On Simulation-Sound Trapdoor Commitments

Philip MacKenzie* Ke Yang†

December 2, 2003

Abstract

We study the recently introduced notion of a *simulation-sound trapdoor commitment (SSTC)* scheme. In this paper, we present a new, simpler definition for an SSTC scheme that admits more efficient constructions and can be used in a larger set of applications. Specifically, we show how to construct SSTC schemes from any one-way functions, and how to construct very efficient SSTC schemes based on specific number-theoretic assumptions. We also show how to construct simulation-sound, non-malleable, and universally-composable zero-knowledge protocols using SSTC schemes, yielding, for instance, the most efficient universally-composable zero-knowledge protocols known. Finally, we explore the relation between SSTC schemes and non-malleable commitment schemes by presenting a sequence of implication and separation results, which in particular imply that SSTC schemes are non-malleable.

1 Introduction

The notion of a *commitment* is one of the most important and useful notions in cryptography. Intuitively, a commitment is the digital equivalent of a "sealed envelope." A party Alice would commit to a value by placing it into a sealed envelope, so that the value may later be revealed by Alice opening the envelope, but cannot be viewed by any other party prior to this opening (this is known as the "secrecy" or "hiding" property), and cannot be altered (this is known as the "binding" property). Commitments have been useful in a wide range of applications, from zero-knowledge protocols (e.g., [4, 15, 34]) to electronic commerce (e.g., remote electronic bidding), and have been studied extensively (e.g., [3, 40, 41]).

A *commitment scheme* is simply a method for generating and opening commitments. One can construct a formal definition of security for a commitment scheme directly from the properties inherent in the intuitive description above. However, often these properties turn out to be insufficient when commitments are used as building blocks in larger protocols or when multiple commitments are used concurrently. This has motivated researchers to define and construct commitment schemes with additional properties. We discuss them briefly below.

A *trapdoor commitment (TC) scheme* is a commitment scheme with an additional "equivocability" property. Roughly speaking, for such a commitment scheme there is some *trapdoor* information whose knowledge would allow one to open a commitment in more than one way (and thus "equivocate"). Naturally, without the trapdoor, equivocation would remain computationally infeasible [4, 25, 2].

A *non-malleable commitment (NMC) scheme* is a commitment scheme with the property that (informally) not only is the value v placed inside a commitment secret, but seeing this commitment does not give another party any advantage in generating a new commitment that, once v is revealed, can then be opened to a value related to v [22, 20, 29, 21, 17].[1]

*Bell Labs – Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974. E-mail: philmac@research.bell-labs.com.

†Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213. E-mail: yangke@cs.cmu.edu. Part of the this research was done at Bell Labs. This research was also partially sponsored by National Science Foundation (NSF) grants CCR-0122581 and CCR-0085982.

[1] The original definition of [22] states (informally) that another party does not even have any advantage in creating a new commitment to a value related to v, regardless of the ability to open the new commitment. However, we will use the definition based on opening.

A *universally composable commitment (UCC) scheme* is a commitment scheme with a very strong property that intuitively means that the security of a commitment is guaranteed even when commitment protocols are concurrently composed with arbitrary protocols [6, 7, 18]. To achieve universal composability, a commitment scheme seems to require equivocability, non-malleability, and furthermore, *extractability*. Roughly speaking, an extractable commitment scheme has a modified secrecy definition, which states that there is a *secret key* whose knowledge would allow one to extract the value placed in a commitment. Naturally, without this knowledge, the value would remain hidden. We note that the notion of a UCC scheme appears to be strictly stronger than the other notions of commitment schemes. In particular, Damgård and Groth [17] show that a UCC scheme implies secure key exchange, while both TC schemes and NMC schemes can be constructed from one-way functions.

1.1 Simulation Sound Trapdoor Commitment

In this paper, we focus our attention on another extension of commitment schemes, namely simulation sound trapdoor commitment (SSTC) schemes. An SSTC scheme is a TC scheme with a strengthened binding property, called simulation-sound binding. Roughly speaking, in an SSTC scheme, an adversary cannot equivocate on a commitment with a certain tag, even after seeing the equivocation of an unbounded number of commitments with different tags (i.e., the adversary may request an equivocation oracle to generate an unbounded number of commitments with different tags, and then to open them to arbitrary values). Here, a tag for a commitment is simply a binary string associated with the commitment.

The term "simulation soundness" was first used to describe a property of zero-knowledge proofs by Sahai [47], and intuitively meant that even though an adversary could see simulated proofs of incorrect statements, it could not itself produce a new simulated proof of any incorrect statement. Garay *et al.* [31] first applied this term to trapdoor commitments. They gave a slightly stronger, although more complicated, simulation-sound binding property and an efficient construction based on DSA signatures. Their definition was specifically tailored to the goal of developing a universally-composable zero-knowledge (UCZK) proof that was secure in the presence of adversaries that could adaptively corrupt parties.[2]

Perhaps the most interesting feature of SSTC schemes is that they are both very powerful and very efficient to construct. As we will show later in this paper, SSTC schemes are non-malleable and can be used to construct simulation-sound, non-malleable, and/or universally composable zero-knowledge protocols. On the other hand, SSTC schemes can be constructed from one-way functions only, in contrast to UCC schemes, which are considered highly unlikely to be constructible from one-way functions alone [17]. Also, based on specific number-theoretic assumptions (e.g., strong RSA, or the DSA assumption), very efficient SSTC schemes can be constructed, as we show in the paper. These constructions in turn yield UCZK protocols that are more efficient than all previously known ones, which are either based on UCC schemes [7, 18, 17] or on the previous definition of SSTC schemes [31].[3]

1.2 Summary of Results

Simpler Definition We provide a simpler definition of SSTC schemes than the one by Garay *et. al.* [31]. Though the binding property in our definition is weaker, it is still sufficient in many applications (e.g., to construct UCZK protocols that are secure in the presence of adversaries that can adaptively corrupt parties).

[2] They use the term *identifier* in place of the term *tag*, and intuitively, in their definition [31], a commitment made by the adversary using identifier id is binding, even if the adversary has seen any commitment using identifier id opened (using an oracle that knows a trapdoor) once to any arbitrary value, and moreover, any commitment using identifier $id' \neq id$ opened (again using the oracle) an unbounded number of times to any arbitrary values.

[3] In fact, this (improved efficiency) is one of the main motivations to study the new simpler definition of SSTC.

We also discuss various design issues in the definition, and most notably, the choice between definitions based on the tag of the commitment and on the body of the commitment. Informally, a tag-based definition requires that an adversary cannot equivocate a commitment com with a certain tag so long as it does not see the equivocation of any commitment with the same tag. On the other hand, a body-based definition requires that the adversary cannot equivocate a commitment com so long as the commitment com itself has not been equivocated. (Note that we use the term "body" to refer to the bit-string that is the commitment.) For brevity, a scheme secure according to the tag-based definition will be called a *tag-based scheme*, and a scheme secure according to the body-based definition will be called a *body-based scheme*.

In our paper, we choose to focus on tag-based schemes since they admit simpler constructions and seem to be the most appropriate for our applications. In particular, there exists a conversion from tag-based schemes to body-based ones with the addition of a one-time signature scheme. This is a rather common technique, and we discuss it later in the paper. We also show a rather general transformation from body-based schemes to tag-based ones. Furthermore, in constructing secure zero-knowledge protocols in the UC framework, where the communication is normally assumed to be authenticated, it is natural to use a tag-based scheme, setting the tag to be the pair of the identities of the prover and the verifier. In this way, one can avoid the overhead of the added one-time signature scheme caused by the tag-based scheme to body-based scheme conversion. We give more details later in this paper.

Efficient Constructions We present various constructions of SSTC schemes. The first construction is a generic one based on the (minimal) assumption that one-way functions exist. Our construction is similar to that of a UCC commitment scheme in Canetti *et. al.* [9]. However, because SSTC schemes do not require the extractability property, we are able to simplify the construction, and have it rely on a weaker assumption. The next two constructions are based on specific number-theoretic assumptions, namely the strong RSA assumption and the DSA assumption(see Appendix F). These two constructions are very efficient, both involving only a small constant number of public key operations. The construction based on DSA is similar to the one given by Garay *et. al.* [31], but is about twice as efficient.

Interestingly, all of our constructions are heavily based on signature schemes that are existentially unforgeable against adaptive chosen message attacks. We show that this is not a coincidence, in that there is a straightforward conversion of any SSTC scheme into a signature scheme.

Applications We show constructions of unbounded simulation-sound, unbounded non-malleable, and universally composable zero-knowledge (ZK) protocols using SSTC schemes in the common reference string (CRS) model. In particular, we show how to (1) convert a Σ-protocol [13] (which is a special three-round, honest-verifier protocol where the verifier only sends random bits) into an unbounded simulation-sound ZK protocol; and (2) convert an Ω-protocol [31] (which is a Σ-protocol with a straight-line extractor) into an unbounded non-malleable ZK protocol, and further into a universally-composable ZK protocol. The constructions are conceptually very simple. In fact, they all share the same structure, and all use a technique from Damgård [16] and Jarecki and Lysyanskaya [36]. The same technique was also used in Garay *et. al.* [31] in constructing a universally-composable ZK protocol that is secure against adaptive corruptions.

Our constructions are very efficient, and in particular our construction of a universally-composable ZK protocol is more efficient than previous constructions based on universally-composable commitment schemes [7, 9, 18]. First, we gain efficiency by using an SSTC scheme instead of a UCC scheme, since our most efficient SSTC constructions are more efficient than any known UCC constructions. For instance, the UCC constructions of [7, 9] are for bit commitments, and thus have an expansion factor of at least the security parameter. The UCC construction of [18] has constant expansion factor, but requires a CRS of length proportional to the number of parties times the security parameter.

Recently and independent from this work, Damgård and Groth [17] presented a UCC scheme with a constant expansion factor with a CRS whose length is independent of the number of parties. However, their scheme is still quite complicated, since it requires interaction, and uses two different types of commitments, one a non-malleable commitment scheme, and the other a special "mixed commitment scheme." Second, we gain efficiency by avoiding the Cook-Levin theorem [11, 38].[4]

The second idea was used by Garay *et. al.* [31], who observed that one can construct honest-verifier zero-knowledge protocols with very efficient straightline extractors for many natural problems. They called these Ω-protocols, and showed how to construct UCZK protocols from these Ω-protocols in the CRS model without using the Cook-Levin theorem, thus achieving very efficient constructions. Intuitively, they managed this by "shifting" the burden of extractability from the commitments to the underlying Ω-protocols. In particular, they used a technique involving signatures to convert an Ω-protocol into a UCZK protocol secure against static corruptions, and then they used an SSTC scheme (with a stronger definition than in this paper, as discussed above) to further convert the UCZK protocol secure against static corruptions into a UCZK protocol secure against adaptive corruptions. In this paper, we use an SSTC scheme (with the new definition introduced in this paper) to construct UCZK protocols secure against both static and adaptive corruptions in the CRS model. Compared to that in [31], our construction is simpler and more efficient. The savings are twofold: the simpler SSTC construction (with a weaker definition) cuts the overhead of SSTC by half, and the direct use of the tag-based scheme further eliminates the need for one-time signature schemes.

Relation to Non-malleable Commitments We discuss the relation between SSTC schemes and NMC schemes [22, 20, 21, 17].[5] At first glance, binding and non-malleability (or analogously, equivocation and malleability) seem like very different notions: while the former concerns the adversary's ability to open a commitment to multiple values, the latter concerns the adversary's ability to produce and open a commitment to a single value related to a previously committed value. However, they are actually closely related, and we shall show that simulation-sound binding implies non-malleability (when both are appropriately defined). In fact, a similar observation was used implicitly in [20, 21, 17] to construct NMC schemes. In particular, these NMC schemes are all based on trapdoor commitment schemes that satisfy a weak notion of simulation-sound binding. (Note that these results all use body-based definitions instead of tag-based definitions.) However, the *exact* relationship between the notions of simulation-sound binding and non-malleability was not known, e.g., if simulation-sound binding is strictly stronger than non-malleability, or if they are equivalent.

We study the exact relationship between these two notions in this paper. To do this, we need to resolve some technical issues. First, just as SSTC schemes can be tag-based or body-based, NMC schemes can also be tag-based or body-based, where a tag-based NMC scheme is informally defined as one in which seeing a commitment (to some value v) with a certain tag does not give an adversary any advantage in generating a new commitment with a different tag that can later be opened to a value related to v. Since we focus on tag-based SSTC schemes, we will focus on their relation to tag-based NMC schemes.[6] (Analogous results could be obtained for the relationship between body-based SSTC schemes and body-based NMC schemes.) Second, an SSTC scheme is a TC scheme, so to make a useful comparison, we consider non-malleable trapdoor commitment (NMTC) schemes. Third, since an adversary for an SSTC scheme is allowed to query an equivocation oracle, we will also consider NMTC schemes in which an adversary is allowed to query an equivocation oracle.

[4] In previous constructions, they build a UCZK protocol Π^L for an NP-complete language L (e.g. Hamiltonian Cycle or Satisfiability), and then the UCZK protocols for any NP language is reduced to Π^L via the Cook-Levin theorem, which is not very efficient.

[5] Technically, when we refer to an NMC scheme, we will always mean an ϵ-non-malleable commitment scheme, following the notation proposed in [22].

[6] Tag-based NMC schemes are also related to UCC schemes. In particular, it can be shown that a UCC scheme is also a tag-based NM commitment scheme in which the tag is the identity of the committing party.

Finally, we refine our definitions of SSTC schemes and NMTC schemes by specifying the number of equivocation oracle queries an adversary is allowed to make. An equivocation oracle, on a commit query, produces a commitment $\widetilde{\mathsf{com}}$ and on an decommit query, opens $\widetilde{\mathsf{com}}$ to an arbitrary value. We say a TC scheme is SSTC(ℓ), if it remains secure if the adversary is allowed to make at most ℓ commit queries to the oracle (with no restriction on the number of decommit queries). We define NMTC(ℓ) schemes similarly. We use SSTC(∞) and NMTC(∞) to denote the schemes where the adversary can make an unlimited number of commit queries. With the refined definitions (except for those related to the definition in [17], discussed below), we shall then prove that, for any constant ℓ, SSTC($\ell+1$) is strictly stronger than NMTC(ℓ) and NMTC(ℓ) is strictly stronger than SSTC(ℓ). (In particular, note that even an SSTC(1) scheme is strictly stronger than an NMC scheme, since an NMTC(0) scheme is at least as strong as an NMC scheme.) Furthermore, SSTC(∞) is equivalent to NMTC(∞). See Figure 1. This makes it clear that the two notions, simulation-sound binding and non-malleability, are very closely related.

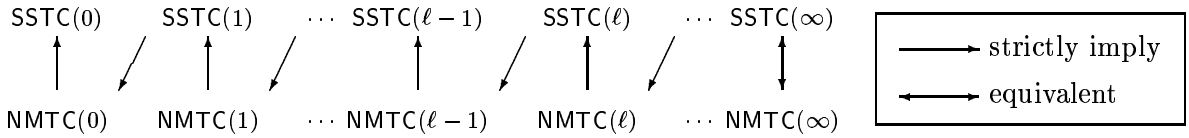

Figure 1: The relation between SSTC and NMTC schemes

As mentioned above, the definition of non-malleable commitments in Damgård and Groth [17] (which they call *reusable* non-malleable commitments) does not quite fit into the equivalence and separation results above. Their definition states that seeing one *or more* commitments does not give another party any advantage in generating one *or more* commitments that can later be opened to values related to the values in the original commitments. However it can be shown that SSTC(∞) implies a reusable NMC scheme. As mentioned above, one can characterize their construction of a reusable NMC scheme as constructing a trapdoor commitment schemes that satisfies a slightly weaker notion of simulation-sound binding, and showing that this implies a reusable NMC scheme.

2 Preliminaries and Definitions

For a distribution $\mathcal{D}$, we say $a \in \mathcal{D}$ to denote any element that has non-zero probability in $\mathcal{D}$, i.e., any element in the support of $\mathcal{D}$. We say $a \stackrel{R}{\leftarrow} \mathcal{D}$ to denote a is randomly chosen according to distribution $\mathcal{D}$. For a set S, we say $a \stackrel{R}{\leftarrow} S$ to denote that a is uniformly drawn from S.

If f and g are functions we say that f is *eventually less than* g, written $f \leq_{\text{ev}} g$, if there is an integer k_0 such that for all $k \geq k_0$, $f(k) \leq g(k)$. A function $\alpha(k)$ is *negligible*, if it is eventually less than k^{-c} for any positive c.

All our definitions will assume that adversaries are non-uniform probabilistic polynomial-time (PPT) algorithms.

Definition 2.1 *Two sequences $\{X_k\}$ and $\{Y_k\}$ of random variables are* computationally indistinguishable *if and only if there exists a negligible function $\alpha(k)$ such that for every non-uniform PPT $\mathcal{A}$,*

$$|\Pr(\mathcal{A}(X_k) = 1) - \Pr(\mathcal{A}(Y_k) = 1)| \leq_{\text{ev}} \alpha(k).$$

A commitment scheme is a two-phase protocol between a sender and a receiver, both probabilistic polynomial-time Turing machines, that operate as follows. In the commitment phase, the sender commits to a value v by computing a pair $(\mathsf{com}, \mathsf{dec})$ and sending com to the receiver, and in the decommitment phase, the sender reveals (v, dec) to the receiver, who checks whether the pair is valid.

Informally, a commitment scheme satisfies the hiding property, meaning that for any $v_1 \neq v_2$ of the same length, a commitment to v_1 is indistinguishable from a commitment to v_2, and the binding property, meaning that once the receiver receives c, the sender cannot open the commitment c to two different values, except with negligible probability.

We will always assume that commitments are labeled with a tag *tag*. While this is not a factor in the security of basic commitment schemes, it will be useful in defining certain enhanced commitment schemes, as will be obvious below. We also assume that there is a commitment generator function that generates a set of parameters for the commitment scheme. In other papers this is often referred to as a *trusted third party* or as the *common reference string* generation,[7] and it is especially important when we define trapdoor commitment schemes below. (We include it in the basic definition to more conveniently define trapdoor commitment schemes.)

Finally, for readability in our formal definitions, when we discuss distribution ensembles and negligible functions, we will often use the phrase "for all x" when we actually mean "for all sequences $\{x_k\}$," where x_k denotes a value of x dependent on the security parameter k, and of length polynomial in k. Formally, we define a commitment scheme as follows.

Definition 2.2 [Commitment Scheme] $\mathsf{CS} = (\mathsf{Cgen}, \mathsf{Ccom}, \mathsf{Cver})$ *is a* commitment scheme *if* Cgen, Ccom, *and* Cver *are probabilistic polynomial-time algorithms such that*

- **Completeness** *For all v and tag,*

$$\Pr[pk \leftarrow \mathsf{Cgen}(1^k); (\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Ccom}(pk, v, tag) : \mathsf{Cver}(pk, \mathsf{com}, v, tag, \mathsf{dec}) = 1] = 1.$$

- **Binding** *There is a negligible function $\alpha(k)$ such that for all non-uniform probabilistic polynomial-time adversaries $\mathcal{A}$,*

$$\Pr[pk \leftarrow \mathsf{Cgen}(1^k); (\mathsf{com}, tag, v_1, v_2, \mathsf{dec}_1, \mathsf{dec}_2) \leftarrow \mathcal{A}(pk) : (\mathsf{Cver}(pk, \mathsf{com}, v_1, tag, \mathsf{dec}_1) = \mathsf{Cver}(pk, \mathsf{com}, v_2, tag, \mathsf{dec}_2) = 1) \wedge (v_1 \neq v_2)] \leq_{\mathrm{ev}} \alpha(k).$$

- **Hiding** *For all pk generated with non-zero probability by $\mathsf{Cgen}(1^k)$, for all v_1, v_2 of equal length, and for all tag, the following probability distributions are computationally indistinguishable:*

$$\{(\mathsf{com}_1, \mathsf{dec}_1) \leftarrow \mathsf{Ccom}(pk, v_1, tag) : \mathsf{com}_1\} \text{ and } \{(\mathsf{com}_2, \mathsf{dec}_2) \leftarrow \mathsf{Ccom}(pk, v_2, tag) : \mathsf{com}_2\}.$$

Next, we define trapdoor commitment schemes.(We borrow some notation from Reyzin [45].)

Definition 2.3 [Trapdoor Commitment Scheme]
$\mathsf{TC} = (\mathsf{TCgen}, \mathsf{TCcom}, \mathsf{TCver}, \mathsf{TCfakeCom}, \mathsf{TCfakeDecom})$ *is a* trapdoor commitment scheme *if* $\mathsf{TCgen}(1^k)$ *outputs a public/secret key pair (pk, sk),* TCgen_{pk} *is the related function that restricts the output of* TCgen *to the public key,* $(\mathsf{TCgen}_{pk}, \mathsf{TCcom}, \mathsf{TCver})$ *is a commitment scheme and* $\mathsf{TCfakeCom}$ *and* $\mathsf{TCfakeDecom}$ *are probabilistic polynomial-time algorithms such that*

- **Trapdoor Property** *For all identifiers tag and values v, the following probability distributions are computationally indistinguishable:*

$$\{(pk, sk) \leftarrow \mathsf{TCgen}(1^k); (\widetilde{\mathsf{com}}, \xi) \leftarrow \mathsf{TCfakeCom}(pk, sk, tag); \widetilde{\mathsf{dec}} \leftarrow \mathsf{TCfakeDecom}(\xi, \widetilde{\mathsf{dec}}, v) : (pk, tag, v, \widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}})\}$$

and

$$\{(pk, sk) \leftarrow \mathsf{TCgen}(1^k); (\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{TCcom}(pk, v, tag) : (pk, tag, v, \mathsf{com}, \mathsf{dec})\}.$$

[7] We do not use the term "common reference string" in our definition, since these parameters may be generated in a number of ways, and in particular, they may be generated by the receiver. In protocols where this value actually comes from a common reference string, we will make this clear.

3 Simulation-Sound Trapdoor Commitments

In [31], simulation-sound trapdoor commitment (SSTC) schemes were introduced, in order to construct a universally-composable zero-knowledge (UCZK) protocol secure against adaptive corruptions. Intuitively, they defined an SSTC scheme as a trapdoor commitment scheme with a *simulation-sound binding* property that guarantees that a commitment made by the adversary using tag *tag* is binding, even if the adversary has seen any commitment using tag *tag* opened (using a simulator that knows a trapdoor) once to any arbitrary value, and moreover, any commitment using tag $tag' \neq tag$ opened (again using the simulator) an unbounded number of times to any arbitrary values.

