Improving Practical UC-Secure Commitments Based on the DDH Assumption

Eiichiro Fujisaki[(⊠)]

NTT Secure Platform Laboratories, Tokyo, Japan
`fujisaki.eiichiro@lab.ntt.co.jp`

Abstract. At Eurocrypt 2011, Lindell presented practical static and adaptively UC-secure commitment schemes based on the DDH assumption. Later, Blazy et al. (at ACNS 2013) improved the efficiency of the Lindell's commitment schemes. In this paper, we present static and adaptively UC-secure commitment schemes based on the same assumption and further improve the communication and computational complexity, as well as the size of the common reference string.

1 Introduction

Universal composability (UC) framework [5] 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. The UC framework allows one to divide the design of a large system into that of simpler sub-protocols, which provides the designer a fundamental benefit.

Commitment schemes are one of the most important tools in the cryptographic protocols. A commitment scheme consists of a two-phase protocol between two parties, a committer and a receiver. In the commit phase, a committer gives a receiver the digital equivalent of a *sealed envelope* containing value x. In the decommit 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. Commitment schemes that are secure in the UC framework were first presented by Canetti and Fischlin [6]. UC commitments are complete for constructing UC zero-knowledge protocols [6,13] and UC two-party and multiparty computation [7]. 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. Since UC commitments cannot be realized without an additional set-up assumption [6], the common reference string (CRS) model is widely used.

Several UC commitment schemes in the CRS model have been proposed so far. After [6], Canetti et al. [7] constructed inefficient schemes from general assumptions. Damgård and Nielsen [13] proposed interactive schemes that are

the first efficient UC-secure commitment schemes. Camenish and Shoup [4] also presented efficient interactive schemes. Although they are asymptotically efficient, their concrete instantiations are implemented on N^{d+1} modulus for RSA modulus N and $d \geq 1$, or $p^2 q$ modulus with primes, p and q.

In [24], Lindell presented the first practical UC commitment schemes based on an ordinary prime-order group. In practice, his constructions are much more efficient when implemented in elliptic curves whose security is equivalent to that of RSA modulus. He proposed two types of UC commitment schemes. One is *static* UC-secure and the other is *adaptively* UC-secure (with secure erasure). If an adversary should decide to corrupt parities only before a protocol starts, it is called *static* corruption. A corrupted party reveals its whole inner states to the adversary. A commitment scheme is called *static UC-secure* if it is UC-secure against static corruptions. On the other hand, if an adversary can decide to corrupt parties at any point in the executions of protocols, it is called *adaptive* corruption. A commitment scheme is called *adaptively UC-secure* if it is UC-secure against adaptive corruptions. Adaptive corruptions are more flexible and powerful attacks. Lindell's adaptively UC-secure commitment scheme assumes *secure erasure*, which means that parties can securely erase their unnecessary inner states that would have risks of their security at future corruptions. Lindell's static UC-secure commitment scheme has total communication complexity of 10 group elements plus 4 scalars, whereas his adaptively UC-secure one has that of 12 group elements plus 6 scalars. Shortly after, Fishlin et al. [15] transform Lindell's static UC-secure scheme into an non-interactive scheme adaptively UC-secure with erasure, by removing the interaction of the Sigma protocol using Groth-Sahai proofs [20]. Although their proposal is non-interactive, the communication and computational complexity is less efficient than [24], because it is implemented in symmetric bilinear groups and requires expensive pairing operations. We note that implementing it in asymmetric bilinear groups does not improve efficiency.

Blazy et al. [3] proposed the improvement of both Lindell's commitment schemes. Their static UC-secure commitment scheme has total communication complexity of 9 group elements plus 3 scalars. The commit phase is non-interactive and the decommit phase consists of 3 rounds (instead of 5 in Lindell's scheme). Their adaptively UC-secure commitment with secure erasure requires 10 group elements and 4 scalars. The commit phase has 3 rounds (instead of 5 in Lindell's scheme) and the decommit phase is non-interactive.

The static and adaptively UC-secure commitment schemes in [3,24] assume the DDH assumption and the existence of the collision resistant hash functions.

More on Related Works. The constructions of [12,17,26] are also asymptotically efficient. The constructions of [4,12,13,17,26] achieve adaptive UC-security without erasure in the CRS model. In [13], the CRS size grows linearly in the number of the parties. In [26], the CRS is one-time, i.e., one needs a new common-reference string for each execution of the commitment protocol. In the other works, the CRS is independent of the number of parties and re-usable. In addition, the work of [17] achieves non-interactiveness. The most efficient

constructions of [12,17,26] are implemented on N^{d+1} modulus for RSA modulus N, which are less efficient than [3,24].

Recently, [8,9,11,16,19] have proposed UC commitment schemes in the UC oblivious transfer (OT) hybrid model. Their constructions are very useful when a huge number of UC commitments are required. Their common significant property is that the schemes are very fast except for the overhead of UC OT protocols. In addition, one can make the number of the execution of UC commitments independent of the number of the execution of OT protocols. However, the proposals are only *static* UC-secure.

Therefore, [3,24] are still the most efficient *adaptively* UC-secure commitment schemes.

Note. Lindell's adaptively UC-secure commitment scheme [24] contains a small bug. Blazy et al. [3] clarified and fixed it. See [3,18] for more details.

1.1   Our Contribution

In this paper we further improve the efficiency of Blazy et al. static and adaptively UC-secure schemes [3]. By observing the security proof in [3], we realize that:

– In the adaptive case, two trapdoor commitments can be reduced to one.
– It is an overkill to use an IND-CCA secure public-key encryption (PKE) scheme in both static and adaptive cases.

The first claim comes from a simple observation. The second claim derives from our main technical contribution. We claim that an IND-PCA secure PKE scheme suffices for the protocols. Here the IND-PCA security notion is formulated by Abdala et al. [1] as a variant of the OW-PCA security notion [27]. The IND-PCA security notion is defined as indistinguishability of PKE in the presence of the *plaintext checkable oracle*, and a short version of Cramer-Shoup cryptosystem [10] satisfies this security notion.

In the concrete instantiation, we present practical static and adaptively UC-secure commitment schemes under the same assumption as in [3,24]. Our *adaptively* UC-secure commitment scheme (with erasure) is more efficient than Blazy et al. *static* UC-secure one. Our statistic and adaptive schemes both have the total communication complexity of 7 group elements and 3 scalars with the computational complexity of 18 exponentiations.

In Table 1, we compare our proposals with the previous works. All schemes below are UC-secure commitment schemes assuming the DDH assumption on cyclic group $\mathbb{G}$ and the existence of the collision resistant hash functions. All adaptively UC-secure ones below assume secure erasure. κ denotes the security parameter. Let q be the order of $\mathbb{G}$. Then, $\log(q) = O(\kappa)$. $|\mathbb{G}|$ denotes the length of the description of an element in $\mathbb{G}$, which depends on the concrete instantiation, but is generally slightly bigger than $\log(q)$. If it is implemented in an elliptic curve, it is at least $|\mathbb{G}| \geq \log(q) + 1$. $T^{\mathsf{exp}}(\mathbb{G})$ denotes the computational cost of one exponentiation on $\mathbb{G}$.

