UC-Commitment Schemes with Phase-Adaptive Security from Trapdoor Functions

Pedro Branco* Manuel Goulão* Paulo Mateus*

Abstract

We propose a generic framework for perfectly hiding UC-Commitment schemes in the Global Random Oracle model of Canetti *el at.* (CCS 14). The main building block of our construction is a novel primitive called Sampleable-Range Trapdoor Function, that is, a trapdoor function for which there is a non-negligible probability of finding preimages when given a uniformly chosen element of its codomain and the corresponding trapdoor. To show the versatility of the framework, we give concrete instantiations based on factoring, code-based, and lattice-based hardness assumptions. Our construction yields the first lattice-based UC-Commitment scheme (not constructed via generic transformations, such as via Oblivious Transfer), and achieves what we call *phase-adaptive security*, a novel security notion we introduce which is stronger than static security.

Achieving adaptive security for UC-Commitment schemes is non-trivial and, usually, comes at the price of efficiency. Phase-adaptive security stands between adaptive and static security, and may be of independent interest. In this model, adversaries can corrupt at the beginning or between the commitment and opening phases of the protocol, but not during their execution. This new model is motivated by the fact that, in practice, it is more likely that parties are corrupted between phases of the protocol (where a relatively long period may elapse) than during their execution.

1 Introduction

A Commitment scheme is a simple, yet powerful, cryptographic primitive. It involves two parties, the committer C and the receiver R, and consists of two independent phases. In the commitment phase, C commits to a message and sends the corresponding commitment to R. Later, in the opening phase, C opens the commitment and reveals the message to R. Concerning security, two properties must hold: C cannot reveal a different message than the one it had committed to in the commitment phase (binding property), and R cannot get information about the message before the opening phase (hiding property).

*SQIG - IT, IST - University of Lisbon.
Email: {pmbranco,mgoulao,pmat}@math.tecnico.ulisboa.pt.

Commitment schemes mostly shine when used as building blocks to construct other, more complex, cryptographic primitives (e.g., signature schemes, zero-knowledge proofs, or secure multiparty computation) since they do not serve many useful purposes by themselves. Therefore, the security of Commitment schemes should be analyzed in the Universal Composability (UC) framework[Can01], which guarantees security even when composed with other protocols. Commitment schemes proven to be secure in this framework are usually called UC-Commitment schemes.

In this work, we present a framework for UC-Commitment schemes in the Global Random Oracle (gRO) model of Canetti, Jain and Scafuro [CJS14], a security model where only one Global Random Oracle is available for all executions of all protocols.[1] The framework makes use of a new flavor of trapdoor functions which we call Sampleable-Range Trapdoor Function. Also, it is secure against malicious phase-adaptive adversaries, a new class of adversaries that can corrupt between phases of the execution. The resulting generic construction is exceptionally versatile, as it can be instantiated not only with several common hardness assumptions (e.g., RSA, Trapdoor Discrete Log, Quadratic Residuosity) but also with many relevant post-quantum hardness assumptions, such as code-based and lattice-based assumptions. As far as we know, we obtain the first lattice-based UC-Commitment scheme, which is not constructed via generic transformations, such as via Oblivious Transfer (OT).

1.1 Previous work

The study of UC-Commitment schemes started with the work of Canetti and Fischlin [CF01] where they prove the impossibility of constructing UC-Commitment schemes in the plain model, and propose schemes in the Common Reference String (CRS) model. However, the schemes of [CF01] only allow users to commit to single bits.

After that, several UC-Commitment schemes in the CRS model were presented [DN02, DG03, HMQ04, Lin11, FLM11, CJS14, Fuj16, BPRS17], but all of these have their security based on number-theoretic assumptions.

Recently, a UC-Commitment in the gRO model was presented in [CJS14], based on the discrete logarithm assumption, and another in [Bra19], based on assumptions from coding theory.

In [CDG$^+$18], several alternative security models to the gRO model (or restricted Observable Random Oracle Model) were presented, such as the restricted Programmable Random Oracle or the restricted Programmable and Observable Random Oracle. In these security models, efficient UC-Commitment schemes were proposed.[2] However, we remark that these security models need

[1] This security model was also studied in [CDG$^+$18], where it is called restricted Observable Random Oracle.

[2] In [CDG$^+$18], it is proven that the folklore Commitment scheme, which consists in computing $\mathsf{H}(M, r)$ (where H is a random oracle, M is the message and r is a random string), is a UC-Commitment scheme in the restricted Programmable and Observable Random Oracle Model.

the simulator to have the capability of programming the random oracle, which differs from the original proposal and motivation given by Canetti *et al.* [CJS14]. While the most truthful random oracle would be the one for which the simulator is neither able to program nor to observe queries, so-called the strict Random Oracle in [CJS14, CDG$^+$18], unfortunately, such random oracle does not suffice to design UC-Commitment schemes [CF01, CJS14]. Therefore, by giving the simulator only the capability of observing adversarial queries, we aim to provide a UC security proof assuming the weakest conditions possible on a non-programmable global random oracle.

We remark that it is possible to achieve Commitment schemes from the OT primitive [Kil88, GIKW14, CDD$^+$16], and some UC OT protocols were proposed in the past (e.g., [PVW08]). However, we are not aware of any (fully-secure) OT scheme in the gRO model proposed in the literature.

2 Contributions of this work

We start by presenting a new type of adaptive security, which we call phase-adaptive security. Then, we schematically present our framework and explain how it can be instantiated with trapdoor functions based on several different hard problems.

2.1 Phase-adaptive security

Most works in literature consider either static or adaptive adversaries. The difference between these two types of adversaries is the moment when they are allowed to interfere in the protocol. While the first type corrupts parties at the beginning of the protocol, the latter may corrupt parties at any point, or even after its execution. The notion of adaptive adversary is quite powerful and, thus, it is more sparsely found in the literature.

In this work, we present a new notion of corruption in the UC-framework, which we call *phase-adaptive* corruption due to its characterization. Phase-adaptive adversaries are a particular case of adaptive adversaries that can corrupt parties at the beginning, but also after the beginning of the protocol (in opposition to static security). However, corruption is only allowed between *phases* of the protocol, and not at any point of the protocol (as in the adaptive case). With this new relaxed way of modeling adversaries, we aim to extend this binary notion (static/adaptive) to a more broad spectrum.

The phases should be defined *a priori* in the ideal functionality, and hence, they should also be defined in the protocol. In fact, when defining an ideal functionality, it is standard to explicitly label phases for sequential segments of the protocol.[3] These might be seen as different functionalities, which depend on the internal state of each other. For example, consider the case of a Commitment scheme. As explained above, a Commitment scheme is composed of two phases,

[3]Examples of such cases are the Commitment and the Commit-and-Prove functionalities defined in [CLOS02], or Joint Gate Evaluation defined in [GMY04].

a commitment phase, and an opening phase, and, clearly, the first one must precede the latter.

The intuition behind this new adversarial model is that, in practice, it is much more likely that a party gets corrupted between phases of the protocol (where a relatively long period of time may elapse), rather than during the execution of a phase (where the period of time between rounds, that is, the time between a message being exchanged from one party to the other, is relatively short). To see this, consider again the case of Commitment schemes: A committer C commits to a message M and, after some time (say several hours, days or even years), C wants to open M. It is more plausible that some party involved in the protocol gets corrupted between the commitment and opening phases rather than during the execution of any of them. Another relevant example is the one of Perfect Forward Secrecy, in the context of Key Exchange, where we want to prevent an adversary that corrupts the parties from obtaining previously established shared session keys. In fact, the notion of phase-adaptive security was already implicitly considered in [CK02], when discussing Perfect Forward Secrecy.

Remarkably, this type of adversaries was never explicitly considered in previous works. However, we believe that considering phase-adaptive adversaries is quite natural and intuitive.

Observe that, for protocols whose phases consist of solely one round, as the case of Non-Interactive Commitment schemes, the notions of adaptive and phase-adaptive security coincide.

