On Composable Security for Digital Signatures*

Christian Badertscher[1], Ueli Maurer[1], and Björn Tackmann[2,**]

[1] Department of Computer Science, ETH Zurich, 8092 Zürich, Switzerland
{badi, maurer}@inf.ethz.ch
[2] IBM Research – Zurich, 8803 Rüschlikon, Switzerland
bta@zurich.ibm.com

Abstract. A digital signature scheme (DSS), which consists of a key-generation, a signing, and a verification algorithm, is an invaluable tool in cryptography. The first and still most widely used security definition for a DSS, existential unforgeability under chosen-message attack, was introduced by Goldwasser, Micali, and Rivest in 1988.
As DSSs serve as a building block in numerous complex cryptographic protocols, a security definition that specifies the guarantees of a DSS under composition is needed. Canetti (FOCS 2001, CSFW 2004) as well as Backes, Pfitzmann, and Waidner (CCS 2003) have described ideal functionalities for signatures in their respective composable-security frameworks. While several variants of these functionalities exist, they all share that the verification key and signature values appear explicitly.
In this paper, we describe digital signature schemes from a different, more abstract perspective. Instead of modeling all aspects of a DSS in a monolithic ideal functionality, our approach characterizes a DSS as a construction of a functionality for authentically reading values written by a certain party from certain assumed functionalities, e.g., for transmitting verification key and signature values. This approach resolves several technical complications of previous simulation-based approaches, captures the security of signature schemes in an abstract way, and allows for modular proofs.
We show that our definition is equivalent to existential unforgeability. We then model two example applications: (1) the certification of values via a signature from a specific entity, which with public keys as values is the core functionality of public-key infrastructures, and (2) the authentication of a session between a client and a server with the help of a digitally signed assertion from an identity provider. Single-sign-on mechanisms such as SAML rely on the soundness of the latter approach.

1 Introduction

A digital signature scheme (DSS) allows a signer to authenticate a message such that everyone can verify the authenticity. The signer initially generates an asym-

* This is the full version of the article due to appear at PKC 2018. The final publication will be available at link.springer.com.

** Work partially done while author was at Department of Computer Science and Engineering, UC San Diego.

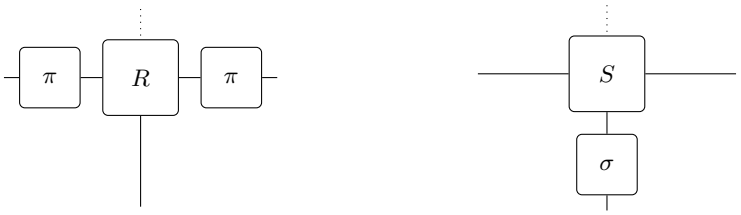

Fig. 1. Left: Execution of protocol π in the real-world model. **Right:** Ideal-world model described by S with simulator σ. In both figures, the dotted lines are "free" interfaces explained below.

metric key pair consisting of a *signing key* and a *verification key*. The signing key, which is kept secret by the signer, allows to generate signatures for messages. The verification key is made public and allows to verify that a message was indeed signed using the corresponding signing key. DSSs are a crucial component in many of today's widely-used cryptographic protocols. They underlie the public-key infrastructure (PKI) that is used to provide authentication in most Internet protocols, and they are used to authenticate e-mails as well as to provide non-repudiation for electronic documents. They are also used as a building block in numerous cryptographic protocols.

1.1 Our Approach

The core idea of our approach is that digitally signing a message can be understood as the signer's declaration that the message belongs to a certain context, which is described by the verification key. This context may be the signer's commitment to be legally liable for the content of the message (e.g., a contract), or simply that the message is meant to originate from the signer. Abstractly, this can be understood as writing the message to a certain type of *repository* that allows other parties to verify for given messages whether they have been written to the repository, i.e., assigned to the context.

The real-world/ideal-world paradigm. Many security definitions, and in particular most composable security frameworks [5,22,20], are based on the real-world/ideal-world paradigm. The real world models the use of a protocol, whereas the ideal world formalizes the security guarantees that the protocol is supposed to achieve. The structure of the real-world model is depicted for a simple setting in Fig. 1 on the left, where R describes the *assumed resources* [20] or *hybrid functionalities* [5] used by the protocol π. The "open lines" on the left and right indicate the interfaces that the honest parties use to access the protocol π, whereas the line on the bottom signifies that a potential attacker would access R directly.

In the ideal world, as depicted in Fig. 1 on the right, the box S formalizes the intended security guarantees and is referred to as *constructed resource* [20]

or *ideal functionality* [5]. The access of the honest parties to S is via direct interfaces, whereas a potential attacker accesses S via the so-called *simulator* σ. The intuition behind the simulator is that a *distinguisher* (sometimes called *environment*) is connected to all open interfaces of either of the two settings in Fig. 1, it interacts with them with the goal to distinguish. If the two settings are indistinguishable, then any attack on π can be translated via σ into an attack on S, but S is secure by definition. Therefore, π is at least as secure as S.

Signature schemes as constructions. We formalize the security of a DSS in the real-world/ideal-world paradigm and based on different types of repositories to which messages can be written and from which messages can be read, by different parties with potentially different access permissions. As described above, the goal of using the signature scheme in the described way can be seen as constructing an *authenticated* repository, where only the signer can write messages and all verifiers can check the validity. This repository takes the role of S in Fig. 1.

Using a signature scheme requires an authenticated repository that can hold one message. This repository is used to transmit the signature verification key. We also require one repository that can hold multiple messages, but this repository can be *insecure*, meaning that write access to the repository is not exclusive to the signer. This repository is used to transmit the signature strings. We also make the storage of the signing key explicit as a *secure* repository where both write and read access is exclusive to the signer. These three assumed repositories correspond to R in Fig. 1.

A signature scheme then uses the described repositories in the obvious way: the signer begins by generating a key pair, writes the signing key to the secure repository and the verification key to the authenticated one. Upon signing a message m, the signer retrieves the signing key from the secure repository, computes the signature, and stores it in the insecure repository. For checking the validity of a message m, a verifier reads the verification key from the authenticated repository and the signature from the insecure one, and runs the signature verification algorithm. Our security statement is, then, that this use of the signature scheme constructs the desired authenticated repository for multiple messages from the three described assumed repositories.