Table 1. Comparison among the UC commitments based on the DDH assumption

Schemes	Public parameter	Communication complexity	Computational complexity	Rounds com/decom	Security
Lin11 [24, Sect. 3]	$7|\mathbb{G}|$	$10|\mathbb{G}| + 4\kappa$	$27T^{\mathsf{exp}}(\mathbb{G})$	1/4	Static
Lin11 [24, Sect. 4]	$8|\mathbb{G}|$	$12|\mathbb{G}| + 6\kappa$	$36T^{\mathsf{exp}}(\mathbb{G})$	5/1	Adaptive
BCPV13 [3, Sect. 5.1]	$7|\mathbb{G}|$	$9|\mathbb{G}| + 3\kappa$	$22T^{\mathsf{exp}}(\mathbb{G})$	1/3	Static
BCPV13 [3, Sect. 5.3]	$7|\mathbb{G}|$	$10|\mathbb{G}| + 4\kappa$	$26T^{\mathsf{exp}}(\mathbb{G})$	3/1	Adaptive
Ours (Sect. 4.2)	$5|\mathbb{G}|$	$7|\mathbb{G}| + 3\kappa$	$18T^{\mathsf{exp}}(\mathbb{G})$	1/3	Static
Ours (Sect. 4.1)	$5|\mathbb{G}|$	$7|\mathbb{G}| + 3\kappa$	$18T^{\mathsf{exp}}(\mathbb{G})$	3/1	Adaptive

2   Preliminaries

2.1   (Tag-Based) Public-Key Encryption

We recall a tag-based public-key encryption (Tag-PKE) scheme (or a PKE scheme supported with labels), following [22,25,31]. A Tag-PKE $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ consists of the following three algorithms. The key-generation algorithm $\mathbf{K}$ is a PPT algorithm that takes 1^κ and outputs a pair of public and secret keys, (pk, sk). The encryption algorithm $\mathbf{E}$ is a PPT algorithm that takes public key pk, tag $t \in \{0,1\}^\kappa$ and message $m \in \mathsf{MSP}^{\mathsf{enc}}$, draws string r uniformly from the coin space $\mathsf{COIN}^{\mathsf{enc}}$, and produces ciphertext (t, c) where $c = \mathbf{E}_{pk}^t(m; r)$. The decryption algorithm $\mathbf{D}$ is a DPT algorithm that takes sk and a presumable ciphertext (t, c) where $c \in \{0,1\}^*$, and returns message $m = \mathbf{D}_{sk}^t(c)$. We require that for every sufficiently large $\kappa \in \mathbb{N}$, it always holds that $\mathbf{D}_{sk}^t(\mathbf{E}_{pk}^t(m)) = m$, for every (pk, sk) generated by $\mathbf{K}(1^\kappa)$ and every $m \in \mathsf{MSP}^{\mathsf{enc}}$. We say that ciphertext (t, c) is **proper** if there exists $(m, r) \in \mathsf{MSP}^{\mathsf{enc}} \times \mathsf{COIN}^{\mathsf{enc}}$ such that $c = \mathbf{E}_{pk}^t(m; r)$.

To suit actual instantiations, we assume $\mathsf{MSP}^{\mathsf{enc}}$ and $\mathsf{COIN}^{\mathsf{enc}}$ are defined by pk.

IND-CCA. We recall CCA security for Tag-PKEs [25], also called *weak* CCA security in [22]. We define the advantage of $A = (A_1, A_2)$ for Π against indistinguishability against chosen ciphertext attacks (IND-CCA) as

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

where experiment $\mathsf{Expt}_{\Pi,A}^{\mathsf{cca}\text{-}b}(\kappa)$ for $b \in \{0,1\}$ is defined in Fig. 1. The constraint of A in the experement is that A_2 is not allowed to submit $(t^*, \star)$ to $\mathbf{D}_{sk}(\cdot, \cdot)$ where t^* is the challenge tag. We say that Π is indistinguishable against chosen-ciphertext attacks (IND-CCA secure) if $\mathsf{Adv}_{\Pi,A}^{\mathsf{cca}}(\kappa) = \mathsf{negl}(\kappa)$ for every non-uniform PPT A.

We note that this security notion is weaker than the standard IND-CCA security notion [2,10,29] for PKE, because an adversary is not only prohibited from asking for the challenge ciphertext (t^*, c^*) but (t^*, c) with $c \neq c^*$.

IND-PCA. Recently, Abdalla et al. [1] proposed a security notion of indistinguishability against plaintext checkable attacks (IND-PCA) for PKE. This paper
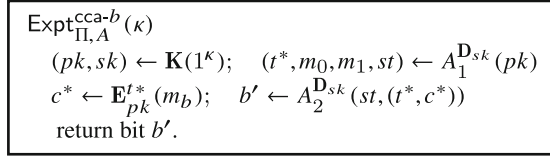
$$
\begin{array}{|l|}
\hline
\mathsf{Expt}^{\mathsf{cca}\text{-}b}_{\Pi,A}(\kappa) \\
\quad (pk, sk) \leftarrow \mathbf{K}(1^\kappa); \quad (t^*, m_0, m_1, st) \leftarrow A_1^{\mathbf{D}_{sk}}(pk) \\
\quad c^* \leftarrow \mathbf{E}^{t^*}_{pk}(m_b); \quad b' \leftarrow A_2^{\mathbf{D}_{sk}}(st, (t^*, c^*)) \\
\quad \text{return bit } b'. \\
\hline
\end{array}
$$

Fig. 1. Experiment of $\mathsf{Expt}^{\mathsf{cca}\text{-}b}_{\Pi,A}$

utilizes a Tag-PKE variant. Let $\mathsf{Expt}^{\mathsf{pca}\text{-}b}_{\Pi,A}(\kappa)$ for $b \in \{0,1\}$ be the experiment as in Fig. 2. Here oracle O^{pca}_{sk} takes (t, m, c) and returns 1 if and only if c is a proper ciphertext of m on tag t. The constraint of A in the experiment is that A is not allowed to submit $(t^*, \star, \star)$ to $O^{\mathsf{pca}}_{sk}(\cdot, \cdot, \cdot)$ where t^* is the challenge tag. We define the advantage of A for Π against indistinguishability against the plaintext checkable attacks (IND-PCA) as

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

We say that Π is indistinguishable against the plaintext checkable attacks (IND-PCA secure) if $\mathsf{Adv}^{\mathsf{pca}}_{\Pi,A}(\kappa) = \mathsf{negl}(\kappa)$ for every non-uniform PPT A.

