J Cryptol (2018) 31:226–275
https://doi.org/10.1007/s00145-017-9256-x

Journal of
CRYPTOLOGY

All-But-Many Encryption

Eiichiro Fujisaki
Japan Advanced Institute of Science and Technology, Ishikawa, Japan
fujisaki@jaist.ac.jp

Communicated by Serge Fehr.

Received 15 June 2016 / Revised 13 March 2017
Online publication 11 May 2017

Abstract. We present a new cryptographic primitive, called *all-but-many encryption* (ABME). An ABME scheme is a tag-based public-key encryption scheme with the following additional properties: A sender given the secret key can generate a fake ciphertext to open to any message with consistent randomness. In addition, anyone who does not own the secret key can neither distinguish a fake ciphertext from a real (honestly generated) one, nor produce a fake one (on a fresh tag) even after seeing many fake ciphertexts and their opening. A motivating application of ABME is universally composable (UC) commitment schemes. We prove that an ABME scheme implies a non-interactive UC commitment scheme that is secure against adaptive adversaries in the non-erasure model under a reusable common reference string. Previously, such a "fully equipped" UC commitment scheme has been known only in Canetti and Fischlin (CRYPTO 2001, vol 2139, Lecture notes in computer science. Springer, Heidelberg, pp 19–40, 2001), Canetti et al. (STOC 2002, pp 494–503, 2002), with *expansion factor* $O(\kappa)$, meaning that to commit λ bits, communication strictly requires $O(\lambda\kappa)$ bits, where κ denotes the security parameter. We provide a general framework for constructing ABME and several concrete instantiations from a variety of assumptions. In particular, we present an ABME scheme with expansion factor $O(1)$ from DCR-related assumptions, which results in showing the *first* fully equipped UC commitment scheme with a constant expansion factor. In addition, the DCR-based ABME scheme can be transformed to an all-but-many lossy trapdoor function (ABM-LTF), proposed by Hofheinz (EUROCRYPT 2012, vol 7237, Lecture notes in computer science. Springer, Heidelberg, pp 209–227, 2012), with a better lossy rate than Hofheinz (2012).

Keywords. All-but-many encryption, All-but-many lossy trapdoor function, Fully equipped UC commitments.

The paper was accepted when the author was working for NTT Secure Platform Laboratories, Tokyo, Japan.

1. Introduction

1.1. *Motivating Application: Fully Equipped UC Commitments*

Universal composability (UC) framework [13] 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 stand-alone setting. UC commitment is an essential ingredient to construct high-level UC secure protocols, which imply UC zero-knowledge protocols [14,25] and UC oblivious transfer [16]. Therefore, any UC secure two-party and multi-party computations can be realized in the presence of UC commitments. Since UC commitments cannot be realized without an additional setup assumption [14], the common reference string (CRS) model is widely used.

A commitment scheme consists of a two-phase protocol between two parties, a committer and a receiver. In the commitment phase, a committer gives a receiver the digital equivalent of a *sealed envelope* containing value x, and, in the opening phase, the committer reveals x in a way that the receiver can verify it. From the original concept, it is required that a committer cannot change the value inside the envelope (*binding property*), whereas the receiver can learn nothing about x (*hiding property*) unless the committer helps the receiver open the envelope.

Informally, a UC commitment scheme maintains the above binding and hiding properties under *any concurrent composition with arbitrary protocols*. To achieve this, a UC commitment scheme requires *equivocability* and *extractability* at the same time. Informally, equivocability of UC commitments in the CRS model can be interpreted as follows: An algorithm (called the simulator) that takes the secret behind the CRS string can generate an *equivocal* commitment that can be opened to any value. On the other hand, extractability can be interpreted as the ability of the simulator extracting the contents of a commitment generated by any adversarial algorithm, even after the adversary sees many equivocal commitments generated by the simulator.

Several factors as shown below feature UC commitments:

1.1.1. *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.

1.1.2. *CRS Reusability*

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 *reusable* 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.

1.1.3. *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.

1.1.4. *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. *Related Works*

Canetti and Fischlin [14] presented the first UC secure commitment schemes. One of their proposals is "fully equipped," i.e., *non-interactive, adaptively UC secure in the non-erasure model under a reusable common reference string*. By construction, this scheme requires $O(\lambda\kappa)$ bits when committing to λ-bit secret, where κ denotes the security parameter. Canetti et al. [16] constructed its generalized version from (enhanced) trapdoor permutations, which is simply inefficient. Damgård and Nielsen [25] proposed the first adaptively UC secure commitment schemes in the non-erasure model with expansion factor $O(1)$, meaning that to commit to λ-bit secret, communication requires only $O(\lambda)$ bits. However, the commitment phase must take three-round interactions between a committer and a receiver. In addition, the CRS size grows linearly in the number of the parties. Soon after, Damgård and Groth [24] removed the dependency of the CRS size, using the simulation sound trapdoor commitments, but the improved proposal is still interactive.

The subsequent commitment schemes such as [7,28,45,49] are adaptively UC secure with expansion factor $O(1)$ under a constant size CRS string, but still *sacrifice at least* one or two requirements (see Table 1). Nishimaki, Fujisaki, and Tanaka [49] proposed non-interactive adaptively UC secure commitments, but the CRS is just one time, i.e., the committer and the receiver need a new common reference string for each execution of the commitment protocol. Lindell [45] presented efficient static and adaptively UC secure commitment schemes based on the DDH assumption, which are recently improved by Blazy et al. [7] and Fujisaki [31]. However, these constructions require interaction and secure erasure. Fischlin, Libert, and Manulis [28] transformed Lindell's static UC secure commitment scheme and Camenisch and Shoup verifiable encryption scheme [12] into non-interactive adaptively UC secure commitment schemes, by removing the interaction in the sigma protocol using non-interactive Groth–Sahai proofs [35]. The resulting protocols still require secure erasure.

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

Table 1. Comparison among UC Commitments.

Scheme	Exp. rate of communication.	Non-interactiveness	Adaptive security	Non-erasure	Reusable CRS	Assumption(s)
CF01 [14]	$O(\kappa)$	✓	✓	✓	✓	DDH +CFP
CLOS02 [16]	$\omega(\kappa^5 \log \kappa)$	✓	✓	✓	✓	eTDP
DN02 [25]	$O(1)$		✓	✓	✓*	DCR
DG03 [24]	$O(1)$		✓		✓	DCR+SRSA+sOTS
NFT12 [49]	$O(1)$	✓	✓	✓		DCR
ABBCP13 [1]	$O(\kappa)$	✓	✓		✓	SXDH
Lin11 [45]	$O(1)$		✓		✓	DDH+CRHF
FLM11 [28]	$O(1)$	✓	✓		✓	DLIN+CRHF+Pairing
BCPV13 [7]	$O(1)$		✓		✓	DDH+CRHF
F16 [31]	$O(1)$		✓		✓	DDH+CRHF
GIKW14 [32]	$1 + o(1)$			–	✓	UC OT+PRG
DDGN14 [23]	$O(1)$			–	✓	UC OT+PRG
CDDGNT15 [18]	$O(1)$			–	✓	UC OT+PRG
FJNT16 [29]	$1 + o(1)$			–	✓	UC OT+PRG
CDDDN16 [17]	$1 + o(1)$			–	✓	UC OT+PRG
Section 6	$O(1)$	✓	✓	✓	✓	DCR+Assump. 8 and 9
Section 7	$O(\kappa/\log \kappa)$	✓	✓	✓	✓	DDH+CRHF
Appendix 3	$O(\kappa/\log \kappa)$	✓	✓	✓	✓	DDH
Section 8	$O(\kappa^3)$	✓	✓	✓	✓	eTDP

DDH: Decisional DH assumption. CFP: Claw-free permutations. eTDP: Enhanced trapdoor permutations. DCR: Decisional composite residuosity assumption. SXDH: Symmetric decisional DH assumption. SRSA: Strong RSA assumption. sOTS: Strong one-time signatures. CRHF: Collision-resistant hash family. DLIN: Decisional linear assumption. Pairing: Pairing Groups. UC OT: UC oblivious transfer oracle. PRG: Pseudo-random generator. $\star$: The size of the CRS grows linearly with the number of parties

Fast Static UC Secure Commitments Recently, a series of efficient UC commitment protocols [17,18,23,29,32] have been proposed in the UC oblivious transfer (OT) hybrid model. It is composed of inexpensive symmetric primitives except for using OT. Using the OT extension techniques [2,39,40], one can make the number of the execution of commitments independent of the number of the execution of OT protocols. So, these schemes are much faster than the above schemes relying on public-key primitives, when sufficiently many commitments are executed. In particular, [17,29,32] achieve an expansion factor of $1 + o(1)$ per commitment. However, these schemes are only *static* UC secure.

UC Commitments in the Random Oracle Models Hofheinz and Müller-Quade [38] and Canetti et al. [15] have proposed efficient UC commitment schemes in the different variations of the random oracle model [6].

1.3. *Our Contribution*

We introduce a new primitive, called all-but-many encryption (ABME). We prove that ABME 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, we believe that it is desirable to translate the essence of basic UC secure protocols into simple cryptographic primitives.

We divide the functionality of ABME into two primitives. We then provide a condition to be able to construct ABME from the primitives successfully. We believe that this framework is helpful to find more constructions in the future. We remark that our constructions are inspired by that of all-but-many lossy trapdoor function (ABM-LTF) given by Hofheinz [37]. We will expose the relation in Sections 1.4 and 6.4.

We present a *compact* ABME scheme related to the DCR assumption, which can be seen as the *first* fully equipped UC commitment scheme with *expansion factor* $O(1)$, meaning that to commit to λ-bit secret, it requires $O(\lambda)$ bits, where $\lambda = O(\kappa)$. Our DCR-based ABME scheme can be transformed into an ABM-LTF scheme with a *better* lossy rate than [37] under the same assumption. We also provide ABME from the DDH assumption with overhead $O(\kappa/\log\kappa)$, which is slightly better than prior works with $O(\kappa)$. We also present a fully equipped UC commitment scheme from *weak* ABME under the general assumption that (enhanced) trapdoor permutations exist, which is far more efficient than the previous work [16] under the same assumption.

In the following, we describe more details.

1.3.1. *All-But-Many Encryption*

All-but-many encryption (ABME) enables a party with a secret key (e.g., the simulator in the UC framework) to generate a *fake* ciphertext and to open it to any message with consistent randomness. In the case that a party is not given the secret key (e.g., the adversary in the UC framework), he cannot distinguish a fake ciphertext from a *real* (honestly generated) ciphertext even after the message and randomness are revealed. In addition, he cannot produce a fake ciphertext (on a fresh tag) even after seeing many fake ciphertexts and their openings. We construct ABME from two new primitives, denoted probabilistic pseudorandom functions and extractable sigma protocols. The former is a probabilistic version of a pseudorandom function. The latter is a special type of a sigma protocol [20] with some *extractability*.

1.3.2. *Probabilistic Pseudorandom Function*

A $\mathsf{pPRF} = (\mathsf{KG}, \mathsf{Spl})$ is a probabilistic version of a pseudorandom function associated with a key-generation algorithm KG. Let $L_{pk}(t) := \{u | \exists(sk, v) : u = \mathsf{Spl}(pk, sk, t; v)\}$, where (pk, sk) is generated by KG and v denotes random coins of Spl. The PPT algorithm Spl is a sampling algorithm that takes tag t and samples u in $L_{pk}(t)$ according to the random choice of v. It should be assumed that $L_{pk}(t)$ is a hidden subset in a universe set U_{pk} and the distribution following $\mathsf{Spl}(pk, w, t)$ on any tag t is computationally indistinguishable from the uniform distribution over U_{pk}. The universe set U_{pk} should be efficiently samplable and an explainable domain [27]. It should be also assumed that pPRF be unforgeable—it is difficult to sample $u \in L_{pk}(t)$ for fresh t, if sk is not given. Sometimes, it should be unforgeable even on some superset $\widehat{L}_{pk}(t)$. The superset $\widehat{L}_{pk}(t)$ is determined in relation to the corresponding extractable sigma protocol mentioned below. The meaning will be clearer later in this section.