The advantage of such a composable security statement is that applications and higher-level protocols can be designed and analyzed using the abstraction of such a repository; in particular, no reduction proof is required since the composition theorem immediately guarantees the soundness of this approach.

Abstract communication semantics. The purpose of a repository is to model the fact that a certain message written by one party can be made accessible to a different party in an abstract manner. Indeed, a DSS is a generic security mechanism and can be used by various applications; the definition of a DSS should abstract from the particular way in which the verification key and the signature are delivered to the verifier. For instance, a communication network used for

transmission may guarantee messages to be delivered within a certain time, or an attacker may be able to eavesdrop on messages. Using a DSS—intuitively—preserves such properties of the communication network. The repositories used in this work are general enough to model various different such concrete types of transferring the values.

This generality is, more technically, achieved through a *free* interface that appears in both the real-world and the ideal-world model and that is indicated by the dotted lines in Figure 1. In the random experiment, this interface is accessed directly by the distinguisher. The free interface is reminiscent of the environment access to the global setup functionality in the GUC model [9], but in our model each resource/functionality can have such a free interface.[3]

A free interface allows the distinguisher to interact with both resources R and S directly. This results in a stronger and more general condition compared to considering the capabilities at that interface as part of the attacker's interface and, therefore, in the ideal-world model providing them to the simulator. More intuitively, the free interface can be seen as a way for the distinguisher to enforce that certain aspects in the real and the ideal world are the same. We will use the free interface to let the distinguisher control the transmission semantics; this leaves our statements general and independent of any concrete such semantics.

In more detail, the write and read interfaces of the repository are defined to write to or read from buffers associated to the interface. The repository also has free interfaces that control the transfer of messages from write buffers to read buffers. In other words, capabilities such as writing messages to a buffer in the repository or reading messages from one are separated from the mechanisms for making messages written to the repository visible at a specific reader interface. Control over the operations governing the visibility is granted to the environment—this makes the security statements independent of specific network models. In particular, the statements imply those in which these capabilities are granted to an attacker controlling the network.

Interfaces and partitioning of capabilities. The interfaces of a resource group capabilities. Often, each interface can be seen as corresponding to one particular party in a given application scenario, which can then attach a protocol machine to this interface, as in Fig. 1. Yet, for a general security definition such as that of a DSS we do not want to fix the number of possible verifiers in advance, or even prohibit that the signing key may be transmitted securely between and used by different parties. As one can always merge several interfaces and provide them to the same party, it is beneficial to target a fine-grained partitioning of capabilities into interfaces, and therefore a fine-grained partitioning of the protocol into individual protocol machines.

For our repositories, this means that if each interface gives access to one basic operation (such as writing or reading one value), one can always subsume

[3] The direct communication between the environment and the functionality requires a modification of the control function in UC, but does not affect the composition theorem. In most formal frameworks [22,16,20], no modification is necessary.

a certain subset of these capabilities into one interface and assign it to a single party. We achieve the most fine-grained partitioning by modeling each invocation of an algorithm of the signature scheme as a single protocol machine, and capture passing values between the machines explicitly via repositories.

Specifications. For generality or conciseness of description, it is often desirable to not fully specify a resource or functionality. For instance, a complete specification of the construction would entail the behavior of the signature scheme in the case where a signature shall be verified before the verification key is delivered to the verifier. The approach generally used in the literature on UC in such cases is to delegate such details to the adversary, to model the worst possible behavior. In this work, we follow a more direct approach, and explicitly leave the behavior undefined in such cases.

Our formalization follows the concept of *specifications* by Maurer and Renner [21], which can alternatively be seen either as an incomplete description of a resource, or as the set of all resources that adhere to the description. Maurer and Renner describe concrete types of specifications such as all resources that can be distinguished from a specific one by at most a certain advantage, or all resources that are obtained from a specific one by applying certain transformations.

We use specifications in this work to describe the behavior of a resource in environments that use the resource in a restricted way, in the sense that the inputs given to the resource satisfy certain conditions, such as that the verification key must have been delivered before messages can be verified. This alleviates the requirement of specifying the behavior of the resource for input patterns that do not occur in applications, and simplifies the description. Needless to say, this also means that for each application one has to show that the use of the resource indeed adheres to the specified conditions.

The repositories in this work. In summary, we consider specifications of repositories as described above. Repositories provide multiple interfaces, each of which allows exactly one write or read operation. A repository that allows for k write operations has k writer interfaces, and for n read operations it has n reader interfaces, and each operation can be understood as writing to or reading from one specific buffer. A write interface may allow the writer to input an arbitrary value from the message space, or it may allow the writer to only copy values from buffers at some read interfaces. A read interface may either allow to retrieve the contents of the corresponding buffer, or to input a value and check for equality with the one in the buffer.

The resource additionally provides free interfaces for transferring the contents of write buffers to read buffers. As discussed above, the access to these interfaces for managing the visibility of messages is given to the distinguisher, not the attacker, to abstract from specific communication semantics.

All repositories in this work can be viewed as specific instances of the one described above, where different types of capabilities are provided at different

parties' interfaces. For instance, a repository in which an attacker has only read-interfaces, but cannot write, can be considered as *authenticated*, since all messages must originate from the intended writers. A repository where the attacker can also write can be considered as *insecure*, since messages obtained by honest readers could originate either from honest writers or the attacker.

1.2 Background and Previous Work

The concept of a DSS was first envisioned by Diffie and Hellman and referred to as *one-way authentication* [13]. Early instantiations of this concept were given by Rivest, Shamir, and Adleman [23] and by Lamport [17]. The provable-security treatment of DSS was initiated by Goldwasser, Micali, and Rivest [14], who also introduced the first and still widely-used security definition called *existential unforgeability under chosen-message attack*. In this definition, a hypothetical attacker that has access to honestly computed signatures on messages of his own choice aims at creating a signature for some new message. A scheme is deemed secure if no efficient attacker can provide such a forgery with non-negligible probability.

