Computational Soundness of Symbolic Zero-Knowledge Proofs: Weaker Assumptions and Mechanized Verification

Michael Backes[1,2], Fabian Bendun[1], and Dominique Unruh[3]

[1] Saarland University, Saarbrücken, Germany
{backes,bendun}@cs.uni-saarland.de
[2] MPI-SWS, Saarbrücken, Germany
[3] University of Tartu, Tartu, Estonia
unruh@ut.ee

Abstract. The abstraction of cryptographic operations by term algebras, called symbolic models, is essential in almost all tool-supported methods for analyzing security protocols. Significant progress was made in proving that symbolic models offering basic cryptographic operations such as encryption and digital signatures can be sound with respect to actual cryptographic realizations and security definitions. Even abstractions of sophisticated modern cryptographic primitives such as zero-knowledge (ZK) proofs were shown to have a computationally sound cryptographic realization, but only in ad-hoc formalisms and at the cost of placing strong assumptions on the underlying cryptography, which leaves only highly inefficient realizations.

In this paper, we make two contributions to this problem space. First, we identify weaker cryptographic assumptions that we show to be sufficient for computational soundness of symbolic ZK proofs. These weaker assumptions are fulfilled by existing efficient ZK schemes as well as generic ZK constructions. Second, we conduct all computational soundness proofs in CoSP, a recent framework that allows for casting computational soundness proofs in a modular manner, independent of the underlying symbolic calculi. Moreover, all computational soundness proofs conducted in CoSP automatically come with mechanized proof support through an embedding of the applied π-calculus.

1 Introduction

Proofs of security protocols are known to be error-prone and, owing to the distributed-system aspects of multiple interleaved protocol runs, awkward for humans to make. Hence work towards the automation of such proofs started soon after the first protocols were developed. From the start, the actual cryptographic operations in such proofs were idealized into so-called symbolic models, following [18, 19, 29], e.g., see [23, 33, 1, 26, 11]. This idealization simplifies proof construction by freeing proofs from cryptographic details such as computational restrictions, probabilistic behavior, and error probabilities. It was not at all clear

D. Basin and J.C. Mitchell (Eds.): POST 2013, LNCS 7796, pp. 206–225, 2013.

whether symbolic models are a sound abstraction from real cryptography with its computational security definitions. Existing work has largely bridged this gap for symbolic models offering the core cryptographic operations such as encryption and digital signatures, e.g., see [2, 9, 24, 30, 16, 14].

While symbolic models traditionally comprised only basic cryptographic operations, recent work has started to extend them to more sophisticated primitives with unique security features that go far beyond the traditional goal of cryptography to solely offer secrecy and authenticity of communication. Zero-knowledge (ZK) proofs[1] constitute arguably the most prominent such primitive. This primitive's unique security features, combined with the recent advent of efficient cryptographic implementations of this primitive for special classes of problems, have paved the way for its deployment in modern applications. For instance, ZK proofs can guarantee authentication yet preserve the anonymity of protocol participants, as in the Civitas electronic voting protocol [15] or the Pseudo Trust protocol [27], or they can prove the reception of a certificate from a trusted server without revealing the actual content, as in the Direct Anonymous Attestation (DAA) protocol [13]. More recently, ZK proofs have been used to develop novel schemes for anonymous webs of trust [5] as well as privacy-aware proof-carrying authorization [28].

A symbolic abstraction of (non-interactive) ZK proofs has been put forward in [7]. The proposed abstraction is suitable for mechanized proofs [7, 4] and was already successfully used to produce the first fully mechanized proof of central properties of the DAA protocol. A computational soundness result for such symbolic ZK proofs has recently been achieved as well [10]. However, this work imposes strong assumptions on the underlying cryptographic implementation of zero-knowledge proofs: Among other properties, the zero-knowledge proof is required to satisfy the notion of extraction zero-knowledge; so far, only one (inefficient) scheme is known that fulfills this notion [22]. Thus the vast number of recently proposed, far more efficient zero-knowledge schemes, and particularly those schemes that stem from generic ZK constructions, are not comprised by this result. Hence they do not serve as sound instantiations of symbolic zero-knowledge proofs, leaving all actually deployed ZK protocols without any computational soundness guarantee. In addition, the result in [10] casts symbolic ZK proofs within an ad-hoc formalism that is not accessible to existing formal proof tools.

1.1 Our Contribution

In this paper, we make the following two contributions to this problem space:

- First, we identify weaker cryptographic assumptions that we show to be sufficient for obtaining a computational soundness result for symbolic ZK proofs.

[1] A zero-knowledge proof [20] consists of a message or a sequence of messages that combines two seemingly contradictory properties: First, it constitutes a proof of a statement x (e.g, $x =$ "the message within this ciphertext begins with 0") that cannot be forged, i.e., it is impossible, or at least computationally infeasible, to produce a zero-knowledge proof of a wrong statement. Second, a zero-knowledge proof does not reveal any information besides the bare fact that x constitutes a valid statement.

Essentially, we show that the strong notion of extraction zero-knowledge required in [10] can be replaced by the weaker notion of simulation-sound extractability. In contrast to extraction zero-knowledge, simulation-sound extractability constitutes an established property that many existing cryptographic constructions satisfy. In particular, there exist generic constructions for transforming any non-interactive ZK proof into a ZK proof that satisfies simulation-sound extractability (and the remaining properties that we impose for computational soundness) [31], as well as several efficient schemes that are known to satisfy simulation-sound extractability (and the remaining properties), e.g., [25, 21, 32]. Thus requiring simulation-sound extractability instead of extraction zero-knowledge greatly extends the pool of cryptographic constructions for ZK proofs that constitute sound implementations, and it for the first time enables the computationally sound deployment of efficient ZK realizations.

- Second, we conduct all computational soundness proofs in CoSP [3], a recent framework that allows for casting computational soundness proofs in a conceptually modular and generic way: proving x cryptographic primitives sound for y calculi only requires $x + y$ proofs (instead of $x \cdot y$ proofs without this framework), and the process of embedding calculi is conceptually decoupled from computational soundness proofs of cryptographic primitives. In particular, computational soundness proofs conducted in CoSP are automatically valid for the applied π-calculus, and hence accessible to existing mechanized verification techniques.

  The conduction in CoSP has the drawback that the computational soundness is shown for trace properties. However, trace properties are sufficient to verify weak anonymity. Consequently, we can show central properties of the DAA protocol.

1.2 Outline of the Paper

First, we introduce our symbolic abstraction of (non-interactive) ZK proofs within CoSP in Section 2. Section 3 contains the weaker cryptographic assumptions that we show to be sufficient for achieving computational soundness of ZK proofs. Our main theorem is presented in Section 4 for which we give a proof overview in Section 5. Section 6 concludes and outlines future work.

2 Symbolic Model for Zero-Knowledge

In this section, we describe our symbolic abstraction of zero-knowledge proofs.

Terms and Constructors. We model nonces, probabilistic public-key encryption and signatures, pairs, strings, and zero-knowledge proofs. Except for the latter, our modeling closely follows that of [3]. The following grammar describes the set $\mathbf{T}$ of all terms that may occur in the symbolic model:

$$\begin{aligned}\mathbf{T} ::= &\mathrm{enc}(\mathrm{ek}(N), t, N) \mid \mathrm{ek}(N) \mid \mathrm{dk}(N) \mid \mathrm{sig}(\mathrm{sk}(N), t, N) \mid \mathrm{vk}(N) \mid \mathrm{sk}(N) \mid \\ &\mathrm{crs}(N) \mid \mathrm{ZK}(\mathrm{crs}(N), t, t, N) \mid \mathrm{pair}(t, t) \mid S \mid N \mid \\ &\mathrm{garbage}(N) \mid \mathrm{garbageEnc}(t, N) \mid \mathrm{garbageSig}(t, N) \mid \mathrm{garbageZK}(t, t, N) \\ S ::= &\ \mathrm{empty} \mid \mathrm{string}_0(S) \mid \mathrm{string}_1(S)\end{aligned}$$

Here N represents nonces and ranges over $\mathbf{N}_P \cup \mathbf{N}_E$, two disjoint infinite sets of nonces, the protocol nonces and adversary nonces, respectively. $\mathrm{ek}(N), \mathrm{dk}(N)$, $\mathrm{vk}(N), \mathrm{sk}(N)$ represent encryption, decryption, verification, and signing keys. $\mathrm{enc}(\mathrm{ek}(N_1), t, N_2)$ represents an encryption under public key $\mathrm{ek}(N_1)$ of plaintext t using algorithmic randomness N_2. (Symbolically, the algorithmic randomness just allows to distinguish different encryptions of the same plaintext; computationally, it will actually be the randomness used by the encryption algorithm.) $\mathrm{sig}(\mathrm{sk}(N_1), t, N_2)$ is a signature of t under signing key $\mathrm{sk}(N_1)$ with algorithmic randomness N_2. Bitstrings can be expressed using terms matching the nonterminal S. $\mathrm{garbage}(N)$ represents invalid terms, $\mathrm{garbageEnc}(t, N)$ and $\mathrm{garbageSig}(t, N)$ represent invalid encryptions and signatures (but which at a first glance seem to be valid encryptions/signatures with public key t).