1.3.3. *Extractable Sigma Protocols*

A sigma protocol Σ [20] on NP language L is a canonical 3-round public coin interactive proof system, so that a prover can convince a verifier that he knows witness w behind common input $x \in L$, where the prover first sends commitment a; the verifier sends back challenge (public coin) e; the prover responds with z; and the verifier finally accepts or rejects conversation (a, e, z) on x. A sigma protocol is associated with simulation algorithm $\mathsf{simP}_{\Sigma}^{\mathsf{com}}$ that takes x (regardless of whether $x \in L$ or not) and challenge e, and produces an accepting conversation $(a, e, z) \leftarrow \mathsf{simP}_{\Sigma}^{\mathsf{com}}(x, e)$ *without witness* w. If $x \in L$, the distribution of (a, e, z) produced by $\mathsf{simP}_{\Sigma}^{\mathsf{com}}(x, e)$ on random e is statistically indistinguishable from the transcript generated between two honest parties, called *honest-verifier statistical zero-knowledge* (HVSZK). If $x \notin L$, for every a there is at most one e such that (a, e, z) can be an accepting conversation on x, called *special soundness*.

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

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

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

1.3.4. *A General Framework:* $\mathsf{pPRF} + \Sigma^{\mathsf{ext}} \rightarrow$ *ABME*

To instantiate ABME schemes, we first consider an instantiation of pPRF. Then, we try to construct an extractable sigma protocol on the language derived from pPRF. If we succeed to do so, we say that they are well combined. Then, we convert the well-combined primitives to an ABME scheme. Formally, we say that $\mathsf{pPRF} = (\mathsf{KG}, \mathsf{Spl})$ and $\Sigma^{\mathsf{ext}} = (\Sigma, \mathsf{Ext})$ are $\mathsf{well\ combined}$ if:

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

From well-combined pPRF and Σ^{ext}, we can construct an ABME scheme, by taking the similar method to convert an ordinary sigma protocol to an instance-dependent commitment scheme [4,41]. Here is the transform.

- To encrypt message e on tag t, a sender picks random u, runs $\mathsf{simP}_{\Sigma}^{\mathsf{com}}$ on instance (t, u) with challenge e with random z, to compute $(a, e, z) = \mathsf{simP}_{\Sigma}^{\mathsf{com}}(pk, (t, u), e; z)$, and finally outputs (u, a) as a ciphertext. Here z is regarded as the random coins of the ciphertext. Due to the unforgeability condition

of pPRF, it holds that $(t, u) \in U'_{pk} \backslash \widehat{L}_{pk} (= L^{\mathsf{co}}_{pk})$ with an overwhelming probability. Then, e is uniquely determined given $((t, u), a)$. By our precondition, we can decrypt (t, u, a) using sk^{ext}, as $e = \mathsf{Ext}(sk^{\mathsf{ext}}, (t, u), a)$ because $(t, u) \in L^{\mathsf{co}}_{pk}$.
– To make a fake (equivocable) ciphertext on tag t, one picks up random v and compute $u = \mathsf{Spl}(pk, sk^{\mathsf{spl}}, t; v)$ using sk^{spl}. Then he computes a, as same as an honest prover computes the first message on common input (t, u) with witness (sk^{spl}, v). To open a to arbitrary e, he produces the response z in the sigma protocol. By construction, he can open a to any e because $(t, u) \in L_{pk}$.

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

1.3.5. *Realizing Extractable Sigma Protocols*

Although sigma protocols (with HVSZK) exist on *many* NP languages, it is not known how to extract the challenge as discussed above. Here we observe that sigma protocols are often implemented on Abelian groups associated with homomorphic maps, in which the first message of such sigma protocols implies a system of linear equations with e and z. Hence, there is a matrix derived from the linear systems. Due to completeness and special soundness, there is an invertible (sub) matrix if and only if $x \notin L_{pk}$ (provided that the linear system is defined in a finite field). Therefore, if one knows the contents of the matrix, one can solve the linear systems when $x \notin L_{pk}$ and obtain e if its length is logarithmic. Suppose for instance that L_{pk} is the DDH language—it does not form a pPRF, but a good toy example to explain how to extract the challenge. Let $x = (g_1, g_2, h_1, h_2) \notin L_{pk}$, meaning that $x_1 \neq x_2$ where $x_1 := \log_{g_1}(h_1)$ and $x_2 := \log_{g_2}(h_2)$. The first message (A_1, A_2) of a canonical sigma protocol on L_{pk} implies linear equations

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

where $A_1 = g_1^{a_1}$, $A_2 = g_2^{a_2}$, and $g_2 = g_1^{\alpha}$. The above matrix is invertible if and only if $(g_1, g_2, h_1, h_2) \notin L_{pk}$. We note that e is expressed as a linear combination of a_1 and a_2, i.e., $(\beta_1 (\det A)^{-1}) a_1 + (\beta_2 (\det A)^{-1}) a_2$, where the coefficients are determined by the matrix. Therefore, if the decryption algorithm takes (α, x_1, x_2) and the length of e is logarithmic, it can find e by checking whether $(g_1^{\det A})^e = A_1^{\beta_1} A_2^{\beta_2}$ or not. In a general case where a partial information on the values of the matrix is given, the decryption algorithm can still find logarithmic length e if the matrix is made so that e can be expressed as a *linear* combination of *unknown values*—the unknown values do not appear with a quadratic form or a more degree of forms in the equations.

In a good case, the decryption algorithm can invert homomorphic map $f(a) = g^a$, using trapdoor f^{-1}. Then, one can obtain (a_1, a_2) as well as the entire values of the matrix and hence extract even *polynomial length* e. This corresponds to the case of our DCR-based implementation, where the corresponding linear system is defined on a finite *ring*, such as $\mathbb{Z}_{n^d}$. The matrix (say A) derived from the linear system is invertible if and

only if $(\det A)^{-1} \bmod n^d$ exists, which corresponds to the condition $x \notin \widehat{L}_{pk}$ for some superset $\widehat{L}_{pk}$. We note that although $x \notin L_{pk}$ iff $\det A \neq 0 \pmod{n^d}$, it does not suffice for the above because of the divisors. We require unforgeability not on L_{pk} but on $\widehat{L}_{pk}$, so that the output produced by an adversary can be forced in $L^{\mathsf{co}}_{pk} = U'_{pk} \backslash \widehat{L}_{pk}$.

1.3.6. *Concrete Instantiations*

We present ABME schemes from three different types of pPRFs. We first propose a pPRF from Waters signature scheme [56] defined over a ring *equipped with no bilinear map*. As the associated homomorphic map, we employ Damgård–Jurik (DJ) PKE [22]. The output of the Waters signature-based pPRF looks pseudorandom, thanks to IND-CPA security of DJ PKE. The construction inherits unforgeability from the original Waters signature scheme under an analogue of the DH assumption in the additive homomorphic encryption. Precisely, we require one more assumption related to DJ PKE, because we require unforgeability on some superset of the language derived from the Waters signature-based pPRF. we construct an extractable sigma protocol on it. Since the homomorphic map is invertible using the secret key of DJ PKE, we can obtain a compact ABME scheme and hence a fully equipped UC commitment scheme with expansion factor $O(1)$ with a constant number of computational complexity.

In "Appendix 3", we simply use as pPRF the Waters signature scheme on a pairing-free prime-order group and provide the DDH version of the ABME scheme above. Although its expansion factor is just $O(\kappa/\log\kappa)$, it is better than the prior work [14] (with $O(\kappa)$). This scheme is helpful to understand our main proposal, because of the simpler construction. So, we recommend the reader to read that section first, if the proposal above looks complicated.

We present another construction of pPRF by combining an IND-CPA secure PKE scheme with an IND-CCA secure Tag-PKE scheme. We combine ElGamal PKE with tag-based Twin-Cramer–Shoup PKE [19] and construct an ABME scheme from the resulting pPRF under the DDH assumption. The expansion factor of this scheme is also $O(\kappa/\log\kappa)$. The advantage of this scheme is that it has a short public key (of a constant number of group elements), unlike the proposed schemes above.

We also provide a generic construction of pPRF from a pseudorandom function family and an IND-CPA secure PKE scheme. We employ this type of pPRFs to construct a UC commitment scheme from general assumptions.

1.4. *Other Related Works*

Fehr et al. [27] proposed a PKE scheme secure against simulation-based selective opening chosen ciphertext attack (SIM-SO-CCA). In general, the notion of SIM-SO-CCA secure PKE is related to that of ABME, but both are incomparable. Indeed, Fehr et al. scheme [27] does not satisfy the requirements of ABME, while ABME does not satisfy SIM-SO-CCA PKE in general, because it does not support CCA security. Although [27] 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 presented the notion of all-but-many lossy trapdoor function (ABM-LTF) [37], mainly to construct indistinguishability-based selective opening CCA (IND-SO-CCA) secure PKE. ABM-LTF is a lossy trapdoor function (LTF) [52] with (unbounded) *many* lossy tags. The relation between ABM-LTF and ABME is a generalized analogue of LTF and lossy encryption [3,51] with unbounded many loss tags. However, unlike the lossy encryption, ABME always requires an *efficient* opening algorithm that can open a ciphertext on a lossy tag to any message with consistent randomness. As mentioned earlier, our construction idea of ABME is strongly inspired by that of ABM-LTF [37]. Hofheinz provided a matrix-based function $\boldsymbol{Y} = \mathbf{A}\boldsymbol{X}$, where $\mathbf{A}$ denotes a square matrix and $\boldsymbol{Y}$, $\boldsymbol{X}$ denote column vectors. The algorithm to produce lossy tags is pPRF in our definition. The lossy tags are carefully embedded in matrix $\mathbf{X}$ so that the matrix can be non-invertible if tags are lossy, otherwise invertible. Hofheinz proposed two instantiations. In the DCR-based ABM-LTF, the lossy tags are an analogue of Waters signatures defined in DJ PKE, which is the same as our DCR-based pPRF. Therefore, it is not surprising that our DCR-based ABME scheme requires the same assumptions as the Hofheinz's ABM-LTF counterpart does. In the latest e-print version [37], Hofheinz has shown that the DCR-based ABM-LTF can be converted to SIM-SO-CCA PKE. To realize this, *an opening algorithm* for ABM-LTF is essentially needed. So, he gave it by sacrificing efficiency. We remark that ABM-LTF *equipped with an opening algorithm* meets the notion of ABME. However, compared to our DCR-based ABME scheme in Sect. 6, Hofheinz's ABM-LTF-based ABME scheme is less efficient for practical use. Indeed, its expansion rate of ciphertext length per message length is ≥ 31. In addition, you must use a modulus of $\geq n^6$. On the other hand, our DCR-based ABME scheme has a small expansion rate of $(5 + 1/d)$ and you can use modulus of n^{d+1} for any $d \geq 1$. On the contrary, our DCR-based ABME can be converted to ABM-LTF and is more efficient than Hofheinz's ABM-LTF scheme. We compare them in Sect. 6.4.

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 $X \overset{\mathrm{s}}{\approx} Y$, if $\mathsf{Dist}(X_\kappa, Y_\kappa) = \mathsf{negl}(\kappa)$. We say that X and Y are computationally indistinguishable (in κ), denoted $X \overset{\mathrm{c}}{\approx} Y$, if for every (non-uniform) PPT D (ranging over $\{0,1\}$), it holds that $\{D(1^\kappa, X_\kappa)\}_{\kappa\in\mathbb{N}} \overset{\mathrm{s}}{\approx} \{D(1^\kappa, Y_\kappa)\}_{\kappa\in\mathbb{N}}$. Let A and B be PPT algorithms that both take $x \in S(\kappa)$, where $S(\kappa)$ is a set associated with each $\kappa \in \mathbb{N}$. We

write $\{A(x)\}_{\kappa\in\mathbb{N},x\in S(\kappa)} \overset{s}{\approx} \{B(x)\}_{\kappa\in\mathbb{N},x\in S(\kappa)}$ to denote $\{A(x_\kappa)\}_{\kappa\in\mathbb{N}} \overset{s}{\approx} \{B(x_\kappa)\}_{\kappa\in\mathbb{N}}$ for every sequence $\{x_\kappa\}_{\kappa\in\mathbb{N}}$ such that $x_\kappa \in S(\kappa)$.