2.2 Framework for UC-Commitment

We first define a new class of trapdoor functions, which we call Sampleable-Range Trapdoor Functions (SRTF). In a nutshell, an SRTF is a trapdoor function $F : \mathcal{X} \to \mathcal{Y}$ such that we are able to find preimages for a non-negligible fraction of elements in $\mathcal{Y}$. To this end, consider an SRTF F to be such that the density of the set $F(\mathcal{X})$ in $\mathcal{Y}$ is, at least, $1/p(\kappa)$ where $p(\kappa)$ is some polynomial and κ is the security parameter. This implies that, given an SRTF F, for a uniformly random element $y \in \mathcal{Y}$, there is a non-negligible probability of existing $x \in \mathcal{X}$ such that $F(x) = y$.

As our main result, we propose a framework for UC-Commitment schemes using SRTF. The framework is schematically presented in Figure 1 (commitment phase) and Figure 2 (opening phase).

A UC-Commitment scheme must be both extractable, meaning that the simulator must be able to extract the message in the commitment phase, and equivocal, i.e., the simulator must be able to open any message in the opening phase [CF01]. In our scheme, the simulator can extract the message M since a corrupted committer needs to query the gRO on an input containing M (and the simulator has access to adversarial queries). To equivocate, the simulator first extracts the trapdoor, which is queried to the gRO, and then, using the fact that F is an SRTF, it can find a valid opening for any message.

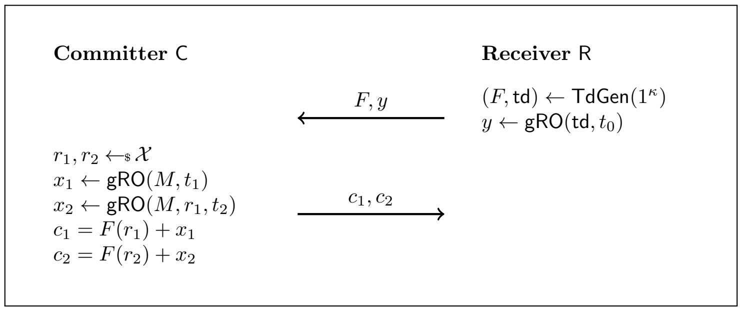
Committer C
Receiver R
$(F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa)$
$y \leftarrow \mathsf{gRO}(\mathsf{td}, t_0)$
F, y
$r_1, r_2 \leftarrow_\$ \mathcal{X}$
$x_1 \leftarrow \mathsf{gRO}(M, t_1)$
$x_2 \leftarrow \mathsf{gRO}(M, r_1, t_2)$
$c_1 = F(r_1) + x_1$
$c_2 = F(r_2) + x_2$
c_1, c_2

Figure 1: Commitment phase. The algorithm TdGen generates an SRTF $F : \mathcal{X} \to \mathcal{Y}$ along with the trapdoor td, M is the message C is committing to, gRO is the global random oracle. t_0, t_1, t_2 are random strings and κ denotes the security parameter.

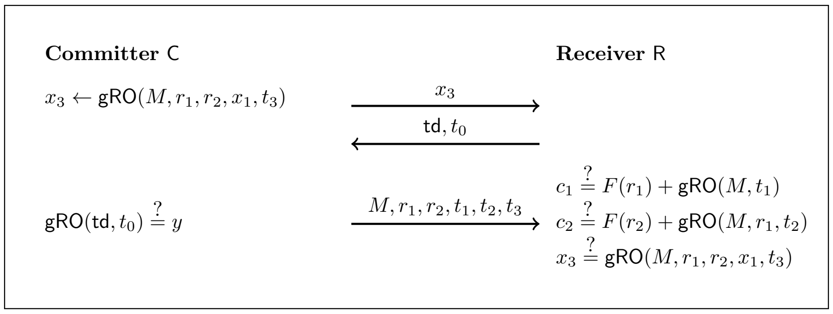
Committer C
Receiver R
$x_3 \leftarrow \mathsf{gRO}(M, r_1, r_2, x_1, t_3)$
x_3
td, t_0
$\mathsf{gRO}(\mathsf{td}, t_0) \overset{?}{=} y$
$M, r_1, r_2, t_1, t_2, t_3$
$c_1 \overset{?}{=} F(r_1) + \mathsf{gRO}(M, t_1)$
$c_2 \overset{?}{=} F(r_2) + \mathsf{gRO}(M, r_1, t_2)$
$x_3 \overset{?}{=} \mathsf{gRO}(M, r_1, r_2, x_1, t_3)$

Figure 2: Opening phase. t_3 is a random string. By $x \overset{?}{=} y$, we denote the experiment of testing if x is equal to y. If the test fails, the party running the experiment aborts the protocol.

Our framework can be viewed as a generalization of the UC-Commitment scheme of [CJS14]. However, using an SRTF in a generic way (instead of a trapdoor Pedersen Commitment scheme, as in [CJS14]) allows us to obtain UC-Commitment schemes from numerous hardness assumptions and to heavily improve on efficiency.

Concrete instantiations. We impose extremely weak conditions on the trapdoor function used in the generic construction. Hence, in order to show the versatility of our framework, we present multiple instantiations of our generic constructions using trapdoor functions based on different hardness assumptions.

We show how to instantiate the framework using standard factoring-based trapdoor functions such as the RSA cryptosystem [RSA78], discrete log trapdoor functions [PS09], and Rabin's cryptosystem. Moreover, we take advantage of SRTF, whose security is based on assumptions believed to be robust even against quantum adversaries. Particularly, we show constructions of UC-Commitment schemes from code-based and lattice-based assumptions (using the trapdoor functions of [CFS01] and [GPV08], respectively). As far as we know, our construction yields the first ever UC-Commitment whose security is based on lattice assumptions (thus, resisting quantum adversaries) in the gRO model.[4]

2.3 Open problems

Our generic construction for UC-Commitment cannot be proven secure against adaptive adversaries: Observe that, after the first round of the opening phase, it is impossible for a simulator to come up with a valid internal state for the committer. We leave as future work to design a generic construction using trapdoor functions that is adaptively secure.

As mentioned before, for protocols whose phases consist of solely one round, the notions of adaptive and phase-adaptive coincide. However, we conjecture that this does not hold in general. For example, our scheme cannot be proven secure against adaptive adversaries, despite the fact that we do not know an adaptive adversary that is able to break it. We leave as future work to prove the separation between these two types of adversaries.

3 Preliminaries

Throughout this work, κ denotes the security parameter. We abbreviate probabilistic polynomial-time algorithm as PPT. If A is an algorithm, $y \leftarrow \mathsf{A}(x)$ denotes running A on input x yielding y as output. If S is a finite set, $x \leftarrow_{\$} S$ denotes sampling x from S according to the uniform distribution and $|S|$ denotes its cardinality. A function is negligible in κ, written $\mathsf{negl}(\kappa)$, if it vanishes faster than $1/\mathsf{poly}(\kappa)$ where $\mathsf{poly}(\kappa)$ is any polynomial in κ.

We denote by

$$\mathsf{Pr}\left[A \;\middle|\; \begin{array}{c} B_1 \\ \vdots \\ B_n \end{array} \right]$$

the probability of event A given that events $B_1, \ldots, B_n$ happened sequentially.

Let X and Y be two probability distributions. By $X \approx Y$ we denote that X and Y are computationally indistinguishable.

3.1 Trapdoor functions

Definition 1 (Trapdoor Function)**.** A *Collection of Trapdoor Functions* is a tuple of algorithms $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ such that:

[4]As mentioned before, we are aware of several generic compilers for UC-Commitment schemes [Kil88, GIKW14, CDD^{+}16]. All of them use OT as a building block and we are not aware of any OT scheme secure in the gRO model.