$$
\begin{array}{|l|}
\hline
\mathsf{Expt}^{\mathsf{pca}\text{-}b}_{\Pi,A}(\kappa) \\
\quad (pk, sk) \leftarrow \mathbf{K}(1^\kappa); \quad (t^*, m_0, m_1, st) \leftarrow A_1^{O^{\mathsf{pca}}_{sk}}(pk) \\
\quad c^* \leftarrow \mathbf{E}^{t^*}_{pk}(m_b); \quad b' \leftarrow A_2^{O^{\mathsf{pca}}_{sk}}(st, (t^*, c^*)) \\
\quad \text{return bit } b'. \\
\hline
\end{array}
$$
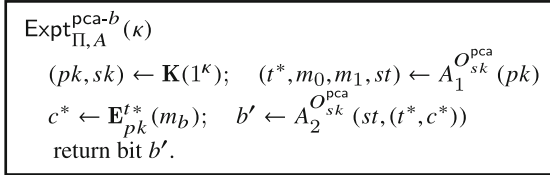
Fig. 2. Experiment of $\mathsf{Expt}^{\mathsf{pca}\text{-}b}_{\Pi,A}$

2.2 Trapdoor Commitments

We define a trapdoor commitment scheme. Let $\mathsf{TCOM} = (\mathsf{Gen}^{\mathsf{tc}}, \mathsf{Com}^{\mathsf{tc}}, \mathsf{TCom}^{\mathsf{tc}}, \mathsf{TCol}^{\mathsf{tc}})$ be a tuple of the following four algorithms. $\mathsf{Gen}^{\mathsf{tc}}$ is a PPT algorithm takes as input security parameter κ and outputs a pair of public and trap-door keys (pk, tk). $\mathsf{Com}^{\mathsf{tc}}$ is a PPT algorithm takes as input pk and message $x \in \{0,1\}^{\lambda_m}$ committed to, chooses $r \leftarrow \mathsf{COIN}^{\mathsf{com}}$, and outputs a $\psi = \mathsf{Com}^{\mathsf{tc}}_{pk}(m; r)$. $\mathsf{TCom}^{\mathsf{tc}}$ is a PPT algorithm takes as input tk and outputs $(\psi, \chi) \leftarrow \mathsf{TCom}^{\mathsf{tc}}_{tk}(1^\kappa)$. $\mathsf{TCol}^{\mathsf{tc}}$ is a DPT algorithm that takes $(tk, \psi, \chi, \hat{x})$ where $\hat{x} \in \{0,1\}^{\lambda_m}$ and outputs $\hat{r} \in \mathsf{COIN}^{\mathsf{com}}$ such that $\psi = \mathsf{Com}^{\mathsf{tc}}_{pk}(\hat{x}; \hat{r})$.

We call TCOM is a trapdoor commitment scheme if the following two conditions hold.

Trapdoor Collision. For all pk generated by $\mathsf{Gen}^{\mathsf{tc}}(1^\kappa)$, and all $x \in \{0,1\}^{\lambda_m(\kappa)}$, the following ensembles are statistically indistinguishable in κ:

$$\left\{ (\psi, x, r) \mid r \leftarrow \mathsf{COIN}^{\mathsf{com}}; \psi = \mathsf{Com}^{\mathsf{tc}}_{pk}(x; r) \right\}_{\kappa \in \mathbb{N}, pk \in \mathsf{Gen}^{\mathsf{tc}}(1^\kappa), x \in \{0,1\}^{\lambda_m}}$$

$$\overset{\mathsf{s}}{\approx} \left\{ (\psi, x, r) \mid (\psi, \chi) \leftarrow \mathsf{TCom}^{\mathsf{tc}}_{tk}(1^\kappa); r = \mathsf{TCol}^{\mathsf{tc}}_{tk}(\psi, \chi, x) \right\}_{\kappa \in \mathbb{N}, pk \in \mathsf{Gen}^{\mathsf{tc}}(1^\kappa), x \in \{0,1\}^{\lambda_m}}.$$

Computational Binding. For all non-uniform PPT adversary A,

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

2.3 Sigma Protocol

Let L be an NP language and R_L be the relation derived from L. Let $\Sigma = (\mathsf{P}^{\mathsf{com}}_\Sigma, \mathsf{P}^{\mathsf{ans}}_\Sigma, \mathsf{V}^{\mathsf{vrfy}}_\Sigma, \mathsf{simP}^{\mathsf{com}}_\Sigma)$ be a tuple of algorithms (associated with L) as follows:

- $\mathsf{P}^{\mathsf{com}}_\Sigma$ is a PPT algorithm that takes $(x, w) \in R_L$ and outputs $(\alpha, \xi) \leftarrow \mathsf{P}^{\mathsf{com}}_\Sigma(x, w)$. For simplicity, we assume that ξ is inner coins of $\mathsf{P}^{\mathsf{com}}_\Sigma$.
- $\mathsf{P}^{\mathsf{ans}}_\Sigma$ is a DPT algorithm that takes (x, w, ξ, β) and outputs $\gamma = \mathsf{P}^{\mathsf{ans}}_\Sigma(x, w, \xi, \beta)$ where $\beta \in \{0,1\}^{\lambda_{\mathsf{ch}}}$.
- $\mathsf{V}^{\mathsf{vrfy}}_\Sigma$ is a DPT algorithm that accepts or rejects $(x, \alpha, \beta, \gamma)$.
- $\mathsf{simP}^{\mathsf{com}}_\Sigma$ is a PPT algorithm that takes (x, β) and outputs $(\alpha, \beta, \gamma) \leftarrow \mathsf{simP}^{\mathsf{com}}_\Sigma(x, \beta)$.

Σ is called a Sigma protocol if it satisfies the following requirements:

Completeness: For every $(x, w) \in R_L$, every $(\alpha, \xi) \in \mathsf{P}^{\mathsf{com}}_\Sigma(x, w)$, and every $\beta \in \{0,1\}^{\lambda_{\mathsf{ch}}}$, it always holds that $\mathsf{V}^{\mathsf{vrfy}}_\Sigma(x, \alpha, \beta, \gamma) = 1$ where $\gamma = \mathsf{P}^{\mathsf{ans}}_\Sigma(x, w, \xi, \beta)$.

Special Soundness: If there are two different accepting conversations for the same α on x, i.e., (α, β, γ) and $(\alpha, \beta', \gamma')$, with $\beta \neq \beta'$, it must hold that $x \in L$ and there is an efficient extractor that takes (α, β, γ) and $(\alpha, \beta', \gamma')$ as input and outputs w such that $(x, w) \in R_L$. We call such a pair a *collision* on x. Special soundness implies that there is at most one e such that $\mathsf{V}^{\mathsf{vrfy}}_\Sigma(x, \alpha, \beta, \gamma) = 1$ for every $x \notin L$ and every α.

