has had access to a PPRF. We run S on pk. It picks up at random $x_1, x_2 \leftarrow X$, $h_0, h_1, \ldots, h_\kappa \leftarrow Y$, and sets $g_1 \leftarrow \mathbf{E}_{pk}(x_1)$ and $g_2 \leftarrow \mathbf{E}_{pk}(x_2)$. It sends (m_0, m_1) to the challenger, where $m_0 = 0$, 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 $PK = (pk, g_1, g_2, \boldsymbol{h})$, where $\boldsymbol{h} = (h_0, h_1, \ldots, h_\kappa)$. For any query t, the simulator picks up random $r \leftarrow X$ and returns (u_r, u_t) such that $u_r = \mathbf{E}_{pk}(r)$ 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, $\mathsf{Adv}_{\Pi}^{\mathsf{ind\text{-}cpa}} S(\kappa) = \Pr[S = 1 \,|\, b = 1] - \Pr[S = 1 \,|\, b = 0] = \Pr[A = 1 \,|\, b = 1] - \Pr[A = 1 \,|\, b = 0] = \mathsf{Adv}_{\mathsf{pprf}} A(\kappa)$. Therefore, $\mathsf{Adv}_{\mathsf{pprf}} A(\kappa) = \mathsf{Adv}_{\Pi}^{\mathsf{ind\text{-}cpa}} S(\kappa) = \mathsf{negl}(\kappa)$.

The proof of unforgeability on this scheme is substantially similar to that in [BB04,Wat05,BR09]. We provide a sketch of the proof.

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

In game G_1, we modify the choice of $\boldsymbol{h}$ as follows: Recall now that $(X, +, \times)$ is a finite commutative ring such that $(X, +) \simeq \mathbb{Z}/n\mathbb{Z}$ for some integer n. Let Gen_1 be the generator in game G_1. Let $\theta = O(\frac{q_s}{\varepsilon_0})$, where q_s denotes the maximum number of queries A submits to Spl. Gen_1 picks up (pk, g_1, g_2) as Gen does. It then picks up $a_0, a_1, \ldots, a_\kappa \leftarrow \mathbb{Z}/n\mathbb{Z}$. It picks up $y_1, \ldots, y_\kappa \leftarrow [0, \cdots, (\theta-1)]$ and $y_0 \in [0, \ldots, \kappa(\theta-1)]$. It finally outputs $PK = (pk, g_1, g_2, \boldsymbol{h})$, by setting $h_i = g^{a_i} g_2^{y_i}$ for $i \in [0, \cdots, \kappa]$. Since $(X, +) \simeq \mathbb{Z}/n\mathbb{Z}$ and $\mathbf{E}_{pk}$ is additively homomorphic, $Y \subset \mathbb{Z}/n\mathbb{Z}$. Hence, the distribution of $\boldsymbol{h}$ is identical to that in the previous game, and this change is conceptual. Therefore, the advantage of A in G_1, ε, is equal to ε_0.

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

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

We now borrow the following lemmas due to [BR09].

Lemma 7. *[BR09]. Let $Q(\boldsymbol{t})$ be the event in game G_1 mentioned above. Then,*

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

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

Lemma 8. *[BR09]. Let n, θ, κ be positive integers, such that $\kappa\theta < n$. Let $y_0, y_1, \ldots, y_\kappa$ be elements in the domains mentioned above and let $y(t) = y_0 + \sum t_i \cdot y_i \in \mathbb{Z}$. Then, for every $t_0, \ldots, t_\kappa \in \{0,1\}^\kappa$, we have*

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

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

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

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

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

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

Therefore, the righthand side is $g_1^{-\frac{a(t)}{y(t)}} \star \mathbf{E}_{pk}(0; v) \star (H(t))^{r'}$ for some $v \in R$. This is substantially equivalent to the technique of all-but-one simulation technique in [BB04]. As in game G_2, the simulator always abort if $\gamma_{\boldsymbol{y}}(\boldsymbol{t}) = 1$ holds. Hence, the advantage of A in this game, denoted ε_3, is equivalent to ε_2.

