DL-Extractable UC-Commitment Schemes

Behzad Abdolmaleki[1], Karim Baghery[1], Helger Lipmaa[1], Janno Siim[1], and Michał Zając[2]

[1] University of Tartu, Estonia
[2] Clearmatics, UK

Abstract. We define a new UC functionality (DL-extractable commitment scheme) that allows committer to open a commitment to a group element g^x; however, the simulator will be able to extract its discrete logarithm x. Such functionality is useful in situations where the secrecy of x is important since the knowledge of x enables to break privacy while the simulator needs to know x to be able to simulate the corrupted committer. Based on Fujisaki's UC-secure commitment scheme and the Damgård-Fujisaki integer commitment scheme, we propose an efficient commitment scheme that realizes the new functionality. As another novelty, we construct the new scheme in the weaker RPK (registered public key) model instead of the CRS model used by Fujisaki.

Keywords: CRS model, extractable commitment, RPK model, universal composability, UC commitment

1 Introduction

A commitment scheme is one of the most basic primitives in cryptography. Essentially, it implements a digital safe: in the commitment phase, the committer puts her message to the safe, locks it, and hands it to the receiver. In the opening phase, the committer uses her key to open the safe. Thus, a commitment scheme satisfies at least the following two properties: it is binding (the committer cannot change the committed message) and hiding (before the opening, the receiver does not know which message was committed to).

In many applications, commitment schemes must satisfy stronger properties. In the case of *UC-security* [8], one first defines an ideal functionality (e.g., the functionality of the commitment scheme) and then constructs a protocol that UC-realizes this functionality. Such protocol is said to be UC-secure. Due to Canetti's composition theorem [8], a UC-secure protocol enjoys secure composability with arbitrary protocols, without the need to reprove its security. Importantly, UC-secure protocols do not have to be modified to be secure in a specific software environment and thus can be used as a black-box by practitioners. As such, UC is the recommended best practice in cryptographic engineering.

The first UC-commitment scheme was proposed by Canetti and Fischlin [9]. A UC-commitment scheme was shown to be complete for the construction of

UC-secure zero knowledge protocols [9,13] and two-party and multi-party computations [10]. UC-commitment schemes have to satisfy the properties of extractability (the simulator can unambiguously extract the committed message) and equivocability (the simulator can open a commitment to an arbitrary value) at the same time, and thus they cannot be constructed without an additional setup assumption [9]. The most widely known setup assumption is the common reference string (CRS, [6]) model that allows for a *universally trusted* entity that generates the CRS from the correct distribution without revealing its trapdoor.

Many different CRS-model UC-commitment schemes are known, starting with [9,10,13,7]. Lindell [18] proposed the first efficient scheme based on an ordinary prime-order group. Blazy *et al.* [5] corrected a bug in Lindell's scheme and proposed a new scheme with additional optimizations. Fujisaki [15] further optimized the scheme of Blazy *et al.*, obtaining the most efficient currently known UC-commitment scheme Fuj in an ordinary prime-order group.

The main idea of the UC-commitment schemes of [18,5,15] is that the committer C encrypts the message m. During the opening phase, C outputs m together with an interactive proof (a Σ-protocol) that she really encrypted m. She also erases the used randomizer (hence, these commitments schemes assume secure erasure). The UC simulator simulates the Σ-protocol using the CRS trapdoor; to achieve UC-security, the Σ-protocol has to be straight-line extractable. Due to the use of a Σ-protocol, [18,5,15] have either an interactive commit phase (resulting in adaptive security) or an interactive opening phase (resulting in static security). Within this paper, we will concentrate on adaptively secure variants. Fischlin, Libert, and Manulis [14] used a Groth-Sahai proof [16] instead of a Σ-protocol to construct a non-interactive adaptive UC-commitment scheme; however, their scheme is computationally less efficient and uses bilinear pairings.

An important question, often asked by practitioners, is how to implement the CRS model. More precisely, how can one guarantee the existence of a *single* party $\mathcal{R}$ that can be trusted by *everybody* to choose the CRS from the correct distribution without leaking its trapdoors? Fortunately, weaker setup models are known. Barak, Canetti, Nielsen, and Pass [2] introduced the weaker registered public key (RPK) model where it is essentially required that each party $\mathcal{G}_i$ must trust *some* key registration authority $\mathcal{R}_i$ who registers his key. The authorities $\mathcal{R}_i$ can coincide or be different, depending on the application. They do not need to trust each other. In particular, the CRS model is a very strong case of the RPK model where there is only one authority $\mathcal{R}$ whom all parties have to trust. Barak *et al.* [2] proposed a UC-commitment scheme that is secure in the RPK model: in fact, they used the property of a known UC-commitment scheme in the CRS model that its CRS can be divided into two parts: a binding part (trusted by the receiver R) and a hiding part (trusted by the committer C). Thus, the binding part can be registered by the authority of R and the hiding part can be registered by the authority of C. Unfortunately, their scheme is quite inefficient.

Moreover, the functionality of UC-commitments is not always sufficient. E.g., consider the following generic class of (UC-secure) pairing-based multiplicative

public key generation protocols. (This protocol is motivated by a non-UC-secure CRS-generation protocol for SNARKs from [4] that can be used also to generate the CRS of UC-secure SNARKs like [17].) Let $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2, g_T)$ be an (asymmetric) prime-order bilinear group where g_i is a generator of $\mathbb{G}_i$. Different parties $\mathcal{G}_i$, $i \in [1 \,..\, \mathsf{N}_p]$, sample their one-time public keys $(g_1^{\sigma_i}, g_2^{\sigma_i})$, where σ_i is the secret key, and UC-commit to them. After all parties have committed, everybody will open the commitments to their public keys. Next, they will enact a sequential protocol where the ith party computes $g_1^{\sigma_i^*} := g_1^{\prod_{j=1}^{i} \sigma_j}$ as $g_1^{\sigma_i^*} \leftarrow \left(g_1^{\sigma_{i-1}^*}\right)^{\sigma_i}$, by using (a public group element) $g_1^{\sigma_{i-1}^*}$ and (a secret integer) σ_i. Under the minimal assumption that at least one of the parties $\mathcal{G}_i$ is honest, it is required that the joint public key $g_1^{\sigma_{\mathsf{N}_p}^*}$ is uniformly random and that no coalition of less than N_p knows the corresponding secret key $\sigma_{\mathsf{N}_p}^*$. Due to this, σ_i should not be leaked while opening to $g_2^{\sigma_i}$ is needed for public verification of the correctness of the operation of $\mathcal{G}_i$. Namely, for this, one needs to check that $\hat{e}(g_1^{\sigma_i^*}, g_2) = \hat{e}(g_1^{\sigma_{i-1}^*}, g_2^{\sigma_i})$; thus, avoiding the use of costly zero-knowledge protocols.

On the other hand, in the security proof, the UC simulator Sim needs to recover σ_i (and not only $(g_1^{\sigma_i}, g_2^{\sigma_i})$) to be able to simulate the operation of a corrupted party. Hence, we have arrived to the requirement that after the committer commits to a message m, it should be opened to (g_1^m, g_2^m) while the simulator must be able to extract m from the functionality.

Similar functionality is needed to achieve security in other UC protocols, especially in the setting where one uses a DL-based cryptosystem (or a commitment scheme) to encrypt the witness yet needs to extract the witness for simulation purposes. It can be implemented by encrypting the witness (that has to be extractable) bitwise, and then giving a NIZK argument that each ciphertext encrypts a Boolean value $m \in \{0, 1\}$. Protocols using such a technique have obviously huge communication.

Finally, non-falsifiable assumptions are usually used to (i) extract a unique long message from a succinct commitment, one can avoid such use of non-falsifiable assumptions by having a linearly-long commitment (as done, say, in [17]), and (ii) extract the exponent from a group element, for example, in the case one uses the Groth-Sahai commitment scheme for scalars [16]. To avoid using knowledge assumptions in this case, one can use a DL-extractable commitment scheme that we define in the current paper.

Our Contributions. Let $\mathbb{G}$ be a prime-order group with generator g. We will define the new ideal functionality $\mathcal{F}_{\mathsf{mcomdl}}$ of a *DL-extractable commitment scheme*. Intuitively, the main difference between $\mathcal{F}_{\mathsf{mcomdl}}$ and the standard functionality $\mathcal{F}_{\mathsf{mcom}}$ of UC-commitment schemes [9] is that in $\mathcal{F}_{\mathsf{mcomdl}}$, the committer sends m to the functionality who stores m. When opening the commitment, the functionality $\mathcal{F}_{\mathsf{mcomdl}}$ only sends $g^m \in \mathbb{G}$ (while $\mathcal{F}_{\mathsf{mcom}}$ sends m itself) to the receiver. Since the functionality stores m, it means that after the committer is corrupted, the UC simulator will get to know m.

We seem to be the first to formalize $\mathcal{F}_{\mathsf{mcomdl}}$ as a separate functionality (see Remark 1 in Section 3 for a comparison to the notion of P-extractability of Belenkiy *et al.* [3]); such a formalization creates a common language and enables other researchers to use our implementation of $\mathcal{F}_{\mathsf{mcomdl}}$ as a black-box. At this moment it is even difficult to search for papers that implicitly use this functionality due to lack of agreed-upon language and notation. We expect there to be more applications after the current work establishes the common language.

After that, we construct a commitment scheme Γ_{dl} that UC-realizes $\mathcal{F}_{\mathsf{mcomdl}}$ in the $\mathcal{F}_{\mathsf{rpk}}$-hybrid model, i.e., assuming availability of a UC-secure realization of the RPK model. Essentially, Γ_{dl} is based on Fujisaki's CRS-model UC-commitment scheme Fuj [15] with the following important modifications. First, [18,5,15] all work in the CRS model. We crucially observe that the commitment key of Fuj consists of two independent parts, one guaranteeing hiding and another one guaranteeing binding. Relying on this separation, we will lift Fuj (and also its DL-extractable version) to the weaker RPK model. Since the RPK model seems to be relatively unknown in the community, reintroducing it and constructing an efficient commitment scheme in this model can be seen as another major contribution of the current work.

Second, to guarantee DL-extractability, we proceed as follows. Recall that one of the optimizations of Fujisaki compared to the prior schemes of [18,5] is the use of the efficient IND-PCA secure Short Cramer-Shoup (SCS, [1]) public-key cryptosystem. We couple the SCS encryption of g^m with an additively homomorphic Paillier encryption [19] of m, an integer commitment [12] to m, and a straight-line extractable Σ-protocol showing that these three encryptions/commitments of m are mutually consistent. The UC simulator uses the additively homomorphic encryption (importantly, the simulator does not have to rewind the Σ-protocol) to extract m from a corrupted committer. Thus, the Paillier encryption is needed for extraction while the integer commitment is needed to prove that the SCS plaintext g^{m_1} and the Paillier plaintext m_2 satisfy $m_1 \equiv m_2 \pmod{p}$ where p is the order of $\mathbb{G}$.

The construction of Γ_{dl} and its security proof are somewhat subtle due to the use of three different algebraic/number-theoretic settings (prime-order bilinear groups, Paillier encryption modulo $N = PQ$, and an integer commitment scheme). However, most of this subtlety is needed to construct the Σ-protocol and to prove its security.

Finally, the functionality of a DL-extractable commitment scheme can be straightforwardly generalized to that of a preimage-extractable commitment scheme where the map $m \mapsto g^m$ is replaced by $m \mapsto F(m)$ for any one-way permutation F. We leave study of such a generalization to the future work.

2 Preliminaries

Let PPT denote probabilistic polynomial-time. Let $\lambda \in \mathbb{N}$ be the information-theoretic security parameter, in practice, e.g., $\lambda = 128$. All adversaries will be stateful. For an algorithm $\mathcal{A}$, let $\mathsf{RND}(\mathcal{A})$ denote the random tape of $\mathcal{A}$, and

Functionality $\mathcal{F}_{\mathsf{rpk}}^{f}$

$\mathcal{F}_{\mathsf{rpk}}^{f}$ proceeds as follows, given function f and security parameter λ, running with parties $\mathcal{G}_i$ and an adversary Sim. Initially, a set R of strings is set to be empty.