Honest-Verifier Statistical Zero-Knowledgeness (HVSZK): For all $(x, w) \in R_L$, and all $\beta \in \{0,1\}^{\lambda_{\mathsf{ch}}}$, the following ensembles are statistically indistinguishable in κ:

$$\{\mathsf{simP}^{\mathsf{com}}_\Sigma(x, \beta; r_\gamma)\}_{\kappa \in \mathbb{N}, (x,w) \in R_L, \beta \in \{0,1\}^{\lambda_{\mathsf{ch}}}}$$

$$\overset{\mathsf{s}}{\approx} \{(\mathsf{P}^{\mathsf{com}}_\Sigma(x, w; \xi)_1, \beta, \mathsf{P}^{\mathsf{ans}}_\Sigma(x, w, \xi, \beta))\}_{\kappa \in \mathbb{N}, (x,w) \in R_L, \beta \in \{0,1\}^{\lambda_{\mathsf{ch}}}},$$

where $\mathsf{P}^{\mathsf{com}}_\Sigma(x, w)_1$ denotes the first output of $\mathsf{P}^{\mathsf{com}}_\Sigma(x, w)$. Here the probability of the left-hand side is taken over random variable r_γ and the right-hand side is taken over random variable ξ.

3   Universal Composable Framework

The UC framework defines a non-uniform PPT environment machine $\mathcal{Z}$ that oversees the execution of a protocol in one of two worlds. In both worlds, there are an PPT adversary and honest parties (some of which may be corrupted by the adversary). In the *real world*, the real protocol is run among the parties with some possible attacks given by the real-world adversary. In the *ideal world*, there additionally exists a trusted uncorrupted party, *ideal functionality* $\mathcal{F}$, where the honest parties in the ideal world do not interact with each other and instead send their inputs to the ideal functionality $\mathcal{F}$, which carries out the computation of the protocol in the trusted manner and sends back to the outputs to each party. We say that protocol π UC-realizes ideal functionality $\mathcal{F}$ if there exists an ideal-world adversary (simulator) $\mathcal{S}$ such 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-world adversary (simulator) $\mathcal{S}$.

In both worlds, the environment adaptively chooses the inputs for the honest parties and receives the outputs that they get. The environment can control the adversary and order it to corrupt any honest party at the beginning of the execution of the protocol (**static corruption**) or at any timing during the execution of the protocol (**adaptive corruption**). When a honest party is corrupted, the adversary may read the inner state of the honest party and fully control it. In the ideal world, after a party is corrupted, the ideal-world adversary $\mathcal{S}$ may access to the ideal functionality as the party does. The environment can see the *inside* of the execution of the protocol – the actual interactions between the honest parties or between the honest parties and the adversary – via the adversary's view. Since there is no interaction between the honest parties or between the honest parties and the adversary in the ideal world, the ideal-world simulator has to simulate the real-world adversary's view as it comes from the inside of the protocol in the real world.

We consider a model with ideal authentication channels, and so the adversary is allowed to read the messages sent by uncorrupted honest party but cannot modify them. Our protocols are executed in the common reference string (CRS) model. This means that the protocol 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. Our adaptively UC-secure protocol requires the **secure erasure** assumption that the honest parties can securely erase their unnecessary inner states, as with [3,24].

We denote by $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(\kappa, z)$ 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 only consider black-box simulators $\mathcal{S}$ and denote the simulator by $\mathcal{S}^{\mathcal{A}}$, which means that it works with the adversary $\mathcal{A}$ attacking the real protocol. We denote by $\mathrm{HYBRID}_{\pi,\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{crs}}}(\kappa, z)$ the output of the environment $\mathcal{Z}$ with input z after an execution of the protocol π in the $\mathcal{F}_{\mathsf{crs}}$ hybrid model (or in the real world in the CRS model). 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

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

The importance of the universal composability framework is that it satisfies a composition theorem that states that any protocol that is universally composable is secure when it runs concurrently with many other arbitrary protocols. For more details, see [5].

We consider UC commitment schemes that can be used repeatedly under a single common reference string (**re-usable common reference string**). The multi-commitment ideal functionality $\mathcal{F}_{\text{MCOM}}$ from [7] is the ideal functionality of such commitments. We formally provide it in Fig. 3.
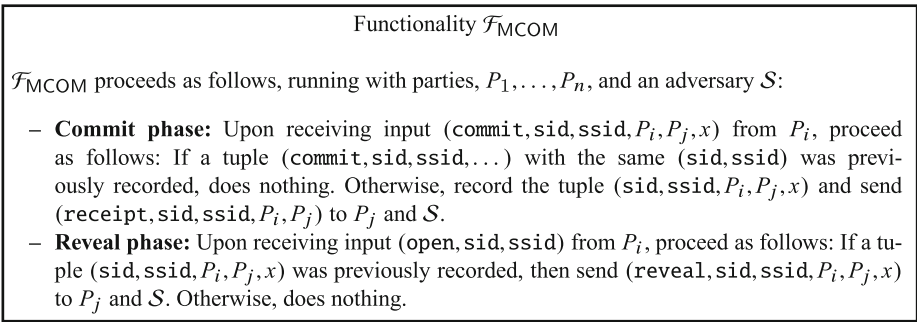

Functionality $\mathcal{F}_{\text{MCOM}}$

$\mathcal{F}_{\text{MCOM}}$ 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. 3. The ideal multi-commitment functionality

4 Our Proposal

For the space limitation, we focus on the adaptively UC-secure case. The static case is just a simplified version of the adaptive case and hence the proof is omitted to avoid a redundant exposition.

4.1 Our Adaptively UC-Secure Commitment with Erasure

We start by explaining the basic idea of Lindell's scheme [24]. As mentioned before, UC commitments require *extractability* and *equivocability*. Therefore, it is natural to use a PKE scheme as an extractable commitment scheme in the CRS model, where the committer commits to a secret value by encrypting it using public-key *pk* put in the common reference string. In the simulation, the simulator can choose the public-key along with the corresponding secret-key and use it by extracting the committed value. However, UC commitments should be equivocable at the same time. So, it is not possible at the decommit phase to simply reveal the committed value and the randomness used to encrypt,

because encryptions are perfectly binding. Therefore, the committer instead sends the committed value m and makes *a concurrent (straight-line) non-malleable zero-knowledge proof* such that CT is a proper ciphertext of m. The straight-line zero-knowledge simulation is needed, because in the UC setting, the rewinding simulation is not allowed. In addition, concurrent non-malleability is needed because the simulator makes a number of *fake* proofs (i.e., valid (simulated) proofs on false statements), but ensures that the adversary cannot produce any fake proof even after it sees many fake ones. To do so, Lindell utilized a dual mode encryption scheme, an IND-CCA secure PKE scheme, and a Sigma protocol. To make the scheme secure against the adaptive corruptions, he additionally used a trapdoor commitment scheme. It enables the committer to switch the order of messages in the proof and to run most of the proof in the commit phase. Then, the committer can erase the randomness used to encrypt before sending ciphertext CT, which makes the scheme adaptively UC-secure with erasure. Blazy et al. [3] showed that the dual mode encryption can be removed from the proofs in both static and adaptive cases. By this observation, they improved the number of the rounds from five to three at the commit phase in the adaptive case (resp. from four to three at the decommit phase in the static case). See Table 1.