- $\mathsf{TdGen}(1^\kappa)$ outputs a pair (F, td);
- For $x \in \mathcal{X}$, $\mathsf{Eval}(F, x)$ returns $y = F(x) \in \mathcal{Y}$ where $F : \mathcal{X} \to \mathcal{Y}$ is a trapdoor function. To ease the presentation, we will usually write $F(x)$ to denote the experiment of running $\mathsf{Eval}(F, x)$;
- For every $(F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa)$ and every $x \in \mathcal{X}$, $\mathsf{Invert}(\mathsf{td}, F, F(x))$ outputs $x' \in \mathcal{X}$ such that $F(x) = F(x')$, except with negligible probability.

For every PPT adversary $\mathcal{A}$, we say that the Collection of Trapdoor Functions $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ is *secure* if

$$\mathsf{Pr}\left[F(x) = F(x') \;\middle|\; \begin{array}{l} (F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa) \\ y \leftarrow F(x) \\ x' \leftarrow \mathcal{A}(F, y) \end{array} \right] \le \mathsf{negl}(\kappa)$$

for every $x \in \mathcal{X}$.

Let $(F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa)$ and $y \in \mathcal{Y}$. If there is no $x \in \mathcal{X}$ such that $F(x) = y$, then running $\mathsf{Invert}(\mathsf{td}, F, y)$ gives an error message $\perp$ as output.

An Almost-Surjective Trapdoor Function (ASTF) is a trapdoor function $F : \mathcal{X} \to \mathcal{Y}$ such that, given any $y \in \mathcal{Y}$, there is, at least, one $x \in \mathcal{X}$ with image y, except with negligible probability.

Definition 2 (Almost-Surjective Trapdoor Function)**.** A Collection of Trapdoor Functions $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ is a *Collection of Almost-Surjective Trapdoor Functions* (ASTF), if for every $(F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa)$, where $F : \mathcal{X} \to \mathcal{Y}$, we have the following:

For every $y \in \mathcal{Y}$,

$$\mathsf{Pr}\left[\perp \leftarrow \mathsf{Invert}(\mathsf{td}, F, y)\right] \le \mathsf{negl}(\kappa)\,.$$

We relax the notion of ASTF by requiring only that a non-negligible fraction of elements in $\mathcal{Y}$ have preimages. In other words, we want that the density of $F(\mathcal{X})$ in $\mathcal{Y}$ to be non-negligible, that is, $|F(\mathcal{X})|/|\mathcal{Y}| \ge 1/p(\kappa)$, where $p(\kappa)$ is some polynomial in $\mathsf{poly}(\kappa)$ and κ is the security parameter. We call such a family of trapdoor functions Sampleable-Range Trapdoor Functions, and we formalize this notion below. Informally, an SRTF ensures that, when choosing an element uniformly at random from its codomain $\mathcal{Y}$, the probability of existing an element in its domain with such image is non-negligible.

Definition 3 (Sampleable-Range Trapdoor Function)**.** Let p be some polynomial and $y \leftarrow_{\$} \mathcal{Y}$ be an element chosen uniformly from $\mathcal{Y}$. A Collection of Trapdoor Functions $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ is a *Collection of Sampleable-Range Trapdoor Functions*, if for every $(F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa)$, where $F : \mathcal{X} \to \mathcal{Y}$, we have

$$\mathsf{Pr}\left[\perp \leftarrow \mathsf{Invert}(\mathsf{td}, F, y)\right] \le 1 - \frac{1}{p(\kappa)}.$$

From the definition above, we have that the probability of algorithm Invert outputting something other than $\perp$ is at least $1/p(\kappa)$. This means that, if we sample elements of $\mathcal{Y}$ uniformly at random and try to invert them, the expected number of trials until we obtain an element $x \in \mathcal{X}$ (and not an error message $\perp$) is $p(\kappa)$.

It is clear that an ASTF is an SRTF, while the converse may not be true (examples are given in Section 6).

3.2 Universal Composability

The Universal Composable (UC) framework [Can01] has been widely adopted to analyze the security of cryptographic primitives. This framework allows proving the security of a protocol, even under arbitrary composition with other protocols. Here, we will give a brief description, for more details see [Can01].

Let $\mathcal{F}$ be an ideal functionality, and π be a protocol implementing the same primitive. Let $\mathcal{Z}$ be an environment, an entity that determines the inputs to the parties involved in the protocol, and has the outputs of these parties revealed to it. In a nutshell, we say that π implements $\mathcal{F}$ if no environment $\mathcal{Z}$ can distinguish between the real-world execution of π and the ideal-world execution of $\mathcal{F}$ (see Figure 3).

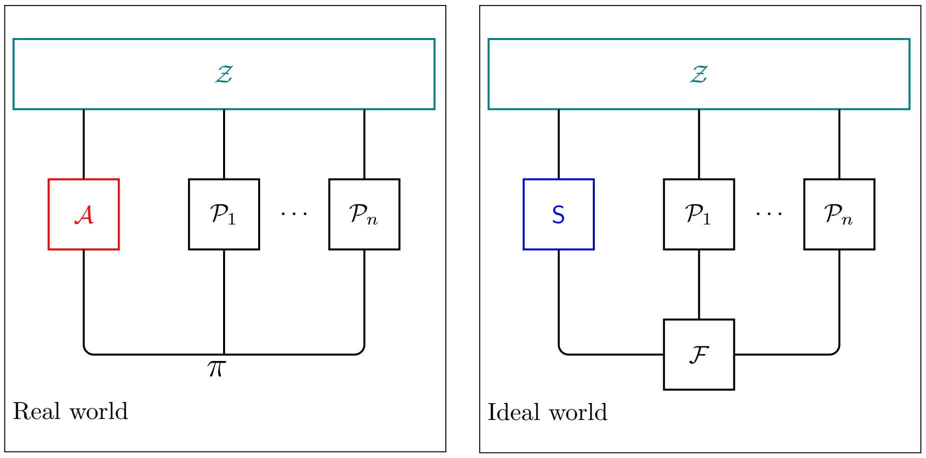

Figure 3: UC framework: In the real-world, an adversary $\mathcal{A}$ can corrupt parties and control them. In the ideal-world, we also have an adversary S, usually called the simulator. We say that a protocol π UC-realizes an ideal functionality $\mathcal{F}$ if, for any PPT adversary $\mathcal{A}$ running in the real-world, there is a PPT simulator S in the ideal world such that no PPT environment $\mathcal{Z}$ can distinguish the executions.

Global Random Oracle. In this paper, we work in the *Global Random Oracle* (gRO) model [CJS14]. The idea of this model is that a unique global random oracle is available for every party involved in the protocol. Thus, the simulator is incapable of programming the oracle, having only access to the queries made by adversaries in the real-world execution. This model has also been studied in [CDG$^+$18], where it is called Restricted Observable Random Oracle. A formal description of the gRO ideal functionality $\mathcal{G}_{\mathsf{gRO}}$ is described in Figure 4, as in [CJS14].

$\mathcal{G}_{\mathsf{gRO}}$ functionality

- **Parameters:** List $\bar{\mathcal{F}}$ of ideal functionalities, output size $p(\kappa)$ where p is a polynomial, and two initially empty lists $\mathcal{Q}$ and $\mathcal{Q}_{|\mathsf{sid}}$.
- Upon receiving a query q from $\mathsf{P} = (\mathsf{pid}, \mathsf{sid})$ or from S do:
  - If $(q, h) \in \mathcal{Q}$, then return h to P. Else, choose $h \leftarrow_{\$} \{0,1\}^{\ell(\kappa)}$, store $(q, h) \in \mathcal{Q}$ and return h to P.
  - Parse q as (s, q'). If $\mathsf{sid} \neq s$, store $(s, q', h) \in \mathcal{Q}_{|\mathsf{sid}}$.
- Upon receiving a message with sid from an instance of an ideal functionality in $\bar{\mathcal{F}}$, return $\mathcal{Q}_{|\mathsf{sid}}$.