Canetti [6] and independently Pfitzmann and Waidner [22] developed security frameworks that allow for security-preserving composition of cryptographic schemes. In these frameworks, the security of a cryptographic scheme, such as a DSS, is modeled by idealizing the algorithms and their security properties, and a concrete scheme is then proved to satisfy the idealization under certain computational assumptions. Higher-level schemes and protocols that make use of a DSS can be analyzed using the idealized version of the scheme. One main advantage of composable frameworks is that they guarantee the soundness of this approach; a higher-level protocol proven secure with respect to an idealized signature scheme will retain its security even if the idealized scheme is replaced by any concrete scheme that is proven secure. In contrast to standard reductionist proofs, this method does *not* require to prove an explicit reduction from breaking the signature scheme to breaking the higher-level protocol; this follows generically from the composition theorem. Still, even in protocol analyses within composable frameworks, existential unforgeability remains widely used, despite the existence of composable models within these formal frameworks.

The first composable notion for digital signatures has been proposed by Canetti [5,7] via an ideal signing functionality $\mathcal{F}_{\text{SIG}}$. The functionality idealizes the process of binding a message m to a public key vk via an ideal signature string s. In a nutshell, when the honest sender signs a message, he receives an idealized signature string. This signature string allows any party to verify that the message has indeed been signed by the signer. $\mathcal{F}_{\text{SIG}}$ enforces consistency and unforgeability in an ideal manner: if the honest signer has never signed a message m, no signature string leads to successful verification. Likewise, verification with a legitimately generated signature string for a message m always succeeds. Special care has to be taken in case the signer is dishonest, in which case the above guarantees for unforgeability are generally lost. The formalization given

by Backes, Pfitzmann, and Waidner [2] in their framework follows a by and large similar approach.

Several versions of the signature functionality have been suggested in previous work [5,10,1,7,8,11]. All these versions, however, require interaction with the ideal-model adversary for operations that correspond to local computations in any real-world scheme, such as the initial creation of the key pair or the generation of a signature. Camenisch *et al.* [4] point out that this unnatural weakness, allowing the adversary to delay operations in the idealized security guarantee, has often gone unnoticed and even lead to flaws in proofs of higher-level schemes based on signatures. As a further example, consider a signer S that has never signed a message m. If an honest party P verifies m with respect to some signature string s, the verification should fail. Yet, the adversary gets activated during any local verification request and can corrupt the signer just before providing the response. The adversary thus has complete freedom on whether to let P accept or reject the signature string s on message m. This behavior is arguably counter-intuitive and it is a property that signature schemes do not possess. The solution of Camenisch *et al.* [4] requires to modify the universal composability framework by introducing the concept of *responsive* environments and adversaries that are mandated to answer specific requests immediately to model local tasks. While Camenisch *et al.* do re-prove the composition theorem for their modified framework, such a modification of the framework has the downside of further increasing its complexity and, at least in principle, making security analyses in the original and modified frameworks incompatible.

Besides the technical difficulties in defining the signature functionality $\mathcal{F}_{\text{SIG}}$ consistently, it is less abstract than what one would expect, since the signature string and the verification key are an explicit part of the interface. Indeed, Canetti [7, page 5] writes:

> The present formalization of $\mathcal{F}_{\text{SIG}}$ and $\mathcal{F}_{\text{CERT}}$ is attractive in that it allows a very modular approach where each instance of the ideal functionality handles only a single instance of a signature scheme (i.e., a single pair of signature and verification keys). This has several advantages as described in this work. However, the interface of the signature scheme is somewhat less abstract than we may have wanted. Specifically, the interface contains an idealized "signature string" that is passed around among parties [...].

Indeed, Canetti [7, page 7] starts by describing a "first attempt" functionality $\mathcal{F}_1$ that is a "depository of signed messages," where the signer can input a message and the verifiers can check. This functionality can be seen as a simplified version of the authenticated repository we described above. He then argues, however, that including the technical details in the functionality's interface is inevitable, see [7, page 7]:

> The lack of explicit signature strings also causes some other modeling problems. For instance, modeling natural operations such as sending an "encrypted signature" that is usable only by the holders of the decryption key cannot be done in a modular way [...] We conclude that in order to capture our intuitive notion of signature schemes, an ideal signature functionality should make the "signature string" part of its interface. [...]

We want to argue here that, despite the similarity, the arguments given in [7] do not apply to our definition. The first argument is that the formulation binds the messages to the signer's identity instead of the verification key, which requires prior communication to transmit the verification key. While this argument is correct, and our definition makes the repository for transmitting the verification key explicit, we stress that the repositories abstract from concrete types of communication and merely specify that the correct verification key generated by the signer is accessible, in some way, to the verifier. The means of *how* it became accessible do not have to be specified.

The second argument is that (beyond requiring the communication of the signature string, which is analogous to the verification key), protocols that communicate a signature over a different connection than specified, such as an encrypted one, is a modeling challenge. One such protocol is SAML [15], where a signed assertion on the identity of a party is sent through a TLS connection. Despite the fact that this assertion is indeed encrypted, and SAML would therefore appear to be in the class of protocols referred to by Canetti, we show that our model, which does not explicitly expose the signature string, indeed allows to analyze the security of protocols like SAML. The reason is again that our model abstracts from the concrete communication semantics and in particular also allows to model the case where a signature is transferred securely.

There are protocols that make explicit use of the verification key or signature as a bit string and for which our model in its current form does not support a modular analysis. One example is the transformation from CPA-secure public-key encryption (PKE) to non-malleable PKE by Choi *et al.* [12], where each ciphertext is protected via an instance of a one-time signature scheme, and the bits of the verification key are used to select a certain subset of instances of the CPA-secure PKE. For the security reduction to succeed, however, it is necessary that the verification key be not only a bit string, but that it also be different for each instance, with high probability. While this property is clearly satisfied by every secure DSS, and therefore also each DSS that realizes $\mathcal{F}_{\text{SIG}}$, it is not captured in the functionality alone, where the adversary can freely choose the verification key. Hence, a composable analysis of the Choi *et al.* scheme in the $\mathcal{F}_{\text{SIG}}$-hybrid model is inherently impossible. In summary, this shows that the property of outputting *some* string as the verification key is not sufficient at least for the application of [12]. Another example are protocols that require parties to provide proofs (e.g., of knowledge) over inputs and outputs of the DSS algorithms. Yet, also here, the same issues appear with the formalization $\mathcal{F}_{\text{SIG}}$ that is independent of any concrete scheme. In summary, it remains open whether there is a natural scheme that can be modularly proved based on $\mathcal{F}_{\text{SIG}}$, but not using the more abstract definition we put forth in this paper.