3. Definitions

In this section, we define new cryptographic primitives. We put the definitions of known primitives in "Appendix 1". We formally introduce a probabilistic pseudorandom function (pPRF), an extractable sigma protocol, and all-but-many encryption (ABME). As already mentioned, the first two primitives are used to construct an ABME scheme.

3.1. *Probabilistic Pseudorandom Function*

A probabilistic pseudorandom function $\mathsf{pPRF} = (\mathsf{KG}, \mathsf{Spl})$ consists of the following two algorithms:

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

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

3.1.1. *Efficiently Samplable and Explainable Domain*

For all pk given by KG and all $t \in \{0,1\}^\kappa$, U_{pk} is efficiently samplable and explainable [27], that is, there is an PPT sampling algorithm U that takes (pk, t), picks up random coins R, and outputs u that is uniformly distributed in domain U_{pk}. In addition, for every pk, every $t \in \{0,1\}^\kappa$, and every $u \in U_{pk}$, there is an efficient explaining algorithm that takes (pk, t) and outputs random coins R behind u, where R is uniformly distributed subject to $U(pk, t; R) = u$.

3.1.2. *Pseudorandomness*

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

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

is negligible in κ, where

$\mathsf{Expt}^{\mathsf{prf}}_{\mathsf{pPRF},A}(\kappa)$:
$(pk, sk) \leftarrow \mathsf{KG}(1^\kappa)$
$b \leftarrow A^{\mathsf{Spl}(pk,sk,\cdot)}(pk)$
return b.

$\mathsf{Expt}^{\mathsf{prf}}_{U,A}(\kappa)$:
$(pk, sk) \leftarrow \mathsf{KG}(1^\kappa)$
$b \leftarrow A^{U(pk,\cdot)}(pk)$
return b

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

3.1.3. *Unforgeability*

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

3.2. *Extractable Sigma Protocol*

We define extractable sigma protocols. Let $L = \{L_{pk}\}_{pk}$ be an NP language consisting of a collection of set L_{pk} indexed by $pk \in \mathcal{PK}$, where $\mathcal{PK}$ is an infinite sequence of pk. Let R_{pk} be the relation derived from L_{pk}. Let $\Sigma^{\mathsf{ext}} = (\mathsf{P}^{\mathsf{com}}_\Sigma, \mathsf{P}^{\mathsf{ans}}_\Sigma, \mathsf{V}^{\mathsf{vrfy}}_\Sigma, \mathsf{simP}^{\mathsf{com}}_\Sigma, \mathsf{Ext})$ be a tuple of algorithms (associated with L) as follows:

- $\mathsf{P}^{\mathsf{com}}_\Sigma$ is a PPT algorithm that takes $(x, w) \in R_{pk}$, picks up inner coins r_a, and outputs $a = \mathsf{P}^{\mathsf{com}}_\Sigma(x, w; r_a)$.
- $\mathsf{P}^{\mathsf{ans}}_\Sigma$ is a DPT algorithm that takes (x, w, r_a, e) and outputs $z = \mathsf{P}^{\mathsf{ans}}_\Sigma(x, w, r_a, e)$, where e is an element in a specific domain determined by pk.

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

$\mathsf{Expt}^{\mathsf{seuf}\text{-}\widehat{L}}_{\mathsf{pPRF},A}(\kappa)$:
$(pk, w) \leftarrow \mathsf{KG}^{\mathsf{pprf}}(1^\kappa)$
$(t^*, u^*) \leftarrow A^{\mathsf{Spl}(pk,w,\cdot)}(pk)$
$(t^*, u^*) \notin \mathcal{QA}$
and $u^* \in \widehat{L}_{pk}(t^*)$,
return 1; otherwise 0.

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

- $\mathsf{V}_{\Sigma}^{\mathsf{vrfy}}$ is a DPT algorithm that accepts or rejects (x, a, e, z).
- $\mathsf{simP}_{\Sigma}^{\mathsf{com}}$ is a PPT algorithm that takes (x, e) and outputs $(a, e, z) = \mathsf{simP}_{\Sigma}^{\mathsf{com}}(x, e; r_z)$, where r_z is inner coins. For our purpose,*we additionally require that* $r_z = z$, i.e., $(a, e, r_z) = \mathsf{simP}_{\Sigma}^{\mathsf{com}}(x, e; r_z)$. We note that many sigma protocols satisfy this property.
- Ext is a DPT algorithm that takes (sk, x, a) and outputs e or $\perp$, where sk is a string with respects to pk.

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

3.2.1. *Completeness*

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

3.2.2. *Special Soundness*

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

3.2.3. *Extractability*

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

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

3.2.4. *Enhanced Honest-Verifier Statistical Zero-Knowledge (eHVSZK)*

For all $(pk, sk^{\mathsf{ext}}) \in R^{\mathsf{ext}}$, all $(x, w) \in R_{pk}$, and all e in a specific domain, the following ensembles are statistically indistinguishable in κ:

$$\begin{aligned}&\left\{\mathsf{simP}_{\Sigma}^{\mathsf{com}}(x, e; r_z)\right\}_{\kappa \in \mathbb{N},\, pk,\, (x,w) \in R_{pk},\, e}\\ &\quad \overset{\mathrm{s}}{\approx} \left\{\left(\mathsf{P}_{\Sigma}^{\mathsf{com}}(x, w; r_a), e, \mathsf{P}_{\Sigma}^{\mathsf{ans}}(x, w, r_a, e)\right)\right\}_{\kappa \in \mathbb{N},\, pk,\, (x,w) \in R_{pk},\, e}\end{aligned}$$

Here the probability of the left-hand side is taken over random variable r_z and the right-hand side is taken over random variable r_a. We remark that since $(a, e, r_z) = \mathsf{simP}_{\Sigma}^{\mathsf{com}}(x, e; r_z)$ (by our precondition), we have $\mathsf{Vrfy}(x, a, e, z) = 1$ if and only if

$(a, e, z) = \mathsf{simP}_{\Sigma}^{\mathsf{com}}(x, e; z)$. *Therefore, one can instead use* $\mathsf{simP}_{\Sigma}^{\mathsf{com}}$ *to verify* (a, e, z) on x.

We note that the concept of the extractable sigma protocol is not entirely new. A weaker notion, called *weak* extractable sigma protocol, appears in [30] to construct (interactive) simulation sound trapdoor commitment (SSTC) schemes (see [33,34,47] for SSTC). This paper requires a stronger notion, which is used in a different way.

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

$$L^{\mathsf{td}}_{pk}(t) = \left\{ u \in U_{pk} \mid \exists\, sk^{\mathsf{spl}},\ \exists\, v :\ u = \mathsf{ABM.spl}\left(pk, sk^{\mathsf{spl}}, t; v\right) \right\}.$$

  We let $L^{\mathsf{td}}_{pk} = \{(t, u) \mid t \in \{0,1\}^{\kappa} \text{ and } u \in L^{\mathsf{td}}_{pk}(t)\}$. Define $\widehat{L}^{\mathsf{td}}_{pk} = U'_{pk} \backslash L^{\mathsf{ext}}_{pk}$. 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 $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$.
- $\mathsf{ABM.dec}$ is a DPT algorithm that takes sk^{ext}, (t, u), and ciphertext c, and outputs $x = \mathsf{ABM.dec}^{(t,u)}(sk^{\mathsf{ext}}, c)$.
- $\mathsf{ABM.col} = (\mathsf{ABM.col}_1, \mathsf{ABM.col}_2)$ is a pair of PPT and DPT algorithms, respectively, such that
  - $\mathsf{ABM.col}_1$ takes $(pk, (t, u), sk^{\mathsf{spl}}, v)$ and outputs $(c, \xi) \leftarrow \mathsf{ABM.col}_1^{(t,u)}(pk, sk^{\mathsf{spl}}, v)$, 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 pseudorandom function (pPRF) as defined in Sect. 3.1 with unforgeability on $\widehat{L}^{\mathsf{td}}_{pk} (= U'_{pk} \backslash L^{\mathsf{ext}}_{pk})$.
2. **Dual mode property.**
   - **(Decryption mode)** For all $\kappa \in \mathbb{N}$, all $(pk, sk^{\mathsf{ext}}) \in \mathsf{ABM.gen}(1^{\kappa})$, all $(t, u) \in L^{\mathsf{ext}}_{pk}$, and every $x \in \mathsf{MSP}$, it always holds that

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

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

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

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

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 $(t, u) \in L_{pk}^{\mathsf{ext}}$, otherwise *fake*. We remark that as long as c is a real ciphertext, regardless of how it is generated, there is only one consistent x in MSP and it is equivalent to $\mathsf{ABM.dec}^{(t,u)}(sk, c)$.

To suit actual instantiations, we assume that $\mathsf{COIN}^{\mathsf{spl}}$ and MSP are defined by pk. We further allow $\mathsf{COIN}^{\mathsf{enc}}$ to depend on message x to be encrypted as well as pk, in order to be consistent with our weak ABM encryption scheme from general assumption in Sect. 8.

4. ABME Implies Fully Equipped UC Commitment

In this section, we prove that ABME implies fully equipped UC commitments.

We work in the standard universal composability (UC) framework of Canetti [13]. We concentrate on the same model in [14] where the network is asynchronous, the communication is public but ideally authenticated, and the adversary is adaptive in corrupting parties and is active in its control over corrupted parties. Any number of parties can be corrupted and parties cannot erase any of their inner state. We provide a brief description of the UC framework and the ideal commitment functionality for multiple commitments, denoted $\mathcal{F}_{\mathsf{MCOM}}$, in "UC Framework and Ideal Commitment Functionality of Appendix 2".

To construct fully equipped UC commitment, we first put public key pk of ABME in the common reference string. A committer P_i takes tag $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_j)$ 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 $(\mathtt{sid}, \mathtt{ssid}, x, r)$ 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 $(\mathtt{open}, t, x)$, otherwise do nothing. We formally describe our framework for constructing a UC commitment scheme from ABME in Fig. 2.

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 $\iota : \{0,1\}^\kappa \to \mathsf{MSP}$ 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 $(\mathtt{reveal}, \mathtt{sid}, \mathtt{ssid}, P_i, P_j, x)$ 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. 2. Fully Equipped UC commitment from ABME.

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

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

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

In the right interaction: In the real world, $\mathcal{Z}$ receives $(\mathtt{open}, \mathtt{sid}', \mathtt{ssid}', P_a, R, x')$ opened by $\mathcal{A}$ from R at the decommit phase. In the ideal world, $\mathcal{S}$ must correctly extract

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

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

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

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

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

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

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

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

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

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

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

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

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

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

and

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

defined in Sect. 3.3, are statistically indistinguishable in κ.

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

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

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

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

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

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

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

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

5. A General Framework for Constructing ABME

To instantiate an ABME scheme, we use the same construction strategy. We first focus on an instantiation of $\mathsf{pPRF} = (\mathsf{KG}, \mathsf{Spl})$. We then manage to construct an extractable sigma protocol $\Sigma^{\mathsf{ext}} = (\Sigma, \mathsf{Ext})$ on the language derived from pPRF. If we can do so, we say that pPRF and Σ^{ext} are well combined. Then we can always convert such well-combined primitives to an ABME scheme.

We formally say that pPRF and Σ^{ext} are $\mathsf{well\ combined}$ if:

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

Fig. 3. ABME from Σ^{ext} on language derived from pPRF.

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

We can convert these well-combined primitives into an ABME scheme as described in Fig. 3.

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

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

Hence, the resulting scheme meets the notion of ABME.

We note that this conversion originally comes from the transform that converts an ordinary sigma protocol into an instance-dependent commitment scheme [4,41]. We instead apply the transform to an extractable sigma protocol well combined with a pPRF. It is up to each construction how to really instantiate a pPRF and construct Σ^{ext} on it. In the following sections, 6, 7, and "Appendix 3", we provide concrete instantiations of ABME.

