Strengthening Zero-Knowledge Protocols using Signatures

JUAN A. GARAY* PHILIP MACKENZIE* KE YANG†

August 15, 2003

Abstract

Recently there has been an interest in zero-knowledge protocols with stronger properties, such as concurrency, unbounded simulation soundness, non-malleability, and universal composability. In this paper, we show a novel technique to convert a large class of existing honest-verifier zero-knowledge protocols into ones with these stronger properties in the common reference string model. More precisely, our technique utilizes a signature scheme existentially unforgeable against adaptive chosen-message attacks, and transforms any Σ-protocol (which is honest-verifier zero-knowledge) into an unbounded simulation sound concurrent zero-knowledge protocol. We also introduce Ω-*protocols*, a variant of Σ-protocols for which our technique further achieves the properties of non-malleability and/or universal composability.

In addition to its conceptual simplicity, a main advantage of this new technique over previous ones is that it avoids the Cook-Levin theorem, which tends to be rather inefficient. Indeed, our technique allows for very efficient instantiation based on the security of some efficient signature schemes and standard number-theoretic assumptions. For instance, one instantiation of our technique yields a universally composable zero-knowledge protocol under the Strong RSA assumption, incurring an overhead of a small constant number of exponentiations, plus the generation of two signatures.

1 Introduction

The concept of a zero-knowledge (ZK) proof, as defined by Goldwasser, Micali, and Rackoff [32], has become a fundamental tool in cryptography. Informally, if a prover proves a statement to a verifier in ZK, then the verifier gains no information except for being convinced of the veracity of that statement. In particular, whatever the verifier could do after the ZK proof, it could have done before the ZK proof, in some sense because it can "simulate" the proof itself. In early work, Goldreich, Micali and Wigderson [31] showed that any NP statement could be proven in (computational) ZK. In another early work, Goldreich, Micali and Wigderson [30] showed the usefulness of ZK proofs in multiparty protocols, in particular, in having the parties prove the correctness of their computations. There has been a great deal of work since then on all properties of ZK proofs. Here we focus on a few such properties, namely, concurrency, non-malleability, simulation soundness, and universal composability, with our main goal being to construct *efficient* protocols that achieve these properties.

The problem of concurrency was first discussed in Dwork, Naor and Sahai [21]. Informally, the problem arises when many verifiers are interacting with a prover. An adversary controlling all the verifiers may coordinate the timing of their messages so that a simulator would not be able to simulate

*Bell Labs – Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974. E-mail: {garay,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 DIMACS, and by National Science Foundation (NSF) grants CCR-0122581 and CCR-0085982.

the execution of the prover in polynomial time. Canetti *et al.* [11] showed that without additional assumptions, such as timing constraints or a common reference string, logarithmic rounds are necessary to achieve concurrent (black-box) ZK. Prabhakaran, Rosen, and Sahai [47] showed that logarithmic rounds suffice. On the other hand, Damgård [17] showed that concurrent, constant-round ZK protocols can be achieved in the common reference string model. Furthermore, Barak [1] showed that by using a non black-box simulator, constant-round, concurrent protocols can be constructed in the plain model.[1]

The problem of malleability was first pointed out by Dolev, Dwork and Naor [20]. Roughly speaking, the problem is that an adversary may be able to play a "man-in-the-middle" attack on a ZK protocol, playing the role of the verifier in a first protocol, and that of the prover in a second protocol, and such that using information from the first protocol he is able to prove something in the second protocol that he could not prove without that information. A ZK protocol that does not suffer from this problem is said to achieve *one-time non-malleability* (since the adversary only interacts with one prover). Dolev, Dwork and Naor give a construction of a one-time non-malleable ZK protocol that uses a polylogarithmic number of communication rounds. Katz [36] describes efficient protocols for one-time non-malleable proofs of plaintext knowledge for several encryption schemes. His protocols work in the common reference string model, and consist of three rounds and constant number of exponentiations. However, since the witness extractor uses "rewinding," the resulting protocols were only proven secure in a concurrent setting with the introduction of timing constraints. Barak [2] gives a construction of constant-round, one-time non-malleable ZK protocols in the plain model. His construction uses a non-blackbox proof of security and is not very efficient. Sahai [50] provides a definition for one-time non-malleability in the case of non-interactive ZK (NIZK) proofs. De Santis *et al.* [19] generalize this to *unbounded non-malleability* of NIZK proofs, where even any polynomial number of simulator-constructed proofs does not help an adversary to construct any new proof. (As they do, for the remainder of this paper we will simply refer to this property as *non-malleability*, leaving off the "unbounded" modifier.) Their definition is very strong in that (in some sense) it requires a witness to be extractable from the adversary. Further, they introduce the notion of a *robust* NIZK argument, which in addition to being non-malleable, requires the so-called "simulator" of the zero-knowledge property to use a common reference string with the same distribution (uniform) as the one used by the real prover. (Following [19], we call this the *same-string ZK* property.) Finally, they give two constructions of non-malleable (and robust) ZK proofs for any NP language. In fact, these proofs are non-interactive, and thus achieve concurrent (constant-round) ZK.

The notion of simulation soundness for NIZK proofs was introduced by Sahai [50] in the context of chosen-ciphertext security of the Naor-Yung [42] encryption scheme. Informally, an NIZK proof is one-time simulation sound if even after seeing a "simulated proof" (which could be of a false statement) generated by the simulator, the adversary cannot generate a proof for a false statement. Sahai notes that the Naor-Yung encryption scheme would be adaptive chosen-ciphertext secure if it used a one-time simulation-sound NIZK proof. De Santis *et al.* [19] further generalized this notion to *unbounded simulation soundness.* An NIZK proof is unbounded simulation sound if even after seeing any polynomial number of simulated proofs, the adversary cannot generate a proof of a false statement. The non-malleable NIZK protocols given in [19] are also unbounded simulation sound.

The notions of unbounded simulation soundness, non-malleability, and robustness extend naturally to the case of interactive proof systems; we do this in Section 2. Informally, we say an interactive ZK protocol is unbounded simulation-sound if the adversary cannot generate a proof of a false statement, even after interacting with any number of (simulated) provers. (See MacKenzie *et al.* [41] for an application of unbounded simulation sound ZK protocols in a threshold password-authenticated key exchange protocol.) We say a ZK protocol is non-malleable, if there exists an efficient witness extractor

[1] His construction, however, only admits bounded concurrency, meaning that the number of sessions that the protocol can execute concurrently and still retain its zero-knowledge property is at most a *fixed* polynomial in the security parameter.

that successfully extracts a witness from an adversary if the adversary would cause the verifier to accept, even when the adversary is also allowed to interact with any number of (simulated) provers. We note that this definition of non-malleability implies that the ZK protocol is a proof of knowledge, and also that it satisfies the notion of "witness-extended emulation" from Barak and Lindell [3]. Naturally, a non-malleable zero knowledge protocol is also unbounded simulation-sound. Finally, we call a ZK protocol that is non-malleable and same-string, a *robust* ZK protocol.

Universal composability is a notion proposed by Canetti [9] to describe protocols that behave like ideal functionalities, and can be composed in arbitrary ways. Universal composability can be defined in either the adaptive model or the static model, denoting whether the adversary is allowed to adaptively corrupt parties, or must decide which parties to corrupt before the protocol starts, respectively. Universal composability is a very strong notion. For example, a universally composable ZK (UCZK) protocol is both non-malleable (at least in an intuitive sense) and concurrent.

Canetti [9] proved that UCZK protocols do not exist in the "plain" model, where there is no assumption about the system set-up. On the other hand, UCZK is possible in the common reference string model, which is the model we focus on in this paper. As pointed out by Canetti *et al.* [12], the non-malleable NIZK protocols of [19] are also UCZK protocols in the static corruption model. Since they use non-interactive proof techniques and general NP reductions, these protocols are not very efficient. Canetti and Fischlin [10] give a construction of a UCZK protocol for any NP language secure in the adaptive model. Basically, they use a standard three-round ZK protocol for Hamiltonian Cycle, except that they use universally composable commitments as a building block. Damgård and Nielsen [18] use the same general ZK protocol construction as Canetti and Fischlin, but with a more efficient UC commitment scheme.[2] Specifically, for a security parameter k, their UC commitment scheme allows commitment to k bits using a constant number of exponentiations and $O(k)$ bits of communication. Their most efficient UC commitment schemes are based on the p-subgroup assumption [43] or the decisional composite residuosity assumption (DCRA) [44]. Note that even with the more efficient UC commitment scheme, this approach to constructing UCZK protocols tends to be fairly inefficient, since a general NP reduction to Hamiltonian Cycle or SAT is used.

Our results. We show a new technique that allows us to convert certain types of honest-verifier ZK protocols into ZK protocols with the stronger properties described above, i.e., concurrency, unbounded simulation-soundness, non-malleability, robustness, and/or universal composability, in the common reference string model. More precisely, we can

1. transform any Σ-protocol [15] (which are special three-round, honest-verifier protocols where the verifier only sends random bits) into an unbounded simulation-sound ZK protocol; and
2. transform any Ω-protocol (which we introduce in this paper as a variant of Σ-protocols) into a non-malleable ZK protocol, and further into a universally-composable ZK protocol.

The main transformations (sufficient to achieve all results except for UCZK protocols secure in the adaptive model) use a signature scheme that is existentially unforgeable against adaptive chosen-message attacks [32], which exists if one-way functions exist [49], as well as a Σ-protocol to prove knowledge of a signature. Note that one-way functions can be used to construct commitments, and thus if one-way functions exist, Σ-protocols exist for any NP statement (say, through a Cook-Levin reduction, and a standard Σ-protocol for Hamiltonian Cycle). Hence the requirement of our main transformations is the existence of one-way functions. On the other hand, certain signature schemes, such as the Cramer-Shoup [16] scheme and the DSA scheme [38], admit very efficient Σ-protocols. Using these schemes (and at the price of specific number-theoretic assumptions), we are able to construct strengthened ZK protocols that are more efficient than all previously known constructions, since we can completely avoid the Cook-Levin theorem [14, 39]. To further achieve a UCZK protocol

[2] In a later version of their paper, Damgård and Nielsen use SAT instead of Hamiltonian Cycle [18].

that is secure in the adaptive model, we also require a *simulation-sound trapdoor commitment* scheme, a new type of commitment scheme that we introduce and which may be of independent interest. This may be based on trapdoor permutations, but we show a more efficient version based on DSA.

We now sketch the intuition behind our technique. We first select two signature schemes, the second of which being a one-time signature scheme [24].[3] The common reference string will contain a randomly generated verification key vk for the first signature scheme, and hence neither the prover nor the verifier will know the corresponding signing key. We then take an HVZK protocol Π for an NP statement ϕ, and we modify it to Π^*, which consists of (1) a witness indistinguishable (WI) proof for the statement

"Either ϕ is true, or I know the signature for the message vk' w.r.t. verification key vk,"

where vk' is a freshly generated verification key for the one-time signature scheme that is also sent to the verifier, and (2) a signature on the transcript of the WI proof using the secret key corresponding to vk'. Informally, Π^* is the "OR" of Π and a proof of knowledge of a signature on vk'. It turns out that if both Π and the proof of knowledge of the signature are so-called Σ-protocols [15] (see Section 2.2), then Π^* can be constructed from Π very efficiently [15]. Furthermore, if the signature scheme admits a very efficient proof, then the total overhead is very small. In particular, we show that if the Cramer-Shoup signature scheme [16] or the DSA signature scheme [38] is used, then the total overhead is only constant number of exponentiations plus the generation of two signatures.

After the transformation, the completeness of protocol Π is obviously preserved. Protocol Π^* is also zero-knowledge, since a simulator generating the verification key in the common reference string can simultaneously generate the corresponding signing key, and thus has no problem simulating Π^*, by the witness indistinguishability of Π^*. Furthermore, we show that Π^* is unbounded simulation sound: If an adversary $\mathcal{A}$ is able to cause the verifier to accept a false statement after interacting with a polynomial number of (simulated) prover instances, then we show how to construct a machine M, which, having access to the signing oracle and interacting with $\mathcal{A}$, manages to forge a signature.

