On Simulatability Soundness and Mapping Soundness of Symbolic Cryptography

Michael Backes[1], Markus Dürmuth[1], and Ralf Küsters[2]

[1] Saarland University, Saarbrücken, Germany, {backes|duermuth}@cs.uni-sb.de
[2] ETH Zürich, Switzerland, ralf.kuesters@inf.ethz.ch

Abstract. The abstraction of cryptographic operations by term algebras, called Dolev-Yao models or symbolic cryptography, is essential in almost all tool-supported methods for proving security protocols. Recently significant progress was made – using two conceptually different approaches – in proving that Dolev-Yao models can be sound with respect to actual cryptographic realizations and security definitions. One such approach is grounded on the notion of simulatability, which constitutes a salient technique of Modern Cryptography with a longstanding history for a variety of different tasks. The other approach strives for the so-called mapping soundness – a more recent technique that is tailored to the soundness of specific security properties in Dolev-Yao models, and that can be established using more compact proofs. Typically, both notions of soundness for similar Dolev-Yao models are established separately in independent papers.

This paper relates the two approaches for the first time. Our main result is that simulatability soundness entails mapping soundness provided that both approaches use the same cryptographic implementation. Hence, future research may well concentrate on simulatability soundness whenever applicable, and resort to mapping soundness in those cases where simulatability soundness constitutes too strong a notion.

1 Introduction

Tool-supported verification of cryptographic protocols almost always relies on abstractions of cryptographic operations by term algebras with cancellation rules, called *symbolic cryptography* or *Dolev-Yao models* after the first authors [16]. An example term is $\mathsf{D}_{ske}(\mathsf{E}_{pke}(\mathsf{E}_{pke}(N)))$, where E and D denote public-key encryption and decryption, ske and pke are corresponding private and public encryption keys, and N is a nonce (random string). The keys are written as indices for readability; formally they are normal operands in the term. A typical cancellation rule is $\mathsf{D}_{ske}(\mathsf{E}_{pke}(t)) = t$ for all public/private key pairs (pke, ske) and terms t, thus the above term is equivalent to $\mathsf{E}_{pke}(N)$. The proof tools handle these terms symbolically, i.e., they never evaluate them to bit strings. In other words, the tools perform abstract algebraic manipulations on trees consisting of operators and base messages, using only the cancellation rules, the message-construction rules of a particular protocol, and an abstract model of networks and adversaries.

It is not at all clear from the outset whether Dolev-Yao models are a sound abstraction from real cryptography with its computational security definitions, where messages are bit strings and the adversary is an arbitrary probabilistic polynomial-time (ppt) Turing machine. In particular, the tools assume that *only* the modeled operations and cancellation

rules are possible manipulations on terms, and that terms that cannot be constructed with these rules are completely secret. For instance, if an adversary (also called intruder) only saw the example term above and only the mentioned cancellation rule was given, then N would be considered secret. Bridging this long-standing gap between Dolev-Yao models and real cryptographic definitions has recently received considerable attention, and remarkable progress has been made using two conceptually different approaches.

One such approach, henceforth called *simulatability soundness*, is grounded on the security notion of (black-box reactive) simulatability (BRSIM), which relates a real system (also called implementation or real protocol) with an ideal system (also called ideal functionality or ideal protocol). The real system is said to be as secure as the ideal system if every attack on the real system can be turned into an "equivalent" attack on the ideal system, where "equivalent" means indistinguishable by an environment (also called honest users). This security notion essentially means that the real system can be plugged into an arbitrary protocol instead of the ideal system without any noticeable difference [20, 21, 10]. Basically the same notion is also called UC (universal composability) for its universal composition properties [11].[1] In terms of the semantics community, BRSIM/UC could be called an implementation or refinement relation, with a particular emphasis on also retaining secrecy properties, in contrast to typical implementation relations. Now, results on simulatability soundness show that a (possibly augmented) Dolev-Yao model, specified as an ideal system, can be implemented in the sense of BRSIM/UC by a real system using standard cryptographic definitions. The first such result was presented in [8] and was extended to more cryptographic primitives in [9, 7]. The use of these results in protocol proofs was illustrated in [6, 3, 22, 2]. Simulatability soundness of a slightly simpler Dolev-Yao model and a restricted class of protocols using it was proven in [12].

The other approach, henceforth called *mapping soundness*, is tailored to the soundness of specific security properties in standard Dolev-Yao models. Mapping soundness of a given protocol is established by showing the existence of a mapping from bit strings to terms such that applying the mapping to an arbitrary trace of the real cryptographic execution of the protocol yields a trace of an ideal, Dolev-Yao style execution of the protocol. Compared to simulatability soundness, mapping soundness can often be established by more compact proofs and sometimes more relaxed cryptographic assumptions. Unlike simulatability soundness however, mapping soundness is restricted to specific protocol classes, and it does not entail universal composition properties. The first result on mapping soundness considered symmetric encryption under passive attacks [1]. Various later papers extended this approach to active attacks and to different cryptographic primitives and security properties [19, 18, 15, 14, 12]. In this paper, we are concerned with mapping soundness for active attacks.

1.1 Our Results

Our paper relates these two approaches for the first time. Our main result is that simulatability soundness entails mapping soundness provided that both approaches use the same cryptographic implementation. More precisely, we show that given an arbitrary ideal system $\mathsf{M}^{\mathsf{ideal}}$ and an arbitrary real protocol $\mathsf{M}^{\mathsf{real}}$, mapping soundness of $\mathsf{M}^{\mathsf{real}}$ necessarily

[1] While the definitions of BRSIM and UC have not been rigorously mapped, we believe that for the results in this paper the differences do not matter, in particular if one thinks of the equivalent blackbox version of UC [11]. Similarly, we believe that the results would hold in the formalism put forward in [17].

holds provided that the following two assumptions are met: First, the traces of the ideal system constitute Dolev-Yao style traces, i.e., traces that can be constructed according to the rules of the term algebra and of the protocol under consideration; second, $\mathsf{M}^{\mathsf{real}}$ is as secure as $\mathsf{M}^{\mathsf{ideal}}$ in the sense of BRSIM/UC, i.e., simulatability soundness holds for the ideal and real systems under consideration. Interestingly, this result does not dependent on details of the simulator, which translates between cryptographic bit strings and their Dolev-Yao abstractions in simulatability soundness.