Registration: When receiving a message $(\mathtt{register}, \mathsf{sid})$ from a party $\mathcal{G}_i$ (either corrupted or uncorrupted) send $(\mathtt{register}, \mathsf{sid}, \mathcal{G}_i)$ to Sim and receive p' from Sim. If $p' \in R$ then let $p \leftarrow p'$. Otherwise, $r \leftarrow_{\$} \{0,1\}^{\lambda}$, let $p \leftarrow f(r)$, and add p to R. Record $(\mathcal{G}_i, p)$ and return (sid, p) to $\mathcal{G}_i$ and to Sim.

Registration by a corrupted party: When receiving a message $(\mathtt{register}, \mathsf{sid}, r)$ from a corrupted party $\mathcal{G}_i$, record $(\mathcal{G}_i, f(r))$. In this case, $f(r)$ is not added to R.

Retrieval: When receiving a message $(\mathtt{retrieve}, \mathsf{sid}, \mathcal{G}_i)$ from party $\mathcal{G}_j$, send $(\mathtt{retrieve}, \mathsf{sid}, \mathcal{G}_i, \mathcal{G}_j)$ to Sim, and obtain a value p from Sim. If $(\mathcal{G}_i, p)$ is recorded then return $(\mathsf{sid}, \mathcal{G}_i, p)$ to $\mathcal{G}_j$. Else, return $(\mathsf{sid}, \mathcal{G}_i, \perp)$ to $\mathcal{G}_j$.

Functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$

$\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$ is parametrized by a distribution $\mathcal{D}$. It proceeds as follows, running with parties $\mathcal{G}_i$ and an adversary Sim.

CRS generation: Sample $\mathsf{crs} \leftarrow_{\$} \mathcal{D}$.

Retrieval: upon receiving $(\mathtt{retrieve}, \mathsf{sid})$ from $\mathcal{G}_i$, send $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$ to $\mathcal{G}_i$.

Fig. 1: Functionalities $\mathcal{F}_{\mathsf{rpk}}^{f}$ and $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D}}$

let $r \leftarrow_{\$} \mathsf{RND}(\mathcal{A})$ denote sampling of a randomizer r of sufficient length for $\mathcal{A}$'s needs. By $y \leftarrow \mathcal{A}(x; r)$ we denote that $\mathcal{A}$, given an input x and a randomizer r, outputs y. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function, and by $\mathsf{poly}(\lambda)$ an arbitrary polynomial function. $\mathcal{D}_1 \approx_c \mathcal{D}_2$ means that the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable.

UC Security. We work in the standard universal composability framework of Canetti [8] with static corruptions of parties. For consistency, we use the definition of computational indistinguishability, denoted by $\approx_c$, from that work. The UC framework defines a PPT environment machine $\mathcal{Z}$ that oversees the execution of a protocol in one of two worlds. The "ideal world" execution involves "dummy parties" (some of whom may be corrupted by an ideal adversary/simulator Sim) interacting with a functionality $\mathcal{F}$. The "real world" execution involves PPT parties (some of whom may be corrupted by a PPT real world adversary $\mathcal{A}$) interacting only with each other in some protocol π. We refer to [8] for a detailed description of the executions, and a definition of the real world ensemble $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and the ideal world ensemble $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}$.

A protocol π *UC-securely computes* $\mathcal{F}$ if there exists a PPT Sim such that for every non-uniform PPT $\mathcal{Z}$ and PPT $\mathcal{A}$, $\{\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}(\lambda, \mathsf{x})\}_{\lambda\in\mathbb{N},\mathsf{x}\in\{0,1\}^*} \approx_c \{\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, \mathsf{x})\}_{\lambda\in\mathbb{N},\mathsf{x}\in\{0,1\}^*}$.

The importance of this definition is a composition theorem that states that any protocol that is universally composable is secure when run concurrently with many other arbitrary protocols; see [8,10] for discussions and definitions.

RPK and CRS Models. In the registered public key (RPK, [2]) model, it is assumed that each party $\mathcal{G}_i$ trusts *some* key-registration authority $\mathcal{R}_i$ and has registered her key with $\mathcal{R}_i$. (The same $\mathcal{R}_i$ can be used by several parties, or each

party can choose to trust a separate authority.) If $\mathcal{G}_i$ is honest, then the secret key exists and the public key comes from correct distribution (in this case, the public key is said to be *"safe"*). If $\mathcal{G}_i$ is dishonest, the secret key still exists (and the public key has been computed from it honestly) but there is no guarantee about its distribution (in this case, the public key is said to be *"well-formed"*). See Fig. 1 for the description of the functionality of the key registration from [2].

Several different variants (most importantly, the "traditional proof-of-knowledge" version where the secret key and the public key are generated by $\mathcal{G}_i$ who then sends the public key to $\mathcal{R}_i$ and proves the knowledge of the secret key to $\mathcal{R}_i$ by using a stand-alone zero-knowledge proof) of the RPK model are known. The new commitment can be implemented in any of such variants of the RPK model; in particular the definition of the $\mathcal{F}_{\mathsf{rpk}}$-hybrid model does not depend on the variant. We assume that each party knows the identities of all other parties and their key-registration authorities, see [2] for discussion.

In the CRS model [6], there is a single, universally trusted, third party (TTP) that picks a common reference string crs from a well-defined probability distribution and makes it available to all parties. An ideal functionality realizing the CRS model is presented on Fig. 1. In a usual implementation, crs comes with a secret trapdoor td, such that td is sampled from a well-defined distribution $\mathcal{D}_{\mathsf{td}}$, and for some public function f, we have $\mathsf{crs} \leftarrow f(\mathsf{td})$. In the case of a NIZK argument system, the knowledge of td allows the simulator to prove statements outside of the language. Here, it is assumed that TTP only provides td to the simulator but not to the adversary. The CRS model can be seen as a very strong version of the RPK model where all parties $\mathcal{G}_i$ trust the same TTP $\mathcal{R}$.

We denote an execution of π in the RPK-hybrid (the CRS-hybrid case is similar) model by $\mathsf{HYBRID}^{\mathcal{F}^f_{\mathsf{rpk}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda,\mathsf{x})$. A protocol π *UC-securely computes* $\mathcal{F}$ in the $\mathcal{F}^f_{\mathsf{rpk}}$-hybrid model if there exists a PPT Sim such that every non-uniform PPT $\mathcal{Z}$ and PPT $\mathcal{A}$, $\{\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}(\lambda,\mathsf{x})\}_{\lambda\in\mathbb{N},\mathsf{x}\in\{0,1\}^*} \approx_c \{\mathsf{HYBRID}^{\mathcal{F}^f_{\mathsf{rpk}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda,\mathsf{x})\}_{\lambda\in\mathbb{N},\mathsf{x}\in\{0,1\}^*}$.

Root Assumption. An integer is $C(\lambda)$-*smooth* if all its prime factors are at most $C(\lambda)$, and $C(\lambda)$-*rough* [12] if all its prime factors are larger than $C(\lambda)$.

Let $\tilde{\mathbb{G}} = \mathbb{U} \times \mathbb{H}$ be a multiplicative abelian group such that $\mathbb{H}$ has order divisible only by large primes. That is, let $C(\lambda)$ and $l(\lambda)$ be two functions from $\mathbb{Z}^+$ to $\mathbb{Z}^+$, such that $C(\lambda)$ is superpolynomial and $l(\lambda)$ is polynomial. Let 2^B be an efficiently computable upperbound on $|\tilde{\mathbb{G}}|$, $2^B \geq \mathrm{ord}(\tilde{\mathbb{G}})$. Denote $l_{\tilde{\mathbb{G}}} := \mathrm{ord}(\mathbb{U})$. We assume $l_{\tilde{\mathbb{G}}} \leq l(\lambda)$, the description $\mathrm{descr}(\tilde{\mathbb{G}})$ of $\tilde{\mathbb{G}}$ includes $l_{\tilde{\mathbb{G}}}$, and that it is easy to verify whether some bitstring represents an element of $\tilde{\mathbb{G}}$. Let $\mathcal{G}(1^\lambda)$ generate $\mathrm{descr}(\tilde{\mathbb{G}})$ that has the mentioned properties. In the following instantiation, the root assumption is the same as the well-known Strong RSA assumption. (Another known instantiation [12] is based on class groups.)

Note that if $\tilde{\mathbb{G}} = \mathbb{U} \times \mathbb{H}$ is the multiplicative group modulo $N = PQ$ where $P = 2P' + 1$ and $Q = 2Q' + 1$ are safe primes, then $\mathrm{ord}(\tilde{\mathbb{G}}) = \varphi(N) = 4P'Q'$. (This setting is often recommended if one uses the RSA or the Paillier cryp-

tosystem [19].). In this case, $\mathbb{U} \cong \mathbb{Z}_2 \times \mathbb{Z}_2$ is a group of order $l_{\tilde{\mathbb{G}}} = 4$ and $\mathbb{H}$ is a group of order $P'Q'$. Here, $\mathrm{descr}(\tilde{\mathbb{G}}) = \{N, l_{\tilde{\mathbb{G}}}\}$.

Consider the following experiment:

$\mathsf{Expt}^{\mathsf{root}}_{\Pi,\mathcal{A}}(\lambda)$

$\mathrm{descr}(\tilde{\mathbb{G}}) \leftarrow \mathcal{G}(1^\lambda); Y \leftarrow_\$ \tilde{\mathbb{G}}; (e, X, \mu) \leftarrow \mathcal{A}(\mathrm{descr}(\tilde{\mathbb{G}}), Y);$
if $e \in \mathbb{Z} \wedge e > 1 \wedge X \in \tilde{\mathbb{G}} \wedge \mu \in \mathbb{U} \wedge Y = \mu X^e$
then return 1; **else return** 0; **fi**

The *root assumption* [12] holds relative to $\mathcal{G}$, if for all λ and PPT $\mathcal{A}$, $\Pr[\mathsf{Expt}^{\mathsf{root}}_{\Pi,\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$.

Commitment Schemes. A *commitment scheme* $\Gamma = (\Gamma.\mathsf{Gen}, \Gamma.\mathsf{Com}, \Gamma.\mathsf{Vf})$ is defined by three PPT algorithms: (i) $\Gamma.\mathsf{Gen}(1^\lambda)$ generates a public key (CRS) $\Gamma.\mathsf{ck}$ and a secret key (trapdoor) $\Gamma.\mathsf{td}$; (ii) $\Gamma.\mathsf{Com}(\Gamma.\mathsf{ck}; m; r)$ commits to m under the CRS ck, using the random coins r. It outputs commitment c and opening information op; (iii) $\Gamma.\mathsf{Vf}(\Gamma.\mathsf{ck}; \mathsf{c}, m, \mathsf{op})$ verifies that c is a commitment to m.

It is required that for any $(\Gamma.\mathsf{ck}, \Gamma.\mathsf{td}) \leftarrow \Gamma.\mathsf{Gen}(1^\lambda)$ (where $\Gamma.\mathsf{td}$ is unused unless Γ has a trapdoor property), message m, randomizer r, and $(\mathsf{c}, \mathsf{op}) \leftarrow \Gamma.\mathsf{Com}(\Gamma.\mathsf{ck}; m; r)$, it holds that $\Gamma.\mathsf{Vf}(\Gamma.\mathsf{ck}; \mathsf{c}, m, \mathsf{op}) = 1$. Γ is *statistically hiding*, if the distributions of commitment c, corresponding to any two values of m, are statistically indistinguishable. Γ is *computationally binding*, if given ck and c, no PPT adversary $\mathcal{A}$ can create two different messages m_i with corresponding openings op_i, such that $\Gamma.\mathsf{Vf}(\Gamma.\mathsf{ck}; \mathsf{c}, m_1, \mathsf{op}_1) = \Gamma.\mathsf{Vf}(\Gamma.\mathsf{ck}; \mathsf{c}, m_2, \mathsf{op}_2) = 1$ with a non-negligible probability.