In order to achieve non-malleability (and also robustness and universal composability) in this paper we introduce Ω-*protocols*, a variant of Σ-protocols that may be of independent interest. In a nutshell, an Ω-protocol is similar to a Σ-protocol but it assumes the existence of a common reference string and allows for the extraction of a witness from a single execution of the protocol without rewinding. As one example, we present an efficient Ω-protocol for the discrete logarithm relation based on the strong RSA assumption [4] and DCRA [44]. As another example, we present a "partial-extracting" Ω-protocol for proving knowledge of the plaintext of an ElGamal ciphertext [23] based on the Decision Diffie-Hellman assumption [5]. We show that if the original protocol Π is an Ω-protocol, then the transformed protocol Π^* is non-malleable, basically by noting that if one could not extract a witness for Π, then one could extract (and thus forge) a signature. Furthermore, the distribution of reference strings output by the simulator in our construction is identical to the distribution of reference strings in the real protocol. Therefore our construction is also robust ZK.

We then show that a non-malleable ZK protocol can be easily augmented to obtain a universally-composable ZK protocol in the static model. Invoking this result, we show as a corollary that (an "augmented" version of) Π^* is also a universally-composable ZK protocol in the static model. Finally, we show that we can further modify Π^* to be a universally composable ZK protocol in the adaptive model (with erasures), while still maintaining efficiency. To achieve this we follow the approach of Damgård [17] and Jarecki and Lysyanskaya [35] of using a trapdoor commitment to commit to the first message of a Σ-protocol, which is then opened when sending the last message. However, it turns out that a "plain" trapdoor commitment scheme does not provide the properties we need to deal with adaptive corruptions. We thus introduce a stronger type of trapdoor commitment scheme, which

[3]The second signature scheme may be the same as the first, although for greater efficiency, a signature scheme that is specifically designed for one-time use may be employed.

we call a *simulation-sound trapdoor commitment* (SSTC) scheme. Furthermore, we demonstrate an efficient construction of an SSTC scheme under the DSA assumption.

Organization of the paper. In Section 2 we present formulations of the various notions of interactive ZK protocols in the common reference string setting, together with some of the building blocks that we will be using in our protocols. In Section 3 we present the construction of unbounded simulation-sound ZK protocols. In Section 4 we introduce Ω-protocols and present the construction of non-malleable (and robust) ZK protocols. In Section 5, we first show that non-malleable ZK implies universally composable ZK assuming static corruptions, and then we demonstrate how to achieve universally composable ZK in the adaptive model with erasures using an SSTC scheme. Finally, in Section 6 we present some efficient instantiations of the constructions above. They include using the Cramer-Shoup signature scheme and/or the DSA signature scheme to construct unbounded simulation-sound ZK protocols and non-malleable ZK protocols; an SSTC scheme based on DSA; an efficient Ω-protocol for the discrete logarithm relation (implying efficient non-malleable ZK and UCZK protocols for discrete logarithm); and a *generalized* Ω-protocol for proving knowledge of the plaintext of an ElGamal ciphertext (implying an efficient non-malleable ZK protocol for ElGamal plaintext knowledge).

2 Preliminaries and Definitions

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.

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

2.1 Zero-knowledge proofs and proofs of knowledge

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.[4] 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:

- $(\textsc{start}, \pi, x, w)$: 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 $\boxed{A}$.

[4] This is similar to the "multi-session extension" concept in Canetti and Rabin [13].

– (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.

Definition 2.1 [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 $\sigma \xleftarrow{R} \mathcal{D}_k$, 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 $|\Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathcal{S}}_{\mathcal{A}}(k) = 1]| \le \beta(k)$, where the experiments $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathcal{S}}_{\mathcal{A}}(k)$ are defined as follows:*

$\mathsf{Expt}_{\mathcal{A}}(\kappa)$:	$\mathsf{Expt}^{\mathcal{S}}_{\mathcal{A}}(\kappa)$:
$\sigma \xleftarrow{R} \mathcal{D}_k$	$(\sigma, \tau) \leftarrow \mathcal{S}_1(1^k)$
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.[5]

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.[6] However, $\mathcal{S}_2$ does not access the witness and will simulate a proof from the input x only.

Definition 2.2 [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.

[5] 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.

[6] $\mathcal{A}$ 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 [50] for more discussion.

The following defines unbounded simulation-sound zero-knowledge (USSZK). This has been useful in applications. In particular, as shown in [50], 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 [41].

Definition 2.3 [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:

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

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 2.4 [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 $tr' \in Q$, $tr \not\lhd tr'$	Return 1 iff $b = 1$, $(x, w) \in R$, and for all $tr' \in Q$, $tr \not\lhd tr'$

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.

To conclude with the ZK definitions, we generalize the notion of robust NIZK in [19] to the interactive setting.

Definition 2.5 [Robust ZK] Π *is a* robust ZK argument of knowledge system *for an NP language L with witness relation R if* Π *is a non-malleable* and *same-string unbounded ZK argument of knowledge system for L.*

2.2 Σ-protocols

Here we overview the basic definitions and properties of Σ-protocols [15]

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[7] 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 $x \in \hat{L}_R$.

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

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 $|\Pr[\mathsf{Expt}_{\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{M}_{\mathcal{A}}(k) = 1]| \le \alpha(k)$, where the experiments $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{M}_{\mathcal{A}}(k)$ are defined as follows:

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

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.

A simple but important fact (see [15]) 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.

In our results to follow, we need a particular, simple instance of the main theorem from [15]. Specifically, we use a slight generalization of a corollary in [15] which enables a prover, given two relations (R_1, R_2), values $(x_1, x_2) \in \hat{L}_{R_1} \times \hat{L}_{R_2}$, and corresponding 3-move Σ-protocols $((A_1, B_1), (A_2, B_2))$, to present a 3-move Σ-protocol (A_{or}, B_{or}) for proving the existence of a w such that either $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$. We call this the "OR" protocol for $((A_1, B_1), (A_2, B_2))$,

We will describe the protocol assuming the challenges from (A_1, B_1) and (A_2, B_2) are of the same length. This can easily be generalized, as long as the challenge length in the combined protocol is at least as long as the challenges from either protocol. The protocol consists of (A_1, B_1) and (A_2, B_2) running in parallel, but with the verifier's challenge c split into $c = c_1 \oplus c_2$, with c_1 as the challenge for (A_1, B_1), and c_2 as the challenge for (A_2, B_2).

The protocol for A_{or} is as follows: Without loss of generality, say A_{or} knows w such that $(x_1, w) \in R_1$. Let M_2 be the simulator for S_2. Then A_{or} runs $M_2(x_2)$ to generate (m, e, z). It sends the first message of (A_1, B_1), along with m as the first message of (A_2, B_2). On challenge c, it chooses $c_2 = e$, and $c_1 = c \oplus c_2$. It is able to provide the final response in (A_1, B_1) because it knows w, and the final response in (A_2, B_2) is simply z. The final message of A_{or} includes c_1 along with the final responses for (A_1, B_1) and (A_2, B_2).

We note that if (A_2, B_2) satisfies special soundness, then (A_{or}, B_{or}) satisfies the following property.

- *Half-weak special soundness*: Let (a, c, z) and (a, c', z') be two conversations, that are accepting for some given (x_1, x_2), with $c \neq c'$. Then either there exists a w_1 such that $(x_1, w_1) \in R_1$ or given x and those two conversations, a witness w_2 such that $(x_2, w_2) \in R_2$ can be computed efficiently.

For two Σ-protocols, (A_1, B_1) and (A_2, B_2), let $(A_1, B_1) \vee (A_2, B_2)$ denote the "OR" protocol for $((A_1, B_1), (A_2, B_2))$.

2.3 Signature schemes

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

prover		verifier
$(vk', sk') \leftarrow \mathsf{sig\_gen}_1(1^k)$	$\xrightarrow{\quad vk' \quad}$	
	$\xleftrightarrow{\Sigma^R(x) \vee \Sigma^{R_{vk}}(vk')}$	
$s \leftarrow \mathsf{sig\_sign}_1(sk', \text{transcript})$	$\xrightarrow{\quad s \quad}$	$\mathsf{sig\_verify}_1(vk', \text{transcript}) \stackrel{?}{=} 1$

Figure 1: $\mathsf{USS}^R_{[vk]}(x)$: An unbounded simulation-sound ZK protocol for relationship R with CRS vk (drawn from the distribution $\mathsf{sig\_gen}_0(1^k)$), and common input x. The prover also knows the witness w such that $R(x, w) = 1$.

input 1^k and outputs a public key pair (sk, vk), i.e., $(sk, vk) \leftarrow \mathsf{sig\_gen}(1^k)$. $\mathsf{sig\_sign}$ takes a message m and a secret key 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, $b \leftarrow \mathsf{sig\_verify}(vk, m, \sigma')$. Naturally, if $\sigma \leftarrow \mathsf{sig\_sign}(sk, m)$, then $\mathsf{sig\_verify}(vk, m, \sigma) = 1$.

Security for signature schemes We specify existential unforgeability against adaptive chosen-message attacks [33] for a signature scheme $\mathsf{SIG} = (\mathsf{sig\_gen}, \mathsf{sig\_sign}, \mathsf{sig\_verify})$. A forger is given vk, where $(sk, vk) \leftarrow \mathsf{sig\_gen}(1^k)$, and tries to forge signatures with respect to vk. It is allowed to query a signature oracle (with respect to sk) on messages of its choice. It succeeds if after this it can output a valid forgery (m, σ), where $\mathsf{sig\_verify}(vk, m, \sigma) = 1$, 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].

3 Unbounded Simulation-Sound ZK

We are now ready to present the first result achieved with our technique: An unbounded simulation-sound zero-knowledge protocol for a relation $R = \{(x, w)\}$. We assume that we have the following building blocks:

1. Σ^R: a Σ-protocol for the binary relation R.
2. $\mathsf{SIG}_0 = (\mathsf{sig\_gen}_0, \mathsf{sig\_sign}_0, \mathsf{sig\_verify}_0)$: a signature scheme secure against adaptive chosen-message attack.
3. $R_{vk} = \{(m, s) \mid \mathsf{sig\_verify}_0(vk, m, s) = 1\}$: a binary relation of message-signature pairs.
4. $\Sigma^{R_{vk}}$: a Σ-protocol with the special soundness property for the binary relation R_{vk}.
5. $\mathsf{SIG}_1 = (\mathsf{sig\_gen}_1, \mathsf{sig\_sign}_1, \mathsf{sig\_verify}_1)$: a one-time signature scheme secure against chosen-message attack.

The protocol $\mathsf{USS}^R_{[vk]}(x)$ is shown in Figure 1. It assumes the prover and verifier share a common input x to a Σ-protocol Σ^R, and the prover knows w such that $(x, w) \in R$. The CRS σ is the verification key vk of a signature scheme that is existentially unforgeable against adaptive chosen-message attacks. The prover generates a pair (vk', sk') for a one-time signature scheme, and sends vk' to the verifier. After this, vk' is the common input to a Σ-protocol $\Sigma^{R_{vk}}$ satisfying special soundness. Then the prover uses the OR construction for Σ-protocols to prove that either $x \in L_R$ or it knows a signature for vk' under verification key vk. (Note that since $\Sigma^{R_{vk}}$ satisfies special soundness, intuitively it is a proof of knowledge.) Finally, the prover signs the transcript with sk', and sends the resulting signature to the verifier.

Now we must describe $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for $\mathsf{USS}^R_{[vk]}(x)$. $\mathcal{S}_1(1^k)$ first generates signature keys $(vk, sk) \leftarrow \mathsf{sig\_gen}_0(1^k)$ and outputs $(\sigma, \tau) = (vk, sk)$. $\mathcal{S}_2(sk)$ first checks that common input $x \in \hat{L}_R$. If not, it aborts. Otherwise it runs the protocol as normal, except generating $s' \leftarrow \mathsf{sig\_sign}_0(sk, vk')$, and using knowledge of s' to complete the Σ-protocol $\Sigma^R(x) \vee \Sigma^{R_{vk}}(vk')$.