Here we introduce a simpler definition for an SSTC scheme where the simulation-sound binding property is such that adversary can only succeed on a tag that has never been used in a commitment, rather than on a tag that has never been used in a commitment that has been decommitted in two different ways.[8] Since this can only reduce the success probability of the adversary, it is a weaker property. However, we will show that it also suffices for the desired application in [31], namely, for constructing UCZK protocols secure against adaptive adversaries.

Definition 3.1 [SSTC **Scheme**] $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{TCcom}, \mathsf{TCver}, \mathsf{TCfakeCom}, \mathsf{TCfakeDecom})$ *is an* SSTC scheme *if* TC *is a trapdoor commitment scheme such that*

- **Simulation-Sound Binding** *There is a negligible function $\alpha(k)$ such that for all non-uniform probabilistic polynomial-time adversaries $\mathcal{A}$,*

$$\Pr[(pk, sk) \leftarrow \mathsf{TCgen}(1^k); (\mathsf{com}, tag, v_1, v_2, \mathsf{dec}_1, \mathsf{dec}_2) \leftarrow \mathcal{A}^{\mathcal{O}_{pk,sk}}(pk) : (\mathsf{TCver}(pk, \mathsf{com}, v_1, tag, \mathsf{dec}_1) = \mathsf{TCver}(pk, \mathsf{com}, v_2, tag, \mathsf{dec}_2) = 1) \wedge (v_1 \neq v_2) \wedge tag \notin Q] \leq_{\text{ev}} \alpha(k),$$

where $\mathcal{O}_{pk,sk}$ operates as follows, with Q initially set to $\emptyset$:

- *On input* (commit, tag)*:*
  compute $(\widetilde{\mathsf{com}}, \xi) \leftarrow \mathsf{TCfakeCom}(pk, sk, tag)$*, store* $(\widetilde{\mathsf{com}}, tag, \xi)$*, and add tag to Q. Return* $\widetilde{\mathsf{com}}$*.*
- *On input* $(\mathsf{decommit}, \widetilde{\mathsf{com}}, v)$*:*
  if for some tag and some ξ, a tuple $(\widetilde{\mathsf{com}}, tag, \xi)$ *is stored, compute* $\widetilde{\mathsf{dec}} \leftarrow \mathsf{TCfakeDecom}(\xi, \widetilde{\mathsf{com}}, v)$*. Return* $\widetilde{\mathsf{dec}}$*.*

For the remainder of the paper, SSTC will refer to this new definition, and SSTC(GMY) will refer to the old definition of [31].

3.1 SSTC scheme based on any one-way function

Here we present an efficient SSTC scheme TC based on a signature scheme, which in turn may be based on any one-way function [46]. TC is the aHC scheme from Canetti *et al.* [9] with the following changes:

1. The underlying commitment scheme based on one-way permutations is replaced by the commitment scheme of Naor [40] based on pseudorandom generators (which can be built from any one-way function).

2. An extra parameter *tag* is included, and the one-way function f and corresponding NP language $\{y | \exists x \text{ s.t. } y = f(x)\}$ used in the underlying non-interactive Feige-Shamir trapdoor commitment [26] is replaced by the signature verification relation $\{((\mathsf{sig\_vk}, tag), \sigma) | 1 = \mathsf{sig\_verify}(\mathsf{sig\_vk}, tag, \sigma)\}$.

[8] Note that in addition to the simulation-sound binding property being modified, our definition of the underlying trapdoor commitment scheme is slightly different than the one given in [31].

In detail, the scheme goes as follows. $\mathsf{TCgen}(1^k)$ generates a verification/signing key pair for a signature scheme $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$. For a bit m, $\mathsf{TCcom}(\mathsf{sig\_vk}, m, tag)$ uses the NP-reduction of the relation $\{(\mathsf{sig\_vk}, tag) | \exists \sigma \text{ s.t. } 1 = \mathsf{sig\_verify}(\mathsf{sig\_vk}, tag, \sigma)\}$ to the Hamiltonicity relation, to obtain a graph G (with q nodes) so that finding a Hamiltonian cycle in G is equivalent to finding σ. Then it follows the aHC scheme of [9]:

- To commit to 0, pick a random permutation π of the nodes of G, and commit to the entries of the adjacency matrix of the permuted graph one by one, using *Com* (an underlying non-interactive perfectly-binding commitment scheme that produces pseudorandom commitments). To decommit, send π and decommit to every entry of the adjacency matrix. The receiver verifies that the graph it received is $\pi(G)$.

- To commit to 1, choose a randomly labeled q-cycle, and for all the entries in the adjacency matrix correspond to edges on the q-cycle, use *Com* to commit to 1 values. For all the other entries, produce random values. (These will be indistinguishable from commitments due to the pseudorandomness of the commitments.) To decommit, open only the entries corresponding to the randomly chosen q-cycle in the adjacency matrix.

$\mathsf{TCfakeCom}(\mathsf{sig\_vk}, \mathsf{sig\_sk}, tag)$ computes the graph G associated with (vk, tag), computes $\sigma = \mathsf{sig\_sign}(\mathsf{sig\_sk}, tag)$, and using σ finds a Hamiltonian cycle in G. Then it picks a random permutation π of the nodes of G, commit to the entries of the adjacency matrix of the permuted graph one by one, using *Com*, and sets $\xi \leftarrow (G, \mathrm{HC}(G))$.

$\mathsf{TCfakeDecom}(\xi, \mathsf{com}, v)$ runs as follows. If $v = 0$, it decommits using a normal decommitment to 0. If $v = 1$, it decommits using a normal decommitment to 1, using the Hamiltonian cycle $\mathrm{HC}(G)$ as the q-cycle.

To show the simulation-sound binding property, we show that if an adversary can break this property, we can break the underlying signature scheme as follows. (We assume that the underlying signature scheme is existentially unforgeable against an adaptive chosen-message attack.) Take a verification key $\mathsf{sig\_vk}$ and its corresponding signature oracle (from the definition of existential unforgeability against an adaptive chosen-message attack). For each commitment to a value v using tag', compute a signature σ on tag' using the signature oracle. From signature σ, one can compute a Hamiltonian cycle in G, and thus run $\mathsf{TCfakeCom}$ as above (except using the signature oracle to compute σ) to produce a commitment com. To open a commitment c to a value m, run $\mathsf{TCfakeDecom}$ as above.

Now say the adversary gives a double opening with tag, for which no commitment was requested, and thus no call to the signature oracle was made. In particular, say the adversary decommits to 0 and 1 for a commitment com. Then one can extract a Hamiltonian cycle in G, and thus a signature on tag, breaking the signature scheme.

3.2 SSTC scheme based on DSA

Here we present an efficient SSTC scheme TC based on DSA. For a definition of DSA, see Appendix F. It is a simplified version of the DSA-based SSTC(GMY) scheme from [31]. $\mathsf{TCgen}(1^k)$ generates a DSA public/private key pair (pk, sk), where $pk = (g, p, q, y)$ and $sk = (g, p, q, x)$. For a message $m \in \mathbb{Z}_q$, $\mathsf{TCcom}((g, p, q, y), m, tag)$ first computes $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_q$, $g' \leftarrow g^\alpha \bmod p$, and $h = g^{H(tag)} y^{g'} \bmod p$. (Note that if s is the discrete log of h over g', then $(g' \bmod q, s)$ is a DSA signature for tag.) Then it generates a Pedersen commitment [44] to m over bases (g', h), i.e., it generates $\beta \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and computes the commitment/decommitment pair $((g', c), (m, \beta))$, where $c \leftarrow (g')^\beta h^m$. $\mathsf{TCfakeCom}((g, p, q, y), (g, p, q, x), tag')$ computes a DSA signature (g'', s) on tag' using the secret key, computes $g' \leftarrow (g^{H(tag')} y^{g''})^{s^{-1}} \bmod p$ and $h \leftarrow (g')^s \bmod p$, generates $\beta' \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and sets $c \leftarrow h^{\beta'} \bmod p$. It outputs commitment (g', c) and

auxiliary information (β', s). Then $\mathsf{TCfakeDecom}((\beta', s), m)$ outputs $(m, (\beta' - m)s \bmod q)$, which is a decommitment to m.

To show the simulation-sound binding property, we show that if an adversary can break this property, we can break DSA as follows. (We assume that DSA is existentially unforgeable against an adaptive chosen-message attack.) Take a DSA key vk_0 and its corresponding DSA signature oracle (from the definition of existential unforgeability against an adaptive chosen-message attack). It is easy to see that the equivocation oracle, and in particular the commit queries to that oracle, may be implemented using the DSA signature oracle on the requested tag's.

Now say the adversary gives a double opening with tag, for which no commitment was requested, and thus no call to the DSA signature oracle was made. In particular, say it gives openings (m, β) and (m', β') of (g', c). Then $(g' \bmod q, (\beta' - \beta)/(m - m') \bmod q)$ is a signature on tag, breaking DSA.

3.3 SSTC scheme based on Cramer-Shoup signatures

Here we present an efficient SSTC scheme TC based on Cramer-Shoup signatures (for a definition of Cramer-Shoup signatures, see Appendix F).

$\mathsf{TCgen}(1^k)$ generates a public/private key pair (pk, sk) for Cramer-Shoup signatures, where $pk = (N, h, x, e', H)$ and $sk = (p, q)$. For a message $m \in \mathbb{Z}_e$, $\mathsf{TCcom}((N, h, x, e', H), m, tag)$ first computes (y', x', e) as in the Cramer-Shoup signature protocol for tag, and sets $x'' \leftarrow xh^{-H(x')} \bmod N$. (Note that if y is eth root of x'' modulo N, then $\langle e, y, y' \rangle$ is a Cramer-Shoup signature for tag.) Then it uses the unconditionally-hiding commitment scheme from [12] based on e-one-way homomorphisms (specifically, based on the RSA encryption function with public key (e, N), i.e., $f(a) : a^e \bmod N$) over base x''. That is, it chooses $\beta \stackrel{R}{\leftarrow} \mathbb{Z}_N^*$ and computes the commitment/decommitment pair $((y', e, c), (m, \beta))$, where $c \leftarrow (x'')^m f(\beta) \bmod N$. $\mathsf{TCfakeCom}((N, h, x, e', H), (p, q), tag')$ computes a signature $\langle e, y, y' \rangle$ on tag' using the secret key, computes $x' \leftarrow (y')^{e'} h^{-H(tag')} \bmod N$ and $x'' \leftarrow xh^{-H(x')} \bmod N$, generates $\beta' \stackrel{R}{\leftarrow} \mathbb{Z}_N^*$, and sets $c \leftarrow (\beta')^e \bmod N$. It outputs commitment (y', e, c) and auxiliary information (β', y). Then $\mathsf{TCfakeDecom}((\beta', y), m)$ output $(m, \beta' y^{-m} \bmod N)$, which is a decommitment to m.

To show the simulation-sound binding property, we show that if an adversary can break this property, we can break the Cramer-Shoup signature scheme as follows. (We assume that Cramer-Shoup signatures are existentially unforgeable against an adaptive chosen-message attack.) Take a Cramer-Shoup key vk_0 and its corresponding signature oracle (from the definition of existential unforgeability against an adaptive chosen-message attack). It is easy to see that the equivocation oracle, and in particular the commit queries to that oracle, may be implemented using the Cramer-Shoup signature oracle on the requested tag's.

Now say the adversary gives a double opening with tag, for which no commitment was requested, and thus no call to the signature oracle was made. In particular, say it gives openings (m, β) and (m', β') of (y', e, c) with $m > m'$. Then $(x'')^{m-m'} \equiv (\beta' \beta^{-1})^e \bmod N$ and by e-one-wayness of the RSA encryption function, the value y such that $y^e \equiv x \bmod N$ may be computed. and $\langle e, y, y' \rangle$ is a signature on tag, breaking Cramer-Shoup.

SSTC Signatures All three of our previous constructions of SSTC schemes are heavily based on signature schemes. In fact, this is not a coincidence, since one can easily derive a digital signature scheme from any SSTC scheme, as the next theorem demonstrates. Intuitively, to sign a message m, one exhibits the ability to open a commitment with label m to both the message 0 and the message 1.

More precisely, let $\mathsf{SIG_{TC}} = (\mathsf{sig\_gen_{TC}}, \mathsf{sig\_sign_{TC}}, \mathsf{sig\_verify_{TC}})$ be specified as follows.

- $\mathsf{sig\_gen_{TC}}(1^k)$ computes $(pk, sk) \leftarrow \mathsf{TCgen}(1^k)$, sets $\mathsf{sig\_vk} = pk$, $\mathsf{sig\_sk} = (pk, sk)$, and outputs $(\mathsf{sig\_vk}, \mathsf{sig\_sk})$.

- $\mathsf{sig\_sign}((pk, sk), m)$ generates $(\widetilde{\mathsf{com}}, \xi) \leftarrow \mathsf{TCfakeCom}(pk, sk, m)$, $\widetilde{\mathsf{dec}}_0 \leftarrow \mathsf{TCfakeDecom}(\xi, \widetilde{\mathsf{com}}, 0)$, and $\widetilde{\mathsf{dec}}_1 \leftarrow \mathsf{TCfakeDecom}(\xi, \widetilde{\mathsf{com}}, 1)$, and then outputs $(\widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}}_0, \widetilde{\mathsf{dec}}_1)$.
- $\mathsf{sig\_verify}(pk, m, (\widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}}_0, \widetilde{\mathsf{dec}}_1))$ outputs $\mathsf{TCver}(pk, \widetilde{\mathsf{com}}, 0, m, \widetilde{\mathsf{dec}}_0) \wedge \mathsf{TCver}(pk, \widetilde{\mathsf{com}}, 1, m, \widetilde{\mathsf{dec}}_1)$.

Theorem 3.2 *Given an SSTC* TC, $\mathsf{SIG}_{\mathsf{TC}}$ *is a signature scheme that is existentially unforgeable against an adaptive chosen message attack.*

Proof: Say a forger $\mathcal{F}$, given public key $\mathsf{sig\_vk} = pk$ and a signature oracle, is able to forge a signature in $\mathsf{SIG}_{\mathsf{TC}}$. Then we give an adversary $\mathcal{A}$ that breaks the simulation-sound binding property of the TC as follows. $\mathcal{A}$ takes a TC public key pk and an oracle $\mathcal{S}$, gives $\mathcal{F}$ $\mathsf{sig\_vk} = pk$ as the public key of $\mathsf{SIG}_{\mathsf{TC}}$ and plays the part of the signature oracle by running the $\mathsf{sig\_sign}$ procedure, but using $\mathcal{S}$ to generate commitments and decommitments. Since $\mathcal{F}$ breaks $\mathsf{SIG}_{\mathsf{TC}}$, it produces a message m and a signature $(\widetilde{\mathsf{com}}, \widetilde{\mathsf{dec}}_0, \widetilde{\mathsf{dec}}_1)$ for some "fresh" m with non-negligible probability. Then $\mathcal{A}$ outputs $(\widetilde{\mathsf{com}}, m, 0, 1, \widetilde{\mathsf{dec}}_0, \widetilde{\mathsf{dec}}_1)$ with non-negligible probability, where $\mathsf{TCver}(vk, \widetilde{\mathsf{com}}, 0, m, \widetilde{\mathsf{dec}}_0) = \mathsf{TCver}(vk, \widetilde{\mathsf{com}}, 1, m, \widetilde{\mathsf{dec}}_1) = 1$, and m (as a tag) was not used in any commit queries to $\mathcal{S}$. This contradicts the simulation-sound binding property of TC. ∎

4 Application to ZK proofs

We show how an SSTC scheme can be used to construct unbounded simulation-sound ZK protocols, unbounded non-malleable ZK protocols, and universally composable ZK protocols. Our constructions are conceptually simpler than those given by Garay *et al.* [31].

All our results will be in the *common reference string* (CRS) model, which assumes that there is a string uniformly generated from some distribution and is available to all parties at the start of a protocol. Note that this is a generalization of the *public random string* model, where a uniform distribution over fixed-length bit strings is assumed.

4.1 Unbounded Simulation Sound ZK

Intuitively, a ZK protocol is unbounded simulation sound if an adversary cannot convince the verifier of a false statement with non-negligible probability, even after interacting with an arbitrary number of (simulated) provers. We use the formal definition from [31], and present this definition in Appendix A for completeness.

Our construction starts with a class of three-round, public-coin, honest-verifier zero-knowledge protocols, also known as Σ-protocols [13]. We briefly describe Σ-protocols here and defer the formal definitions to Appendix B.

Consider a binary relation $R(x, w)$ that is computable in polynomial time. A Σ-protocol Π for the relation R proves membership of x in the language $L_R = \{x \mid \exists\, w,\ s.t.\ R(x, w) = 1\}$. For a given x, let (a, c, z) denote the conversation between the prover and the verifier. To compute the first and the final messages, the prover invokes efficient algorithms $a_\Pi(x, w, r)$ and $z_\Pi(x, w, r, c)$, respectively, where w is the witness, r is the random bits, and c is the challenge from the verifier (as the second message). Using an efficient predicate $\phi(x, a, c, z)$, the verifier decides whether the conversation is accepting with respect to x. The relation R, and the algorithms $a(\cdot)$, $z(\cdot)$ and $\phi(\cdot)$, are public.

We assume the protocol Π has a simulator $\mathcal{S}_\Pi$ that, taking the challenge as input, generates an accepting conversation. More precisely, we have $(a, c, z) \leftarrow \mathcal{S}_\Pi(c)$, such that the distribution of (a, c, z) is computationally indistinguishable from the real conversation.

The protocol $\mathsf{USS}^R_{[pk]}(x)$ is shown in Figure 1, and uses an SSTC scheme TC. Say Π is a Σ-protocol for relation R. The prover generates a pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk})$ for a strong one-time signature scheme

and sends $\mathsf{sig\_vk}$ to the verifier. Then the prover generates the first message a of Π and sends its commitment com_a to the verifier, using the signature verification key $\mathsf{sig\_vk}$ as the commitment *tag*. After receiving the challenge c, the prover generates and sends the third message z of Π, opens the commitment com_a, signs the entire transcript using the signing key $\mathsf{sig\_sk}$, and sends the signature on the transcript to the verifier. (To be specific, the *transcript* consists of all values sent or received by the prover in the protocol, except the final signature.)

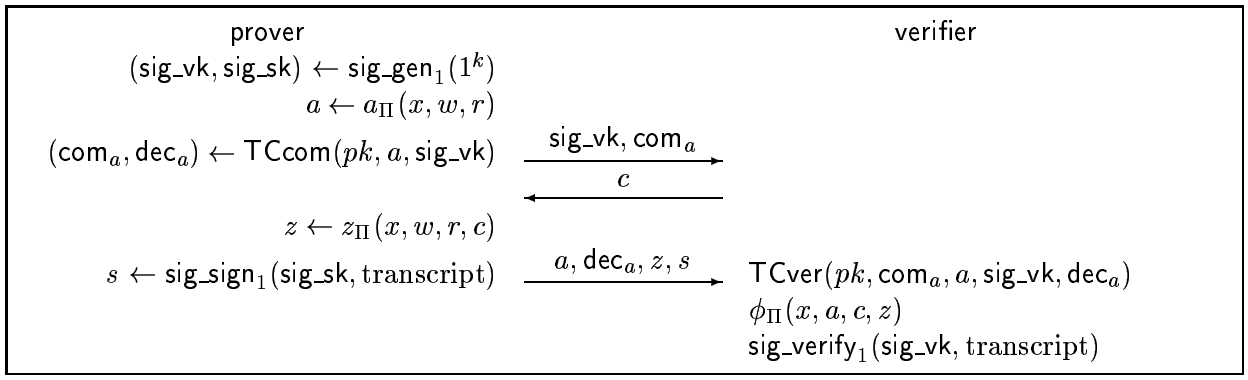

Figure 2: $\mathsf{USS}^R_{[pk]}(x)$: An unbounded simulation-sound ZK protocol for relationship R with common input x and common reference string pk, where pk is drawn from the distribution $\mathsf{TCgen}(1^k)$. The prover also knows the witness w such that $R(x, w) = 1$.

Now we describe the simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for protocol $\mathsf{USS}^R_{[pk]}(x)$. $\mathcal{S}_1(1^k)$ generates a key pair of the SSTC scheme by invoking $(pk, sk) \leftarrow \mathsf{TCgen}(1^k)$, and then outputs (pk, sk). The behavior of $\mathcal{S}_2(sk)$ is more involved. On input x, it first checks if $x \in \hat{L}_R$ and aborts if not. Then, it generates a strong one-time signature key pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk})$ as the prover. Next, $\mathcal{S}_2(sk)$ fakes a commitment by generating $(\widetilde{\mathsf{com}}, \xi) \leftarrow \mathsf{TCfakeCom}(pk, sk, \mathsf{sig\_vk})$ and sends $\widetilde{\mathsf{com}}$ to the verifier. On receiving the challenge c, it uses the simulator of protocol Π to compute an accepting conversation: $(a, c, z) \leftarrow \mathcal{S}_\Pi(c)$. Next, $\mathcal{S}_2$ generates a decommitment to a by setting $\widetilde{\mathsf{dec}} \leftarrow \mathsf{TCfakeDecom}(\xi, \widetilde{\mathsf{com}}, \mathsf{sig\_vk}, a)$ and signs the transcript using the strong one-time signature scheme; let s be the signature. Finally $\mathcal{S}_2$ sends over $(a, \widetilde{\mathsf{dec}}, z, s)$ as the third message.

Theorem 4.1 *The protocol* $\mathsf{USS}^R_{[pk]}(x)$ *is a USSZK argument.*

The proof is postponed to Appendix D.

4.2 Unbounded Non-malleable ZK

Intuitively, a ZK protocol is unbounded non-malleable if an efficient witness extractor successfully extracts a witness from any adversary that causes the verifier to accept, even when the adversary is also allowed to interact with any number of (simulated) provers. We use the formal definition from [31] and present this definition in Appendix A for completeness.