Trapdoor Commitment Schemes. A commitment scheme Γ is *trapdoor* if there exists a PPT algorithm $\Gamma.\mathsf{tdOpen}$, such that given the trapdoor $\Gamma.\mathsf{td}$ (corresponding to commitment key $\Gamma.\mathsf{ck}$), two messages m_1 (with opening op_1) and m_2, and any commitment c: if $\Gamma.\mathsf{Vf}(\Gamma.\mathsf{ck}; m_1, \mathsf{c}, \mathsf{op}_1) = 1$ then $\Gamma.\mathsf{tdOpen}(\Gamma.\mathsf{td}; m_1, \mathsf{op}_1, m_2) = \mathsf{op}_2$, such that $\Gamma.\mathsf{Vf}(\Gamma.\mathsf{ck}; m_2, \mathsf{c}, \mathsf{op}_2) = 1$. The Pedersen trapdoor commitment scheme $\mathsf{Ped} = (\mathsf{Ped.Gen}, \mathsf{Ped.Com}, \mathsf{Ped.Vf}, \mathsf{Ped.tdOpen})$ [20] in cyclic group $\mathbb{G}$, with generator g, is defined as follows:

$\mathsf{Ped.Gen}(1^\lambda)$**:** sample $\mathsf{td} \leftarrow_\$ \mathbb{Z}_p$, set $h \leftarrow g^{\mathsf{td}}$, and output $(\mathsf{Ped.ck} = (g, h), \mathsf{Ped.td} \leftarrow \mathsf{td})$.
$\mathsf{Ped.Com}(\mathsf{Ped.ck}; m; r)$ for $m \in \mathbb{Z}_p$, $r \leftarrow_\$ \mathbb{Z}_p$: output $(\mathsf{c}, \mathsf{op}) = (g^m h^r, r)$.
$\mathsf{Ped.Vf}(\mathsf{Ped.ck}; m, \mathsf{c}, \mathsf{op} = r)$**:** output 1 if $\mathsf{c} = g^m h^r$ and 0 otherwise.
$\mathsf{Ped.tdOpen}(\mathsf{Ped.td}; m_1, \mathsf{op}_1 = r_1, m_2)$**:** output $\mathsf{op}_2 = r_2 \leftarrow (m_1 - m_2)/\mathsf{td} + r_1$.

It is well-known that Ped is perfectly hiding, computationally binding under the discrete logarithm assumption, and trapdoor.

Integer Commitment Schemes (ICS). A commitment scheme is an ICS if the messages come from domain $\mathbb{Z}$. Thus, statistical hiding means that it is intractable to compute two different *integers* $m_1, m_2 \in \mathbb{Z}$ and corresponding openings op_1 and op_2, such that $\mathsf{Vf}(\mathsf{ck}; \mathsf{c}, m_1, \mathsf{op}_1) = \mathsf{Vf}(\mathsf{ck}; \mathsf{c}, m_2, \mathsf{op}_2) = 1$. In the case of Pedersen, m and $m + p$ have the same commitments and thus Ped is not an ICS. Let $\tilde{\mathbb{G}}$ be a group where the root assumption holds. The Damgård-Fujisaki ICS [12] over $\tilde{\mathbb{G}}$ works as follows:

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

$\mathcal{F}_{\mathsf{mcom}} = \mathcal{F}_{\mathsf{mcom}}^{\mathcal{M}}$ interacts with parties $\mathcal{G}_i$ and adversary Sim as follows.

- **Upon receiving** $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m \in \mathcal{M})$ **from** $\mathcal{G}_i$**:** if a tuple $(\mathsf{sid}, \mathsf{cid}, \ldots)$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously stored, do nothing. Otherwise, store $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ and send $(\mathtt{rcpt}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j)$ to $\mathcal{G}_j$ and Sim.
- **Upon receiving** $(\mathtt{open}, \mathsf{sid}, \mathsf{cid})$ **from** $\mathcal{G}_i$**:** if a tuple $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ was previously stored then send $(\mathtt{open}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ to $\mathcal{G}_j$ and Sim. Otherwise, ignore.

Fig. 2: Functionality $\mathcal{F}_{\mathsf{mcom}}$ for committing multiple messages

$\mathsf{DF.Gen}(1^\lambda)$**:** V chooses an $\tilde{h} \in \tilde{\mathbb{G}}$ s.t. $\mathrm{ord}(\tilde{h})$ is $C(\lambda)$-rough, and sets $\tilde{g} \leftarrow \tilde{h}^\alpha$ where $\alpha \leftarrow_\$ \mathbb{Z}_{2^{2B+\lambda}}$. V sends $\mathsf{DF.ck} = (\tilde{g}, \tilde{h})$ to P and proves that $\tilde{g} \in \langle \tilde{h} \rangle$.
$\mathsf{DF.Com}(\mathsf{DF.ck}; m; r)$ for $m \in \mathbb{Z}$, $r \leftarrow_\$ \mathbb{Z}_{2^{B+\lambda}}$: output $\mathsf{c} \leftarrow \tilde{g}^m \tilde{h}^r$, $\mathsf{op} = (1, r)$.
$\mathsf{DF.Vf}(\mathsf{DF.ck}; m, \mathsf{c}, \mathsf{op} = (\mu, r))$**:** check that $\mathsf{c} = \mu \tilde{g}^m \tilde{h}^r$ and $\mu^{l_{\tilde{\mathbb{G}}}} = 1$.

See [12] for a discussion on μ and other details. As proven in [12], DF is statistically hiding and computationally binding under the root assumption.

UC Commitments. A (multi-use) UC-commitment scheme [9] implements the functionality $\mathcal{F}_{\mathsf{mcom}}$, see Fig. 2. The $\mathcal{F}_{\mathsf{mcom}}$ functionality takes as an additional input another unique "commitment identifier" cid, which is used if a sender commits to the same receiver multiple times within a session. We assume that the combination of $(\mathsf{sid}, \mathsf{cid})$ is globally unique, [9]. UC-commitment schemes have to satisfy the properties of extractability (the simulator can unambiguously extract the committed message) and equivocability (the simulator can open a commitment to an arbitrary value) at the same time, and thus they cannot be constructed without an additional setup assumption [9].

Cryptosystems. A *labelled public-key cryptosystem* Π is defined by three PPT algorithms: (i) $\Pi.\mathsf{KGen}(1^\lambda)$ generates a public key $\Pi.\mathsf{pk}$ and a secret key $\Pi.\mathsf{sk}$; (ii) $\Pi.\mathsf{Enc}^{\mathsf{lbl}}_{\Pi.\mathsf{pk}}(m; r)$ encrypts the message m under the key $\Pi.\mathsf{pk}$ with label lbl, using the random coins r; (iii) $\Pi.\mathsf{Dec}^{\mathsf{lbl}}_{\Pi.\mathsf{sk}}(\mathsf{c})$ decrypts the ciphertext c, using the secret key $\Pi.\mathsf{sk}$ with label lbl. It is required that for all $(\Pi.\mathsf{pk}, \Pi.\mathsf{sk}) \in \Pi.\mathsf{KGen}(1^\lambda)$, all labels lbl, all random coins r and all messages m, $\Pi.\mathsf{Dec}^{\mathsf{lbl}}_{\Pi.\mathsf{sk}}(\Pi.\mathsf{Enc}^{\mathsf{lbl}}_{\Pi.\mathsf{pk}}(m; r)) = m$.

IND-CPA (indistinguishability under the chosen plaintext attack) and IND-PCA (indistinguishability under the plaintext checking attacks, [1]) are defined by using the following experiments:

$\mathsf{Expt}^{\mathsf{pca}}_{\Pi,\mathcal{A}}(\lambda)$ / $\mathsf{Expt}^{\mathsf{cpa}}_{\Pi,\mathcal{A}}(\lambda)$

$\mathcal{Q} \leftarrow \emptyset$; $(\Pi.\mathsf{pk}, \Pi.\mathsf{sk}) \leftarrow \Pi.\mathsf{KGen}(1^\lambda)$; $(\mathsf{lbl}^*, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot,\cdot)}(\Pi.\mathsf{pk})$;
$b \leftarrow_\$ \{0,1\}$; $r \leftarrow_\$ \mathsf{RND}(\Pi)$; $\mathsf{c}^* \leftarrow \Pi.\mathsf{Enc}^{\mathsf{lbl}^*}_{\Pi.\mathsf{pk}}(m_b; r)$; $b' \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot,\cdot)}(\mathsf{c}^*)$;
if $(\mathsf{lbl}^*, \mathsf{c}^*) \notin \mathcal{Q}$ **then return** $b = b'$; **fi** ;

The experiment-dependent oracle is defined as follows: (i) in $\mathsf{Expt}^{\mathsf{cpa}}_{\Pi,\mathcal{A}}(\lambda)$, $\mathcal{O}(\cdot,\cdot,\cdot)$ returns always 0. (ii) in $\mathsf{Expt}^{\mathsf{pca}}_{\Pi,\mathcal{A}}(\lambda)$, $\mathcal{O}(\mathsf{lbl}, \mathsf{c}, m)$ adds $(\mathsf{lbl}, \mathsf{c})$ to $\mathcal{Q}$. It returns 1 if the decryption of c under the label lbl is m. Otherwise, it returns 0.

Π is *IND-CPA secure* if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{cpa}}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Expt}^{\mathsf{cpa}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$. Π is *IND-PCA secure* if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pca}}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Expt}^{\mathsf{pca}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$.

The IND-PCA-secure Short Cramer-Shoup (SCS) labelled cryptosystem $\mathsf{SCS} = (\mathsf{SCS.KGen}, \mathsf{SCS.Enc}, \mathsf{SCS.Dec})$ [1] works as follows:

$\mathsf{SCS.KGen}(1^\lambda)$**:** $g \leftarrow_\$ \mathbb{G}^*$; $x_1, x_2, y_1, y_2, z \leftarrow_\$ \mathbb{Z}_p$; $h \leftarrow g^z$, $c \leftarrow g^{x_1} h^{x_2}$, $d \leftarrow g^{y_1} h^{y_2}$. Choose H from a collision-resistant hash function family $\mathcal{H}$. Return $\mathsf{SCS.pk} = (g, h, c, d, \mathsf{H})$ and $\mathsf{SCS.sk} = (x_1, x_2, y_1, y_2, z)$.

$\mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk}}(g^m \in \mathbb{G}; \cdot)$**:** sample $r \leftarrow_\$ \mathbb{Z}_p$; set $(u, e, v) \leftarrow (g^r, g^m h^r, (cd^\tau)^r)$, where $\tau \leftarrow \mathsf{H}(\mathsf{lbl}, u, e)$. Return the ciphertext $(u, e, v)^\top$.

$\mathsf{SCS.Dec}^{\mathsf{lbl}}_{\mathsf{SCS.sk}}((u, e, v)^\top \in \mathbb{G}^3)$**:** set $\tau \leftarrow \mathsf{H}(\mathsf{lbl}, u, e)$, $g^m \leftarrow e/u^z$; if $u^{x_1+y_1\tau}(e/g^m)^{x_2+y_2\tau} \neq v$ then abort. Otherwise, output g^m.

Abdalla *et al.* [1] proved that SCS is IND-PCA secure given $\mathcal{H}$ is a collision-resistant hash function family and DDH is hard in $\mathbb{G}$.

Additively Homomorphic Cryptosystems. An additively homomorphic public-key cryptosystem has plaintext space equal to $\mathbb{Z}_N$ for integer N, such that the product of two ciphertexts decrypts to the sum of the two corresponding plaintexts. We will use the Paillier cryptosystem Pai [19]. It encrypts plaintexts from $\mathbb{Z}_N$, where N is a well-chosen RSA modulus, and outputs ciphertexts from $\mathbb{Z}_{N^2}$. More precisely, $\mathsf{Pai.Enc}_{\mathsf{Pai.pk}}(m \in \mathbb{Z}_N; r \in \mathbb{Z}_N^*) = (1+N)^m r^N \equiv (1+mN)r^N \pmod{N^2}$. See [19] for more details, including the decryption algorithm. Pai is IND-CPA secure under the standard DCRA assumption [19].

Σ-protocols [11] in the RPK model. Let $\mathbf{R} = \{\mathsf{x}, \mathsf{w}\}$ be an NP-relation. A Σ-protocol $\Sigma = (\Sigma.\mathsf{P}_1, \Sigma.\mathsf{P}_2, \Sigma.\mathsf{Vf}, \Sigma.\mathsf{Sim})$ is a three-round protocol between the prover P and the verifier V, such that the first and the third messages are by the prover, and the second message is by the verifier. Let $\mathsf{rpk_V}$ be the public key of the verifier. P has input $(\mathsf{rpk_V}; \mathsf{x}, \mathsf{w})$ and V has input $(\mathsf{rpk_V}; \mathsf{x})$. The first message is denoted as $a \leftarrow \Sigma.\mathsf{P}_1(\mathsf{rpk_V}; \mathsf{x}, \mathsf{w}; \boldsymbol{s})$, where $\boldsymbol{s} \leftarrow_\$ \mathsf{RND}(\Sigma)$ is sampled from the randomizer space of the protocol. The second message e is chosen uniformly at random from $\{0,1\}^\lambda$, $e \leftarrow_\$ \{0,1\}^\lambda$. The third message is denoted as $z \leftarrow \Sigma.\mathsf{P}_2(\mathsf{rpk_V}; \mathsf{x}, \mathsf{w}; e; \boldsymbol{s})$. The verifier accepts iff $\Sigma.\mathsf{Vf}(\mathsf{rpk_V}; \mathsf{x}; a, e, z) = 1$.