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

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

C.4 Proof of Lemma 1

We now complete the proof of Lemma 1. We note that we have already proved in Theorem 7 that the scheme is a PPRF with unforgeability on L_{pk}^{td} under Assumption 3 and Assumption 6 because Assumption 6 is a generalized version of Assumption 5. We now show the following.
Lemma 1 (restated) $\mathsf{PPRF} = (\mathsf{ABM.gen}, \mathsf{ABM.spl})$ is a PPRF with unforgeability on $\widehat{L}_{pk}^{\mathsf{td}} := U'_{pk} \backslash L_{pk}^{\mathsf{ext}}$, under the assumptions, 3, 4 and 5.

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

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

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

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

We first show that our target PPRF has unforgeability on $L_{pk}^{(1)}$. In the proof of Theorem 7, we change the proof as follows: In the final game, the simulator instead takes $(pk_{\mathsf{dj}}, c_1, c_2)$ where $pk_{\mathsf{dj}} = (n, 1)$ 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 $pk'_{\mathsf{dj}} := (n, d)$ 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 5. Then the simulator start game G_3 with $(pk'_{\mathsf{dj}}, g_1, g_2)$ 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 $\frac{u_t(t_0)}{u_r^{\alpha(t_0)}} \bmod n$. Therefore, the probability of A outputting such pairs is negligible; otherwise, it contradicts Assumption 5.

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 $(t^*, (u_r^*, u_t^*)) \in L_{\mathsf{divisor}}$. Then, it outputs $c^* := \frac{u_t^*}{g_1^{x_2} (u_r^*)^{y(t^*)}}$. $(t^*, (u_r^*, u_t^*)) \in L_{\mathsf{divisor}}$, means that $1 < \gcd(\mathbf{D}_{sk_{\mathsf{dj}}}(c^*), n) < n$. Therefore, the probability that $(t^*, (u_r^*, u_t^*)) \in L_{\mathsf{divisor}}$ is negligible; otherwise, it contradicts Assumption 4.

D 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 works (with $O(\kappa)$). This scheme also has a short public key.

We first construct a PPRF. Let $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{CHEval}, \mathsf{CHColl})$ 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 $\mathsf{TwinCS} = (\mathsf{CS.gen}$, $\mathsf{CS.enc}$, $\mathsf{CS.dec})$. be a tag-based twin DDH version of Cramer-Shoup PKE [CS04,CKS08], where

- $\mathsf{CS.gen}(1^\kappa)$**:** Via $(pk_{\mathsf{cs}}, sk_{\mathsf{cs}}) \leftarrow \mathsf{CS.gen}(1^\kappa)$, it picks up hash $(pk_{\mathcal{CH}}, sk_{\mathcal{CH}}) \leftarrow \mathsf{CHGen}(1^\kappa)$, a generator of G, g, 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 $pk_{\mathsf{cs}} := (pk_{\mathcal{CH}}, g, X, \hat{X}, Y, \hat{Y})$ and $sk_{\mathsf{cs}} := (pk_{\mathsf{cs}}, x, \hat{x}, y, \hat{y})$.
- $\mathsf{CS.enc}(pk_{\mathsf{cs}}, t, m)$: Via $c \leftarrow \mathsf{CS.enc}(pk_{\mathsf{cs}}, t, m)$, 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 $r \stackrel{\cup}{\leftarrow} \mathsf{COIN}_{\mathcal{CH}}$, and computing $d := g^v$, $e := m \cdot X^v$, $\tau := \mathsf{CHEval}(pk_{\mathcal{CH}}, (t, d, e); r)$, $\pi_x := (X^\tau \hat{X})^v$, and $\pi_y := (Y^\tau \hat{Y})^v$, where $v \stackrel{\cup}{\leftarrow} \mathbb{Z}/q\mathbb{Z}$.
- $\mathsf{CS.dec}(sk_{\mathsf{cs}}, t, c)$: Via $m \leftarrow \mathsf{CS.dec}(sk_{\mathsf{cs}}, t, c)$, 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 $\tau = \mathsf{CHEval}(pk_{\mathcal{CH}}, (t, d, e); r)$ and outputs $m := e \cdot d^{-x}$ if the above equations both hold, otherwise $\perp$.