Zero-Knowledge Proofs. The interesting part are the zero-knowledge proofs. To understand the meaning of a term $\mathrm{ZK}(\mathrm{crs}(N), x, w, M)$, we first need to introduce the relation $R^{\mathrm{sym}}_{\mathrm{adv}}$. This relation is part of the symbolic modeling, but all our results are parametric in $R^{\mathrm{sym}}_{\mathrm{adv}}$. (I.e., our result holds for any choice of $R^{\mathrm{sym}}_{\mathrm{adv}}$, as long as $R^{\mathrm{sym}}_{\mathrm{adv}}$ satisfies certain constraints.) $R^{\mathrm{sym}}_{\mathrm{adv}}$ specifies what a valid witness for a particular statement would be. For example, if we wish to show that we know a decryption key w that decrypts a given ciphertext x, then we define $R^{\mathrm{sym}}_{\mathrm{adv}} := \{(x, w) : \exists N, M, t. x = \mathrm{enc}(\mathrm{ek}(N), t, M), w = \mathrm{dk}(N)\}$.[2] The term $\mathrm{ZK}(\mathrm{crs}(N), x, w, M)$ then represents a zero-knowledge proof constructed with respect to a common reference string $\mathrm{crs}(N)$ with statement x and witness w and using algorithmic randomness M. A valid proof satisfies $(x, w) \in R^{\mathrm{sym}}_{\mathrm{adv}}$. Note that our symbolic model does not ensure that any term $\mathrm{ZK}(\mathrm{crs}(N), x, w, M)$ is a valid proof. Instead, we provide a destructor $\mathrm{verify}_{\mathrm{ZK}}$ below that allows to check the validity. As we will see below, the statement x can be extracted from a proof, but the witness w is hidden.

Destructors. Protocol operations on terms are described by a set of destructors. These are partial functions from $\mathbf{T}^n$ to $\mathbf{T}$ (where n depends on the destructor). The destructors are specified in Figure 1. Note that there are a number of destructors that do not modify their input ($\mathrm{isek}, \mathrm{iszk}, \mathrm{equals}, \ldots$). These are useful for testing properties of terms: The protocol can, e.g., compute $\mathrm{isek}(t)$ and then branch depending on whether the destructor succeeds. We only describe the destructors related to ZK proofs. $\mathrm{getPub}(t)$ returns the statement x proven by a ZK proof t. getPub does not check whether the proof is actually valid; for this, we have $\mathrm{verify}_{\mathrm{ZK}}(t_1, t_2)$ which checks whether t_2 is a valid proof with respect to

[2] Notice that it is no restriction that we use the same relation for all ZK-proofs: To encode relations $R_1, \ldots, R_n$, we define a relation $R := \{((a, x), w) : \exists i. a = a_i \wedge (x, w) \in R_i\}$ where a_i are distinct terms.

$$\begin{aligned}
\mathrm{dec}(\mathrm{dk}(t_1), \mathrm{enc}(\mathrm{ek}(t_1), m, t_2)) &= m \\
\mathrm{verify}_{\mathrm{sig}}(\mathrm{vk}(t_1), \mathrm{sig}(\mathrm{sk}(t_1), t_2, t_3)) &= t_2 \\
\mathrm{isek}(\mathrm{ek}(t)) &= \mathrm{ek}(t) \\
\mathrm{isvk}(\mathrm{vk}(t)) &= \mathrm{vk}(t) \\
\mathrm{isenc}(\mathrm{enc}(\mathrm{ek}(t_1), t_2, t_3)) &= \mathrm{enc}(\mathrm{ek}(t_1), t_2, t_3) \\
\mathrm{isenc}(\mathrm{garbageEnc}(t_1, t_2)) &= \mathrm{garbageEnc}(t_1, t_2) \\
\mathrm{issig}(\mathrm{sig}(\mathrm{sk}(t_1), t_2, t_3)) &= \mathrm{sig}(\mathrm{sk}(t_1), t_2, t_3) \\
\mathrm{issig}(\mathrm{garbageSig}(t_1, t_2)) &= \mathrm{garbageSig}(t_1, t_2) \\
\mathrm{iscrs}(\mathrm{crs}(t_1)) &= \mathrm{crs}(t_1) \\
\mathrm{iszk}(\mathrm{ZK}(t_1, t_2, t_3, t_4)) &= \mathrm{ZK}(t_1, t_2, t_3, t_4) \\
\mathrm{iszk}(\mathrm{garbageZK}(t_1, t_2, t_3)) & \\
= \mathrm{garbageZK}(t_1, t_2, t_3) & \\
\mathrm{ekof}(\mathrm{enc}(\mathrm{ek}(t_1), t_2, t_3)) &= \mathrm{ek}(t_1) \\
\mathrm{ekof}(\mathrm{garbageEnc}(t_1, t_2)) &= t_1
\end{aligned}$$

$$\begin{aligned}
\mathrm{crsof}(\mathrm{ZK}(\mathrm{crs}(t_1), t_2, t_3, t_4)) &= \mathrm{crs}(t_1) \\
\mathrm{crsof}(\mathrm{garbageZK}(t_1, t_2, t_3)) &= t_1 \\
\mathrm{vkof}(\mathrm{sig}(\mathrm{sk}(t_1), t_2, t_3)) &= \mathrm{vk}(t_1) \\
\mathrm{vkof}(\mathrm{garbageSig}(t_1, t_2)) &= t_1 \\
\mathrm{fst}(\mathrm{pair}(t_1, t_2))) &= t_1 \\
\mathrm{snd}(\mathrm{pair}(t_1, t_2)) &= t_2 \\
\mathrm{unstring}_0(\mathrm{string}_0(s)) &= s \\
\mathrm{unstring}_1(\mathrm{string}_1(s)) &= s \\
\mathrm{getPub}(\mathrm{ZK}(t_1, t_2, t_3, t_4)) &= t_2 \\
\mathrm{getPub}(\mathrm{garbageZK}(t_1, t_2, t_3)) &= t_2 \\
\mathrm{equals}(x, x) &= x \\
\mathrm{verify}_{\mathrm{ZK}}(\mathrm{crs}(t_1), \mathrm{ZK}(\mathrm{crs}(t_1), t_2, t_3, t_4)) & \\
= \mathrm{ZK}(\mathrm{crs}(t_1), t_2, t_3, t_4) \text{ if } (t_2, t_3) \in R^{\mathrm{sym}}_{\mathrm{adv}} &
\end{aligned}$$

Fig. 1. Definition of destructors. If no rule matches, a destructor returns $\bot$.

the CRS t_1. If so, t_2 is returned (and can, e.g., be fed into getPub); otherwise $\bot$ is returned. iscrs(t) and iszk(t) allow us to test if t is a CRS or a (possibly invalid) zero-knowledge proof.

Protocols. We use the protocol model from the CoSP framework [3]. There, a protocol is modeled as a (possibly infinite) tree of nodes. Each node corresponds to a particular protocol action such as receiving a term from the adversary, sending a previously computed term to the adversary, applying a constructor or destructor to previously computed terms (and branching depending on whether the application is successful), or picking a nonce. We do not describe the protocol model in detail here, but it suffices to know that a protocol can freely apply constructors and destructors (computation nodes), branch depending on destructor success, and communicate with the adversary. Despite the simplicity of the model, it is powerful enough to embed powerful calculi such as the applied π-calculus (shown in [3]) or RCF, a core calculus for F# (shown in [8]). (In the Appendix C, we present our computational soundness result in the applied π-calculus.)

Protocol Conditions. The protocols we consider are subject to a number of conditions, listed in Figure 2. The most interesting protocol condition is *valid proofs condition*: During the symbolic execution of the protocol, whenever the protocol constructs a ZK proof $\mathrm{ZK}(c, x, w, N)$ we have $(x, w) \in R^{\mathrm{sym}}_{\mathrm{honest}}$. Here $R^{\mathrm{sym}}_{\mathrm{honest}}$ is some fixed but arbitrary relation with $R^{\mathrm{sym}}_{\mathrm{honest}} \subseteq R^{\mathrm{sym}}_{\mathrm{adv}}$ (like in $R^{\mathrm{sym}}_{\mathrm{adv}}$, our results are parametric in $R^{\mathrm{sym}}_{\mathrm{honest}}$). In the simplest case, we would have $R^{\mathrm{sym}}_{\mathrm{honest}} := R^{\mathrm{sym}}_{\mathrm{adv}}$. Then the valid proofs condition simply requires that the protocol never tries to construct a ZK-proof with an invalid witness. (We only impose this condition on the honest protocol, not on the adversary.) In some cases, however, it may be advantageous to let $R^{\mathrm{sym}}_{\mathrm{honest}}$ be strictly smaller than $R^{\mathrm{sym}}_{\mathrm{adv}}$. This permits us to model a certain asymmetry in guarantees given by a zero-knowledge

proof system: To honestly generate a valid proof, we need a witness with $(x, w) \in R^{\text{sym}}_{\text{honest}}$, but given a malicious prover, we only have the guarantee that the prover knows a witness with $(x, w) \in R^{\text{sym}}_{\text{adv}}$. We call $R^{\text{sym}}_{\text{honest}}$ the *usage restriction*.