Figure 4: $\mathcal{G}_{\mathsf{gRO}}$ ideal functionality.

Let $\mathsf{IDEAL}_{\mathcal{F},\mathsf{S},\mathcal{Z}}$ denote the output of the environment $\mathcal{Z}$ after the ideal-world execution of $\mathcal{F}$ with adversary S, and $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{gRO}}}_{\pi,\mathcal{A},\mathcal{Z}}$ the output of $\mathcal{Z}$ after the real-world execution of π with adversary $\mathcal{A}$, where every party has access to the ideal functionality gRO (the environment has access through $\mathcal{A}$). Security in the gRO-hybrid model is defined as follows.

Definition 4 ([CJS14])**.** Let π be a protocol with n parties, and an adversary $\mathcal{A}$. We say that π *UC-realizes $\mathcal{F}$ in the $\mathcal{G}_{\mathsf{gRO}}$-hybrid model* if for every PPT adversary $\mathcal{A}$ there is a PPT simulator S such that for all PPT environments $\mathcal{Z}$,

$$\mathsf{IDEAL}^{\mathcal{G}_{\mathsf{gRO}}}_{\mathcal{F},\mathsf{S},\mathcal{Z}} \approx \mathsf{EXEC}^{\mathcal{G}_{\mathsf{gRO}}}_{\pi,\mathcal{A},\mathcal{Z}}$$

where $\mathcal{F}$ is an ideal functionality.

Moreover, we also give the formal description of the Commitment ideal functionality $\mathcal{F}_{\mathsf{tcom}}$ in Figure 5, as in [CJS14]. Here, S denotes the simulator in the ideal-world execution.

3.3 Adversarial model

In this work, we consider *phase-adaptive malicious adversaries.* Malicious adversaries are a class of adversaries that may arbitrarily deviate from the protocol. Phase-adaptive adversaries are adversaries that may corrupt between phases of the protocol. We elaborate more on this below.

$\mathcal{F}_{\text{tcom}}$ **functionality**

- **Commitment phase:** Upon receiving $(\mathsf{commit}, \mathsf{sid}, \mathsf{C}, \mathsf{R}, M)$ from C, $\mathcal{F}_{\mathsf{tcom}}$ stores $(\mathsf{sid}, \mathsf{C}, \mathsf{R}, M)$ and sends $(\mathsf{receipt}, \mathsf{sid}, \mathsf{C}, \mathsf{R})$ to R and S. It ignores any future commit messages with the same sid.
- **Opening phase:** Upon receiving $(\mathsf{open}, \mathsf{sid}, \mathsf{C}, \mathsf{R})$ from C, $\mathcal{F}_{\mathsf{tcom}}$ checks if it has stored $(\mathsf{sid}, \mathsf{C}, \mathsf{R}, M)$. If so, it sends $(\mathsf{open}, \mathsf{sid}, \mathsf{C}, \mathsf{R}, M)$ to R and to S. Else, it does nothing.
- When S queries $\mathcal{F}_{\mathsf{tcom}}$ for the list $\mathcal{Q}_{|\mathsf{sid}}$ of $\mathcal{G}_{\mathsf{gRO}}$, it returns $\mathcal{Q}_{|\mathsf{sid}}$.

Figure 5: $\mathcal{F}_{\mathsf{tcom}}$ ideal functionality.

Phase-adaptive security. As explained in Section 2.1, an ideal functionality $\mathcal{F}$ may be divided in several phases (e.g., Bit Commitment). A *phase-adaptive* adversary is an adversary that can corrupt at the beginning or between the phases of the protocol. Hence, at the beginning of the protocol and after the execution of each phase (in the real and ideal worlds), there is a corruption phase, where $\mathcal{A}$ (resp., S) can corrupt honest parties involved in the protocol in the real-world (resp., ideal world). This corruption phase is instantaneous and marks the point where corruption might happen. Corruption in any other point in the protocol is off-limits to an adversary.

We model adversaries precisely as in the adaptive case [CDD$^+$04]. More precisely, an adversary $\mathcal{A}$ after corrupting a party in the real-world execution gets its internal state and controls it from then on. Similarly, the simulator S, after corrupting a party in the ideal-world, also gets its internal state. Corrupted parties remain so until the end of the protocol.

4 UC-Commitment from Trapdoor Functions

We present our generic construction for UC-Commitment based on SRTF. Also, we prove that the construction is perfectly hiding and computationally binding.

4.1 Construction

Let $(\mathcal{Y}, +)$ be a group. Let $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ be a Collection of Sampleable-Range Trapdoor Funcions with $F : \mathcal{X} \to \mathcal{Y}$. We interpret the outputs of gRO as elements in $\mathcal{Y}$.

We assume the existence of an algorithm Ver for the Collection of Trapdoor Functions $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ such that $\mathsf{Ver}(F, \mathsf{td}) = 1$, if $(F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa)$, and $\mathsf{Ver}(F, \mathsf{td}) = 0$, otherwise. Note that this is the case for most of the existing trapdoor functions, as the ones presented in Section 6.

Since we want to avoid an all-powerful adversary from knowing preimages of

gRO, we concatenate uniformly chosen random strings t_i to the queries q_i. The size of these strings is chosen such that there exists exponentially many possible strings $x = q_i|t_i$ with the given size satisfying $\mathsf{gRO}(x) = y$, for every y.In this way, the probability of a computationally-unbounded adversary guessing the preimage of an element y is negligible in κ.[5]

Suppose we have two parties, C and R, and C wants to commit to a message $M \in \{0,1\}^\lambda$.

Commitment phase.

1. R computes $(F, \mathsf{td}) \leftarrow \mathsf{TdGen}(1^\kappa)$ and $y \leftarrow \mathsf{gRO}(\mathsf{td}, t_0)$. It sends (F, y) to C.

2. C chooses $r_1, r_2 \leftarrow_\$ \mathcal{X}$ and computes
$$x_1 \leftarrow \mathsf{gRO}(M, t_1) \text{ and } x_2 \leftarrow \mathsf{gRO}(M, r_1, t_2).$$
Then, it computes
$$c_1 = F(r_1) + x_1 \text{ and } c_2 = F(r_2) + x_2$$
and sends (c_1, c_2) to R.

Opening phase.

1. C computes $x_3 \leftarrow \mathsf{gRO}(M, r_1, r_2, x_1, t_3)$ and sends x_3 to R.

2. R replies with (td, t_0).

3. C checks if $\mathsf{gRO}(\mathsf{td}, t_0) = y$ and aborts if the test fails. Otherwise, it sends
$$(M, r_1, r_2, t_1, t_2, t_3)$$
and halts.

4. R accepts the opening if
$$\begin{cases} c_1 = F(r_1) + \mathsf{gRO}(M, t_1) \\ c_2 = F(r_2) + \mathsf{gRO}(M, r_1, t_2) \\ x_3 = \mathsf{gRO}(M, r_1, r_2, x_1, t_3). \end{cases}$$
Otherwise, it rejects. Finally, it halts.

[5]For example, if $x', y \in \{0,1\}^\kappa$, we use random strings t_i of size $\{0,1\}^\kappa$ such that $x = x'|t_i \in \{0,1\}^{2\kappa}$ is the concatenation of x' and t_i. Thus, we have an expected 2^κ possibilities for the preimage of any $y \in \{0,1\}^\kappa$.

Efficiency. Let $\ell_{\mathcal{X}}$ be the size of the elements of $\mathcal{X}$, and $\ell_{\mathcal{Y}}$ be the size of the elements of $\mathcal{Y}$ (and consequently of the output of gRO). Let ν be the size of the random strings t_i.

The efficiency of the scheme heavily depends on the underlying SRTF. So, we describe the size of the SRTF, generically, as a function of the security parameter, i.e. (F, td) having size $f_{\mathsf{td}}(\kappa)$.