We note that requiring the same cryptographic implementations for both simulatability soundness and mapping soundness means that existing results on simulatability soundness do not necessarily fully supersede existing results on mapping soundness: the former results may, e.g., require stronger assumptions on the security of cryptographic primitives, specific techniques from robust protocol design such as explicit type tags, additional randomization, etc. in order to establish simulatability between the cryptographic implementation and its Dolev-Yao abstraction. However, we believe that it is now fair to say that future research may concentrate on simulatability soundness whenever applicable, and resort to mapping soundness in those cases where simulatability soundness constitutes too strong a notion.

1.2 Paper Outline

Section 2 reviews the basic terminology of symbolic cryptography, its deduction rules, and the syntax of protocols. Section 3 reviews the notion of simulatability and points out necessary requirements for a Dolev-Yao model to be sound in the sense of BRSIM/UC. Section 4 defines executions of protocols within the reactive simulatability framework [21, 10], thus preparing a common ground for comparing both notions of soundness. Section 5 finally proves that simulatability soundness implies mapping soundness.

The long version of this paper [4] contains further expositions that are omitted here for space reasons; in particular, it reviews the substantial body of literature substantiating the relevance of simulatability in Modern Cryptography, and the newly arising area of formulating syntactic calculi for dealing with probabilism and polynomial-time considerations directly (without relying on Dolev-Yao models).

2 Symbolic Cryptography

In this section we review basic terminology concerning Dolev-Yao models and the corresponding deduction rules for deriving new messages from a given set of messages. In addition, we describe the syntax of protocols along the lines of works on the mapping approach [19, 15, 14].

2.1 Basic Terminology, Dolev-Yao Terms, and Deduction Rules

We define $\{0,1\}^*$ to be the set of *payloads*. Payloads will typically be identifiers of protocol parties, which is why we often refer to this set by ID. By $\mathsf{ek}(A)$, $\mathsf{dk}(A)$, $\mathsf{sk}(A)$, and $\mathsf{vk}(A)$ we denote the encryption, decryption, signing, and verification key of party $A \in \mathsf{ID}$, respectively. Let Nonce be a set of nonces (random strings). Now, the set M of (Dolev-Yao) messages is defined by the following grammar:

$$\mathsf{M} ::= \mathsf{ID} \mid \langle \mathsf{M}, \mathsf{M} \rangle \mid \mathsf{Nonce} \mid \mathsf{E}_{\mathsf{ek}(\mathsf{ID})}(\mathsf{M}) \mid \mathsf{Sig}_{\mathsf{vk}(\mathsf{ID})}(\mathsf{M}). \tag{1}$$

Given a set φ of messages, additional messages can be derived from φ according to the following rules.

– *Initial knowledge:* $\varphi \vdash m$ for all $m \in \varphi$,
– *Pairing and unpairing:* If $\varphi \vdash m_1$ and $\varphi \vdash m_2$, then $\varphi \vdash \langle m_1, m_2 \rangle$; conversely, if $\varphi \vdash \langle m_1, m_2 \rangle$, then $\varphi \vdash m_1$ and $\varphi \vdash m_2$.
– *Encryption and decryption:* If $\varphi \vdash \mathsf{ek}(b)$ and $\varphi \vdash m$, then $\varphi \vdash \mathsf{E}_{\mathsf{ek}(b)}(m)$ for all $b \in \mathsf{ID}$; conversely, if $\varphi \vdash \mathsf{E}_{\mathsf{ek}(b)}(m)$ and $\varphi \vdash \mathsf{dk}(b)$, then $\varphi \vdash m$ for all $b \in \mathsf{ID}$.
– *Encryption-key retrieval:* If $\varphi \vdash \mathsf{E}_{\mathsf{ek}(b)}(m)$, then $\varphi \vdash \mathsf{ek}(b)$ for all $b \in \mathsf{ID}$.
– *Signature:* If $\varphi \vdash \mathsf{sk}(b)$ and $\varphi \vdash m$, then $\varphi \vdash \mathsf{Sig}_{\mathsf{vk}(b)}(m)$ for all $b \in \mathsf{ID}$.
– *Plaintext retrieval:* If $\varphi \vdash \mathsf{Sig}_{\mathsf{vk}(b)}(m)$, then $\varphi \vdash m$ for all $b \in \mathsf{ID}$.
– *Verification-key retrieval:* If $\varphi \vdash \mathsf{Sig}_{\mathsf{vk}(b)}(m)$, then $\varphi \vdash \mathsf{vk}(b)$ for all $b \in \mathsf{ID}$.

2.2 Syntax of Protocols

A k-party protocol is defined by k roles, where a role specifies the behavior of a party in a protocol run. Defining roles requires to first introduce variables. We assume disjoint sets of typed variables $\mathsf{X}.n$ for nonces and $\mathsf{X}.d$ for payloads.