The Adversary. The capabilities of the adversary are described by a deduction relation $\vdash$. $S \vdash t$ means that from the terms S, the adversary can deduce t. $\vdash$ is defined by the following rules:

$$\frac{m \in S}{S \vdash m} \qquad \frac{N \in \mathbf{N}_E}{S \vdash N} \qquad \frac{F \text{ constructor or destructor} \qquad S \vdash t_1, \ldots, t_n \qquad t_1, \ldots, t_n \in \mathbf{T} \qquad F(t_1, \ldots, t_n) \in \mathbf{T}}{S \vdash F(t_1, \ldots, t_n)}$$

Note that the adversary cannot deduce protocol nonces. These are secret until explicitly revealed. The capabilities of the adversaries with respect to the network (intercept/modify messages) are modeled explicitly by the protocol: if the adversary is allowed to intercept a message, the protocol explicitly communicates it through the adversary.

Protocol Execution. Given a particular protocol Π (modeled as a tree), the set of possible protocol traces is defined by traversing the tree: in case of an input node the adversary nondeterministically picks a term t with $S \vdash t$ where S are the terms sent so far through output nodes; at computation nodes, a new term is computed by applying a constructor or destructor to terms computed/received at earlier nodes; then the left or right successor is taken depending on whether the destructor succeeded. The sequence of nodes we traverse in this fashion is called a *symbolic node trace* of the protocol. By specifying sets of node traces, we can specify trace properties for a given protocol. We refer to [3] for details on the protocol model and its semantics.

3 Computational Implementation

We now describe how to implement the constructors and destructors from the preceding section computationally. Following [3], we do so by specifying a partial deterministic function $A_F : (\{0,1\}^*)^n \to \{0,1\}^*$ (the *computational implementation of* F) for each constructor or destructor $F : \mathbf{T}^n \to \mathbf{T}$. Intuitively, A_F should behave as F, only on bitstrings, e.g., $A_{\text{enc}}(ek, m, r)$ should encrypt m using encryption key ek and algorithmic randomness r. The distribution A_N specifies the distribution according to which nonces are picked. In Appendix A.1 we give the full list of implementation conditions that the computational implementation must fulfill. These are mostly simple syntactic conditions (such as $A_{\text{fst}}(A_{\text{pair}}(x, y)) = x$). Furthermore, we require that A_{enc} and A_{sig} correspond to an IND-CCA secure encryption scheme and a strongly unforgeable signature scheme. These conditions are essentially the same as in [3]. Here, we will only discuss the cryptographic properties the implementation of ZK proofs should satisfy.

Properties of ZK Proofs. In [10], it was shown that for getting computational soundness of (non-interactive) zero-knowledge proofs, we need at least the following properties:[3] *Completeness* (if prover and verifier are honest, the proof is

[3] It was not shown that these are the minimal properties, but it was shown that none of these properties can be dropped without suitable substitute.

accepted), *extractability* (given a suitable trapdoor, one can get a witness out of a valid proof – this models the fact that the prover knows the witness), *zero-knowledge* (given a suitable trapdoor and a true statement x, a ZK-simulator can produce proofs without knowing a witness that are indistinguishable from normally generated proofs for x), *unpredictability* (two proofs are equal only with negligible probability), *length-regularity* (the length of a proof only depends on the length of statement and witness), and some variant of *non-malleability* (see below). Furthermore, they required for convenience that the verification and the extraction algorithm are deterministic.

The interesting property is non-malleability: Intuitively, non-malleability means that given a proof for some statement x, it is not possible to derive a proof for some other statement x', even if x logically entails x'. (For example, given a proof that the ciphertext c contains a plaintext $i < 5$ it should not be possible to construct a proof that c contains $i < 6$.) There are several variants of non-malleability; [10] used the notion of *extraction zero-knowledge* which is a strong variant of extractability (we are aware of only one scheme in the literature that has this property [22]). They left it as an open problem whether weaker variants also lead to computational soundness. We answer this question positively. We use the weaker and more popular notion of *simulation-sound extractability*. In a nutshell, this notion guarantees that the adversary cannot produce proofs from which no witness can be extracted, even when given access to a ZK-simulator.

We actually need an even weaker property: *honest simulation-sound extractability*. Here the adversary may ask the ZK-simulator to produce a simulated proof for x if he knows a witness w for x.

In the symbolic model, we have distinguished two relations $R^{\mathrm{sym}}_{\mathrm{adv}}$ and $R^{\mathrm{sym}}_{\mathrm{honest}}$, the first modeling what the adversary is able to do, the second modeling what honest participants are allowed to do. Similarly, our definition of *weakly symbolically-sound zero-knowledge proof* distinguishes two relations $R^{\mathrm{comp}}_{\mathrm{adv}} \supseteq R^{\mathrm{comp}}_{\mathrm{honest}}$. All conditions assume that honest participants use $(x,w) \in R^{\mathrm{comp}}_{\mathrm{honest}}$. ("Weakly" distinguishes our notion from that in [10] which requires extraction ZK.)

Definition 1 (Weakly Symbolically-Sound ZK Proofs). *A weakly symbolically-sound zero-knowledge proof system for relations $R^{\mathrm{comp}}_{\mathrm{honest}}, R^{\mathrm{comp}}_{\mathrm{adv}}$ is a tuple of polynomial-time algorithms $(\mathbf{K}, \mathbf{P}, \mathbf{V})$ such that there exist polynomial-time algorithms $(\mathbf{E}, \mathbf{S})$ and the following properties hold:*

- Completeness: *Let a polynomial-time adversary $\mathcal{A}$ be given. Let* $(\mathrm{crs}, \mathrm{simtd}, \mathrm{extd}) \leftarrow \mathbf{K}(1^\eta)$. *Let* $(x,w) \leftarrow \mathcal{A}(1^\eta, \mathrm{crs})$. *Let* $\mathrm{proof} \leftarrow \mathbf{P}(x,w,\mathrm{crs})$. *Then with overwhelming probability in η, it holds $(x,w) \notin R^{\mathrm{comp}}_{\mathrm{adv}}$ or $\mathbf{V}(x,\mathrm{proof},\mathrm{crs}) = 1$.*
- Zero-Knowledge: *Fix a polynomial-time oracle adversary $\mathcal{A}$. For given* $\mathrm{crs}, \mathrm{simtd}$, *let $\mathcal{O}_{\mathbf{P}}(x,w) := \mathbf{P}(x,w,\mathrm{crs})$ if $(x,w) \in R^{\mathrm{comp}}_{\mathrm{honest}}$ and $\mathcal{O}_{\mathbf{P}}(x,w) := \perp$ otherwise, and let $\mathcal{O}_{\mathbf{S}}(x,w) := \mathbf{S}(x,\mathrm{crs},\mathrm{simtd})$ if $(x,w) \in R^{\mathrm{comp}}_{\mathrm{honest}}$ and $\mathcal{O}_{\mathbf{S}}(x,w) := \perp$ otherwise. Then*

  $$|\Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{P}}}(1^\eta,\mathrm{crs}) = 1 : (\mathrm{crs},\dots) \leftarrow \mathbf{K}(1^\eta)] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{S}}}(1^\eta,\mathrm{crs}) = 1 : (\mathrm{crs},\dots) \leftarrow \mathbf{K}(1^\eta)]|$$

  is negligible in η.

- Honest simulation-sound extractability: *Let a polynomial-time oracle adversary* $\mathcal{A}$ *be given. Let* $(\mathrm{crs}, \mathrm{simtd}, \mathrm{extd}) \leftarrow \mathbf{K}(1^\eta)$. *Let* $\mathcal{O}(x,w) := \mathbf{S}(x, \mathrm{crs}, \mathrm{simtd})$ *if* $(x,w) \in R^{\mathrm{comp}}_{\mathrm{honest}}$ *and* $\perp$ *otherwise. Let* $(x, \mathrm{proof}) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, \mathrm{crs})$. *Let* $w \leftarrow \mathbf{E}(x, \mathrm{proof}, \mathrm{extd})$. *Then with overwhelming probability, if* $\mathbf{V}(x, \mathrm{proof}, \mathrm{crs}) = 1$ *and* proof *was not output by* $\mathcal{O}$ *then* $(x,w) \in R^{\mathrm{comp}}_{\mathrm{adv}}$.
- Unpredictability: *Let a polynomial-time adversary* $\mathcal{A}$ *be given. Let* $(\mathrm{crs}, \mathrm{simtd}, \mathrm{extd}) \leftarrow \mathbf{K}(1^\eta)$. *Let* $(x, w, \mathrm{proof}') \leftarrow \mathcal{A}(1^\eta, \mathrm{crs}, \mathrm{simtd}, \mathrm{extd})$. *Then with overwhelming probability, it holds* $\mathrm{proof}' \neq \mathbf{P}(x, w, \mathrm{crs})$ *or* $(x,w) \notin R^{\mathrm{comp}}_{\mathrm{honest}}$.
- Length-regularity: *Let two witnesses* w *and* w'*, and statements* x *and* x' *be given such that* $|x| = |x'|$*, and* $|w| = |w'|$. *Let* $(\mathrm{crs}, \mathrm{simtd}, \mathrm{extd}) \leftarrow \mathbf{K}(1^\eta)$. *Then let* $\mathrm{proof} \leftarrow \mathbf{P}(x, w, \mathrm{crs})$ *and* $\mathrm{proof}' \leftarrow \mathbf{P}(x', w', \mathrm{crs})$. *Then we get* $|\mathrm{proof}| = |\mathrm{proof}'|$ *with probability* 1.
- Deterministic verification and extraction: *The algorithms* V *and* E *are deterministic.*