Our construction of the NMZK protocol is very similar to that of the USSZK protocol presented above, where the only difference is that the Σ-protocol is replaced by an Ω-protocol. Recall that an Ω-protocol [31] is like a Σ-protocol with the additional property that it admits a polynomial-time, straight-line extractor (an Ω-protocol works in the CRS model). A bit more formally, there exists a pair of polynomial-time algorithms $(\mathcal{E}_1, \mathcal{E}_2)$ with the following properties. $\mathcal{E}_1$ generates a pair (σ, τ): $(\sigma, \tau) \leftarrow \mathcal{E}_1$, where σ is a "simulated CRS" that is computationally indistinguishable from the real distribution and τ is the "backdoor information". $\mathcal{E}_2$ will produce a "potential-witness" $\tilde{w}$ from the

backdoor information τ and an accepting conversation (a, c, z): $\tilde{w} \leftarrow \mathcal{E}_2(x, \tau, (a, c, z))$. Furthermore, we have the property that the potential-witness $\tilde{w}$ is indeed a witness if there exists another accepting conversation (a, c', z') with the same first-message, but different challenges. We include the formal definitions in Appendix B.

The protocol $\mathsf{NM}^R_{[pk,\sigma]}(x)$ is shown in Figure 3. It is very similar to the protocol in Figure 2, but note that here we assume that Π is an Ω-protocol with σ being the CRS.

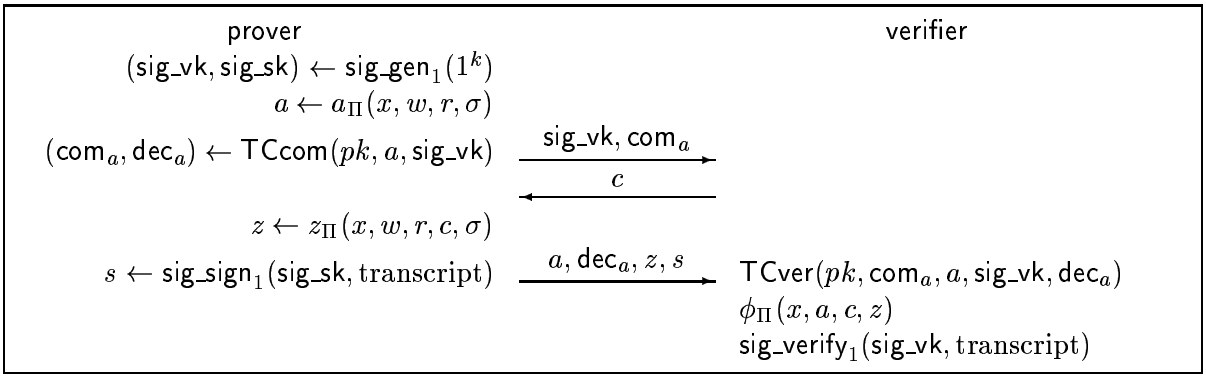

Figure 3: $\mathsf{NM}^R_{[pk,\sigma]}(x)$: A non-malleable ZK protocol for relationship R with common input x and common reference string (pk, σ), where pk is drawn from the distribution $\mathsf{TCgen}(1^k)$ and σ is drawn from the distribution of the CRS for protocol Π.

The simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for protocol $\mathsf{NM}^R_{[pk,\sigma]}(x)$ works almost exactly the same as in protocol $\mathsf{USS}^R_{[pk]}(x)$. $\mathcal{S}_1(1^k)$ generates a key pair of the SSTC scheme by invoking $(pk, sk) \leftarrow \mathsf{TCgen}(1^k)$, and then sets $\sigma \stackrel{R}{\leftarrow} \mathcal{D}_k$, where $\mathcal{D}$ is the distribution ensemble for the CRS of protocol Π. Next, $\mathcal{S}_1(1^k)$ outputs $((pk, \sigma), sk)$. $\mathcal{S}_2(sk)$ first checks that common input $x \in \hat{L}_R$ and aborts if not. Then it generates a strong one-time signature key pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk})$ as the prover. Next, $\mathcal{S}_2$ generates $(\widetilde{\mathsf{com}}, \xi) \leftarrow \mathsf{TCfakeCom}(pk, sk, \mathsf{sig\_vk})$, and sends $\widetilde{\mathsf{com}}$ to the verifier. On receiving the challenge c, it uses the simulator of protocol Π to compute an accepting conversation: $(a, c, z) \leftarrow \mathcal{S}_\Pi(c)$. Next, $\mathcal{S}_2$ generates a decommitment to a by setting $\widetilde{\mathsf{dec}} \leftarrow \mathsf{TCfakeDecom}(\xi, \widetilde{\mathsf{com}}, \mathsf{sig\_vk}, a)$ and signs the transcript using the strong one-time signature scheme; let s be the signature. Finally $\mathcal{S}_2$ sends over $(a, \widetilde{\mathsf{dec}}, z, s)$ as the third message.

The extractor $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ for protocol $\mathsf{NM}^R_{[pk,\sigma]}(x)$ is straightforward. $\mathcal{E}_1(1^k)$ generates a key pair of the SSTC scheme by invoking $(pk, sk) \leftarrow \mathsf{TCgen}(1^k)$, and then generates $(\sigma, \tau) \leftarrow \mathcal{E}_{\Pi,1}(1^k)$. Next, $\mathcal{E}_1(1^k)$ outputs $((pk, \sigma), sk, \tau)$. $\mathcal{E}_2(\tau)$ simply runs as the verifier $\mathcal{V}$ until $\mathcal{V}$ outputs a bit b. If $b = 1$, then $\mathcal{E}_2(\tau)$ takes the conversation (a, c, z) of protocol Π and invokes the extractor for protocol Π: $w \leftarrow \mathcal{E}_{\Pi,2}(x, \tau, (a, c, z))$; if $b = 0$, then $\mathcal{E}_2(\tau)$ sets $w \leftarrow \perp$. Finally $\mathcal{E}_2(\tau)$ outputs (b, w).

Theorem 4.2 *The protocol $\mathsf{NM}^R_{[pk,\sigma]}(x)$ is an NMZK argument of knowledge for the relation R.*

The proof to this theorem is very similar to that to Theorem 4.1 and is postponed to Appendix D.

4.3 Universally Composable ZK

The universal composability paradigm was proposed by Canetti [6] for defining the security and composition of protocols. To define security one first specifies an *ideal functionality* using a trusted party that describes the desired behavior of the protocol. Then one proves that a particular protocol operating in a real-life model securely realizes this ideal functionality, as defined below. Here we briefly summarize the framework as defined in Canetti and Krawczyk [8].

A (real-life) protocol π is defined as a set of n interactive Turing Machines $P_1, \ldots, P_n$, designating the n parties in the protocol. It operates in the presence of an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$, both of which are also modeled as interactive Turing Machines. The environment $\mathcal{Z}$ provides inputs and receives outputs from honest parties, and may communicate with $\mathcal{A}$. $\mathcal{A}$ controls (and may view) all communication between the parties. (Note that this models asynchronous communication on open point-to-point channels.) We will assume that messages are authenticated, and thus $\mathcal{A}$ may not insert or modify messages between honest parties.[9] $\mathcal{A}$ also may corrupt parties, in which case it obtains the internal state of the party. (In the non-erasing model, the internal state would encompass the complete internal history of the party.)

The ideal process with respect to a functionality $\mathcal{F}$, is defined for n parties $P_1, \ldots, P_n$, an environment $\mathcal{Z}$, and an (ideal-process) adversary $\mathcal{S}$. However, $P_1, \ldots, P_n$ are now dummy parties that simply forward (over secure channels) inputs received from $\mathcal{Z}$ to $\mathcal{F}$, and forward (again over secure channels) outputs received from $\mathcal{F}$ to $\mathcal{Z}$. Thus the ideal process is a trivially secure protocol with the input-output behavior of $\mathcal{F}$.

More details are given in Appendix C.

The zero-knowledge functionality. The (multi-session) ZK functionality as defined by Canetti [6] is given in Figure 4. In the functionality, parameterized by a relation R, the prover sends to the functionality the input x together with a witness w. If $R(x, w)$ holds, then the functionality forwards x to the verifier. As pointed out in [6], this is actually a proof of knowledge in that the verifier is assured that the prover actually knows w.

$\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$ proceeds as follows, running parties $P_1, \ldots, P_n$, and an adversary $\mathcal{S}$:

- Upon receiving $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, x, w)$ from P_i: If $R(x, w)$ then send $(\mathsf{ZK\text{-}PROOF}, sid, ssid, P_i, P_j, x)$ to P_j and $\mathcal{S}$. Otherwise, ignore.

Figure 4: The (multi-session) zero-knowledge functionality (for relation R)

Garay *et al.* [31] proved that any "augmentable" NMZK protocol can be easily converted to a UCZK protocol in the $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$-hybrid model, assuming static corruptions. Intuitively, an NMZK protocol is augmentable if the first message sent by the prover contains the common input x and a special field aux in which the prover can fill with an arbitrary string without compromising security. (In the conversion to UCZK in [31], the auxiliary string contains the sid, the $ssid$, and the identities of the prover and verifier.)

It can be readily verified that the protocol $\mathsf{NM}_{[pk,\sigma]}^{R}(x)$ can be easily made augmentable by adding x and aux in the first message. We denote the slightly modified protocol where the aux field is set to $(sid, ssid, P_i, P_j)$ by $\mathsf{ANM}_{[pk,\sigma]}^{R}(x)$. Then it follows that $\mathsf{ANM}_{[pk,\sigma]}^{R}(x)$ is a UCZK protocol for relation R, assuming static corruptions.

However, one can simplify this protocol by removing the one-time signature scheme, only including the identities of the prover and verifier in the auxiliary string, and using this auxiliary string as the tag of the commitment scheme. This simplified scheme, $\mathsf{MYZK}_{[pk,\sigma]}^{R}(x)$, is shown in Figure 5. (Note that since we are assuming authenticated communication in the UC framework, the identities P_i and P_j will be known to both parties, and thus do not need to be explicitly sent in our protocol.) Furthermore, this protocol can be easily modified into one that remains secure against adaptive corruption in the erasing model. In fact, all that is needed is to have the prover erase the randomness used in the Ω-protocol before sending the final message.

[9] This feature could be added to an unauthenticated model using a message authentication functionality as described in [6].

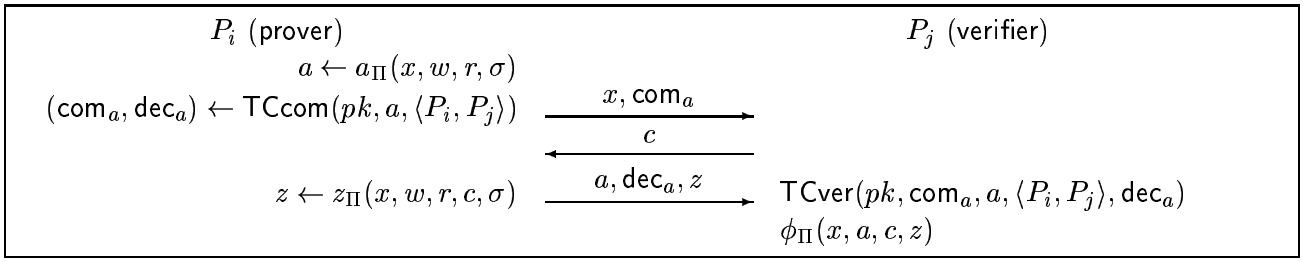

Figure 5: $\mathsf{MYZK}^R_{[pk,\sigma]}(x)$: A UCZK protocol for relationship R with common reference string (pk, σ) where pk is drawn from the distribution $\mathsf{TCgen}(1^k)$ and σ is drawn from the distribution of the CRS for protocol Π.

Theorem 4.3 *The protocol $\mathsf{MYZK}^R_{[pk,\sigma]}(x)$ is a UCZK protocol for relation R, assuming static corruptions. By erasing the randomness (r) used in the Ω-protocol before the final message, it is a UCZK protocol for relation R, assuming adaptive corruption (in the erasing model).*

The proof is postponed to Appendix D.

5 Comparison to Non-Malleable Commitments

We explore the exact relation between SSTC schemes and NMC schemes.

5.1 Definitions of NM commitments

Our definition for non-malleable (NM) commitments is based on the definition in [21], which, technically speaking, defines the notion of ϵ-non-malleability, instead of strict non-malleability. For the clarity of presentation, we shall use the term "non-malleability" to mean ϵ-non-malleability, and will note any places where our results have application to strict non-malleability.

Informally, similar to the definition in [21], we say a commitment scheme is non-malleable if when an adversary sees a commitment com_1, generates its own commitment com_2, and sees com_1 opened, it cannot then open com_2 to a value related to com_1 with any greater probability than a simulator that never saw com_1 in the first place. Note that this is also called *non-malleability with respect to opening* [20] and differs from the original definition of [22] that was discussed in the introduction, and which is also called *non-malleability with respect to commitment*. Our definition differs from the definition in [21] as follows.

- We only define NM *trapdoor* commitment (NMTC) schemes, since that is what will be of most interest in comparisons to SSTC schemes. Non-trapdoor versions of these definitions are straightforward.
- We use tag-based definitions instead of body-based definitions. Again this is what will be of most interest in comparisons to SSTC schemes. Body-based definitions are straightforward. In fact, most of our results relating SSTC schemes and NMTC schemes also hold when these schemes are defined using body-based definitions. We will discuss this later.

As mentioned in the introduction, the recent work of Damgård and Groth [17] generalizes and strengthens the definition of non-malleable commitments to be reusable, i.e., to have the property that seeing one *or more* commitments does not give another party any advantage in generating one *or more* commitments that can later be opened to values related to the values in the original commitments.

Their definition also stipulates that the distribution of committed messages is dependent on the public key. However, we will continue to use the simpler definition, since it exemplifies the relation between SSTC schemes and NMTC schemes. Later we will discuss how to obtain similar relations to reusable NMTC schemes.

In the following we assume tags are strings of length polynomial in the security parameter k.

Definition 5.1 [Non-Malleable Trapdoor Commitment (NMTC) Scheme] $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{TCcom}, \mathsf{TCver}, \mathsf{TCfakeCom}, \mathsf{TCfakeDecom})$ *is an* NMTC *scheme if* TC *is a trapdoor commitment scheme with the following property:*

Non-Malleability *There exists a negligible function $\alpha(\cdot)$ such that for all polynomials $r(\cdot)$ and all probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a polynomial $q(\cdot,\cdot)$, such that for all non-negligible[10] $\epsilon > 0$, there exists a simulator $\mathcal{S}$ running in time $q(k, \epsilon^{-1})$ such that for all polynomial-time computable valid relations R (see below), for all tags tag_1, and all distributions $\mathcal{D}$ samplable in time $r(k)$,*

$$\pi_{\mathcal{A},tag_1,\mathcal{D},R}(k) - \pi'_{\mathcal{S},tag_1,\mathcal{D},R}(k) \leq_{\text{ev}} \epsilon + \alpha(k),$$

where

$$\begin{aligned}\pi_{\mathcal{A},tag_1,\mathcal{D},R}(k) \stackrel{\text{def}}{=} \Pr[&(pk, sk) \leftarrow \mathsf{TCgen}(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; (\mathsf{com}_1, \mathsf{dec}_1) \leftarrow \mathsf{TCcom}(pk, m_1, tag_1);\\ &(\mathsf{com}_2, tag_2, \xi) \leftarrow \mathcal{A}_1(pk, \mathsf{com}_1, tag_1, \mathcal{D}); (m_2, \mathsf{dec}_2) \leftarrow \mathcal{A}_2(pk, \xi, m_1, \mathsf{dec}_1) :\\ &(\mathsf{TCver}(pk, \mathsf{com}_2, m_2, tag_2, \mathsf{dec}_2) = 1) \wedge (tag_1 \neq tag_2) \wedge R(m_1, m_2)]\end{aligned}$$

and

$$\pi'_{\mathcal{S},tag_1,\mathcal{D},R}(k) \stackrel{\text{def}}{=} \Pr[m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathcal{S}(1^k, tag_1, \mathcal{D}) : R(m_1, m_2)].$$

A relation R is valid if for all m, $R(m, \perp) = 0$.

Remark 5.2 *As in the definitions of [20, 21, 17], our definition does not allow the adversary to receive any history (side information) about the messages to which commitments are made.*

We generalize the definition above and consider $\mathsf{NMTC}(\ell)$ schemes, which are NMTC schemes in which $\mathcal{A}_1$ and $\mathcal{A}_2$ are allowed to query an oracle $\mathcal{O}_{pk,sk}$ as defined in the SSTC definition, but with at most ℓ commit queries allowed. (Note that there is just one oracle that both $\mathcal{A}_1$ and $\mathcal{A}_2$ call, and thus at most a total of ℓ commit queries between them.) Also the condition in the definition of $\pi_{\mathcal{A},tag_1,\mathcal{D},R}(k)$ is restricted to $tag_2 \notin Q$, where Q is the list of tags used in commit queries to $\mathcal{O}_{pk,sk}$. Note that an NMTC scheme is an $\mathsf{NMTC}(0)$ scheme. We use $\ell = \infty$ to denote an oracle which accepts an unbounded number of commit queries.

We similarly generalize the definition of SSTC schemes and consider $\mathsf{SSTC}(\ell)$ schemes. Then an $\mathsf{SSTC}(0)$ scheme is just a TC scheme, and an $\mathsf{SSTC}(\infty)$ scheme is what we have called an SSTC scheme.

Notice that we have defined NMTC schemes as tag-based, as opposed to body-based, as usually seen in literature [22, 20, 29, 21, 17]. As we have explained in the introduction, this is because we defined our SSTC schemes to be tag-based as well. However, this is not a significant distinction since there exists fairly generic reductions from one to the other. Our next theorem shows such a reduction from body-based NMTC schemes to tag-based ones.

Here, we assume the commitment scheme allows commitments to strings of arbitrary length. A similar theorem could be shown for commitment schemes which allow only fixed length commitments, say of length equal to the security parameter.

[10] In other words, ϵ may be a function of k such that ϵ^{-1} is bounded by a polynomial in k.

Theorem 5.3 *Let* TC *be a body-based* NMTC *scheme. Let* TC' *be* TC*, but with the tag added to the message being committed. That is,* $\mathsf{TCgen}'(1^k)$ *returns the result of* $\mathsf{TCgen}(1^k)$*,* $\mathsf{TCcom}'(pk, v, tag)$ *returns the result of* $\mathsf{TCcom}(pk, \langle v, tag\rangle, tag)$*, and* $\mathsf{TCver}'(pk, \mathsf{com}, v, tag, \mathsf{dec})$ *returns the result of* $\mathsf{TCver}(pk, \mathsf{com}, \langle v, tag\rangle, tag, \mathsf{dec})$*. Then* TC' *is a tag-based* NMTC *scheme.*

Proof: This proof relies on the binding property of TC, as well as the non-malleability property. Take any $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for TC'. Then construct $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for TC as follows. $\mathcal{A}_1(pk, \mathsf{com}_1, tag_1)$ computes $(\mathsf{com}_2, tag_2, \xi) \leftarrow \mathcal{A}'_1(pk, \mathsf{com}_1, tag_1)$ and returns $(\mathsf{com}_2, tag_2, (\xi, tag_2))$. $\mathcal{A}_2(pk, (\xi, tag_2), m_1, \mathsf{dec}_1)$ computes $(m_2, \mathsf{dec}_2) \leftarrow \mathcal{A}'_2(pk, \xi, m_1, \mathsf{dec}_1)$ and returns $(\langle m_2, tag_2\rangle, \mathsf{dec}_2)$. Take the simulator $\mathcal{S}$ guaranteed to exist by the non-malleability of TC. Then we construct a simulator $\mathcal{S}'$ for TC' as follows. $\mathcal{S}'(1^k, tag_1, \mathcal{D}')$ computes $\langle m_2, tag_2\rangle \leftarrow \mathcal{S}(1^k, tag_1, \mathcal{D})$ and returns m_2, where $\mathcal{D} = \{\langle m, tag_1\rangle : m \leftarrow \mathcal{D}'\}$. Now take any distribution $\mathcal{D}'$, and any relation R'. Let $\mathcal{D}$ be constructed as above, and define $R(\langle m_1, tag_1\rangle, \langle m_2, tag_2\rangle) = R'(m_1, m_2)$. Then by the non-malleability of TC,

$$\pi_{\mathcal{A}, tag_1, \mathcal{D}, R}(k) - \pi'_{\mathcal{S}, tag_1, \mathcal{D}, R}(k) \leq_{\text{ev}} \alpha(k).$$

It is easy to verify that $\pi'_{\mathcal{S}, tag_1, \mathcal{D}, R}(k) = \pi'_{\mathcal{S}', tag_1, \mathcal{D}', R'}(k)$, so to prove the theorem we only need to show that $\pi_{\mathcal{A}', tag_1, \mathcal{D}', R'}(k) - \pi_{\mathcal{A}, tag_1, \mathcal{D}, R}(k)$ is negligible. (Note that $\pi_{\mathcal{A}', tag_1, \mathcal{D}', R'}(k)$ is defined using the tag-based definition, while $\pi_{\mathcal{A}, tag_1, \mathcal{D}, R}(k)$ is not.) Here it is easy to verify that this difference is bounded by the probability of the adversary generating an identical commitment with a different tag for a related message. Formally,

$$\begin{aligned}
&\pi_{\mathcal{A}', tag_1, \mathcal{D}', R'}(k) - \pi_{\mathcal{A}, tag_1, \mathcal{D}, R}(k) \\
&\quad \leq \Pr[pk \leftarrow \mathsf{TCgen}(1^k); v_1 \stackrel{R}{\leftarrow} \mathcal{D}; (\mathsf{com}_1, \mathsf{dec}_1) \leftarrow \mathsf{TCcom}(pk, v_1, tag_1); \\
&\qquad (\mathsf{com}_2, tag_2, \xi) \leftarrow \mathcal{A}_1(pk, \mathsf{com}_1, tag_1); (v_2, \mathsf{dec}_2) \leftarrow \mathcal{A}_2(pk, s, v_1, \mathsf{dec}_1) : \\
&\qquad (\mathsf{TCver}(pk, \mathsf{com}_2, v_2, tag_2, \mathsf{dec}_2) = 1) \wedge (\mathsf{com}_1 = \mathsf{com}_2) \wedge (tag_1 \neq tag_2) \wedge R(v_1, v_2)].
\end{aligned}$$

We show that this probability is negligible by showing that an adversary $\mathcal{B}$ may be constructed that breaks the binding property of TC with the same probability. Let $\mathcal{B}$ run as follows, given $pk \leftarrow \mathsf{TCgen}_{pk}(1^k)$. $\mathcal{B}$ chooses $m_1 \stackrel{R}{\leftarrow} \mathcal{D}'$ and sets $v_1 \leftarrow \langle m_1, tag_1\rangle$. Then $\mathcal{B}$ computes

$$\begin{aligned}
(\mathsf{com}_1, d_1) &\leftarrow \mathsf{TCcom}(pk, v_1, tag_1), \\
(\mathsf{com}_2, tag_2, \xi) &\leftarrow \mathcal{A}_1(pk, \mathsf{com}_1, tag_1), \text{and} \\
(v_2, \mathsf{dec}_2) &\leftarrow \mathcal{A}_2(pk, \xi, \mathsf{dec}_1).
\end{aligned}$$

Then if $(\mathsf{TCver}(pk, \mathsf{com}_2, v_2, tag_2, \mathsf{dec}_2) = 1) \wedge (\mathsf{com}_1 = \mathsf{com}_2) \wedge (tag_1 \neq tag_2)$, $\mathcal{B}$ outputs the tuple $(\mathsf{com}_1, tag_1, v_1, v_2, \mathsf{dec}_1, \mathsf{dec}_2)$. Note that $v_1 \neq v_2$, since $tag_1 \neq tag_2$, $v_1 = \langle m_1, tag_1\rangle$ and $v_2 = \langle m_2, tag_2\rangle$ for some $m_2 \in \mathcal{D}$ (by the definition of $\mathcal{A}_2$). Also, TCver does not check the identifiers, so $\mathsf{TCver}(pk, \mathsf{com}_2, \langle m_1, tag_1\rangle, tag_2, \mathsf{dec}_2) = \mathsf{TCver}(pk, \mathsf{com}_2, \langle m_2, tag_2\rangle, tag_2, \mathsf{dec}_2) = 1$. Thus $\mathcal{B}$ breaks the binding property of TC with probability at least $\hat{\pi}_{\mathcal{A}', tag_1, \mathcal{D}', R'}(k) - \pi_{\mathcal{A}, tag_1, \mathcal{D}, R}(k)$, so this difference must be negligible in k. ∎

Note that Theorem 5.3 could be generalized to apply to non-trapdoor commitment schemes and to strict non-malleable commitment schemes (as opposed to ϵ-non-malleable commitment schemes). However, we do not know any easy way (e.g., without adding a more complicated construction, like a zero-knowledge proof) to convert a body-based $\mathsf{NMTC}(\ell)$ scheme into a tag-based $\mathsf{NMTC}(\ell)$ scheme, for any $\ell > 0$. The problem is dealing with the oracles, and the fact that one restricts success using a tag-based definition, and the other restricts success using a body-based definition.