Finally, our work can be seen as orthogonal to the work of Canetti et al. [11], which extends the model of Canetti [5,7] to the case where verification keys are available globally. While our model does not restrict the use of the constructed resource, the central aspect of our work is the different paradigm underlying the specification of the functionalities.

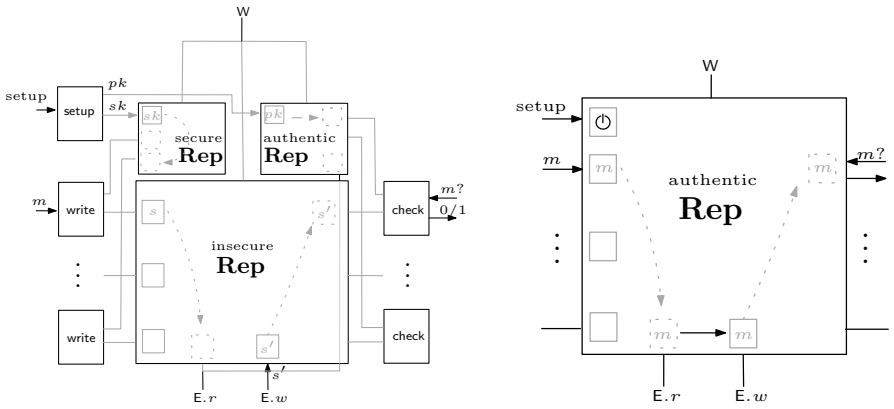

Fig. 2. Illustration of the main construction that characterizes a digital signature scheme. The real system with the protocol (left) and the desired resource (right). The adversarial interfaces are denoted by $\mathsf{E}.w$ (write) and $\mathsf{E}.r$ (read) and the free interface is denoted by W. The protocol is applied at the honest users' interfaces of the assumed resources.

1.3 Contributions

The first main contribution of our work is the formal model sketched in Sect. 1.1 above, which we formally specify in Sect. 3. We additionally prove several statements about DSSs using this model; in particular, we exemplify the use of the construction by two applications.

Capturing the security of a DSS. We define, in Sect. 4.1, the security of a DSS as constructing an authenticated repository, shown on the right-hand side of Fig. 2, from an insecure repository, called "insecure **Rep**" on the left-hand side of Fig. 2, an "authenticated **Rep**" to which one message can be written, and a "secure **Rep**" that allows to write a single message, but to which the adversary has neither read- nor write-interfaces. As shown in Fig. 2, using the signature scheme, which consists of the systems labeled setup, write, and check, requires the two single-message repositories for distributing the signing and verification keys. In more detail, in write each message is signed and the signature input into the insecure repository. Checking whether a given message m has been written to the repository is done by verifying the received signature for m within check.

We then prove that this construction statement is equivalent to the existential unforgeability of secure digital signature schemes in the sense of [14]:

Theorem (informal). *A DSS constructs an authenticated multi-message repository from an insecure multi-message repository, an authenticated single-message repository and a secure single-message repository if and only if it is existentially unforgeable.*

Following the discussion in [7], we have to argue that our abstract formalization of a signature scheme indeed models the intuitively expected properties of such a scheme. In particular, in Sect. 5, we show that the formalization directly models the *transferability* property of signature schemes in the sense that a receiver of a signature can forward it to another party, who can also verify it.

Message registration resource. We show that the security of a DSS in our model immediately implies that it can be used to construct a (authenticated) message registration resource. This resource allows multiple parties to input messages, which are then authenticated by one party referred to as the *issuer*. Letting the messages be public keys corresponds to the use of signatures in a public-key infrastructure.

Assertions and SAML. Finally, we show how our constructive definition can be used to prove the soundness of an important step in single-sign-on (SSO) mechanisms, which is to authenticate a session between a client and a server (often denoted service provider in this context) with the help of a digitally signed assertion from an identity provider.

2 Preliminaries

2.1 Discrete Systems and Notation

We model all components as discrete reactive systems and describe them in pseudo-code using the following conventions: We write $x \leftarrow y$ for assigning the value y to the variable x. For a distribution $\mathcal{X}$ over some set, $x \twoheadleftarrow \mathcal{X}$ denotes sampling x according to $\mathcal{X}$. For a finite set X, $x \twoheadleftarrow X$ denotes assigning to x a uniformly random value in X. For a table T of key-value pairs, with values in a set $\mathcal{V}$ and keys in a set $\mathcal{S}$, we denote by the assignment $T[s] \leftarrow v$ the binding of a key $s \in \mathcal{S}$ to a value $v \in \mathcal{V}$. This assignment overwrites any prior binding of s to some value. Analogously, $v \leftarrow T[s]$ denotes the look-up of the value that is currently bound to key s. If no value is bound to s, this look-up is defined to return $\perp$. The *empty table* is defined as the table where any look-up returns $\perp$.

More formally, discrete reactive systems are modeled by random systems [18]. An important similarity measure on those is given by the distinguishing advantage. More formally, the advantage of a distinguisher $\mathbf{D}$ in distinguishing two discrete systems, say $\mathbf{R}$ and $\mathbf{S}$, is defined as

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \;=\; \mathsf{Pr}\left[\mathbf{DR} = 1\right] - \mathsf{Pr}\left[\mathbf{DS} = 1\right],$$

where $\mathsf{Pr}\left[\mathbf{DR} = 1\right]$ denotes the probability that $\mathbf{D}$ outputs 1 when connected to the system $\mathbf{R}$. More concretely, $\mathbf{DR}$ is a random experiment, where the distinguisher repeatedly provides an input to one of the interfaces and observes the output generated in reaction to that input before it decides on its output bit.