Theorem 3.1 *The protocol* $\mathsf{USS}^R_{[vk]}(x)$ *is a USSZK argument.*

Proof: Completeness: Straightforward.
Unbounded ZK: By inspection, $\mathcal{S}_1(1^k)$ produces exactly the same distribution as the real protocol. Then by the fact that $\mathcal{S}'(\tau)$ runs $\mathcal{S}_2(\tau)$ only when $(x, w) \in L_R$, and by the fact that $\Sigma^R(x) \vee \Sigma^{R_{vk}}(vk')$ is a Σ-protocol, and thus witness indistinguishable, unbounded ZK follows by a straightforward hybrid argument.
Unbounded simulation soundness: 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 one-time verification key vk' in that session was used by $\mathcal{S}_2(\tau)$, but on a different transcript. 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 vk' and a one-time signature oracle $\mathsf{OSign}_{vk'}$. $\mathcal{B}_1$ 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 vk' into the dth instance of $\mathcal{S}_2(\tau)$ and using $\mathsf{OSign}_{vk'}$ to perform the signature operation for vk' in that instance. If a forgery occurs with verification key 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 vk' with probability $\frac{1}{c}$, $\mathcal{B}_1$ breaks SIG_1 with probability $\epsilon_1 = \frac{1}{c}(p - p')$.

Now by the existential unforgeability property of SIG_1, we show that p' is negligible. We do this by constructing a non-uniform probabilistic polynomial-time attacker $\mathcal{B}_0$ that can break SIG_0 with at most $2c$ signature oracle queries (again, where c is the number of sessions $\mathcal{A}_2$ starts with the simulator in $\mathsf{Expt}_{\mathcal{A}}(k)$), and with probability at least $\epsilon_0 = (p')^2 - 2^{-k}$.[8] The input to $\mathcal{B}_0$ is a verification key vk and a signature oracle OSign_{vk}. $\mathcal{B}_0$ runs experiment $\mathsf{Expt}^1_{\mathcal{A}}(k)$, running the simulator and verifier as normal, except for inserting vk into the CRS and using OSign_{vk} to perform all signature operations with respect to vk. Also, before $\mathcal{V}$ sends a challenge to $\mathcal{A}_2$, $\mathcal{B}_0$ forks the experiment and continues independently in each sub-experiment (thus giving independent random challenges to $\mathcal{A}_2$). $\mathcal{B}_0$ then 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 $x \notin L_R$

[8]The following argument is a simple version of the Forking Lemma [46], although it does not follow directly, since we are using a signature oracle, and the adversary's output is not actually a signature from that scheme, but a Σ-protocol of knowledge of the signature. Consequently, rather than trying to force our results into the notation of [46] and prove why the Forking Lemma should hold in our situation, we simply prove our result directly.

(call this a *successful* sub-experiment), and also the challenges in each sub-experiment are distinct, then since $\Sigma^R(x) \vee \Sigma^{R_{vk}}(vk')$ satisfies half-weak special soundness, $\mathcal{B}_0$ can generate a signature s on vk' with respect to key vk using the two transcripts tr_1 and tr_2. (Here vk' is the one-time verification key sent in the first message of both tr_1 and tr_2. By the definition of $\mathsf{Expt}^1_{\mathcal{A}}(k)$, vk' could not have been used in any instance of $\mathcal{S}_2$ in either sub-experiment.) Thus $\mathcal{B}_0$ generates a signature (on a new message vk') with respect to vk, and breaks SIG_0. By inspection, $\mathcal{B}_0$ makes at most $2c$ calls to the signature oracle.

Now we determine the success probability of $\mathcal{B}_0$. 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 $\mathcal{B}_0$ using Lemma A.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 A.1, we see that $\mathcal{B}_0$ succeeds with probability at least $\epsilon_0 = (p')^2 - 2^{-k}$. □

4 Non-malleable ZK

Our general NMZK construction will be similar to the USSZK construction above, but with a Σ-protocol replaced by an *Ω-protocol*, defined here.

4.1 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 $(x, w) \in R$.[9]

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.

[9]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.

prover		verifier
$(vk', sk') \leftarrow \mathsf{sig\_gen}_1(1^k)$	$\xrightarrow{\quad vk' \quad}$	
	$\Omega^R_{[\sigma']}(x) \vee \Sigma^{R_{vk}}(vk')$ $\rightleftarrows$	
$s \leftarrow \mathsf{sig\_sign}_1(sk', \text{transcript})$	$\xrightarrow{\quad s \quad}$	$\mathsf{sig\_verify}_1(vk', \text{transcript}) \stackrel{?}{=} 1$

Figure 2: $\mathsf{NM}^R_{[vk,\sigma']}(x)$: A non-malleable ZK protocol for relationship R with common reference string (vk, σ') where σ' is drawn from the distribution associated with $\Omega^R_{\sigma'}$, and common input x.

4.2 NMZK protocol

Let $\Omega^R_{[\sigma']}(x)$ be an Ω-protocol for a relation R with common reference string σ' and common input x. Let $\mathsf{NM}^R_{[vk,\sigma']}(x)$ be the $\mathsf{USS}^R_{[vk]}(x)$ protocol with $\Sigma^R(x)$ replaced by $\Omega^R_{[\sigma']}(x)$. (For every σ', the resultant protocol is also a Σ-protocol.) Let $\mathcal{E}_\Omega = (\mathcal{E}_{\Omega,1}, \mathcal{E}_{\Omega,2})$ be the extractor for $\Omega^R_{[\sigma']}(x)$. The protocol $\mathsf{NM}^R_{[vk,\sigma']}(x)$ is shown in Figure 2.

We now describe $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for $\mathsf{NM}^R_{[vk,\sigma']}$. $\mathcal{S}_1(1^k)$ generates signature keys $(vk, sk) \leftarrow \mathsf{sig\_gen}_0(1^k)$ and then sets $\sigma' \stackrel{R}{\leftarrow} \mathcal{D}_k$, where $\mathcal{D}$ is the distribution ensemble for $\Omega^R_{[\sigma']}$. Next, $\mathcal{S}_1(1^k)$ outputs $((vk, \sigma'), sk)$. $\mathcal{S}_2(sk)$ first checks that common input $x \in \hat{L}_R$. If not, it aborts. Otherwise it runs the protocol as normal, except generating $s' \leftarrow \mathsf{sig\_sign}_0(sk, vk')$, and using knowledge of s' to complete the protocol $\Omega^R_{[\sigma']}(x) \vee \Sigma^{R_{vk}}(vk')$.

Finally, we must describe $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ for $\mathsf{NM}^R_{[vk,\sigma']}(x)$. $\mathcal{E}_1(1^k)$ first generates signatures keys $(vk, sk) \leftarrow \mathsf{sig\_gen}_0(1^k)$, first generates $(\sigma', \tau') \leftarrow \mathcal{E}_{\Omega,1}(1^k)$, and then outputs $((vk, \sigma'), sk, \tau')$. $\mathcal{E}_2(\tau')$ simply runs as $\mathcal{V}$ until $\mathcal{V}$ outputs a bit b. If $b = 1$, $\mathcal{E}_2(\tau')$ takes the conversation (a, c, z) produced by $\Omega^R_{[\sigma']}(x)$, and generates $w \leftarrow \mathcal{E}_{\Omega,2}(x, \tau', (a, c, z))$. If $b = 0$, $\mathcal{E}_2(\tau')$ sets $w \leftarrow \perp$. Then $\mathcal{E}_2(\tau')$ outputs (b, w).

Theorem 4.1 *The protocol* $\mathsf{NM}^R_{[vk,\sigma']}(x)$ *is an NMZK argument of knowledge for the relation R.*

Proof: Completeness: Straightforward.
Reference string indistinguishability: Straightforward.
Extractor indistinguishability: It follows from the extractor indistinguishability of $\Omega^R_{[\sigma']}(x)$.
Unbounded ZK: By inspection, $\mathcal{S}_1(1^k)$ produces exactly the same distribution as the real protocol. Then by the fact that $\mathcal{S}'(\tau)$ runs $\mathcal{S}_2(\tau)$ only when $(x, w) \in L_R$, and by the fact that for every σ', $\Omega^R_{[\sigma']}(x) \vee \Sigma^{R_{vk}}(vk')$ is a Σ-protocol, and thus witness indistinguishable, unbounded ZK follows by a straightforward hybrid argument.
Extraction: 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 one-time verification key vk' in that session was used by $\mathcal{S}_2(\tau)$, but on a different transcript. 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 3.1, 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. (Here we use the fact that by extractor indistinguishability, the number of sessions $\mathcal{A}_2$ starts with the simulator in $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$ is equal to the number of sessions $\mathcal{A}_2$ starts with the simulator in $\mathsf{Expt}_{\mathcal{A}}(k)$.)

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 $\Omega^R_{[\sigma']}(x)$, as in the proof of Theorem 3.1 one can show that there is a non-uniform probabilistic polynomial-time breaker $\mathcal{B}_0$ that makes at most $2c$ oracle queries and breaks SIG_0 with probability at least $\epsilon_0 = (p'')^2 - 2^k$. Thus by the existential unforgeability of SIG_0, 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. ∎

We observe that the construction for protocol $\mathsf{NM}^R_{[vk,\sigma']}(x)$ is in fact same-string unbounded ZK, and thus we have the following.

Corollary 4.2 *The protocol $\mathsf{NM}^R_{[vk,\sigma']}(x)$ is a robust ZK argument of knowledge for the relation R.*

5 Universally Composable ZK

First we review the framework of universal composability [9]. Then we prove that any NMZK protocol with certain simple properties can be augmented to be UCZK in the model of static corruptions. This result implies as a corollary that a slight generalization of our protocol from the previous section can be augmented to be UCZK in this model. Then we give a new construction that is UCZK in the model of adaptive corruptions.

5.1 The universal composability framework

This framework was suggested by Canetti for defining the security and composition of protocols [9]. To define security in this framework, one first specifies an *ideal functionality*, describing the desired behavior of the protocol using a trusted-party. Then one proves that a particular protocol operating in the real world securely realizes this ideal functionality, as defined below. We briefly summarize this framework:[10]

- **Communication model:** We assume an *asynchronous* network, without guaranteed delivery of messages. Further, we assume that the messages are authenticated, since authentication can be added in standard ways (i.e., the $\mathcal{F}_{\text{AUTH}}$ model in [9]).
- **Entities:** The basic entities involved are n parties $P_1, \ldots, P_n$, an adversary $\mathcal{A}$, and an environment $\mathcal{Z}$. All the entities are modeled as probabilistic interactive Turing Machines.
- **Corruptions:** We will specify either *static* or *adaptive* corruptions, as in [9]. In the static case, the adversary corrupts parties only at the onset of the computation; in the adaptive case, the adversary chooses which parties to corrupt as the computation evolves. Once the adversary corrupts a party, it learns all its internal information, including the private input, the communication history, and the random bits used, *except* the information explicitly erased by the party before the corruption. Once they are corrupted, the behavior of the parties is arbitrary, or malicious.
- **Real-life execution:** At a high level, the execution of a protocol π, run by the parties in the presence of $\mathcal{A}$ and an environment machine $\mathcal{Z}$, with input z, is modeled as a sequence of

[10] The material in this section is taken from [9, 12, 13]; refer to these references for further detail.

activations of the entities, with $\mathcal{Z}$ activated first. When $\mathcal{Z}$ is activated, it may write messages on the other entities input tapes (and thus activate it next), and read messages from the other entities output tapes. When $\mathcal{A}$ is activated, it may read messages from a party's outgoing communication tapes, and write a message to a party's incoming communication tapes, thus activating the party. It may also corrupt parties, as discussed above. When a party is activated, it runs the protocol π. (See [9] for more detail on the exact description of all the activations.) Finally, the environment outputs one bit, which is the output of the protocol.

For security parameter $k \in \mathbb{N}$ and input $z \in \{0,1\}^*$, let $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variables describing $\mathcal{Z}$'s output when interacting with adversary $\mathcal{A}$ and parties running protocol π, with input z, security parameter k, and uniformly-chosen random tapes for all the entities.