6. ABME from Damgård–Jurik PKE with Expansion Factor $O(1)$

We present an ABME scheme with compact ciphertexts, based on Damgård–Jurik public-key encryption scheme [22]. Since ABME implies the fully equipped UC commitments, this scheme can be seen as the *first* fully equipped UC commitment scheme with *expansion factor* $O(1)$. We start by recalling Damgård–Jurik PKE.

6.1. *Damgård–Jurik PKE*

Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a tuple of algorithms of Damgård–Jurik (DJ) PKE [22]. 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 [50]). 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 $R \leftarrow \mathbb{Z}_n^\times$.[1] 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 "pPRF from Waters Signature on General Additively Homomorphic Encryptions of Appendix 4". 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 $\mathbb{Z}_{n^{d+1}}^\times$ is isomorphic to $\mathbb{Z}_{n^d} \times \mathbb{Z}_n^\times$ (the product of a cyclic group of order n^d and a group of order $\phi(n)$), and, for any $d < p, q$, element $g = (1 + n)$ has order n^d in $\mathbb{Z}_{n^{d+1}}^\times$ [22]. Therefore, $\mathbb{Z}_{n^{d+1}}^\times$ is the image of $\mathbf{E}(\cdot; \cdot)$. We note that it is known that $\mathbb{Z}_{n^{d+1}}^\times$ is efficiently samplable and explainable [25,27]. It is also known that DJ PKE is IND-CPA if the DCR assumption (Assumption 7) holds true [22].

6.2. *Construction Idea*

(ABM.gen, ABM.spl) described below forms an analogue of Waters signature scheme [56] defined over a ring equipped with no associated bilinear map, where no signing verification algorithm exists. The "signatures" look pseudorandom assuming that DJ PKE is IND-CPA. We then construct an extractable sigma protocol on the language derived from (ABM.gen, ABM.spl), as discussed in Sect. 1.3.1. Here, the decryption algorithm works only when the matrix below in (2) is invertible, which is equivalent to that $(t, (u_r, u_t)) \in L_{pk}^{\mathsf{ext}}$, where

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

Therefore, we require that (ABM.gen, ABM.spl) should be unforgeable on $\widehat{L}_{pk}^{\mathsf{td}} (= U'_{pk} \backslash L_{pk}^{\mathsf{ext}})$. To prove this statement, we additionally require two more assumptions on DJ PKE, called the non-multiplication assumption and the non-trivial divisor assumption. The first one is an analogue of the DH assumption in an additively homomorphic encryption. If we consider unforgeability on L_{pk}^{td}, this assumption suffices, but we require unforgeability on $\widehat{L}_{pk}^{\mathsf{td}}$. Then we need the non-trivial divisor assumption, too. We formally define these assumptions in "Appendix 4". We note that the assumptions are originally introduced in [37] to obtain the DCR-based ABM-LTF scheme.

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

Note. In "Appendix 3", we present the DDH version of this ABME scheme with expansion factor $O(\kappa/\log\kappa)$. If the reader feels that the proposal here is complicated, we recommend the reader to read "Appendix 3" first, to obtain more intuition behind the construction.

6.3. *ABME from Damgård–Jurik*

- $\mathsf{ABM.gen}(1^\kappa)$: 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 computes $g_1 = \mathbf{E}(x_1; R_1)$ and $g_2 = \mathbf{E}(x_2; R_2)$ by picking up randomly $x_1, x_2 \leftarrow \mathbb{Z}_{n^d}$ and $R_1, R_2 \leftarrow \mathbb{Z}^\times_{n^{d+1}}$. It chooses $\tilde{h} \leftarrow \mathbf{E}(1)$ and $\boldsymbol{y} = (y_0, \ldots, y_\kappa)$ where $y_j \leftarrow \mathbb{Z}_{n^{d+1}}$ for $j = 0, 1, \ldots, \kappa$. It then computes $\boldsymbol{h} = (h_0, \ldots, h_\kappa)$ such that $h_j := \tilde{h}^{y_j}$. Let $H(t) = h_0 \prod_{i=1}^{\kappa} h_i^{t_i} \pmod{n^{d+1}}$, and let $y(t) = y_0 + \sum_{i=1}^{\kappa} y_i t_i \pmod{n^d}$, where $(t_0, \ldots, t_\kappa)$ is the bit representation of t. We note that $H(t) = \tilde{h}^{y(t)}$. It outputs $(pk, sk^{\mathsf{spl}}, sk^{\mathsf{ext}})$ where $pk := (n, d, g_1, g_2, \boldsymbol{h})$, $sk^{\mathsf{spl}} := x_2$, and $sk^{\mathsf{ext}} := (p, q, y_0, \boldsymbol{y})$, where $U'_{pk} := \{0,1\}^\kappa \times (\mathbb{Z}^\times_{n^{d+1}})^2$ that contains the disjoint sets of L^{td}_{pk} and L^{ext}_{pk} as described below.
- $\mathsf{ABM.spl}(pk, sk^{\mathsf{spl}}, t; (r, R_r, R_t))$ where $sk^{\mathsf{spl}} = x_2$: It chooses $r \leftarrow \mathbb{Z}_{n^d}$ and outputs $u := (u_r, u_t)$ such that $u_r := \mathbf{E}(r; R_r)$ and $u_t := g_1^{x_2}\mathbf{E}(0; R_t) \cdot H(t)^r$ where $R_r, R_t \leftarrow \mathbb{Z}^\times_{n^{d+1}}$. We let

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

  We then define

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

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

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

- $\mathsf{ABM.col}_1^{(t,(u_r,u_t))}(pk, sk^{\mathsf{spl}}, (r, R_r, R_t))$ where $sk^{\mathsf{spl}} = x_2$: It picks up $\omega, \eta \xleftarrow{\cup} \mathbb{Z}_{n^d}$, $R'_A, R'_a, R'_b \xleftarrow{\cup} \mathbb{Z}^\times_{n^{d+1}}$. It then computes $A := g_1^\omega \cdot H(t)^\eta \cdot {R'_A}^{n^d}$

$(\mathrm{mod}\ n^{d+1})$, $a := g^{\omega} {R'_a}^{n^d} \pmod{n^{d+1}}$, 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) m. Notice that (A, a, b) implies the following linear system on $\mathbb{Z}_{n^d}$,

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

The matrix is invertible if

$$\mathbf{D}(u_t) \neq (x_1 x_2 + y(t)\mathbf{D}(u_r)) \pmod{p} \text{ and } \mathbf{D}(u_t) \neq (x_1 x_2 + y(t)\mathbf{D}(u_r)) \pmod{q},$$

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

Lemma 1. *(Implicit in [37])* $(\mathsf{ABM.gen}, \mathsf{ABM.spl})$ *is* pPRF *with unforgeability on* $\widehat{L}^{\mathsf{td}}_{pk} (= U'_{pk} \backslash L^{\mathsf{ext}}_{pk})$*, under the assumptions, 7, 8 and 9.*

The proof is given in Sect. 8. By this lemma, we have:

Theorem 2. *The scheme constructed as above is an ABME scheme if the DCR assumption (Assumption 7), the non-trivial divisor assumption (Assumption 8), and the non-multiplication assumption (Assumption 9) 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.

6.4. *ABM-LTF from DCR-based ABME and Vice Versa*

Hofheinz [37] has presented the notion of all-but-many lossy trapdoor function (ABM-LTF). We provide the definition in "All-But-Many Lossy Trapdoor Functions of Appendix 1". 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 [37], unforgeable pPRF can be converted into strongly unforgeable pPRF via a chameleon commitment scheme. Therefore, this difference is not

Table 3. Comparison among ABMEs.

ABME	Expansion factor	Ciphertext length	Message length	pk-length
ABME from [37]	$\geq 31^*$	$(5(d+1)+1)\log n$	$\log n$	$(\kappa+3)d\log n$
Section 6.3 ($d \geq 1$)	$5+1/d$	$5(d+1)\log n$	$d\log n$	$(\kappa+3)d\log n$
Section 7	$5\kappa/\log\kappa$	$(5\ell+5)\log q$	$\ell\log\kappa$	$7\log q$

$*$ $d \geq 5$ is needed

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 [37], 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 Sect. 7.2 in [37], if $d \geq 5$, it can open an ciphertext arbitrarily using Barvinok's algorithm, 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 3 shows the comparison.

On the contrary, our DCR-based ABME (strengthened with strong unforgeability) can be converted to ABM-LTF.[2] 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 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 for any $d \geq 4$ (Table 4).

[2] Our approach is specific to our DCR-based ABME scheme. On the one hand, Hemenway and Ostrovsky [36] have shown that if the message space of lossy encryption is one bit longer than the coin space, the
Footnote 2 continued
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 4. Comparison among ABM-LTFs.

ABM-LTF	Expansion factor	Output length	Input length	Lossyness	Notes
Hof12 [37]	5/3	$(5(d+1)+1)\log n$	$3d\log n$	$(d-3)\log n$	$d \geq 4$
ABM-LTF (Sect. 6)	5/3	$(5(d+1)+1)\log n$	$3(d+1)\log n$	$d\log n$	$d \geq 1$

7. ABME from Twin-Cramer–Shoup with Short Public Key

We construct an ABME scheme from the DDH assumption. The expansion factor of this scheme is not optimal but $O(\kappa/\log\kappa)$. However, this expansion rate is still better than the previous work [14] (with $O(\kappa)$). We note that we provide an alternative ABME scheme with the same expansion factor from the DDH assumption in "Appendix 3", which is the DDH version of the scheme in Sect. 6. So, its public key includes $O(\kappa)$ group elements. On the other hand, this scheme has a short public key only with a constant number of group elements.

We consider the following pPRF. Let Π^{cpa} be an IND-CPA (or even one-way) PKE scheme and let Π^{cca} be an IND-CCA tag-based PKE scheme. Let pk^{cpa} and pk^{cca} be public keys of both schemes, respectively. Then, see $pk = (pk^{\mathsf{cpa}}, pk^{\mathsf{cca}}, \mathbf{E}^{\mathsf{cpa}}(\xi))$ as the public key of pPRF, where ξ is a random message. Then, we see $\mathbf{E}^{\mathsf{cca}}(t, \xi)$ as the output of Spl on tag t, where $sk^{\mathsf{spl}} = \xi$. This indeed forms pPRF. We now describe a concrete construction by using ElGamal PKE and a tag-based version of Twin-Cramer–Shoup PKE [19,21] as ingredients, with a slight optimization.

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 version of Twin-Cramer–Shoup PKE [19,21], where

- $\mathsf{CS.gen}(1^\kappa)$: Via $(pk_{\mathsf{cs}}, sk_{\mathsf{cs}}) \leftarrow \mathsf{CS.gen}(1^\kappa)$, it picks up hash $(pk_{\mathcal{CH}}, sk_{\mathcal{CH}}) \leftarrow \mathsf{CHGen}(1^\kappa)$, generator $g \leftarrow G^\times$, and sets $X = g^x$, $\hat{X} = g^{\hat{x}}$, $Y = g^y$, and $\hat{Y} = g^{\hat{y}}$, where $x, \hat{x}, y, \hat{y} \leftarrow \mathbb{Z}/q\mathbb{Z}$, and finally outputs $pk_{\mathsf{cs}} := (pk_{\mathcal{CH}}, g, X, \hat{X}, Y, \hat{Y})$ 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 $r \xleftarrow{\cup} \mathsf{COIN}_{\mathcal{CH}}$, 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 $v \xleftarrow{\cup} \mathbb{Z}/q\mathbb{Z}$.
- $\mathsf{CS.dec}(sk_{\mathsf{cs}}, t, c)$: Via $m = \mathsf{CS.dec}(sk_{\mathsf{cs}}, t, c)$, 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 $m := \perp$.