A further important concept for discrete systems is a *monotone binary output (MBO)* [19] or *bad event* [3]. This concept is used to define a similarity between two systems, the *game equivalence* [18] or *equivalence until bad* [3], which means that two systems behave equivalently until the MBO is set (i.e., as long as the bad event does not occur), but may deviate arbitrarily thereafter. A widely-used result is the so-called *Fundamental Lemma of Game Playing* [18,3], which states that the distinguishing advantage between two such systems is bounded by the probability of provoking the MBO (i.e., bad event).

We stress that while especially the notion of bad event carries the connotation that such an event is supposed to occur only with small probability, this need not be the case. In particular, we will define specifications by means of the equivalence of two systems until an MBO is set, irrespective of how likely or unlikely this event is for a particular adversary. Such a specification is still interesting if, for each particular setting of interest, this probability turns out to be small.

2.2 Definition of Security

We use a term algebra to concisely write security statements. The *resources*, such as repositories, are written in bold-face font and provide *interfaces*, which are labeled by identifiers from a set $\mathcal{I}$, which can be accessed by parties. Protocol machines used by parties are also referred to as *converters* and are attached to some interface of a resource. This composition, which for a converter π, interface I, and resource $\mathbf{R}$ is denoted by $\pi^I\mathbf{R}$, again yields a resource. For a vector of converters $\pi = (\pi_{I_1}, \ldots, \pi_{I_n})$ with $I_i \in \mathcal{I}$, and a subset of interfaces $\mathcal{P} \subseteq \{I_1, \ldots, I_n\}$, $\pi_{\mathcal{P}}\mathbf{R}$ denotes the resource where π_I is connected to interface I of $\mathbf{R}$ for every $I \in \mathcal{P}$. For $\mathcal{I}$-resources $\mathbf{R}_1, \ldots \mathbf{R}_m$ the *parallel composition* $[\mathbf{R}_1, \ldots, \mathbf{R}_m]$ is again an $\mathcal{I}$-resource that provides at each interface access to the corresponding interfaces of all subsystems.

In this paper, we make statements about resources with interface sets of the form $\mathcal{I} = \mathcal{P} \cup \{\mathsf{E}, \mathsf{W}\}$ where $\mathcal{P}$ is the set of (honest) interfaces. A *protocol* is a vector $\pi = (\pi_{I_1}, \ldots, \pi_{I_{|\mathcal{P}|}})$ that specifies one converter for each interface $I \in \mathcal{P}$. Intuitively, $\mathcal{P}$ can be thought of as the interfaces that honestly apply the specified protocol π. On the other hand, interface E corresponds to the interface with potentially dishonest behavior and no protocol is applied at this interface. Intuitively, this interface models the attacker's capabilities to interfere with the honest protocol execution. Interface W is the free interface that models the influence of the environment on the resource. A constructive security definition then specifies the goal of a protocol in terms of *assumed* and *constructed* resources. We state the definition of a construction of [20].

Definition 1. *Let* $\mathbf{R}$ *and* $\mathbf{S}$ *be resources with interface set* $\mathcal{I}$*. Let* ε *be a function that maps distinguishers to a value in* $[-1, 1]$ *and let the interface label set be* $\mathcal{I} = \mathcal{P} \cup \{\mathsf{E}, \mathsf{W}\}$ *with* $\mathcal{P} \cap \{\mathsf{E}, \mathsf{W}\} = \emptyset$*. A protocol, i.e., a vector of converters* $\pi = (\pi_{I_1}, \ldots, \pi_{I_{|\mathcal{P}|}})$*,* constructs $\mathbf{S}$ *from* $\mathbf{R}$ *within* ε *and with respect to the*

simulator sim*, if*

$$\forall \mathbf{D}: \quad \Delta^{\mathbf{D}}(\pi_{\mathcal{P}}\mathbf{R},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{S}) \leq \varepsilon(\mathbf{D}). \tag{1}$$

This condition ensures that whatever an attacker can do with the assumed resource, she could do as well with the constructed resource by using the simulator sim. Turned around, if the constructed resource is secure by definition, there is no successful attack on the protocol.

The notion of construction is composable, which intuitively means that the constructed resource can be replaced in any context by the assumed resource with the protocol attached without affecting the security. This is proven in [20,21].

Specifications and relaxed specifications. As discussed in the introduction, we consider *specifications* [21] of reactive discrete systems, meaning systems that are not fully specified. The specifications can be understood in the sense of game equivalence: we define an event on the inputs (and outputs) of the discrete system, and the specification states that a system must show a certain specified behavior until the condition is fulfilled, but may deviate arbitrarily afterward.

The security statements according to Definition 1 can then be understood as follows. A protocol constructs from a specification $\mathcal{S}$ another specification $\mathcal{T}$ if for each system $\mathbf{S}$ that satisfies $\mathcal{S}$ there exists a system $\mathbf{T}$ that satisfies $\mathcal{T}$ such that the protocol constructs $\mathbf{T}$ from $\mathbf{S}$ [21].

While game equivalence in general is defined based on an arbitrary MBO of the system, the MBOs considered in this paper will be of a specific and simple form: they only depend on the order in which specific inputs are given to the systems. This formalizes the guarantee that the resource behaves according to the specification if the inputs have been given in that order. A stronger condition therefore corresponds to a weaker specification, and it is easy to see that if a protocol constructs $\mathcal{T}$ from $\mathcal{S}$, and the same additional condition is specified to obtain weakened specifications $\mathcal{S}^-$ from $\mathcal{S}$ and $\mathcal{T}^-$ from $\mathcal{T}$, then the same protocol also constructs $\mathcal{T}^-$ from $\mathcal{S}^-$. (This assumes that $\mathcal{S}^-$ and $\mathcal{T}^-$ are weakened in the same way. The statement can equivalently be seen as requiring the distinguishing advantage to be small only for a subset of distinguishers.)

As the specifications in this work, as described above, can be seen as partially defined discrete systems, we use the same notation, i.e., boldface fonts. In particular, we can understand equation (1) as extending to such partially defined discrete systems, by changing the system to respond with a constant output to the distinguisher once the MBO has been provoked. Due to the specific property of the MBO, a distinguisher cannot gain advantage by provoking the MBO.

2.3 Digital Signature Schemes

We recall the standard definition of a DSS from the literature.