- **Ideal process:** The security of the protocols is defined by comparing the real execution of the protocol (as described above) to an ideal process in which an additional entity, the ideal functionality $\mathcal{F}$, is introduced; essentially, $\mathcal{F}$ is an incorruptible trusted party that is programmed to produce the desired functionality of the given task. Additionally, the parties are replaced by dummy parties, who do not communicate with each other, but instead have access to $\mathcal{F}$. In this idealized execution, again the environment is activated first, generating the inputs. Whenever a dummy party is activated, it forwards its input to $\mathcal{F}$. Let $\mathcal{S}$ denote the adversary in this idealized execution. $\mathcal{S}$ can see the destinations of the messages between the parties and $\mathcal{F}$, but not the contents. (Again, see [9] for the exact description of the activations.) As in the real-life execution, the output of the protocol execution is the one-bit output of $\mathcal{Z}$.

  Let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the distribution ensemble of random variables describing $\mathcal{Z}$'s output after interacting with adversary $\mathcal{S}$ in the ideal process for $\mathcal{F}$, with input z, security parameter k, and uniformly-chosen random tapes for all the participating entities ($\mathcal{Z}$, $\mathcal{S}$, and $\mathcal{F}$).

- **Security:** In this framework, 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 input, can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running π in the real-life execution, or with $\mathcal{S}$ in the ideal process for $\mathcal{F}$. More precisely, two corresponding binary distribution ensembles are *indistinguishable*, denoted $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ in [9] (meaning 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, $|\Pr[\,\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\,] - \Pr[\,\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\,]| < k^{-d}$).

- **The hybrid model:** Protocols typically would invoke other sub-protocols. The hybrid model is like a real-life execution, except that some invocations of the sub-protocols are replaced by the invocation of an instance of an ideal functionality $\mathcal{F}$; this is called the "$\mathcal{F}$-hybrid model." Specifically, the model is identical to the real-life model, with the addition that besides sending messages to each other, the parties may exchange messages with an unbounded number of copies of $\mathcal{F}$, where each copy is identified via a unique *session identifier* (*sid*). The communication between the parties and each one of these copies mimics the ideal execution.

  Let $\mathrm{HYB}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variables describing the output of $\mathcal{Z}$, after interacting with $\mathcal{A}$ and parties running protocol π in the $\mathcal{F}$-hybrid model. Now let ρ be a protocol that secures realizes $\mathcal{F}$. The composed protocol π^ρ is 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.

Functionality $\mathcal{F}_{\mathrm{ZK}}^{R}$

$\mathcal{F}_{\mathrm{ZK}}^{R}$ proceeds as follows, running with security parameter k, a prover $\mathcal{P}$, a verifier $\mathcal{V}$, and an adversary $\mathcal{S}$:

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

Figure 3: *The zero-knowledge functionality (for relation R)*

- **The composition theorem:** The composition theorem basically says that if ρ secure realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model, for some functionality $\mathcal{G}$, then an execution of the composed protocol π^{ρ}, running in the $\mathcal{G}$-hybrid model, "emulates" an execution of protocol π in the $\mathcal{F}$-hybrid model. That is, no environment machine $\mathcal{Z}$ can distinguish whether it is interacting with $\mathcal{A}$ and π^{ρ} in the $\mathcal{G}$-hybrid model, or it is interacting with $\mathcal{S}$ and π in the $\mathcal{F}$-hybrid model. In other words, $\mathrm{HYB}^{\mathcal{G}}_{\pi^{\rho},\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} \mathrm{HYB}^{\mathcal{F}}_{\pi,\mathcal{S},\mathcal{Z}}$.

The zero-knowledge functionality. We now recall the ideal ZK functionality [9]. As a convention, all the messages from the parties to the ideal functionality take form $(\mathsf{action}, sid, ...)$, where action is in lower case, and all messages from the ideal functionality take form $(\mathsf{OBJECT}, sid, ...)$, where OBJECT is in upper case. The functionality is given in Figure 3. 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.[11] As pointed out in [9], this is actually a proof of knowledge in that the verifier is assured that the prover actually knows w.

One shortcoming of the above formulation is that we will be designing and analyzing protocols in the common reference string model, and so they will be operating in the $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$-hybrid model, where $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ is the functionality that, for a given security parameter k, chooses a string from distribution $\mathcal{D}_k$ and hands it to all parties. However, directly realizing $\mathcal{F}_{\mathrm{ZK}}^{R}$ in the $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$-hybrid model and using the universal composition theorem would result in a composed protocol where a new instance of the reference string is needed for each proof, which 1) is extremely inefficient, and 2) does not reflect the notion of the CRS model, where an unbounded number of protocol instances would use the same copy of the string. Canetti and Rabin [13] suggested the following notion to cope with this problem:

- **Universal composition with joint state:** Let $\mathcal{F}$ and $\mathcal{G}$ be ideal functionalities, and let $\hat{\mathcal{F}}$ denote the "multi-session extension of $\mathcal{F}$," in that $\hat{\mathcal{F}}$ will run multiple copies of $\mathcal{F}$, where each copy is identified by a special *sub-session identifier* (*ssid*). Now let π be a protocol in the $\mathcal{F}$-hybrid model, and let $\hat{\rho}$ be a protocol that securely realizes $\hat{\mathcal{F}}$ in the $\mathcal{G}$-hybrid model. Then construct the composed protocol $\pi^{[\hat{\rho}]}$ by replacing all the copies of $\mathcal{F}$ in π by a single copy of $\hat{\rho}$. The universal composition with joint state theorem states that $\pi^{[\hat{\rho}]}$, running in the $\mathcal{G}$-hybrid model, correctly emulates π in the $\mathcal{F}$-hybrid model.

The definition of $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$, the multi-session extension of $\mathcal{F}_{\mathrm{ZK}}^{R}$, is shown in Figure 4. Note the two types of indices: the *sid*, which, as before, differentiates messages to $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$ from messages sent to other functionalities, and *ssid*, the sub-session ID, which is unique per input message (or proof).

[11] As in [12], we assume there is a symbol $\perp$ such that for any relation R and any string x, $(x, \perp) \notin R$.

Functionality $\hat{\mathcal{F}}_{\text{ZK}}^R$

$\hat{\mathcal{F}}_{\text{ZK}}^R$ proceeds as follows, running with security parameter k, 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 $\mathcal{S}$ and halt. Otherwise, ignore.

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

5.2 NMZK implies UCZK

Let Π be an NMZK protocol between a prover and verifier. We say Π is *augmentable* if the prover sends the first message, and this message contains the common input x, along with auxiliary data aux that may contain any arbitrary public values. (The reason for aux is discussed below.) We will show how to augment Π with additional information in each message to allow it to be used between two parties in the universal composability framework. This augmented protocol is denoted $\hat{\Pi}$, and is constructed as follows.

For an instance of $\hat{\Pi}$ run between parties P_i and P_j, set aux to $(ssid, P_i, P_j)$, where $ssid$ is defined in the previous section, P_i is the identity of the prover, and P_j is the identity of the verifier.[12] Then the ℓth prover message is formatted as $(\mathsf{prv}_\ell, sid, ssid, P_i, \mathsf{prv\text{-}data}_\ell)$, where prv_ℓ is the label for the ℓth prover message, and $\mathsf{prv\text{-}data}_\ell$ is the data field containing the ℓth message sent by the prover in Π. Analogously, the ℓth verifier message is formatted as $(\mathsf{ver}_\ell, sid, ssid, P_j, \mathsf{ver\text{-}data}_\ell)$, where ver_ℓ is the label for the ℓth verifier message, and $\mathsf{ver\text{-}data}_\ell$ is the data field containing the ℓth message sent by the verifier in Π. Finally, before accepting, the verifier checks that aux corresponds to the values $(ssid, P_i, P_j)$ outside the prover data field, and that aux was not used previously.

Theorem 5.1 *Let $\Pi = (\mathcal{D}, \mathcal{P}, \mathcal{V}, \mathcal{S}_\Pi = (\mathcal{S}_{\Pi,1}, \mathcal{S}_{\Pi,2}), \mathcal{E}_\Pi = (\mathcal{E}_{\Pi,1}, \mathcal{E}_{\Pi,2}))$ be an augmentable NMZK protocol for a relation R. Then the augmented protocol $\hat{\Pi}$ securely realizes functionality $\hat{\mathcal{F}}_{\text{ZK}}^R$ in the $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$-hybrid model, assuming static corruptions.*

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

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

Simulator $\mathcal{S}$ generates $(\sigma, \tau_1, \tau_2) \leftarrow \mathcal{E}_{\Pi,1}(1^k)$, uses σ as the common reference string for $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$, and stores τ_1 and τ_2.

Simulator $\mathcal{S}$ 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 $\hat{\mathcal{F}}_{\text{ZK}}^R$, i.e., P_i is uncorrupted and wishes to perform a ZK proof for common input x, then $\mathcal{S}$ simulates P_i in $\hat{\Pi}$. In particular, $\mathcal{S}$ sets the prover data field in the messages of P_i using protocol $\mathcal{S}_{\Pi,2}(\tau_1)$. If P_j is also uncorrupted, then $\mathcal{S}$ simulates P_j in $\hat{\Pi}$, setting the verifier data field in the messages of P_j using the actual verifier

[12]This auxiliary data aux is necessary since NMZK allows copying proofs exactly, but the ZK functionality does not, and thus we need some way to make every proof distinct.

protocol. In this case, when the simulated P_j receives the final message from the simulated P_i, $\mathcal{S}$ forwards $(\mathsf{ZK\text{-}PROOF}, sid, ssid, P_i, P_j, x)$ to the actual uncorrupted P_j.

If $\mathcal{A}$, controlling a corrupted party P_i, starts an interaction as a prover with an uncorrupted party P_j using $ssid$, then $\mathcal{S}$ learns common input x (since it is included in the first message) and simulates P_j in $\hat{\Pi}$. In particular, it sets the verifier data field in the messages of P_j using protocol $\mathcal{E}_{\Pi,2}(\tau_2)$. At the end of the interaction $\mathcal{E}_{\Pi,2}(\tau_2)$ will output (b, w). If $b = 1$, $\mathcal{S}$ sends $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, x, w)$ to $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$; otherwise, it sends nothing. Then it forwards any response from $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$ to P_j.

Now we show that $\mathrm{HYB}^{\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}}_{\hat{\Pi},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\hat{\mathcal{F}}^R_{\mathrm{ZK}},\mathcal{S},\mathcal{Z}}$.

First we define a new experiment $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$. The new experiment runs simulated copies of $\mathcal{Z}$ and $\mathcal{A}$. 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 $\mathcal{Z}$. The simulator for Π, $\mathcal{S}_{\Pi,1}(1^k)$ is run to produce (σ, τ), and queries to $\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}$ are answered with σ. 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 sets the prover data field of its messages by running protocol $\mathcal{S}_{\Pi,2}(\tau)$ with reference string σ, and common input x. An uncorrupted party P_j responds to a prover as in the actual verifier protocol in $\hat{\Pi}$. The output of each experiment is the output of $\mathcal{Z}$.

Let $\mathrm{MIX}_{\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variable describing the outputs of $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$.

By the unbounded ZK property, we have $\mathrm{HYB}^{\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}}_{\hat{\Pi},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{MIX}_{\mathcal{A},\mathcal{Z}}$. To see this, note that we could construct an adversary $\mathcal{A}'$ that takes a reference string and runs the protocol $\hat{\Pi}$, except that $\mathcal{A}'$ calls a protocol wrapper with label $\mathsf{aux} = (ssid, P_i, P_j)$ when simulating uncorrupted parties acting as provers. If the wrapper contains an actual prover, then the distribution of outputs of $\mathcal{A}'$ will be the same as $\mathrm{HYB}^{\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}}_{\hat{\Pi},\mathcal{A},\mathcal{Z}}$, and if the wrapper contains a simulator, then the distribution will be the same as $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$.