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

$\mathsf{pPRF} = (\mathsf{Gen}^{\mathsf{spl}}, \mathsf{Spl})$ 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 $sk_{\mathsf{cs}} = (x, \hat{x}, y, \hat{y})$. It picks up $\zeta \xleftarrow{\cup} G^\times$, $v_0 \xleftarrow{\cup} \mathbb{Z}/q\mathbb{Z}$, 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 $sk^{\mathsf{spl}} := \zeta$.
– $\mathsf{Spl}(pk, sk^{\mathsf{spl}}, t)$: It takes $(pk, sk^{\mathsf{spl}}, t)$ and outputs $u = (r, d, e, \pi_x, \pi_y) = \mathsf{CS.enc}(pk_{\mathsf{cs}}, t, \zeta; v)$ where $v \xleftarrow{\cup} \mathbb{Z}/q\mathbb{Z}$.

We let

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

and

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

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

Lemma 2. *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 pseudorandomness. The unforgeability follows from the following analysis.

Let us define G_0 as the original unforgeability game, in which the challenger sets up all secrets and public parameter $pk = (pk_{\mathsf{cs}}, d_0, e_0)$. The challenger 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 challenger 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 $\zeta \xleftarrow{\cup} G^\times$ and sets $(d_0, e_0) := (g^{v_0}, \zeta^{-1}X^{v_0})$ where $v_0 \xleftarrow{\cup} \mathbb{Z}/q\mathbb{Z}$. 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 $\epsilon_{1,i}(\kappa) - \epsilon_{1,i+1}(\kappa) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{TwinCS},A}(\kappa)$, for every $i \in \{0, \ldots, q-1\}$, which is negligible in κ if the DDH assumption holds on G and $\mathcal{CH}$ is a chameleon hash commitment scheme. We note that B needs the decryption oracle only once, to check that c' is a ciphertext of ζ.

In Game G_2, the challenger behaves as follows: It is given pk_{cs} and $|\zeta|$ as input, chooses a random tag t, and obtains ciphertext (d, e, π_x, π_y) of a random message ζ^{-1} on tag t. It then sets $(d_0, e_0) := (d, e)$. Here, the challenger is not given ζ. For every query t_i of A, $1 \leq i \leq q$, the challenger returns $\mathsf{CS.enc}(pk_{\mathsf{cs}}, t_i, 0^{|\zeta|})$. 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 challenger submits it to the decryption oracle and outputs its reply. We note that the challenger did not reveal any information on t to A, because it feeds only (d_0, e_0) to A. Hence, it holds that $t' \neq t$ with probability $1 - \frac{q}{2^\kappa}$. If c' is a ciphertext of ζ, the challenger results in decrypting $c = (d, e, \pi_x, \pi_y)$ on tag t, which is bounded by the advantage of an adversary that breaks one-wayness of TwinCS in the chosen ciphertext attack. The advantage is bounded by twice of that of IND-CCA security of TwinCS.

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

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

$$L^{\mathsf{td}}_{pk} = \widehat{L}^{\mathsf{td}}_{pk} = \Big\{ (t, (r, d, e, \pi_x, \pi_y)) \,|\, \exists\, (\tilde{v}, v) : d_0 d = g^{\tilde{v}},\ e_0 e = h^{\tilde{v}},\ d = g^v,\ \pi_x = (X^\tau \hat{X})^v,\ \text{and}\ \pi_y = (Y^\tau \hat{Y})^v \Big\}.$$

  We note that $\tilde{v} = v_0 + v$. We define $L^{\mathsf{ext}}_{pk} = U'_{pk} \backslash \widehat{L}^{\mathsf{td}}_{pk}$.
- $\mathsf{ABM.enc}^{(t,u)}(pk, m; (\hat{z}, z))$: 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, $\tilde{z}, z \xleftarrow{\cup} G^\ell$,

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

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

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

– $\mathsf{ABM.dec}^{(t,u)}(sk^{\mathsf{ext}}, c)$: 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

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

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

– $\mathsf{ABM.col}_1^{(t,u)}(pk, t, sk^{\mathsf{spl}}, v; (\tilde{\boldsymbol{w}}, \boldsymbol{w}))$: It picks up $\tilde{w}_i, w_i \xleftarrow{\cup} \mathbb{Z}/q\mathbb{Z}$ 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 $(t, (r, d, e, \pi_x, \pi_y)) \in L_{pk}^{\mathsf{td}}$. 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 $(t, (r, d, e, \pi_x, \pi_y)) \in L_{pk}^{\mathsf{td}}$ if and only if $\mathrm{rank}(A(t, u)) = 1$ 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 $(t, (r, d, e, \pi_x, \pi_y)) \in L_{pk}^{\mathsf{ext}} (= U'_{pk} \backslash L_{pk}^{\mathsf{td}})$, $\mathrm{rank}(A(t, u)) = 2$ 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 the simulator can decrypt *valid* ciphertexts correctly, because an adversary cannot correctly open an invalid ciphertext on $(t, u) \in L^{\mathsf{ext}}_{pk}$.

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

This scheme has a ciphertext consisting of $5\ell + 4$ group elements plus $|\mathsf{COIN}_{\mathcal{CH}}|$-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 [14] 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 trapdoor commitment. This expansion factor in [14] is strict, by construction, which cannot be improved.

8. Fully Equipped UC Commitment from Trapdoor Permutations

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

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

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

Let $\mathcal{F} = \{(f, f^{-1}) \mid f : \{0,1\}^\kappa \to \{0,1\}^\kappa\}_{\kappa\in\mathbb{N}}$ 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 $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be the generalized version of Blum–Goldwasser cryptosystem [8] 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 $\{0,1\}^{\kappa+k}$ [14,27]. Let us denote by $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ a pseudorandom 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 $\mathcal{F}$. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be the Blum–Goldwasser cryptosystem mentioned above. Let F be a pseudorandom function derived from f'. It then picks up random $s \leftarrow \{0,1\}^\kappa$ and encrypt it to $e' = \mathbf{E}_{f'}(s; r)$. It outputs $(pk, sk^{\mathsf{spl}}, sk^{\mathsf{ext}})$, where $pk = (F, f, f', e')$, $sk^{\mathsf{spl}} = (s, r)$, and $sk^{\mathsf{ext}} = f^{-1}$. We define $U'_{pk} = \{0,1\}^\kappa \times \{0,1\}^k$.
– $\mathsf{ABM.spl}(pk, sk^{\mathsf{spl}}, t)$: It takes tag $t \in \{0,1\}^\kappa$ and outputs $u = F_s(t)$ where $sk^{\mathsf{spl}} = (s, r)$. We define

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

– $\mathsf{ABM.enc}^{(t,u)}(pk, x)$: 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 $e' = \mathbf{E}_{f'}(s; r)$, by using the NP-reduction. We note that one can find such G without knowing (s, r). In addition, if such (s, r) does not exist for given (t, u), G so obtained does not have a Hamiltonian cycle.

  • To encrypt 0, it picks a random permutation $\pi = (\pi_1, \ldots, \pi_q)$ 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 $(i \in [q])$. It then chooses a randomly labeled Hamiltonian cycle, and for all the entries in the adjacency matrix corresponding to edges on the Hamiltonian cycle, it encrypts 1's. For all the other entries, it picks up random $\kappa+1$-bit strings. It outputs $\{A_i\}_{i\in[q]}$ and $\{B_{i,j}\}_{i,j\in[q]}$, where a Hamiltonian cycle is embedded in $\{B_{i,j}\}_{i,j\in[q]}$, but the other strings are merely random strings.

  This encryption procedure is the same as the adaptive Hamiltonian commitment protocol in [16], except that a commitment in our scheme is encrypted under a public key f independent of F.
– $\mathsf{ABM.dec}^{(t,u)}(sk, c)$: To decrypt $c = (\{A_i\}_{i\in[q]}, \{B_{i,j}\}_{i,j\in[q]})$, it firstly decrypts all elements to retrieve π and matrix H, using $sk = f^{-1}$. Then it checks that $H = \pi(G)$. If it holds, it outputs 0; otherwise, 1.
– $\mathsf{ABM.col}_1^{(t,u)}(pk, sk^{\mathsf{spl}}, v)$: It first obtains a graph G (of q nodes) so that finding a Hamiltonian cycle in G is equivalent to finding $sk^{\mathsf{spl}} = (s, r)$ such that $u = F_s(t)$ and $e' = \mathbf{E}_{f'}(s; r)$, by using the NP-reduction. It picks a random permutation $\pi = (\pi_1, \ldots, \pi_q)$ 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, ξ) where $c = (\{A_i\}_{i\in[q]}, \{B_{i,j}\}_{i,j\in[q]})$ and $\xi = ((t, u), \zeta, \pi)$. Here ζ denotes the Hamiltonian cycle of G.
– $\mathsf{ABM.col}_2(\xi, x)$: If $x = 0$, it opens π and every entry of the adjacency matrix; otherwise, if $x = 1$, it opens only the entries corresponding to the Hamiltonian cycle ζ in the adjacency matrix.

Then, we apply this weak ABME scheme to our framework (Fig. 2).

Theorem 4. *The scheme in Fig. 2 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, in the trapdoor mode when $(t, u) \in L_{pk}^{\mathsf{td}}$, the output of ABM.col is statistically indistinguishable from that of ABM.enc. However, this case only guarantees computational difference. To show that the environment views in both games are computationally indistinguishable, we need to construct, for contradiction, a distinguisher that can distinguish the output of ABM.col from the output of ABM.enc without knowing sk^{spl}, while it can extract the values committed to by corrupted parties at the same time. Fortunately, in this construction, the decryption key $sk^{\mathsf{ext}} = f^{-1}$ is independent of the equivocable key $sk^{\mathsf{spl}} = (s, r)$. It is not the case of the rest of our constructions, in which one can obtain sk^{spl} if one knows sk^{ext}. Therefore, we require statistical closeness in there. Hence, we can construct a distinguisher that takes $sk^{\mathsf{ext}} = f^{-1}$ and starts either with the ideal world or Hybrid Game 1. Here, the environment views in both games are bounded by the distinguisher's advantage, which is negligible. □

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

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

This construction does not require non-interactive zero-knowledge proof systems. To the best of our knowledge, the most efficient non-interactive zero-knowledge proofs from trapdoor permutations is given by Kilian and Petrank [43], which requires a CRS size of $\omega(|C|\kappa^2 \log \kappa)$ and a proof size of $\omega(|C|\kappa^2 \log \kappa)$, where $|C|$ is the circuit size of the statement. We compare our construction with the previous result [16] with the most efficient NIZK proof system in Table 5.

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

Schemes	CRS size	Communication	Complexity of each user
CLOS02 [16]	$\omega(\kappa^3 \log(\kappa))$	$\omega(\lambda \cdot q^2 \kappa^3 \log \kappa)$	$\lambda q^2 T_{\mathsf{NP}} + \omega(\lambda q^2 T_{\mathsf{tdp}}(\kappa^3 \log \kappa))$
Section 8	$O(\kappa)$	$O(\lambda \cdot q^2 \kappa)$	$T_{\mathsf{NP}} + \lambda q^2 T_{\mathsf{tdp}}(\kappa)$

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

Acknowledgements

We thank Kirill Morozov and his students for nice feedback in the early version of this work. We thank Dennis Hofheinz for valuable discussion. Finally, we thank the anonymous referees and editor Serge Fehr for useful comments, which help us to improve the final version significantly.

Appendix 1: Some Other Definitions

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, $\Pr[\iota \leftarrow \mathcal{I}_\kappa;\ (x, y) \leftarrow C_\kappa(H_\iota) : x \neq y \wedge H_\iota(x) = H_\iota(y)] = \mathsf{negl}(\kappa)$.

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,

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

Tag-Based PKEs

A Tag-PKE $\Pi = (\mathsf{Tag.Gen}, \mathsf{Tag.Enc}, \mathsf{Tag.Dec})$ is a tag-based PKE [44,46,53] 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 $r \xleftarrow{\cup} \mathsf{COIN}$, 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 [46], called weak CCA security [44]. 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 $\mathsf{Expt}^{\mathsf{ind\text{-}cca}}_{\Pi,A,b}(\kappa)$.