A Σ-protocol is *complete* for $\mathbf{R}$ if a honest verifier always accepts a honest prover. A Σ-protocol is *specially sound* for $\mathbf{R}$ if given an input x and two acceptable views (a, e_1, z_1) and (a, e_2, z_2), $e_1 \neq e_2$, one can efficiently extract a witness w, such that $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$. A Σ-protocol is *statistically special honest-verifier zero-knowledge (SSHVZK)* for $\mathbf{R}$ if for any $\mathsf{rpk_V}$, x and e, $\Sigma.\mathsf{Sim}(\mathsf{rpk_V}; \mathsf{x}, e)$ can first choose a z and then a, such that the simulated view (a, e, z) and the real view, given the same e, have negligible statistical distance.

3 New Functionality $\mathcal{F}_{\mathsf{mcomdl}}$ and Instantiation

In a DL-extractable UC-commitment scheme, one commits to an *integer* m from $\mathbb{Z}_p$ but the opening is to a *group element* $g^m \in \mathbb{G}$. (In particular, m should stay secret from other participants even after the opening.) Nevertheless, we require that there exists an efficient extraction algorithm that can retrieve the discrete logarithm (i.e., the committed *integer*) $m \in \mathbb{Z}_p$ of g^m. That is, while opening

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

$\mathcal{F}_{\mathsf{mcomdl}}$, parametrized by $\mathcal{M} = \mathbb{Z}_p$ and $\mathbb{G}$, interacts with $\mathcal{G}_1, \ldots, \mathcal{G}_{\mathsf{N}_p}$ as follows.

- Upon receiving $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ from $\mathcal{G}_i$, where $m \in \mathbb{Z}_p$: if a tuple $(\mathsf{sid}, \mathsf{cid}, \cdots)$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously recorded, do nothing. Otherwise, record $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ and send $(\mathtt{rcpt}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j)$ to $\mathcal{G}_j$ and Sim.
- Upon receiving $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ from $\mathcal{G}_i$, proceed as follows: if a tuple $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ was previously recorded then send $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, y \leftarrow g^m)$ to $\mathcal{G}_j$ and Sim. Otherwise do nothing.

Fig. 3: DL-extractable functionality $\mathcal{F}_{\mathsf{mcomdl}}$ for committing multiple messages

returns g^m, the extraction returns m. See Fig. 3 for the corresponding functionality $\mathcal{F}_{\mathsf{mcomdl}}$ that is parametrized by $\mathbb{Z}_p$ and $\mathbb{G}$ (this means that $\mathbb{Z}_p$ and $\mathbb{G}$ are "hard-coded" into the functionality). We formalize our goal by letting parties to commit to an integer m (which will be stored by the functionality and thus can be extracted) but opening the commitment to g^m. Hence, any commitment scheme that implements $\mathcal{F}_{\mathsf{mcomdl}}$ must necessarily be DL-extractable.

Remark 1. Belenkiy *et al.* [3] defined P-extractable commitment scheme for an *efficient* function P, as a commitment scheme where one commits to m and opens to m but where the extractor is able to extract $P(m)$. DL-extractable commitment is a variant of P-extractable commitment for $P = \mathrm{DL}$ being an intractable function. If $P(m) = g^m =: \exp_g(m)$ then one obtains a functionality, dual to $\mathcal{F}_{\mathsf{mcomdl}}$. (However, [3] did not consider UC-security and thus did not use the language of functionalities.) Compared to DL-extractability, $\exp_g$-extractability is straightforward to implement: indeed, the notion of $\exp_g$-extractability was motivated by the fact that several well-known commitment schemes like the Groth-Sahai commitment scheme for scalars [16] already had this property.[3] Obtaining DL-extractability is non-trivial since DL is a hard function and thus one has to take special care about making the DL of a message extractable. □

Remark 2. The functionality $\mathcal{F}_{\mathsf{mcomdl}}$ can be straightforwardly generalized to the functionality $\mathcal{F}_{\mathsf{mcom}\text{-}F^{-1}}$ for an arbitrary one-way permutation F, where the opening message includes $y \leftarrow F(m)$ instead of $y \leftarrow g^m$. Since we are interested in the applications of $\mathcal{F}_{\mathsf{mcomdl}}$, we will omit further discussion. □

We implement $\mathcal{F}_{\mathsf{mcomdl}}$ as follows: for $m \in \mathbb{Z}_p$, we encrypt the group element g^m by using the Short Cramer-Shoup encryption [1], encrypt the integer m by using the Paillier [19] additively homomorphic public-key cryptosystem, and finally commit to the integer m by using the Damgård-Fujisaki [12] ICS. We add a Σ-protocol Σ_{eq} proving the knowledge of m that was used in all cases; importantly, only g^m can be extracted from Σ_{eq} and in particular, m will remain secret. Since UC-security does not permit to use rewinding to retrieve m, we use straight-line extraction techniques from [15]. The Σ-protocol is started during

[3] The extractor of this commitment scheme obtains g^m by Elgamal-decrypting the commitment. Since computing DL is intractable, one arrives to the notion of a $\exp_g$-extractable commitment commitment.

1. Denote $\mathsf{RND}(\Sigma_{\mathsf{eq}}) := \mathbb{Z}_{2^{B+\lambda}} \times \mathbb{Z}_p \times [0\,..\,\max(2^{2\lambda}p, T \cdot C(\lambda) \cdot 2^{\lambda}) - 1] \times \mathbb{Z}^*_{N^2} \times [0\,..\,C(\lambda)2^{B+2\lambda} - 1]$. P samples $\boldsymbol{s} := (s_1, s_2, s_3, s_4, s_5) \leftarrow_{\$} \mathsf{RND}(\Sigma_{\mathsf{eq}})$.
   P sets $\tilde{a}_1 \leftarrow \mathsf{DF.Com}(\mathsf{DF.ck_V}; m; s_1)$, $\boldsymbol{a}_2 \leftarrow (g^{s_2}, h^{s_2}, (cd^{\tau})^{s_2})^{\top}$, $a_3 \leftarrow g^{s_3}$, $a_4 \leftarrow (1 + s_3 N)s_4^N \mod N^2$, $\tilde{a}_5 \leftarrow \tilde{g}^{s_3}\tilde{h}^{s_5}$.
   P sends $a \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{P}_1(\mathsf{rpk_V}; \mathsf{x}, \mathsf{w}; \boldsymbol{s}) := (\tilde{a}_1, \boldsymbol{a}_2, a_3, a_4, \tilde{a}_5)$ to V.
2. V sends $e \leftarrow_{\$} \{0,1\}^{\lambda}$ to P.
3. P sets $z_1 \leftarrow r_1 e + s_2$, $z_2 \leftarrow me + s_3$, $z_3 \leftarrow r_2^e s_4 \mod N^2$, $z_4 \leftarrow s_1 e + s_5$. Let $\boldsymbol{z} := (z_1, z_2, z_3, z_4)$. P sends $\Sigma_{\mathsf{eq}}.\mathsf{P}_2(\mathsf{rpk_V}; \mathsf{x}, \mathsf{w}; e; \boldsymbol{s}) := \boldsymbol{z}$ to V.
4. V outputs 1 iff the following holds (otherwise, V outputs 0):
   (a) $\mathsf{c}_1^e \boldsymbol{a}_2 = (g^{z_1}, g^{em}h^{z_1}, (cd^{\tau})^{z_1})^{\top}$,
   (b) $g^{em} \cdot a_3 = g^{z_2}$,
   (c) $\mathsf{c}_2^e a_4 \equiv (1+N)^{z_2} z_3^N \pmod{N^2}$,
   (d) $\tilde{a}_1^e \tilde{a}_5 = \tilde{g}^{z_2}\tilde{h}^{z_4}$, and
   (e) $z_2 \in [-T \cdot C(\lambda)\,..\,T \cdot C(\lambda)(2^{\lambda} + 1)]$.
   Denote this check by $\Sigma_{\mathsf{eq}}.\mathsf{Vf}(\mathsf{rpk_V}; \mathsf{x}; a, e, \boldsymbol{z}) \in \{0,1\}$.

Fig. 4: Σ-protocol Σ_{eq} for $\mathbf{R}_{\mathsf{eq}}$, where in the honest case, $\mathsf{c}_1 = (\mathsf{c}_{11}, \mathsf{c}_{12}, \mathsf{c}_{13})^{\top} \leftarrow \mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk_P}}(g^m; r_1) = (g^{r_1}, g^m h^{r_1}, (cd^{\tau})^{r_1})^{\top}$ and $\mathsf{c}_2 \leftarrow \mathsf{Pai.Enc}_{\mathsf{Pai.pk_P}}(m; r_2) = (1+N)^m r_2^N \equiv (1+mN)r_2^N \mod N^2$. Here, $r_1 \leftarrow_{\$} \mathbb{Z}_p$, $\tau = \mathsf{H}(\mathsf{lbl}, \mathsf{c}_{11}, \mathsf{c}_{12})$, and $r_2 \leftarrow_{\$} \mathbb{Z}^*_N$.

the commit phase, and after that the committer C erases the used random coins. In the opening phase, C opens the commitment to g^m by finishing Σ_{eq}. When simulating an honest committer, the UC simulator Sim first commits to 0; Sim uses the properties of a trapdoor commitment scheme and the SSHVZK property to simulate Σ_{eq}. (This guarantees equivocability.) If C is corrupted then Sim uses the knowledge of the Paillier secret key to decrypt the Paillier encryption of m and thus obtains m. (This guarantees extractability.) Thus, we obtain a DL-extractable commitment scheme.

3.1 Σ-Protocol Σ_{eq}

Let SCS be the SCS cryptosystem and Pai be the Paillier cryptosystem. Recall that the plaintext space of SCS is $\mathbb{G}$ (of order p) and the plaintext space of Pai is $\mathbb{Z}_N$ for an $N > p$. (The case $N = p$ is straightforward to handle.) Let

$$\mathbf{R}_{\mathsf{eq}} = \left\{ \begin{array}{l} (\mathsf{x} = (\mathsf{p}, \mathsf{SCS.pk_P}, \mathsf{Pai.pk_P}, g^m, \mathsf{c}_1, \mathsf{c}_2, \mathsf{lbl}), \mathsf{w} = (m', r_1, r_2)) : \\ \mathsf{c}_1 = \mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk_P}}(g^m; r_1) \wedge \mathsf{c}_2 = \mathsf{Pai.Enc}_{\mathsf{Pai.pk_P}}(m'; r_2) \wedge \\ m \equiv m' \pmod{p} \wedge m' < N \end{array} \right\},$$

where $\mathsf{p} \leftarrow \mathsf{Pgen}(1^{\lambda})$. Let $\mathbf{L}_{\mathsf{eq}} = \{\mathsf{x} : \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathsf{eq}}\}$ be the corresponding language. Thus, $\mathsf{x} \in \mathbf{L}_{\mathsf{eq}}$ iff the two ciphertexts encrypt g^m and m' respectively, such that $m \equiv m' \pmod{p}$. Note that g^m is public while m is not; this corresponds to the use of g^m in the new DL-extractable UC-commitment scheme.

The proof of the following theorem uses ideas from the proof given in Sect. 5.1 of [12]. Note that in the next theorem, we actually do not need the public key to be registered. We will assume it here for the sake of convenience since registration is needed in the new DL-extractable UC-commitment scheme.