(We do not explicitly list soundness because it is implied by honest simulation-sound extractability.) ◇

We then require that $A_{\mathrm{crs}}, A_{\mathrm{ZK}}, A_{\mathrm{verify}_{\mathrm{ZK}}}$ correspond to the key generation $\mathbf{K}$, prover $\mathbf{P}$, and verifier $\mathbf{V}$ of a weakly symbolically-sound ZK proof system for some relations $R^{\mathrm{comp}}_{\mathrm{honest}}, R^{\mathrm{comp}}_{\mathrm{adv}}$. We stress that using the the construction in [31] on a length-regular and extractable NIZK leads to weakly symbolically-sound ZK proof system. The proof is postponed to the Appendix B.

The Relations. It remains to specify what conditions we place on the relations $R^{\mathrm{comp}}_{\mathrm{honest}}, R^{\mathrm{comp}}_{\mathrm{adv}}$. Obviously, we cannot expect computational soundness if we allow arbitrary $R^{\mathrm{comp}}_{\mathrm{honest}}, R^{\mathrm{comp}}_{\mathrm{adv}}$. Instead, we need to formulate the fact that $R^{\mathrm{comp}}_{\mathrm{honest}}, R^{\mathrm{comp}}_{\mathrm{adv}}$ somehow correspond to the symbolic relations $R^{\mathrm{sym}}_{\mathrm{honest}}, R^{\mathrm{sym}}_{\mathrm{adv}}$. We thus give minimal requirements on the relationship between those relations. Essentially, we want that whenever $(x,w) \in R^{\mathrm{sym}}_{\mathrm{honest}}$ then for the corresponding computational bitstrings m_x, m_w we have $(m_x, m_w) \in R^{\mathrm{comp}}_{\mathrm{honest}}$; this guarantees that if symbolically, we respect the usage restriction $R^{\mathrm{sym}}_{\mathrm{honest}}$, then computationally we only use witnesses the honest protocol is allowed to use. And whenever $(m_x, m_w) \in R^{\mathrm{comp}}_{\mathrm{adv}}$ we have $(x,w) \in R^{\mathrm{sym}}_{\mathrm{adv}}$; this guarantees that a computational adversary will not be able to prove statements m_x that do not also correspond to statements x that can be proven symbolically. (Formally, these conditions are used to show Lemmas 4 and 6 in the computational soundness proof below.) To model correspondence between the symbolic terms x, w and the bitstrings m_x, m_w, we define a function img_η that translates a term to a bitstring (essentially by applying A_F for each constructor F). The function img_η depends on an environment η, a partial function $\mathbf{T} \to \{0,1\}^*$ that assigns bitstrings to nonces and adversary-generated terms. We use the definition of a *consistent environment* that lists various natural properties an environment will have (such as mapping ZK-terms to bitstrings of the right type); the definition of consistent environments is deferred to Appendix C.1. Given these notions, we can formalize the conditions $R^{\mathrm{comp}}_{\mathrm{honest}}, R^{\mathrm{comp}}_{\mathrm{adv}}$ should satisfy:

Definition 2 (Implementation of Relations). *A pair of relations* $R^{\text{comp}}_{\text{honest}}, R^{\text{comp}}_{\text{adv}}$ *on* $\{0,1\}^*$ *implement a relation* $R^{\text{sym}}_{\text{adv}}$ *on* $\mathbf{T}$ *with usage restriction* $R^{\text{sym}}_{\text{honest}}$ *if the following conditions hold for any consistent* $\eta \in \mathcal{E}$*:*

(i) $(x,w) \in R^{\text{sym}}_{\text{honest}}$ *and* $\text{img}_\eta(x) \neq \bot \neq \text{img}_\eta(w) \implies (\text{img}_\eta(x), \text{img}_\eta(w)) \in R^{\text{comp}}_{\text{honest}}$

(ii) $(\text{img}_\eta(x), \text{img}_\eta(w)) \in R^{\text{comp}}_{\text{adv}} \implies (x,w) \in R^{\text{sym}}_{\text{adv}}$

(iii) $R^{\text{sym}}_{\text{honest}} \subseteq R^{\text{sym}}_{\text{adv}}$ *and* $R^{\text{comp}}_{\text{honest}} \subseteq R^{\text{comp}}_{\text{adv}}$ ◇

In the Appendix C.5 we give some practical examples satisfying this definition.

Protocol Execution. The CoSP framework specifies semantics for executing a given protocol in the computational model given a computational implementation A_F. The execution is analogous to the symbolic execution, except that computation nodes apply functions A_F instead of constructors and destructors (with branching depending on $A_F(\dots) \stackrel{?}{=} \bot$). Input and output nodes receive and send bitstrings to a probabilistic polynomial-time adversary. This probabilistic process yields a trace of nodes, the *computational node trace*. Details are specified in [3].

4 Computational Soundness

Using the definitions from Section 2 and 3, we can finally state our main result. A *trace property* is a prefix-closed, efficiently decidable set $\mathcal{P}$ of node traces. We say a protocol Π *symbolically satisfies* $\mathcal{P}$ if every symbolic node trace (see page 211) of Π is in $\mathcal{P}$. We say Π *computationally satisfies* $\mathcal{P}$ if the computational node trace (see page 214) is in $\mathcal{P}$ with overwhelming probability.

Theorem 1 (Computational Soundness of ZK Proofs). *Let* Π *be a protocol satisfying the protocol conditions listed in figure 2. Let* A_F *be a computational implementation satisfying the implementation conditions from Section 3. Then for any node trace* $\mathcal{P}$*, if* Π *symbolically satisfies* $\mathcal{P}$*, then* Π *computationally satisfies* $\mathcal{P}$*.* ◇

We describe the proof in Section 5.

5 The Proof

In this section, we describe our proof of computational soundness (Theorem 1). First, we describe how the computational soundness proof for encryptions and signatures is done in the CoSP framework (Section 5.1). To understand our proof it is essential to understand that proof first. Then, we sketch how computational soundness of zero-knowledge proofs that have the extraction zero-knowledge property was shown in [10] (Section 5.2). It is instructive to compare their approach to ours. In Section 5.3, we describe the idea underlying our proof (using simulation-sound extractability instead of extraction-zero knowledge). Finally, in Section 5.4 we give an overview of our proof. The full proof is given in appendix C.3. The lemmas in this overview are simplified for readability and informal.

1. The annotation of each crs-node, each key-pair $(\mathsf{ek}, \mathsf{dk})$ and $(\mathsf{vk}, \mathsf{sk})$ is a fresh nonce. which does not occur anywhere else.
2. There is no node annotated with a $\mathsf{garbage}, \mathsf{garbageEnc}$, $\mathsf{garbageSig}$, $\mathsf{garbageZK}$, or $N \in \mathbf{N}_E$ constructor in the protocol.
3. The last argument of a $\mathsf{enc}, \mathsf{sig}, \mathsf{ZK}$ constructor are fresh nonces. These nonces are not used anywhere else except in case of enc and sig as part of a subterm of the third argument in a ZK-node.
4. A dk-node is only used as first argument for dec-node or as subterm of the third argument in a ZK-node.
5. A sk-node is only used as first argument for sig-node or as subterm of the third argument in a ZK-node.
6. The first argument of a dec-computation node is a dk-node.
7. The first argument of a sig-computation node is a sk-node.
8. The first argument of a ZK-computation is a crs-computation node which is annotated by a nonce $N \in \mathbf{N}_P$. This nonce is only used as annotation of this crs node and nowhere else.
9. The first argument of a $\mathsf{verify}_{\mathrm{ZK}}$-computation is a crs-computation node which is annotated by a nonce $N \in \mathbf{N}_P$. This nonce is only used as annotation of this crs node and nowhere else.
10. The protocol respects the usage restriction $R^{\mathrm{sym}}_{\mathrm{honest}}$ in the following sense:
In the symbolic execution of the protocol, whenever a ZK-computation-node ν is reached, then $(f(\nu_x), f(\nu_w)) \in R^{\mathrm{sym}}_{\mathrm{honest}}$ where f is the function mapping nodes to terms (cf. the definition of the symbolic execution in [3]) and ν_x and ν_w are the second and third argument of ν.
11. For the relation $R^{\mathrm{sym}}_{\mathrm{adv}}$ it holds: There is an efficient algorithm **SymbExtr**, that given a term M together with a set S of terms , outputs a term N, such that $S \vdash N$ and $(N, M) \in R^{\mathrm{sym}}_{\mathrm{adv}}$ or outputs $\perp$ if there is no such term N. We call a relation satisfying this property symbolically extractable.
12. The relation $R^{\mathrm{sym}}_{\mathrm{adv}}$ is efficiently decidable.