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

$\mathsf{PPRF} = (\mathsf{Gen}^{\mathsf{spl}}, \mathsf{Spl})$ from TwinCS is constructed as follows:

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

Here, we let $U' := \{0,1\}^\kappa \times \mathsf{COIN}_{\mathcal{CH}} \times G^4$ and

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

We let $\widehat{L}^{\mathsf{td}}_{pk} := \{(t, (r, d, e, \pi_x, \pi_y)) \mid \exists\, (\tilde{v}, v) : (d_0 d, e_0 e) = (g^{\tilde{v}}, X^{\tilde{v}})$ and $(d, \pi_x, \pi_y) = (g^v, (X^\tau \hat{X})^v, (Y^\tau \hat{Y})^v)\}$, where $\tilde{v} = v_0 + v$ and $\tau = \mathsf{CHEval}(pk_{\mathcal{CH}}, (t, d, e); r)$. We note that $L^{\mathsf{td}}_{pk} = \widehat{L}^{\mathsf{td}}_{pk}$. We note that PPRF consists of two encryptions of El Gamal and Twin Cramer-Shoup that encrypt the same message.

Lemma 9. *The scheme obtained above is a PPRF with unforgeability on $\widehat{L}^{\mathsf{td}}_{pk}$ 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 simulator sets up all secrets and public parameter $pk = (pk_{\mathsf{cs}}, d_0, e_0)$. The simulator returns $(d, e, \pi_x, \pi_y) \leftarrow \mathsf{CS.enc}(pk_{\mathsf{cs}}, t, \zeta)$ 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 $(d', e', \pi'_x, \pi'_y) \in \mathsf{CS.enc}(pk_{\mathsf{cs}}, t', \zeta)$ where t' is not queried.

We consider a sequence of $q+1$ games, $G_{1,0}, \ldots,, G_{1,q}$, where q denotes the number of queries that A submits. We define Game $G_{1,0}$ as G_0. Let $t_1, \ldots, t_q$ be a sequence of queries from A. In game $G_{1,i}$, where $i \in \{0, \ldots, q\}$, the simulator returns $(d, e, \pi_x, \pi_y) \leftarrow \mathsf{CS.enc}(pk_{\mathsf{cs}}, t_j, 0^{|\zeta|})$ for $j \le i$, whereas returns $(d, e, \pi_x, \pi_y) \leftarrow \mathsf{CS.enc}(pk_{\mathsf{cs}}, t_j, \zeta)$ for $j > i$. Let $\epsilon_{1,i}$ be the advantage of A in game $G_{1,i}$, i.e., the probability that it outputs $(d', e', \pi'_x, \pi'_y) \in \mathsf{CS.enc}(pk_{\mathsf{cs}}, t', \zeta)$ 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 $i \in \{0, \ldots, q-1\}$, 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 $\zeta \overset{\mathsf{U}}{\leftarrow} G^\times$ and sets $(d_0, e_0) := (g^{v_0}, \zeta^{-1} X^{v_0})$ where $v_0 \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}/q\mathbb{Z}$. For the first j queries of A, with $j \le i$, B returns $\mathsf{CS.enc}(pk_{\mathsf{cs}}, t_j, 0^{|\zeta|})$. 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 $(d^*, e^*, \pi^*_x, \pi^*)$. For the remaining queries, B returns $\mathsf{CS.enc}(pk_{\mathsf{cs}}, t_j, \zeta)$ 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 $\epsilon_{1,i}(\kappa) - \epsilon_{1,i+1}(\kappa) \le \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{TwinCS},A}(\kappa)$, for every $i \in \{0, \ldots, q-1\}$, which is negligible in κ if the DDH assumption holds and $\mathcal{H}$ is a collision resistant hash families.