Now considering the problem of converting tag-based SSTC or NMTC schemes to body-based SSTC or NMTC schemes, it seems that a simple construction like the one in Theorem 5.3 does not suffice.

Instead, one could construct a body-based scheme by generating a verification/signing key pair for a strong one-time signature scheme (see Appendix E), using the verification key as the tag in the tag-based commitment, signing the tag-based commitment using the signing key, and giving the pair (the tag-based commitment and the associated signature) as the full commitment. As this is a fairly standard technique, used in, e.g. [31], we omit the analysis here.

5.2 Relations between SSTC and NMTC

First we show that for all $\ell \geq 0$, an $\mathsf{SSTC}(\ell+1)$ scheme is also an $\mathsf{NMTC}(\ell)$ scheme. We use the notation $A(*;\omega)$ to denote a probabilistic algorithm A that has its random bits fixed to ω. (Note that all probabilistic subroutines called by A will also have their random bits fixed.)

Theorem 5.4 *Let* TC *be an* $\mathsf{SSTC}(\ell+1)$ *scheme. Then* TC *is an* $\mathsf{NMTC}(\ell)$ *scheme.*

Proof: This proof has the same structure as, but is a generalization of, the proof of Theorem 1 in [21].[11]

Take any polynomial $r(\cdot)$, any PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for the unbounded non-malleability of TC. Construct a simulator $\mathcal{S}$ that depends on a parameter ϵ and runs as follows, with $\mathcal{O}_{pk,sk}$ queries answered by $\mathcal{S}$ (which it can do since it will know sk).

$\mathcal{S}(1^k, tag_1, \mathcal{D})$:
  $(pk, sk) \leftarrow \mathsf{TCgen}(1^k)$
  Fix random tape ω
  $(\widetilde{\mathsf{com}}_1, \xi_1) \leftarrow \mathsf{TCfakeCom}(pk, sk, tag_1)$
  $(\mathsf{com}_2, tag_2, \xi_2) \leftarrow \mathcal{A}_1^{\mathcal{O}_{pk,sk}}(pk, \widetilde{\mathsf{com}}_1, tag_1, \mathcal{D})$
  Repeat at most $2\epsilon^{-1}\ln 2\epsilon^{-1}$ *times:*
    $m_1 \stackrel{R}{\leftarrow} \mathcal{D}$
    $m_2 \leftarrow \mathsf{DecommitValid}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, m_1; \omega)$
    If $m_2 \neq \perp$ *break*
  Output m_2

In the definition of M above, $\mathsf{DecommitValid}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, m_1)$ is defined as follows.

$\mathsf{DecommitValid}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, m_1)$
  $\widetilde{\mathsf{dec}}_1 \leftarrow \mathsf{TCfakeDecom}(\xi_1, \widetilde{\mathsf{com}}1, tag_1, m_1)$
  $(m_2, \mathsf{dec}_2) \leftarrow \mathcal{A}_2(pk, \xi_2, m_1, \widetilde{\mathsf{dec}}_1)$
  If $(\mathsf{TCver}(pk, \mathsf{com}_2, m_2, tag_2, \mathsf{dec}_2) = 1) \wedge (tag_1 \neq tag_2)\}$
  then output m_2
  else output $\perp$

By inspection, for any $r(\cdot)$ and PPT $\mathcal{A}$, assuming $\mathcal{D}$ is samplable in time $r(k)$ then there is a polynomial $q(\cdot,\cdot)$ such that for all non-negligible $\epsilon > 0$, $\mathcal{S}$ runs in time $q(k, \epsilon^{-1})$.

Define $\mathsf{Expt}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, \omega)$ as follows:[12]

$\mathsf{Expt}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, \omega)$:
  Repeat at most $2\epsilon^{-1}\ln 2\epsilon^{-1}$ *times:*
    $m_1 \stackrel{R}{\leftarrow} \mathcal{D}$
    $m_2^* \leftarrow \mathsf{DecommitValid}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, m_1; \omega)$
    If $m_2^* \neq \perp$ *then break*
  output m_2^*

[11] Their proof shows that a specific trapdoor commitment scheme with a slightly stronger binding property (similar to a body-based $\mathsf{SSTC}(1)$ but not quite as strong) is also an body-based NMTC scheme.

[12] Note that in Expt, ω is preceded by a comma and not a semicolon. Therefore it is a parameter to Expt, and is not the random bits of the tape for running Expt. ω will be used to fix the random bits for a subroutine of Expt.

Now for any identifier tag_1, any distribution $\mathcal{D}$ samplable in $r(k)$ time, and any relation R, let

$$\begin{aligned}\hat{\pi}_{\mathcal{A},tag_1,\mathcal{D},R}(k) \quad = \quad & \Pr[(pk,sk) \leftarrow \mathsf{TCgen}(1^k); \omega \stackrel{R}{\leftarrow} \Omega; (\widetilde{\mathsf{com}}_1, \xi_1) \leftarrow \mathsf{TCfakeCom}(pk, sk, tag_1); \\ & (\mathsf{com}_2, tag_2, \xi_2) \leftarrow \mathcal{A}_1^{\mathcal{O}_{pk,sk}}(pk, \widetilde{\mathsf{com}}1, tag_1, \mathcal{D}); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; \\ & m_2 \leftarrow \mathsf{DecommitValid}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, m_1; \omega) : R(m_1, m_2) = 1]\end{aligned}$$

and notice that $\pi_{\mathcal{A},tag_1,\mathcal{D},R}(k) - \hat{\pi}_{\mathcal{A},tag_1,\mathcal{D},R}(k)$ is negligible, by the trapdoorness property. Specifically, since the only difference between the two experiments is that one uses TCfakeCom and TCfakeDecom and the other uses TCcom, by trapdoorness there exists a negligible function $\beta(k)$ such that $\pi_{\mathcal{A},tag_1,\mathcal{D},R}(k) - \hat{\pi}_{\mathcal{A},tag_1,\mathcal{D},R}(k) \leq_{\text{ev}} \beta(k)$. (This is where we use the fact that R is a polynomial-time relation, because the distinguisher runs it, and we need the distinguisher to be polynomial time.)

Now notice that

$$\begin{aligned}\pi'_{\mathcal{S},tag_1,\mathcal{D},R}(k) \quad = \quad & \Pr[(pk,sk) \leftarrow \mathsf{TCgen}(1^k); \omega \stackrel{R}{\leftarrow} \Omega; (\widetilde{\mathsf{com}}_1, \xi_1) \leftarrow \mathsf{TCfakeCom}(pk, sk, tag_1); \\ & (\mathsf{com}_2, tag_2, \xi_2) \leftarrow \mathcal{A}_1^{\mathcal{O}_{pk,sk}}(pk, \widetilde{\mathsf{com}}1, tag_1, \mathcal{D}); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; \\ & m_2^* \leftarrow \mathsf{Expt}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1; \omega) : R(m_1, m_2^*) = 1]\end{aligned}$$

As in [21], let us denote the generation of a random tuple $\gamma = (pk, \widetilde{\mathsf{com}}_1, \xi_1, \mathsf{com}_2, tag_2, \xi_2, \omega)$ by $\gamma \leftarrow \Gamma(1^k)$. For a given tag_1 and $\mathcal{D}$, define a tuple $\gamma = (pk, \widetilde{\mathsf{com}}_1, \xi_1, \mathsf{com}_2, tag_2, \xi_2, \omega)$ as being good if it satisfies

$$\Pr[m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, m_1; \omega) : m_2 \neq \perp] \geq \epsilon/2$$

Note that this probability distribution is only over the random choice of m_1. Let $\mathsf{GOOD}(\gamma)$ be the event of that γ is good. For a tuple $\gamma = (pk, \widetilde{\mathsf{com}}_1, \xi_1, \mathsf{com}_2, tag_2, \xi_2, \omega)$, we write $\mathsf{DecommitValid}(m_1, \gamma)$ to denote experiment $\mathsf{DecommitValid}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, m_1; \omega)$, and we write $\mathsf{Expt}(\gamma)$ to denote experiment $\mathsf{Expt}(pk, \mathsf{com}_2, tag_2, \xi_2, \widetilde{\mathsf{com}}_1, \xi_1, tag_1, \omega)$.

Then

$$\begin{aligned}& \hat{\pi}_{\mathcal{A},tag_1,\mathcal{D},R}(k) - \pi'_{\mathcal{S},tag_1,\mathcal{D},R}(k) \\ & \leq \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma) : R(m_1, m_2) \wedge \mathsf{GOOD}(\gamma)] \\ & \quad + \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma) : R(m_1, m_2) \wedge \neg\mathsf{GOOD}(\gamma)] \\ & \quad - \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2^* \leftarrow \mathsf{Expt}(\gamma) : R(m_1, m_2^*) \wedge \mathsf{GOOD}(\gamma)] \\ & \quad - \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2^* \leftarrow \mathsf{Expt}(\gamma) : R(m_1, m_2^*) \wedge \neg\mathsf{GOOD}(\gamma)] \\ & \leq \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma) : R(m_1, m_2) \wedge \mathsf{GOOD}(\gamma)] \\ & \quad + \epsilon/2 \\ & \quad - \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2^* \leftarrow \mathsf{Expt}(\gamma) : R(m_1, m_2^*) \wedge \mathsf{GOOD}(\gamma)]\end{aligned}$$

This implies

$$\begin{aligned}& \hat{\pi}_{\mathcal{A},tag_1,\mathcal{D},R}(k) - \pi'_{\mathcal{S},tag_1,\mathcal{D},R}(k) \\ & \leq \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma); m_2^* \leftarrow \mathsf{Expt}(\gamma) : \\ & \quad R(m_1, m_2) \wedge \overline{R(m_1, m_2^*)} \wedge \mathsf{GOOD}(\gamma)] \\ & \quad + \epsilon/2\end{aligned}$$

$$
\begin{aligned}
= \;& \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(M, \gamma); m_2^* \leftarrow \mathsf{Expt}(\gamma) : \\
& R(m_1, m_2) \wedge \overline{R(m_1, m_2^*)} \wedge (m_2^* = \perp) \wedge \mathsf{GOOD}(\gamma)] \\
& + \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma); m_2^* \leftarrow \mathsf{Expt}(\gamma) : \\
& R(m_1, m_2) = 1 \wedge \overline{R(m_1, m_2^*)} \wedge (m_2^* \neq \perp) \wedge \mathsf{GOOD}(\gamma)] \\
& + \epsilon/2
\end{aligned}
$$

To bound the first probability, recall that in the experiment for choosing m_2^*, we are given up to $2\epsilon^{-1} \ln 2\epsilon^{-1}$ tries to obtain $m_2^* \neq \perp$, and in each attempt, the probability is at least $\epsilon/2$ assuming that $\mathsf{GOOD}(\gamma)$ occurs. Then

$$
\begin{aligned}
& \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma); m_2^* \leftarrow \mathsf{Expt}(\gamma) : \\
& R(m_1, m_2) \wedge \overline{R(m_1, m_2^*)} \wedge (m_2^* = \perp) \wedge \mathsf{GOOD}(\gamma)] \\
\leq \;& \Pr[\gamma \leftarrow \Gamma(1^k); m_1 \stackrel{R}{\leftarrow} \mathcal{D}; m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma); m_2^* \leftarrow \mathsf{Expt}(\gamma) : \\
& R(m_1, m_2) \wedge \overline{R(m_1, m_2^*)} \wedge (m_2^* = \perp) | \mathsf{GOOD}(\gamma)] \\
\leq \;& \epsilon/2.
\end{aligned}
$$

Now call the second probability $\sigma_{\mathcal{A}, tag_1, \mathcal{D}, R}(k)$. Note that if TC is not an $\mathsf{NMTC}(\ell)$ scheme, then for all negligible $\alpha(k)$ there is a function $r(\cdot)$, a PPT adversary $\mathcal{A}$, a non-negligible $\epsilon > 0$, a distribution $\mathcal{D}$ samplable in $r(k)$ time, an identifier tag_1 and a polynomial-time computable valid relation R such that for an infinite set of natural numbers k,

$$\sigma_{\mathcal{A}, tag_1, \mathcal{D}, R}(k) \geq \alpha(k).$$

Using this we will show how to construct a PPT adversary $\mathcal{B}$ that contradicts the fact that TC is an $\mathsf{SSTC}(\ell + 1)$ scheme. For $(pk, sk) \leftarrow \mathsf{TCgen}(1^k)$, $\mathcal{B}(pk)$ generates $\omega \stackrel{R}{\leftarrow} \Omega$, calls its oracle with (commit, tag_1) to get $\widetilde{\mathsf{com}}_1$ (with $(\widetilde{\mathsf{com}}_1, \xi_1)$ stored by the oracle), computes $(\mathsf{com}_2, tag_2, \xi_2) \leftarrow \mathcal{A}_1^{\mathcal{O}_{pk,sk}}(pk, \widetilde{\mathsf{com}}_1, tag_1, \mathcal{D})$ by forwarding $\mathcal{O}_{pk,sk}$ calls by $\mathcal{A}$ to its own oracle.

Let $\gamma = (pk, \widetilde{\mathsf{com}}_1, \xi_1, \mathsf{com}_2, tag_2, \xi_2, \omega)$ (Note that $\mathcal{B}$ does not see ξ_1, but it is stored by the oracle.) Thus γ is generated with the same distribution as $\Gamma(1^k)$. Then $\mathcal{B}(pk)$ generates $m_1 \stackrel{R}{\leftarrow} \mathcal{D}$ and runs $m_2 \leftarrow \mathsf{DecommitValid}(m_1, \gamma)$ and $m_2' \leftarrow \mathsf{Expt}(\gamma)$, but replacing $\mathsf{TCfakeDecom}(-, \widetilde{\mathsf{com}}, tag, m)$ in $\mathsf{DecommitValid}$ with a query $\mathsf{decommit}(\widetilde{\mathsf{com}}, m)$. $\mathcal{B}(pk)$ outputs $(\mathsf{com}_2, tag_2, m_2, m_2', \mathsf{dec}_2, \mathsf{dec}_2')$ (where dec_2 and dec_2' are generated along with m_2 and m_2').

Note that if $\mathcal{A}$ makes at most ℓ commit queries to its oracle, then $\mathcal{B}$ makes at most $\ell + 1$ commit queries to its oracle. Also note that since ϵ is non-negligible, $\mathcal{B}$ runs in probabilistic polynomial time. To analyze the success probability, note that if the condition defining $\sigma_{\mathcal{A}, tag_1, \mathcal{D}, R}(k)$ is satisfied, then $\mathcal{B}$ succeeds, since neither m_2 nor m_2' is $\perp$, they are different, and the commitment com_2 has been opened to each one. So $\mathcal{B}$ succeeds with probability at least $\sigma_{\mathcal{A}, tag_1, \mathcal{D}, R}(k)$. This contradicts the fact that TC is an $\mathsf{SSTC}(\ell + 1)$ scheme. Thus TC is an $\mathsf{NMTC}(\ell)$ commitment scheme. □

To relate our results to reusable non-malleable commitment schemes as defined in [17], we need to consider adversaries that input a vector of commitments (and later decommitments), and output a vector of commitments (and later decommitments). To be specific, let (t, u)-$\mathsf{NMTC}(\ell)$ denote a reusable NMTC commitment scheme with an input vector of size t and an output vector of size u. Then using a proof similar to above, but with some additional ideas from [17], we can prove the following theorem.[13]

[13] As in [17], we change the definition of a valid relation (over vectors of messages) to one in which all messages including $\perp$ are allowed, but where the probability of the relation being true cannot be increased by changing a message in the second (adversarially-chosen) vector to $\perp$.

Theorem 5.5 *Let* TC *be an* SSTC($\ell + t$) *scheme. Then* TC *is a* (t, u)-NMTC(ℓ) *scheme.*

Now we look at the opposite direction.

Theorem 5.6 *Let* TC *be an* NMTC(ℓ) *scheme. Then* TC *is an* SSTC(ℓ) *scheme.*

Proof: First note that if there is at most one possible tag, then the TC must be an SSTC(ℓ) scheme by the binding property of TC. So assume there are more than one possible tags. Say a scheme TC is not an SSTC(ℓ) scheme. Then there exists a polynomial $\alpha(\cdot)$ and a PPT adversary $\mathcal{B}$ that, when given a public key generated by $\mathsf{TCgen}(1^k)$ and with access to an oracle to which it can make at most ℓ commit queries, for infinitely many k's, produces with probability more than $\alpha(k)$ a tuple $(\mathsf{com}, tag, v_1, v_2, \mathsf{dec}_1, \mathsf{dec}_2)$ such that tag was not queried to commit, $v_1 \neq v_2$, and $\mathsf{TCver}(pk, \mathsf{com}, v_1, tag, \mathsf{dec}_1) = \mathsf{TCver}(pk, \mathsf{com}, v_2, tag, \mathsf{dec}_2) = 1$.

From $\mathcal{B}$ we construct an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the non-malleability of the scheme. $\mathcal{A}_1^{\mathcal{O}_{pk,sk}}(pk, \widetilde{\mathsf{com}}_1, tag_1, \mathcal{D})$ runs $\mathcal{B}(pk)$, answering queries to $\mathcal{O}_{pk,sk}$ with its own oracle, until it gets an output $(\mathsf{com}_2, tag_2, v_{21}, v_{22}, \mathsf{dec}_{21}, \mathsf{dec}_{22})$. If $tag_2 = tag_1$, tag_2 was queried to commit, $v_{21} = v_{22}$, $\mathsf{TCver}(pk, \mathsf{com}_2, v_{21}, tag_2, \mathsf{dec}_{21}) \neq 1$, or $\mathsf{TCver}(pk, \mathsf{com}_2, v_{22}, tag_2, \mathsf{dec}_{22}) \neq 1$, then generate a random commitment and output it, and have $\mathcal{A}_2$ simply open that commitment. Otherwise, output $(\mathsf{com}_2, tag_2, (v_{21}, v_{22}, \mathsf{dec}_{21}, \mathsf{dec}_{22}))$. Then $\mathcal{A}_2(pk, (v_{21}, v_{22}, \mathsf{dec}_{21}, \mathsf{dec}_{22}), \widetilde{\mathsf{dec}}_1, v_1)$ checks if $R(v_1, v_{21})$, and if so $\mathcal{A}_2$ outputs $(v_{21}, \mathsf{dec}_{21})$, else $\mathcal{A}_2$ outputs $(v_{22}, \mathsf{dec}_{22})$. Note that since $\mathcal{B}$ calls the oracle at most ℓ times, so does $\mathcal{A}$.

Since $\mathcal{B}$ is polynomial time, there must be a polynomial $p(k)$ such that $|v_{21}|, |v_{22}| \leq p(k)$. Let $C(1^k, v)$ be an polynomial-time deterministic encoding function that maps a string of length at most $p(k)$ into a unique string of length exactly $p'(k)$, where $p'(k)$ is also a polynomial. For instance, $C(1^k, v)$ could map v to the string $z|v|0^{p(k)-|v|}$,where $z = |v|$ encoded in $\lceil \log p(k) \rceil$ bits.

Let $\mathcal{D}$ be uniform over $\{0,1\}^{p'(k)+1}$ and let $r(k)$ be the time to sample that distribution, which is obviously polynomial.