The ith *role*, $i = 1, \ldots k$, is defined to be a directed, edge-labeled finite tree where the edges originating in the same node are linearly ordered. Each edge is labeled with a rule (l, r) for terms l and r, where terms are messages which may contain variables. The left-hand side l of a rule serves as a pattern for received messages; these messages are matched against the pattern and the pattern's variables are instantiated accordingly. The right-hand side r of a rule specifies the response message. We use certain distinguished variables $A_1, \ldots, A_k \in \mathsf{X}.d$ and $N_j \in \mathsf{X}.n$ for $j \geq 0$. When the ith role is instantiated with parties $a_1, \ldots, a_k$, then A_j is substituted by a_j for every $j = 1, \ldots, k$, and fresh nonces are generated for the variables N_j occurring in the role. An instance of the ith role is carried out by party a_i.

Similar to [14], we put syntactic restrictions on the kind of terms that can occur on the left-hand side and right-hand side of rules to ensure that the corresponding role is executable and hence can be given a meaningful computational interpretation. For the ith role of a protocol, terms on the left-hand side of a rule are of the following form:

$$\mathsf{T}_i^l ::= \mathsf{ID} \mid \mathsf{X}.n \mid \mathsf{X}.d \mid \langle \mathsf{T}_i^l, \mathsf{T}_i^l \rangle \mid \mathsf{E}_{\mathsf{ek}(A_i)}(\mathsf{T}_i^l) \mid \mathsf{Sig}_{\mathsf{vk}(A)}(\mathsf{T}_i^l),$$

where $A \in \{A_1, \ldots, A_k\}$. Here $\mathsf{E}_{\mathsf{ek}(A_i)}(t)$ intuitively means that the party A_i (carrying out the ith role) decrypts the received message with $\mathsf{dk}(A_i)$ and then parses the plaintext according to t. Since A_i only knows its own decryption key $\mathsf{dk}(A_i)$, terms of the form $\mathsf{E}_{\mathsf{ek}(A_j)}(t)$ for $j \neq i$ are excluded since they correspond to decryptions with secret keys unknown to A_i. We however allow A_i to check the validity of the signatures of all other parties since their respective verification keys are considered public, i.e., A_i is assumed to know $\mathsf{vk}(A_j)$ for all j. A more comprehensive set of terms T_i^l is conceivable, e.g., by including terms that contain specific ciphertexts, variables for encryption/verification keys, or variables for ciphertexts in order to model ciphertext forwarding. While our results can be lifted to these cases, we concentrate on T_i^l as to not encumber our main ideas with details that are of only minor importance in this paper. For the ith role of a protocol, terms on the right-hand side of a rule are of the following form:

$$\mathsf{T}_i^r ::= \mathsf{ID} \mid \mathsf{X}.n \mid \mathsf{X}.d \mid \langle \mathsf{T}_i^r, \mathsf{T}_i^r \rangle \mid \mathsf{E}_{\mathsf{ek}(A)}(\mathsf{T}_i^r) \mid \mathsf{Sig}_{\mathsf{vk}(A_i)}(\mathsf{T}_i^r),$$

where $A \in \{A_1, \ldots, A_k\}$. A term $\mathsf{E}_{\mathsf{ek}(A_j)}(t)$ means that party A_i computes a bit string b for t and then encrypts b with the public key of A_j; $\mathsf{Sig}_{\mathsf{vk}(A_i)}(t)$ has a similar meaning. We

require that variables on the right-hand side of a rule belong to $\{A_1, \ldots, A_k\} \cup \{N_j \mid j \geq 0\}$, or occur on the left-hand side of the rule, or occur on the left-hand side of a preceding rule in a role to ensure that these variables have been instantiated by the time they are used. Several extensions of T_i^l are conceivable but not considered here for reasons of clarity.

Finally, let Roles denote the set of all roles. Then, a *k-party protocol* is a mapping $\Pi \colon \{1, \ldots, k\} \to \mathsf{Roles}$.

3 Simulatability and Requirements for Simulatability-sound Dolev-Yao Models

In this section, we review the notion of simulatability and point out necessary requirements for a Dolev-Yao model to be sound in the sense of BRSIM/UC.

3.1 Review of Simulatability

Simulatability constitutes a general approach for comparing two systems, typically called real and ideal system. In terms of the semantics community one might speak of an implementation or refinement relation, specifically geared towards the preservation of what one might call secrecy properties compared with functional properties. We believe that all our following results are independent of the differences between the definition styles of the various recent papers on simulatability [20, 21, 11, 10, 17]. However, we have to fix a specific formalism, and we use that from [21, 10].

The ideal system in [21, 10] typically consists of a single machine TH, the *trusted host*, see Figure 1. In the context of simulatability soundness, TH represents a Dolev-Yao model. The real system consists of a set of machines M_u, one for every user u. In the context of simulatability soundness, the real system describes the cryptographic implementation. The ideal or real system interacts with arbitrary so-called *honest users*, collectively represented by a single machine H; this corresponds to potential protocols or human users interacting with the ideal or real system. Furthermore, the ideal or real system interacts with an adversary A, who is often given more power than the honest users; in particular in real systems A typically controls the network and can manipulate messages on the bit string level. The adversary is also granted the ability to interact with the honest users H in order to influence their behavior, e.g., to suggest which messages are to be sent. Technically, the interaction with H models known-message and chosen-message attacks.

Black-box reactive simulatability (BRSIM) states that there exists a simulator Sim such that for all A, no H can distinguish (in the sense of computational indistinguishability of families of random variables [23]) if it interacts with the real system and the real adversary, or with the ideal system and a combination of the real adversary and the simulator (which together form the ideal adversary). This is depicted in Figure 1. Indistinguishability in particular entails that the ideal and real system offer identical interfaces to the honest users to prevent trivial distinguishability. We write $\mathsf{M}_1 \parallel \cdots \parallel \mathsf{M}_k \leq^{BRSIM} \mathsf{TH}$ to denote that the real system $\mathsf{M}_1 \parallel \cdots \parallel \mathsf{M}_k$ is *as secure as* the ideal system TH *in the sense of BRSIM/UC*.