We note that B needs the decryption oracle only once, to check that c' is a ciphertext of ζ.

In Game G_2, the simulator behaives as follows: It is given pk_{cs} and $|\zeta|$ as input, chooses a random 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 simulator is not given ζ. For every query t_i of A, $1 \le i \le q$, the simulator returns $\mathsf{CS.enc}(pk_{\mathsf{cs}}, t_i, 0^{|\zeta|})$. 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 simulator submits it to the decryption oracle and outputs its reply. We note that the simulator does not reveal any information on t to A. Hence, it holds that $t' \neq t$ with (overwhelming) probability $1 - \frac{1}{q}$. If c' is a ciphertext of ζ, the simulator 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 attacks. The advantage is bounded by twice of that of IND-CCA security of TwinCS. Hence, we have $\epsilon_0(\kappa) \le (q+2)\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{TwinCS},B}(\kappa) + \frac{1}{q}$. ■

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

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

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

We note that $\tilde{v} = v_0 + v$. We define $L^{\mathsf{ext}}_{pk} = U'_{pk} \backslash \widehat{L}^{\mathsf{td}}_{pk}$.
– $\mathsf{ABM.enc}^{(t,u)}(pk, m; (\hat{\boldsymbol{z}}, \boldsymbol{z}))$**:** To encrypt message $m \in \{0,1\}^n$, it parses m as $(m_1, \ldots, m_\ell)$ where $\ell = n/\lambda$ and $m_i \in \{0,1\}^\lambda$. It picks up vectors, $\tilde{\boldsymbol{z}}, \boldsymbol{z} \overset{\mathsf{U}}{\leftarrow} G^\ell$, where $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \ldots, \tilde{z}_\ell)$ and $\boldsymbol{z} = (z_1, \ldots, z_\ell)$, and computes 2-by-$\ell$ matrix A 3-by-ℓ matrix B such that

$$A = \begin{pmatrix} g & d_0 d \\ X & e_0 e \end{pmatrix} \begin{pmatrix} \tilde{z_1} & \ldots & \tilde{z_\ell} \\ m_1 & \ldots & m_\ell \end{pmatrix}, \text{ and } B = \begin{pmatrix} g & d \\ X^\tau \hat{X} & \pi_x \\ Y^\tau \hat{Y} & \pi_y \end{pmatrix} \begin{pmatrix} z_1 & \ldots & z_\ell \\ m_1 & \ldots & m_\ell \end{pmatrix}. \tag{3}$$

It finally outputs $c = (A, B)$.
– $\mathsf{ABM.dec}^{(t,u)}(sk, c)$: Let $A = (\boldsymbol{a_1}, \ldots, \boldsymbol{a_\ell})$ and $B = (\boldsymbol{b_1}, \ldots, \boldsymbol{b_\ell})$, where $\boldsymbol{a_i} = (a_{1,i}, a_{2,i})^{\mathrm{T}}$ and $\boldsymbol{b_i} = (b_{1,i}, b_{2,i}, b_{3,i})^{\mathrm{T}}$. For all $i \in [\ell]$, it searches "consistent" $m_i \in \{0,1\}^\lambda$ such that

$$\frac{(a_{1,i})^x}{a_{2,i}} = \Big(\frac{(d_0 d)^x}{e_0 e}\Big)^{m_i} \text{ if } e_0 e \neq (d_0 d)^x, \quad \frac{(b_{1,i})^{\tau x + \hat{x}}}{b_{2,i}} = \Big(\frac{d^{\tau x + \hat{x}}}{\pi_x}\Big)^{m_i} \text{ if } \pi_x \neq d^{\tau x + \hat{x}},$$
$$\text{and} \quad \frac{(b_{1,i})^{\tau y + \hat{y}}}{b_{3,i}} = \Big(\frac{d^{\tau y + \hat{y}}}{\pi_y}\Big)^{m_i} \text{ if } \pi_y \neq d^{\tau y + \hat{y}}, \quad \text{where } \tau = H(t, d, e). \tag{4}$$