Let R be a polynomial-time relation taking two inputs of length $p'(k) + 1$ and at most $p(k)$, respectively, and having the properties that (1) for any $v_2 \in \{0,1\}^{\leq p(k)}$,

$$\Pr(v_1 \stackrel{R}{\leftarrow} \mathcal{D} : R(v_1, v_2)) = \frac{1}{2},$$

and (2) there is no correlation or only negative correlation between different second inputs, i.e., for any $v_{21}, v_{22} \in \{0,1\}^{\leq p(k)}$ with $v_{21} \neq v_{22}$,

$$\Pr(v_1 \stackrel{R}{\leftarrow} \mathcal{D} : R(v_1, v_{22}) | \overline{R(v_1, v_{21})}) \geq \frac{1}{2}.$$

As an example, let R be the following relation, where $v[j]$ denotes bit j of string v. (We are slightly abusing notation here by assuming that boolean operations result in 1 or 0, and these are then used as elements of $\mathbb{Z}_2$.)

$$R(v_1, v_2) = \left[\sum_{j=1}^{p'(k)} (v_1[j] \cdot w_2[j]) + v_1[p'(k) + 1] \right] \bmod 2,$$

where $w_2 = C(1^k, v_2)$. In other words, R is the inner product of the first $p'(k)$ bits of v_1 with the $p'(k)$-length encoding of v_2, exclusive-or the $(p'(k) + 1)$st bit of v_1.[14] Let tag_1 be an identifier that is output by $\mathcal{B}$ at most half the time $\mathcal{B}$ succeeds. This identifier must exist since there are at least

[14]Note that one can also construct relations with the two desired properties if the commitment scheme fixes the bit length of a message, e.g., to 1 (a single bit) or k.

two possible identifiers. Then using the properties of R, and using the fact that the probability that $tag_2 = tag_1$ is at most $\frac{1}{2}$ when $\mathcal{B}$ succeeds, it is easy to see that for infinitely many k's,

$$\pi_{\mathcal{A},tag_1,\mathcal{D},R}(k) \geq (1 - \frac{\alpha(k)}{2})\left(\frac{1}{2}\right) + \frac{\alpha(k)}{2}\left(\frac{3}{4}\right) = \frac{1}{2} + \frac{\alpha(k)}{8},$$

and that for any simulator $\mathcal{S}$,

$$\pi'_{\mathcal{S},tag_1,\mathcal{D},R}(k) \leq \frac{1}{2}.$$

Thus TC is not an NMTC(ℓ) scheme. □

5.3 Separations between SSTC and NMTC

Theorem 5.7 *Assuming the hardness of the discrete logarithm problem, there exists an* SSTC(ℓ) *scheme that is not* NMTC(ℓ)*, for every* $\ell \geq 0$.

Proof: We shall prove the theorem constructively. For every $\ell \geq 0$, we construct a scheme DL_ℓ that is SSTC(ℓ) (assuming the hardness of discrete logarithm) but not NMTC(ℓ).

Our construction is a modified version of the non-malleable commitment scheme based on discrete logarithms by Di Crescenzo et.al. [21]. Before describing the construction in more detail, we discuss some intuition behind the construction.

Given security parameter k, let G_q denote a finite (cyclic) group of order q, where q is prime and $|q| = k$. Let g be a generator of G_q, and assume it is included in the description of G_q. We will assume that elements in G_q can be efficiently sampled uniformly at random, and that for a random $y \in G_q$, it is computationally infeasible to compute x such that $y = g^x$. This value x is the discrete log of y, and this assumption is called the Discrete Logarithm (DL) assumption. (For instance, G_q may be a multiplicative subgroup of $\mathbb{Z}_p^*$, for some large prime p where $q|(p-1)$.)

Notice that the group G is isomorphic to the additive group $\mathbb{Z}_q$ in a straightforward manner (in fact $\mathbb{Z}_q$ is a field, a fact we shall use later). Now let us turn our attention to polynomials over $\mathbb{Z}_q$. We write a degree-ℓ polynomial as $P(x) = a_0 + a_1 \cdot x + \cdots a_\ell \cdot x^\ell$. We state two extremely useful facts that shall be employed in our construction. These facts are used as well in secret sharing and threshold cryptography [49].

1. $(\ell+1)$**-wise independence**
   A random degree-ℓ polynomial is $(\ell+1)$-wise independent. In other words, for a degree-ℓ polynomial $P(x)$, the knowledge of its value on ℓ positions does not yield *any* information of its value on a new position. More precisely, by a *random polynomial*, we mean one whose coefficients are chosen from $\mathbb{Z}_q$ uniformly at random. Then, for any $x_1, x_2, ..., x_{\ell+1} \in \mathbb{Z}_q$, such that the x_i's are all distinct for $i = 1, 2, ..., \ell+1$, and any $y_1, y_2, ..., y_\ell \in \mathbb{Z}_q$, $P(x_{\ell+1})$ is still uniformly random over $\mathbb{Z}_q$, for a random polynomial $P(\cdot)$ *conditioned* on $P(x_i) = y_i$, for $i = 1, 2..., \ell$.

2. $(\ell+2)$**-wise dependence**
   A degree-ℓ polynomial is $(\ell+2)$-wise dependent. For any distinct $x_1, x_2, ..., x_{\ell+2} \in \mathbb{Z}_q$, there exist constants $\lambda_1, \lambda_2, .., \lambda_{\ell+1} \in \mathbb{Z}_q$ such that for any degree-ℓ polynomial $P(x)$, $\lambda_1 \cdot P(x_1) + \lambda_2 \cdot P(x_2) + \cdots + \lambda_{\ell+1} \cdot P(x_{\ell+1}) = P(x_{\ell+2})$.[15] Furthermore, these constants can be efficiently computed. As a direct corollary, given ℓ pairs $\{(x_i, y_i)\}_\ell$, and $\alpha, \alpha' \in \mathbb{Z}_q$, one can efficiently compute $u, v \in \mathbb{Z}_q$, such that
   $$P(\alpha') = u \cdot P(\alpha) + v \tag{1}$$
   for any degree-ℓ polynomial $P(x)$ satisfying that $P(x_i) = y_i$ for $i = 1, 2, ..., \ell$.

[15] This is essentially Lagrange Interpolation.

Very roughly speaking, we shall use the first fact to show that our construction is $\mathsf{SSTC}(\ell)$, and the second fact to show that it is not $\mathsf{NMTC}(\ell)$. We explain it in more detail next.

The DL_l scheme is a modified version of the commitment scheme by Di Crescenzo et. al. [21], which in turn is based on Pedersen commitments [44]. (A Pedersen commitment to a message m is $g^m h^r$ for a random r, where $h \in G_q$ is a value such that the discrete log of h base g is unknown.) In DL_l, the public key pk consists of a universal one-way hash function $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ and a signature scheme $\mathsf{SIG} = (\mathsf{sig\_gen}, \mathsf{sig\_sign}, \mathsf{sig\_verify})$ secure against one-time existential forgery. The public key pk also consists of the description of G_q (which we will henceforth denote simply as G_q), and $(\ell+1)$ random elements in G_q, denoted by $g_0, g_1, ..., g_\ell$. The corresponding secret key consists of the discrete logarithms of the g_i's base g. In other words, $sk = (a_0, a_1, ..., a_\ell)$ such that $g^{a_i} = g_i$, for $i = 0, 1, ..., \ell$.

To commit to a message $m \in \mathbb{Z}_q$ with tag, a sender first generates a fresh verification/signing key pair for a strong one-time signature scheme using $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$ and then computes $\alpha \leftarrow H(\mathsf{sig\_vk})$. We call α the "seed" of the commitment. Notice that since $H(\cdot)$ is a universal one-way hash function, α is in some sense "fresh." Next, the sender picks a random $r \in \mathbb{Z}_q$ and computes

$$B = \left(g_0 \cdot g_1^{\alpha} \cdot g_2^{\alpha^2} \cdots g_\ell^{\alpha^\ell}\right)^m \cdot g^r. \tag{2}$$

We call B the "body" of the commitment. Finally the sender generates a signature s on tag using $\mathsf{sig\_sk}_1$, and sends over $\mathsf{com} = (\mathsf{sig\_vk}, tag, B, s)$ as the commitment. To decommit, the sender simply exhibits (m, r), and then the receiver verifies that Eq.(2) holds and the signature is valid.

The protocol is described in Figure 6. Notice that the DKOS protocol [21] can be regarded as a variation (where a message authentication code replaces the strong one-time signature scheme) of the special case of DL_ℓ with $\ell = 1$.

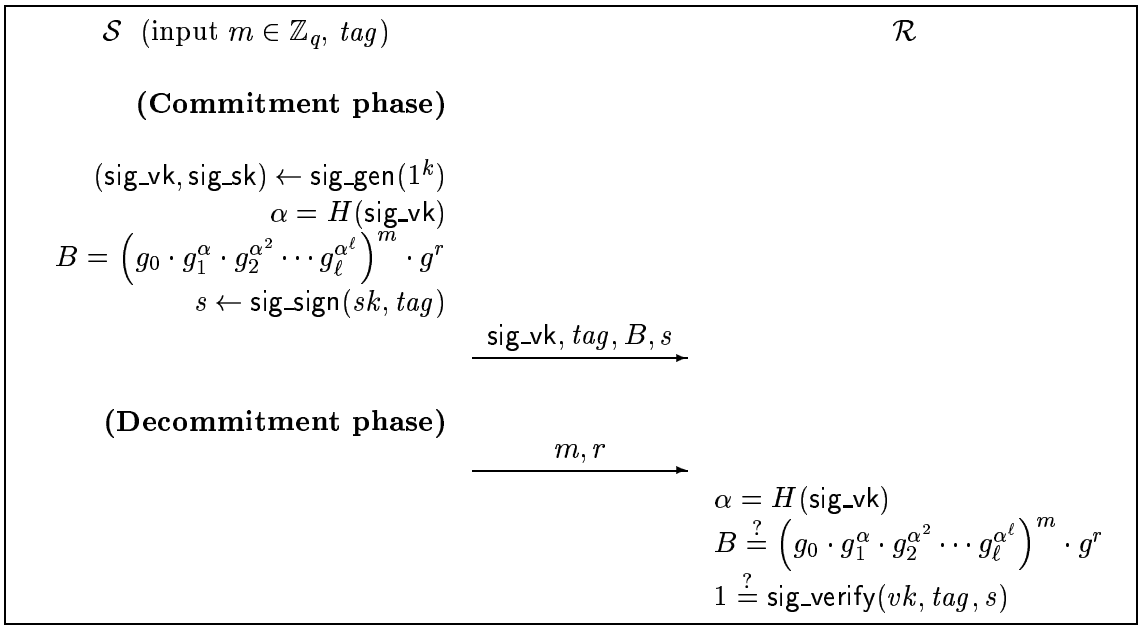

Figure 6: The DL_ℓ commitment scheme. The public key is $pk = (H, p, q, g, g_0, ...g_\ell)$.

To see that this is a trapdoor commitment scheme, we show how to produce commitments that can be equivocated with the secret key (i.e., we construct $\mathsf{TCfakeCom}$ and $\mathsf{TCfakeDecom}$). Using the secret key sk, we define a polynomial $P(x) = a_0 + a_1 \cdot x + \cdots + a_\ell \cdot x^\ell$. Notice that if g_i's are randomly chosen, then a_i's are random elements in $\mathbb{Z}_q$, and thus $P(x)$ is a random degree-ℓ polynomial. Now, Eq.(2) can be simplified to $B = g^{P(\alpha) \cdot m + r}$. Notice that the polynomial $P(x)$ is explicitly expressed in

the secret key sk, but only implicit given by the public key pk. Notice that with knowledge of $P(\alpha)$, where α is the seed of a commitment, one may equivocate that commitment. More precisely, say one wishes to produce a commitment with tag that can be equivocated on decommitment. One generates a signature key pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk})$, computes the seed $\alpha = H(\mathsf{sig\_vk})$, picks a random $t \in \mathbb{Z}_q$, produces $B = g^t$ as the body of the commitment, and sets the commitment to $(\mathsf{sig\_vk}, tag, B, s)$, where s is the signature on tag using $\mathsf{sig\_sk}$. Note that using sk, one can efficiently compute $P(\alpha)$. Thus to open this commitment to a message m, one simply computes $r = t - P(\alpha) \cdot m$ and sends (m, r) as the decommitment. This shows that DL_ℓ is a trapdoor commitment scheme.

Next, we show that the DL_ℓ scheme has the simulation-sound binding property.

First consider a commitment $\mathsf{com} = (\mathsf{sig\_vk}, B, s)$ with seed $\alpha = H(\mathsf{sig\_vk})$. Suppose it is opened in two different ways: $\mathsf{dec}_0 = (m_0, r_0)$ and $\mathsf{dec}_1 = (m_1, r_1)$. Then $g^{P(\alpha) \cdot m_0 + r_0} = B = g^{P(\alpha) \cdot m_1 + r_1}$, so $P(\alpha) = (r_1 - r_0)/(m_0 - m_1)$. In other words, given two openings to the same commitment, com, one can easily extract $P(\alpha)$ for the seed α.

Now we can see the intuitive reason why DL_ℓ is secure. Imagine an adversary $\mathcal{A}$ that interacts with an equivocation oracle (i.e., the oracle given in the description of the simulation-sound binding property) to obtain arbitrary decommitments for ℓ commitments. Intuitively, $\mathcal{A}$ obtains the values of $P(x)$ on ℓ different seeds (we denote them by $\alpha_1, \alpha_2, ..., \alpha_\ell$). If we can force $\mathcal{A}$ to use a new seed α in the commitment it wishes to equivocate (which is achieved by means of the universal one-way hash function H and the strong one-time signature scheme SIG), then one can easily extract $P(\alpha)$. However, since the values $P(\alpha_1), ..., P(\alpha_\ell)$ do not carry any information about $P(\alpha)$ (by the $(\ell+1)$-wise independence property), this value should have been computationally infeasible to produce, by the security of the Pedersen commitment scheme (which is based on the DL assumption).

More precisely, say an adversary $\mathcal{A}$ breaks the simulation-sound binding property of DL_ℓ. Then we will construct a "breaker" $\mathcal{B}$ that breaks the DL assumption. $\mathcal{B}$ takes as input G_q and an element $h \in G_q$ chosen uniformly at random, and will output $\log_g h$ with a probability that negligibly close to the probability that $\mathcal{A}$ breaks the simulation-sound binding property. We use the variable X to denote $\log_g h$ (which is a priori unknown to $\mathcal{B}$). $\mathcal{B}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$ as the equivocation oracle. $\mathcal{B}$ works in three phases.

1. **Key Generation**

   $\mathcal{B}$ generates the public key in the following way. First, $\mathcal{B}$ generates ℓ signature key pairs $(\mathsf{sig\_vk}_i, \mathsf{sig\_sk}_i)$ and computes the corresponding seeds $\alpha_i = H(\mathsf{sig\_vk}_i)$ for $i = 1, 2, ..., \ell$. Since $H(\cdot)$ is a universal one-way hash function, we may assume that the α_i's are distinct. $\mathcal{B}$ also picks ℓ random elements β_1, β_2, ..., β_ℓ. Next, $\mathcal{B}$ picks an $\alpha_0 \in \mathbb{Z}_q$ uniformly at random (and thus we may assume that α_0 is distinct from all previous α_i's) and sets $\beta_0 = X$ (symbolically). Then, $\mathcal{B}$ solves for the unique degree-ℓ polynomial $P(x) = a_0 + a_1 \cdot x + \cdots + a_\ell \cdot x^\ell$ such that $P(\alpha_i) = \beta_i$ for $i = 0, 1, ..., \ell$. This involves inverting a Vandermonde matrix and can be done efficiently. As a result, $\mathcal{B}$ is able to obtain two series of constants $\{A_i\}$, $\{B_i\}$, such that $a_i = A_i \cdot X + B_i$ for $i = 0, 1, \ldots, \ell$. Now, $\mathcal{B}$ sets $g_i = h^{A_i} \cdot g^{B_i}$ for $i = 0, 1, ..., \ell$ and outputs $pk' = (H, \mathsf{SIG}_1, G_q, g_0, g_1, ..., g_\ell)$ as the public key of the DL_ℓ scheme. In this way, we have that

   $$g_0 \cdot g_1^{\alpha_i} \cdots g_\ell^{\alpha_i^\ell} = g^{\beta_i} \tag{3}$$

   for $i = 0, 1, 2, ..., \ell$. (Note that $g^{\beta_0} = g^X = h$.) It is also easy to verify that since h was chosen uniformly at random from G_q, the distribution of pk' is identical to the distribution of the public key of DL_ℓ.

2. **Simulating the Equivocation Oracle**

   After generating the public key, $\mathcal{B}$ runs a copy of $\mathcal{A}$ and answers oracle queries, but at most ℓ commit queries. For the ith query (commit, tag_i), $\mathcal{B}$ uses the ith stored key pair $(\mathsf{sig\_vk}_i, \mathsf{sig\_sk}_i)$

and the corresponding seed $\alpha_i = H(\mathsf{sig\_vk}_i)$ as the seed for the commitment. Then $\mathcal{B}$ chooses a random t_i, computes $B_i = g^{t_i}$ as the body, and sends $c_i = (\mathsf{sig\_vk}_i, tag_i, B_i, \mathsf{sig\_sign}(\mathsf{sig\_sk}_i, tag_i))$ as the commitment. Notice that in the way the public key is set up, $\mathcal{B}$ knows $P(\alpha_i) = \beta_i$ and thus can decommit to any message easily. In particular, on receiving a query $(\mathsf{decommit}, c_i, v)$, $\mathcal{B}$ opens c_i to value v by replying $(v, t_i - \beta_i \cdot v)$.

3. **Extracting the Discrete Log**

   Say $\mathcal{A}$ produces a commitment $\widetilde{\mathsf{com}} = (\mathsf{sig\_vk}', tag', B, s)$ and two associated decommitments $\widetilde{\mathsf{dec}}_0 = (m_0, r_0)$ and $\widetilde{\mathsf{dec}}_1 = (m_1, r_1)$, with $m_0 \neq m_1$ and $tag' \neq tag_i$ for all $i \in \{1, \ldots, \ell\}$. By the fact that the strong one-time signature scheme SIG_1 is existential unforgeable and that H is a universal one-way hash function, we may assume that $\mathsf{sig\_vk}' \neq \mathsf{sig\_vk}_i$ and $H(\mathsf{sig\_vk}') \neq \alpha_i$, for all $i \in \{1, \ldots, \ell\}$. Let $\alpha = H(\mathsf{sig\_vk}')$. Then $\mathcal{B}$ may compute $P(\alpha) = (r_1 - r_0)/(m_0 - m_1)$. Now $\mathcal{B}$ knows the value of $P(\cdot)$ on $\ell + 1$ distinct points α_1, α_2, ..., α_ℓ, and α. Thus it computes u, v as in Eq.(1), such that $P(\alpha_0) = u \cdot P(\alpha) + v$, and hence compute $P(\alpha_0)$. But $X = \beta_0 = P(\alpha_0)$, and X is the discrete log of h base g. Therefore $\mathcal{B}$ is able to compute the discrete log of h.

Finally, we show that DL_ℓ is not $\mathsf{NMTC}(\ell)$, due to the $(\ell + 2)$-wise dependence of degree-ℓ polynomials. We shall present an adversary $\mathcal{A}$ that asks the equivocation oracle for multiple openings to ℓ commitments, receives a commitment com and then produces a commitment com', such that it can always open com' to whatever message com is opened to. Clearly, such an adversary completely breaks non-malleability (specifically, for the equality relation).

We now describe the adversary $\mathcal{A}$. Recall that associated with the public key is a "hidden" random polynomial $P(x)$. First, $\mathcal{A}$ obtains the value of $P(x)$ on ℓ different inputs by means of the equivocation oracle, and then receives a commitment $\mathsf{com} = (\mathsf{sig\_vk}, tag, B, s)$. Let $\alpha = H(\mathsf{sig\_vk})$. Then $\mathcal{A}$ picks a seed α' (by generating a new signature key pair and applying H to the verification key) and computes u, v as in Eq.(1), such that $P(\alpha') = u \cdot P(\alpha) + v$. Next, $\mathcal{A}$ submits a commitment com' with α' as the seed and $B' = B^u$ as the body. After receiving the opening (m, r) for the commitment com, $\mathcal{A}$ can also open com' to m by computing $r' = u \cdot r - v \cdot m \pmod q$. It is easy to verify that (m, r') is a valid opening:

$$B' = B^u = (g^{P(\alpha) \cdot m + r})^u = g^{u \cdot P(\alpha) \cdot m + u \cdot r} = g^{(P(\alpha') - v) \cdot m + u \cdot r} = g^{P(\alpha') \cdot m + (u \cdot r - v \cdot m)}$$

□

Theorem 5.8 *If there exists an* $\mathsf{NMTC}(\ell)$ *scheme, then there exists an* $\mathsf{NMTC}(\ell)$ *scheme that is not* $\mathsf{SSTC}(\ell + 1)$.

Proof: The idea behind this proof is that we can modify any $\mathsf{NMTC}(\ell)$ scheme and have it "leak" some information about the secret key when answering oracle queries (from the definition of simulation-sound binding) to equivocate a commitment. We control the leak in such a way that ℓ commit queries do not yield any information, but $\ell + 1$ commit queries will leak the secret key. This will imply that the modified scheme is still $\mathsf{NMTC}(\ell)$, but not $\mathsf{SSTC}(\ell + 1)$.

More precisely, consider an arbitrary $\mathsf{NMTC}(\ell)$ scheme $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{TCcom}, \mathsf{TCver}, \mathsf{TCfakeCom}, \mathsf{TCfakeDecom})$. Without the loss of generality, we may assume that the secret key sk produced by $\mathsf{TCgen}(1^k)$ is an element in $\mathbb{Z}_q$ for some prime number q (we can alway encode sk as a field element in a field large enough), and furthermore that q is at least k bits long. Now we modify TC slightly to produce a new commitment scheme $\mathsf{TC}' = (\mathsf{TCgen}', \mathsf{TCcom}', \mathsf{TCver}', \mathsf{TCfakeCom}', \mathsf{TCfakeDecom}')$. In TC', the decommitment dec' contains an additional pair of elements $(x, y) \in \mathbb{Z}_q \times \mathbb{Z}_q$ which we call the "leaking channel". The commitment function TCcom' fills the leaking channel with a random element pair in $\mathbb{Z}_q$, and the verification function TCver' ignores it. Thus the "basic" commitment/decommitment

functionality remains unchanged with the addition of the leaking channel. In fact, the leaking channel is only used by the functions $\mathsf{TCfakeCom}'$ and $\mathsf{TCfakeDecom}'$ to "leak" the information about the secret key sk, as we explain next.