- *Commitment phase*: two rounds, where $f_{\mathsf{td}}(\kappa) + 3\ell_{\mathcal{Y}}$ bits are exchanged.
- *Opening phase*: three rounds, where $f_{\mathsf{td}}(\kappa) + 4\nu + \lambda + 2\ell_{\mathcal{X}} + \ell_{\mathcal{Y}}$ bits are exchanged.

Communication complexity	Computation complexity
$\mathcal{O}\left(f_{\mathsf{td}}(\kappa) + \nu + \lambda + \ell_{\mathcal{X}} + \ell_{\mathcal{Y}}\right)$	8 calls to gRO 1 run of TdGen 4 evaluations of F

Table 1: Complexity analysis of our framework.

4.2 Security

Theorem 5. Let $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ be a collection of SRTF. Then, the scheme is perfectly hiding and computationally binding in the gRO model.

Proof. We need to show that the hiding property holds for any adversary, even if it has unbounded computational power, and that the binding property holds for any PPT adversary.

(Perfectly) Hiding property. The commitment corresponds to the values $c_1 = F(r_1) + \mathsf{gRO}(M, t_1)$ and $c_2 = F(r_2) + \mathsf{gRO}(M, r_1, t_2)$. Observe that the message is always hidden by gRO. More precisely, the message is hidden by the queries (M, t_1) and (M, r_1, t_2). From the assumption on t_1 and t_2, there is a negligible probability that an adversary, even with unlimited computational power, finds (M, t_1) when given $y \leftarrow \mathsf{gRO}(M, t_1)$, since there are an exponential number of queries of the form (M', t_1') such that $y \leftarrow \mathsf{gRO}(M', t_1')$. The same applies for (M, r_1, t_2). Hence, the message is perfectly hidden from R.

Binding property. For the sake of contradiction, suppose that there is a PPT adversary $\mathcal{A}$ that can come up with a commitment such that $\mathcal{A}$ can open two different messages. That is, $\mathcal{A}$ finds c_1, c_2 and two openings, $(x_3, (M, r_1, r_2, t_1, t_2, t_3))$ and $(x_3', (M', r_1', r_2', t_1', t_2', t_3'))$, such that:

1. $M \neq M'$,
2. $x_3 = \mathsf{gRO}(M, r_1, r_2, x_1, t_3)$ and $x_3' = \mathsf{gRO}(M', r_1', r_2', x_1', t_3')$,

3. $c_1 = F(r_1) + \mathsf{gRO}(M, t_1) = F(r'_1) + \mathsf{gRO}(M', t'_1)$,

4. $c_2 = F(r_2) + \mathsf{gRO}(M, r_1, t_2) = F(r'_2) + \mathsf{gRO}(M', r'_1, t'_2)$.

Observe that, if $r_1 = r'_1$ or $r_2 = r'_2$, then $\mathcal{A}$ is able to find collisions for gRO. If $r_1 = r'_1$, then $\mathsf{gRO}(M, t_1) = \mathsf{gRO}(M', t'_1)$ and, analogously, if $r_2 = r'_2$ then $\mathsf{gRO}(M, r_1, t_2) = \mathsf{gRO}(M', r'_1, t'_2)$. Thus, $r_1 \neq r'_1$ and $r_2 \neq r'_2$, except with negligible probability.

Now, without loss of generality, let us fix $r_1 \in \mathcal{X}$ and any two messages M and M'. In order to find another valid opening that fulfills the conditions, $\mathcal{A}$ must be able to find r'_1 such that $F(r'_1) = z = F(r_1) + \mathsf{gRO}(M, t_1) - \mathsf{gRO}(M', t'_1)$. First, note that the distribution of z is uniform in $\mathcal{Y}$ since the outputs gRO are uniform in $\mathcal{Y}$. This means that, if $\mathcal{A}$ is able to find such r'_1, then it is able to invert F for a uniformly random z. However, since F is a trapdoor function, it is infeasible for $\mathcal{A}$ to invert F, except with negligible probability.

The same argument can be adapted to the condition in Item 4.

Hence, we conclude that it is infeasible for a PPT $\mathcal{A}$ to open two different messages for any commitment (c_1, c_2), except with negligible probability. □

5 Simulation

In this section, we demonstrate how to construct the simulator to prove that the scheme is Universally Composable. We first describe the simulator for static adversaries and then for phase-adaptive adversaries.

Theorem 6. The protocol securely UC-realizes $\mathcal{F}_{\text{tcom}}$ against phase-adaptive malicious adversaries in the $\mathcal{G}_{\mathsf{gRO}}$-hybrid model, given that $(\mathsf{TdGen}, \mathsf{Eval}, \mathsf{Invert})$ is a Collection of SRTF.

As usual, the simulator S simulates the communication between the real-world adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ by writing all messages received from $\mathcal{Z}$ in $\mathcal{A}$'s input tape, and by forwarding all messages from $\mathcal{A}$ to $\mathcal{Z}$.

The trivial case where both parties are corrupted by $\mathcal{A}$, S simply runs $\mathcal{A}$ and lets it generate all the transcript between C and R, forwarding all the messages to $\mathcal{Z}$.

5.1 Simulation when only C is corrupted

When C is corrupted, the simulator S extracts the message that C is committing to, and sends this message to the ideal functionality $\mathcal{F}_{\text{tcom}}$. Extraction is possible by the fact that C needs to query gRO on M and on (M, r_1). Otherwise, it is not able to open a valid message in the opening phase.

Commitment phase. In the commitment phase, S proceeds as follows:

- It runs TdGen and gets (F, td). It queries gRO on (td, t_0) and sets the output to y. It sends (F, y) as the honest R would do.

- Upon receiving (c_1, c_2) from C, it asks for $\mathcal{Q}_{|\mathsf{sid}}$ to $\mathcal{F}_{\text{tcom}}$. It checks if there are queries (M, t_1) and (M, r_1, t_2) such that $c_1 = F(r_1) + x_1$ where $x_1 \leftarrow \mathsf{gRO}(M, t_1)$. If so and if these queries are unique, it sets $M^* = M$. Else, it chooses $M^* \leftarrow_{\$} \{0,1\}^\lambda$. Finally, it sends the message $(\mathsf{commit}, \mathsf{sid}, \mathsf{C}, \mathsf{R}, M^*)$ to $\mathcal{F}_{\text{tcom}}$.

Opening phase. In the opening phase, S proceeds as follows:

- Upon receiving x_3 from C, S reveals (td, t_0) to C, as the honest R would.
- Upon receiving $(M, r_1, r_2, t_1, t_2, t_3)$, S checks if

$$\begin{cases} c_1 = F(r_1) + \mathsf{gRO}(M, t_1) \\ c_2 = F(r_2) + \mathsf{gRO}(M, r_1, t_2) \\ x_3 = \mathsf{gRO}(M, r_1, r_2, x_1, t_3) \end{cases}$$

  and aborts if any of these tests fail, as the honest R would. Moreover, it checks whether $M = M^*$. If the test fails, S aborts the execution. Else, it sends the message $(\mathsf{open}, \mathsf{sid}, \mathsf{C}, \mathsf{R})$ to $\mathcal{F}_{\text{tcom}}$ and halts.

Indistinguishability of executions. Both the real-world and ideal-world executions are indistinguishable from the point-of-view of $\mathcal{Z}$, unless $\mathcal{Q}_{|\mathsf{sid}}$ has no queries (M, t_1) and (M, r_1, t_2) such that $c_1 = F(r_1) + x_1$, where $x_1 \leftarrow \mathsf{gRO}(M, t_1)$. However, in this case, the probability of C opening a valid message is negligible. More precisely, given c_1, it is infeasible for C to find an opening $(M, r_1, r_2, t_1, t_2, t_3)$ such that $c_1 = F(r_1) + x_1$ without either breaking the security of the underlying SRTF (as in the proof of Theorem 5) or finding collisions for gRO.

Also, remark that the queries (M, t_1) and (M, r_1, t_2) fulfilling these conditions are unique, except with negligible probability. Otherwise, the binding property would not hold.