It aborts if it find no m_i or "inconsistent" one for some $i \in [\ell]$; otherwise outputs $m = (m_1, \ldots, m_\ell) \in \{0,1\}^n$.
– $\mathsf{ABM.col}_1^{(t,u)}(pk, t, ((\zeta, v_0), v); (\tilde{\boldsymbol{w}}, \boldsymbol{w}))$: It picks up $\tilde{w}_i, w_i \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}/q\mathbb{Z}$ 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 $\xi = (v_0, v, \tilde{\boldsymbol{w}}, \boldsymbol{w})$, where $\tilde{\boldsymbol{w}} = (\tilde{w}_1, \ldots, \tilde{w}_l)$ and $\boldsymbol{w} = (w_1, \ldots, w_l)$.
– $\mathsf{ABM.col}_2^{(t,u)}(\xi, m)$: To open $c = (A, B)$ to m, it parses m as $(m_1, \ldots, m_\ell)$ 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 $(\tilde{\boldsymbol{z}}, \boldsymbol{z})$, consistent with m in Equation (3).

Suppose that $(t, (r, d, e, \pi_x, \pi_y)) \in L^{\mathsf{td}}_{pk}$. Each column vector $\boldsymbol{a_i} = (a_{1,i}, a_{2,i})^{\mathrm{T}}$ in A from $\mathsf{ABM.col}_1$ can be seen as the first message in a canonical sigma protocol on common input $(d_0 d, e_0 e)$ to prove that $\log_g(d_0 d) = \log_X(e_0 e)$, and $\tilde{z}_i$ from $\mathsf{ABM.col}^2$ corresponds to the response on challenge m_i. Hence, $(A, \boldsymbol{m}, \tilde{\boldsymbol{z}})$ is the accepting conversation of the parallel execution of the sigma protocol with parallel challenge $\boldsymbol{m} = (m_1, \ldots, m_\ell)$, where $m_i \in \{0,1\}^\lambda$. Similarly, $(B, \boldsymbol{m}, \boldsymbol{z})$ is the accepting conversation of the parallel execution of a sigma protocol on common input (d, π_x, π_y) with parallel challenges $\boldsymbol{m}$ 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 $(t, (d, e, \pi_x, \pi_y)) \in L^{\mathsf{td}}_{pk}$ iff $\mathrm{rank}(A(t,u)) = 1$ and $\mathrm{rank}(B(t,u)) = 1$, where $A(t,u) := \begin{pmatrix} g & d_0 d \\ X & e_0 e \end{pmatrix}$ and $B(t,u) := \begin{pmatrix} g & d \\ X^\tau \hat{X} & \pi_x \\ Y^\tau \hat{Y} & \pi_y \end{pmatrix}$. So, when $(t, (d, e, \pi_x, \pi_y)) \in L^{\mathsf{ext}}_{pk} (= U_{pk} \backslash L^{\mathsf{td}}_{pk})$, $\mathrm{rank}(A(t,u)) = 2$ or $\mathrm{rank}(B(t,u)) = 2$. Hence, each m_i can be retrieved by checking either of equations in (4). We note that if $\mathrm{rank}(A(t,u)) = \mathrm{rank}(B(t,u)) = 2$, the linear system (3) is overdetermined. Then, one should check if $\boldsymbol{m}$ 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 a simulator can decrypt *valid* ciphertexts correctly, because an adversary cannot correctly open an invalid ciphertext on $(t, u) \in L^{\mathsf{ext}}_{pk}$.

Theorem 8. *The scheme constructed as above is an ABME scheme if the DDH assumption holds true and $\mathcal{H}$ is a collision-resistant hash family ensemble.*

This scheme has a ciphertext consisting of $5\ell + 4$ group elements plus $|\mathsf{COIN}_{\mathcal{CH}}|$-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 [CF01] 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 [CF01] is strict, by construction, which cannot be improved.