Our starting point is the BCPV adaptively UC-secure commitment scheme. Before exposing the difference, we give the description of our adaptively UC-secure commitment scheme.

The Adaptively UC-Secure Commitment Scheme. Let $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ be a tag-based PKE scheme. Let $\Sigma = (\mathsf{P}^{\mathsf{com}}_{\Sigma}, \mathsf{P}^{\mathsf{ans}}_{\Sigma}, \mathsf{V}^{\mathsf{vrfy}}_{\Sigma}, \mathsf{simP}^{\mathsf{com}}_{\Sigma})$ be a Sigma protocol on a language such that

$$L = \{(\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT}) \mid \exists w \in \mathsf{COIN}^{\mathsf{enc}} \text{ s.t. } \mathsf{CT} = \mathbf{E}^{t}_{\mathsf{pk}^{\mathsf{enc}}}(m; w)\}.$$

Let $\mathsf{TCOM} = (\mathsf{Gen}^{\mathsf{tc}}, \mathsf{Com}^{\mathsf{tc}}, \mathsf{TCom}^{\mathsf{tc}}, \mathsf{TCol}^{\mathsf{tc}})$ be a trap-door commitment scheme. Our adaptively UC-secure commitment scheme is constructed as follows.

Common Reference String. The trusted party computes $(\mathsf{pk}^{\mathsf{enc}}, \mathsf{sk}^{\mathsf{enc}}) \leftarrow \mathbf{K}(1^{\kappa})$ and $(\mathsf{pk}^{\mathsf{tc}}, \mathsf{tk}^{\mathsf{tc}}) \leftarrow \mathsf{Gen}^{\mathsf{tc}}(1^{\kappa})$. It chooses a collision-resistant hash $H \leftarrow \mathbb{H}$ such that $H : \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda_m}$ and sets $\mathsf{crs} = (\mathsf{pk}^{\mathsf{enc}}, \mathsf{pk}^{\mathsf{tc}}, H)$.

The Commit Protocol.

1. Upon receiving $(\texttt{commit}, \texttt{sid}, \texttt{ssid}, C, R, m)$ where $m \in \mathsf{MSP}_{\mathsf{pk}^{\mathsf{enc}}}$, committer C sets $t = (\texttt{sid}, \texttt{ssid}, C, R)$, chooses random $w \leftarrow \mathsf{COIN}_{\mathsf{pk}^{\mathsf{enc}}}$, and computes $\mathsf{CT} = \mathbf{E}_{\mathsf{pk}^{\mathsf{enc}}}(t, m; w)$.
2. Let $L = \{(\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT}) \mid \exists w \in \mathsf{COIN}^{\mathsf{enc}} \text{ s.t. } \mathsf{CT} = \mathbf{E}_{\mathsf{pk}^{\mathsf{enc}}}(t, m; w)\}$. C computes $(\alpha, \xi) \leftarrow \mathsf{P}^{\mathsf{com}}_{\Sigma}(x, w)$ as the first message of Sigma protocol on $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$.
3. C computes $\phi = H(t, x, \alpha)$ where $t = (\texttt{sid}, \texttt{ssid}, C, R)$.
4. C chooses random $r_{\mathsf{tc}} \leftarrow \mathsf{COIN}^{\mathsf{com}}$ and computes $\psi = \mathsf{Com}^{\mathsf{tc}}_{\mathsf{pk}^{\mathsf{tc}}}(\phi; r_{\mathsf{tc}})$.
5. C sends (t, ψ) to receiver R.

6. Receiver R checks $t = (\mathtt{sid}, \mathtt{ssid}, C, R)$. If there is nothing wrong, then it sends back $\beta \leftarrow \{0,1\}^{\lambda_{\mathsf{ch}}}$.
7. C computes $\gamma = \mathsf{P}_{\Sigma}^{\mathsf{ans}}(x, w, \xi, \beta)$.
8. C erases (w, ξ).
9. C sends CT to R.
10. R stores $(t, \mathsf{CT}, \psi, \beta)$ and outputs $(\mathtt{receipt}, \mathtt{sid}, \mathtt{ssid}, C, R)$.

The Decommit Protocol.

1. Upon receiving $(\mathtt{open}, \mathtt{sid}, \mathtt{ssid})$, committer C sends $(t, m, \alpha, \gamma, r_{\mathsf{tc}})$ to receiver R where $t = (\mathtt{sid}, \mathtt{ssid}, C, R)$.
2. R computes $\phi = H(t, x, \alpha)$, where $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$, and verifies $\psi = \mathsf{Com}_{\mathsf{pk}^{\mathsf{tc}}}^{\mathsf{tc}}(\phi; r_{\mathsf{tc}})$ and $\mathsf{V}_{\Sigma}^{\mathsf{vrfy}}(x, (\alpha, \beta, \gamma)) = 1$. If all relations hold, R accepts and outputs $(\mathtt{reveal}, \mathtt{sid}, \mathtt{ssid}, C, R, m)$.

Protocol Idea. The difference of our scheme from the BCPV scheme is the following two: Our scheme commits to ciphertext CT and the first message of the Sigma prorocol, denoted α, in the same *sealed envelope* ψ, whereas the BCPV scheme commits to CT and α in the distinct envelopes, ψ_1 and ψ_2, respectively. However, the committer can simply reveal CT (without any witness) at the commit phase and postpone to show the *witness* that ψ_1 really contains CT until the decommit phase. So, the two envelops can be unified. This is because in the ideal world, the value $\tilde{m}$ extracted by the simulator at the commit phase is revealed to the environment only when the corrupted committer (controlled by the adversary) successfully executes the decommit phase.

The second improvement comes from realizing that IND-PCA secure PKE [1] suffices, instead of IND-CCA secure PKE. We note that a simplified variant of Cramer-Shoup scheme, the Short Cramer-Shoup (SCS) scheme [1], is IND-PCA secure. The ciphertext of the SCS scheme consists of three group elements, instead of four. Hence, the first message of the Sigma protocol is also reduced to three group elements (instead of four).