Now we must show that $\mathrm{MIX}_{\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\hat{\mathcal{F}}^R_{\mathrm{ZK}},\mathcal{S},\mathcal{Z}}$. This will follow from the unbounded extraction property (see Lemma 5.3). Say that the two distributions can be distinguished with probability $\gamma(k)$. Since both $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ and $\mathcal{S}$ run the same simulation for the prover, and the output messages of the extractor run by $\mathcal{S}$ are perfectly indistinguishable from the output messages of the verifier, the only difference comes from when the extractor outputs an incorrect witness for a session started by $\mathcal{A}$, and thus $\mathcal{Z}$ receives an output message (indicating a correct proof) in $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ but not when interacting with $\mathcal{S}$. (Note that the transcripts of corrupted prover/uncorrupted verifier sessions will never be the same as transcripts of uncorrupted prover/corrupted verifier sessions because of the auxiliary data aux.) Let $\vec{b}$ be the vector corresponding to simulated verifier sessions, with $b = 1$ corresponding to whether $\mathcal{Z}$ receives an output message. Then the statistical difference between the distribution of vectors $\vec{b}$ resulting from $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ and vectors $\vec{b}$ resulting from $\mathcal{S}$ is at least $\gamma(k)$.

Now we construct an adversary $\mathcal{A}'$ that takes a reference string and runs $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ except that it uses the given reference string instead of generating a new one, and that it calls a "simulator" protocol wrapper when simulating uncorrupted parties acting as provers with corrupted verifiers, and a "verifier" protocol wrapper when simulating uncorrupted parties acting as verifiers with corrupted provers. Then in $\mathsf{Expt}_{\mathcal{A}'}(k)$, the vector $\vec{b}$ will have the same distribution as the one resulting from $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$. On the other hand, in $\mathsf{Expt}^{\mathcal{E}_\Pi}_{\mathcal{A}'}(k)$, the vector $\vec{b}$ will have the same distribution as the one resulting from $\mathcal{S}$, up until $\mathcal{Z}$ receives an output message in $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ that would not have appeared in $\mathcal{S}$. It should be clear that the distributions of $\vec{b}$ in the two experiments are statistically distinguishable with the same probability as the distributions of $\vec{b}$ resulting from $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ and $\mathcal{S}$, i.e., $\gamma(k)$. By the unbounded extraction property, $\gamma(k)$ is negligible. □

Definition 5.2 [Unbounded-Extraction NMZK 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 an* unbounded-extraction non-malleable ZK proof *(resp.,*

argument*) of knowledge system for an NP language L with witness relation R if Π is an NMZK proof (resp. argument) system for L and furthermore, there exists a negligible function $\beta(k)$ such that for all k,*

Unbounded 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 $\sum_{v \in \{0,1\}^*} |\Pr[\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k) = v] - \Pr[\mathsf{Expt}_{\mathcal{A}}(k) = v]| \le \beta(k)$, where the experiments $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$ are defined as follows:*

$\mathsf{Expt}_{\mathcal{A}}(k)$: $\sigma \leftarrow \mathcal{S}_1(1^k)$ $(\vec{x}, \vec{tr}, \vec{b}) \leftarrow (\langle \boxed{\mathcal{S}''(\tau_1)}, \mathcal{A}_1\rangle, \langle \mathcal{A}_2, \boxed{\mathcal{V}}\rangle)_{[\sigma]}$ Let Q be the set of transcripts of machines in $\boxed{\mathcal{S}''(\tau_1)}$. For all i, if $\exists tr' \in Q$, $tr[i] \bowtie tr'$ then $b[i] \leftarrow 0$ Return $\vec{b}$	$\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$: $(\sigma, \tau_1, \tau_2) \leftarrow \mathcal{E}_1(1^k)$ $(\vec{x}, \vec{tr}, (\vec{b}, \vec{w})) \leftarrow (\langle \boxed{\mathcal{S}''(\tau_1)}, \mathcal{A}_1\rangle, \langle \mathcal{A}_2, \boxed{\mathcal{E}_2(\tau_2)}\rangle)_{[\sigma]}$ Let Q be the set of transcripts of machines in $\boxed{\mathcal{S}''(\tau_1)}$. For all i, if $(x[i], w[i]) \notin R$ or $\exists tr' \in Q$, $tr[i] \bowtie tr'$ then $b[i] \leftarrow 0$ Return $\vec{b}$

where we use the vector output notation to denote that the ith instance started in a wrapper protocol $\boxed{A}$ returns $(x[i], tr[i], v[i])$, where $v[i]$ is the output of A, and where $\mathcal{S}''(\tau)$ runs as follows on common reference string σ, common input x and private input w: $\mathcal{S}''(\tau)$ runs $\mathcal{S}_2(\tau)$ on common reference string σ and common input x.

Lemma 5.3 *Let $\Pi = (\mathcal{D}, \mathcal{P}, \mathcal{V}, \mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2), \mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2))$ be an NMZK protocol for a relation R. Then Π is an unbounded extraction NMZK protocol for R.*

Proof: First notice that since $\mathcal{V}$ and $\mathcal{E}_2(\tau_2)$ have exactly the same behavior, there will be an exact correspondence of vectors returned in the two experiments, except that in some cases, some bits that were 1 in $\mathsf{Expt}_{\mathcal{A}}(k)$ would be 0 in $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$. Let $\beta_{\mathcal{A}}(k) = \sum_{v \in \{0,1\}^*} |\Pr[\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k) = v] - \Pr[\mathsf{Expt}_{\mathcal{A}}(k) = v]|$.

Now we perform a hybrid argument. Let $\mathsf{Expt}^{\mathcal{E},j}_{\mathcal{A}}(k)$ be the same as $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$ except that "For all i" is replaced with "For all $i \le j$." Let ℓ denote the maximum number of sessions of $\mathcal{E}_2(\tau_2)$ started by $\mathcal{A}_2$, and notice that ℓ is polynomial in k. Then $\mathsf{Expt}^{\mathcal{E},0}_{\mathcal{A}}(k)$ is the same as $\mathsf{Expt}_{\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathcal{E},\ell}_{\mathcal{A}}(k)$ is the same as $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$. By a telescoping argument, $\sum_{j=1}^{\ell} \sum_{v \in \{0,1\}^*} |\Pr[\mathsf{Expt}^{\mathcal{E},j}_{\mathcal{A}}(k) = v] - \Pr[\mathsf{Expt}^{\mathcal{E},j-1}_{\mathcal{A}}(k) = v]| \ge \beta_{\mathcal{A}}(k)$. Now let $\mathsf{Expt}^{\mathcal{E},j,1}_{\mathcal{A}}(k)$ be the same as $\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}}(k)$ except that "For all i" is replaced with "For $i = j$." Because $\mathcal{V}$ and $\mathcal{E}_2(\tau_2)$ have exactly the same behavior, it is easy to verify that $\sum_{j=1}^{\ell} \sum_{v \in \{0,1\}^*} |\Pr[\mathsf{Expt}^{\mathcal{E},j,1}_{\mathcal{A}}(k) = v] - \Pr[\mathsf{Expt}^{\mathcal{E},j-1,1}_{\mathcal{A}}(k) = v]| \ge \beta_{\mathcal{A}}(k)$.

Now consider a new adversary $\mathcal{A}' = (\mathcal{A}_1, \mathcal{A}'_2)$ that chooses $j \in \{1, \ldots, \ell\}$ randomly, where $\mathcal{A}'_2$ runs $\mathcal{A}_2$ but simulates $\mathcal{V}$ in all but the jth session. In the jth session it calls the one-time wrapper given to it. From the definition of NMZK, $|\Pr[\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}'}(k) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}'}(k) = 1]| \le \alpha(k)$, and by the analysis above,

$$
\begin{aligned}
&|\Pr[\mathsf{Expt}^{\mathcal{E}}_{\mathcal{A}'}(k) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}'}(k) = 1]| \\
&= \frac{1}{\ell} \sum_{j=1}^{\ell} \sum_{v \in \{0,1\}^*} |\Pr[\mathsf{Expt}^{\mathcal{E},j,1}_{\mathcal{A}}(k) = v] - \Pr[\mathsf{Expt}^{\mathcal{E},j-1,1}_{\mathcal{A}}(k) = v]| \\
&\ge \frac{\beta_{\mathcal{A}}(k)}{\ell},
\end{aligned}
$$

so $\beta_{\mathcal{A}}(k) \le \ell \cdot \alpha(k)$. The theorem follows. $\square$

We say a protocol $\hat{\Pi}$ is a *UCZK protocol* for R if it securely realizes functionality $\hat{\mathcal{F}}_{\mathrm{ZK}}^{R}$ in the $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$-hybrid model, for some $\mathcal{D}$.

Corollary 5.4 *Let* Π *be protocol* $\mathsf{NM}^{R}_{[vk,\sigma']}(x)$ *from Figure 2 with the addition of the common input* x *and* $\mathsf{aux} = (ssid, P_i, P_j)$ *in the first message. Then the augmented protocol* $\hat{\Pi}$ *is a UCZK protocol for* R*, assuming static corruptions.*

5.3 UCZK: Adaptive corruptions

To deal with adaptive corruption, we apply a technique proposed by Damgård [17] and Jarecki and Lysyanskaya [35] in which a *trapdoor commitment* is used to commit to the first message of a Σ-protocol, and then this commitment is opened when sending the third message. Informally, a trapdoor commitment is a commitment scheme with the additional property that there is a secret trapdoor such that knowing the trapdoor allows a committer to decommit to an arbitrary value. More precisely, $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{TCcom}, \mathsf{TCver}, \mathsf{TCkeyver}, \mathsf{TCfake})$ is a trapdoor commitment scheme if it satisfies the properties of completeness, binding, perfect secrecy, and trapdoorness. The first three properties are the same as in any unconditionally-hiding commitment scheme. The trapdoor property says (informally) that $\mathsf{TCgen}(1^k)$ outputs a secret key (the trapdoor) along with the public key, and that using this secret key and a commitment/decommitment pair (c, d) associated with a value v, (i.e., $(c, d) \leftarrow \mathsf{TCcom}(pk, v)$), the function TCfake can for any value v' output a decommitment d' that is a valid decommitment of c resulting in v' (i.e., $\mathsf{TCver}(pk, c, v', d') = 1$).

However, this technique alone does not seem to yield a UCZK protocol for adaptive corruption. There are two problems remaining. First, it doesn't yield a non-rewinding witness extractor, which is needed for UCZK. Second, in the setting of UCZK, an ideal adversary $\mathcal{S}$ might use the trapdoor to "cheat", i.e., to decommit to arbitrary values, while at the same time it still needs the binding property for the real-life adversary $\mathcal{A}$. A "plain" trapdoor commitment scheme doesn't provide such a guarantee.

We solve these two problems by 1) using an Ω-protocol in the place of the Σ-protocol; recall that Ω-protocols allow for non-rewinding extractors, and 2) introducing a stronger type of trapdoor commitment scheme, which we call a *simulation-sound trapdoor commitment* (SSTC) scheme.[13] Roughly speaking, an SSTC scheme is a trapdoor commitment scheme with an extra input id to the commitment protocol, which guarantees that a commitment made by the adversary using input id is binding, even if the adversary has seen any commitment using input id opened (using a simulator that knows a trapdoor) once to any arbitrary value, and moreover, any commitment using $id' \neq id$ opened (again using the simulator) an unbounded number of times to any arbitrary values. Such a trapdoor commitment scheme enables an ideal adversary to "cheat" while maintaining the binding property for the real-life adversary. We shall see that when we apply these two solutions, the protocol becomes universally composable with respect to adaptive corruption.[14]

Here we formally define an SSTC scheme, building on the formalization for trapdoor commitment schemes by Reyzin [48].

Definition 5.5 [Simulation-Sound Trapdoor Commitment (SSTC) Scheme] $\mathsf{TC} = (\mathsf{TCgen}, \mathsf{TCcom}, \mathsf{TCver}, \mathsf{TCkeyver}, \mathsf{TCfake})$ *is an* SSTC scheme *if* TCgen, TCcom, TCver, $\mathsf{TCkeyver}$, *and* TCfake *are probabilistic polynomial-time algorithms such that*

[13] Universally-composable commitments [10, 12] would also suffice, and can be constructed using trapdoor permutations. However, this construction is not as efficient as the SSTC scheme in this paper.

[14] As a technical note, we comment that on the face, this construction doesn't use the technique of adding a proof of knowledge of signature, as in previous constructions. However, such a technique will be used in the construction of the SSTC schemes.