Fig. 2. Protocol conditions

5.1 Computational Soundness Proofs in CoSP

Remember that in the CoSP framework, a protocol is modeled as a tree whose nodes correspond to the steps of the protocol execution; security properties are expressed as sets of node traces. Computational soundness means that for any polynomial-time adversary A the trace in the computational execution is, except with negligible probability, also a possible node trace in the symbolic execution. The approach for showing this is to construct a so-called simulator Sim. The simulator is a machine that interacts with a symbolic execution of the protocol Π on the one hand, and with the adversary A on the other hand; we call this a hybrid execution. The simulator has to satisfy the following two properties:

- Indistinguishability: The node trace in the hybrid execution is computationally indistinguishable from that in the computational execution with adversary A.
- Dolev-Yaoness: The simulator Sim never (except for negligible probability) sends terms t to the protocol with $S \nvdash t$ where S is the list of terms Sim received from the protocol so far.

The existence of such a simulator then guarantees computational soundness: Dolev-Yaoness guarantees that only node traces occur in the hybrid execution that are possible in the symbolic execution, and indistinguishability guarantees that only node traces occur in the computational execution that can occur in the hybrid one.

How to Construct a Simulator? In [3], the simulator Sim is constructed as follows: Whenever it gets a term from the protocol, it constructs a corresponding bitstring and sends it to the adversary, and when receiving a bitstring from the adversary it parses it and sends the resulting term to the protocol. Constructing bitstrings is done using a function β, parsing bitstrings to terms using a function τ. The simulator picks all random values and keys himself: For each protocol nonce N, he initially picks a bitstring r_N. He then translates, e.g., $\beta(N) := r_N$ and $\beta(\mathrm{ek}(N)) := A_{\mathrm{ek}}(r_N)$ and $\beta(\mathrm{enc}(\mathrm{ek}(N), t, M)) := A_{\mathrm{enc}}(A_{\mathrm{ek}}(r_N), \beta(t), r_M)$. Translating back also is natural: Given $m = r_N$, we let $\tau(m) := N$, and if c is a ciphertext that can be decrypted as m using $A_{\mathrm{dk}}(r_N)$, we set $\tau(c) := \mathrm{enc}(\mathrm{ek}(N), \tau(m), M)$. However, in the last case, a subtlety occurs: what nonce M should we use as symbolic randomness in $\tau(c)$? Here we distinguish two cases: If c was earlier produced by the simulator: Then c was the result of computing $\beta(t)$ for some $t = \mathrm{enc}(\mathrm{ek}(N), t', M)$ and some nonce M. We then simply set $\tau(c) := t$ and have consistently mapped c back to the term it came from.

If c was not produced by the simulator: In this case it is an adversary generated encryption, and M should be an adversary nonce to represent that fact. We could just use a fresh nonce $M \in \mathbf{N}_E$, but that would introduce the need of additional bookkeeping: If we compute $t := \tau(c)$, and later $\beta(t)$ is invoked, we need to make sure that $\beta(t) = c$ in order for the Sim to work consistently (formally, this is needed in the proof of the indistinguishability of Sim). And we need to make sure that when computing $\tau(c)$ again, we use the same M. This bookkeeping can be avoided using the following trick: We identify the adversary nonces with symbols N^m annotated with bitstrings m. Then $\tau(c) := \mathrm{enc}(\mathrm{ek}(N), \tau(m), N^c)$, i.e., we set $M := N^c$. This ensures that different c get different randomness nonces N^c, the same c is always assigned the same N^c, and $\beta(t)$ is easy to define: $\beta(\mathrm{enc}(\mathrm{ek}(N), m, N^c)) := c$ because we know that $\mathrm{enc}(\mathrm{ek}(N), m, N^c)$ can only have been produced by $\tau(c)$. To illustrate, here are excerpts of the definitions of β and τ (the first matching rule counts):

- $\tau(c) \coloneqq \mathrm{enc}(\mathrm{ek}(M), t, N)$ if c has earlier been output by $\beta(\mathrm{enc}(\mathrm{ek}(M), t, N))$ for some $M \in \mathbf{N}, N \in \mathbf{N}_P$
- $\tau(c) \coloneqq \mathrm{enc}(\mathrm{ek}(M), \tau(m), N^c)$ if c is of type ciphertext and $\tau(A_{\mathrm{ekof}}(c)) = \mathrm{ek}(M)$ for some $M \in \mathbf{N}_P$ and $m \coloneqq A_{\mathrm{dec}}(A_{\mathrm{dk}}(r_M), c) \neq \bot$
- $\beta(\mathrm{enc}(\mathrm{ek}(N), t, M)) \coloneqq A_{\mathrm{enc}}(A_{\mathrm{ek}}(r_N), \beta(t), r_M)$ if $M \in \mathbf{N}_P$
- $\beta(\mathrm{enc}(\mathrm{ek}(M), t, N^m)) \coloneqq m$ if $M \in \mathbf{N}_P$

Bitstrings m that cannot be suitably parsed are mapped into terms $\mathrm{garbage}(N^m)$ and similar that can then be mapped back by β using the annotation m.

Showing Indistinguishability. Showing indistinguishability essentially boils down to showing that the functions β and τ consistently translate terms back and forth. More precisely, we show that $\beta(\tau(m)) = m$ and $\tau(\beta(t)) = t$. Furthermore, we need to show that in any protocol step where a constructor or destructor F is applied to terms $t_1, \ldots, t_n$, we have that $\beta(F(t_1, \ldots, t_n)) = A_F(\beta(t_1), \ldots, \beta(t_n))$. This makes sure that the computational execution (where A_F is applied) stays in sync with the hybrid execution (where F is applied and the result is translated using β). The proofs of these facts are lengthy (involving case distinctions over all

constructors and destructors) but do not provide much additional insight; they are very important though because they are responsible for most of the implementation conditions that are needed for the computational soundness result.

Showing Dolev-Yaoness. The proof of Dolev-Yaoness is where most of the actual cryptographic assumptions come in. In this sketch, we will slightly deviate from the original proof in [3] for easier comparison with the proof in the present paper. The differences are, however, inessential. Starting from the simulator Sim, we introduce a sequence of simulators Sim_4, Sim_5, Sim_f. (We start the numbering with 4 because we later introduce additional simulators.)

In Sim_4, we change the function β as follows: When invoked as $\beta(\mathrm{enc}(\mathrm{ek}(N), t, M))$ with $M \in \mathbf{N}_P$, instead of computing $A_{\mathrm{enc}}(A_{\mathrm{ek}}(r_N), \beta(t), r_M)$, β invokes an encryption oracle $\mathcal{O}^N_{\mathrm{enc}}$ to produce the ciphertext c. Similarly, $\beta(\mathrm{ek}(N))$ returns the public key provided by the oracle $\mathcal{O}^N_{\mathrm{enc}}$. The hybrid executions of Sim and Sim_4 are then indistinguishable. (Here we use that the protocol conditions guarantee that no randomness is used in two places.) Also, the function τ is changed to invoke $\mathcal{O}^N_{\mathrm{enc}}$ whenever it needs to decrypt a ciphertext while parsing. Notice that if c was returned by $\beta(t)$ with $t := \mathrm{enc}(\dots)$, then $\tau(c)$ just recalls the term t without having to decrypt. Hence $\mathcal{O}^N_{\mathrm{enc}}$ is never asked to decrypt a ciphertext it produced.

In Sim_5, we replace the encryption oracle $\mathcal{O}^N_{\mathrm{enc}}$ by a fake encryption oracle $\mathcal{O}^N_{fake}$ that encrypts zero-plaintexts instead of the true plaintexts. Since $\mathcal{O}^N_{\mathrm{enc}}$ is never asked to decrypt a ciphertext it produced, IND-CCA security guarantees that the hybrid executions of Sim_4 and Sim_5 are indistinguishable. Since the plaintexts given to $\mathcal{O}^N_{fake}$ are never used, we can further change $\beta(\mathrm{enc}(N, t, M))$ to never even compute the plaintext $\beta(t)$.

Finally, in Sim_f, we additionally change β to use a signing oracle in order to produce signatures. As in the case of Sim_4, the hybrid executions of Sim_5 and Sim_f are indistinguishable.

Since the hybrid executions of Sim and Sim_f are indistinguishable, in order to show Dolev-Yaoness of Sim, it is sufficient to show Dolev-Yaoness of Sim_f.