E Fully-Equipped UC Commitment from Trapdoor Permutations

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

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

for every $(pk, (sk, w)) \in \mathsf{ABM.gen}(1^\kappa)$, every $x \in \mathsf{MSP}$, every $t \in \{0,1\}^\kappa$. Here $\mathsf{ABM.col}_1^{(t,u)}(pk, w, v)[i]$ denotes i-th output of $\mathsf{ABM.col}_1^{(t,u)}(pk, w, v)$. We construct a weak ABM encryption scheme from trapdoor permutations as follows.

Let $\mathcal{F} = \{(f, f^{-1}) \mid f : \{0,1\}^\kappa \to \{0,1\}^\kappa\}_{\kappa\in\mathbb{N}}$ be a trapdoor permutation family and let b: $\{0,1\}^\kappa \to \{0,1\}$ be a hard-core predicate for a trapdoor permutation f. Let $\mathit{\Pi} = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be the Blum-Goldwasser cryptosystem [BG85] that is a semantic secure public key encryption scheme, derived from the following encryption algorithm $\mathbf{E}_f(x; r) = f^{(k+1)}(r) \,||\, (x_1 \oplus b(r)) \,||\, \ldots \,||\, (x_k \oplus b(f^{(k)}(r)))$, where $(x_1, \ldots, x_\kappa)$, $x_i \in \{0,1\}$, denotes the bit representation of x. $r \in \{0,1\}^\kappa$ denotes inner randomness of this encryption and $f^{(k)}$ denotes k times iteration of f. We note that this public key encryption scheme has efficiently samplable and explainable presumable ciphertext space $\{0,1\}^{\kappa+k}$ [CF01,FHKW10], namely, $\{\mathbf{E}_f(x)\} \stackrel{c}{\approx} \{U_{\kappa+k}\}$ for every message $x \in \{0,1\}^\kappa$, where $U_{\kappa+k}$ denotes a uniform distribution over $\{0,1\}^{\kappa+k}$. Let us denote by F: $\{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ a pseudo-random function (constructed from f in a standard way).

- $\mathsf{ABM.gen}(1^\kappa)$: It draws two trapdoor permutations, (f, f^{-1}) and (f', f'^{-1}), over $\{0,1\}^\kappa$ uniformly and independently from $\mathcal{F}$. It then construct the BG encryption scheme $\mathit{\Pi} = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ with public key f and secret key f^{-1}. It also construct the BG encryption scheme $\mathit{\Pi}' = (\mathbf{K}', \mathbf{E}', \mathbf{D}')$ with (f', f'^{-1}) and pseudo random function F from f'. It then picks up random $s \leftarrow \{0,1\}^\kappa$ and encrypt it to $e' = \mathbf{E}'(s; r)$. It outputs (pk, sk, w), where $pk = (F, \mathit{\Pi}, \mathit{\Pi}', e')$ and $sk = f^{-1}$,$w = (s, r)$. We define $U'_{pk} = \{0,1\}^\kappa \times \{0,1\}^\kappa$.
- $\mathsf{ABM.spl}(pk, w, t)$: It takes tag $t \in \{0,1\}^\kappa$ and outputs $u = F_s(t)$ where $w = (s, r)$. We define
$$L_{pk}^{\mathsf{td}} = \widehat{L}_{pk}^{\mathsf{td}} = \{(t, u) \mid \exists (s, r) \text{ such that } e' = \mathbf{E}'(s; r) \text{ and } u = F_s(t)\}.$$
- $\mathsf{ABM.enc}^{(t,u)}(pk, x)$: It takes (t, u) and one bit message $x \in \{0,1\}$ along with pk, and first obtains a graph G (of q nodes) so that finding a Hamiltonian cycle in G is equivalent to finding (s, r) such that $u = F_s(t)$ and $e' = \mathbf{E}'(s; r)$, by using the NP-reduction. (If such (s, r) does not exist for given (t, u), G so obtained does not have a Hamiltonian cycle.) This encryption procedure is the same as the commitment described in [CLOS02], called the adaptive Hamiltonian commitment, except that in our scheme a commitment is encrypted under a public key f independent of F and $\mathit{\Pi}'$, and an encrypted permutation or a pseudo ciphertext is also sent to the verifier.
  - To encrypt 0, it picks a random permutation $\pi = (\pi_1, \ldots, \pi_q)$ of q nodes, where $\pi_i \in \{0,1\}^{\log q}$, and encrypts every π_i and all the entries of the adjacency matrix of the 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 $(i \in [q])$ (corresponding to a pseudo ciphertext of π_i). It then chooses a randomly labeled Hamiltonian cycle, and for all the entries in the adjacency matrix corresponding to edges on the Hamiltonian cycle, it encrypts 1's. For all the other entries, it picks up random $\kappa + 1$-bit strings (corresponding to pseudo ciphertexts of the entries). It outputs $\{A_i\}_{i\in[q]}$ and $\{B_{i,j}\}_{i,j\in[q]}$, where a Hamiltonian cycle is embedded in $\{B_{i,j}\}_{i,j\in[q]}$, but the other strings are merely random strings.