We informally explain the reason that IND-PCA security suffices. In the ideal world, the simulator simulates an honest committer without knowing the committed *value* at the commit phase. In addition, when interacting with a corrupted committer as an honest receiver, the simulator must extract the committed value m' that the corrupted committer has committed to before the decommit phase. The extracted value $\tilde{m}'$ is revealed to the environment when the corrupted committer successfully executes the decommit phase. Therefore, if the extracted value is different from the value opened by the corrupted committer, the environment can distinguish the real world from the ideal world. By construction, at the decomit phase, a committer opens the committed value m' with the proof that CT is a proper ciphertext of m'. If it is a real proof, $\tilde{m}' = m'$ always holds. As long as the adversary only see the real proofs produced by the honest committer (or the simulator), the corrupted committer (controlled by the adversary) cannot make a fake proof (i.e., a "valid" proof on a false statement), because of the binding property of TCOM and the soundness property of the Sigma protocol. Hence, the valid proofs produced by the corrupted committer

should be real. Thus, the extracted value $\tilde{m}'$ should be the same as the opened value m'. This corresponds to Game 1. In Game 2, the simulator simulates the honest committer, by producing the simulated proofs on the *true* statements *that* $\mathsf{CT} = \mathbf{E}_{pk}(m)$ *is a proper ciphertext of m.* Still, the adversary cannot make a fake proof. This comes from the trapdoor collision property of TCOM and the HVSZK property of the Sigma protocol. Indeed, the simulated proofs on the true statements are statistically indistinguishable from the real proofs. In the next game, the simulator finally makes *fake* proofs when simulating the honest committer, i.e., simulated proofs on the *false* statements *that* $\mathsf{CT} = \mathbf{E}_{pk}(0)$ *is a proper ciphertext of m.* Here, to prove the environment's view is indistinguishable from that in the former game, the works of [3,24] rely on the power of IND-CCA secure PKE. However, *it is an overkill.* In Game 2, we know that the adversary cannot make a fake proof. Hence, if it can make a fake proof, it means that we are playing the latter game. To realize in which game we are playing, *we need the power of the PCA oracle.* We can then construct an IND-PCA adversary A whose advantage can be reduced to the probability of distinguishing these two games. If the adversary makes a fake proof, then A can see, with the power of the PCA oracle, that it is playing in the latter game. Then, it can halt and make a precise decision. If the adversary does not make fake proofs, then A can perfectly simulate either of two games according to which message, $\mathbf{E}_{pk}(m)$ or $\mathbf{E}_{pk}(0)$, is encrypted. We let A output the output of the environment. Then, if the difference of the environment's output in the two games is significant, the advantage of A in the IND-PCA game is also significant, which contradicts IND-PCA security.

We now state the main theorem.

Theorem 1. *Let Π be IND-PCA. Then, the above construction UC-securely realizes the $\mathcal{F}_{\mathsf{MCOM}}$ functionality in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model against the adaptive corruptions with secure erasure.*

Due to the space limitation, we provide the formal proof in the extended version [18].

4.2 Our Static UC-Secure Commitment

Our static UC-secure commitment scheme is constructed as follows.

Common Reference String. The trusted party computes $(\mathsf{pk}^{\mathsf{enc}}, \mathsf{sk}^{\mathsf{enc}}) \leftarrow \mathbf{K}(1^\kappa)$ and $(\mathsf{pk}^{\mathsf{tc}}, \mathsf{tk}^{\mathsf{tc}}) \leftarrow \mathsf{Gen}^{\mathsf{tc}}(1^\kappa)$. It chooses a collision-resistant hash $H \leftarrow \mathbb{H}$ such that $H : \{0,1\}^* \rightarrow \{0,1\}^{\lambda_m}$ and sets $\mathsf{crs} = (\mathsf{pk}^{\mathsf{enc}}, \mathsf{pk}^{\mathsf{tc}}, H)$.

The Commit Protocol.

1. Upon receiving $(\mathtt{commit}, \mathsf{sid}, \mathsf{ssid}, C, R, m)$ where $m \in \mathsf{MSP}_{\mathsf{pk}^{\mathsf{enc}}}$, committer C sets $t = (\mathsf{sid}, \mathsf{ssid}, C, R)$, chooses random $w \leftarrow \mathsf{COIN}_{\mathsf{pk}^{\mathsf{enc}}}$, and computes $\mathsf{CT} = \mathbf{E}_{\mathsf{pk}^{\mathsf{enc}}}(t, m; w)$.
2. C sends (t, CT) to receiver R.
3. R stores (t, CT) and outputs $(\mathtt{receipt}, t)$.

The Decommit Protocol.

1. Upon receiving $(\texttt{open}, \texttt{sid}, \texttt{ssid})$, committer C sets $t = (\texttt{sid}, \texttt{ssid}, C, R)$, and computes $(\alpha, \xi) \leftarrow \mathsf{P}_\Sigma^{\mathsf{com}}(x, w)$ as the first message of Sigma protocol on $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$ for $L = \{(\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT}) \mid \exists w \in \mathsf{COIN}^{\mathsf{enc}} \text{ s.t. } \mathsf{CT} = \mathbf{E}_{\mathsf{pk}^{\mathsf{enc}}}(t, m; w)\}$.
2. C computes $\phi = H(t, x, \alpha)$ where $t = (\texttt{sid}, \texttt{ssid}, C, R)$.
3. C chooses random $r_{\mathsf{tc}} \leftarrow \mathsf{COIN}^{\mathsf{com}}$ and computes $\psi = \mathsf{Com}_{\mathsf{pk}^{\mathsf{tc}}}^{\mathsf{tc}}(\phi; r_{\mathsf{tc}})$.
4. C sends (t, ψ) to receiver R.
5. Receiver R checks $t = (\texttt{sid}, \texttt{ssid}, C, R)$. If there is nothing wrong, then it sends back $\beta \leftarrow \{0, 1\}^{\lambda_{\mathsf{ch}}}$.
6. C computes $\gamma = \mathsf{P}_\Sigma^{\mathsf{ans}}(x, w, \xi, \beta)$.
7. Committer C sends $(t, m, \alpha, \gamma, r_{\mathsf{tc}})$ to receiver R where $t = (\texttt{sid}, \texttt{ssid}, C, R)$.
8. R computes $\phi = H(t, x, \alpha)$, where $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$, and verifies $\psi = \mathsf{Com}_{\mathsf{pk}^{\mathsf{tc}}}^{\mathsf{tc}}(\phi; r_{\mathsf{tc}})$ and $\mathsf{V}_\Sigma^{\mathsf{vrfy}}(x, (\alpha, \beta, \gamma)) = 1$. If all relations hold, R accepts and outputs $(\texttt{reveal}, \texttt{sid}, \texttt{ssid}, C, R, m)$.

Theorem 2. *Let* PKE *be IND-PCA. Then, the above construction UC-realizes the* $\mathcal{F}_{\mathsf{MCOM}}$ *functionality in the* $\mathcal{F}_{\mathsf{CRS}}$*-hybrid model against the static corruptions.*

The proof is omitted due to the similarity of the proof of Theorem 1.

4.3 Actual Instantiations

In the above constructions, we use the following building blocks.

The Short Cramer-Shoup (Tag-PKE) Scheme $\Pi^{\mathsf{pca}} = (\mathbf{K}, \mathbf{E}, \mathbf{D})$. This is a Tag-PKE variant of the short version of Cramer-Shoup (SCS) cryptosystem introduced in [1].