Theorem 1 (Security of Σ_{eq}). *Let* H *be sampled from a collision-resistant hash function family,* SCS *be the SCS cryptosystem,* Pai *be the Paillier cryptosystem, and* DF *be the Damgård-Fujisaki ICS. Assume* V *has registered her public key* $\mathsf{rpk}_{\mathsf{V}} = \mathsf{DF.ck}_{\mathsf{V}}$*. Let* T *be a public constant such that* $m < T$*, e.g.* $T = p$*; let* $C(\lambda) = 2^{\lambda}$ *and let* 2^B *be a close upperbound on* $\mathrm{ord}(\tilde{\mathbb{G}})$*. Assume* $2^{2\lambda+1}p < N$*. The* Σ*-protocol* Σ_{eq} *in Fig. 4 (where* $\Sigma_{\mathsf{eq}}.\mathsf{Sim}$ *will be defined in the SSHVZK proof) is complete and SSHVZK for* $\mathbf{R}_{\mathsf{eq}}$*. The protocol* Σ_{eq} *is computationally specially sound under the root assumption in* $\tilde{\mathbb{G}}$*.*

Proof. **Special soundness:** consider two accepting views $(a, e, \boldsymbol{z})$ and $(a, e', \boldsymbol{z}')$ with $e \neq e'$. Let $m^* \leftarrow (z_2' - z_2)/(e' - e) \mod p$ and $r^* \leftarrow (z_1' - z_1)/(e' - e) \mod p$. We get from the first four verification equations respectively that

$$\begin{aligned}
\mathbf{c}_1 &= (g^{r^*}, g^m h^{r^*}, (cd^{\tau})^{r^*})^{\top} = \mathsf{SCS.Enc}_{\mathsf{SCS.pk_P}}(g^m; r^*) \ , \\
m &\equiv m^* \pmod{p} \ , \\
\mathsf{c}_2^{e'-e} &\equiv (1+N)^{z_2'-z_2}(z_3'/z_3)^N \pmod{N^2} \ , \qquad (1) \\
\tilde{a}_1^{e'-e} &= \tilde{g}^{z_2'-z_2}\tilde{h}^{z_4'-z_4} \ . \qquad (2)
\end{aligned}$$

First, consider Eq. (2). Since $\tilde{g} = \tilde{h}^{\alpha}$, $\tilde{a}_1^{e'-e} = \tilde{h}^{\delta}$ for $\delta := \alpha(z_2' - z_2) + (z_4' - z_4)$. We will next consider three possible cases. Let bad be the event that we either have the case (i) or the case (ii).

(i) $(e' - e) \nmid \delta$ *as an integer.*
Write $\gamma = \gcd(\delta, e' - e)$. By the Extended Euclidean algorithm, there exist i and j (where $j < |e' - e| < C(\lambda)$), such that $j\delta + i(e' - e) = \gamma$. Thus, $\tilde{h}^{\gamma} = \tilde{h}^{j\delta + i(e'-e)} = \tilde{a}_1^{j(e'-e)}\tilde{h}^{i(e'-e)} = (\tilde{a}_1^j\tilde{h}^i)^{e'-e}$. Set now $\mu \leftarrow (\tilde{a}_1^j\tilde{h}^i)^{(e'-e)/\gamma}/\tilde{h}$. Thus, $\mu^{\gamma} = 1$. Since $\gamma < C(\lambda)$, $\mathrm{ord}(\mu)$ is $C(\lambda)$-smooth and thus $\mu^{l_{\tilde{\mathbb{G}}}} = 1$. Since $\tilde{h} = \mu^{-1}(\tilde{a}_1^j\tilde{h}^i)^{(e'-e)/\gamma}$, $((e' - e)/\gamma, \tilde{a}_1^j\tilde{h}^i, \mu^{-1})$ is a solution to the root problem.

(ii) $(e' - e) \mid \delta$ *as an integer, but either* $(e' - e) \nmid (z_2' - z_2)$ *or* $(e' - e) \nmid (z_4' - z_4)$.
Let q be a prime factor of $e' - e$, such that q^j is the highest power of q dividing $e' - e$ and at least one of $z_2' - z_2$ or $z_4' - z_4$ is non-zero modulo q^j (such q exists due to the assumption of non-divisibility). If $q^j \mid (z_2' - z_2)$ then (due to the definition of δ and q^j) also $q^j \mid (z_4' - z_4)$, a contradiction. Thus, $z_2' - z_2 \not\equiv 0 \pmod{q^j}$.

Write $\alpha = a + b \cdot \mathrm{ord}(\tilde{h})$ for some $a < \mathrm{ord}(\tilde{h})$ and b. The adversary only has information about α via the value $\tilde{g}$; moreover, $\tilde{g}$ completely determines a while it contains no information about b. Since $q^j \mid \delta$,

$$\delta = b(z_2' - z_2) \cdot \mathrm{ord}(\tilde{h}) + a(z_2' - z_2) + (z_4' - z_4) \equiv 0 \pmod{q^j} \ . \qquad (3)$$

Because q is a prime factor of $e' - e$ and $e' - e < C(\lambda)$, $q < C(\lambda)$ and thus $\mathrm{ord}(\tilde{h}) \not\equiv 0 \pmod{q}$. From the adversary's viewpoint, b is chosen uniformly at

random from a set of at least $2^{B+\lambda}$ values, and it must satisfy Eq. (3) for bad to be true. Eq. (3) has at most $\eta := \gcd((z_2' - z_2) \cdot \mathrm{ord}(\tilde{h}), q^j)$ solutions. Clearly, η is a power of q but it is at most q^{j-1}. Since $2^{B+\lambda} > 2^\lambda q^j$, the distribution of $b \mod q^j$ is statistically close to uniform in $\mathbb{Z}_{q^j}$, with the probability that b satisfies Eq. (3) being at most $1/q - 2^{-\lambda} \leq 1/2 - 2^{-\lambda}$. Thus, given the event bad, the case (i), where we *can* solve the root problem, happens with high probability.

(iii) $(e' - e) \mid (z_2' - z_2)$ *and* $(e' - e) \mid (z_4' - z_4)$ *as an integer.*
Let $m^\dagger \leftarrow (z_2' - z_2)/(e' - e) \in \mathbb{Z}$ and $r^\dagger \leftarrow (z_4' - z_4)/(e' - e) \in \mathbb{Z}$. Let $\mu \leftarrow \tilde{g}^{m^\dagger} \tilde{h}^{r^\dagger} / \tilde{a}_1$. W.l.o.g., assume $e' > e$. By Eq. (2), $\mu^{e'-e} = (\tilde{g}^{m^\dagger} \tilde{h}^{r^\dagger} / \tilde{a}_1)^{e'-e} = \tilde{g}^{z_2'-z_2} \tilde{h}^{z_4'-z_4} / \tilde{a}_1^{e'-e} = 1$. Since $e' - e < C(\lambda)$ then $\mathrm{ord}(\mu)$ is $C(\lambda)$-smooth and hence $\mu^{l_{\mathbb{G}}} = 1$. Thus, we can open $\tilde{a}_1$ to $(m^\dagger, r^\dagger, \mu)$.

Since $z_2 < T \cdot C(\lambda)(2^\lambda + 1) < 2^{2\lambda+1} p$ by the last verification equation (Item 4e), we get that $|m^\dagger| < 2^{2\lambda+1} p < N$.

Next, assume that (iii) holds and consider Eq. (1). Since N and $e' - e \in [-2^\lambda + 1 \,..\, 2^\lambda - 1]$ are coprime, there exist integers α and β, such that $\alpha N + \beta(e' - e) = 1$. Let $r_2 \leftarrow \mathsf{c}_2^\alpha (z_3'/z_3)^\beta \mod N^2$. Thus, due to Eq. (1), $\mathsf{c}_2^{1-\alpha N} = \mathsf{c}_2^{\beta(e'-e)} \equiv (1 + N)^{\beta(z_2'-z_2)} (z_3'/z_3)^{\beta N} \pmod{N^2}$, and thus $\mathsf{c}_2 \equiv (1 + N)^{\beta(z_2'-z_2)} r_2^N \pmod{N^2}$. Clearly, $\beta(z_2' - z_2) = \beta(e' - e) m^\dagger$ as an integer. Thus, due to the definition of β, $\beta(z_2' - z_2) = \beta(e' - e) m^\dagger = (1 - \alpha N) m^\dagger \equiv m^\dagger \pmod N$ and thus $\mathsf{c}_2 \equiv (1+N)^{m^\dagger} r_2^N \pmod{N^2}$. Since directly by the definition of m^* and $m^\dagger$, $m^* \equiv m^\dagger \pmod p$, we get that c_1 and c_2 encrypt the same element m^* modulo p.

SSHVZK: $\Sigma_{\mathsf{eq}}.\mathsf{Sim}(\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{V}}; \mathsf{x}, e)$ sets $s_1 \leftarrow\!\!\$\, \mathbb{Z}_{2^{B+\lambda}}$, $s_5 \leftarrow\!\!\$\, [0 \,..\, C(\lambda) 2^{B+2\lambda} - 1]$, $z_1 \leftarrow\!\!\$\, \mathbb{Z}_p$, $z_2 \leftarrow\!\!\$\, \mathbb{Z}_{2^{2\lambda} p}$ (thus, Σ_{eq} is statistically but not perfectly zero knowledge), $z_3 \leftarrow\!\!\$\, \mathbb{Z}_{N^2}^*$, $z_4 \leftarrow s_1 e + s_5$, $\tilde{a}_1 \leftarrow \mathsf{DF}.\mathsf{Com}(\mathsf{ck}_{\mathsf{V}}; 0; s_1)$ (this is indistinguishable from a commitment to m since DF is statistical hiding), $\boldsymbol{a}_2 \leftarrow ((g^{z_1}, g^{em} h^{z_1}, (cd^\tau)^{z_1}) / \mathsf{c}_1)^\top$, $a_3 \leftarrow g^{z_2 - em}$, $a_4 \leftarrow (1 + z_2 N) z_3^N \mathsf{c}_2^{-e} \mod N^2$, $\tilde{a}_5 \leftarrow \tilde{g}^{z_2} \tilde{h}^{z_4} \tilde{a}_1^{-e}$. The simulator outputs $(a, \boldsymbol{z})$. The claim follows. □

3.2 New DL-Extractable UC-Commitment Scheme

The following DL-extractable UC-commitment scheme Γ_{dl} (see Fig. 5) is similar to Fujisaki's UC-commitment scheme Fuj [15], with the following two key differences. (i) Based on our observation that the CRS of Fuj can be divided into two parts, one guaranteeing binding and the second one needed to guarantee hiding, we redefine it in the (weaker) RPK model while originally Fuj relies on the CRS model. Importantly, the RPK model can also be used after the modification in the next step. (ii) We replace the Σ-protocol (a proof of the knowledge of the SCS-encrypted message g^m) from [15] with Σ_{eq}, interpreted as *the proof of knowledge of the discrete logarithm m of the* SCS*-encrypted message.* As explained above, Σ_{eq} achieves this by additionally encrypting m by using Pai; hence, the UC simulator, knowing the secret key $\mathsf{Pai.sk}$, decrypts c_2 to get m, and returns $m \bmod p$. (See the beginning of Section 3 for a longer intuition behind the construction of Γ_{dl}.)

Due to this, if one assumes the security of Σ_{eq} then the security proof of Γ_{dl} is similar to that given in [15]. Hence, we refer the reader to [15] for any

additional intuition about Fujisaki's commitment scheme. While the description of Γ_{dl} in Fig. 5 looks long, it is mainly so because of the use of three different encryptions/commitments which means that certain steps in the Fujisaki's commitment scheme are tripled.

We divide the public key rpk_i of $\mathcal{G}_i$ in Γ_{dl} into the binding part (used when $\mathcal{G}_i$ acts as the receiver R) and the hiding part (used when $\mathcal{G}_i$ acts as the committer C). C and R use $\mathsf{rpk}^{\mathsf{h}}_{\mathsf{C}} = (\mathsf{Pai.pk}_{\mathsf{C}} = N, \mathsf{SCS.pk}_{\mathsf{C}} = (g, h, c, d, \mathsf{H}^{\mathsf{h}}_{\mathsf{C}}))$ from C's public key $\mathsf{rpk}_{\mathsf{C}}$ and $\mathsf{rpk}^{\mathsf{b}}_{\mathsf{R}} = (\mathsf{Ped.ck}_{\mathsf{R}}, \mathsf{DF.ck}_{\mathsf{R}}, \mathsf{H}^{\mathsf{b}}_{\mathsf{R}})$ from R's public key $\mathsf{rpk}_{\mathsf{R}}$. Obviously, C knows $\mathsf{rpk}_{\mathsf{C}}$ while she has to retrieve $\mathsf{rpk}_{\mathsf{R}}$ from $\mathcal{R}_{\mathsf{R}}$.