The first step to showing this is to show that whenever Sim_f invokes $\beta(t)$, then $S \vdash t$ holds (where S are the terms received from the protocol). This follows from the fact that β is invoked on terms t_0 sent by the protocol (which are then by definition in S), and recursively descends only into subterms that can be deduced from t_0. In particular, in Sim_5 we made sure that $\beta(t)$ is not invoked by $\beta(\mathrm{enc}(\mathrm{ek}(N), t, M))$; t would not be deducible from $\mathrm{enc}(\mathrm{ek}(N), t, M)$. Next we prove that whenever $S \nvdash t$, then t contains a visible subterm t_{bad} with $S \nvdash t_{bad}$ such that t_{bad} is a protocol nonce, or a ciphertext $\mathrm{enc}(\dots, N)$ where N is a protocol nonces, or a signature, or a few other similar cases. (Visibility is a purely syntactic condition and essentially means that t_{bad} is not protected by an honestly generated encryption.)

Now we can conclude Dolev-Yaoness of Sim_f: If it does not hold, Sim_f sends a term $t = \tau(m)$ where m was sent by the adversary A. Then t has a visible subterm t_{bad}. Visibility implies that the recursive computation of $\tau(m)$ had a subinvocation $\tau(m_{bad}) = t_{bad}$. For each possible case of t_{bad} we derive a con-

tradiction. At this point we use the cryptographic arguments like for example unforgeability of signature schemes. Thus, Sim_f is Dolev-Yao, hence Sim is indistinguishable and Dolev-Yao. Computational soundness follows.

5.2 Computational Soundness Based on Extraction ZK

We now describe how computational soundness for zero-knowledge proofs was shown in [10], based on the strong assumption of extraction zero-knowledge. Our presentation strongly deviates from the details of the proof in [10]; we explain what their proof would be like if recast in the CoSP framework. This makes it easier to compare the proof to our proof and the proof described in the preceding section.

Extraction zero-knowledge is a strong property that guarantees the following: It is not possible to distinguish a prover-oracle from the a simulator-oracle, even when given access to an extraction oracle that extracts the witnesses from arbitrary proofs except the ones produced by the prover/simulator-oracle. Notice that there is a strong analogy to IND-CCA secure encryption. The prover-oracle corresponds to an encryption-oracle, the witness to the plaintext, the simulator-oracle to a fake encryption-oracle encrypting zero-plaintexts, and the extractor-oracle to a decryption-oracle.

This analogy allows us to adapt the idea for proving computational soundness of encryptions to the case of ZK proofs. As in the proof described in Section 5.1, we construct a simulator Sim with translation functions τ and β. We extend β and τ to deal with ZK proofs analogue to the cases of encryptions in Section 5.1.

The proof of indistinguishability is analogous to that in Section 5.1, except that we use the extractability property of the proof system to make sure that the simulator does not abort when invoking the extraction algorithm while trying to parse a ZK proof z in $\tau(z)$. Notice that plain extractability (as opposed to simulation-sound extractability) can be used here since we do not use a ZK-simulator in the construction of Sim.

To prove Dolev-Yaoness, we proceed as in Section 5.1, except that we introduce three more intermediate simulators Sim_1, Sim_2, and Sim_3. (See Figure 3.) In Sim_1, we invoke a prover-oracle $\mathcal{O}^N_{\mathrm{ZK}}$ with statement $\beta(t)$ and witness $\beta(t')$ in $\beta(\mathrm{ZK}(\mathrm{crs}(N), t, t', M))$ instead of computing $A_{\mathrm{ZK}}(A_{\mathrm{crs}}(r_N), \beta(t), \beta(t'), r_M)$. (This is analogous to Sim_4 above.) $\mathcal{O}^N_{\mathrm{ZK}}$ aborts if the witness is not valid.

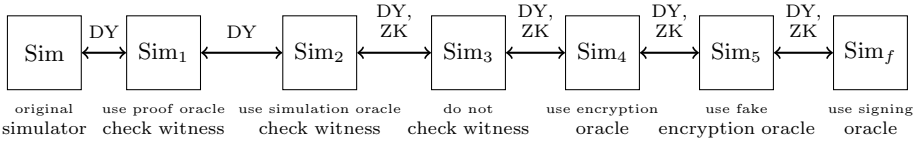

Fig. 3. Simulators used in the proof. An arrow marked DY means Dolev-Yaoness is propagated from one simulator to the other. An arrow marked ZK means ZK-breaks are propagated (needed in Section 5.4).

In Sim_2, we replace the prover-oracle $\mathcal{O}^N_{\mathrm{ZK}}$ by a ZK-simulator-oracle $\mathcal{O}^N_{\mathrm{sim}}$. That oracle runs the ZK-simulator (after checking that the witness is valid).

Extraction zero-knowledge guarantees that this replacement leads to an indistinguishable hybrid execution. (We need that the witness is checked before running the simulator because extraction zero-knowledge gives no guarantees in the case of invalid witnesses, even if the witness is not actually used by the ZK-simulator.) Finally, in Sim_3 we modify the ZK-simulator-oracle $\mathcal{O}^N_{\mathrm{sim}}$ such that it does not check the witness any more. A protocol condition guarantees that this check would succeed anyway, so this change leads to an indistinguishable hybrid execution. Furthermore, since witnesses given to $\mathcal{O}^N_{\mathrm{sim}}$ are never used, we can further change $\beta(\mathrm{ZK}(\mathrm{crs}(N), t, t', M))$ to never even compute the witness $\beta(t')$.

The rest of the proof is analogous to that in Section 5.1. I.e., we continue with the simulator $\mathrm{Sim}_3, \mathrm{Sim}_4, \mathrm{Sim}_f$ as described there and show that Sim_f is Dolev-Yao.

Note that this computational soundness proof crucially depends on the extraction ZK property. We need to use the extractor in the construction of τ, and we need to replace the prover-oracle by a ZK-simulator-oracle in order to make sure that β does not descend into witnesses. And that replacement takes place in a setting where the parsing function τ and thus the extractor is used.

5.3 Proof Idea

We now describe the idea of our approach that allows us to get rid of extraction ZK. As explained in Section 5.2, we cannot use the extractor as part of the parsing function τ if we do not have extraction ZK. However, the following observation shows that we might not need to run the extractor: Although in the computational setting, the only way to compute a witness is to extract it (unless the relation is trivial), in the symbolic setting, given a symbolic statement x, it is typically easy to compute a corresponding symbolic witness w. (E.g., when proving the knowledge of a secret key that decrypts a term $c = \mathrm{enc}(\mathrm{ek}(N), t, M)$, then the witness is $\mathrm{dk}(N)$ which can just be read off c.) We stress that we do not claim that the witness can be deduced (in the sense of $\vdash$) from x, only that its symbolic representation can be efficiently computed from the statement.

Thus, for an adversary-generated proof z with CRS $A_{\mathrm{crs}}(r_N)$ and statement m_x and that passes verification, we define $\tau(z)$ as follows: We run $w := \mathbf{SymbExtr}(S, x)$ and return $\tau(z) := \mathrm{ZK}(\mathrm{crs}(N), x, w, N^z)$. Here S is the list of terms send by the protocol so far, $\mathbf{SymbExtr}(S, x)$ denotes an arbitrary witness w satisfying the following two conditions: w is a valid witness for x (i.e., $(x, w) \in R^{\mathrm{sym}}_{\mathrm{adv}}$) and $S \vdash w$. (Our result assumes that $w = \mathbf{SymbExtr}(S, x)$ is efficiently computable whenever w exists, this will be the case for most natural relations.)

The condition $S \vdash w$ is necessary since otherwise the simulator Sim would produce a proof that the adversary could not have deduced (since he could not have deduced the witness), and thus the simulator would not be Dolev-Yao.

Assume for the moment that $\mathbf{SymbExtr}(S, x)$ always succeeds (i.e., in the hybrid execution, there always is a w with $(x, w) \in R^{\mathrm{sym}}_{\mathrm{adv}}$ and $S \vdash w$). In this case, we can finish the proof analogously to that in Section 5.2: Indistinguishability of Sim follows by carefully checking all cases, and the Dolev-Yaoness by the same sequence of simulators as in Section 5.2. We do not need extraction

zero-knowledge when going from Sim_1 to Sim_2, though, because in Sim_1, no extractor is used (we use symbolic extraction instead). Thus the zero-knowledge property is sufficient instead of extraction zero-knowledge.

But how to show that $\mathbf{SymbExtr}(S, x)$ always succeeds? Two things might go wrong. First, no valid witness w with $(x, w) \in R_{\mathrm{adv}}^{\mathrm{sym}}$ might exist. Note that the extractability property only guarantees that computationally, a valid witness for the computational statement m_x exists. This does not necessarily imply that translating that witness into a term (e.g., using τ) yields a valid symbolic witness. Second, there might be a valid witness w, but that witness is not deducable ($S \not\vdash w$). Again, extractability only guarantees that the adversary "knows" a witness in the computational setting, this does not imply deducability in the symbolic setting.

In essence, to show that $\mathbf{SymbExtr}(S, x)$ succeeds, we need a kind of computational soundness result: Whenever computationally, the adversary knows a valid witness, then symbolically, the adversary knows a valid witness. This seems problematic, because it seems that we need to use a computational soundness result within our proof of computational soundness – a seeming circularity. Fortunately, this circularity can be resolved: The fact that $\mathbf{SymbExtr}(S, x)$ succeeds is used only when proving that Sim is indistinguishable (i.e., mimics the computational execution well). But the fact that $\mathbf{SymbExtr}(S, x)$ succeeds does not relate to the computational execution at all. In fact, it turns out to be closely related to the Dolev-Yaoness and can be handled in the same proof. And that proof does not use the fact that symbolic extraction succeeds.