5.2 Simulation when only R is corrupted

When R is corrupted, the simulator S needs to commit to any message M in the commitment phase (e.g., $M = 0^\lambda$). After receiving the message M^* from $\mathcal{F}_{\text{tcom}}$ (most likely, different from M), S needs to equivocate the real-world R and open M^*.

Commitment phase. In the commitment phase, S proceeds as follows:

- Upon receiving a message $(\mathsf{receipt}, \mathsf{sid}, \mathsf{C}, \mathsf{R})$ from $\mathcal{F}_{\text{tcom}}$, it starts running $\mathcal{A}$.
- Upon receiving a message (F, y) from R, S commits to the message $M = 0^\lambda$ by following the protocol.

Opening phase. In the opening phase, S proceeds as follows:

- Upon receiving a message $(\mathsf{open}, \mathsf{sid}, \mathsf{C}, \mathsf{R}, M^*)$ from $\mathcal{F}_{\text{tcom}}$, S asks $\mathcal{F}_{\text{tcom}}$ for $\mathcal{Q}_{|\mathsf{sid}}$. It checks if there is a query q in $\mathcal{Q}_{|\mathsf{sid}}$ whose output is y. Now, there are two cases to consider:

    1. If this query exists, it parses $q = (\mathsf{td}, t_0)$ and sets $\mathsf{td}^* = \mathsf{td}$. It computes $r_1^* \leftarrow \mathsf{Invert}(\mathsf{td}, F, c_1 - x_1^*)$, where $x_1^* \leftarrow \mathsf{gRO}(M^*, t_1^*)$. If $r_1^* =\perp$, S chooses a new random string t_1^* and tries again. It repeats this process until $r_1^* \neq\perp$. Note that, since F is SRTF, then $\mathcal{S}$ can find $r_1 \neq\perp$ in a polynomial number of tries.

        Then, it computes $r_2^* \leftarrow \mathsf{Invert}(\mathsf{td}, F, c_2 - x_2^*)$, where $x_2^* \leftarrow \mathsf{gRO}(M^*, r_1^*, t_2^*)$. Again, it repeats the process until $r_2^* \neq\perp$, by choosing a new random string t_2^* at each new try. It queries gRO on $(M^*, r_1^*, r_2^*, x_1^*, t_3^*)$, where t_3^* is a random string, and sets the output to x_3.
    2. If there is no such query, it sets $\mathsf{td}^* =\perp$, where $\perp$ is an error message. Also, it chooses uniformly at random x_3.

    It sends x_3 to R.

- Upon receiving (td, t_0) from R, S checks if $y = \mathsf{gRO}(\mathsf{td}, t_0)$. If not, aborts the execution. Else, there are again two cases to consider:

    1. If $\mathsf{td}^* = \mathsf{td}$, it reveals $(M^*, r_1^*, r_2^*, t_1^*, t_2^*, t_3^*)$ to R.
    2. Else, it chooses random values for M^*, r_1^*, r_2^*, t_1^*, t_2^*, and t_3^*, and reveals them to R.

    It halts the execution.

Indistinguishability of executions. Both the real-world and ideal-world executions are indistinguishable from the point-of-view of $\mathcal{Z}$ unless $\mathsf{td}^* \neq \mathsf{td}$. However, if $\mathsf{td}^* \neq \mathsf{td}$, then R is able to find a collision for gRO,which happens only with a negligible probability. So the two executions are indistinguishable, except with negligible probability.

5.3 Simulation when neither party is corrupted

When both parties are honest, S generates all the messages between the real-world C and R. Since S generates the messages, it has access to the trapdoor td of F and, thus, it can open any message that it wants (as in the case where only S is corrupted). More precisely, it can open the same message that the ideal-world C opens. It is straightforward to see that this case is indistinguishable from the ideal-world execution, since the transcript created by S is indistinguishable from a honestly generated one.

5.4 Phase-adaptive corruption of C

In this case, we assume C gets phase-adaptively corrupted, that is, it may be corrupted before the commitment phase or exactly during the period between the commitment and opening phase. Note that in the case of Commitment schemes, it does not make sense to consider post-execution corruption, as there is no secret information subsequent to the protocol.

There are two cases to consider: C is corrupted before the commitment phase, or C is corrupted between the commitment and the opening phases. In each case, S plays the role of C until the corruption happens. At that point, it has to deliver an internal state for C to $\mathcal{A}$. This state must be coherent with both the transcript and C's input.

C is corrupted before the commitment phase. There are two possibilities:

1. If R is corrupted then S behaves as in the case where both parties are statically corrupted.
2. If R is not corrupted then S behaves as in the case where C (but not R) is statically corrupted.

Indistinguishability of executions. Since the probability of opening a valid message without making any query to the gRO functionality, even for phase-adaptive adversary corrupting C, is negligible, the same argument of static corruption applies here.

C is corrupted before the opening phase. In this case, $\mathcal{A}$ corrupts C after the commitment phase but before the opening phase. S runs the protocol honestly committing to $M = 0^\lambda$ and sending c_1, c_2 to R. S corrupts the committer C^* in the ideal-world and gets its internal state. In particular, it gets M^*, the message that C^* is committing to. There are three cases to consider:

1. If R is corrupted since the beginning of the protocol, then S asks for $\mathcal{Q}_{|\mathsf{sid}}$ and extracts the trapdoor td. Now it can equivocate (as in the case where R is statically corrupted) and find r_1^*, r_2^*, t_1^* and t_2^* such that $c_1 = F(r_1^*) + \mathsf{gRO}(M^*, t_1^*)$ and $c_2 = F(r_2^*) + \mathsf{gRO}(M^*, r_1^*, t_2^*)$. In other words, $(M^*, r_1^*, r_2^*, t_1^*, t_2^*)$ is a coherent internal state for C between phases. From now on, S behaves as in the case where both parties are statically corrupted.
2. If R is corrupted, but not since the beginning of the protocol (that is, R is also corrupted between both phases), then S simulates the protocol up to this point and, thus, it has the trapdoor for F which allows for equivocation. Therefore, it can proceed as before to reveal a valid internal state of C to $\mathcal{A}$. From now on, S behaves as in the case where both parties are statically corrupted.
3. If R is not corrupted, then S also has the trapdoor. From now on, S behaves as in the case where C is statically corrupted.

Indistinguishability of executions. Since R needs to commit to a trapdoor, except with negligible probability, then S is always able to open a valid message. So, the same argument used for static corruption can also be applied here.

5.5 Phase-adaptive corruption of R

In this case, we prove security when R gets phase-adaptively corrupted. We describe how S is able to create a transcript up to the point of corruption, such that it can deliver an internal state of R to $\mathcal{A}$ that is coherent with that transcript.

R **is corrupted before the commitment phase.** Again, there are two cases to consider:

1. If C is corrupted then S behaves as in the case where both parties are statically corrupted.
2. If C is not corrupted then S behaves as in the case where R (but not C) is statically corrupted.

R **is corrupted before the opening phase.** In this case, $\mathcal{A}$ corrupts R after the commitment phase but before the opening phase. Then, S corrupts the ideal R^* and gets its internal state. Since S has generated the pair (F, td) being used, it can reveal this information to $\mathcal{A}$, which is a coherent internal state with the one of R. Note that this is independent of whether the committer is corrupted or not.

Indistinguishability of executions. Again, the argument is similar to the case where parties are statically corrupted. That is, the inability to find a collision or a preimage in gRO up to a negligible probability guarantees the indistinguishability of the executions in the real and ideal-worlds.

6 Instantiations

In order to demonstrate the versatility of our framework, we present several instantiations from well-known collections of Trapdoor Functions that may be found in the literature. From our generic construction, we construct UC-Commitment schemes based on the RSA assumption, Trapdoor Discrete Log groups, lattice-based and code-based assumptions. As far as we know, this is the first time a UC-Commitment based on lattice-based assumptions is proposed.