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

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

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

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

All-But-Many Lossy Trapdoor Functions

We recall all-but-many lossy trapdoor functions (ABM-LTF) [37], by slightly modifying the notation to fit our purpose.

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 $(pk, sk^{\mathsf{spl}}, sk^{\mathsf{ext}})$, where pk defines a set U_{pk}. We let $U'_{pk} = \{0,1\}^\kappa \times U_{pk}$. pk also determines two disjoint sets, L^{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 $(pk, sk^{\mathsf{spl}}, t)$, 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.,

$$
L^{\mathsf{loss}}_{pk}(t) := \left\{ u \in U_{pk} \mid \exists\, sk^{\mathsf{spl}},\ \exists\, v :\ u = \mathsf{ABM.spl}\left(pk, sk^{\mathsf{spl}}, t; v\right) \right\}.
$$

  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^{ext}, (t, u), and c, and computes $x = \mathsf{ABM.inv}^{(t,u)}(sk^{\mathsf{ext}}, c)$.

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 pseudorandom function (pPRF), as defined in Sect. 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 [37].
2. **Inversion.** For every $\kappa \in \mathbb{N}$, every $(pk, sk^{\mathsf{spl}}, sk^{\mathsf{ext}})) \in \mathsf{ABM.gen}(1^\kappa)$, every $(t, u) \in L^{\mathsf{inj}}_{pk}$, and every $x \in \mathsf{MSP}$, it always holds that

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

3. ℓ-**Lossyness.** For every $\kappa \in \mathbb{N}$, every $(pk, sk^{\mathsf{spl}}, sk^{\mathsf{ext}}) \in \mathsf{ABM.gen}(1^\kappa)$, 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 [37] 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.

Appendix 2: UC Framework and Fully Equipped UC Commitments from ABME

UC Framework and Ideal Commitment Functionality

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

In slightly more detail, the task of honest parties in the ideal world is only to convey inputs from the environment to the ideal functionality and vice versa (i.e., the honest parties in the ideal world communicate only with the environment and ideal functionalities). The environment may order the adversary to corrupt any honest party in any timing during the execution of the protocol ($\mathsf{adaptive\ corruption}$), and it may receive the inner state of the honest party from the adversary. Therefore, the ideal-world simulator must

simulate the inner state of the real-world honest party as if it comes from the real world, because the honest parties in the ideal world do nothing except storing inputs to them). The inner state of the real-world honest party includes randomness it has used. We insist that honest parties cannot erase any of its state (non-erasure model).

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

Our protocols are executed in the common reference string (CRS) model. This means that the protocol π is run in a hybrid model where the parties have access to an ideal functionality $\mathcal{F}_{\mathsf{crs}}$ 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 $\text{HYBRID}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$. 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

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

The importance of the universal composability framework is that it satisfies a composition theorem that states that any protocol that is universally composable is secure when it runs concurrently with many other arbitrary protocols. For more details, see [13].

We consider UC commitment schemes that can be used repeatedly under a single common reference string (reusable common reference string). The multi-commitment ideal functionality $\mathcal{F}_{\mathsf{MCOM}}$ from [16] is the ideal functionality of such commitments, which is given in Fig. 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.

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

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

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

Fig. 4. The ideal multi-commitment functionality.

Proof of Theorem 1

Theorem 1 (restated) The proposed scheme in Fig. 2 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 $u = \mathsf{ABM.spl}(pk, sk^{\mathsf{spl}}, t; v)$ so that $(t,u) \in L^{\mathsf{td}}_{pk}$, where $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_j)$, and computes $(c, \xi) = \mathsf{ABM.col}_1^{(t,u)}(pk, sk^{\mathsf{spl}}, v)$, 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 $(\mathtt{sid}, \mathtt{ssid}, P_i, P_j, x)$, whose 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 R such that $u = U_{pk}(t; R)$, which can be efficiently computable because U_{pk} is an explainable domain. Finally, it reveals (x, r, R) to $\mathcal{A}$.
- **Simulating the commit phase when the committer** P_i **is corrupted and the receiver** P_j **is honest:** Upon receiving $(\mathtt{sid}, \mathtt{ssid}, (t,u), c)$ from $\mathcal{A}$, $\mathcal{S}$ decrypts $x = \mathsf{ABM.dec}^{(t,u)}(sk^{\mathsf{ext}}, c)$. 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 $(\mathtt{open}, \mathtt{sid}, \mathtt{ssid})$ to $\mathcal{F}_{\mathsf{MCOM}}$. ($\mathcal{F}_{\mathsf{MCOM}}$ follows its codes: If a tuple $(\mathtt{sid}, \mathtt{ssid}, P_i, P_j, x)$ with the same $(\mathtt{sid}, \mathtt{ssid})$ was previously stored by $\mathcal{F}_{\mathsf{MCOM}}$, $\mathcal{F}_{\mathsf{MCOM}}$ sends $(\mathtt{sid}, \mathtt{ssid}, P_i, P_j, x)$ 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 $(\mathtt{sid}, \mathtt{ssid}, (t,u,c))$ 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 $p(\cdot)$,

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

To prove this, we then consider a sequence of the following games on which the probability spaces are identical, but we change the rules of games step by step.

Hybrid Game 1. In this game, the ideal commitment functionality, denoted $\mathcal{F}^1_{\mathsf{MCOM}}$, 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 commit phase, $\mathcal{F}^1_{\mathsf{MCOM}}$ gives simulator $\mathcal{S}_1$ the committed value x together with $(\mathtt{receipt}, \mathtt{sid}, \mathtt{ssid}, P_i, P_j)$. $\mathcal{S}_1$ then sets up $(t,u) \in L^{\mathsf{td}}_{pk}$ in the same way as $\mathcal{S}$ does (using sk^{spl}), but $\mathcal{S}_1$ instead computes c as $c = \mathsf{ABM.enc}^{(t,u)}(pk, x; r)$, by picking up $r \xleftarrow{\cup} \mathsf{COIN}^{\mathsf{enc}}$. When simulating the decommit phase or simulating adaptive corruption of P_i before the decommit phase, $\mathcal{S}_1$ reveals (u, x, r, R) after computing R such that $u = U_{pk}(t; R)$.

Consider the simulation that honest P_i opens commitment (t, u, c) in both games. The distribution of (u, c, r) on $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_j)$ as generated in Hybrid Game 1 is statistically indistinguishable from those on the same t as generated in the ideal world, because the two distribution ensembles, $\{\mathsf{dist}^{\mathsf{col}}(pk, t, sk^{\mathsf{spl}}, sk^{\mathsf{ext}}, x)\}_{\kappa\in\mathbb{N}}$ and $\{\mathsf{dist}^{\mathsf{enc}}(pk, t, sk^{\mathsf{spl}}, sk^{\mathsf{ext}}, x)\}_{\kappa\in\mathbb{N}}$, defined in Sect. 3.3, are statistically indistinguishable in κ. So, we have

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

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

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

and thus, $\mathrm{BD}_1 = \mathrm{BD}_2$. 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, $\mathrm{BD} := \mathrm{BD}_1 = \mathrm{BD}_2$.

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

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

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

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

Lemma 3. *Event* BD *occurs 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. Since BD occurs with the same probability in both games, we consider the probability in Hybrid Game 2. We construct the following algorithm B_0 that takes pk from $\mathsf{ABM.gen}$ 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 oracle $\mathsf{ABM.spl}(pk, sk^{\mathsf{spl}}, \cdot)$ as follows:

In the case when P_i is honest: In the commit phase when $\mathcal{Z}$ sends $(\mathtt{commit.sid}, \mathtt{ssid}, P_i, P_j, x)$ to $\mathcal{F}^2_{\mathsf{MCOM}}$ (via honest P_i), B_0 submits $t = (\mathtt{sid}, \mathtt{ssid}, P_i, P_j)$ to $\mathsf{ABM.spl}(pk, sk^{\mathsf{spl}}, \cdot)$ 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 a commitment in the same way as $\mathcal{S}_2$ $(= \mathcal{S}_1)$ does.

In the case where P_i is corrupted and P_j is honest: In the commit phase when corrupted P_i controlled by $\mathcal{A}$ sends a commitment (t, u, c) to $\mathcal{S}_2$ as it expects to send to honest P_j, B_0 simply plays the roles of $\mathcal{S}_2$ and $\mathcal{F}^2_{\mathsf{MCOM}}$. Later, in the opening phase when corrupted P_i controlled by $\mathcal{A}$ sends $(\mathtt{sid}, \mathtt{ssid}, x', r)$ to $\mathcal{S}_2$ as it expects to send to honest P_j, B_0 simply plays the role of $\mathcal{F}^2_{\mathsf{MCOM}}$.

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

We now construct an algorithm B_χ, where $\chi \in [q_A]$, that is the same as B_0 except that it aborts and outputs (t, u) when $\mathcal{A}$ generates χth (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

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

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

By this lemma, we have

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

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

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

We note that in Hybrid Game 2, $\mathcal{S}_2$ makes use of sk^{spl} only when it computes $u \leftarrow \mathsf{ABM.spl}(pk, sk^{\mathsf{spl}}, t)$, whereas in Hybrid Game 3, $\mathcal{S}_3$ does not use sk^{spl} any more. The difference of the views of $\mathcal{Z}$ between these two games is bounded by pseudorandomness 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 $\mathsf{ABM.spl}(sk^{\mathsf{spl}}, \cdot)$ or $U_{pk}(\cdot)$. When D has access to $\mathsf{ABM.spl}(sk^{\mathsf{spl}}, \cdot)$, it simulates Hybrid Game 2; otherwise, it simulates Hybrid Game 3. Therefore, we have

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

$\text{Hybrid}^{\mathcal{F}_{\mathsf{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}$ Game. This is the real world in the CRS model (or in the CRS hybrid model), where a honest party activated for the commitment functionality follows the code of the protocol in Fig. 2. The common reference string functionality $\mathcal{F}_{\mathsf{CRS}}$ parameterized by $\mathsf{ABM.gen}$ is given in Fig. 5.

It is obvious by construction that two worlds are identical.

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

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

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

- $\mathcal{F}_{\mathsf{CRS}}$ runs $(pk, sk^{\mathsf{spl}}, sk^{\mathsf{ext}}) \leftarrow \mathsf{ABM.gen}(1^\kappa)$; 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.

In the end, we have

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

□

Appendix 3: ABME from Waters Signature in Pairing-Free Prime-Order Group

We present an ABME scheme derived from an analogue of Waters Signature [56] defined over a pairing-free cyclic group on which the DDH assumption holds. The expansion factor of this scheme is $O(\kappa/\log\kappa)$, but slightly better than the previously known construction [14] (with $O(\kappa)$). This scheme can be seen as a warm-up of the proposal in Sect. 6.

pPRF *from Waters Signature in Pairing-Free Prime-Order Group*

The construction idea is to use as pPRF Waters signature [56] defined in a *pairing-free* prime order group. Since there is no associated bilinear map, there is no verification algorithm. Instead, the output of the pairing-free analogue of Waters signature looks pseudorandom, due to the DDH assumption. On the other hand, it inherits unforgeability from the original Waters signature scheme.

Let g be a generator of a multiplicative group G of prime order q, on which the DDH assumption holds. For $\kappa+1$ elements in G, let us define $H(t)=h_0\prod_{i=1}^{\kappa}h^{t_i}$, where $t=(t_1,\ldots,t_\kappa)\in\{0,1\}^\kappa$ in which $t_i\in\{0,1\}$ denotes ith-bit representation of string t.