5.4 Proof Overview

We now give a more detailed walk-through through our proof. This exposition can also be seen as a guide through the full proof in appendix C.3.

The Simulator. The first step is to define the simulator Sim, i.e., the translation function β and τ. Here, we only present the parts of the definition related to ZK proofs (the first matching rule counts):

1. $\tau(z) := \mathrm{ZK}(\mathrm{crs}(N_1), t_1, t_2, N_2)$ if z has earlier been output by $\beta(\mathrm{ZK}(\mathrm{crs}(N_1), t_1, t_2, N_2))$ for some $N_1, N_2 \in \mathbf{N}_P$
2. $\tau(z) := \mathrm{ZK}(\mathrm{crs}(N), x, w, N^z)$ if z is of type zero-knowledge proof and $\tau(z)$ was computed earlier and has output $\mathrm{ZK}(\mathrm{crs}(N), x, w, N^z)$
3. $\tau(z) := \mathrm{ZK}(\mathrm{crs}(N), x, w, N^z)$ if z is of type zero-knowledge proof, $\tau(A_{\mathrm{crsof}}(z)) = \mathrm{crs}(N)$ for some $N \in \mathbf{N}_P$, $A_{\mathrm{verify_{ZK}}}(A_{\mathrm{crsof}}(z), z) = z$, $m_x := A_{\mathrm{getPub}}(z) \neq \bot$, $x := \tau(m_x) \neq \bot$ and $w := \mathbf{SymbExtr}(S, x)$ where S is the set of terms sent to the adversary so far.
4. $\tau(z) := \mathrm{garbageZK}(c, x, N^z)$ if z is of type zero-knowledge proof, $c := \tau(A_{\mathrm{crsof}}(z))$ and $x := \tau(A_{\mathrm{getPub}}(z))$.
5. $\beta(\mathrm{ZK}(\mathrm{crs}(N_1), t_1, t_2, N_2)) := A_{\mathrm{ZK}}(A_{\mathrm{crs}}(r_{N_1}), \beta(t_1), \beta(t_2), r_{N_2})$ if $N_1, N_2 \in \mathbf{N}_P$
6. $\beta(\mathrm{ZK}(\mathrm{crs}(t_0), t_1, t_2, N^s)) := s$
7. $\beta(\mathrm{garbageZK}(t_1, t_2, N^z)) := z$

Here $\mathbf{SymbExtr}(S, x)$ returns a witness w with $(x, w) \in R_{\text{adv}}^{\text{sym}}$ and $S \vdash w$ if such w exists, and $\bot$ otherwise. A key point is what to do when $\mathbf{SymbExtr}(S, x)$ fails. We will later show that this happens with negligible probability only, but for now we need to specify the behavior in this case: When $\mathbf{SymbExtr}(S, x)$ returns $\bot$ in the rule 3), we say an *extraction failure* occurred. In this case, the simulator runs the extractor (using the extraction trapdoor corresponding to $A_{\text{crs}}(r_N)$) to get a (computational) witness m_w for m_x. Then Sim computes $w := \tau^*(m_w)$ where τ^* is defined like τ, except that the rule 3) is dropped (hence τ^* will map an adversary-generated ZK-proof always to a garbageZK-term). Then the simulator aborts. If $(x, w) \notin R_{\text{adv}}^{\text{sym}}$, we say a *ZK-break* occurred.

The reader may wonder why we let the simulator compute a symbolic witness w in case of an extraction failure even though w is never used. The reason is that we later show that this w always has $(x, w) \in R_{\text{adv}}^{\text{sym}}$ and $S \vdash w$, which contradicts the fact that we get an extraction failure in the first place. The reason for using τ^* instead of τ is that we have to avoid getting extraction failures within extraction failures. We use the same sequence of simulator modifications as in Section 5.2 (see Figure 3). The only difference is that the simulator Sim handles zero-knowledge proofs as defined above.

We can now show that Sim_f is Dolev-Yao. The proof of this fact is analogous to the case the proof sketched in Section 5.2. We even show something slightly stronger, namely that neither τ nor τ^* outputs an undeducable term:

Lemma 1 (Sim_f **is Dolev-Yao).** *For any invocation $t := \tau(m)$ or $t := \tau^*(m)$, we have $S \vdash t$ where S are the terms sent to the simulator so far. In particular,* Sim_f *is Dolev-Yao.* ◇

As in Section 5.2, we show Sim is Dolev-Yao iff Sim_f is Dolev-Yao. We will also need preservation of the property that ZK-breaks occur with negligible probability.

Lemma 2 (Preservation of Simulator-Properties). • Sim *is Dolev-Yao iff* Sim_f *is.* • *In the hybrid execution of* Sim *extraction failures occur with negligible probability iff the same holds for* Sim_f. • *In the hybrid execution of* Sim_2 *(not* Sim*!) ZK-breaks occur with negligible probability iff the same holds for* Sim_f. ◇

Dolev-Yaoness, extraction failures, and ZK-breaks carry over from Sim_3 to Sim_4 and from Sim_5 to Sim_f because the randomness used in encrypting and signing is not re-used by protocol condition 3. (Notice that randomness might have occurred within a witness, but due to the change in Sim_3, we do not invoke $\beta(w)$ on witnesses any more.) Dolev-Yaoness, extraction failures, and ZK-breaks carry over from Sim_4 to Sim_5 due to the IND-CCA property. Dolev-Yaoness and extraction failures carry over from Sim to Sim_1 because the randomness used for constructing ZK-proofs is not reused by protocol condition 3.

Furthermore, Dolev-Yaoness and extraction failures carry over from Sim_1 to Sim_2 because of the zero-knowledge property of the proof system. There is a subtlety here: Sim_1 does use the extractor (namely after an extraction failure). So usually, we would not be allowed to apply the zero-knowledge property (we

would need extraction ZK). But fortunately, after an extraction failure, no terms are sent by the simulator. Thus, anything that happens after an extraction failure has no impact on whether the simulator is Dolev-Yao or not. Thus, for analyzing whether Dolev-Yaoness carries over from Sim_1 to Sim_2, we can assume that those simulators abort directly after incurring an extraction failure (without invoking the extractor afterwards). Then no extractions occur in the simulator, and we can use the zero-knowledge property. Analogously, extraction failures carry over from Sim_1 to Sim_2.

Notice that we cannot use the same trick to show that ZK-breaks carry over from Sim_1 to Sim_2: Whether ZK-breaks occur is determined after the invocation of the extractor. Fortunately, we only need that ZK-breaks carry over from Sim_2 to Sim_f.

To show Lemma 2, it remains to show that Dolev-Yaoness, extraction failures, and ZK-breaks carry over from Sim_2 to Sim_3. The only difference between these simulators is that Sim_3 does not check whether the witness m_w given to the ZK-simulation-oracle is valid (i.e., $(\beta(t_1), \beta(t_2)) \in R^{\mathrm{comp}}_{\mathrm{honest}}$ in rule 5). Thus, to conclude the proof of Lemma 2, we need to show that the probability that the ZK-simulation-oracle is called with an invalid witness is negligible.

No Invalid Witnesses. To show that the ZK-simulation-oracle is only called by Sim_2 with valid computational witnesses $\beta(t_1)$, we need to show two things:

Lemma 3 (No Invalid Symbolic Witnesses). *If* Sim_3 *is Dolev-Yao, then in rule 5), we have* $(t_1, t_2) \in R^{\mathrm{sym}}_{\mathrm{honest}}$ *with overwhelming probability. The same holds for* Sim. ◇

Lemma 4 (Relating the Relations, Part 1). *In an execution of* Sim_3 *the following holds with overwhelming probability: if* $(x, w) \in R^{\mathrm{sym}}_{\mathrm{honest}}$ *then* $(\beta(x), \beta(w)) \in R^{\mathrm{comp}}_{\mathrm{honest}}$. *The same holds for* Sim. ◇

Once we have these lemmas, Lemma 2 follows: We know from Lemma 1 that Sim_f is Dolev-Yao. We have already shown that this property carries over to Sim_3. Thus by Lemmas 3 and 4, $(\beta(t_1), \beta(t_2)) \in R^{\mathrm{comp}}_{\mathrm{honest}}$ in rule 5).

To show Lemma 3, we observe the following: If the simulator sends only terms that are deducible (i.e., that a symbolic adversary could also have sent), then in the hybrid execution, no execution trace occurs that could not have occurred in the symbolic execution either. By protocol condition 10, in a symbolic execution, $(t_1, t_2) \in R^{\mathrm{sym}}_{\mathrm{honest}}$ whenever the protocol constructs an $\mathrm{ZK}(\mathrm{crs}(N), t_1, t_2, M)$-term. Since rule 5) only applies to such protocol-generated terms (ZK-terms from τ have $M \in \mathbf{N}_E$), it follows that $(t_1, t_2) \in R^{\mathrm{sym}}_{\mathrm{honest}}$ in rule 5). Lemma 3 follows. Lemma 4 follows because we required that $R^{\mathrm{comp}}_{\mathrm{honest}}, R^{\mathrm{comp}}_{\mathrm{adv}}$ implement $R^{\mathrm{sym}}_{\mathrm{adv}}$ with usage restriction $R^{\mathrm{sym}}_{\mathrm{honest}}$; Definition 2 was designed to make Lemma 4 true.