Completeness *For all id and for all values v,*

$$\Pr[(pk, sk) \overset{R}{\leftarrow} \mathsf{TCgen}(1^k); (c, d) \overset{R}{\leftarrow} \mathsf{TCcom}(pk, v, id) : \mathsf{TCkeyver}(pk, 1^k) = \mathsf{TCver}(pk, c, v, id, d) = 1] = 1.$$

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

$$\Pr[(pk, sk) \overset{R}{\leftarrow} \mathsf{TCgen}(1^k); (c, id, v_1, v_2, d_1, d_2) \overset{R}{\leftarrow} \langle \mathcal{S}(sk), \mathcal{A}\rangle(pk) : (\mathsf{TCver}(pk, c, v_1, id, d_1) = \mathsf{TCver}(pk, c, v_2, id, d_2) = 1) \wedge (v_1 \neq v_2) \wedge id \notin Q] \leq \alpha(k),$$

where $\mathcal{S}(sk)$ operates as follows, with Q initially set to $\emptyset$:

- *On input* (commit, v, id)*:* *compute $(c, d) \leftarrow \mathsf{TCcom}(pk, v, id)$, store (c, v, id, d), and return c.*
- *On input* $(\mathsf{decommit}, c, v')$*:* *if for some v, id, d a tuple (c, v, id, d) is stored, compute $d' \leftarrow \mathsf{TCfake}(pk, sk, c, v, id, d, v')$. If some previous $(\mathsf{decommit}, c, *)$ has been input, add id to Q. Return d'.*

Hiding *For all pk such that $\mathsf{TCkeyver}(pk, 1^k) = 1$, for all id, and for all v_1, v_2 of equal length, the following probability distributions are identical:*

$$\{(c_1, d_1) \overset{R}{\leftarrow} \mathsf{TCcom}(pk, v_1, id) : c_1\} \text{ and } \{(c_2, d_2) \overset{R}{\leftarrow} \mathsf{TCcom}(pk, v_2, id) : c_2\}.$$

Trapdoor Property *For all (pk, sk) generated with non-zero probability by $\mathsf{TCgen}(1^k)$, for all id, and for all v_1, v_2 of equal length, the following probability distributions are identical:*

$$\{(c, d_1) \overset{R}{\leftarrow} \mathsf{TCcom}(pk, v_1, id); d_2' \overset{R}{\leftarrow} \mathsf{TCfake}(pk, sk, c, v_1, id, d_1, v_2) : (c, d_2')\}$$

and

$$\{(c, d_2) \overset{R}{\leftarrow} \mathsf{TCcom}(pk, v_2, id) : (c, d_2)\}.$$

(In particular, faked commitments are correct.)

Now, let Π be an augmentable Ω-protocol with common input x, auxiliary input aux, prover random bits r, and common reference string σ. As for Σ-protocols, we use the notation $a_\Pi(\cdot)$, $z_\Pi(\cdot)$, and $\mathsf{verify}_\Pi(\cdot)$ to denote the algorithms for computing the two messages of the prover, and for verifying the proof, respectively. Using this notation, the protocol $\mathsf{UC}^R_{[pk^*,\sigma]}(x; \mathsf{aux})$ is shown in Figure 5.

Theorem 5.6 *Let Π be the protocol $\Omega^R_{[pk^*,\sigma]}(x; \mathsf{aux})$, where $\mathsf{aux} = (ssid, P_i, P_j)$. Then the augmented protocol $\hat{\Pi}$ securely realizes functionality $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$ in the $\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}$-hybrid model where erasing is allowed, assuming adaptive corruptions.*

Proof: 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 an Ω-protocol 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 $(\sigma, \tau) \leftarrow \mathcal{E}_{\Pi,1}(1^k)$, generates $(pk^*, sk^*) \overset{R}{\leftarrow} \mathsf{TCgen}(1^k)$, uses (pk^*, σ) as the common reference string for $\mathcal{F}^{\mathcal{D}}_{\mathrm{CRS}}$, and stores sk^* and τ.

prover		verifier
$(x, \mathsf{aux}, a) \leftarrow a_\Pi(x, \mathsf{aux}, w, r, \sigma)$		
$(a^*, d^*) \leftarrow \mathsf{TCcom}(pk^*, a, \mathsf{aux})$	$\xrightarrow{x,\, \mathsf{aux},\, a^*}$	
		$c \xleftarrow{R} \{0,1\}^k$
	$\xleftarrow{\quad c \quad}$	
$z \leftarrow z_\Pi(x, \mathsf{aux}, w, r, c, \sigma)$		
$\mathsf{erase}(r, w)$	$\xrightarrow{z,\, a,\, d^*}$	$\mathsf{TCkeyver}(pk^*, 1^k)$
		$\mathsf{TCver}(pk^*, a^*, a, \mathsf{aux}, d^*)$
		$\mathsf{verify}_\Pi(x, \mathsf{aux}, a, c, z, \sigma)$

Figure 5: $\mathsf{UC}^R_{[pk^*,\sigma]}(x; \mathsf{aux})$: A UCZK protocol for R with common reference string (pk^*, σ) drawn from $\mathcal{D}_{pk}(\mathsf{TC}) \times \mathcal{D}_\sigma(\Omega^R)$, common input x, and auxiliary input aux where $\Pi = \Omega^R_\sigma(x; \mathsf{aux})$.

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 $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$, i.e., P_i is uncorrupted and has given a witness w to $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$ such that $(x, w) \in R$, then $\mathcal{S}$ simulates P_i in $\hat{\Pi}$. In particular, $\mathcal{S}$ sets the prover data field in the first message of P_i by generating a commitment (as in the actual prover protocol) to an arbitrary string $\hat{a}$ with appropriate length (say, $\hat{a} = 0^l$, where l is the size of field "a" in the output of $a_\Pi(\cdot)$). More precisely, $\mathcal{S}$ invokes $(\hat{a}^*, \hat{d}^*) \leftarrow \mathsf{TCcom}(pk^*, \hat{a}, \mathsf{aux})$ and sends $(x, \mathsf{aux}, \hat{a}^*)$ to P_j as the first message. After receiving the challenge (as the second message) c from P_j, $\mathcal{S}$ invokes the simulator $\mathcal{S}_\Pi$ and obtains $(a, c, z) = M_\Pi(x, \sigma, c)$. Then, $\mathcal{S}$ fakes a decommitment for a by invoking $d^* = \mathsf{TCfake}(pk^*, sk^*, \hat{a}^*, \hat{a}, \mathsf{aux}, \hat{d}^*, a)$, and sends (z, a, d^*) to P_j as the final message. If P_i is corrupted before receiving a challenge, then the witness w is revealed. In this case, $\mathcal{S}$ invokes the actual first-message function a_Π to produce the first message a, instead of using the simulator $\mathcal{S}_\Pi$. Again, $\mathcal{S}$ fakes a decommitment in this case.

If P_j is also uncorrupted, then $\mathcal{S}$ simulates P_j in $\hat{\Pi}$, setting the verifier data field in the message of P_j (in particular, the random challenge) using the actual verifier protocol. In this case, when the simulated P_j receives the final message from the simulated P_i, $\mathcal{S}$ forwards $(\mathsf{ZK\text{-}PROOF}, sid, ssid, P_i, P_j, x)$ to the actual uncorrupted P_j.

If $\mathcal{A}$, controlling a corrupted party P_i, starts an interaction as a prover with an uncorrupted party P_j using $ssid$, then $\mathcal{S}$ learns common input x (since it is included in the first message) and simulates P_j (as the verifier) in $\hat{\Pi}$.More precisely, it fills the verifier data field with a random challenge c, receives as the final message (z, a, d^*) from $\mathcal{A}$, and verifies the messages. At the end of the interaction, if all the verifications pass, the extractor $\mathcal{E}_{\Pi,2}(x, \tau, (a, c, z))$ will be invoked and output a witness w. If $R(x, w) = 1$, $\mathcal{S}$ sends $(\mathsf{zk\text{-}prover}, sid, ssid, P_i, P_j, x, w)$ to $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$; otherwise, it sends nothing. Then it forwards any response from $\hat{\mathcal{F}}^R_{\mathrm{ZK}}$ to P_j.

Now we show that

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

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., $\mathcal{S}$ will use the trapdoor to simulate a proof), but an uncorrupted party acting as a verifier is handled as in the hybrid model (i.e., no extraction takes place). More precisely, the new experiment runs simulated copies of $\mathcal{Z}$ and $\mathcal{A}$. 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 $\mathcal{Z}$. $\mathcal{E}_{\Pi,1}(1^k)$ is run to produce (σ, τ), then $(pk^*, sk^*) \xleftarrow{R} \mathsf{TCgen}(1^k)$ are generated. Just as in the case of $\mathrm{IDEAL}_{\hat{\mathcal{F}}_{\mathrm{ZK}}^R, \mathcal{S}, \mathcal{Z}}$, (pk^*, σ) is used as the common reference string for $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$, and sk^* and τ are stored. 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 sets the prover data field of 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 $\hat{\Pi}$. The output of each experiment (hybrid model, ideal process, and $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$) is the output of $\mathcal{Z}$.

Let $\mathrm{MIX}_{\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variable describing the outputs of $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$. First, we can show that $\mathrm{HYB}_{\hat{\Pi},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}} \stackrel{c}{\approx} \mathrm{MIX}_{\mathcal{A},\mathcal{Z}}$. In fact, it comes from the fact that the SSTC scheme is perfectly hiding and a straightforward hybrid reduction to the simulator $\mathcal{S}_{\Pi}$ of the Ω-protocol Π.[15]

Now we must show that $\mathrm{MIX}_{\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} \mathrm{IDEAL}_{\hat{\mathcal{F}}_{\mathrm{ZK}}^R, \mathcal{S}, \mathcal{Z}}$, which will finish the proof to our theorem. This will follow similar to the proof of Theorem 5.1, but also using the simulation-sound binding property of the trapdoor commitment scheme.

Let $p = \Pr[\mathrm{IDEAL}_{\hat{\mathcal{F}}_{\mathrm{ZK}}^R, \mathcal{S}, \mathcal{Z}}(k)]$ and $p' = \Pr[\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)]$. Similar to the proof of Theorem 5.1, the only difference between $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ and $\mathcal{S}$ comes from when the extractor in $\mathcal{S}$ outputs an incorrect witness for a session started by $\mathcal{A}$, and thus $\mathcal{Z}$ receives an output message (indicating a correct proof) in $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ but not when interacting with $\mathcal{S}$. (Note that the transcripts of corrupted prover/uncorrupted verifier sessions will never be the same as transcripts of uncorrupted prover/corrupted verifier sessions because of the auxiliary data aux.) Let $\vec{b}$ be the vector corresponding to simulated verifier or extractor sessions, with $b = 1$ corresponding to whether $\mathcal{Z}$ receives an output message. Let ρ be the statistical difference between the distribution of vectors $\vec{b}$ resulting from $\mathsf{Mix}_{\mathcal{A},\mathcal{Z}}(k)$ and vectors $\vec{b}$ resulting from $\mathcal{S}$. (Note that $\rho \geq |p - p''|$.) Let u be an upper bound on the number of verifier sessions. Then the average probability of a difference in a given bit position is at least ρ/u.

To complete the proof, we simply need to show that ρ is negligible. Let C be the number of times $\mathcal{Z}$ sends $\mathsf{zk\text{-}prover}$ messages to the parties. Now we construct an adversary $\mathcal{B}$ that breaks the SSTC scheme TC with probability $\frac{1}{2}((\rho/u)^2 - 2^{-k})$ and with at most $2C$ calls to the commitment revealing oracle. Therefore, it will follow that ρ is negligible.