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

Theorem 5. *The above construction is* pPRF *under the DDH assumption.*

Proof. Spl is the same as Waters signature scheme when applied to a pairing-free group. So, the unforgeability is immediately guaranteed if the computational DH assumption holds true. Pseudorandomness also holds under the DDH assumption because $(g^r,H(t)^r)$ is computationally indistinguishable from two independent random elements in G: To explain more details, suppose that $(g,\hat{g},h,\hat{h})$ is a tuple of four group elements in G, which is either a DDH instance or a random tuple. To break the DDH problem, a simulator sets $g_1:=g^{x_1}$, $g_2:=g^{x_2}$, $K:=g^{x_1x_2}$, and $h_i:=\hat{g}^{a_i}$, where $x_1,x_2,a_0,\ldots,a_\kappa \leftarrow G$. It then runs adversary A on the above parameters, where A is an adversary to break pseudorandomness. For any query t, the simulator picks up random $s,v\leftarrow\mathbb{Z}/q\mathbb{Z}$ and returns (u_r,u_t) such that $u_r=g^sh^v$ and $u_t=K\cdot(\hat{g}^{a_0})^s(\hat{h}^{a_0})^v\prod_{i\geq1}(\hat{g}^{a_i})^{st_i}(\hat{h}^{a_i})^{vt_i}$. We note that $u_r=g^{s+\log_g(h)v}$ and $u_t=KH(t)^{s+\log_{\hat{g}}(\hat{y})v}$. Hence, (u_r,u_t) is a Waters

signature if $(g, \hat{g}, h, \hat{h})$ is a DDH tuple; otherwise it is a pair of two random elements. The simulator outputs the same bit that A outputs. The simulator's advantage is the same as that of A. Under the DDH assumption, its advantage is bounded by a negligible (in κ) function. Therefore, it also satisfies pseudorandomness. Hence, the scheme above is an instantiation of pPRF if the DDH assumption holds true. □

ABME from Waters Signature

Let g be a generator of a multiplicative group G of prime order q, where we assume that G is efficiently samplable. We let $g_i = g^{x_i}$ $(i = 1, 2)$ and $\boldsymbol{h} = (h_0, \ldots, h_\kappa)$ with $h_j = g^{y_j}$, where $x_1, x_2, y_0, y_1, \ldots, y_\kappa \leftarrow \mathbb{Z}/q\mathbb{Z}$. We write $t = (t_1, \ldots, t_\kappa) \in \{0,1\}^\kappa$ where $t_i \in \{0,1\}$ $(i \in [\kappa])$. We let $y(t) = y_0 + \sum_{i=1}^{\kappa} t_i y_i \pmod q$ and define $H(t) = h_0 \prod_{i=1}^{\kappa} h_i^{t_i}$, that is, $H(t) = g^{y(t)}$. We let $U'_{pk} = \{0,1\}^\kappa \times G^2$. Then we define the set under $pk = (g, g_1, g_2, \boldsymbol{h})$ as $L^{\mathsf{td}}{}_{pk} = \{(t,u) \mid (t,u) \in \{0,1\}^\kappa \times L_{pk}(t)\}$ such that

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

We let $\widehat{L}^{\mathsf{td}}_{pk} = L^{\mathsf{td}}_{pk}$ and define $L^{\mathsf{ext}} = U'_{pk} \backslash L^{\mathsf{td}}_{pk}$. We note that as mentioned above, Waters signature defined on a pairing-free cyclic group on which the DDH assumption holds forms a pPRF. We then construct an ABME scheme as follows.

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

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

  It aborts if it cannot find such x in a-priori bounded time $T = O(2^\lambda)$.
- $\mathsf{ABM.col}_1^{(t,u)}(sk^{\mathsf{spl}}, v)$ where $sk^{\mathsf{spl}} = x_2$: It picks up at random $\omega, \eta \leftarrow \mathbb{Z}/q\mathbb{Z}$ and computes $A = g_1^\omega H(t)^\eta$, $a = g^\omega$, and $b = g^\eta$. It outputs $c = (A, a, b)$ and $\xi = (x_2, t, u, v, \omega, \eta)$.
- $\mathsf{ABM.col}_2(\xi, x)$: To open c to $x \in \{0,1\}^\lambda$, it computes $z = \omega - m x_2 \bmod q$ and $s = \eta - m v \bmod q$ and outputs (z, s).

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

In the trapdoor mode when $(t, u) \in L_{pk}^{\mathsf{td}}$, we can consider a canonical sigma protocol so that the prover knows (x_2, v) such that $u_t = g_1^{x_2} H(t)^v$, $g_2 = g^{x_2}$, and $u_v = g^v$. Then, the first message of the canonical sigma protocol is (A, a, b), where $A = g_1^{\omega} H(t)^{\eta}$, $a = g^{\omega}$, and $b = g^{\eta}$ over randomly chosen $\omega, \eta \in \mathbb{Z}/q\mathbb{Z}$. For any challenge $m \in \{0,1\}^{\kappa}$, the answer can be computed by $z = \omega - mx_2$ and $s = \eta - mv$. It is verified as $A = g_1^z H(t)^s u_t^m$, $a = g^z g_2^m$, and $b = g^s u_v^m$.

In the decryption mode when $(t, u) \in L_{pk}^{\mathsf{ext}} (= U'_{pk} \backslash L_{pk}^{\mathsf{td}})$, the first message (A, a, b) from the simulator for the above canonical sigma protocol commits to m in the perfect binding manner. We now define ω, η, v as $a = g^{\omega}$, $b = g^{\eta}$, and $u_v = g^v$. Then, x'_2 is uniquely defined as $u_t = g_1^{x'_2} H(t)^v$. If (A, a, b) can be opened with (z, s, m), it implies that

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

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

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

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

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

Since (ABM.gen, ABM.spl) is pPRF (under the DDH assumption), the proposed scheme is an ABME scheme.

Theorem 6. *The scheme as above is an ABME if the DDH assumption holds true.*

Appendix 4: The Proof of Lemma 1

In this section, we provide the formal proof of Lemma 1. Although the proof is implicitly shown in [37], we provide it for completeness.

To prove the statement, we require two more assumptions related to DJ PKE, along with the standard DCR assumption, called the non-multiplication assumption and the non-trivial divisor assumption, which originally appeared in [37]. We first prove that our target scheme is a pPRF with unforgeability on L_{pk}^{td} (not on $\widehat{L}_{pk}^{\mathsf{td}}$) under the DCR assumption and the non-multiplication assumption. We prove this in a generalized case that DJ PKE is replaced with an arbitrary enhanced additive homomorphic encryption scheme. We then prove that the resulting scheme has unforgeability on $\widehat{L}_{pk}$, additionally assuming the non-divisor assumption.

Assumptions and Some Useful Lemmas

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

Assumption 7. (*Decisional Composite Residue Assumption* [50]) We say that the DCR assumption holds if for every PPT A, there exists a key-generation algorithm $\mathbf{K}$ such that $\mathsf{Adv}^{\mathsf{dcr}}_{A}(\kappa) =$

$$\Pr\left[\mathsf{Expt}^{\mathsf{dcr}-0}_{A}(\kappa)=1\right]-\Pr\left[\mathsf{Expt}^{\mathsf{dcr}-1}_{A}(\kappa)=1\right]$$

is negligible in κ, where

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

Assumption 8. (*Non-Trivial Divisor Assumption* [37]) 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

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

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

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

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

Assumption 9. (*Non-Multiplication Assumption* [37]) We say that the non-multiplication assumption holds on DJ PKE $\Pi^{(d)}$ if for every PPT adversary A, the advantage of A, $\mathsf{Adv}^{\mathsf{mult}}_{A,\Pi^{(d)}}(\kappa)=\mathsf{negl}(\kappa)$, where

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

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 (Fig. 6).

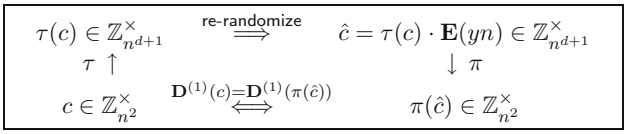

Fig. 6. Diagram of lifting up and re-randomization.

Lemma 5. *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 [22] to prove that $\Pi^{(d)}$ for any $d \geq 1$ is IND-CPA secure under the DCR assumption. In order to prove Lemmas, 4 and 5, these lemmas are essential.

Lemma 6. (from [22,37]) *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 6, we have the following lemma.

Lemma 7. (from [22,37]) *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 $y \xleftarrow{\cup} \{0, 1, \ldots, n^{d-1} - 1\}$; set $\hat{c} = \tau(c) \cdot \mathbf{E}^{(d)}(yn)$; output $\hat{c}$.

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

pPRF *from Waters Signature on General 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 10. (*Generalized 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 generalized non-multiplication assumption holds on Π if for every non-uniform PPT algorithm A, $\mathsf{Adv}^{\mathsf{mult}}_A(\kappa) = \mathsf{negl}(\kappa)$, where $\mathsf{Adv}^{\mathsf{mult}}_A(\kappa) \triangleq$

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

This assumption is a generalized version of Assumption 9.

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 $L_{pk}(t)$ such that

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

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

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

Proof. The proof of pseudorandomness is almost straightforward: Suppose that pk is generated by $\mathbf{K}(1^\kappa)$. Let S be a simulator such that it breaks IND-CPA of Π using A, where A is an adversary to output 1 when it decides that it has access to a pPRF. We run S on pk. It picks up at random $x_1, x_2 \leftarrow X$, $h_0, h_1, \ldots, h_\kappa \leftarrow Y$, 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, $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi} S(\kappa) = \Pr[S = 1 \mid b = 1] - \Pr[S = 1 \mid b = 0] = \Pr[A = 1 \mid b = 1] - \Pr[A = 1 \mid b = 0] = \mathsf{Adv}_{\mathsf{pprf}} A(\kappa)$. Therefore, $\mathsf{Adv}_{\mathsf{pprf}} A(\kappa) = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi} S(\kappa) = \mathsf{negl}(\kappa)$.

The proof of unforgeability on this scheme is substantially similar to that in [5,10,56]. 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 [5].

Lemma 8. *[5]. Let $Q(\boldsymbol{t})$ be the event in game G_1 mentioned above. Then,*

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

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

Lemma 9. *[5]. Let n, θ, κ be positive integers, such that $\kappa\theta < n$. Let $y_0, y_1, \ldots, y_\kappa$ be elements in the domains mentioned above and let $y(t) = y_0 + \sum t_i \cdot y_i \in \mathbb{Z}$. Then, for every $t_0, \ldots, t_\kappa \in \{0,1\}^\kappa$, we have*

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

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 right-hand side under sk is

$$\begin{aligned} x_1x_2 + (a(t) + y(t)x_2)r &= x_1x_2 + (a(t) + y(t)x_2) \cdot \left(-\frac{x_1}{y(t)} + r'\right) \\ &= -\frac{a(t)}{y(t)} \cdot x_1 + (a(t) + y(t)x_2) \cdot r'. \end{aligned}$$

Therefore, the right-hand 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 [10]. 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. □

Completing Proof of Lemma 1

We now complete the proof of Lemma 1. We note that we have already shown in Theorem 11 that the proposed pPRF scheme in Sect. 6 is unforgeable on L_{pk}^{td} under Assumption 7 and Assumption 9, since DJ PKE is IND-CPA under Assumption 7 and Assumption 10 is a generalized version of Assumption 9. We now show the following.

Lemma 1 (restated) $\mathsf{pPRF} = (\mathsf{ABM.gen}, \mathsf{ABM.spl})$ is a probabilistic PRF with unforgeability on $\widehat{L}_{pk}^{\mathsf{td}}$ as defined above, under the assumptions, 7, 8 and 9.

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

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

where $\mathbf{D}$ is the decryption algorithm of $\Pi^{(d)}$. By construction, it is clear that $L_{pk}^{(d)} = L_{pk}^{\mathsf{td}}$. We note that $L_{pk}^{\mathsf{td}} \subset L_{pk}^{(1)}$. We remark 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 11, 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 6. 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 $\frac{u_t(t_0)}{u_r^{a(t_0)}} \bmod n$. Therefore, the probability of A outputting such pairs is negligible; otherwise, it contradicts Assumption 9.