- $\mathbf{K}(1^\kappa, (\mathbb{G}, q))$: It picks up hash function $H' : \{0, 1\}^* \rightarrow \mathbb{Z}/q\mathbb{Z}$ and a random generator g in $\mathbb{G}$. It picks up independent random elements $x_e, x_1, x_2, y_1, y_2 \leftarrow \mathbb{Z}/q\mathbb{Z}$ and computes $h = g^{x_e}$, $c = g^{x_1} h^{x_2}$, and $d = g^{y_1} h^{y_2}$. It finally outputs $(\mathsf{pk}^{\mathsf{enc}}, \mathsf{sk}^{\mathsf{enc}}) = ((\mathbb{G}, q, H', g, h, c, d), (x_e, x_1, x_2, y_1, y_2))$.
- $\mathbf{E}_{\mathsf{pk}^{\mathsf{enc}}}(t, m)$: To encrypt $m \in \mathbb{G}$ on tag $t \in \{0, 1\}^\kappa$, it picks up random $w \leftarrow \mathsf{COIN}^{\mathsf{enc}}$, sets $\tau = H'(t, g^w)$, and outputs $\mathsf{CT} = (g^w, mh^w, (c^\tau d)^w)$.
- $\mathbf{D}_{\mathsf{sk}^{\mathsf{enc}}}(t, \mathsf{CT})$: It first parses $\mathsf{CT} = (C_1, C_2, C_3)$ and computes $m = C_2 C_1^{-x_e}$. It aborts if $C_3 = C_1^{\tau x_1 + y_1} (C_2/m)^{\tau x_2 + y_2}$ where $\tau = H(t, C_1)$; otherwise, it outputs m.

The SCS cryptosystem is proven (in [1]) IND-PCA secure if the DDH assumption holds and H' is a collision-resistant hash. The proof that the SCS Tag-PKE scheme is (the tag version of) IND-PCA secure defined in Sect. 2 is straightforward from the original proof in [1].

Pedersen Commitment $\mathsf{TCOM} = (\mathsf{Gen}^{\mathsf{tc}}, \mathsf{Com}^{\mathsf{tc}}, \mathsf{TCom}^{\mathsf{tc}}, \mathsf{TCol}^{\mathsf{tc}})$. The following is the description of Pedersen commitment scheme [28].

- $\mathsf{Gen}^{\mathsf{tc}}(1^\kappa, (\mathbb{G}, q, g))$: It picks up random $x_{\mathsf{tc}} \leftarrow \mathbb{Z}/q\mathbb{Z}$ and computes $\hat{h} = g^{x_{\mathsf{tc}}}$. It outputs $\mathsf{pk}^{\mathsf{tc}} = (\mathbb{G}, q, g, \hat{h})$ and $\mathsf{tk}^{\mathsf{tc}} = (\mathsf{pk}^{\mathsf{tc}}, x_{\mathsf{tc}})$.

- $\mathsf{Com}^{\mathsf{tc}}_{\mathsf{pk}^{\mathsf{tc}}}(\phi)$: To commit to $\phi \in \{0,1\}^{\lambda_m}$, it picks up random $r_{\mathsf{tc}} \leftarrow \mathbb{Z}/q\mathbb{Z}$ and outputs $\psi = g^{r_{\mathsf{tc}}} \hat{h}^{\phi}$.
- $\mathsf{TCom}^{\mathsf{tc}}_{\mathsf{tk}^{\mathsf{tc}}}(1^{\kappa})$: It picks up random $\xi \leftarrow \mathbb{Z}/q\mathbb{Z}$ and outputs $\psi = g^{\xi}$.
- $\mathsf{TCol}^{\mathsf{tc}}_{\mathsf{tk}^{\mathsf{tc}}}(\xi, \hat{\phi})$: To open ψ to $\hat{\phi} \in \{0,1\}^{\lambda_m}$, it outputs $\hat{r_{\mathsf{tc}}} = \xi - \hat{\phi} \cdot x_{\mathsf{tc}} \bmod q$. One can note that $\psi = g^{\hat{r_{\mathsf{tc}}}} \hat{h}^{\hat{\phi}}$.

The Pedersen commitment scheme holds the trapdoor collision property unconditionally and the computational binding property under the discrete log (DL) assumption on $\mathbb{G}$.

The Sigma Protocol on the Language Derived from the SCS Tag-PKE Scheme. Let

$$L^{\mathsf{enc}} = \{(\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT}) \,|\, \exists w \text{ s.t. } C_1 = g^w, \, C_2 = m \cdot h^w, \text{ and } C_3 = (c^{\tau} d)^w\},$$

where $\tau = H'(t, C_1)$. Sigma protocol $\Sigma = (\mathsf{P}^{\mathsf{com}}_{\Sigma}, \mathsf{P}^{\mathsf{ans}}_{\Sigma}, \mathsf{V}^{\mathsf{vrfy}}_{\Sigma}, \mathsf{simP}^{\mathsf{com}}_{\Sigma})$ on L^{enc} is described as follows.

- $(\alpha, \xi) \leftarrow \mathsf{P}^{\mathsf{com}}_{\Sigma}(x, w)$, where $x = (\mathsf{pk}^{\mathsf{enc}}, m, \tau, \mathsf{CT})$ and $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ such that $\xi \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\alpha_1 = g^{\xi}$; $\alpha_2 = h^{\xi}$; and $\alpha_3 = (c^{\tau} d)^{\xi}$.
- $\gamma \leftarrow \mathsf{P}^{\mathsf{ans}}_{\Sigma}(x, w, \xi, \beta)$, where $\beta \in \{0,1\}^{\lambda_{\mathsf{ch}}}$ and $\gamma = \xi - \beta w \bmod q$.
- $\mathsf{V}^{\mathsf{vrfy}}_{\Sigma}(x, (\alpha, \beta, \gamma)) = 1$ if and only if it holds that $\alpha_1 = g^{\gamma} C_1^{\beta}$, $\alpha_2 = h^{\gamma}(C_2/m)^{\beta}$, and $\alpha_3 = (c^{\tau} d)^{\gamma} C_3^{\beta}$.
- $(\alpha, \beta, \gamma) \leftarrow \mathsf{simP}^{\mathsf{com}}_{\Sigma}(x, \beta)$, where $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ such that $\gamma \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\alpha_1 = g^{\gamma} C_1^{\beta}$; $\alpha_2 = h^{\gamma}(C_2/m)^{\beta}$; $\alpha_3 = (c^{\tau} d)^{\gamma} C_3^{\beta}$.

Applied to Our Adaptively UC-Secure Commitment Scheme.