See Fig. 5 for the full description of Γ_{dl}. Here, $\Gamma_{\mathsf{dl}}.\mathsf{Gen}$ for party $\mathcal{G}_i \in \{\mathsf{C}, \mathsf{R}\}$ is executed by the key registration authority $\mathcal{R}_i$ as usual in the RPK model, $\Gamma_{\mathsf{dl}}.\mathsf{Com}$ and $\Gamma_{\mathsf{dl}}.\mathsf{Open}$ are executed by C, and $\Gamma_{\mathsf{dl}}.\mathsf{Vf}$ is executed by R. The algorithms $\Gamma_{\mathsf{dl}}.\mathsf{tdOpen}$ and $\Gamma_{\mathsf{dl}}.\mathsf{Ext}$ are only executed within the security proof. To get straight-line simulation, we use the same method as [15]. Finally, note we have included $(\mathsf{lbl}, \mathsf{c}_3, e)$ to op mainly to simplify the notation.

Theorem 2. *Assume that* SCS *is an IND-PCA secure and* Pai *is an IND-CPA secure additively homomorphic cryptosystem,* Ped *is a computationally binding and perfectly hiding trapdoor commitment scheme and* DF *is a computationally binding and statistically hiding ICS. Assume secure erasure. Then* Γ_{dl} *from Fig. 5 UC-realizes* $\mathcal{F}_{\mathsf{mcomdl}}$ *in the* $\mathcal{F}_{\mathsf{rpk}}$*-hybrid model against adaptive attackers, i.e., it is a secure DL-extractable UC-commitment scheme in the RPK model.*

The proof of Theorem 2 follows closely the security proof of Fujisaki's UC-commitment scheme [15], with a few notable differences (the use of the RPK model instead of the CRS model, and the use of a different Σ-protocol, which causes us to use one more game to handle Paillier encryption).

Proof. As usual, we consider a sequence of hybrid games in which we change the rules of games step by step. We denote the changes by using gray background.

$\boxed{\mathsf{Game}_0 = \textbf{HYBRID}^{\mathcal{F}_{\mathsf{rpk}}}\textbf{:}}$ This is the *real world* game in the RPK model ($\textsc{Hybrid}^{\mathcal{F}_{\mathsf{rpk}}}$). In Game_0, the real protocol is executed between the committer C and the receiver R. The environment $\mathcal{Z}$ adaptively chooses the input for honest committer C and receives the output of honest parties. Adversary $\mathcal{A}$ attacks the real protocol in the real world, i.e., she can see the interactions between the honest parties or interact with the honest parties as playing the role of some parties after they are corrupted. When a party is corrupted, $\mathcal{A}$ can read her current inner state and $\mathcal{A}$ also fully controls her. $\mathcal{Z}$ can control $\mathcal{A}$ and see the inside of the execution of the protocol (the interactions between the honest parties or between the honest parties and the adversary) via the view of $\mathcal{A}$.

$\boxed{\mathsf{Game}_1\textbf{:}}$ In Game_1, Sim simulates the authorities $\mathcal{R}_{\mathsf{C}}, \mathcal{R}_{\mathsf{R}}$ generating the registered public keys $\mathsf{rpk}_{\mathsf{C}}$ and $\mathsf{rpk}_{\mathsf{R}}$ used by C and R. Sim stores $\mathsf{td}_{\mathsf{CR}} = (\mathsf{td}^{\mathsf{h}}_{\mathsf{C}}, \mathsf{td}^{\mathsf{b}}_{\mathsf{R}})$. Sim simulates honest parties as in Game_0, **except for the case where** R **is honest but** C **is corrupted**. After obtaining $(\mathsf{lbl}, \mathsf{c}_3; e; \mathsf{c})$ from the view of the protocol between C and R in the commit phase, where $\mathsf{lbl} = (\mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R})$, Sim

$\Gamma_{\mathsf{dl}}.\mathsf{Gen}(1^\lambda)$: Generate new keys $(\mathsf{SCS.pk}, \mathsf{SCS.sk})$ for SCS, $(\mathsf{Pai.pk}, \mathsf{Pai.sk})$ for Pai, $(\mathsf{DF.ck}, \mathsf{DF.td})$ for DF, and $(\mathsf{Ped.ck}, \mathsf{Ped.td})$ for Ped. Choose collision-resistant hash functions $\mathsf{H}^{\mathsf{h}}, \mathsf{H}^{\mathsf{b}}$. Let $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$. Let $\mathsf{rpk} = (\mathsf{rpk}^{\mathsf{h}}, \mathsf{rpk}^{\mathsf{b}})$ where $\mathsf{rpk}^{\mathsf{h}} = (\mathsf{p}, \mathsf{SCS.pk}, \mathsf{Pai.pk}, \mathsf{H}^{\mathsf{h}})$ and $\mathsf{rpk}^{\mathsf{b}} = (\mathsf{p}, \mathsf{Ped.ck}, \mathsf{DF.ck}, \mathsf{H}^{\mathsf{b}})$. Let $\mathsf{td} = (\mathsf{td}^{\mathsf{h}}, \mathsf{td}^{\mathsf{b}})$, where $\mathsf{td}^{\mathsf{h}} = (\mathsf{SCS.sk}, \mathsf{Pai.sk})$ and $\mathsf{td}^{\mathsf{b}} = (\mathsf{Ped.td}, \mathsf{DF.td})$.
Return $(\mathsf{rpk}, \mathsf{td})$. // The equivocability td is Ped.td; The extraction td is Pai.sk;

$\Gamma_{\mathsf{dl}}.\mathsf{Com}(\mathsf{rpk}_{\mathsf{C}}; \mathsf{lbl}, m)$ **where** $\mathsf{lbl} = (\mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R})$**:** to commit to $m \in \mathbb{Z}_p$ for R upon receiving $(\mathtt{commit}, \mathsf{lbl}, m)$, C does the following.

1. Obtain $\mathsf{rpk}_{\mathsf{R}} = (\mathsf{rpk}^{\mathsf{h}}_{\mathsf{R}}, \mathsf{rpk}^{\mathsf{b}}_{\mathsf{R}})$ from $\mathcal{R}_{\mathsf{R}}$;
   $\mathsf{ck}_{\mathsf{CR}} \leftarrow (\mathsf{rpk}^{\mathsf{h}}_{\mathsf{C}}, \mathsf{rpk}^{\mathsf{b}}_{\mathsf{R}})$; $\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{R}} \leftarrow \mathsf{DF.ck}_{\mathsf{R}}$;
   $r_1 \leftarrow\!\!\$\, \mathsf{RND}(\mathsf{SCS})$; $r_2 \leftarrow\!\!\$\, \mathsf{RND}(\mathsf{Pai})$;
   $\mathbf{c}_1 \leftarrow \mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk}_{\mathsf{C}}}(g^m; r_1)$; $\mathsf{c}_2 \leftarrow \mathsf{Pai.Enc}_{\mathsf{Pai.pk}_{\mathsf{C}}}(m; r_2)$;
   $\mathsf{c} \leftarrow (\mathbf{c}_1, \mathsf{c}_2)$;
   $\mathsf{x} \leftarrow (\mathsf{p}, \mathsf{SCS.pk}_{\mathsf{C}}, \mathsf{Pai.pk}_{\mathsf{C}}, g^m, \mathsf{c}, \mathsf{lbl})$; $\mathsf{w} \leftarrow (m, r_1, r_2)$;
   $\boldsymbol{s} \leftarrow\!\!\$\, \mathsf{RND}(\Sigma_{\mathsf{eq}})$; $a \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{P}_1(\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{R}}; \mathsf{x}, \mathsf{w}; \boldsymbol{s})$; $h_{\mathsf{x}} \leftarrow \mathsf{H}^{\mathsf{b}}_{\mathsf{R}}(\mathsf{lbl}, \mathsf{x}, a)$;
   (*) $r_3 \leftarrow\!\!\$\, \mathsf{RND}(\mathsf{Ped})$; $\mathsf{c}_3 \leftarrow \mathsf{Ped.Com}(\mathsf{Ped.ck}_{\mathsf{R}}; h_{\mathsf{x}}; r_3)$;
   Send $(\mathsf{lbl}, \mathsf{c}_3)$ to R;
2. After obtaining $(\mathsf{lbl}, \mathsf{c}_3)$, R fetches $\mathsf{rpk}_{\mathsf{C}} = (\mathsf{rpk}^{\mathsf{h}}_{\mathsf{C}}, \mathsf{rpk}^{\mathsf{b}}_{\mathsf{C}})$ from $\mathcal{R}_{\mathsf{C}}$, sets $\mathsf{ck}_{\mathsf{CR}}$ as above, and checks that c_3 is a valid ciphertext. If yes, he sets $e \leftarrow\!\!\$\, \{0,1\}^\lambda$ and sends (lbl, e) to C. Otherwise, R ignores it.
3. After receiving (lbl, e), C does the following.
   $\boldsymbol{z} \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{P}_2(\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{R}}; \mathsf{x}, \mathsf{w}; e; \boldsymbol{s})$;
   Securely delete $(\mathsf{w} = (m, r_1, r_2), \boldsymbol{s})$;
   $\mathsf{op} \leftarrow (\mathsf{lbl}, \mathsf{c}_3, e; a, \boldsymbol{z}, r_3)$; Store $\mathsf{st}_{\mathsf{C}} = (\mathsf{c}, g^m, \mathsf{op})$;
   Output $(\mathsf{c}, \mathsf{op})$ (privately); Send $(\mathtt{com}, \mathsf{lbl}, \mathsf{c})$ to R;
4. R checks that $\mathsf{c} = (\mathbf{c}_1, \mathsf{c}_2) \in \mathbb{G}^3 \times \mathbb{Z}_{N^2}$. If yes, R outputs $(\mathtt{rcpt}, \mathsf{lbl})$, and stores $\mathsf{st}_{\mathsf{R}} \leftarrow (\mathsf{lbl}, \mathsf{c}_3, e, \mathsf{c} = (\mathbf{c}_1, \mathsf{c}_2))$. Otherwise, R ignores it.

$\Gamma_{\mathsf{dl}}.\mathsf{Open}(\mathsf{st}_{\mathsf{C}})$**:** upon receiving $(\mathtt{open}, \mathsf{sid}, \mathsf{cid})$, C sends (g^m, op) to R.

$\Gamma_{\mathsf{dl}}.\mathsf{Vf}(\mathsf{ck}_{\mathsf{CR}}; \mathsf{c}, g^m, \mathsf{op})$ **where** $\mathsf{c} = (\mathbf{c}_1, \mathsf{c}_2)$**:** upon receiving $(g^m, \mathsf{op} = (\mathsf{lbl}, \mathsf{c}_3, e; a, \boldsymbol{z}, r_3))$ where $\mathsf{lbl} = (\mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R})$, R does the following.

1. $\mathsf{x} \leftarrow (\mathsf{p}, \mathsf{SCS.pk}_{\mathsf{C}}, \mathsf{Pai.pk}_{\mathsf{C}}, g^m, \mathsf{c}, \mathsf{lbl})$; $h_{\mathsf{x}} \leftarrow \mathsf{H}^{\mathsf{b}}_{\mathsf{R}}(\mathsf{lbl}, \mathsf{x}, a)$;
2. If cid has not been used with C, $(\mathsf{lbl}, \mathsf{c}_3, e)$ are the same as in the commit phase, $\mathsf{c}_3 = \mathsf{Ped.Com}(\mathsf{Ped.ck}_{\mathsf{R}}; h_{\mathsf{x}}; r_3)$, and $\Sigma_{\mathsf{eq}}.\mathsf{Vf}(\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{R}}; \mathsf{x}; a, e, \boldsymbol{z}) = 1$, then output $(\mathtt{open}, \mathsf{lbl}, g^m)$. Otherwise, ignore the message.