The reader may regard the machines, i.e., the individual boxes in Figure 1, as probabilistic I/O automata, Turing machines, CSP or pi-calculus processes etc. The only requirement on the underlying system model is that the notion of an execution of a system when run together with an honest user and an adversary is well-defined. In [21, 10], the machines are a type of probabilistic I/O automata. We always assume that all parties are polynomial-time.

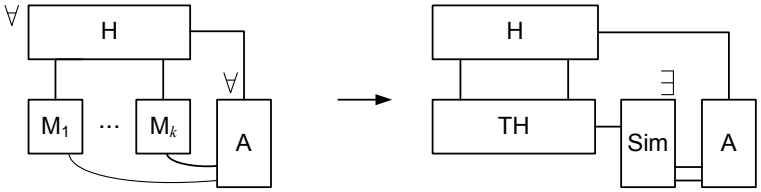

Fig. 1. Black-box reactive simulatability (BRSIM) between the real system $\mathsf{M}_1 \parallel \cdots \parallel \mathsf{M}_k$ and the ideal system TH, where M_u is the machine of user $u \in \{1, \ldots, k\}$.

3.2 On Simulatability-sound Dolev-Yao Models and their Cryptographic Implementations

We now outline necessary requirements a Dolev-Yao model $\mathsf{M}^{\mathsf{ideal}}$ offering the capabilities described in Section 2 and an implementation $\mathsf{M}^{\mathsf{real}} = \mathsf{M}_1^{\mathsf{real}} \parallel \cdots \parallel \mathsf{M}_k^{\mathsf{real}}$ realized by actual cryptographic primitives have to fulfill for being simulatability-sound. Solely fixing minimal requirements expected from Dolev-Yao models instead of considering a specific Dolev-Yao model frees our results from specific details and idiosyncrasies of existing models.

For achieving simulatability, the Dolev-Yao model $\mathsf{M}^{\mathsf{ideal}}$ and its cryptographic implementation $\mathsf{M}^{\mathsf{real}}$ have to offer an identical I/O interface which the honest users connect to. We hence assume that the interaction at the I/O interface is based on handles (pointers) to objects stored in the system, i.e., the user never obtains real bit strings (nonces, ciphertexts, etc.) from the cryptographic implementation but only handles to such objects. The only exception are payloads which obviously have to be retrievable in their bit string representation in some way. Note that we do not fix any specific instantiation of these handles but we only assume that they can be operated on in the expected manner as discussed below.

The I/O interface has to permit suitable commands for constructing terms according to the Dolev-Yao style deduction rules given in Section 2, and for sending them to other principals. This in particular comprises the generation of nonces, pairs of messages (i.e., concatenations of messages), pairs of public and private keys, to perform public-key encryption/decryption, to generate and verify signatures, to retrieve payloads from their handles, and to send and receive messages to/from the network. Moreover, there have to exist commands for parsing handles, in particular for testing handles for equality (for simplicity, we assume that each user u is deterministically given the same handle again if a term is reused), and for querying the types of handles.

Concerning the network interface, $\mathsf{M}^{\mathsf{ideal}}$ and $\mathsf{M}^{\mathsf{real}}$ differ. The network interface of $\mathsf{M}^{\mathsf{ideal}}$ offers the adversary commands for constructing and parsing terms according to the Dolev-Yao style deduction rules, and for sending terms to users. The machines $\mathsf{M}_u^{\mathsf{real}}$ output bit strings to and receive bit strings from the adversary at their network interfaces.

Note that we did not describe the internal behavior of the Dolev-Yao model $\mathsf{M}^{\mathsf{ideal}}$. It turns out not to be relevant for achieving our results, but we later only have to require two properties of $\mathsf{M}^{\mathsf{ideal}}$: First, $\mathsf{M}^{\mathsf{real}}$ is as secure as $\mathsf{M}^{\mathsf{ideal}}$ in the sense of BRSIM/UC; second, the behavior of $\mathsf{M}^{\mathsf{ideal}}$ in fact ensures that the adversary can only manipulate messages according to the Dolev-Yao rules presented in Section 2. More precisely, the second property requires that when $\mathsf{M}^{\mathsf{ideal}}$ is run with arbitrary honest users and an arbitrary adversary, the resulting protocol traces are so-called Dolev-Yao traces, which are formally defined in Section 4.

4 Reactive Execution of Protocols

We now describe the execution of a k-party protocol Π along with an adversary who controls the network. More precisely, we describe the *concrete* execution of Π, i.e., the execution in which actual cryptographic algorithms are used, rather than their Dolev-Yao abstractions. Our definition corresponds to the one for mapping approaches [19, 15, 14]. However, we present the definition in the reactive simulatability framework [21, 10] using $\mathsf{M}^{\mathsf{real}}$ in order to facilitate the presentation of our main result (Section 5).

4.1 Emulating Concrete Executions via H_{Π}

We use an honest user machine H_{Π} to emulate the execution of Π. This machine makes use of $\mathsf{M}^{\mathsf{real}}$ to carry out the necessary cryptographic operations. Recall that $\mathsf{M}^{\mathsf{real}}$ uses actual cryptographic algorithms to perform the cryptographic operations and that handles are used at its I/O interface to point to the bit strings (payloads, ciphertexts, nonces etc.) stored in $\mathsf{M}^{\mathsf{real}}$. While $\mathsf{M}^{\mathsf{real}}$ is a composition of machines $\mathsf{M}_u^{\mathsf{real}}$, $u \in \{1, \ldots, k\}$, H_{Π} can emulate the execution of instances of Π by only using one $\mathsf{M}_u^{\mathsf{real}}$ since within this machine key pairs for every party can be generated. This is even more general than using a separate machine for each party since it allows to model that the adversary dynamically generates new parties. We emphasize that the communication between the parties is still carried out over the network, so by using just one machine $\mathsf{M}_u^{\mathsf{real}}$ we do not introduce any idealization. As usual, the network is controlled by the adversary A. The adversary can instruct H_{Π} to generate a new instance of a role $\Pi(i)$ of Π. Before the execution of Π starts, A can additionally corrupt parties; this corresponds to the prevalent static corruption model of Dolev-Yao models. Altogether, the run of the system $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$ corresponds to a concrete execution of instances of Π.