- Common Reference String: $\mathsf{crs} = (\mathbb{G}, q, H, H', g, h, c, d, \hat{h})$.
- The Commit phase:
  - Communication: $(\psi, \beta, \mathsf{CT}) \in \mathbb{G} \times \{0,1\}^{\lambda_{\mathsf{ch}}} \times \mathbb{G}^3$.
  - Committer's Computation: $w \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\mathsf{CT} = (C_1, C_2, C_3) = (g^w, m \cdot h^w, (c^{\tau} d)^w)$ with $\tau = H'(t, C_1)$ for $t = (\mathsf{sid}, \mathsf{ssid}, C, R)$; $\xi \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\alpha = (\alpha_1, \alpha_2, \alpha_3) = (g^{\xi}, h^{\xi}, (c^{\tau} d)^{\xi})$; $\gamma = \xi - \beta w \bmod q$; $r_{\mathsf{tc}} \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\psi = g^{\phi} \hat{h}^{r_{\mathsf{tc}}}$, where $\phi = H(t, x, \alpha)$ with $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$.
  - Receiver's Computation: $\beta \leftarrow \{0,1\}^{\kappa}$.
- The Decommit phase:
  - Communication: $(m, \alpha, \gamma, r_{\mathsf{tc}})$ where $\alpha \in \mathbb{G}^3$ and $\gamma, r_{\mathsf{tc}} \in \mathbb{Z}/q\mathbb{Z}$.
  - Committer's Computation: None
  - Receiver's Computation: Verify $\psi = g^{\phi} \hat{h}^{r_{\mathsf{tc}}}$, $\alpha_1 = g^{\gamma} C_1^{\beta}$, $\alpha_2 = h^{\gamma}(C_3/m)^{\beta}$, and $\alpha_3 = (c^{\tau} d)^{\gamma} C_3^{\beta}$, where $\tau = H'(t, C_1)$ and $\phi = H(t, x, \alpha)$ with $t = (\mathsf{sid}, \mathsf{ssid}, C, R)$ and $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$.

Applied to Our Static UC-Secure Commitment Scheme.

– Common Reference String: $\mathsf{crs} = (\mathbb{G}, q, H, H', g, h, c, d, \hat{h})$.
– The Commit phase:
  • Communication: $\mathsf{CT} \in \mathbb{G}$.
  • Committer's Computation: $w \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\mathsf{CT} = (C_1, C_2, C_3) = (g^w, m \cdot h^w, (c^\tau d)^w)$ with $\tau = H'(t, C_1)$ for $t = (\mathsf{sid}, \mathsf{ssid}, C, R)$.
  • Receiver's Computation: None.
– The Decommit phase:
  • Communication: $(m, \psi, \alpha, \beta, \gamma, r_{\mathsf{tc}})$ where $\psi \in \mathbb{G}$, $\alpha \in \mathbb{G}^3$, $\beta \in \{0,1\}^{\lambda_{\mathsf{ch}}}$, and $\gamma, r_{\mathsf{tc}} \in \mathbb{Z}/q\mathbb{Z}$.
  • Committer's Computation: $\xi \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\alpha = (\alpha_1, \alpha_2, \alpha_3) = (g^\xi, h^\xi, (c^\tau d)^\xi)$; $\gamma = \xi - \beta w \mod q$; $r_{\mathsf{tc}} \leftarrow \mathbb{Z}/q\mathbb{Z}$; $\psi = g^\phi \hat{h}^{r_{\mathsf{tc}}}$, where $\phi = H(t, x, \alpha)$ with $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$.
  • Receiver's Computation: $\beta \leftarrow \{0,1\}^\kappa$; Verify $\psi = g^\phi \hat{h}^{r_{\mathsf{tc}}}$, $\alpha_1 = g^\gamma C_1{}^\beta$, $\alpha_2 = h^\gamma (C_3/m)^\beta$, and $\alpha_3 = (c^\tau d)^\gamma C_3{}^\beta$, where $\tau = H'(t, C_1)$ and $\phi = H(t, x, \alpha)$ with $t = (\mathsf{sid}, \mathsf{ssid}, C, R)$ and $x = (\mathsf{pk}^{\mathsf{enc}}, m, t, \mathsf{CT})$.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Katz [21], pp. 332–352. See also http://eprint.iacr.org/2014/609
2. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption scheme. In: Krawczyk [23], pp. 26–45
3. Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Analysis and improvement of Lindell's UC-secure commitment schemes. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 534–551. Springer, Heidelberg (2013)
4. Camenisch, J.L., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
5. Canetti, R.: 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
6. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
7. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC 2002, pp. 494–503. ACM (2002). The full version is available at http://eprint.iacr.org/2002/140
8. Cascudo, I., Damgård, I., David, B., Döttling, N., Nielsen, J.B.: Rate-1, linear time and additively homomorphic UC commitments. IACR Cryptology ePrint Archive 2016:137 (2016)

9. Cascudo, I., Damgård, I., David, B.M., Giacomelli, I., Nielsen, J.B., Trifiletti, R.: Additively homomorphic UC commitments with optimal amortized overhead. In: Katz [21], pp. 495–515
10. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk [23], pp. 13–25
11. Damgård, I., David, B.M., Giacomelli, I., Nielsen, J.B.: Compact VSS and efficient homomorphic UC commitments. In: Sarkar and Iwata [30], pp. 213–232
12. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: STOC 2003, pp. 426–437. ACM (2003)
13. Damgård, I.B., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
14. Feigenbaum, J. (ed.): CRYPTO 1991. LNCS, vol. 576. Springer, Heidelberg (1991)
15. Fischlin, M., Libert, B., Manulis, M.: Non-interactive and re-usable universally composable string commitments with adaptive security. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 468–485. Springer, Heidelberg (2011)
16. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Trifiletti, R.: On the complexity of additively homomorphic UC commitments. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A. LNCS, vol. 9562, pp. 542–565. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49096-9_23
17. Fujisaki, E.: All-but-many encryption - a new framework for fully-equipped UC commitments. In: Sarkar and Iwata [30], pp. 426–447
18. Fujisaki, E.: Improving practical UC-secure commitments based on the DDH assumption. IACR Cryptology ePrint Archive 2016:656 (2016)
19. Garay, J.A., Ishai, Y., Kumaresan, R., Wee, H.: On the complexity of UC commitments. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 677–694. Springer, Heidelberg (2014)
20. Groth, J., Sahai, A.: Efficient noninteractive proof systems for bilinear groups. SIAM J. Comput. **41**(5), 1193–1232 (2012)
21. Katz, J. (ed.): PKC 2015. LNCS, vol. 9020. Springer, Heidelberg (2015)
22. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
23. Krawczyk, H. (ed.): CRYPTO 1998. LNCS, vol. 1462. Springer, Heidelberg (1998)
24. Lindell, Y.: Highly-efficient universally-composable commitments based on the DDH assumption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 446–466. Springer, Heidelberg (2011)
25. MacKenzie, P.D., Reiter, M.K., Yang, K.: Alternatives to non-malleability: definitions, constructions, and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 171–190. Springer, Heidelberg (2004)
26. Nishimaki, R., Fujisaki, E., Tanaka, K.: An efficient non-interactive universally composable string-commitment scheme. IEICE Trans. **95−A**(1), 167–175 (2012)
27. Okamoto, T., Pointcheval, D.: REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)
28. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum [14], pp. 129–140

29. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum [14], pp. 434–444
30. Sarkar, P., Iwata, T. (eds.): ASIACRYPT 2014. LNCS, vol. 8874. Springer, Heidelberg (2014)
31. Shoup, V.: A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, December 2001