$\Gamma_{\mathsf{dl}}.\mathsf{tdOpen}(\mathsf{rpk}_{\mathsf{C}}, \mathsf{td}^{\mathsf{b}}_{\mathsf{R}}; \mathsf{c}, g^m, \mathsf{op}, g^{m'})$:
$\mathsf{x} \leftarrow (\mathsf{p}, \mathsf{SCS.pk}_{\mathsf{C}}, \mathsf{Pai.pk}_{\mathsf{C}}, g^m, \mathsf{c}, \mathsf{lbl})$; $\mathsf{x}' \leftarrow (\mathsf{p}, \mathsf{SCS.pk}_{\mathsf{C}}, \mathsf{Pai.pk}_{\mathsf{C}}, g^{m'}, \mathsf{c}, \mathsf{lbl})$;
$(a', \boldsymbol{z}') \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{Sim}(\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{R}}; \mathsf{x}', e)$;
$\mathsf{op}' \leftarrow \mathsf{Ped.tdOpen}(\mathsf{Ped.td}_{\mathsf{R}}; \mathsf{H}^{\mathsf{b}}_{\mathsf{R}}(\mathsf{lbl}, \mathsf{x}, a), r_3, \mathsf{H}^{\mathsf{b}}_{\mathsf{R}}(\mathsf{lbl}, \mathsf{x}', a'))$;
return op';

$\Gamma_{\mathsf{dl}}.\mathsf{Ext}(\mathsf{td}^{\mathsf{h}}_{\mathsf{C}}; \mathsf{c})$**:** Return $\mathsf{Pai.Dec}_{\mathsf{Pai.sk}}(\mathsf{c}_2) \mod p$;

Fig. 5: The commitment scheme Γ_{dl} in the RPK model

stores $m^* \leftarrow \mathsf{Pai.Dec}_{\mathsf{Pai.sk}_{\mathsf{C}}}(\mathsf{c}_2)$ as a part of the state. In the open phase, when C successfully opens to g^m, Sim sends $\left(\mathtt{open}, \mathsf{lbl}, g^{m^*}\right)$ to $\mathcal{Z}$.

In the case of adaptive corruption of R before the open phase, Sim simply reveals $\mathsf{st}_{\mathsf{C}} = (\mathsf{c}, g^{m^*}, \mathsf{op})$ to $\mathcal{A}$. Honest R has no secret. The proof of Lemma 1 is given in Appendix A.

Lemma 1. *If* Σ_{eq} *is specially sound,* Ped *is computationally binding, and* $\mathsf{H}^{\mathsf{b}}_{\mathsf{R}}$ *is collision-resistant then* $\mathcal{Z}$ *distinguishes* Game_0 *and* Game_1 *with a negligible probability.*

$\boxed{\mathsf{Game}_2}$**:** identical to Game_1 except following cases.

Honest C**:** In the open phase, upon receiving $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ from $\mathcal{Z}$, Sim sets $(a^*, e, \boldsymbol{z}^*) \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{Sim}(\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{R}}; \mathsf{x}, e)$ and sends $(g^m, \mathsf{op} = (\mathsf{lbl}, \mathsf{c}_3, e; a^*, \boldsymbol{z}^*, r_3))$ to R; *Importantly, in the simulation of honest* C *in the open phase,* Sim *does not have to know* w.

C **was adaptively corrupted before receiving** e**:** in the commit phase, Sim sets $r_3^* \leftarrow \mathsf{Ped.tdOpen}(\mathsf{Ped.td}_{\mathsf{R}}; h_{\mathsf{x}}, r_3, h_{\mathsf{x}}^*)$ and then reveals the current secret state $(\mathsf{w} = (m, r_1, r_2), \boldsymbol{s}, r_3^*)$ to $\mathcal{Z}$.

In the case of adaptive corruption of C **after receiving** e **but before the open phase:** Sim simulates C honestly. Note that $(\mathsf{w}, \boldsymbol{s})$ is supposed to be erased by honest C before sending c, and thus, Sim does not need to reveal it. The proof of the following lemma is straightforward.

Lemma 2. *If* Σ_{eq} *is SHVZK and* Ped *is trapdoor, then* $\mathcal{Z}$ *distinguishes* Game_1 *and* Game_2 *with negligible probability.*

$\boxed{\mathsf{Game}_3}$**:** In this game, we do the following changes.

Honest C**:** In the commit phase, after receiving $(\mathtt{commit}, \mathsf{lbl}, m)$ from $\mathcal{Z}$, when it receives e, Sim computes $\mathbf{c}_1^* \leftarrow \mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk}_{\mathsf{C}}}(1; r_1)$ and sends $\mathsf{c}^* \leftarrow (\mathbf{c}_1^*, \mathsf{c}_2)$ to R. In the open phase, upon receiving input $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ from $\mathcal{Z}$, Sim first sets $\mathsf{x}^* \leftarrow (\mathsf{p}, \mathsf{SCS.pk}_{\mathsf{C}}, \mathsf{Pai.pk}_{\mathsf{C}}, g^m, \mathsf{c}^*, \mathsf{lbl})$ where $\mathsf{x}^* \notin \mathbf{L}_{\mathsf{eq}}$ because $\mathbf{c}_1^* = \mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk}_{\mathsf{C}}}(1; r_1)$.

In the case of adaptive corruption of C**:** Sim simulates C as in Game_2.

Security analysis. The only difference from the previous game is that in Game_3, the simulator Sim (playing as honest C) computes $\mathbf{c}_1^* \leftarrow \mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk}_{\mathsf{C}}}(1; r_1)$ encrypting 1 instead of g^m. As in [15], we run the (multi-message) IND-PCA game to show this game is indistinguishable from the previous game. Denote by bad_i the event in Game_i that $m^* \not\equiv m \pmod p$ where m is the value successfully opened by C. As analysed above, $\Pr[\mathsf{bad}] = \Pr[\mathsf{bad}_1] = \mathsf{negl}(\lambda)$. In addition, Game_1 is statistically close to Game_2 and so, $\Pr[\mathsf{bad}_1] \approx \Pr[\mathsf{bad}_2] = \mathsf{negl}(\lambda)$. In Appendix B, we use this fact to prove the following lemma.

Lemma 3. *If* SCS *is IND-PCA secure then* $\mathcal{Z}$ *distinguishes* Game_2 *and* Game_3 *with only a negligible probability.*

$\boxed{\mathsf{Game}_4}$**:** In this game, Sim enacts the following changes compared to Game_3.

If C **is honest:** upon receiving input $(\mathtt{commit}, \mathsf{lbl}, m)$ from $\mathcal{Z}$, after receiving e, Sim computes $\mathsf{c}_2^* = \mathsf{Pai.Enc}_{\mathsf{Pai.pk}_{\mathsf{C}}}(0; r_2)$ and returns $\mathsf{c}^* \leftarrow (\mathbf{c}_1^*, \mathsf{c}_2^*)$ to R.

In the open phase, upon receiving input $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ from $\mathcal{Z}$, Sim first sets $\mathsf{x} = (\mathsf{p}, \mathsf{SCS.pk_C}, \mathsf{Pai.pk_C}, g^m, \mathsf{c}, \mathsf{lbl})$ where $\mathsf{x} \notin \mathbf{L}_{\mathsf{eq}}$ because $\mathbf{c}_1^* = \mathsf{SCS.Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk_C}}(1; r_1)$ and $\mathsf{c}_2^* = \mathsf{Pai.Enc}_{\mathsf{Pai.pk_C}}(0; r_2)$.

In the case of adaptive corruption of C: Sim simulates C identically as in Game_3.

Security analysis. The only difference from Game_3 is that in Game_4, the simulator Sim (playing as honest C) computes $\mathsf{c}_2^* \leftarrow \mathsf{Pai.Enc}_{\mathsf{Pai.pk_C}}(0; r_2)$ instead of $\mathsf{c}_2 \leftarrow \mathsf{Pai.Enc}_{\mathsf{Pai.pk_C}}(m; r_2)$. We run the (multi-message) IND-CPA game to show Game_4 is indistinguishable from Game_3.

Lemma 4. *If* Pai *is IND-CPA secure then* $\mathcal{Z}$ *distinguishes* Game_3 *and* Game_4 *with only a negligible probability.*

Proof. The proof is a variation of the proof of Lemma 3. We now analyse Pai, and define CPA-related security games (like mIND-CPA) instead of PCA-related security games. □

$\boxed{\mathsf{Game}_5\text{:}}$ In the ideal world, there additionally exists an ideal functionality $\mathcal{F}_{\mathsf{mcomdl}}$ and the task of the honest parties in the ideal world is simply to convey inputs from $\mathcal{Z}$ to the ideal functionalities and vice versa (the ideal honest parties communicate only with $\mathcal{Z}$ and the ideal functionalities).

Initialization step: Sim generates $\mathsf{rpk_R}$ and $\mathsf{rpk_C}$ and saves the trapdoors.
Simulating the communication with $\mathcal{Z}$: Every input value that Sim 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 C is honest: Upon receiving $(\mathtt{rcpt}, \mathsf{lbl} = (\mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}))$ from $\mathcal{F}_{\mathsf{mcomdl}}$, Sim sets $m^* \leftarrow 0$ and uses it instead of m in what follows. Unless explicitly said otherwise, we will denote any variable X that uses m^* instead of m as X^* without making all the details explicit. For example, $\mathbf{c}_1^* \leftarrow \mathit{\Pi}.\mathsf{Enc}^{\mathsf{lbl}}_{\mathsf{SCS.pk_R}}(g^{m^*}; r_1)$, $\mathsf{c}^* \leftarrow (\mathbf{c}_1^*, \mathsf{c}_2)$, $a^* \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{P}_1(\Sigma_{\mathsf{eq}}.\mathsf{rpk_R}; \mathsf{x}^*, \mathsf{w}^*; \boldsymbol{s})$, $h_{\mathsf{x}}^* \leftarrow \mathsf{H}^{\mathsf{b}}_{\mathsf{R}}(\mathsf{lbl}, \mathsf{x}^*, a^*)$, and Sim reveals $(\mathsf{lbl}, \mathsf{c}_3^*)$.
Simulating the commit phase when C is corrupted and R is honest: After receiving $(\mathsf{lbl}, \mathsf{c}_3, e, \mathsf{c})$ from C in the commit phase, Sim sets $m^* \leftarrow \mathsf{Pai.Dec}_{\mathsf{Pai.sk_C}}(\mathsf{c}_2)$ and uses m^* instead of m after that.

Simulating adaptive corruption of C before receiving e in the commit phase: When C is corrupted, Sim immediately reads C's inner state and obtains m. After that, Sim uses m to compute all variables as in the real protocol, except that setting $r_3 \leftarrow \mathsf{Ped.tdOpen}(\mathsf{Ped.td_R}; h_{\mathsf{x}}^*, r_3^*, h_{\mathsf{x}})$ and revealing $(m, \mathsf{w}, \boldsymbol{s}, r_3)$.
Simulating adaptive corruption of C after the commit phase but before the open phase: When C is corrupted, Sim immediately reads ideal committer C's inner state and obtains m. Sim sets all variables as in the real protocol, except $(a, \boldsymbol{z}) \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{Sim}(\Sigma_{\mathsf{eq}}.\mathsf{rpk_R}; \mathsf{x}, e)$. Sim sets $r_3 \leftarrow \mathsf{Ped.tdOpen}(\mathsf{Ped.td_R}; h_{\mathsf{x}}^*, r_3^*, h_{\mathsf{x}})$ and reveals $\mathsf{st_C}$.

Simulating adaptive corruption of R after the commit phase but before the open phase: Sim simply stores $\mathsf{st}_{\mathsf{R}} = (\mathsf{lbl}, \mathsf{c}_3, e, \mathsf{c})$ as if it comes from R.

Simulating the open phase when C is honest: Upon receiving input $(\mathsf{open}, \mathsf{lbl}, g^m)$ from $\mathcal{F}_{\mathsf{mcomdl}}$, Sim uses g^m to compute all variables as in the real protocol, except $(a, \boldsymbol{z}) \leftarrow \Sigma_{\mathsf{eq}}.\mathsf{Sim}(\Sigma_{\mathsf{eq}}.\mathsf{rpk}_{\mathsf{R}}; \mathsf{x}, e)$. Sim sets $r_3 \leftarrow \mathsf{Ped.tdOpen}(\mathsf{Ped.td}_{\mathsf{R}}; h_{\mathsf{x}}^*, r_3^*, h_{\mathsf{x}})$. Sim reveals (g^m, op).
Simulating the open phase when C is corrupted and the receiver R is honest: Upon receiving (g^m, op) from C, Sim sends $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ to $\mathcal{F}_{\mathsf{mcomdl}}$. $\mathcal{F}_{\mathsf{mcomdl}}$ follows its code: if a tuple $(\mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}, g^{m^*})$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously stored by $\mathcal{F}_{\mathsf{mcomdl}}$, $\mathcal{F}_{\mathsf{mcomdl}}$ sends $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}, g^{m^*})$ to the ideal receiver R and Sim. Then, R conveys it to $\mathcal{Z}$.