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

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

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

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

Theorem 9. *The scheme in Fig.3 obtained by applying the above weak ABME UC-securely realizes the $\mathcal{F}_{\mathsf{MCOM}}$ functionality in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model in the presence of adaptive adversaries in the non-erasure setting.*

Proof. The only difference from the proof of Theorem 1 is when we compare the ideal world with Hybrid Game 1. In the proof of Theorem 1, the outcome from $\mathsf{ABM.col}$ is statistically indistinguishable from the outcome from $\mathsf{ABM.enc}$ in the trapdoor mode when $(t, u) \in L_{pk}^{\mathsf{td}}$. However, the difference is computational now. To show that the distributions are computationally indistinguishable, we need to construct a distinguisher to distinguish the first outcome from the second outcome, while it decrypts commitments from corrupted P_i at the same time.

Fortunately, in this construction, the decryption key $sk = \{f^{-1}\}$ is independent of the equivocable key $w = (s, r)$. It is not the case of the rest of our instantiations, in which one can obtain w if one knows sk. (Therefore, we require statistical closeness there.)

Hence, we construct a distinguisher that takes $sk = \{f^{-1}\}$ and starts either the ideal game or Hybrid Game 1, but when making a commitment on tag $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_j)$, it receives (t, u, c) either from $(\mathsf{ABM.spl}, \mathsf{ABM.col})$ or $(\mathsf{ABM.spl}, \mathsf{ABM.enc})$ such that $(t, u) \in L_{pk}^{\mathsf{td}}$. Hence, the views of the environment in both games are bounded by the distinguisher's advantage, which is negligible because of computational indistinguishability between the two views above. ■

We note that if the common reference string must strictly come from the uniform distribution, we require trapdoor permutations with dense public descriptions. We also note that a parallel execution of arbitrary weak ABMEs on the same tag with independent public-keys (pk_1, pk_2) yields a fully-equipped UC commitment scheme, because of the similar reason in the proof mentioned above.

We also note that in a weak ABME, a decryption key can be independent of an equivocable key (or a sampling key for L), unlike an ABME, because the distribution of $\mathsf{ABM.col}$ on (t, u) is only computationally indistinguishable from that of $\mathsf{ABM.enc}$.

We further note that any weak ABME can be transformed to a fully-equipped UC commitment scheme by sending parallel ciphertexts of the same message on the same tag under independent public keys. The proof is substantially equivalent to the proof above.

The above construction does not require non-interactive zero-knowledge proof systems. So, it is far more efficient than the previous fully-equipped UC commitment scheme from trapdoor permutation [CLOS02] (See Table 4).

F All-But-Many Lossy Trapdoor Functions

We recall all-but-many lossy trapdoor functions (ABM-LTF) [Hof12], by slightly modifying the notation to fit our purpose.

Table 4. Fully-Equipped UC commitments (to λ bit secret) from general assumptions (enhanced trapdoor permutations).