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 8. □

References

[1] M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, Sphf-friendly non-interactive commitments. in K. Sako, P. Sarkar, editors, *ASIACRYPT 2013 (1)*, *Lecture Notes in Computer Science*, vol. 8269 (Springer, Heidelberg, 2013), pp. 214–234

[2] D. Beaver, Correlated pseudorandomness and the complexity of private computations, in *STOC '96*, (ACM, Philadelphia, Pennsylvania, USA, 1996)

[3] M. Bellare, D. Hofheinz, S. Yilek, Possibility and impossibility results for encryption and commitment secure under selective opening, in *Joux*, vol. 42, pp. 1–35

[4] M. Bellare, S. Micali, R. Ostrovsky, Perfect zero-knowledge in constant rounds, in *STOC '90*, (ACM, 1990), pp. 482–493

[5] M. Bellare, T. Ristenpart, Simulation without the artificial abort: simplified proof and improved concrete security for waters' ibe scheme, in *Joux*, vol. 42, pp. 407–424

[6] M. Bellare, P. Rogaway, Random oracle are practical: a paradigm for designing efficient protocols, in *CCS '93*, (ACM, 1993), pp. 62–73

[7] O. Blazy, C. Chevalier, D. Pointcheval, D. Vergnaud. Analysis and improvement of lindell's uc-secure commitment schemes, in M. J. Jacobson, Jr., M. E. Locasto, P. Mohassel, R. Safavi-Naini, editors, *ACNS 2013*, *Lecture Notes in Computer Science*, vol. 7954 (Springer, Heidelberg, 2013), pp. 534–551

[8] M. Blum, S. Goldwasser, An efficient probabilistic public-key encryption scheme which hides all partial information, in G. Robert Blakley, D. Chaum, editors, *CRYPTO '84*, *Lecture Notes in Computer Science*, vol. 196 (Springer, Heidelberg, 1985), pp. 289–299

[9] D. Boneh, editor, Advances in cryptology—CRYPTO 2003, in *23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 2003, Proceedings*, *Lecture Notes in Computer Science*, vol. 2729 (Springer, Heidelberg, 2003)

[10] D. Boneh, X. Boyen, Efficient selective-ID secure identity based encryption without random oracles, in *Cachin and Camenisch*, vol. 11, pp. 223–238

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

[12] J. Camenisch, V. Shoup, Practical verifiable encryption and decryption of discrete logarithms, in *Boneh*, vol. 9, pp. 289–299

[13] R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, in *FOCS 2001*, pp. 136–145. IEEE Computer Society, 2001. The full version available at Cryptology ePrint Archive http://eprint.iacr.org/2000/067.

[14] R. Canetti, M. Fischlin, Universally composable commitments, in J. Kilian, editor, *CRYPTO 2001*, *Lecture Notes in Computer Science*, vol. 2139 (Springer, Heidelberg, 2001), pp. 19–40

[15] R. Canetti, A. Jain, A. Scafuro, Practical UC security with a global random oracle, in G.-J. Ahn, M. Yung, N. Li, editors, *CCS 2014* (ACM, 2014), pp. 597–608

[16] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, Universally composable two-party and multi-party secure computation, in *STOC 2002* (ACM, 2002), pp. 494–503. The full version is available at http://eprint.iacr.org/2002/140

[17] I. Cascudo, I. Damgård, B.D. Nico Döttling, J.B. Nielsen, Rate-1, linear time and additively homomorphic UC commitments, in M. Robshaw, J. Katz, editors, *CRYPTO 2016, Part III*, *Lecture Notes in Computer Science*, vol. 9816 (Springer, Heidelberg, 2016), pp. 179–207. The full version is available at http://eprint.iacr.org/2016/137

[18] I. Cascudo, I. Damgård, B.M. David, I. Giacomelli, J.B. Nielsen, R. Trifiletti, Additively homomorphic UC commitments with optimal amortized overhead, in J. Katz, editor, *PKC 2015*, *Lecture Notes in Computer Science*, vol. 9020 (Springer, Heidelberg, 2015), pp. 495–515

[19] D. Cash, E. Kiltz, V. Shoup, The twin Diffie-Hellman problem and applications. *Smart*, **54**, 127–145
[20] R. Cramer, I. Damgård, B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols. *Desmedt*, **26**, 174–187
[21] R. Cramer, V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.***33**(1), 167–226, (2004) Early version in CRYPTO'98
[22] I. Damgård, M. Jurik, A generalisation, a simplification and some applications of Paillier's probabilistic public-key system, in K. Kim, editor, *PKC 2001*, *Lecture Notes in Computer Science*, vol. 1992 (Springer, Heidelberg, 2001), pp. 125–140
[23] I. Damgård, B.M. David, I. Giacomelli, J.B. Nielsen, Compact VSS and efficient homomorphic UC commitments, in P. Sarkar, T. Iwata, editors, *ASIACRYPT 2014 (2)*, *Lecture Notes in Computer Science*, vol. 8874 (Springer, Heidelberg, 2014), pp. 213–232
[24] I. Damgård, J. Groth, Non-interactive and reusable non-malleable commitment schemes, in *STOC 2003* (ACM, 2003), pp. 426–437
[25] I. Damgård, J.B. Nielsen, Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor, in M. Yung, editor, *CRYPTO 2002*, *Lecture Notes in Computer Science*, vol. 2442 (Springer, Heidelberg, 2002), pp. 581–596. The full version is available at http://www.brics.dk/RS/01/41/.
[26] Y.G. Desmedt, editor, in *Advances in Cryptology—CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21–25, 1994, Proceedings*, *Lecture Notes in Computer Science*, vol. 839 (Springer, Heidelberg, 1994)
[27] S. Fehr, D. Hofheinz, E. Kiltz, H. Wee, Encryption schemes secure against chosen-ciphertext selective opening attacks, in H. Gilbert, editor, *EUROCRYPT 2010*, *Lecture Notes in Computer Science*, vol. 6110 (Springer, Heidelberg, 2010), pp. 381–402
[28] M. Fischlin, B. Libert, M. Manulis, Non-interactive and re-usable universally composable string commitments with adaptive security, in D. Hoon Lee, X. Wang, editors, *ASIACRYPT 2011*, *Lecture Notes in Computer Science*, vol. 7073 (Springer, Heidelberg, 2011) pp. 468–485
[29] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, R. Trifiletti, On the complexity of additively homomorphic UC commitments, in E. Kushilevitz, T. Malkin, editors, *TCC 2016-A (1)*, *Lecture Notes in Computer Science*, vol. 9562 (Springer, Heidelberg, 2016), pp. 542–565
[30] Ei. Fujisaki, New constructions of efficient simulation-sound commitments using encryption and their applications, in O. Dunkelman, editor, *CT-RSA*, *Lecture Notes in Computer Science*, vol. 7178 (Springer, Heidelberg, 2012), pp. 136–155
[31] E. Fujisaki, Improving practical UC-secure commitments based on the DDH assumption, in V. Zirkas, R. De Prisco, editors, *Security and Cryptography for Networks—10th International Conference, SCN 2016, Amalfi, Italy, August 31–September 2, 2016. Proceedings*, *Lecture Notes in Computer Science*, vol. 9841 (Springer, Heidelberg, 2016), pp. 257–272
[32] J.A. Garay, Y. Ishai, R. Kumaresan, H. Wee, On the complexity of UC commitments, in P.Q. Nguyen, E. Oswald, editors, *EUROCRYPT 2014*, *Lecture Notes in Computer Science*, vol. 8441 (Springer, Heidelberg, 2014), pp. 677–694
[33] J.A. Garay, P.P. Mackenzie, K. Yang, Strengthening zero-knowledge protocols using signatures, in E. Biham, editor, *EUROCRYPT 2003*, *Lecture Notes in Computer Science*, volume 2656 (Springer, Heidelberg, 2003), pp. 177–194
[34] R. Gennaro, Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks, in M.K. Franklin, editor, *CRYPTO 2004*, *Lecture Notes in Computer Science*, vol. 3152 (Springer, Heidelberg, 2004), pp. 220–236. The full version available at Cryptology ePrint Archive http://eprint.iacr.org/2003/214
[35] J. Groth, A. Sahai, Efficient non-interactive proof systems for bilinear groups. *Smart*, **54**, 415–432
[36] B. Hemenway, R. Ostrovsky, Building lossy trapdoor functions from lossy encryption, in K. Sako, P. Sarkar, editors, *ASIACRYPT 2013 (2)*, *Lecture Notes in Computer Science*, volume 8270 (Springer, Heidelberg, 2013), pp. 241–260
[37] D. Hofheinz, All-but-many lossy trapdoor functions, in D. Pointcheval, T. Johansson, editors, *EUROCRYPT 2012*, *Lecture Notes in Computer Science*, vol. 7237 (Springer, Heidelberg, 2012), pp. 209–227. (last revised 18 Mar 2013 at http://eprint.iacr.org/2011/230)
[38] D. Hofheinz, J. Müller-Quade, Universally composable commitments using random oracles. *Naor*, **48**, 58–76

[39] Y. Ishai, J. Kilian, K. Nissim, E. Petrank, Extending oblivious transfers efficiently. *Boneh*, **9**, 145–161
[40] Y. Ishai, M. Prabhakaran, A. Sahai, Founding cryptography on oblivious transfer - efficiently. *Wagner*, **55**, 572–591
[41] T. Itoh, Y. Ohta, H. Shizuya, Language dependent secure bit commitment. *Desmedt*, **26**, 188–201
[42] A. Joux, editor. *Advances in Cryptology—EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26–30, 2009. Proceedings*, *Lecture Notes in Computer Science*, vol. 5479 (Springer, Heidelberg, 2009)
[43] J. Kilian, E. Petrank, An efficient noninteractive zero-knowledge proof system for NP with general assumptions, *JOC*, **11**(1), 1–27 (1998)
[44] E. Kiltz, Chosen-ciphertext security from tag-based encryption, in S. Halevi, T. Rabin, editors, *TCC 2006*, *Lecture Notes in Computer Science*, vol. 3876 (Springer, Heidelberg, 2006), pp. 581–600
[45] Y. Lindell, Highly-efficient universally-composable commitments based on the DDH assumption, in K.G. Paterson, editor, *EUROCRYPT 2011*, *Lecture Notes in Computer Science*, vol. 6632 (Springer, Heidelberg, 2011), pp. 446–466. The full version available at Cryptology ePrint Archive http://eprint.iacr.org/2011/180
[46] P. MacKenzie, M.K. Reiter, K. Yang, Alternatives to non-malleability: definitions, constructions, and applications (extended abstract), *Naor*, **48**, 71–190
[47] P. MacKenzie, K. Yang, On simulation-sound trapdoor commitments, in *Cachin and Camenisch*, **11**, pp. 382–400
[48] M. Naor, editor. *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, *Lecture Notes in Computer Science*, vol. 2951 (Springer, Heidelberg, 2004)
[49] R. Nishimaki, E. Fujisaki, K. Tanaka, An efficient non-interactive universally composable string-commitment scheme. *IEICE Trans.*, **95-A**(1), 167–175 (2012)
[50] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in J. Stern, editor, *EUROCRYPT '99*, *Lecture Notes in Computer Science*, vol. 1592 (Springer, Heidelberg, 1999), pp. 223–238
[51] C. Peikert, V. Vaikuntanathan, B. Waters, A framework for efficient and composable oblivious transfer, in *Wagner*, **55**, 554–571
[52] C. Peikert, B. Waters, Lossy trapdoor functions and their applications, in R.E. Ladner, C. Dwork, editors, *STOC 2008*, (ACM, 2008), pp. 187–196
[53] V. Shoup, A proposal for an ISO standard for public key encryption, Cryptology ePrint Archive, Report 2001/112, December 2001
[54] N.P. Smart, editor, *Advances in Cryptology—EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, *Lecture Notes in Computer Science*, vol. 4965 (Springer, Heidelberg, 2008)
[55] D. Wagner, editor, *Advances in Cryptology—CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2008. Proceedings*, *Lecture Notes in Computer Science*, vol. 5157 (Springer, Heidelberg, 2008)
[56] B. Waters, Efficient identity-based encryption without random oracles, in R. Cramer, editor, *EUROCRYPT 2005*, *Lecture Notes in Computer Science*, volume 3494 (Springer, Heidelberg, 2005), pp. 114–127