In TC', the public key pk' is the same as pk. The secret key sk' consists of the secret key sk of the original scheme TC and a random degree-ℓ polynomial $P(x) = a_0 + a_1 \cdot x + \cdots + a_\ell \cdot x^\ell$ over $\mathbb{Z}_q$ satisfying that $P(0) = sk$ (or equivalently, $a_0 = sk$). Here, a random polynomial over $\mathbb{Z}_q$ is a polynomial whose coefficients are chosen uniformly at random from $\mathbb{Z}_q$.

The leaking channel is used by the faking functions to leak the values of $P(x)$ on random elements of $\mathbb{Z}_q$. More precisely, the new commitment-faking function $\mathsf{TCfakeCom}'(pk, sk, tag)$ first invokes $\mathsf{TCfakeCom}$ by setting $(c, \xi) \leftarrow \mathsf{TCfakeCom}(pk, sk, tag)$, and then picks a random $x \in \mathbb{Z}_q$, sets $\xi' = (\xi, x, P(x))$ and outputs (c, ξ'). The new decommitment-faking function $\mathsf{TCfakeDecom}'(\xi', c, tag, v)$, where $\xi' = (\xi, x, P(x))$, outputs $(\mathsf{TCfakeDecom}(\xi, c, tag, v), x, P(x))$. Notice that $\mathsf{TCfakeDecom}'$ fills the leaking channel with the information of $P(x)$ over a particular input x.

We now prove that the new scheme TC' remains $\mathsf{NMTC}(\ell)$. First, the standard hiding/binding property of TC' follows straightforwardly from that of TC. Next, the trapdoor property of TC' remains essentially unchanged from that of TC. This is because $(x, P(x))$ is uniformly distributed over $\mathbb{Z}_q \times \mathbb{Z}_q$ for random $x \in \mathbb{Z}_q$ and random $P(\cdot)$, except when $x = 0$, which happens with probability $\frac{1}{q}$, and therefore the leaking channel in TCcom' is statistically indistinguishable from the leaking channel in $\mathsf{TCfakeCom}'/\mathsf{TCfakeDecom}'$.

Finally, the non-malleability of TC' follows almost straightforwardly from that of TC. Notice that an adversary making up to ℓ commit queries learns at most ℓ pairs $(x_i, P(x_i))$ from the leaking channel. Since each x_i is uniformly chosen at random, the probability that $x_i = 0$ for some i, or that $x_i = x_j$ for some i and j, is at most $\frac{\ell+\ell^2}{q}$, which is negligible. Now we may suppose the x_i's are all nonzero and distinct. Since $P(x)$ is a random degree-ℓ polynomial, $P(x_1)$, $P(x_2)$,..., $P(x_\ell)$ are all independent uniformly random elements in $\mathbb{Z}_q$ and therefore they don't carry any information about $sk = P(0)$. Thus for any adversary $\mathcal{A}'$ that breaks TC', we can easily construct an $\mathcal{A}$ that breaks TC. Essentially $\mathcal{A}$ runs a copy of $\mathcal{A}'$, and simulates the faking oracle for TC' by filling the leaking channel with a random element pair $(x_i, y_i) \in \mathbb{Z}_q \times \mathbb{Z}_q$ in the reply to the decommitment of the ith commitment $\mathcal{A}$ requests.

However, the TC' scheme is obviously not $\mathsf{SSTC}(\ell + 1)$, since with $\ell + 1$ queries, an adversary can determine $P(0)$ by Lagrange interpolation (except when there is a collision, i.e., $x_i = x_j$ for some i and j in the leaking channel, which happens with negligible probability as discussed above). Thus the adversary completely breaks the scheme. □

We mention that Theorems 5.4, 5.5, 5.6, 5.8, and 5.7 would also apply to the body-based definitions of SSTC and NMTC schemes.

Acknowledgments

We would like to thank Ivan Damgård for helpful comments and suggestions, particularly in relating the results in our paper to those in [17].

References

[1] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology – EUROCRYPT '97* (LNCS 1233), pp. 480–494, 1997.

[2] D. Beaver. Adaptive zero-knowledge and computational equivocation. In *28th ACM Symp. on Theory of Computing*, pp. 629–638, 1996.

[3] M. Blum. Coin flipping by telephone. In *IEEE Spring COMPCOM*, pp. 133–137, 1982.

[4] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *JCSS*, 37(2):156–189, 1988.

[5] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology – CRYPTO '99* (LNCS 1666), pages 414–430, 1999.

[6] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symp. on Foundations of Computer Sci.*, pp. 136–145, 2001.

[7] R. Canetti and M. Fischlin. Universally composable commitments. In *Advances in Cryptology – CRYPTO 2001* (LNCS 2139), pp. 19–40, 2001.

[8] R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *EUROCRYPT 2002* (LNCS 2332), pp. 337–351, 2002. Full version in *ePrint Archive*, report 2002/059. `http://eprint.iacr.org/`.

[9] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally composable two-party computation. In 34th ACM Symp. on Theory of Computing, pp. 494–503, 2002. Full version in *ePrint archive*, Report 2002/140. `http://eprint.iacr.org/`, 2002.

[10] R. Canetti and T. Rabin. Universal Composition with Joint State In *ePrint archive*, Report 2002/047, `http://eprint.iacr.org/`, 2002.

[11] S. A. Cook. The complexity of theorem-proving procedures. In *3rd IEEE Symp. on Foundations of Computer Sci.*, pp. 151–158, 1971.

[12] R. Cramer and I. Damgård. Zero-Knowledge Proofs for Finite Field Arithmetic, or: Can Zero-Knowledge Be for Free? In *Advances in Cryptology – CRYPTO '98* (LNCS 1462), pages 424–441, 1998.

[13] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO '94* (LNCS 839), pages 174–187, 1994.

[14] R. Cramer and V. Shoup. Signature scheme based on the strong RSA assumption. In *ACM Trans. on Information and System Security* 3(3):161-185, 2000.

[15] I. Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In *Advances in Cryptology – CRYPTO '89* (LNCS 435), pp. 17–29, 1989.

[16] I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *Advances in Cryptology – EUROCRYPT 2000* (LNCS 1807), pp. 418–430, 2000.

[17] I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th ACM Symp. on Theory of Computing*, pp. 426–437, 2003.

[18] I. Damgård and J. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Advances in Cryptology – CRYPTO 2002* (LNCS 2442), pp. 581–596, 2002. Full version in *ePrint Archive*, report 2001/091. `http://eprint.iacr.org/`, 2001.

[19] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology – CRYPTO 2001* (LNCS 2139), pp. 566–598, 2001.

[20] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *30th ACM Symp. on Theory of Computing*, pp. 141–150, 1998.

[21] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and Non-Interactive Non-Malleable Commitment. In *Advances in Cryptology – EUROCRYPT 2001* (LNCS 2045), pp. 40–59, 2001.

[22] D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. *SIAM J. on Comput.*, 30(2):391–437, 2000. Also in *23rd ACM Symp. on Theory of Computing*, pp. 542–552, 1991.

[23] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. on Information Theory*, 31:469–472, 1985.

[24] S. Even, O. Goldreich, and S. Micali. On-line/Off-line digital signatures. *J. Cryptology* 9(1):35-67 (1996).

[25] U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *22nd ACM Symp. on Theory of Computing*, pp. 416–426, 1990.

[26] U. Feige and A. Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *Advances in Cryptology – CRYPTO '89* (LNCS 435), pp. 526–544, 1989.

[27] FIPS 180-1. Secure hash standard. Federal Information Processing Standards Publication 180-1, U.S. Dept. of Commerce/NIST, National Technical Information Service, Springfield, Virginia, 1995.

[28] FIPS 186. Digital signature standard. Federal Information Processing Standards Publication 186, U.S. Dept. of Commerce/NIST, National Technical Information Service, Springfield, Virginia, 1994.

[29] M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. In *Advances in Cryptology – CRYPTO 2000* (LNCS 1880), pp. 413–431, 2000.

[30] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – CRYPTO '97* (LNCS 1294), pp. 16-30, 1997.

[31] J. A. Garay, P. MacKenzie, and K. Yang. Strengthening Zero-Knowledge Protocols using Signatures. In *Advances in Cryptology – EUROCRYPT 2003* (LNCS 2656), pp. 177–194, 2003.

[32] R. Gennaro, S.Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT '99* (LNCS 1592), pp. 123–139, 1999.

[33] O. Goldreich. Foundations of Cryptography - Volume 1 (Basic Tools). *Cambridge University Press*, ISBN 0-521-79172-3, 2001.

[34] O. Goldreich, S. Micali and A. Wigderson. Proofs that yield nothing but their validity or All languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[35] S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17:281–308, 1988.

[36] S. Jarecki and A. Lysyanskaya. Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. In *Advances in Cryptology – EUROCRYPT '00* (LNCS 1807), pp. 221–242, 2000.

[37] D. W. Kravitz. Digital signature algorithm. U.S. Patent 5,231,668, 27 July 1993.

[38] L. A. Levin. Universal sorting problems. *Problemy Peredaci Informacii*, 9:115–116, 1973. In Russian. Engl. trans.: *Problems of Information Transmission* 9:265–266.

[39] P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. In *Advances in Cryptology – CRYPTO 2002* (LNCS 2442), pp. 385–400, 2002.

[40] M. Naor. Bit commitment Using Pseudo-Randomness. *J. Cryptology* 4(2):151–158 (1991).

[41] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP can be based on general complexity assumptions. In *Advances in Cryptology – CRYPTO '92* (LNCS 740), pp. 196–214, 1992.

[42] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM Symp. on Theory of Computing*, pp. 427–437, 1990.

[43] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In *Advances in Cryptology – EUROCRYPT '99* (LNCS 1592), pp. 223–238, 1999.

[44] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology – CRYPTO '91* (LNCS 576), pp. 129–140, 1991.

[45] L. Reyzin. Zero-knowledge with public keys. Ph.D. Thesis, MIT, 2001.

[46] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM Symp. on Theory of Computing*, pp. 387–394, 1990.

[47] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th IEEE Symp. on Foundations of Computer Sci.*, pp. 543–553, 1999.

[48] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology – EUROCRYPT '89* (LNCS 434), pp. 688–689, 1989.

[49] A. Shamir. How to Share a Secret. In *CACM* 22(11), pp. 612–613 (1979).

A ZK proof Definitions

Here we provide definitions related to zero-knowledge proofs and proofs of knowledge. They are based on definitions of NIZK proofs from [19], but modified to allow interaction.

For a relation R, let $L_R = \{x : (x, w) \in R\}$ be the *language* defined by the relation. For any NP language L, note that there is a natural *witness relation* R containing pairs (x, w) where w is the witness for the membership of x in L, and that $L_R = L$. We will use k as the security parameter.

For two interactive machines A and B, we define $\langle A, B\rangle_{[\sigma]}(x)$ as the local output of B after an interactive execution with A using CRS σ, and common input x. The transcript of a machine is simply the messages on its input and output communication tapes. Two transcripts *match* if the ordered input messages of one are equivalent to the ordered output messages of the other, and vice-versa. We use the notation $tr \bowtie tr'$ to indicate tr matches tr'.

For some definitions below, we need to define security when an adversary is allowed to interact with more than one instance of a machine. Therefore it will be convenient to define a common *wrapper* machine that handles this "multi-session" type of interaction.[16] For an interactive machine A, we define $\boxed{A}$ to be a protocol wrapper for A, that takes two types of inputs on its communication tape:

- (START, π, x, w): For this message $\boxed{A}$ starts a new interactive machine A with label π, common input x, private input w, a freshly generated random input r, and using the CRS of $\boxed{A}$.
- (MSG, π, m): For this message $\boxed{A}$ sends the message m to the interactive machine with label π (if it exists), and returns the output message of that machine.

We define the output of $\boxed{A}$ to be a tuple (x, tr, v), where x is the common input (from the START message), tr is the transcript (the input and output messages A) and v is the output of A. (In particular, if A is a verifier in a zero-knowledge protocol, this output will be 1 for accept, and 0 for reject.) We say $\boxed{A}_1$ is the wrapper of A that ignores all the subsequent START messages after seeing the first one. Effectively, $\boxed{A}_1$ is a "single-session" version of A.

We say two interactive machines B and C are *coordinated* if they have a single control, but two distinct sets of input/output communication tapes. For four interactive machines A, B, C, and D we define $(\langle A, B\rangle, \langle C, D\rangle)_{[\sigma]}$ as the local output of D after an interactive execution with C and after an interactive execution of A and B, all using CRS σ. Note that we will only be concerned with this if B and C are coordinated.

We note that all our ZK definitions use black-box, non-rewinding simulators, and our proofs of knowledge use non-rewinding extractors.

[16] This is similar to the "multi-session extension" concept in Canetti and Rabin [10].

Definition A.1 [Unbounded ZK Proof] $\Pi = (\mathcal{D}, \mathcal{P}, \mathcal{V}, \mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2))$ *is an* unbounded ZK proof *(resp.,* argument*)* system *for an NP language L with witness relation R if $\mathcal{D}$ is an ensemble of polynomial-time samplable distributions, $\mathcal{P}$, $\mathcal{V}$, and $\mathcal{S}_2$ are probabilistic polynomial-time interactive machines, and $\mathcal{S}_1$ is a probabilistic polynomial-time machine, such that there exist negligible functions α and β (the simulation error), such that for all k,*

Completeness *For all $x \in L$ of length k, all w such that $R(x,w) = 1$, and all $\sigma \in \mathcal{D}_k$ the probability that $\langle \mathcal{P}(w), \mathcal{V} \rangle_{[\sigma]}(x) = 0$ is less than $\alpha(k)$.*

Soundness *For all unbounded (resp., polynomial-time) adversaries $\mathcal{A}$, if $\sigma \stackrel{R}{\leftarrow} \mathcal{D}_k$, then for all $x \notin L$, the probability that $\langle \mathcal{A}, \mathcal{V} \rangle_{[\sigma]}(x) = 1$ is less than $\alpha(k)$.*

Unbounded ZK *For all non-uniform probabilistic polynomial-time interactive machines $\mathcal{A}$, we have that $|\Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathcal{S}}_{\mathcal{A}}(k) = 1]| \leq \beta(k)$, where the experiments $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathcal{S}}_{\mathcal{A}}(k)$ are defined as follows:*

$\mathsf{Expt}_{\mathcal{A}}(\kappa)$:	$\mathsf{Expt}^{\mathcal{S}}_{\mathcal{A}}(\kappa)$:
$\sigma \stackrel{R}{\leftarrow} \mathcal{D}_k$	$(\sigma, \tau) \leftarrow \mathcal{S}_1(1^k)$
Return $\langle \boxed{\mathcal{P}}, \mathcal{A} \rangle_{[\sigma]}$	Return $\langle \boxed{\mathcal{S}'(\tau)}, \mathcal{A} \rangle_{[\sigma]}$

where $\mathcal{S}'(\tau)$ runs as follows on common reference string σ, common input x and private input w: if $R(x,w) = 1$, $\mathcal{S}'(\tau)$ runs $\mathcal{S}_2(\tau)$ on common reference string σ and common input x; otherwise $\mathcal{S}'(\tau)$ runs $\mathcal{S}_{\text{null}}$, where $\mathcal{S}_{\text{null}}$ is an interactive machine that simply aborts.[17]

We point out that this definition only requires the simulator to simulate a valid proof, which is implemented by having $\mathcal{S}'$ have access to the witness w and only invoking $\mathcal{S}_2$ when w is valid.[18] However, $\mathcal{S}_2$ does not access the witness and will simulate a proof from the input x only.

Definition A.2 [Same-String Unbounded ZK] $\Pi = (\mathcal{D}, \mathcal{P}, \mathcal{V}, \mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2))$ *is a* same-string unbounded ZK argument *system for an NP language L with witness relation R if Π is an unbounded ZK argument system for L with the additional property that the distribution of the reference string output by $\mathcal{S}_1(1^k)$ is exactly $\mathcal{D}_k$.*

We only define same-string unbounded ZK arguments since, as shown in [19], any protocol that is same-string unbounded ZK must be an argument, and not a proof.

The following defines unbounded simulation-sound zero-knowledge (USSZK). This has been useful in applications. In particular, as shown in [47], the one-time version suffices for the security of a (non-interactive) ZK protocol in the construction of adaptive chosen-ciphertext secure cryptosystems using the Naor-Yung [42] paradigm. We directly define the unbounded version, needed in other applications such as threshold password-authenticated key exchange [39].

Definition A.3 [Unbounded Simulation-Sound ZK]
$\Pi = (\mathcal{D}, \mathcal{P}, \mathcal{V}, \mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2))$ *is an* unbounded simulation-sound ZK proof *(resp.,* argument*)* system *for an NP language L if Π is an unbounded ZK proof (resp., argument) system for L and furthermore, there exists a negligible function α such that for all k,*

Unbounded Simulation Soundness
For all non-uniform probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are coordinated, we have that $\Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1] \leq \alpha(k)$, where $\mathsf{Expt}_{\mathcal{A}}(k)$ is defined as follows:

[17] Without loss of generality, we assume that if the input to $\mathcal{P}$ is not a witness for the common input, $\mathcal{P}$ simply aborts.

[18] $\mathcal{A}$ must supply a witness, since $\mathcal{P}$ is restricted to polynomial time, and thus may not be able to generate a witness itself. This may seem odd compared to definitions of standard ZK that assume an unbounded prover, but it does seem to capture the correct notion of unbounded ZK, and in particular does not allow $\mathcal{A}$ to test membership in L. See Sahai [47] for more discussion.

$\mathsf{Expt}_{\mathcal{A}}(k)$:
$(\sigma, \tau) \leftarrow \mathcal{S}_1(1^k)$
$(x, tr, b) \leftarrow (\langle \boxed{\mathcal{S}''(\tau)}, \mathcal{A}_1 \rangle, \langle \mathcal{A}_2, \boxed{\mathcal{V}}_1 \rangle)_{[\sigma]}$
Let Q be the set of transcripts of machines in $\boxed{\mathcal{S}''(\tau)}$
Return 1 iff $b = 1$, $x \notin L$, and for all $tr' \in Q$, $tr \not\bowtie tr'$

where $\mathcal{S}''(\tau)$ runs as follows on CRS σ, common input x and private input w: $\mathcal{S}''(\tau)$ runs $\mathcal{S}_2(\tau)$ on CRS σ and common input x.

In the above definition, we emphasize that $\mathcal{S}_2$ may be asked to simulate *false* proofs for $x \notin L_R$, since $\mathcal{S}''$ does not check whether $(x, w) \in R$. The idea is that even if the adversary is able to obtain acceptable proofs on false statements, it will not be able to produce any new acceptable proof on a false statement.

The following defines non-malleable zero-knowledge (NMZK) proofs (resp., arguments) of knowledge. If a protocol is NMZK according to our definition, then this implies the protocol is also a NMZK in the explicit witness sense (as defined in [19]). Moreover, we show that the protocol is also UCZK in the model of static corruptions. Also note that simulation soundness is implied by this definition.

Definition A.4 [Non-malleable ZK Proof/Argument of Knowledge] $\Pi = (\mathcal{D}, \mathcal{P}, \mathcal{V}, \mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2), \mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2))$ *is a* non-malleable ZK proof *(resp.,* argument*)* of knowledge system *for an NP language L with witness relation R if Π is an unbounded ZK proof (resp., argument) system for L and furthermore, $\mathcal{E}_1$ and $\mathcal{E}_2$ are probabilistic polynomial-time machines such that there exists a negligible function α (the knowledge error) such that for all k,*

Reference String Indistinguishability *The distribution of the first output of $\mathcal{S}_1(1^k)$ is identical to the distribution of the first output of $\mathcal{E}_1(1^k)$.*

Extractor Indistinguishability *For any $\tau \in \{0,1\}^*$, the distribution of the output of $\boxed{\mathcal{V}}_1$ is identical to the distribution of the restricted output of $\boxed{\mathcal{E}_2(\tau)}_1$, where the restricted output of $\boxed{\mathcal{E}_2(\tau)}_1$ does not include the extracted value.*

Extraction *For all non-uniform probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are coordinated machines, we have that $|\Pr[\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1]| \leq \alpha(k)$, where the experiments $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$ are defined as follows:*

$\mathsf{Expt}_{\mathcal{A}}(k)$:	$\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$:
$(\sigma, \tau) \leftarrow \mathcal{S}_1(1^k)$	$(\sigma, \tau_1, \tau_2) \leftarrow \mathcal{E}_1(1^k)$
$(x, tr, b) \leftarrow (\langle \boxed{\mathcal{S}''(\tau)}, \mathcal{A}_1 \rangle, \langle \mathcal{A}_2, \boxed{\mathcal{V}}_1 \rangle)_{[\sigma]}$	$(x, tr, (b, w)) \leftarrow (\langle \boxed{\mathcal{S}''(\tau_1)}, \mathcal{A}_1 \rangle, \langle \mathcal{A}_2, \boxed{\mathcal{E}_2(\tau_2)}_1 \rangle)_{[\sigma]}$
Let Q be the set of transcripts of machines in $\boxed{\mathcal{S}''(\tau)}$.	Let Q be the set of transcripts of machines in $\boxed{\mathcal{S}''(\tau_1)}$.
Return 1 iff $b = 1$ and for all $tr' \in Q$, $tr \not\bowtie tr'$	Return 1 iff $b = 1$, $(x, w) \in R$, and for all $tr' \in Q$, $tr \not\bowtie tr'$

where $\mathcal{S}''(\tau)$ runs as follows on CRS σ, common input x and private input w: $\mathcal{S}''(\tau)$ runs $\mathcal{S}_2(\tau)$ on CRS σ and common input x.

In the above definition, as in the definition of USSZK protocols, we emphasize that $\mathcal{S}_2$ may be asked to simulate *false* proofs for $x \notin L_R$, since $\mathcal{S}''$ does not check whether $(x, w) \in R$. The idea is that even if the adversary is able to obtain acceptable proofs on false statements, it will not be able to produce any new acceptable proof for which a witness cannot be extracted.