Definition 2. *A* digital signature scheme $\Sigma = (K, S, V)$ *for a message space* $\mathcal{M}$ *and signature space* Ω *consists of a (probabilistic) key generation algorithm* K

that returns a key pair (sk, vk)*, a (possibly probabilistic) signing algorithm* S*, that given a message* $m \in \mathcal{M}$ *and the signing key* sk *returns a signature* $s \leftarrow S_{sk}(m)$*, and a (possibly probabilistic, but usually deterministic) verification algorithm* V*, that given a message* $m \in \mathcal{M}$*, a candidate signature* $s' \in \Omega$*, and the verification key* vk *returns a bit* $V_{vk}(m, s')$*. The bit* 1 *is interpreted as a successful verification and* 0 *as a failed verification. It is required that* $V_{vk}(m, S_{sk}(m)) = 1$ *for all* m *and all* (vk, sk) *in the support of* K*. We generally assume* $\mathcal{M} = \Omega = \{0,1\}^*$*.*

The standard security definition for DSS is existential unforgeability under chosen message attack [14], as described in the introduction. Since we target concrete security, we directly define the advantage of an adversary.

Definition 3 (EU–CMA). *For a digital signature scheme* $\Sigma = (K, S, V)$*, the* EU–CMA advantage of an adversary $\mathbf{A}$ *is defined using the security game* $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$ *in Fig. 3, in more detail,*

$$\mathit{\Gamma}^{\mathbf{A}}(\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}) := \mathsf{Pr}^{\mathbf{A}\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[\textsc{won} = 1].$$

$\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$

```
function Init
    (vk, sk) ←R K
    L ← ∅
    won ← 0

function Sign(m)
    L ← L ∪ {m}
    return (m, S_sk(m))

function Forge(m, s)        ▷ m ∈ M; s ∈ Ω
    if m ∉ L and V_vk(m, s) = 1 then
        won ← 1
    return won
```

Fig. 3. The security game EU-CMA.

Signature schemes may or may not allow to recover the message from the signature. Each signature scheme can easily be turned into one with message recovery by viewing (m, s) as the signature instead of s.

Definition 4. *A* digital signature scheme with message recovery $\Sigma_{\mathrm{rec}} = (K, S, R)$ *is a digital signature scheme where the verification algorithm* V *is replaced by a recovery algorithm* R*, that takes a candidate signature* s' *and outputs a value* $R_{vk}(s') \in \mathcal{M} \cup \{\perp\}$*, where* $\perp$ *is used to indicate that the signature* s' *is invalid. The correctness condition demands that* $R_{vk}(S_{sk}(m)) = m$ *for all* m *and all* (vk, sk) *in the support of* K*. The security notion is as in Definition 3, except for the winning condition: a successful adversary provides a signature* s' *such that* $m' := R_{vk}(s') \neq \perp$ *and* m' *was not a query to the signing oracle.*

3 Message Repositories

We formalize the message repositories described in the introduction, and show how they can be instantiated to model specific communication networks.

3.1 Description of Message Repositories

We consider general message repositories that export a certain capability, such as reading or writing a single message, at each of its interfaces. There are four types of ways in which one can access the repository to read or write its content: each interface $\mathsf{A} \in \mathcal{W}$ allows to insert one message into the repository. Interface $\mathsf{B} \in \mathcal{R}$ allows to read a message that has been written to the repository and made visible for B. Each interface $\mathsf{C} \in \mathcal{C}$ allows to write values into the repository by specifying from which (reader) interfaces the values should be copied; no new values can be inserted at interface C. For each copy-interface, there is a set of associated read-interfaces from which they can copy. Each interface $\mathsf{V} \in \mathcal{V}$ allows to verify whether a certain value m is visible at the interface; this can be seen as a restricted type of read access. Finally, the free interface W allows to manage the visibility of messages. On a call $\text{TRANSFER}(\mathsf{A}, \mathsf{B})$, the message written at A becomes visible at reader interface B. We often call the receiving interfaces *the receivers*. A precise specification of the repository appears in Fig. 4. As indicated by the keyword **Assume**, the behavior of the repository may be undefined if this assumption is not fulfilled, this is according to the discussion of specifications in Sect. 1 and 2. In contrast, "$\triangleright\ m \in \mathcal{M}$" is to be understood as a reminder or comment for the reader; the input m given to the system is necessarily in the alphabet $\mathcal{M}$ by definition of the system. (More technically, while the condition in **Assume** may be violated by an input, which may provoke an MBO, $m \in \mathcal{M}$ will always be satisfied.)

Note that one can easily generalize this basic specification to other types of read- or write-interfaces, for example to model output of partial information about a message, such as the length, but which we do not consider here and consider it as part of future work. Following the motivation of Sect. 1, for generality, we consider each described operation as associated with a separate interface.[4]

Definition 5. *For finite and pairwise disjoint sets $\mathcal{W}, \mathcal{R}, \mathcal{C}, \mathcal{V}$, and a family $\{\mathcal{R}_\mathsf{C}\}_{\mathsf{C}\in\mathcal{C}}$ of sets $\mathcal{R}_\mathsf{C} \subset \mathcal{R}$ for all $\mathsf{C} \in \mathcal{C}$, we define the repository $\mathbf{Rep}^{\mathcal{C},\mathcal{W}}_{\mathcal{R},\mathcal{V},\{\mathcal{R}_\mathsf{C}\}_{\mathsf{C}\in\mathcal{C}}}$ as in Fig. 4. For later reference, we define for $n, m, \ell, k \in \mathbb{N}$, the standard sets $\mathcal{W} = \{\mathsf{A}_i\}_{i\in[n]}$, $\mathcal{R} = \{\mathsf{B}_i\}_{i\in[\ell]}$, $\mathcal{C} = \{\mathsf{C}_i\}_{i\in[m]}$ and $\mathcal{V} = \{\mathsf{V}_i\}_{i\in[k]}$. If nothing else is specified, these standard interface names are used. We define the shorthand notation $\mathbf{Rep}^{m,n}_{\ell,k} := \mathbf{Rep}^{\mathcal{C},\mathcal{W}}_{\mathcal{R},\mathcal{V},\{\mathcal{R}_\mathsf{C}\}_{\mathsf{C}\in\mathcal{C}}}$ for these standard sets and $\mathcal{R}_\mathsf{C} = \mathcal{R}$ for all $\mathsf{C} \in \mathcal{C}$. For $\mathcal{C} = \emptyset$ we use the simplified notation $\mathbf{Rep}^{\mathcal{W}}_{\mathcal{R},\mathcal{V}}$.*