State of H_{Π}. It remains to describe H_{Π}, i.e., the way H_{Π} emulates instances of Π. Similar to the definition of concrete executions in mapping approaches, the machine H_{Π} keeps a global state to remember which instances of Π are running and in which local state these instances are. The *global state* is a tuple $(\mathsf{SId}, f, \varphi)$, where (i) SId is a finite set of session IDs, (ii) φ keeps track of the knowledge of the adversary at the current point in time, and (iii) f maps every session identifier sid in SId to the current (local) state $f(sid) = (i, \nu, p, (a_1, \ldots, a_k))$ of that session, see below, where a session is an instance of one role of the protocol. A *local state* is a tuple $(i, \nu, p, (a_1, \ldots, a_k))$ with the following components: $i \in \{1, \ldots, k\}$ is the index of the role $\Pi(i)$ that is executed in this session, ν is a substitution that maps those variables in $\Pi(i)$ that were bound in the matching processes so far to handles (pointing to bit strings stored in $\mathsf{M}^{\mathsf{real}}$), p is a node in the role $\Pi(i)$ marking the current point in the execution of $\Pi(i)$, and $(a_1, \ldots, a_k)$ are the parties participating in this session. Recall that the session is carried out by a_i with the parties a_j, $j \in \{1, \ldots, k\} \setminus \{i\}$. The *initial global state* is $(\emptyset, \emptyset, \emptyset)$. The machine H_{Π} additionally keeps a table in which it remembers handles to the names of honest and dishonest parties along with their encryption/decryption/signing/verification keys (in case of honest parties) and encryption/verification keys (in case of dishonest parties). It also keeps a set of known handles to payloads and nonces. The table and the set are updated in the obvious way; we will not further describe it but simply assume that H_{Π} knows the names and keys of all honest and dishonest parties as well as the (handles of) payloads and nonces which occurred so far in the protocol run.

Transitions of H_{Π}. We now describe how global states evolve in H_{Π} in terms of transitions. We often do not distinguish between payloads and their handles in the following, since we assumed that payloads can be efficiently retrieved from $\mathsf{M}^{\mathsf{real}}$ using their handles. In particular, we do not distinguish between the name of a party (represented as payload data) and the handle to this name.

Corrupt message (from A *via* $\mathsf{M}_u^{\mathsf{real}}$*):* Following the prevalent static corruption model of Dolev-Yao models, the adversary can corrupt parties only at the beginning of a protocol execution. This is captured by the adversary sending a message (a bit string) of the form $(\mathsf{corrupt}, a_1, \ldots, a_l, g_1, \ldots, g_l, h_1, \ldots, h_l)$ for $l \geq 0$ to $\mathsf{M}^{\mathsf{real}}$ where a_i are names of parties, and g_i and h_i are their encryption and verification keys, respectively, provided by A. This corruption message is forwarded by $\mathsf{M}^{\mathsf{real}}$ in terms of a handle to H_{Π} which then tests if all a_i are payloads (interpreted as names of parties), all g_i are handle to encryption keys and all h_i are handles to verification keys. Otherwise, the execution is aborted. Now, the knowledge of the adversary is recorded by H_{Π} as $\varphi' := \{a_i, \mathsf{ek}(a_i), \mathsf{dk}(a_i), \mathsf{sk}(a_i), \mathsf{vk}(a_i) \mid 1 \leq i \leq l\}$, and H_{Π} changes its (initial) global state as follows:

$$(\emptyset, \emptyset, \emptyset) \xrightarrow{(\mathsf{corrupt}, a_1, \ldots, a_l, g_1, \ldots, g_l, h_1, \ldots, h_l)} (\emptyset, \emptyset, \varphi').$$

Initiate new session (from A *via* $\mathsf{M}_u^{\mathsf{real}}$*):* The adversary can initiate a new session at any time by sending a message of the form $(\mathsf{new}, i, a_1, \ldots, a_l)$ for $l \geq 0$ where a_i are names of parties and $i \in \{1, \ldots, l\}$. This message is forwarded by $\mathsf{M}^{\mathsf{real}}$ in terms of a handle to H_{Π} which then tests if all a_i are payloads (interpreted as names of parties) and $i \in \{1, \ldots, l\}$, aborting at failure. Let $(\mathsf{SId}, f, \varphi)$ denote the current global state of H_{Π}. Let $sid := |\mathsf{SId}| + 1$ be the new session identifier and $\mathsf{SId}' := \mathsf{SId} \cup \{sid\}$. M_{Π} uses $\mathsf{M}^{\mathsf{real}}$ to create new encryption and signature pairs $\mathsf{ek}(a_i), \mathsf{dk}(a_i), \mathsf{sk}(a_i), \mathsf{vk}(a_i)$ for all honest parties a_i that do not yet have such pairs, to create new nonces for all variables N_j occurring in $\Pi(i)$, and to create a handle to the payload sid. Let the function f' on SId' be defined by $f'(sid') := f(sid')$ for each $sid' \in \mathsf{SId}$, and $f'(sid) := (i, \nu, \varepsilon, (a_1, \ldots, a_k))$, where ε is the root of the role tree, and where ν maps every A_j in $\Pi(i)$ to a_j and every N_j occurring in $\Pi(i)$ to the handle of the corresponding nonce. Let $\varphi' := \varphi \cup \{sid\} \cup \{\mathsf{ek}(a_j), \mathsf{vk}(a_j) \mid j = 1, \ldots, l\}$. Then M_{Π} changes its global state as follows:

$$(\mathsf{SId}, f, \varphi) \xrightarrow{(\mathsf{new}, i, a_1, \ldots, a_l)} (\mathsf{SId}', f', \varphi').$$

Finally, M_{Π} uses $\mathsf{M}^{\mathsf{real}}$ to create a list containing sid and the created encryption and verification keys and to send this list to the adversary.