B Σ-protocols and Ω-protocols

Here we overview the basic definitions and properties of Σ-protocols [13]

First we start with some definitions and notation. Let $R = \{(x, w)\}$ be a binary relation and assume that for some given polynomial $p(\cdot)$ it holds that $|w| \leq p(|x|)$ for all $(x, w) \in R$. Furthermore, let R be testable in polynomial time. Let $L_R = \{x : (x, w) \in R\}$ be the *language* defined by the relation, and for all $x \in L_R$, let $W_R(x) = \{w : (x, w) \in R\}$ be the *witness set* for x. For any NP language L, note that there is a natural *witness relation* R containing pairs (x, w) where w is the witness for the membership of x in L, and that $L_R = L$.

Now we define a Σ-protocol (A, B) to be a three move interactive protocol between a probabilistic polynomial-time prover A and a probabilistic polynomial-time verifier B, where the prover acts first. The verifier is only required to send random bits as a challenge to the prover. For some $(x, w) \in R$, the common input to both players is x while w is private input to the prover. For such given x, let (a, c, z) denote the conversation between the prover and the verifier. To compute the first and final messages, the prover invokes efficient algorithms $a(\cdot)$ and $z(\cdot)$, respectively, using (x, w) and random bits as input. Using an efficient predicate $\phi(\cdot)$, the verifier decides whether the conversation is accepting with respect to x. The relation R, the algorithms $a(\cdot)$, $z(\cdot)$ and $\phi(\cdot)$ are public. The length of the challenges is denoted t_B, and we assume that t_B only depends on the length of the common string x.

We will need to broaden this definition slightly, to deal with cheating provers. We will define $\hat{L}_R$ to be the input language, with the property that $L_R \subseteq \hat{L}_R$, and membership in $\hat{L}_R$ may be tested in polynomial time. We implicitly assume B only executes the protocol if the common input $x \in \hat{L}_R$.

All Σ-protocols presented here will satisfy the following security properties:

- *Weak special soundness*: Let (a, c, z) and (a, c', z') be two conversations, that are accepting for some given $x \in \hat{L}_R$. If $c \neq c'$, then $x \in L_R$. The pair of accepting conversations (a, c, z) and (a, c', z') with $c \neq c'$ is called a *collision*.

- *Special honest verifier zero knowledge* (SHVZK): There is a (probabilistic polynomial time) simulator M that on input $x \in L_R$ generates accepting conversations with a distribution that is indistinguishable[19] from when A and B execute the protocol on common input x (and A is given a witness w for x), and B indeed honestly chooses its challenges uniformly at random. The simulator is special in the sense that it can additionally take a random string c as input, and output an accepting conversation for x where c is the challenge. In fact, we will assume the simulator has this special property for not only $x \in L_R$, but also any $x \in \hat{L}_R$.

  Specifically, there is a negligible function $\alpha(k)$ such that for all non-uniform probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have that $|\Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{M}_{\mathcal{A}}(k) = 1]| \leq \alpha(k)$, where the experiments $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{M}_{\mathcal{A}}(k)$ are defined as follows:

$\mathsf{Expt}_{\mathcal{A}}(k)$:	$\mathsf{Expt}^{M}_{\mathcal{A}}(k)$:
$(x, w, s) \leftarrow \mathcal{A}_1(1^k)$	$(x, w, s) \leftarrow \mathcal{A}_1(1^k)$
If $(x, w) \notin R$ return 0	If $(x, w) \notin R$ return 0
$r \xleftarrow{R} \{0,1\}^*$	$c \xleftarrow{R} \{0,1\}^k$
$a \leftarrow a(x, w, r)$	Return $\mathcal{A}_2(s, M(x, c))$
$c \xleftarrow{R} \{0,1\}^k$	
Return $\mathcal{A}_2(s, (a, c, z(x, w, r, c)))$	

Some of the Σ-protocols also satisfy the following property.

- *Special soundness*: Let (a, c, z) and (a, c', z') be two conversations, that are accepting for some given x, with $c \neq c'$. Then given x and those two conversations, a witness w such that $(x, w) \in R$ can be computed efficiently.

[19] Often this is required to be perfectly indistinguishable, but we generalize the definition slightly to only require computational indistinguishability.

A simple but important fact (see [13]) is that if a Σ-protocol is HVZK, the protocol is *witness indistinguishable* (WI) [25]. Although HVZK by itself is defined with respect to a very much restricted verifier, i.e. an honest one, this means that if for a given instance x there are at least two witnesses w, then even a malicious verifier cannot distinguish which witness the prover uses.

B.1 Ω-protocols

An Ω-protocol $(A,B)_{[\sigma]}$ for a relation $R = \{(x,w)\}$ and CRS σ, is a Σ-protocol for relation R with the following additional properties.

1. For a given distribution ensemble $\mathcal{D}$, a common reference string σ is drawn from $\mathcal{D}_k$ and each function $a(\cdot)$, $z(\cdot)$, and $\phi(\cdot)$ takes σ as an additional input. (Naturally, the simulator M in the definition of Σ-protocols may also take σ as an additional input.)
2. There exists a polynomial-time extractor $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ such that the reference string output by $\mathcal{E}_1(1^k)$ is statistically indistinguishable from $\mathcal{D}_k$. Furthermore, given $(\sigma, \tau) \leftarrow \mathcal{E}_1(1^k)$, if there exists two accepting conversations (a,c,z) and (a,c',z') with $c \neq c'$ for some given $x \in \hat{L}_R$, then $\mathcal{E}_2(x,\tau,(a,c,z))$ outputs w such that $(x,w) \in R$.[20]

Informally, one way to construct Ω-protocols is as follows. Our common reference string will consist of a random public key pk for a semantically-secure encryption scheme. Then for a given $(x,w) \in R$, we will construct an encryption e of w under key pk, and then construct a Σ-protocol to prove that there is a w such that $(x,w) \in R$ and that e is an encryption of w.

As with Σ-protocols, we will use the $\vee$ notation to denote an "OR" protocol, even if one or both of these protocols are Ω-protocols.

C The Universal Composability Framework

In more detail, the execution in the real-life model and the ideal process proceeds basically as follows. The environment $\mathcal{Z}$ drives the execution. It can provide input to a party P_i or to the adversary, $\mathcal{A}$ or $\mathcal{S}$. If P_i is given an input, P_i is activated. In the ideal process P_i simply forwards the input directly to $\mathcal{F}$ (this is the "direct forwarding" that we discussed in the introduction), which is then activated, possibly writing messages on its outgoing communication tape, and then handing activation back to P_i. In the real-life model, P_i follows its protocol, either writing messages on its outgoing communication tape or giving an output to $\mathcal{Z}$. Once P_i is finished, $\mathcal{Z}$ is activated again. If the adversary is activated, it follows its protocol, possibly giving output to $\mathcal{Z}$, and also either corrupting a party, or performing one of the following activities. If the adversary is $\mathcal{A}$ in the real-life model, it may deliver a message from the output communication tape of one honest party to another, or send a message on behalf of a corrupted party. If the adversary is $\mathcal{S}$ in the ideal process, it may deliver a message from $\mathcal{F}$ to a party, or send a message to $\mathcal{F}$. If a party or $\mathcal{F}$ receives a message, it is activated, and once it finishes, $\mathcal{Z}$ is activated

At the beginning of the execution, all participating entities are given the security parameter $k \in \mathbb{N}$ and random bits. The environment is also given an auxiliary input $z \in \{0,1\}^*$. At the end of the execution, the environment outputs a single bit. Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variables describing $\mathcal{Z}$'s output when interacting in the real-life model with adversary $\mathcal{A}$ and players running protocol π, with input z, security parameter k, and uniformly-chosen random tapes for all participating entities. Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the distribution ensemble of random variables

[20] Notice that this extraction property is similar to that of weak special soundness of Σ-protocols, where there exists an accepting conversation even for an invalid proof, but two accepting conversations guarantees that the proof is valid. Here, the extractor can always extract something from any conversation, but it might not be the witness if there is only one accepting conversation. However, having two accepting conversations sharing the same a guarantees that the extracted information is indeed a witness.

describing $\mathcal{Z}$'s output after interacting with adversary $\mathcal{S}$ and ideal functionality $\mathcal{F}$, with input z, security parameter k, and uniformly-chosen random tapes for all participating entities.

A protocol π *securely realizes* an ideal functionality $\mathcal{F}$ if for any real-life adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any auxiliary input, can tell with non-negligible advantage whether it is interacting with $\mathcal{A}$ and players running π in the real-life model, or with $\mathcal{S}$ and $\mathcal{F}$ in the ideal-process. More precisely, $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} \text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, where $\stackrel{c}{\approx}$ denotes *computational indistinguishability*. (In particular, this means that for any $d \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ such that for all $k > k_0$ and for all inputs z, $|\Pr[\,\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\,] - \Pr[\,\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\,]| < k^{-d}$).

To formulate the composition theorem, one must introduce a hybrid model, a real-life model with access to an ideal functionality $\mathcal{F}$. In particular, this $\mathcal{F}$-hybrid model functions like the real-life model, but where the parties may also exchange messages with an unbounded number of copies of $\mathcal{F}$, each copy identified via a unique *session identifier* (*sid*). The communication between the parties and each one of these copies mimics the ideal process, and in particular the hybrid adversary does not have access to the contents of the messages. Let $\text{HYB}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variables describing the output of $\mathcal{Z}$, after interacting in the $\mathcal{F}$-hybrid model with protocol π. Let π be a protocol in the $\mathcal{F}$-hybrid model, and ρ a protocol that secures realizes $\mathcal{F}$. The composed protocol π^ρ is now constructed by replacing the first message to $\mathcal{F}$ in π by an invocation of a new copy of ρ, with fresh random input, the same *sid*, and with the contents of that message as input; each subsequent message to that copy of $\mathcal{F}$ is replaced with an activation of the corresponding copy of ρ, with the contents of that message as new input to ρ.

Canetti [6] proves the following composition theorem.

Theorem C.1 ([6]) *Let $\mathcal{F}$, $\mathcal{G}$ be ideal functionalities. Let π be an n-party protocol that securely realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model, and let ρ be an n-party protocol that securely realizes $\mathcal{F}$. Then protocol π^ρ securely realizes $\mathcal{G}$.*

D Proofs

We present the proofs to some of the theorems in the paper.

First, we present the exclusive collision lemma to be used in some of the proofs. See [31] for a proof.

Lemma D.1 (The Exclusive Collision Lemma) *Let A be a random variable and B_a a random variable whose distribution is parameterized by a value a in the support of A. For every a in the support of A, and for every b_1 and b_2 in the support of B_a, let $\mathsf{Coll}_a(b_1, b_2)$ be a predicate defining a* collision. *Let q be the maximum (over all a in the support of A) probability of a collision of two independent random variables B_a^1 and B_a^2, i.e., $q = \max_a\{\mathsf{Prob}[\mathsf{Coll}_a(B_a^1, B_a^2)]\}$. Let $\phi(a,b)$ be a predicate, and let $p = \mathsf{Prob}[\phi(A, B_A)]$. Let $\phi'(a, b_1, b_2) = \phi(a, b_1) \wedge \phi(a, b_2) \wedge (\neg\mathsf{Coll}_a(b_1, b_2))$. Then we have $\mathsf{Prob}[\phi'(A, B_A^1, B_A^2)] \geq p^2 - q$, where B_A^1 and B_A^2 are independent conditioned on A.*

Proof: (Proof of Theorem 4.1)
Completeness: Straightforward.
Unbounded ZK: By inspection, $\mathcal{S}_1(1^k)$ produces exactly the same distribution as the real protocol. Next, notice that $\mathcal{S}'$ runs $\mathcal{S}_2$ only when $(x, w) \in R$; that the trapdoor property of the SSTC scheme ensures that the faked commitment/decommitment are computationally indistinguishable from the real commitment/decommitment; and that protocol Π is honest-verifier ZK, and is thus witness indistinguishable. The unbounded ZK-ness follows from these facts by a straightforward hybrid argument.
Unbounded simulation soundness: The proof here is quite similar to the proof of unbounded simulation soundness of the USSZK construction in [31]. Roughly speaking, we prove that any adversary that breaks the unbounded simulation soundness of the ZK protocol can either be used to fake

a signature for the strong one-time signature scheme or to open a commitment in two different ways. The first case will violate the security of the strong one-time signature scheme, and the second case will violate the unbounded simulation soundness of the SSTC scheme.

The basic argument is that for an adversary to break the unbounded simulation soundness, one of two cases must hold. The first case is when the adversary creates a new proof accepted by the verifier that uses one of the public keys for the strong one-time signature scheme that were used by the simulator. In this case, for the transcript to be different, it must be that it signs a new transcript, and thus forges in the strong one-time signature scheme.

The second case is when the adversary uses a new public key for the strong one-time signature scheme. Then the adversary's commitment uses a new identifier. Recall that if $x \notin L_R$, then for each first message to Π there is at most one challenge that leads to an accepting conversation. This implies that if the adversary could answer two challenges, it must open its commitment in two different ways, breaking the simulation soundness of the commitment scheme.

For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, recall the experiment $\mathsf{Expt}_{\mathcal{A}}(k)$ in the definition of unbounded simulation sound ZK. Let $p = \Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1]$. Our goal is to show that p is negligible.

Say a *forgery* occurs if $\mathcal{V}$ accepts, and the verification key $\mathsf{sig\_vk}$ in that session was used by $\mathcal{S}_2$, but with a new transcript/signature pair. Let $\mathsf{Expt}^1_{\mathcal{A}}(k)$ be $\mathsf{Expt}_{\mathcal{A}}(k)$ except that if a *forgery* occurs, the experiment halts and fails. Let $p' = \Pr[\mathsf{Expt}^1_{\mathcal{A}}(k) = 1]$.

First, by the existential unforgeability property of SIG_1, we show that the difference between p and p' is negligible. We do this by constructing a non-uniform probabilistic polynomial-time attacker $\mathcal{B}_1$ that can break SIG_1 with probability $\epsilon_1 = \frac{1}{c}(p - p')$, where c is the number of sessions $\mathcal{A}_2$ starts with the simulator in $\mathsf{Expt}_{\mathcal{A}}(k)$. The input to $\mathcal{B}_1$ is a verification key $\mathsf{sig\_vk}$ and a signature oracle $\mathsf{OSign}_{\mathsf{sig\_vk}}$. $\mathcal{B}_1$ chooses $d \stackrel{R}{\leftarrow} \{1, \ldots, c\}$, and then runs the experiment $\mathsf{Expt}_{\mathcal{A}}(k)$, running the simulator and verifier as normal, except for inserting $\mathsf{sig\_vk}$ into the dth instance of $\mathcal{S}_2$ and using $\mathsf{OSign}_{\mathsf{sig\_vk}}$ to perform the signature operation for $\mathsf{sig\_vk}$ in that instance. If a forgery occurs with verification key $\mathsf{sig\_vk}$, $\mathcal{B}_1$ halts and outputs the forgery, i.e., the transcript and signature provided by $\mathcal{A}_2$ for its session with $\mathcal{V}$. The view of $\mathcal{A}$ in this slightly modified experiment is the same as the view of $\mathcal{A}$ in $\mathsf{Expt}_{\mathcal{A}}(k)$ until a forgery occurs. Thus, since a forgery occurs with probability $p - p'$, and since if a forgery occurs, $\mathcal{B}_1$ will break the SIG_1 on $\mathsf{sig\_vk}$ with probability $\frac{1}{c}$, $\mathcal{B}_1$ breaks SIG_1 with probability $\epsilon_1 = \frac{1}{c}(p - p')$.

Next, we show that p' is negligible. We do so by constructing a probabilistic polynomial-time attacker $\mathcal{B}$ that breaks the simulation-sound binding property of the SSTC scheme TC with probability at least $\epsilon_0 = (p')^2 - 2^{-k}$. The input to $\mathcal{B}$ is a public key pk for the SSTC scheme TC and a simulator $\mathcal{S}_{\mathsf{TC}}(sk)$ with the corresponding secret key, as in the definition of the SSTC scheme (Definition 3.1). The breaker $\mathcal{B}$ runs the experiment $\mathsf{Expt}^1_{\mathcal{A}}(k)$, running the simulator $\mathcal{S}$ and verifier $\mathcal{V}$ as normal, except for using pk as the common reference string. When $\mathcal{S}_2$ needs to open a decommitment, $\mathcal{B}$ asks the supplied simulator S_{TC} to do so. Before $\mathcal{V}$ sends a challenge to $\mathcal{A}_2$, $\mathcal{B}$ forks the experiment and continues independently in each sub-experiment (thus giving independent random challenges to $\mathcal{A}_2$). Then $\mathcal{B}$ examines the output (x, tr_1, b_1) and (x, tr_2, b_2) in each sub-experiment. If $b_1 = b_2 = 1$ and $x \notin L_R$ (call this a *successful* sub-experiment), and also the challenges in each sub-experiment are distinct, then we conclude that the adversary $\mathcal{A}$ must have successfully decommitted to two different first messages of protocol Π. In other words, $\mathcal{A}$ has produced $(a_1, \mathsf{dec}_{a_1})$ and $(a_2, \mathsf{dec}_{a_2})$ such that $\mathsf{TCver}(pk, \mathsf{com}_a, a_1, \mathsf{sig\_vk}, \mathsf{dec}_{a_1}) = 1$ and $\mathsf{TCver}(pk, \mathsf{com}_a, a_2, \mathsf{sig\_vk}, \mathsf{dec}_{a_2}) = 1$ for some com_a. We know that it must be the case that $a_1 \neq a_2$, since by weak special soundness of protocol Π, if $x \notin L_R$, then there do not exist two accepting conversations with the same first-message in Π.

Now we determine the success probability of $\mathcal{B}$. First note that for each sub-experiment, the view of $\mathcal{A}$ is perfectly indistinguishable from the view of $\mathcal{A}$ in $\mathsf{Expt}^1_{\mathcal{A}}(k)$, and thus the probability of success in each sub-experiment is p'. Second, note that the probability of a random collision on k-bit challenges is 2^{-k}. Then we can determine the success probability of $\mathcal{B}$ using Lemma D.1, as follows.

A is a random variable denoting possible runs of experiments up to the challenge from $\mathcal{V}$. B_a is a random variable denoting the remainder of a run of an experiment after initial part a in the support of A. For any a in the support of A, and for any b_1 and b_2 in the support of B_a, the predicate $\mathsf{Coll}_a(b_1, b_2)$ is defined to be true if the challenges from $\mathcal{V}$ are equal in b_1 and b_2. Thus a pair (a, b) indicates a full run of the experiment, the predicate $\phi(a, b)$ indicates success in the experiment, and the predicate $\phi(a, b_1, b_2)$ indicates success in each sub-experiment corresponding to runs (a, b_1) and (a, b_2), with the challenges from $\mathcal{V}$ in b_1 and b_2 being distinct. Therefore $\phi(a, b_1, b_2)$ indicates that $\mathcal{B}_0$ succeeds, and hence by Lemma D.1, we see that $\mathcal{B}_0$ succeeds with probability at least $\epsilon_0 = (p')^2 - 2^{-k}$. ∎

Proof: (Proof of Theorem 4.2)
Completeness: Straightforward.
Reference string indistinguishability: Straightforward.
Extractor indistinguishability: It follows from the extractor indistinguishability of $\Pi_{[\sigma]}(x)$.
Unbounded ZK: By inspection, $\mathcal{S}_1(1^k)$ produces exactly the same distribution as the real protocol. Next, notice that $\mathcal{S}'$ runs $\mathcal{S}_2$ only when $(x, w) \in R$; the trapdoor property of the SSTC scheme ensures that the faked commitment/decommitment are computationally indistinguishable from the real commitment/decommitment; and that protocol Π is honest-verifier ZK, and is thus witness indistinguishable. The unbounded ZK-ness follows from these facts by a straightforward hybrid argument.
Extraction: The proof is very similar to the proof of unbounded non-malleability of the NMZK construction in [31]. It is also very similar to the proof of simulation-soundness of Theorem 4.1.

For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, recall the experiments $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$ in the definition of non-malleable ZK. Let $p_1 = \Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1]$ and $p_2 = \Pr[\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k) = 1]$. Our goal is to show that $|p_2 - p_1|$ is negligible.

Say a *forgery* occurs if $\mathcal{V}$ or $\mathcal{E}_2$ accepts, and the verification key $\mathsf{sig\_vk}$ in that session was used by $\mathcal{S}_2$, but with a new transcript/signature pair. Let $\mathsf{Expt}^1_{\mathcal{A}}(k)$ be $\mathsf{Expt}_{\mathcal{A}}(k)$ except that if a *forgery* occurs, the experiment halts and fails. Let $p'_1 = \Pr[\mathsf{Expt}^1_{\mathcal{A}}(k) = 1]$. Similar to the proof of Theorem 4.1, we can show that $p'_1 = p_1 - c\epsilon_1$, where c is the number of sessions $\mathcal{A}_2$ starts with the simulator in $\mathsf{Expt}_{\mathcal{A}}(k)$, and ϵ_1 is negligible.

Now let $\mathsf{Expt}^2_{\mathcal{A}}(k)$ be $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$ except that if a *forgery* occurs, the experiment halts and fails. As above, we can show that $p'_2 = p_2 - c\epsilon_2$, where ϵ_2 is negligible.

Let p'' be the probability in $\mathsf{Expt}^2_{\mathcal{A}}(k)$ that $\mathcal{E}_2(\tau)$ outputs $(1, w)$ for a session with common input x, and $(x, w) \notin R$. Using the extraction property of protocol Π, as in the proof of Theorem 4.1 one can show that there is a non-uniform probabilistic polynomial-time breaker $\mathcal{B}$ that breaks the simulation-sound binding property of the SSTC scheme TC with probability at least $\epsilon_0 = (p')^2 - 2^{-k}$. Thus p'' is negligible.

By extractor indistinguishability again, the probability of producing output $b = 1$ with a unique transcript in $\mathsf{Expt}^1_{\mathcal{A}}(k)$ and $\mathsf{Expt}^2_{\mathcal{A}}(k)$ is the same, so $p'_2 = p'_1 - p''$.

Then $p_1 = p'_1 + c\epsilon_1 = p'_2 + p'' + c\epsilon_1 = p_2 - c\epsilon_2 + c\epsilon_1 + p''$, so $|p_2 - p_1| \leq c\epsilon_1 + c\epsilon_2 + p''$, which is negligible. ∎

Proof: (Proof of Theorem 4.3)

We will prove the theorem for the case of adaptive corruptions. The case of static corruptions is similar. For simplicity, we denote the protocol $\mathsf{MYZK}^R_{[pk,\sigma]}(x)$ with the extra erasing step by $\hat{\Pi}$.