We describe the adversary $\mathcal{B}$. Let $\mathcal{B}$ take a public key pk of TC along with a TC simulator. First $\mathcal{B}$ chooses a random $\ell \in \{1, \ldots, u\}$, and then it runs as $\mathcal{S}$, except for (1) changing the common reference string from $(*, \sigma)$ to (pk, σ), (2) using the TC simulator to fake commitments. Also, before sending a challenge (as the second message) in session ℓ w, $\mathcal{B}$ forks the experiment and continues independently in each sub-experiment (thus giving random independent challenges to $\mathcal{A}$). 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 know that $\mathcal{A}$ has decommitted differently in two sub-experiment. This is because of the property of the Ω-protocol; if $\mathcal{A}$ had decommitted in the same way, then there exist two accepting conversations with the same first-message, and then a witness should be extracted, indicating that $x \in L_R$. But now $\mathcal{B}$ has obtained two different decommitments, successfully breaking TC. By Lemma A.1, a successful sub-experiment occurs with probability at least $(\rho/u)^2 - 2^{-k}$, and thus either $\mathcal{B}$ will break the SSTC scheme TC with probability $\frac{1}{2}((\rho/u)^2 - 2^{-k})$, as claimed above. □

6 Efficient Instantiations

Here we briefly describe some efficient instantiations of our constructions. First, we discuss two efficient signature schemes (namely, the Cramer-Shoup signature scheme and the DSA signature scheme) and

[15] Note that if a corruption occurs between the first and second messages to the wrapper machine for the simulation, it will be just as if the simulation never received the second message.

two associated efficient Σ-protocols that can be plugged into our constructions of USSZK, NMZK, and UCZK protocols. Second, we construct an efficient SSTC scheme based on DSA that can be used in our construction of a UCZK protocol. Third, we give an example of an efficient Ω-protocol for the discrete logarithm relation, thus implying efficient NMZK and UCZK protocols for discrete logarithm. Finally, we describe a generalized definition of Ω-protocols, which can replace Ω-protocols in an appropriately generalized definition of NMZK protocols.[16] Then we present a very efficient[17] generalized Ω-protocol for proving knowledge of the plaintext of an ElGamal ciphertext, thus implying an efficient NMZK protocol for ElGamal plaintext knowledge.

6.1 Signature schemes

First we note that for our constructions we can use a more general version of the Σ-protocol for proving knowledge of signatures, as follows. Consider the binary relation $R_{vk} = \{(m,s)\}$ for a signature scheme SIG. We say a polynomial-time computable function f is a *partial knowledge function* of SIG, if there exists a probabilistic polynomial-time machine M such that every m and vk, $\{s_1 : s_1 \leftarrow M(m, vk)\}$ and $\{s_1 : s \leftarrow \mathsf{sig\_sign}(vk, m); s_1 \leftarrow f(m, vk, s)\}$ have the same distribution. Intuitively, a partial knowledge function carries part of the information about the signature, yet can be efficiently sampled without even knowing one. If a signature scheme SIG has a partial knowledge function f, then the relation $R'_{vk} = \{((m, s_1), s) : (m, s) \in R_{vk} \wedge s_1 = f(m, vk, s)\}$ can replace R_{vk} in the constructions for $\mathsf{USS}^R_{[vk]}$, $\mathsf{NM}^R_{[vk,\sigma']}(x)$, and $\mathsf{UC}^R_{[pk^*,vk,\sigma']}(x)$, with $\mathcal{P}$ sending a randomly sampled s_1 (partial knowledge) before running the Σ-protocol $\Sigma^R(x) \vee \Sigma^{R'_{vk}}(vk', s_1)$. We say R'_{vk} is a *partial signature relation* for SIG.

Here we show that the Cramer-Shoup signature scheme [16] and the DSA signature scheme [38] both admit efficient Σ-protocols for proving knowledge of signatures using this more general definition, and thus can be plugged into our constructions.

The Cramer-Shoup Signature Scheme Cramer and Shoup [16] presented an efficient signature scheme that is existentially unforgeable against adaptive chosen-message attacks under the Strong RSA Assumption, formally defined in Appendix B. In addition to the main security parameter k, they use a secondary security parameter k' for public key modulus size.[18] 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 $\mathsf{SIG_{CS}} = (\mathsf{sig\_gen_{CS}}, \mathsf{sig\_sign_{CS}}, \mathsf{sig\_verify_{CS}})$.[19]

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

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

[16] We note that this generalization is not applicable to UCZK protocols.
[17] In particular, this protocol is more efficient than the best (strict) Ω-protocol that we have found.
[18] For today's technology, reasonable values may be $k = 256$ and $k' = 1024$.
[19] 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.

- $\mathsf{sig\_verify}_{\mathsf{CS}}(vk, m, \langle e, y, y' \rangle)$:
  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 6.1 ([16]) *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.*

Note that from a public key vk, a message m and a signature $\langle e, y, y' \rangle$ on m, one can extract the pair (e, y'). Also note that for a randomly generated signature, this pair (e, y') is random, i.e., e is a random k-bit prime not equal to e', y' is a random element of QR_N,[20] and they are independent. Therefore, function $f(m, vk, \langle e, y, y' \rangle) = (e, y')$ is a partial knowledge function for Cramer-Shoup. Furthermore, given vk, m, and (e, y'), one can compute $x' \leftarrow (y')^{e'} \cdot h^{-H(m)} \bmod N$, and then y is simply a root of a known element, i.e., y is the e-th root of $x \cdot h^{H(x')} \bmod N$. Guillou and Quisquater [34] presented a Σ-protocol for proving knowledge of roots that has the special soundness property. Their protocol can be directly adopted here for proving the partial signature relation R'_{vk}.

DSA The Digital Signature Algorithm [38] was proposed by NIST in April 1991, and in May 1994 was adopted as a standard digital signature scheme in the U.S. [27]. 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.[21]

- $\mathsf{sig\_gen}_{\mathsf{DSA}}(1^k)$:
  $q \leftarrow \prime(1^k)$; $p \xleftarrow{R} \textsc{Prime}(1^{k'})$, 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 (sk, vk).

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

Note that from a public key vk, a message m and a signature $\langle r', s \rangle$, one can efficiently compute a value $r \leftarrow g^{H(m)s^{-1}} y^{r's^{-1}} \bmod p$. Also note that for a randomly generated signature, the value r is a random element generated by g. Therefore, $f(m, vk, \langle r', s \rangle) = r$ is a partial knowledge function for DSA. Furthermore, given vk, m, and r, s is simply a discrete log base r of the known element $g^{H(m)} y^{r'} \bmod p$. Schnorr [51] presents a Σ-protocol for proving knowledge of a discrete log, which satisfies the special soundness property. This protocol can be used to prove the partial signature relation R'_{vk}.

[20] We assume that e' is not a factor of $\phi(N)$, which is false with only negligible probability.

[21] 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 [26]. However, we will use these parameters as if they could be varied according to the security level desired.

6.2 SSTC scheme

Here we present an efficient SSTC scheme TC based on DSA. First, though, we describe a slightly simpler scheme TC' for *weak* simulation-sound trapdoor commitments, when id is always the empty string (and thus, in essence, no double reveal queries to the trapdoor commitment simulator are allowed). We can implement this simpler scheme over elements from a group $(G, +)$ by using a technique similar to that in Damgård and Nielsen [18] that involves two trapdoor commitment schemes TC_0 and TC_1 that commit to elements in G. The trapdoor in TC' is the trapdoor of one of TC_0 or TC_1 along with a bit indicating which. To commit to a message m, generate random $m_0 \in G$, set $m_1 \leftarrow m - m_0$, and commit to m_0 and m_1 using TC_0 and TC_1, respectively, i.e., generating commitment (c_0, c_1). To open a commitment (c_0, c_1), open each commitment, say to (m_0, m_1). Then $m = m_0 + m_1$ is the decommitted value. To open a commitment of (c_0, c_1), say of (m_0, m_1), to an arbitrary value m' using trapdoor (b, sk_b), i.e., trapdoor sk_b of TC_b, open commitment c_{1-b} normally, and use sk_b to open commitment c_b to $m' - m_{1-b}$. (A proof that this satisfies the weak simulation-sound binding property follows closely from Damgård and Nielsen [18].) This scheme does not satisfy the full notion of simulation-sound binding, since after revealing a commitment in two different ways (even one with an arbitrary id), the adversary can determine which trapdoor is used, and this would cause the proof from [18] to fail.[22]

Our scheme TC that satisfies simulation-sound binding uses the same technique as above of being built over two commitment schemes TC_0 and TC_1, but each of those will be built over DSA as follows. Given a DSA public key (g, p, q, y), a commitment to a message m using id is generated as follows. First compute $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_q$, $g' \leftarrow g^{\alpha} \bmod p$, and $h = g^{H(id)} y^{g'} \bmod p$. (Note that if s is the discrete log of h over g', then $(g' \bmod q, s)$ is the DSA signature for id.) Then use a Pedersen commitment [45] over bases (g', h) to commit to m, i.e., choose $\beta \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and compute commitment (g', c) where $c \leftarrow (g')^m h^{\beta}$. To open this commitment, output (m, β).

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 a adaptive chosen-message attack.) Given a DSA key vk_0 and signature oracle, we generate another DSA key pair (vk_1, sk_1), choose a bit b, and say (vk_b, vk_{1-b}) is the public key for our commitment scheme.

Now say we know which id the adversary is going to use in its commitment with double opening. To commit to a value v using id, we compute an actual signature for id using sk_1, and then commit to some value using that signature. Then we we use the knowledge of the signature to decommit to an arbitrary value m. To commit to a value v using $id' \neq id$, we choose a bit b' to decide which scheme to compute a signature (and thus which scheme will be used in fake decommitments). If $b' = 0$, we compute a signature using the DSA signature oracle on id', and if $b' = 1$ we compute a signature using sk_1.

Now the adversary's view is independent of b, and thus if the adversary gives a double opening with id, then with probability at least $\frac{1}{2}$, there will be different openings (m_0, β_0) and (m_0', β_0') of (g_0', c_0), so $(g_0' \bmod q, (\beta_0' - \beta_0)/(m_0 - m_0') \bmod q)$ is a signature on id, breaking DSA. Note that if we do not know which id will be used by the adversary, we would have to guess this, reducing the probability of breaking DSA by a polynomial factor.

6.3 An efficient Ω-protocol

We describe an efficient Ω-protocol for proving knowledge of a discrete logarithm. This protocol is based on the Decisional Composite Residuosity assumption and the Strong RSA assumption, formally

[22] One could use this scheme with weak simulation soundness in our UCZK adaptive protocol, but it would require the common reference string to contain one trapdoor commitment public key for each party.

defined in Appendix B.

Let (g, p, q) be public parameters, where q and p are primes with $q|(p-1)$, and $g \in \mathbb{Z}_p^*$ with $\mathsf{order}(g) = q$. Let R be the discrete logarithm relation: $R = \{(y, x) : y \equiv g^x \bmod p\}$. Our Ω-protocol for R is constructed as follows: The common reference string consists of two parts: (1) a Paillier public key $pk = \langle N, h\rangle$ where N is an RSA modulus and $h \in \mathbb{Z}_{N^2}^*$ with $N|\mathsf{order}(h)$, and (2) another RSA modulus with 2 generators $\langle \tilde{N}, h_1, h_2\rangle$. The prover and the verifier share a common input y, while the prover also knows x, such that $g^x = y$. In the first message, the prover sends an encryption of x using the Paillier encryption key pk. Then a Σ-protocol is used to prove that the plaintext in the Paillier encryption is indeed the discrete log of y. A technical difficulty is that the discrete logarithm and the Paillier encryption work in different moduli. To overcome this we use the known technique of adding a commitment to x using two generators (h_1, h_2) over a third modulus $\tilde{N}$ of unknown factorization [6, 7, 8, 28, 40]. The detailed construction is presented in Appendix C.

6.4 An efficient generalized Ω-protocol

For an NP relation $R = \{(x, w)\}$ and a polynomial-time computable function f, let $R_f = \{(x, f(w)) : (x, w) \in R\}$. (Note that R_f may not itself be an NP relation.) Then we define an f-extracting Ω-protocol for R as an Ω-protocol for R except that the extractor $\mathcal{E}_2$ outputs $f(w)$, instead of w. Similarly, we can define an f-extracting NMZK protocol in which the extractor $\mathcal{E}_2$ outputs $f(w)$, instead of w, and the extraction condition is changed appropriately.[23] It is easy to see that if we replace the Ω-protocol in our construction of NMZK protocols with an f-extracting Ω-protocol, our construction yields an f-extracting NMZK protocol. Note that the prover in both Ω-protocols and NMZK protocols still receives the "full" witness w. Also note that if f is the identity function, we have the normal definitions of an Ω-protocol and an NMZK protocol.