Send message (from A *via* $\mathsf{M}_u^{\mathsf{real}}$*):* The adversary can at any time transmit a message m by sending a message of the form (send, sid, m). This message is forwarded by $\mathsf{M}^{\mathsf{real}}$ in terms of a handle to H_{Π}. Let $(\mathsf{SId}, f, \varphi)$ denote the current global state, $f(sid) = (i, \nu, p, (a_1, \ldots, a_k))$, and let $(l_1, r_1), \ldots, (l_h, r_h)$ be the labels of the outgoing edges of node p in the given order. Then H_{Π} parses m (see below) according to $\nu(l_j)$ starting with $\nu(l_1)$, then continuing with $\nu(l_2)$, and so on, until the first parsing can be successfully completed. If parsing fails for every $\nu(l_j)$, the local and global state remain unchanged.

The parsing of m according to $l := \nu(l_j)$ is performed by H_{Π} inductively on the structure of l. The parsing updates ν since variables that are not in the domain of the current ν

so far may now be instantiated. H_{Π} furthermore keeps track of new payloads and nonces created by the adversary by maintaining a set φ_{new} which at the beginning of the parsing is defined to be empty. Now, the parsing is performed by H_{Π} as follows: First, it checks if m and l have the same type (by querying $\mathsf{M}^{\mathsf{real}}$ for the type of m and then checking if it corresponds to the one of l), aborting at failure. Otherwise H_{Π} continues as follows: (i) If l is a handle to a payload or a nonce, then it checks if $l = m$. (Note that the same payloads/nonces get the same handles in $\mathsf{M}^{\mathsf{real}}$. Here we use that H_{Π} only employs one machine $\mathsf{M}^{\mathsf{real}}_u$ for some $u \in \{1, \ldots, k\}$. We emphasize that this is not an idealization since checking bit strings or corresponding handles for equality is equivalent.) (ii) If $l \in \{0,1\}^*$ is a payload, then it retrieves the payload of m and checks whether it coincides with l. (iii) If $l \in \mathsf{X}.n$ ($l \in \mathsf{X}.d$), then it checks whether m is a handle to a nonce (to payload data), aborting at failure. Otherwise, it extends ν by mapping l to m. If m has not occurred before (i.e., m is a handle to a new payload or nonce that the adversary generated), then it adds m to φ_{new}. (iv) If $l \in \mathsf{X}.d$, then it checks whether m is a handle to a payload and continues as in the previous case. (v) If $l = \langle t_1, t_2 \rangle$, then it recursively parses the first component of m according to t_1 and ν, and then the second component according to t_2 and (the possibly updated) ν. (vi) If $l = \mathsf{E}_{\mathsf{ek}(a_i)}(t)$, then it decrypts m with $\mathsf{dk}(a_i)$, aborting at failure. Otherwise, it parses the resulting plaintext (given as a handle) according to t. If l is a signature, it proceeds analogously.

If the parsing of m according to l is successful, we say that m and l match and call the resulting substitution (the updated ν) the matching function. We call $\nu(l)$ the Dolev-Yao term corresponding to m. In what follows, let h be minimal such that m matches with $\nu(l_h)$ and let θ be the resulting matching function.

Next, H_{Π} uses $\mathsf{M}^{\mathsf{real}}$ to construct the output message according to $r := \theta(r_h)$. The result is a handle to this message in $\mathsf{M}^{\mathsf{real}}$. The construction is carried out inductively on the structure of r as follows (Note that r does not contain variables since all variables are substituted with handles by θ): (i) If r is a handle, then it returns this handle. (ii) If $r \in \{0,1\}^*$ is a payload, then it creates a handle to this payload and returns this handle. (iii) If r is a pair, then it recursively constructs messages for the two components. With the resulting handles, it retrieves a handle to the pair from $\mathsf{M}^{\mathsf{real}}$. (iv) If $r = \mathsf{E}_{\mathsf{ek}(a_j)}(t)$, then it recursively constructs a message for t. With the resulting handle and the handle to $\mathsf{ek}(a_j)$, it retrieves a handle from $\mathsf{M}^{\mathsf{real}}$ to the corresponding ciphertext and returns this handle. If $r = \mathsf{Sig}_{\mathsf{vk}(a_i)}(t)$, then it proceeds analogously, using the handle to $\mathsf{sk}(a_i)$ to produce the signature.

Let m'^{hnd} denote the handle to the output message. Let f' be defined as $f'(sid') := f(sid')$ for every $sid' \in \mathsf{SId} \setminus \{sid\}$ and $f'(sid) = (i, \theta, ph, (a_1, \ldots, a_k))$ where ph is the hth successor of p in $\Pi(i)$. Let $\varphi' = \varphi \cup \varphi_{new} \cup \{r\}$. Then H_{Π} changes its global state as follows:

$$(\mathsf{SId}, f, \varphi) \xrightarrow{(\mathsf{send}, sid, m)} (\mathsf{SId}, f', \varphi').$$

Finally, H_{Π} sends the message corresponding to m'^{hnd} to the adversary.

4.2 (Dolev-Yao) Traces of Π

We now define traces of Π when executed with an adversary A.