Nevertheless, possible instantiations of the framework are not limited to the ones presented here. For instance, a recent line of research has found trapdoor functions based on the hardness of the Decisional Diffie-Hellman and Computational Diffie-Hellman (e.g., [PW08, GH18, GGH18]), which are adequate for our framework.

6.1 Realizations from factorization-based assumptions

There are numerous trapdoor functions whose security is based on the hardness of factorization (or similar problems). Here, we present examples using the RSA cryptosystem, Trapdoor Discrete Log groups, and Rabin's Quadratic Residue cryptosystem.

6.1.1 RSA

We show how to instantiate our framework using the well-known RSA cryptosystem. Let Φ denote Euler's totient function. We first present the RSA problem.

Definition 7 (RSA)**.** Let $p, q \in \mathbb{N}$ be two different primes and $n = pq$. Let $e, d \in \mathbb{Z}_{\Phi(n)}$ such that $ed \equiv 1 \mod \Phi(n)$. The RSA_n problem is ε-hard, if for every PPT algorithm D we have

$$\Pr[m \leftarrow \mathsf{D}(n, e, m^e \bmod n)] \leq \varepsilon.$$

Lemma 8 ([RSA78])**.** *Let $p, q \in \mathbb{N}$ be two different prime numbers such that RSA_n is hard, for $n = pq$. There exists an algorithm TdGen that receives as input (p, q) and outputs a pair $(e, \mathsf{td} = d)$ where $e \in \mathbb{Z}_{\Phi(n)}$ and $\mathsf{td} = d$ is a trapdoor for e such that $ed \equiv 1 \mod \Phi(n)$. The Eval algorithm takes $x \in \mathbb{Z}_n$ and computes $x^e \mod n$. The Invert algorithm takes as input $y \in \mathbb{Z}_n$ and computes $y^d \mod n$.*

It is trivial to see that this is a Collection of ASTF: to find the preimage of $z \in Z_n$, one has to compute z^d. It is evident that $F(z^d) = z$.

6.1.2 Trapdoor Discrete Log

There are groups for which it is easy to compute discrete logarithms with some additional help, that is, a trapdoor. The study of these groups goes back to the work of Gordon [Gor93]. However, as far as we know, the notion of Trapdoor Discrete Log groups was just formalized in [PS09]. In this work, we use the Trapdoor Function of [PS09] to instantiate our framework.

Lemma 9 ([PS09])**.** *Let $p, q \in \mathbb{N}$ be two different prime numbers such that $p \equiv 3 \mod 4$, $q \equiv 1 \mod 4$, $\gcd(p-1, q-1) = 2$ and $n = pq$, such that the RSA_n is hard. Let g be such that $g \mod p$ is primitive in $\mathbb{Z}_p$ and $g \mod q$ is primitive in $\mathbb{Z}_q$, and let G be a subgroup of $\mathbb{Z}_n$. There exists an algorithm TdGen that receives as input (p, q) and outputs a pair (g, td) where $\mathsf{td} = (p, q)$ is the trapdoor. The Eval algorithm takes x as input and computes $g^x \in G$. The Invert algorithm takes as input $y \in \mathbb{Z}_n$ and td and computes x such that $g^x = y$.*

Since g is a generator of G, every $y \in G$ can be written as $y = g^x$, meaning that every element can be inverted using the trapdoor. We conclude that this collection is, in fact, a Collection of ASTF.

Comparison. The scheme of [CJS14], which is based on the Discrete Log (DL) problem, requires ten exponentiations and six calls to the gRO. In contrast, our scheme, when instantiated using Trapdoor Discrete Log groups, requires four exponentiations and six calls to the gRO. Our scheme also involves generating a trapdoor function, that is, it requires the generation of two primes p, q and the generation of g.

We remark that, although both schemes are based on modular exponentiations, the underlying hardness assumptions are different. The security of our scheme is based on the RSA_n problem, while the security of the scheme in [CJS14] is based on the hardness of the DL problem.

6.1.3 Rabin's Quadratic Residue

Definition 10 (Quadratic Residuosity)**.** Let $p, q \in \mathbb{N}$ be two different primes such that $p, q \equiv 3 \mod 4$ and $n = pq$. Let $y \equiv x^2 \mod n$. The QR_n problem is ε-hard if for every PPT algorithm D we have

$$\Pr[x \leftarrow \mathsf{D}(n, y)] \leq \varepsilon.$$

Lemma 11. Let $p, q \in \mathbb{N}$ be two different prime numbers such that $p, q \equiv 3 \mod 4$. There exists an algorithm TdGen that receives as input (p, q) and outputs a pair $(F, \mathsf{td} = (p, q))$. The Eval algorithm takes $x \in \mathbb{Z}_n$ and computes $x^2 \mod n$. The Invert algorithm takes as input $y \in \mathbb{Z}_n$ and (p, q) and outputs $x \in \mathbb{Z}_n$ such that $x^2 \equiv y \mod n$.

The number of quadratic residues in $\mathbb{Z}_n$ is $(n-1)/2$. So, this defines a Collection of SRTF.

6.2 Realization from lattice-based assumptions

Lattices are prevalent structures in post-quantum cryptography. Since lattice-based trapdoor functions were first constructed in [GPV08, MP12], we can use them to obtain UC-Commitment schemes. We recall that our generic construction yields the first lattice-based UC-Commitment scheme (not constructed via generic transformations, like OT).

We present the Small Integer Solution (SIS) problem, which is the foundation of the trapdoor function used to instantiate our framework. Let $\|x\|$ denote the usual ℓ_2 norm of a vector in $\mathbb{Z}_q^n$.

Definition 12 (Small Integer Solution)**.** Let $q \in \mathbb{Z}$, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\beta \in \mathbb{R}$. The $\mathsf{SIS}_{q,m,\beta}$ problem is ε-hard if for every PPT algorithm D we have

$$\Pr[\mathbf{e} \leftarrow \mathsf{D}(q, \mathbf{A}, \beta)] \leq \varepsilon$$

such that $\mathbf{e} \in \mathbb{Z}_q^m$, $\|\mathbf{e}\| \leq \beta$ and $\mathbf{A}\mathbf{e} = 0 \mod q$.

The Inhomogeneous Small Integer Solution (ISIS) problem is defined similarly, but D is additionally given $\mathbf{u} \in \mathbb{Z}_q^n$ and the goal is to find $\mathbf{e} \in \mathbb{Z}_q^m$ such that $\|\mathbf{e}\| \leq \beta$ and $\mathbf{A}\mathbf{e} = \mathbf{u} \mod q$.

Lemma 13 ([GPV08])**.** Let $m, n, q \in \mathbb{N}$ such that $m > 5n \log q$ and $s \geq L\omega\left(\sqrt{\log m}\right)$ where $L = m^{2.5}$, such that $\mathsf{ISIS}_{q,m,\beta}$ is hard. There exists an algorithm TdGen that receives as input (m, n, q) and outputs a pair $(\mathbf{A}, \mathsf{td})$ where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is a matrix and td is a trapdoor (that is, a *good basis*) for $\mathbf{A}$ with the following properties:

- There is an algorithm Eval which takes $\mathbf{A}$ and $\mathbf{e}$, and outputs $\mathbf{Ae}$. And, an algorithm Invert that takes as input td, $\mathbf{A}$ and a word $\mathbf{z} \in \{0,1\}^n$. It outputs $\mathbf{e} \in \mathbb{Z}_q^m$ such that $\mathbf{Ae} = \mathbf{z} \mod q$ and $\|\mathbf{e}\| \leq s\sqrt{m}$
- $\mathbf{A}$ is indistinguishable from a uniformly chosen matrix in $\mathbb{Z}_q^{n\times m}$, given that $\mathsf{ISIS}_{q,m,\beta}$ is hard

Since in the collection of trapdoor functions described in Lemma 13 the algorithm Invert never fails. It is, in fact, a Collection of ASTF.