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

T_{NP} denotes the cost of one NP reduction from one-way function to a Hamiltonian graph. $T_{\mathsf{tdp}}(k)$ 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.

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

- $\mathsf{ABM.gen}$ is a PPT algorithm that takes 1^κ and outputs $(pk, (sk, w))$, where pk defines a set U_{pk}. We let $U'_{pk} = \{0,1\}^\kappa \times U_{pk}$. pk also determines two disjoint sets, L^{loss}_{pk} and L^{inj}_{pk}, such that $L^{\mathsf{loss}}_{pk} \cup L^{\mathsf{inj}}_{pk} \subset U'_{pk}$.
- $\mathsf{ABM.spl}$ is a PPT algorithm that takes (pk, w, t), where $t \in \{0,1\}^\kappa$, picks up inner random coins $v \leftarrow \mathsf{COIN}^{\mathsf{spl}}$, and computes $u \in U_{pk}$. We write $L^{\mathsf{loss}}_{pk}(t)$ to denote the image of $\mathsf{ABM.spl}$ on t under pk, i.e.,
$$L^{\mathsf{loss}}_{pk}(t) := \{u \in U_{pk} \mid \exists\, w,\, \exists\, v :\ u = \mathsf{ABM.spl}(pk, w, t; v)\}.$$
We require $L^{\mathsf{loss}}_{pk} = \{(t,u) \mid t \in \{0,1\}^\kappa \text{ and } u \in L^{\mathsf{loss}}_{pk}(t)\}$. We set $\widehat{L}^{\mathsf{loss}}_{pk} := U'_{pk} \backslash L^{\mathsf{inj}}_{pk}$. Since $L^{\mathsf{loss}}_{pk} \cap L^{\mathsf{inj}}_{pk} = \emptyset$, we have $L^{\mathsf{loss}}_{pk} \subseteq \widehat{L}^{\mathsf{loss}}_{pk} \subset U'_{pk}$.
- $\mathsf{ABM.eval}$ is a DPT algorithm that takes pk, (t,u), and message $x \in \mathsf{MSP}$ and computes $c = \mathsf{ABM.eval}^{(t,u)}(pk, x)$, where MSP denotes the message space uniquely determined by pk.
- $\mathsf{ABM.inv}$ is a DPT algorithm that takes sk, (t,u), and c, and computes $x = \mathsf{ABM.inv}^{(t,u)}(sk, c)$.

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

1. **Adaptive All-but-many property:** $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$ is a probabilistic pseudo random function (PPRF), as defined in Sec. 3.1, with *strongly* unforgeability on $\widehat{L}^{\mathsf{loss}}_{pk} = U'_{pk} \backslash L^{\mathsf{inj}}_{pk}$. Strong unforgeability in this paper is called $\mathsf{evasiveness}$ in [Hof12].
2. **Inversion** For every $\kappa \in \mathbb{N}$, every $(pk, (sk, w)) \in \mathsf{ABM.gen}(1^\kappa)$, every $(t,u) \in L^{\mathsf{inj}}_{pk}$, and every $x \in \mathsf{MSP}$, it always holds that
$$\mathsf{ABM.inv}^{(t,u)}(sk, \mathsf{ABM.eval}^{(t,u)}(pk, x)) = x.$$
3. **ℓ-Lossyness** For every $\kappa \in \mathbb{N}$, every $(pk, (sk, w)) \in \mathsf{ABM.gen}(1^\kappa)$, and every $(t,u) \in L^{\mathsf{loss}}_{pk}$, the image set $\mathsf{ABM.eval}^{(t,u)}(pk, \mathsf{MSP})$ is of size at most $|\mathsf{MSP}| \cdot 2^{-\ell}$.

Here L^{loss}_{pk} (resp. L^{inj}_{pk}) in ABM-LTFs corresponds to L^{td}_{pk} (resp. L^{ext}_{pk}) in ABMEs. We remark that ABM-LTFs [Hof12] require that $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$ should be *strongly* unforgeable, whereas ABMEs requires that $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$ be only unforgeable.