Definition 1 (Traces). *A* trace *of Π when executed with an adversary A is a sequence $g_0 \xrightarrow{C_1} g_1 \xrightarrow{C_2} g_2 \xrightarrow{C_3} \cdots \xrightarrow{C_n} g_n$ of transitions $g_i \xrightarrow{C_{i+1}} g_{i+1}$ as defined above for H_{Π}*

obtained by executing the system $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$. *The* C_i *are the corrupt, new, and send commands and* $g_0 = (\emptyset, \emptyset, \emptyset)$ *is the initial global state. A send transition only belongs to the trace if* H_{Π} *successfully parsed the input message.*

A trace is called a Dolev-Yao trace if it can be constructed according to the rules of the term algebra and of the protocol Π under consideration (see Section 2).

Definition 2 (Dolev-Yao Traces). *A trace* $(\mathsf{Sld}_0, f_0, \varphi_0) \xrightarrow{C_1} (\mathsf{Sld}_1, f_1, \varphi_1) \xrightarrow{C_2} \cdots \xrightarrow{C_r} (\mathsf{Sld}_r, f_r, \varphi_r)$ *of* Π *when executed with an adversary* A *is called a* Dolev-Yao trace *if and only if the following holds: For all* i *such that* C_i *is of the form* $(\mathsf{send}, sid_i, m_i)$ *we have that* $\varphi_{i-1} \cup (\varphi_{i-1})_{new} \vdash t_{m_i}$ *where* t_{m_i} *is the Dolev-Yao term corresponding to* m_i *and* $(\varphi_{i-1})_{new}$ *contains the new constants in* t_{m_i} *generated by the adversary.*

5 Simulatability Soundness implies Mapping Soundness

In this section we show that mapping soundness is implied by simulatability soundness, i.e., by results that prove cryptographic implementations as secure as Dolev-Yao style abstractions in the sense of BRSIM/UC.

Recall that mapping soundness is established in the following style: One defines *concrete protocol traces* where several instances of the protocol run along with a probabilistic polynomial-time adversary that controls the network. Messages are bit strings and the cryptographic operations are carried out by cryptographic algorithms. This corresponds to runs of the system $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$. In addition, one defines *symbolic protocol traces* where messages are Dolev-Yao terms. Now one aims at constructing a mapping from bit strings to terms such that applying the mapping to an arbitrary trace of the concrete cryptographic execution yields a Dolev-Yao trace: Different payloads and nonces are mapped to different constants, encryption/decryption/verification/signing keys are represented by $\mathsf{ek}(a)$, $\mathsf{dk}(a)$, $\mathsf{vk}(a)$, and $\mathsf{sk}(a)$ where a is the constant representing the name of a party. Pairings, ciphertexts, and signatures are represented by the corresponding Dolev-Yao terms. Given such a mapping, one shows that the resulting symbolic protocol trace constitutes a Dolev-Yao trace up to a negligible probability (measured in the implicit cryptographic security parameter).

Before we can state and prove our result, let us make the following observation about $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$. On the one hand, this system describes concrete protocol executions: The different instances of the protocol exchange cryptographic bit strings over the network, which is fully controlled by the probabilistic polynomial-time adversary. On the other hand, $\mathsf{M}^{\mathsf{real}}$ provides an abstract interface to H_{Π} in the sense that H_{Π} does not obtain bit strings from $\mathsf{M}^{\mathsf{real}}$ (except for payloads), but only abstract representations (handles) to the bit strings stored in $\mathsf{M}^{\mathsf{real}}$. Hence $\mathsf{M}^{\mathsf{real}}$ already realizes the desired mapping from bit strings to handles, and these handles one-to-one correspond to Dolev-Yao terms in the natural manner. (A handle to a payload/nonce corresponds to a constant representing this payload/nonce; a handle to an encryption/decryption/verification/signing key of a party a corresponds to the ground term $\mathsf{ek}(a)$, $\mathsf{dk}(a)$, $\mathsf{vk}(a)$, and $\mathsf{sk}(a)$, respectively; handles to pairs, ciphertexts, and signatures correspond to Dolev-Yao terms representing these objects.) Since all handles are maintained in one machine $\mathsf{M}^{\mathsf{real}}_u$ for some $u \in \{1, \ldots, k\}$, different payloads/nonces/etc. are referred to by different handles. Hence, the mapping from bit strings to Dolev-Yao terms

– and consequently the translation of concrete protocol traces to symbolic traces – is implicitly already performed by $\mathsf{M}^{\mathsf{real}}$, hence freeing us from explicitly defining it. This might be surprising since a natural intuition suggests that this translation is encompassed by the simulator.

While $\mathsf{M}^{\mathsf{real}}$ implicitly provides a mapping from concrete traces to symbolic traces, this does not necessarily entail that the latter trace is a Dolev-Yao trace. Our main result now states that simulatability soundness implies that the symbolic trace derived from this mapping constitutes a Dolev-Yao trace up to a negligible probability, which is exactly what mapping soundness intends to establish. More precisely, the result relies on two assumptions: First, we have that $\mathsf{M}^{\mathsf{real}} \leq^{BRSIM} \mathsf{M}^{\mathsf{ideal}}$, i.e., the cryptographic implementation is as secure as the Dolev-Yao abstraction in the sense of BRSIM/UC. Second, if protocols are executed based on $\mathsf{M}^{\mathsf{ideal}}$ instead of $\mathsf{M}^{\mathsf{real}}$, then the resulting traces are Dolev-Yao traces, i.e., for every ideal adversary A' (which may be a composition of a simulator and a real adversary) all traces of $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{ideal}} \parallel \mathsf{A}'$ are Dolev-Yao traces (which reflects the intuition and purpose behind the Dolev-Yao abstraction $\mathsf{M}^{\mathsf{ideal}}$).