One application of these generalized definitions is in proving plaintext knowledge. See [36] for some applications of proof of plaintext knowledge. Consider a semantically secure encryption scheme. This scheme naturally induces a relation $R = \{(e, (x, r))\}$, where e is the encryption of plaintext x using random bits r. Then consider a function f defined as $x \leftarrow f(x, r)$. It is easy to see that an f-extracting Ω-protocol for R is essentially a proof of plaintext knowledge, so we will call this function f a *plaintext knowledge function.*

We now present a very efficient f-extracting Ω-protocol for ElGamal encryption, where f is a plaintext knowledge function. Let (g, p, q) be public parameters, where q and p are primes with $q|(p-1)$, and $g \in \mathbb{Z}_p^*$ with $\mathsf{order}(g) = q$. Then the ElGamal encryption scheme can be formally defined as follows, with the message space being the subgroup generated by g.

- $\mathsf{enc\_gen}_{\mathsf{EG}}(g, p, q)$:
  $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$; $y \leftarrow g^x \bmod p$; $sk \leftarrow x$; $pk \leftarrow y$;
  return (sk, pk).
- $\mathsf{encrypt}_{\mathsf{EG}}(vk, m)$:
  $r \stackrel{R}{\leftarrow} \mathbb{Z}_q$; $a \leftarrow g^r \bmod p$; $b \leftarrow my^r \bmod p$;
  return (a, b).
- $\mathsf{decrypt}_{\mathsf{EG}}(sk, (a, b))$:
  return b/a^x

The relation for the ElGamal system is $R = \{((a, b), (m, r)) : (a \equiv g^r \bmod p) \wedge (b \equiv my^r \bmod p)\}$, and f is defined such that $m \leftarrow f(m, r)$. The f-restricted Ω-protocol is constructed as follows. The

[23]Note that the resulting NMZK protocols can not necessarily be used to construct UCZK protocols (even with static corruptions), since UCZK protocols are, by definition, proofs of knowledge.

common reference string is a new public key y' for the ElGamal system, which is generated by running $(x', y') \leftarrow \mathsf{enc\_gen}_{\mathsf{EG}}(g, p, q)$ using fresh random bits. The corresponding decryption key x' is discarded. The prover takes $(a, b) = (g^r, my^r)$, which is an encryption of a message m (using random bits r), and then constructs a new encryption using the encryption key in the common reference string $(a', b') \leftarrow (g^{r'}, m(y')^{r'})$, where $r' \stackrel{R}{\leftarrow} \mathbb{Z}_q$. The prover then sends (a', b') to the verifier, and performs a Σ-protocol proving that the two ElGamal encryptions have the same plaintext. The Σ-protocol proceeds as follows. The prover picks $w, w' \leftarrow \mathbb{Z}_q$, computes $d \leftarrow g^w$, $d' \leftarrow g^{w'}$, and $e \leftarrow y^w/(y')^{w'}$, and outputs (d, d', e) as the first message. On challenge c, the prover computes $s \leftarrow rc + w \bmod q$ and $s' \leftarrow r'c + w' \bmod q$, and outputs (s, s') as the third message. Finally the verifier verifies that $g^s = a^c d$, $g^{s'} = (a')^c d'$, and $y^s/(y')^{s'} = (b/b')^c e$.

SHVZK is satisfied since given input $(a, b) \in L_R$ and a challenge c, a simulator can generate an encryption (a', b') of an arbitrary value, and then use the perfect SHVZK property of the Σ-protocol to generate an accepting conversation. By the semantic security of ElGamal, the simulator is still computationally indistinguishable from that of an actual prover. Now we show the f-extraction property is satisfied. Let $\mathcal{E}_1(1^k)$ generate a fresh ElGamal key pair $(sk', vk') \leftarrow \mathsf{enc\_gen}_{\mathsf{EG}}(g, p, q)$, putting vk' in the common reference string, and passing the decryption key sk' to $\mathcal{E}_2$, which then interacts with prover and obtains an accepting transcript tr. Finally $\mathcal{E}_2$ outputs $m' \leftarrow \mathsf{decrypt}_{\mathsf{EG}}(sk', (a', b'))$ where (a', b') is the encryption in the transcript tr. By the weak soundness property of the Σ-protocol, the probability that m' is not the plaintext in the encryption (a, b) is at most 2^{-k} (assuming k-bit challenges).

References

[1] B. Barak. How to Go Beyond the Black-box Simulation Barrier. In *42nd IEEE Symp. on Foundations of Computer Sci.*, pp. 106–115, 2001.

[2] B. Barak. Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model. In *43rd IEEE Symp. on Foundations of Computer Sci.*, pp. 345–355, 2002

[3] B. Barak and Y. Lindell. Strict Polynomial-time in Simulation and Extraction. In *34th ACM Symp. on Theory of Computing*, pp. 484–493, 2002.

[4] 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.

[5] D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symp.* (LNCS 1423), pp. 48–63, 1998.

[6] F. Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In *Advances in Cryptology – EUROCRYPT 2000* (LNCS 1807), pp. 431–444, 2000.

[7] F. Boudot and J. Traoré. Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery. In *Information and Communication Security, Second International Conference, ICICS'99*, pp. 87–102.

[8] 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.

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

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

[11] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In *33rd ACM Symp. on Theory of Computing*, pp. 570–579, 2001.

[12] 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.

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

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

[15] 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.

[16] 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.

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

[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] 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.

[21] C. Dwork, M. Naor and A. Sahai. Concurrent zero-knowledge. In *30th ACM Symp. on Theory of Computing*, pp. 409–418, 1998.

[22] C. Dwork and A. Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In *Advances in Cryptology – CRYPTO '98* (LNCS 1462), pp. 442–457, 1998.

[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] 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.

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

[28] 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.

[29] 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.

[30] O. Goldreich, S. Micali and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *19th ACM Symp. on Theory of Computing*, pp. 218–229, 1987.

[31] 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.

[32] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, February 1989.

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

[34] L. C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing Both Transmission and Memory. In *Advances in Cryptology – EUROCRYPT '88* (LNCS 330), pp. 123–128, 1988

[35] 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.

[36] J. Katz. Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications. In *ePrint Archive*, Report 2002/027, `http://eprint.iacr.org/`, 2002.

[37] J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In *33rd ACM Symp. on Theory of Computing*, pp. 560–569, 2001.

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

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

[40] P. MacKenzie and M. Reiter. Two-Party Generation of DSA Signatures. In *Advances in Cryptology – CRYPTO 2001* (LNCS 2139), pp 137–154, 2001.

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

[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] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology – EUROCRYPT '98* (LNCS 1403), pp. 380–318, 1998.

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

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

[46] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[47] M. Prabhakaran, A. Rosen and A. Sahai. Concurrent zero knowledge with logarithmic round-complexity, In *ePrint Archive, Report 2002/055*, `http://eprint.iacr.org/`, 2002. Also in *43rd IEEE Symp. on Foundations of Computer Sci.*, pp. 366–375, 2002.

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

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

[50] 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.

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

A The Exclusive Collision Lemma

We prove the lemma used in the proof of Theorem 3.1.

Lemma A.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: We define a new predicate $\phi''(a, b_1, b_2) = \phi(a, b_1) \wedge \phi(a, b_2)$, which is essentially predicate ϕ' without the requirement that $\neg\mathsf{Coll}_a(b_1, b_2)$. For every a in the support of A, let $p_a = \mathsf{Prob}[\phi(a, B_a)]$. Let p_A be the function of random variable A taking value p_a when $A = a$. Then we have $p = \mathsf{Prob}[\phi(A, B_A)] = E[p_A]$ and $\mathsf{Prob}[\phi''(A, B_A^1, B_A^2)] = E[(p_A)^2] \geq (E[p_A])^2 = p^2$.

Finally we have

$$\mathsf{Prob}[\phi'(A, B_A^1, B_A^2)] \geq \mathsf{Prob}[\phi''(A, B_A^1, B_A^2)] - \mathsf{Prob}[\mathsf{Coll}_A(B_A^1, B_A^2)] \geq p^2 - q\ .$$

$\square$

We remark that, using a tighter analysis, the lower bound on $\mathsf{Prob}[\phi'(A, B_A^1, B_A^2)]$ in Lemma A.1 can be improved to $p^2 - pq$.

B 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 B.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 [4], and has been used in several applications (see [28, 29, 16]). It is a stronger assumption than the "standard" RSA assumption, yet no method is known for breaking it other than factoring N.

The Paillier cryptosystem and the Decision Composite Residuosity assumption. The Paillier encryption scheme [44] is defined as follows, where $\lambda(N)$ is the Carmichael function of N, and L is a function that takes input elements from the set $\{u < N^2 | u \equiv 1 \bmod N\}$ and returns $L(u) = \frac{u-1}{N}$. This definition differs from that in [44] only in that we define the message space for

$$
\begin{array}{rcl}
\textsf{prover} & & \textsf{verifier} \\
\alpha, \beta \xleftarrow{R} \mathbb{Z}_N^* & & \\
r \xleftarrow{R} \mathbb{Z}_{q^3} & & \\
a \xleftarrow{R} \mathbb{Z}_{q\tilde{N}} & & \\
b \xleftarrow{R} \mathbb{Z}_{q^3\tilde{N}} & & \\
y' \leftarrow g^r \bmod p & & \\
e \leftarrow h^x \alpha^N \bmod N^2 & & \\
e' \leftarrow h^r \beta^N \bmod N^2 & & \\
s \leftarrow (h_1)^x (h_2)^a \bmod \tilde{N} & & \\
s' \leftarrow (h_1)^r (h_2)^b \bmod \tilde{N} & & \\
& \xrightarrow{y', e, e', s, s'} & \\
& & c \xleftarrow{R} \mathbb{Z}_q \\
& \xleftarrow{c} & \\
z_1 \leftarrow cx + r & & \\
z_2 \leftarrow \alpha^c \beta \bmod N & & \\
z_3 \leftarrow ca + b & & \\
& \xrightarrow{z_1, z_2, z_3} & \\
& & z_1 \overset{?}{\in} \mathbb{Z}_{q^3} \\
& & y^c y' \overset{?}{\equiv} g^{z_1} \bmod p \\
& & e^c e' \overset{?}{\equiv} h^{z_1} (z_2)^N \bmod N^2 \\
& & s^c s' \overset{?}{\equiv} (h_1)^{z_1} h_2^{z_2} \bmod \tilde{N}
\end{array}
$$

Figure 6: Ω-protocol for the discrete log relation $\{(y, x) : y \equiv g^x \bmod p\}$. Common reference string is a Paillier public key and a Strong RSA modulus along with two generators $((N, h), (\tilde{N}, h_1, h_2))$.

public key $pk = \langle N, g \rangle$ as $[-(N-1)/2, (N-1)/2]$ (versus $\mathbb{Z}_N$ in [44]), and we restrict h to be $1 + N$. The security of this cryptosystem relies on the *Decision Composite Residuosity Assumption*, DCRA.

For key generation, choose random $k/2$-bit primes p, q, set $N = pq$, and set $h \leftarrow 1 + N$. The public key is $\langle N, h \rangle$ and the private key is $\langle N, h, \lambda(N) \rangle$. To encrypt a message m with public key $\langle N, h \rangle$, select a random $\alpha \in \mathbb{Z}_N^*$ and compute $c \leftarrow g^m \alpha^N \bmod N^2$. To decrypt a ciphertext c with secret key $\langle N, h, \lambda(N) \rangle$, compute $m = \frac{L(c^{\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \bmod N$, and the decryption is m if $m \le (N-1)/2$, and otherwise the decryption is $m - N$. Paillier [44] shows that both $c^{\lambda(N)} \bmod N^2$ and $g^{\lambda(N)} \bmod N^2$ are elements of the form $(1 + N)^d \equiv_{N^2} 1 + dN$, and thus the L function can be easily computed for decryption.

C An Efficient Ω-protocol for Proving Knowledge of Discrete Log

The detailed construction of the Ω-protocol for proving knowledge of discrete logarithm is given in Figure 6.