By construction, this game is identical to the previous game. □

Acknowledgement. The authors were supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780477 (project PRIViLEDGE), and by the Estonian Research Council grant (PRG49). The work was mostly done while Zając was working at the University of Tartu.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: PKC 2015. LNCS, vol. 9020, pp. 332–352
2. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: 45th FOCS, pp. 186–195
3. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: TCC 2008. LNCS, vol. 4948, pp. 356–374
4. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure sampling of public parameters for succinct zero knowledge proofs. In: 2015 IEEE Symposium on Security and Privacy, pp. 287–304
5. Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Analysis and improvement of Lindell's UC-secure commitment schemes. In: ACNS 13. LNCS, vol. 7954, pp. 534–551
6. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC, pp. 103–112
7. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: CRYPTO 2003. LNCS, vol. 2729, pp. 126–144
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145
9. Canetti, R., Fischlin, M.: Universally composable commitments. In: CRYPTO 2001. LNCS, vol. 2139, pp. 19–40
10. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, pp. 494–503

11. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: CRYPTO'94. LNCS, vol. 839, pp. 174–187
12. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142
13. Damgård, I., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: CRYPTO 2002. LNCS, vol. 2442, pp. 581–596
14. Fischlin, M., Libert, B., Manulis, M.: Non-interactive and re-usable universally composable string commitments with adaptive security. In: ASIACRYPT 2011. LNCS, vol. 7073, pp. 468–485
15. Fujisaki, E.: Improving practical UC-secure commitments based on the DDH assumption. In: SCN 16. LNCS, vol. 9841, pp. 257–272
16. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432
17. Kosba, A.E., Zhao, Z., Miller, A., Qian, Y., Chan, T.H., Papamanthou, C., Pass, R., Shelat, A., Shi, E.: CØCØ: A Framework for Building Composable Zero-Knowledge Proofs. Technical Report 2015/1093, IACR (2015) `http://eprint.iacr.org/2015/1093`, last accessed version from 9 Apr 2017.
18. Lindell, Y.: Highly-efficient universally-composable commitments based on the DDH assumption. In: EUROCRYPT 2011. LNCS, vol. 6632, pp. 446–466
19. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238
20. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO'91. LNCS, vol. 576, pp. 129–140

A Proof of Lemma 1

Proof. The only difference from Game_0 is that in Game_1, Sim (playing as honest R) outputs g^{m^*} instead of the value g^m at the open phase. Note that Sim opens g^{m^*} after C decommits to g^m in a verifiable way. If not, Sim outputs nothing. We denote by bad the event that $m^* \not\equiv m \pmod p$ where g^m is the value successfully opened by C. We claim that bad occurs only with a negligible probability; otherwise, either the soundness of Σ_{eq}, the binding of Ped, or the collision resistance of $\mathsf{H}^{\mathsf{b}}_{\mathsf{R}}$ is broken.

Assume that $m^* \not\equiv m \pmod p$ at least in one of such executions. In the first such execution, we rewind the adversary at the step (*) in the commit phase and send a new random challenge e'. Assume, by contradiction, that C returns $\mathsf{c}' = (\mathsf{c}'_1, \mathsf{c}'_2)$ such that $\mathsf{c}' \neq \mathsf{c}$ but still successfully decommits to some value m' with a'. Then it implies breaking of the binding of Ped or the collision-resistancy of $\mathsf{H}^{\mathsf{b}}_{\mathsf{R}}$, because we can simulate it without knowing the trapdoor key. For the same reason, $\mathsf{x}' = \mathsf{x}$ (and thus $m' = m$) holds except with a negligible probability. Thus, rewinding the commit phase, C outputs the same $\mathsf{st}_{\mathsf{C}} = (\mathsf{c}, g^m, \mathsf{op})$ except with a negligible probability when it can successfully decommit. Note that $m^* \not\equiv m \pmod p$ implies that $\mathsf{x} \notin \mathbf{L}_{\mathsf{eq}}$. Since x (and thus m) is now fixed with an overwhelming probability, C can convince R on false instance x only with probability $2^{-\lambda}$ (this follows from the special soundness of Σ_{eq}), which is negligible in λ. Hence, bad occurs only with a negligible probability and the views

of $\mathcal{Z}$ in the two games are computationally indistinguishable. We stress that we rewind just in the proof of binding, but not in the simulation. □

B Proof of Lemma 3

Proof. The proof is a variant of the proof in [15], App. A. We define the multi-message IND-PCA security for a public-key cryptosystem Π. Let $\mathsf{Expt}^{\mathsf{mpca}}_{\Pi,\mathcal{B}}(\lambda)$ be the following experiment:

$\mathsf{Expt}^{\mathsf{mpca}}_{\Pi,\mathcal{B}}(\lambda)$

$\mathcal{Q}_{\mathsf{Enc}} \leftarrow \emptyset; \mathcal{Q}_{\mathsf{pca}} \leftarrow \emptyset; (\Pi.\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi.\mathsf{KGen}(1^\lambda);$
$b \leftarrow_{\$} \{0,1\}; b' \leftarrow \mathcal{B}^{\mathsf{Enc}^b_{\Pi.\mathsf{pk}}(\cdot,\cdot,\cdot), O^{\mathsf{pca}}_{\Pi.\mathsf{sk}}(\cdot,\cdot,\cdot)}(\Pi.\mathsf{pk});$
if $b = b'$ **then return** 1; **else return** 0; **fi**

Here, the oracles are defined as follows:

- $\mathsf{Enc}^b_{\Pi.\mathsf{pk}}(\mathsf{lbl}^*, g^{m_0}, g^{m_1})$ rejects it if $\mathsf{lbl}^* \in \mathcal{Q}_{\mathsf{pca}}$. Otherwise, it adds lbl^* to $\mathcal{Q}_{\mathsf{Enc}}$ and returns $\mathbf{c} \leftarrow \Pi.\mathsf{Enc}^{\mathsf{lbl}^*}_{\Pi.\mathsf{pk}}(g^{m_b})$.
- $O^{\mathsf{pca}}_{\Pi.\mathsf{sk}}(\mathsf{lbl}, g^m, \mathbf{c})$ rejects it if $\mathsf{lbl} \in \mathcal{Q}_{\mathsf{Enc}}$. Otherwise, it adds lbl to $\mathcal{Q}_{\mathsf{pca}}$, and returns 1 iff $\mathbf{c}$ is a proper ciphertext of g^m on label lbl.

Π is *multi-message indistinguishable against the plaintext checkable attacks* (mIND-PCA secure) if $\mathsf{Adv}^{\mathrm{mpca}}_{\Pi,\mathcal{B}}(\lambda) := |\Pr[\mathsf{Expt}^{\mathsf{mpca}}_{\Pi,\mathcal{B}}(\lambda) = 1] - 1/2| = \mathsf{negl}(\lambda)$ for all non-uniform PPT $\mathcal{B}$.

By using the standard hybrid argument, for any mIND-PCA adversary $\mathcal{B}$ against Π with at most $q = q(\lambda)$ queries to the encryption oracle, there exists an IND-PCA adversary $\mathcal{B}'$ against Π such that $\mathsf{Adv}^{\mathrm{mpca}}_{\Pi,\mathcal{B}}(\lambda) \leq q(\lambda) \cdot \mathsf{Adv}^{\mathrm{pca}}_{\Pi,\mathcal{B}'}(\lambda)$, where the running time of $\mathcal{B}'$ is roughly bounded by the running time of $\mathcal{B}$ plus $q-1$ encryption operations. We now construct mIND-PCA adversary $\mathcal{B}$ using $\mathcal{Z}$ and the adversary $\mathcal{A}$ as follows. W.l.o.g., we assume that $\Pr[\mathsf{Game}_2(\mathcal{A}) = 1] \leq \Pr[\mathsf{Game}_3(\mathcal{A}) = 1]$, where $\mathsf{Game}_i(\mathcal{A})$ is the random variable assigning the output bit of the environment $\mathcal{Z}$ in Game_i. $\mathcal{B}$ is given $\mathsf{SCS.pk}_{\mathsf{C}}$ as an instance in the mIND-CPA game. $\mathcal{B}$ sets up $\mathsf{rpk}_{\mathsf{C}}$ and $\mathsf{rpk}_{\mathsf{R}}$ by picking the remaining parameters. Here, she knows $\mathsf{Ped.td}_{\mathsf{R}}$ but does not know $\mathsf{SCS.sk}_{\mathsf{C}}$. $\mathcal{B}$ runs $\mathcal{Z}$ and $\mathcal{A}$ and plays the role of simulator Sim as in Game_2 (or Game_3), except for the following two cases:

(i) If C is honest and $\mathcal{A}$ receives $(\mathsf{lbl}, \mathsf{c}_3)$ from $\mathcal{Z}$, $\mathcal{B}$ submits $(\mathsf{lbl}, g^m, 1)$ to the oracle $\mathsf{Enc}^b_{\mathsf{SCS.pk}_{\mathsf{C}}}$ and receives $\mathbf{c}$. Then, $\mathcal{B}$ plays the role of the simulator in Game_2 (or equivalently, in Game_3).

(ii) If R is honest but C is corrupted, after receiving all three messages in the commit phase with C, $\mathcal{B}$ simply stores it. In the open phase, when C successfully decommits to g^m, $\mathcal{B}$ submits $(\mathsf{lbl}, g^m, \mathbf{c}_1)$ to the oracle $O^{\mathsf{pca}}_{\mathsf{sk}_{\mathsf{C}}}$ and receives the answer bit. If the answer bit is 1, then $\mathcal{B}$ outputs $(\mathsf{open}, \mathsf{lbl}, g^m)$ to the environment. **Otherwise, she halts and outputs 1 (break point).**

If such an event does not occur, $\mathcal{B}$ proceeds the game with $\mathcal{Z}$ and $\mathcal{A}$ as playing the role of Sim. Finally, $\mathcal{B}$ outputs bit b' that $\mathcal{Z}$ outputs, as the output of the mIND-PCA game.

Security analysis. Above, $\mathcal{B}$ perfectly simulates Game_2 when $b = 0$ just before the break point. Recall that bad_i denotes the event in Game_i that $m^* \not\equiv m \pmod{p}$ where g^m is the value successfully decommitted to by corrupted C. The probability that the break occurs is equal to the probability that bad_2 occurs, which is negligible. Similarly, $\mathcal{B}$ perfectly simulates Game_3 when $b = 1$ just before the break point. We do not know $\Pr[\mathsf{bad}_3]$. However, since $\Pr[\mathsf{bad}_2] = \mathsf{negl}(\lambda)$, we can conclude $b = 1$ if the break happens. If the break never happens, $\mathcal{B}$ perfectly simulates either Game_2 or Game_3 according to b. Thus, the difference of the output of $\mathcal{Z}$ is bounded by the advantage of $\mathcal{B}$: $\mathsf{Adv}^{\mathrm{mcpa}}_{\mathsf{SCS},\mathcal{B}}(\lambda) = |\Pr[\mathsf{Game}_3(\mathcal{Z}) = 1 \wedge \neg\mathsf{bad}_3] + \Pr[\mathsf{bad}_3] - (\Pr[\mathsf{Game}_2(\mathcal{Z}) = 1 \wedge \neg\mathsf{bad}_2] + \Pr[\mathsf{bad}_2])|$. Thus, $\Pr[\mathsf{Game}_3(\mathcal{Z}) = 1] - \Pr[\mathsf{Game}_2(\mathcal{Z}) = 1] \leq \mathsf{Adv}^{\mathrm{mcpa}}_{\mathsf{SCS},\mathcal{B}}(\lambda) + \Pr[\mathsf{bad}_2] - \Pr[\mathsf{Game}_2(\mathcal{Z}) = 1 \wedge \mathsf{bad}_2] \leq \mathsf{Adv}^{\mathrm{mcpa}}_{\mathsf{SCS},\mathcal{B}}(\lambda) + \mathsf{negl}(\lambda)$. $\square$