Theorem 1 (Simulatability Soundness implies Mapping Soundness). *Let Π be a protocol. Assume the following two properties about $\mathsf{M}^{\mathsf{real}}$ and $\mathsf{M}^{\mathsf{ideal}}$:*

1. $\mathsf{M}^{\mathsf{real}} \leq^{BRSIM} \mathsf{M}^{\mathsf{ideal}}$.
2. *For every ideal adversary A', all traces of $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{ideal}} \parallel \mathsf{A}'$ are Dolev-Yao traces.*

Then, for all (real) adversaries A, the probability that a trace of $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$ is not a Dolev-Yao trace is negligible.

Proof (Sketch). By construction, H'_{Π} behaves exactly as H_{Π} except that it checks whether each received messages can be deduced by the current intruder knowledge plus the new handles (corresponding to payloads and nonces generated by the adversary) in the received message. If this is not the case, then H'_{Π} outputs $\mathsf{failure}$. Since $\vdash$ can be decided in polynomial time (see, e.g., [13]), H'_{Π} runs in polynomial time. Furthermore, the definition of Dolev-Yao traces implies that the probability that a trace of $\mathsf{H}_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$ is not a Dolev-Yao trace is exactly the probability that H'_{Π} outputs $\mathsf{failure}$ in a run of $\mathsf{H}'_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$.

By the first assumption in the theorem there exists a simulator S such that for every A the view of H'_{Π} in $\mathsf{H}'_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$ and $\mathsf{H}'_{\Pi} \parallel \mathsf{M}^{\mathsf{ideal}} \parallel S \parallel \mathsf{A}$ is indistinguishable. We consider the ideal adversary $\mathsf{A}' = S \parallel \mathsf{A}$. By the second assumption in the theorem we can conclude that H'_{Π} never outputs $\mathsf{failure}$ in a run of $\mathsf{H}'_{\Pi} \parallel \mathsf{M}^{\mathsf{ideal}} \parallel \mathsf{A}'$. Finally, it follows that the probability that H'_{Π} outputs failure in a run $\mathsf{H}'_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$ is negligible as otherwise the views of H'_{Π} in $\mathsf{H}'_{\Pi} \parallel \mathsf{M}^{\mathsf{real}} \parallel \mathsf{A}$ and $\mathsf{H}'_{\Pi} \parallel \mathsf{M}^{\mathsf{ideal}} \parallel \mathsf{A}'$ could be distinguished. □

A more detailed proof can be found in the long version of this paper. We emphasize that the argument is quite generic: The proof only exploits that H_{Π} can be extended so that it is able to efficiently recognize Dolev-Yao traces. Moreover, only the definition of H_{Π} and the extension of H_{Π} depend on the specific cryptographic primitives and the class of protocols under consideration. The rest of the argument is independent of these details, and it resembles property preservation theorems for simulatability [5]. Therefore, the above theorem should also hold for larger classes of cryptographic primitives and protocols. We conclude by pointing out that the two assumptions in Theorem 1 are met by the concrete cryptographic implementation and its Dolev-Yao abstraction put forward in [8].

References

1. M. Abadi and P. Rogaway. Reconciling two views of cryptography: The computational soundness of formal encryption. In *Proc. 1st IFIP TCS*, vol. 1872 of *LNCS*, pp. 3–22. Springer, 2000.
2. M. Backes, I. Cervesato, A. D. Jaggard, A. Scedrov, and J.-K. Tsay. Cryptographically sound security proofs for basic and public-key Kerberos. In *Proc. 11th ESORICS*, vol. 4189 of *LNCS*, pp. 362–383. Springer, 2006.
3. M. Backes and M. Dürmuth. A cryptographically sound Dolev-Yao style security proof of an electronic payment system. In *Proc. 18th IEEE CSFW*, pp. 78–93, 2005.
4. M. Backes, M. Dürmuth, and R. Küsters. On simulatability soundness and mapping soundness of symbolic cryptography. IACR Cryptology ePrint Archive 2007/233, 2007.
5. M. Backes and C. Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In *Proc. 20th STACS*, vol. 2607 of *LNCS*, pp. 675–686. Springer, 2003.
6. M. Backes and B. Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. *IEEE Journal on Selected Areas in Communications*, 22(10):2075–2086, 2004.
7. M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE CSFW*, pp. 204–218, 2004.
8. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proc. 10th ACM CCS*, pp. 220–230, 2003.
9. M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. In *Proc. 8th ESORICS*, vol. 2808 of *LNCS*, pp. 271–290. Springer, 2003.
10. M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability framework for asynchronous systems. *Information and Computation*, 2007. Preprint on IACR ePrint 2004/082.
11. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE FOCS*, pp. 136–145, 2001.
12. R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In *Proc. 3rd TCC*, vol. 3876 of *LNCS*, pp. 380–403. Springer, 2006.
13. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *Proc. 18th IEEE LICS*, pp. 261–270, 2003.
14. V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In *Proc. 26th FSTTCS 2006*, vol. 4337 of *LNCS*, pp. 176–187. Springer, 2006.
15. V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th ESOP*, vol. 3444 of *LNCS*, pp. 157–171. Springer, 2005.
16. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
17. R. Kuesters. Simulation-based security with inexhaustible interactive turing machines. In *Proc. 19th IEEE CSFW*, pp. 309–320, 2006.
18. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. 25th IEEE SSP*, pp. 71–85, 2004.
19. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st TCC*, vol. 2951 of *LNCS*, pp. 133–151. Springer, 2004.
20. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM CCS*, pp. 245–254, 2000.
21. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE SSP*, pp. 184–200, 2001.
22. C. Sprenger, M. Backes, D. Basin, B. Pfitzmann, and M. Waidner. Cryptographically sound theorem proving. In *Proc. 19th IEEE CSFW*, pp. 153–166, 2006.
23. A. C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE FOCS*, pp. 80–91, 1982.