Different security guarantees can be expressed using this repository by considering different allocations of read-, write-, or transfer-interfaces to different parties as discussed in the introduction. For instance, an attacker could have access to both read- and write-interfaces, to model traditional insecure communication. If the attacker only has access to read-interfaces (but not to write-interfaces beyond potentially copy-interfaces to forward received messages), the

[4] Recall that it is always possible to merge several existing interfaces into one interface to model that a party or the attacker, in a certain application scenario, has the capability to write and read many messages.

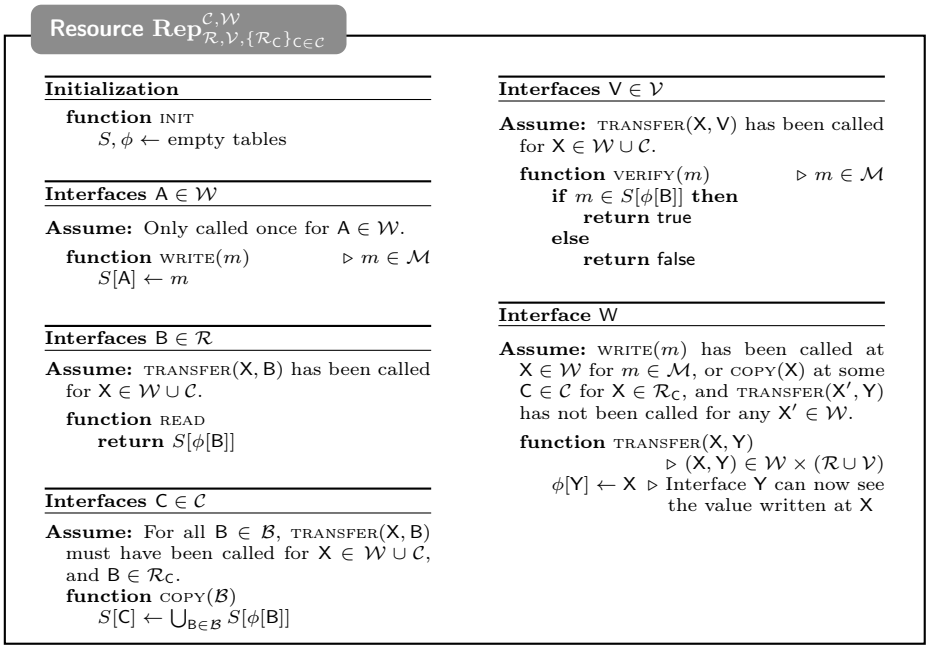

Fig. 4. Specification of a repository resource. For ease of notation, we treat values $m \in \mathcal{M}$ and singular sets $\{m\}$ for $m \in \mathcal{M}$ interchangeably.

repository corresponds to authenticated message transmission from a honest write-interface.

3.2 Modeling Security Guarantees by Access to the Repository

For security statements we need to associate each (non-free) interface to either an honest party or a possible attacker. As additional notation, we define the adversarial interfaces sets $\mathcal{E}_r := \{\mathsf{E}_1.r, \ldots, \mathsf{E}_k.r\}$ (for some $k > 0$), $\mathcal{E}_w := \{\mathsf{E}_1.w, \ldots \mathsf{E}_k.w\}$, and $\mathcal{E}_c := \{\mathsf{E}_1.c, \ldots, \mathsf{E}_k.c\}$ where the size k of this set is typically defined by the context. We can then specify repositories with different security guarantees.

- Insecure repositories allow adversarial write and read access. They can be described by $\mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}$, which means that all interfaces are either read- or write-interfaces.
- An authenticated repository disallows adversarial write-operations of arbitrary messages. Only (the honest) interface $\mathcal{W}$ can input content into the repository. This situation is described by the resource $\mathbf{Rep}^{\mathcal{E}_c,\mathcal{W}}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset,\{\mathcal{E}_r\}_{\mathsf{C}\in\mathcal{E}_c}}$, which indicates that the attacker may still be able to copy values from interfaces $\mathcal{E}_r$ at each interface $\mathcal{E}_c$.
- A repository without adversarial read-access, but with write access, models perfect confidentiality, and is described by $\mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R},\emptyset}$.

While the (natural) variants described above will be the only ones used in this work, the formalism allows to flexibly define various further combinations of honest-user and adversarial capabilities.

3.3 Example: Modeling Networks through Repositories

For considering concrete applications, such as a specific type of network transfer, the repository can be instantiated appropriately. In this section, we briefly describe in which sense statements about repositories imply statements about a network in which senders can *send* a message to a set of desired recipients, but which is under complete control of an attacker. We describe such a network in more detail in Fig. 16 in Appendix A.

In a nutshell, such a network can be described as a repository where for each write-interface of the honest senders, the attacker interface has a read-interface, and for each read-interface of the honest receivers, the attacker interface has a write-interface. Additionally, the attacker interface has the capabilities of the free interface that allow to transfer the values between the write- and the read-interfaces. This enables the attacker to eavesdrop on all values from the writer and to determine all values sent to the receiver; the traditional worst-case assumption.