6.3 Realization from code-based assumptions

Lastly, we show that we can obtain UC-Commitment schemes from code-based assumptions. We present the Syndrome Decoding and Goppa Distinguisher assumptions. Let $\mathfrak{B}_\omega^n = \{\mathbf{e} \in \mathbb{Z}_2^n : w(\mathbf{e}) \leq \omega\}$.

Definition 14 (Syndrome Decoding)**.** Let $n, k, \omega \in \mathbb{N}$, $\mathbf{H} \leftarrow_{\$} \{0,1\}^{(n-k)\times n}$ and $\mathbf{e} \leftarrow_{\$} \mathfrak{B}_\omega^n$. The $\mathsf{SD}_{\omega,n,k}$ problem is ε-hard if for every PPT algorithm D we have

$$\Pr\left[\mathbf{e} \leftarrow \mathsf{D}(\mathbf{H}, \mathbf{He}^T)\right] \leq \varepsilon.$$

Definition 15 (Goppa Distinguisher)**.** Let $n, k \in \mathbb{N}$ such that $k < n$. Let $\mathsf{Gop}(n,k)$ be an algorithm that outputs a matrix defining a binary Goppa code of size $k \times n$. The $\mathsf{GD}_{n,k}$ is ε-hard, if for every PPT algorithms D, we have

$$|\mathsf{Pr}\left[1 \leftarrow \mathsf{D}(\mathbf{A}) : \mathbf{A} \leftarrow \mathsf{Gop}(n,k)\right] - \mathsf{Pr}\left[1 \leftarrow \mathsf{D}(\mathbf{A}) : \mathbf{A} \leftarrow_{\$} \{0,1\}^{k\times n}\right]| \leq \varepsilon.$$

Lemma 16 ([CFS01])**.** Let $n, k, \omega \in \mathbb{N}$ with $k < n$, such that $\mathsf{SD}_{\omega,n,k}$ and $\mathsf{GD}_{n,k}$ are hard. There exists an algorithm TdGen that receives as input n, k and ω and outputs a pair $(\mathbf{H}, \mathsf{td})$, where $\mathbf{H} \in \{0,1\}^{(n-k)\times n}$ is a matrix and td is a trapdoor (decoding algorithm) for $\mathbf{H}$. Eval takes as input $\mathbf{e} \in \mathfrak{B}_\omega^n$ and outputs $\mathbf{He}^T$. Invert takes as input td and $\mathbf{He}^T \in \mathbb{Z}_2^{n-k}$ and outputs $\mathbf{e}$. This Collection of Trapdoor Functions has the following properties:

- There is an algorithm Eval which given $\mathbf{H}$ and $\mathbf{e}$ outputs $\mathbf{He}^T$. And, an algorithm Invert that takes as input td and $\mathbf{s} \in \{0,1\}^{n-k}$. It outputs either $\mathbf{e}$, if $\mathbf{s}^T = \mathbf{He}^T$ and $w(\mathbf{e}) \leq \omega$, or a message error $\perp$, otherwise;
- There is a non-negligible number of words $\mathbf{z}$ in $\{0,1\}^{n-k}$ for which $\mathsf{Invert}(\mathbf{z}, \mathsf{td})$ does not output a message error $\perp$;
- $\mathbf{H}$ is indistinguishable from a uniformly chosen matrix in $\{0,1\}^{(n-k)\times n}$ given that the $\mathsf{GD}_{n,k}$ problem is hard.

We remark that, while the trapdoor of [CFS01] is based on the Goppa Distinguisher (GD) assumption, there is another trapdoor [DAST18] based on the recently introduced problem of distinguishing *generalized admissible* codes $(\mathbf{U}, \mathbf{U}+\mathbf{V})$ from uniformly chosen codes, which is also assumed to be hard (although not so well studied as the GD problem). In fact, the trapdoor of [DAST18] also fulfills the conditions to be used in our framework.

Because the collection of trapdoor functions described in Lemma 16 sometimes fails to decode, i.e. the algorithm Invert returns $\perp$, [CFS01] is a Collection of SRTF.

Comparison. Another UC-Commitment scheme based on coding assumptions was proposed in [Bra19]. However, our scheme achieves a significant improvement in efficiency, both in terms of computation and communication complexity. Indeed, we avoid the use of Zero-Knowledge Proof (ZKP) systems, contrarily to the UC-Commitment of [Bra19], which uses the ZKP system of [JKPT12]. Since the ZKP system of [JKPT12] has a soundness error of 2/3, the number of iterations needed to get a negligible soundness error is too large. These facts give raise to a scheme that has impractical computational and communication complexity, despite its theoretical value.

Acknowledgment

The authors thank Ran Canetti for discussion concerning phase-adaptive adversaries in the UC-framework and João Ribeiro for comments on an early draft of this work. The first author thanks the support from DP-PMI and FCT (Portugal) through the grant PD/BD/135181/2017. The second author thanks the support from DP-PMI and FCT (Portugal) through the grant PD/BD/135182/2017. This work is funded by the projects QuantumMining POCI-01-0145-FEDER-031826, and QDOT POCI-01-0247-FEDER-039728.

References

[BPRS17] Megha Byali, Arpita Patra, Divya Ravi, and Pratik Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. Cryptology ePrint Archive, Report 2017/1165, 2017. `https://eprint.iacr.org/2017/1165`.

[Bra19] Pedro Branco. A post-quantum UC-commitment scheme in the global random oracle model from code-based assumptions. Cryptology ePrint Archive, Report 2019/098, 2019. `https://eprint.iacr.org/2019/098`.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42Nd IEEE Sympo-*

sium on Foundations of Computer Science, FOCS '01, pages 136–, Washington, DC, USA, 2001. IEEE Computer Society.

[CDD$^+$04] Ran Canetti, Ivan Damgard, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. Adaptive versus non-adaptive security of multi-party protocols. *Journal of Cryptology*, 17(3):153–207, Jun 2004.

[CDD$^+$16] Ignacio Cascudo, Ivan Damgård, Bernardo David, Nico Döttling, and Jesper Buus Nielsen. Rate-1, linear time and additively homomorphic uc commitments. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 179–207, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[CDG$^+$18] Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 280–312, Cham, 2018. Springer International Publishing.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 19–40, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[CFS01] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 157–174, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 597–608, New York, NY, USA, 2014. ACM.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, pages 337–351, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 494–503, New York, NY, USA, 2002. ACM.

[DAST18] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new code-based signature scheme. Cryptology ePrint Archive, Report 2018/996, 2018. `https://eprint.iacr.org/2018/996`.

[DG03] Ivan Damgard and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 426–437, New York, NY, USA, 2003. ACM.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 581–596, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 468–485, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Fuj16] Eiichiro Fujisaki. Improving practical UC-secure commitments based on the DDH assumption. In Vassilis Zikas and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 257–272, Cham, 2016. Springer International Publishing.

[GGH18] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. Cryptology ePrint Archive, Report 2018/872, 2018. `https://eprint.iacr.org/2018/872`.

[GH18] Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational Diffie-Hellman assumption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 362–391, Cham, 2018. Springer International Publishing.

[GIKW14] Juan A. Garay, Yuval Ishai, Ranjit Kumaresan, and Hoeteck Wee. On the complexity of UC commitments. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 677–694, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[GMY04] Juan A. Garay, Philip MacKenzie, and Ke Yang. Efficient and universally composable committed oblivious transfer and applications. In Moni Naor, editor, *Theory of Cryptography*, pages 297–316, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Gor93] Daniel M. Gordon. Designing and detecting trapdoors for discrete log cryptosystems. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 66–75, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[HMQ04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *Theory of Cryptography*, pages 58–76, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[JKPT12] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 663–680, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. ACM.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 446–466, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[PS09] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Designs, Codes and Cryptography*, 52(2):219–241, Aug 2009.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 554–571, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 187–196, New York, NY, USA, 2008. ACM.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.