Thus, Lemmas 3 and 4 hold, thus Lemma 2 follows. Since Sim_f is Dolev-Yao by Lemma 1, it follows with Lemma 2 that Sim is Dolev-Yao. It remains to show that Sim is indistinguishable.

Indistinguishability of Sim. As described in Section 5.1, to show indistinguishability of Sim, the main subproof is to show (a) that $\beta(F(t_1, \ldots, t_n))$

$= A_F(\beta(t_1), \ldots, \beta(t_n))$ when the protocol computes $F(t_1, \ldots, t_n)$. And, of course, we need (b) that the simulator does not abort. The proof of (a) is, as before, done by careful checking of all cases. The only interesting case is $F = \text{verify}_{\text{ZK}}$. Here we need that an honestly-generated ZK proof with statement x and witness w passes verification symbolically $(x, w \in R^{\text{sym}}_{\text{honest}})$ iff it passes verification computationally $((\beta(x), \beta(w) \in R^{\text{comp}}_{\text{honest}})$. Fortunately, we have already derived all needed facts: By Lemmas 1 and 3, $(x, w) \in R^{\text{sym}}_{\text{honest}}$ with overwhelming probability. And then by Lemma 4, $(\beta(x), \beta(w)) \in R^{\text{sym}}_{\text{honest}}$.

To show (b), we need to show that extraction failures occur with negligible probability. The approach for this is a bit roundabout, we first analyze Sim_2:

Lemma 5 (No ZK-Breaks). *In the hybrid execution of* Sim_2*, ZK-breaks occur with negligible probability.* ◇

To show this, we use the simulation-sound extractability property of the proof system to show that the values m_x, m_w extracted by the extractor after an extraction failure satisfy $(m_x, m_w) \in R^{\text{comp}}_{\text{adv}}$. And then it follows that $(x, w) \in R^{\text{sym}}_{\text{adv}}$ with $x := \tau(m_x)$, $w := \tau^*(m_w)$ by the converse of Lemma 4:

Lemma 6 (Relating the Relations, Part 2). *In an execution of* Sim_2 *the following holds with overwhelming probability: if* $(m_x, m_w) \in R^{\text{comp}}_{\text{adv}}$ *then* $(\tau(m_x), \tau^*(m_w)) \in R^{\text{sym}}_{\text{adv}}$. ◇

Thus Lemma 5 is shown. From this, with Lemma 2 we get that ZK-breaks occur with negligible probability also for Sim_f. Based on this fact, we can show the following lemma:

Lemma 7 (No Extraction Failures). *In the hybrid execution of* Sim_f*, extraction failures occur with negligible probability.* ◇

To see this, we use that ZK-breaks only occur with negligible probability in the execution of Sim_f. Thus, by definition of ZK-breaks, this means that $(x, w) \in R^{\text{sym}}_{\text{adv}}$ for the terms $x := \tau(m_x)$ and $w := \tau(m_w)$ computed after the extraction failure. Furthermore, by Lemma 1, it follows that $S \vdash w$. But then, by definition, $\textbf{SymbExtr}(x, S)$ would have output a w or another witness, but not $\perp$. Thus the extraction failure would not have occurred. This shows Lemma 7.

Finally, from Lemmas 5 and 2 we get that extraction failures occur with negligible probability in the execution of Sim, too. Thus property (b) also holds, thus we have shown Sim to be indistinguishable.

Notice that the roundabout way through Sim_2 and Sim_f to show that extraction failures occur with negligible probability in the execution of Sim is necessary: We cannot directly show Lemma 5 for Sim_f because Sim_f uses the simulator to prove untrue statements (e.g., it may prove that a ciphertext contains a certain value, but since we use a fake encryption oracle, that ciphertext actually contains a zero-plaintext), so simulation-sound extractability cannot be applied. Also, we cannot use the fact $S \vdash \tau^*(x)$ directly on Sim because this fact cannot be propagated from Sim_f to Sim (since τ^* is executed after the extractor is used, we would need extraction ZK to bridge from Sim_2 to Sim_1).

Concluding the Proof. We have shown that Sim is Dolev-Yao and indistinguishable. From [3, Thm. 1] we then immediately get Theorem 1.

6 Conclusions

In this paper, we have shown that computational soundness of symbolic ZK proofs can be achieved under realistic cryptographic assumptions for which efficient realizations and generic constructions are known. The computational soundness proof has been conducted in CoSP, and hence it holds independent of the underlying symbolic calculi and comes with mechanized proof support.

We conclude by highlighting two open questions that we consider as future work. First, current abstractions model non-interactive ZK proofs, i.e., a ZK proof constitutes a message that can forwarded, put into other terms, etc. Developing a symbolic abstraction to reflect (the more common) interactive ZK proofs thus requires a conceptually different approach, as such proofs cannot be replayed, put into other terms, etc. We plan to draw ideas from a recently proposed symbolic abstraction for (interactive) secure multi-party computation [6] to reflect this behavior. Second, soundness proofs of individual primitives have typically been proved in isolation, without a guarantee that the soundness result prevails when composed. We plan to build on recent work on composable soundness notions [17] to establish a composable soundness result for ZK proofs.

References

1. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. In: ACM CCS (1997)
2. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology 15(2) (2002)
3. Backes, M., Hofheinz, D., Unruh, D.: Cosp: A general framework for computational soundness proofs. In: ACM CCS (2009)
4. Backes, M., Hriţcu, C., Maffei, M.: Type-checking zero-knowledge. In: ACM CCS (2008)
5. Backes, M., Lorenz, S., Maffei, M., Pecina, K.: Anonymous Webs of Trust. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 130–148. Springer, Heidelberg (2010)
6. Backes, M., Maffei, M., Mohammadi, E.: Computationally sound abstraction and verification of secure multi-party computations. In: FSTTCS (2010)
7. Backes, M., Maffei, M., Unruh, D.: Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In: IEEE S&P (2008)
8. Backes, M., Maffei, M., Unruh, D.: Computationally sound verification of source code. In: ACM CCS (2010)
9. Backes, M., Pfitzmann, B.: Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In: IEEE CSFW (2004)
10. Backes, M., Unruh, D.: Computational soundness of symbolic zero-knowledge proofs. Journal of Computer Security 18(6) (2010)

11. Basin, D., Mödersheim, S., Viganò, L.: OFMC: A symbolic model checker for security protocols. International Journal of Information Security (2004)
12. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE CSFW (2001)
13. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: ACM CCS (2004)
14. Canetti, R., Herzog, J.: Universally Composable Symbolic Analysis of Mutual Authentication and Key-Exchange Protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 380–403. Springer, Heidelberg (2006)
15. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a Secure Voting System. In: IEEE S&P (2008)
16. Cortier, V., Warinschi, B.: Computationally Sound, Automated Proofs for Security Protocols. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 157–171. Springer, Heidelberg (2005)
17. Cortier, V., Warinschi, B.: A composable computational soundness notion. In: Proc. 18th ACM CCS (2011)
18. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2) (1983)
19. Even, S., Goldreich, O.: On the security of multi-party ping-pong protocols. In: IEEE CSF (1983)
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1) (1989)
21. Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
22. Groth, J., Ostrovsky, R.: Cryptography in the Multi-string Model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
23. Kemmerer, R., Meadows, C., Millen, J.: Three systems for cryptographic protocol analysis. Journal of Cryptology 7(2) (1994)
24. Laud, P.: Symmetric encryption in automatic analyses for confidentiality against active adversaries. In: IEEE S&P (2004)
25. Li, H., Li, B.: An Unbounded Simulation-Sound Non-interactive Zero-Knowledge Proof System for NP. In: Feng, D., Lin, D., Yung, M. (eds.) CISC 2005. LNCS, vol. 3822, pp. 210–220. Springer, Heidelberg (2005)
26. Lowe, G.: Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996)
27. Lu, L., Han, J., Liu, Y., Hu, L., Huai, J.-P., Ni, L., Ma, J.: Pseudo trust: Zero-knowledge authentication in anonymous p2ps. IEEE Trans. Parallel Distrib. Syst. (2008)
28. Maffei, M., Pecina, K.: Position paper: Privacy-aware proof-carrying authorization. In: PLAS (2011)
29. Merritt, M.: Cryptographic Protocols. PhD thesis, Georgia Tech (1983)
30. Micciancio, D., Warinschi, B.: Soundness of Formal Encryption in the Presence of Active Adversaries. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 133–151. Springer, Heidelberg (2004)
31. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS (1999)
32. Sahai, A.: Simulation-sound non-interactive zero knowledge. Technical report, IBM Research Report RZ 3076 (2001)
33. Schneider, S.: Security properties and CSP. In: IEEE S&P (1996)