Let $\mathcal{A}$ be an adversary that operates against protocol $\hat{\Pi}$ in the $\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}$-hybrid model. We construct an ideal process adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell whether it is interacting with $\mathcal{A}$ and $\hat{\Pi}$ in the $\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}$-hybrid model, or with $\mathcal{S}$ in the ideal process for $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$.

For simplicity, we will assume only one copy of $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$ is accessed by $\mathcal{Z}$. Obviously we could duplicate the actions of $\mathcal{S}$ for each copy of $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$ (differentiated by the *sid* value).

Formally, let Π be the Ω-protocol in the construction of protocol $\hat{\Pi}$ with simulator $\mathcal{S}_\Pi$ and extractor $\mathcal{E}_\Pi = (\mathcal{E}_{\Pi,1}, \mathcal{E}_{\Pi,2})$.

At the beginning of the ideal process, the ideal adversary $\mathcal{S}$ generates $(\sigma, \tau) \leftarrow \mathcal{E}_{\Pi,1}(1^k)$ and $(pk, sk) \stackrel{R}{\leftarrow} \mathsf{TCgen}(1^k)$, uses (pk, σ) as the common reference string for $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$, and stores sk and τ.

During the ideal process, $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$. Messages received from $\mathcal{Z}$ are forwarded to the simulated $\mathcal{A}$, and messages sent by the simulated $\mathcal{A}$ to its environment are forwarded to $\mathcal{Z}$.

If $\mathcal{S}$ receives a message $(\mathsf{ZK\text{-}PROOF}, sid, ssid, P_i, P_j, x)$ from $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$, i.e., P_i is uncorrupted and has given a witness w to $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$ such that $(x, w) \in R$, then $\mathcal{S}$ simulates P_i using the trapdoor property of the SSTC scheme. In particular, $\mathcal{S}$ generates $(\widetilde{\mathsf{com}}, \xi) \leftarrow \mathsf{TCfakeCom}(pk, sk, \langle P_i, P_j \rangle)$ and sends $(x, \widetilde{\mathsf{com}})$ to P_j as the first message. If P_i receives a challenge c from P_j, $\mathcal{S}$ simulates P_i as follows. First it invokes the simulator $\mathcal{S}_\Pi$ to obtain an accepting conversation $(a, c, z) \leftarrow \mathcal{S}_\Pi(x, \sigma, c)$. Then, $\mathcal{S}$ generates a decommitment for a by setting $\widetilde{\mathsf{dec}} \leftarrow \mathsf{TCfakeDecom}(\xi, \widetilde{\mathsf{com}}, \langle P_i, P_j \rangle, a)$. Finally $\mathcal{S}$ sends $(a, \widetilde{\mathsf{dec}}, z)$ to P_j.

If P_i is corrupted before receiving a challenge, then the witness w is revealed. In this case, $\mathcal{S}$ generates a as a normal prover would, by $a \leftarrow a_\Pi(x, w, r, \sigma)$ where r is chosen randomly. Then $\mathcal{S}$ generates a decommitment $\widetilde{\mathsf{dec}}$ for a, just as in the previous case. Thus all session values (w, a, r, and $\widetilde{\mathsf{dec}}$) can be provided to $\mathcal{A}$.

If P_i is corrupted after sending out the final message, again the witness w is revealed. The session values a and $\widetilde{\mathsf{dec}}$ have already been determined (and sent), and the randomness r has been erased. Thus all non-erased session values (w, a, and $\widetilde{\mathsf{dec}}$) can be provided to $\mathcal{A}$.

If P_j is uncorrupted and receives a first message from a prover P_i, say for a value x in session $ssid$, then $\mathcal{S}$ simulates P_j as in the actual protocol $\hat{\Pi}$, i.e., it sends back a random challenge c. When P_j receives the final message (a, dec_a, z) in session $ssid$, $\mathcal{S}$ performs the verifications specified in protocol $\hat{\Pi}$ (for the decommitment and Ω-protocol Π) and, if these pass, proceeds as follows.

1. If P_i is uncorrupted, $\mathcal{S}$ forwards the message $(\mathsf{ZK\text{-}PROOF}, sid, ssid, P_i, P_j, x)$ to the actual uncorrupted P_j.

2. If P_i is corrupted, but $\mathcal{S}$ had previously received a message $(\mathsf{ZK\text{-}PROOF}, sid, ssid, P_i, P_j, x)$ from $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$ — this happens when P_i is corrupted during a UCZK protocol — then using the witness w that was revealed when P_i was corrupted, $\mathcal{S}$ sends $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, x, w)$ to $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$; and forwards the response from $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$ to P_j.

3. Otherwise, $\mathcal{S}$ runs the extractor $\mathcal{E}_{\Pi,2}(x, \tau, (a, c, z))$ which outputs a potential witness w. $\mathcal{S}$ sends $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, x, w)$ to $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$; and forwards any response from $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$ to P_j.

Now we show that

$$\mathrm{HYB}_{\hat{\Pi},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}} \stackrel{c}{\approx} \mathrm{IDEAL}_{\hat{\mathcal{F}}_{\mathrm{ZK}}^{R},\mathcal{S},\mathcal{Z}},$$

which implies our theorem.

First we define a new experiment $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$. Intuitively, this new experiment is a "mixture" of the hybrid model and the ideal process, in that an uncorrupted party acting as a prover is handled as in the ideal process (i.e., the trapdoor property of the commitment scheme is used to enable it the simulation of a prover in the Ω-protocol), but an uncorrupted party acting as a verifier is handled as in the hybrid model (i.e., no extraction takes place).[21] More precisely, the experiment generates $(\sigma, \tau) \leftarrow \mathcal{E}_{\Pi,1}(1^k)$

[21] It may be tempting to switch these (i.e., in the mixed protocol, the prover is handled as in the hybrid protocol, and the verifier is handled as in the ideal protocol), and argue that simple trapdoor commitments would suffice. The argument would go as follows: (1) The output of the hybrid protocol and the mixed protocol would be indistinguishable, by the extraction property of the Ω-protocol; and (2) The output of the mixed protocol and the ideal protocol would be indistinguishable, by the trapdoorness property of the trapdoor commitment protocol, and by the SHVZK property of the Ω-protocol. However, this argument is flawed. In particular, the SHVZK property does not hold if the adversary is also given access to the knowledge of whether the Ω-protocol extractor is successful. In fact, the success/failure of this extractor distinguishes between a real prover's output and the SHVZK simulator's output.

and $(pk, sk) \stackrel{R}{\leftarrow} \mathsf{TCgen}(1^k)$, and just as in the case of $\text{IDEAL}_{\hat{\mathcal{F}}_{\text{ZK}}^R, \mathcal{S}, \mathcal{Z}}$, (pk, σ) is used as the common reference string for $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$, and sk and τ are stored. Then the experiment runs simulated copies of $\mathcal{Z}$ and $\mathcal{A}$. Messages sent by $\mathcal{Z}$ to the adversary are forwarded to $\mathcal{A}$, and messages sent by $\mathcal{A}$ to its environment are forwarded to $\mathcal{Z}$. If an uncorrupted party P_i receives input $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, x, w)$ from $\mathcal{Z}$ with $(x, w) \in R$, it generates its messages in the same way as $\mathcal{S}$ above. Corruptions are handled in the same way as $\mathcal{S}$ above. An uncorrupted party P_j responds to a prover as in the actual verifier protocol in $\hat{\Pi}$. Finally, the output of $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ is the output of $\mathcal{Z}$.

Let $\text{MIX}_{\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variable describing the outputs of $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$. First, we can show that $\text{HYB}_{\hat{\Pi},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\text{CRS}}^{\mathcal{D}}} \stackrel{c}{\approx} \text{MIX}_{\mathcal{A},\mathcal{Z}}$. This follows from the trapdoor property of the SSTC scheme and a straightforward hybrid reduction to the SHVZK property of the Ω-protocol Π.

Now we show that $\text{MIX}_{\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} \text{IDEAL}_{\hat{\mathcal{F}}_{\text{ZK}}^R, \mathcal{S}, \mathcal{Z}}$, which will finish the proof of the theorem.

Let $\rho = |\Pr[\text{IDEAL}_{\hat{\mathcal{F}}_{\text{ZK}}^R, \mathcal{S}, \mathcal{Z}}(k)] - \Pr[\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)]$. We shall prove that ρ is negligible. Notice that the only difference between experiment $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ and the ideal process is in the case when $\mathcal{S}$ is simulating an uncorrupted verifier P_j, and attempts to extract a witness from a corrupted prover P_i but fails. Thus ρ is a lower bound on the probability of failing to extract a witness.

Now we define an experiment $\mathsf{ExptOne}_{\mathcal{S},\mathcal{Z}}(k)$ that runs $\mathcal{S}$ and $\mathcal{Z}$ in the ideal process, returning 1 on the failure of $\mathcal{S}$ to extract a witness. By the discussion above, $\Pr[\mathsf{ExptOne}_{\mathcal{S},\mathcal{Z}}(k)] \geq \rho$. Now say a session has index (α, β) if it is the βth session between prover P_i and verifier P_j, and (P_i, P_j) is the αth different prover/verifier pair for which a session has been started. In this case, we say P_i (P_j) is the prover (verifier) associated with index (α, β). Let $\mathsf{ExptOne}_{\mathcal{S},\mathcal{Z}}^{(\alpha,\beta)}(k)$ denote the same experiment as above, except that it returns 1 if and only if $\mathcal{S}$ fails to extract a witness for the first time in the session with index (α, β). (Note that we may assume this experiment halts and outputs 0 if, assuming P_i and P_j are the prover and verifier, respectively, associated with index (α, β), P_j is ever corrupted, P_i is uncorrupted when P_j receives the first message in the session with index (α, β), or the session with index (α, β) finishes with a successful extraction.) Then if at most u sessions are started, it is easy to see that for some (α, β) with $\alpha, \beta \in \{1, \ldots, u\}$, $\Pr[\mathsf{ExptOne}_{\mathcal{S},\mathcal{Z}}^{(\alpha,\beta)}(k)] \geq \rho/u^2$. Call the lexicographically first such session index (α_0, β_0).

Now let $\mathsf{ExptTwo}_{\mathcal{S},\mathcal{Z}}^{(\alpha,\beta)}(k)$ denote the same experiment as above, except that if the prover P_i associated with index (α, β) is uncorrupted and starts a session with P_j, then the challenge c from P_j is chosen, an accepting conversation is produced by the simulator $(a, c, z) \leftarrow \mathcal{S}_\Pi(x, \sigma, c)$, the commitment/decommitment pair is produced for a as normal by $(\mathsf{com}_a, \mathsf{dec}_a) \leftarrow \mathsf{TCcom}(pk, a, \langle P_i, P_j \rangle)$, and (x, com_a) is sent to P_j. If P_j receives this message, it sends challenge c, and if P_i receives this challenge, it responds using the z value computed by the simulator. Note that P_i does not use the trapdoor property of the commitment scheme for tag $\langle P_i, P_j \rangle$, but it still simulates the Ω-protocol Π exactly as in $\mathsf{ExptOne}_{\mathcal{S},\mathcal{Z}}^{(\alpha,\beta)}(k)$.[22] By the trapdoor property of the commitment scheme, $\Pr[\mathsf{ExptTwo}_{\mathcal{S},\mathcal{Z}}^{(\alpha_0,\beta_0)}(k)] \geq \psi$, for some $\psi \stackrel{c}{\approx} \rho/u^2$.

Now we construct an adversary $\mathcal{B}$ that breaks the SSTC scheme TC with probability at least $\psi^2 - 2^{-k}$. Since u is polynomial, this will imply that ψ, and hence ρ, is negligible. We describe the adversary $\mathcal{B}$. Let $\mathcal{B}$ take a public key pk of TC along with its corresponding equivocation oracle. First $\mathcal{B}$ chooses random $\alpha, \beta \in \{1, \ldots, u\}$, and then it runs $\mathsf{ExptTwo}_{\mathcal{S},\mathcal{Z}}^{(\alpha,\beta)}(k)$ except for (1) changing the common reference string to include pk as the public key of the SSTC scheme, (2) having $\mathcal{S}$ use the equivocation oracle to fake commitments. Also, before sending a challenge in the session with index (α, β) $\mathcal{B}$ forks the experiment and continues independently in each sub-experiment (i.e., sending random independent challenges in the session with index (α, β) in each sub-experiment). Let Φ be the

[22] Note that we would not be able to complete the simulation if P_j were corrupted, but in this case the experiment would halt and output zero anyway.

event that each sub-experiment halts and outputs 1, and the challenges in each sub-experiment are distinct. If Φ occurs, then we know that $\mathcal{A}$ has decommitted differently in the two sub-experiments. This is because of the extraction property of the Ω-protocol: if $\mathcal{A}$ had decommitted in the same way, then there would exist two accepting conversations with the same first-message, and a witness would have been extracted. Thus $\mathcal{B}$ has obtained two different decommitments for a commitment with tag $\langle P_i, P_j \rangle$. Note that by the authenticated channels assumption, since P_j is uncorrupted, no other party could send a message ostensibly from P_j. Then by the definition of the experiment, the equivocation oracle is not called for the tag $\langle P_i, P_j \rangle$ By Lemma D.1, $\Pr(\Phi) \geq \psi^2 - 2^{-k}$. Therefore, $\mathcal{B}$ breaks the SSTC scheme TC with probability at least $\psi^2 - 2^{-k}$, as claimed above. □

E Signature Scheme Definitions

A *signature scheme* SIG is a triple $(\mathsf{sig\_gen}, \mathsf{sig\_sign}, \mathsf{sig\_verify})$ of algorithms, the first two being probabilistic, and all running in polynomial time (with a negligible probability of failing). $\mathsf{sig\_gen}$ takes as input 1^k and outputs a public key pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk})$, i.e., $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$. $\mathsf{sig\_sign}$ takes a message m and a secret key $\mathsf{sig\_sk}$ as input and outputs a signature σ for m, i.e., $\sigma \leftarrow \mathsf{sig\_sign}(sk, m)$. $\mathsf{sig\_verify}$ takes a message m, a public key vk, and a candidate signature σ' for m as input and returns the bit $b = 1$ if σ' is a valid signature for m for the corresponding private key, and otherwise returns the bit $b = 0$. That is, $b \leftarrow \mathsf{sig\_verify}(\mathsf{sig\_vk}, m, \sigma')$. Naturally, if $\sigma \leftarrow \mathsf{sig\_sign}(\mathsf{sig\_sk}, m)$, then $\mathsf{sig\_verify}(\mathsf{sig\_vk}, m, \sigma) = 1$.

Security for signature schemes We specify existential unforgeability against adaptive chosen-message attacks [35] for a signature scheme $\mathsf{SIG} = (\mathsf{sig\_gen}, \mathsf{sig\_sign}, \mathsf{sig\_verify})$. A forger is given $\mathsf{sig\_vk}$, where $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$, and tries to forge signatures with respect to $\mathsf{sig\_vk}$. It is allowed to query a signature oracle (with respect to $\mathsf{sig\_sk}$) on messages of its choice. It succeeds if after this it can output a valid forgery (m, σ), where $\mathsf{sig\_verify}(\mathsf{sig\_vk}, m, \sigma) = 1$, but m was not one of the messages signed by the signature oracle. We say a forger (t, q, ϵ)-breaks a scheme if the forger runs in time $t(k)$ makes $q(k)$ queries to the signature oracle, and succeeds with probability at least $\epsilon(k)$. A signature scheme SIG is existentially unforgeable against adaptive chosen-message attacks if for all t and q polynomial in k, if a forger (t, q, ϵ)-breaks SIG, then ϵ is negligible in k.

In a *one-time* signature scheme, security is formulated as above except that the adversary may only query the signature oracle once, and we call it "existential unforgeability against chosen-message attacks," since the term "adaptive" only makes sense with multiple queries. We note that one-time signatures scheme can be made very efficient since they don't need public-key cryptographic operations [24]. In a *strong* one-time signature scheme [47], we require that a forger is not even able to produce a different valid signature on a message that was signed by the signature oracle. A strong one-time signature scheme can be constructed from any one-way function [47].

F Number-Theoretic Assumptions

We review some of the number-theoretic assumptions used in this paper.

The Strong RSA assumption. The Strong RSA assumption is a generalization of the standard RSA assumption which (informally) states that given an RSA modulus N and an exponent e, it is computationally infeasible to find the e-th root of a random x. Informally, the strong-RSA assumption states that it is infeasible to find an *arbitrary* non-trivial root of a random x.

More formally, we say that p is a *safe prime* if both p and $(p-1)/2$ are prime. Then let $\mathsf{RSA\text{-}Gen}(1^k)$ be a probabilistic polynomial-time algorithm that generates two random $k/2$-bit safe primes p and q, and outputs $N \leftarrow pq$.

Assumption F.1 (Strong-RSA) *For any non-uniform probabilistic polynomial-size circuit $\mathcal{A}$, the following probability is negligible in k:*

$$\Pr[N \leftarrow \mathsf{RSA\text{-}Gen}(1^k); x \leftarrow \mathbb{Z}_N^*; (y, e) \leftarrow \mathcal{A}(1^k, x, N) : y^e \equiv x \bmod N \wedge e \geq 2]$$

The Strong RSA assumption was introduced by Barić and Pfitzmann [1], and has been used in several applications (see [30, 32, 14]). It is a stronger assumption than the "standard" RSA assumption, yet no method is known for breaking it other than factoring N.

The Cramer-Shoup Signature Scheme Cramer and Shoup [14] presented an efficient signature scheme that is existentially unforgeable against adaptive chosen-message attacks under the Strong RSA Assumption, formally defined in Appendix F. In addition to the main security parameter k, they use a secondary security parameter k' for public key modulus size.[23] The value k' is dependent on k and is set so that known attacks on public key systems with modulus size k' are at least as hard as known attacks on hash functions and other brute-force attacks on systems with main security parameter k. Here we describe their scheme, which we denote $\mathsf{SIG_{CS}} = (\mathsf{sig\_gen_{CS}}, \mathsf{sig\_sign_{CS}}, \mathsf{sig\_verify_{CS}})$.[24]

- $\mathsf{sig\_gen_{CS}}(1^k)$:
  $p, q \leftarrow \textsc{SafePrime}(1^{k'/2})$; $N \leftarrow pq$; $x, h \xleftarrow{R} \mathsf{QR}_N$; $e' \leftarrow \textsc{Prime}(1^{k+1})$;
  $H \leftarrow \textsc{Hash}(1^k)$; $sk \leftarrow \langle p, q \rangle$; $vk \leftarrow \langle N, h, x, e', H \rangle$;
  return (vk, sk).

- $\mathsf{sig\_sign_{CS}}(sk, m)$:
  $y' \xleftarrow{R} \mathsf{QR}_N$; $x' \leftarrow (y')^{e'} \cdot h^{-H(m)} \bmod N$; $e \leftarrow \textsc{Prime}(1^{k+1}) \backslash \{e'\}$;
  $y \leftarrow \left(xh^{-H(x')}\right)^{e^{-1} \bmod \phi(N)} \bmod N$;
  return $\langle e, y, y' \rangle$;

- $\mathsf{sig\_verify_{CS}}(vk, m, \langle e, y, y' \rangle)$:
  if e is not an odd $k+1$ bit number, or $e = e'$, return 0;
  $x' \leftarrow (y')^{e'} \cdot h^{-H(m)} \bmod N$;
  if $x \equiv y^e h^{H(x')} \bmod N$ return 1, else return 0.

As a technical note, instead of an expected polynomial-time algorithm for prime generation, we assume a probabilistic *strict* polynomial-time algorithm that has a negligible probability of failing. This has no effect on the following security result.

Theorem F.2 ([14]) *The Cramer-Shoup signature scheme is secure against adaptive chosen-message attack, under the Strong RSA Assumption and the assumption that H is collision-resistant.*

[23] For today's technology, reasonable values may be $k = 256$ and $k' = 1024$.

[24] Some technical notations: a prime number p is a *safe prime*, if $(p-1)/2$ is also a prime number. $\textsc{SafePrime}(1^n)$ is the set of all n-bit safe prime numbers; $\textsc{Prime}(1^n)$ is the set of all n-bit prime numbers; QR_N is the set of all quadratic residues in $\mathbb{Z}_N^*$, and $\textsc{Hash}(1^n)$ is a set of efficient hash functions that maps strings of arbitrary length to an n-bit string.

DSA The Digital Signature Algorithm [37] was proposed by NIST in April 1991, and in May 1994 was adopted as a standard digital signature scheme in the U.S. [28]. It is a variant of the ElGamal signature scheme [23], and is defined as follows, with two security parameters k and k' as in the Cramer-Shoup signature scheme.[25]

- $\mathsf{sig\_gen}_{\mathsf{DSA}}(1^k)$:
  $q \leftarrow \prime(1^k)$; $p \leftarrow \textsc{Prime}(1^{k'})$, where $q|(p-1)$; $g \xleftarrow{R} \mathbb{Z}_p^*$, where $\mathsf{order}(g) = q$;
  $x \xleftarrow{R} \mathbb{Z}_q$; $y \leftarrow g^x \bmod p$; $sk \leftarrow \langle g, p, q, x\rangle$; $vk \leftarrow \langle g, p, q, y\rangle$;
  return (vk, sk).

- $\mathsf{sig\_sign}_{\mathsf{DSA}}(sk, m)$:
  $v \xleftarrow{R} \mathbb{Z}_q$; $r \leftarrow g^v \bmod p$; $s \leftarrow v^{-1}(H(m) + xr) \bmod q$;
  return $\langle r \mod q, s\rangle$.

- $\mathsf{sig\_verify}_{\mathsf{DSA}}(vk, m, \langle r', s\rangle)$:
  If $0 < r' < q$, $0 < s < q$, and $r' \equiv ((g^{H(m)} y^{r'})^{s^{-1} \bmod q} \bmod p) \bmod q$, return 1, else return 0.

The security of DSA intuitively rests on the hardness of computing discrete logarithms, but there is no known security reduction that proves this. However, it is often simply assumed that DSA is existentially unforgeable against adaptive chosen-message attack.

[25] In the DSA standard, k, k', and H are fixed in the following way: $k = 160$, k' is set to a multiple of 64 between 512 and 1024, inclusive, and hash function H is defined as SHA-1 [27]. However, we will use these parameters as if they could be varied according to the security level desired.