4 A Constructive Perspective on Digital Signatures

4.1 The Basic Definitions

Our security definition for DSSs is based on the repositories introduced in Sect. 3. Intuitively, the honest parties execute a protocol to construct from an insecure repository, in which the attacker has full write access, one repository that allows the writer to authenticate a single message (this will be used for the verification key), and one repository that allows to store a single message securely (this will be used for the signing key), an authenticated repository that can be used for multiple messages. We generally use the notation introduced in Sect. 3. We first introduce the specifications that capture authenticated repositories since they are of primary interest in this section. The first type considers repositories where the role of the receiver interfaces is to verify values in the repository:

Definition 6. *Let $\mathcal{W}, \mathcal{R}, \mathcal{E}_w, \mathcal{E}_r$ denote the standard interface names. A specification $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$, in the sense of a partially defined discrete system, is an* authenticated repository for verification *if the following conditions are fulfilled. (1) It has at least the interfaces $\mathcal{I} = \mathcal{W} \cup \mathcal{R} \cup \mathcal{E}_w \cup \mathcal{E}_r$, where all inputs at $I \notin \mathcal{I}$ are ignored (i.e., the resource has the default behavior of directly returning back to the caller). (2) For all inputs at some interface $I \in \mathcal{I}$, the behavior is identical to the one specified in $\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R},\{\mathcal{E}_r\}_{\mathsf{C}\in\mathcal{E}_w}}$ for I, wherever the behavior of $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ is defined. More formally, this means that for a given sequence of inputs, the conditional distribution of $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$, where the outputs for inputs at*

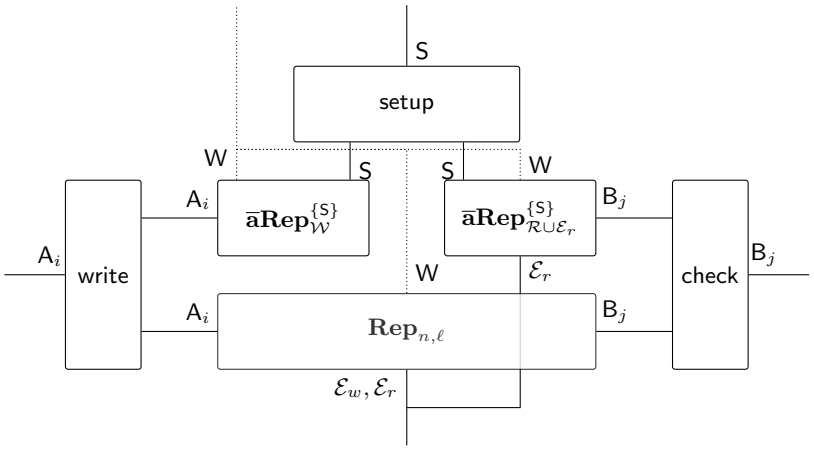

Fig. 5. The real-world setting of the signature construction.

interfaces not in $\mathcal{I}$ are marginalized, is the same as the conditional distribution of $\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R},\{\mathcal{E}_r\}_{\mathsf{C}\in\mathcal{E}_w}}$ *without those inputs.*

The second definition is analogous and considers repositories where the role of the receiver interfaces is to authentically receive values:

Definition 7. *Let $\mathcal{W},\mathcal{R},\mathcal{E}_w,\mathcal{E}_r$ denote the standard interface names. The specification* $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$*, in the sense of a partially defined discrete system, is an* authenticated repository for receiving *if it has at least the interfaces $\mathcal{I}=\mathcal{W}\cup\mathcal{R}\cup\mathcal{E}_w\cup\mathcal{E}_r$, all inputs at $I\notin\mathcal{I}$ are ignored, and for all inputs at some interface $I\in\mathcal{I}$ the behavior is identical to the one specified in* $\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R},\emptyset,\{\mathcal{E}_r\}_{\mathsf{C}\in\mathcal{E}_w}}$ *for I, wherever the behavior of* $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$ *is defined. We omit $\mathcal{E}_w$ in the notation if it is equal to $\emptyset$.*

In the following, whenever referring to the sets $\mathcal{W},\mathcal{R},\mathcal{E}_w$, and $\mathcal{E}_r$, we implicitly refer to the standard names introduced in the previous section.

Assumed resources. As outlined in Sect. 1, to construct an authenticated repository, we require (beyond an insecure repository to transmit the signatures) an additional resource that allows to distribute one value authentically to all verifiers and one value securely to all signers. This assumed communication is described by the specification $\bar{\mathbf{a}}\mathbf{Rep}^{\mathsf{S}}_{\mathcal{W}}$, which specifies resources with one writer interface S and no active adversarial interface. Information can only be transferred from S to the interfaces of $\mathcal{W}$. To model the authenticated (but not confidential) transmission of a value, we assume another resource as specified by $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\mathsf{S}}_{\mathcal{E}_r\cup\mathcal{R}}$ where information can only be transferred from S to the interfaces in $\mathcal{R}$, but is not limited to those as also adversarial interfaces may read this value

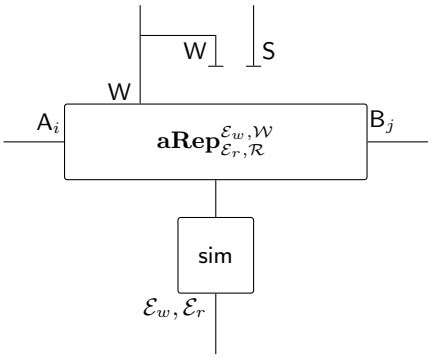

Fig. 6. The ideal-world setting of the signature construction. Inputs at the interfaces whose corresponding lines stop before the box (interfaces W and S in this example) have no effect on the behavior, therefore they are ignored in the specification.

or copy it via the interfaces in $\mathcal{E}_c$. We define the assumed system as consisting of the two above-described resources and an insecure repository $\mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}$, i.e., as

$$\mathbf{R}_{n,\ell} := \left[\overline{\mathbf{a}}\mathbf{Rep}^{\mathsf{S}}_{\mathcal{W}}, \overline{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\mathsf{S}}_{\mathcal{E}_r\cup\mathcal{R}}, \mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}\right] . \tag{2}$$

For clarity, whenever we explicitly refer to the assumed mechanism to distribute the keys, we use the shorthand notation

$$\mathbf{Dist} := \left[\overline{\mathbf{a}}\mathbf{Rep}^{\mathsf{S}}_{\mathcal{W}}, \overline{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\mathsf{S}}_{\mathcal{E}_r\cup\mathcal{R}}\right].$$

Protocol converters. We assign one converter to each of the three roles: a converter write for the (honest) writer interfaces, a converter check for the (honest) reader interfaces and a setup-converter setup at interface C. We define the vector of converters $\mathsf{DSS} := (\mathsf{setup}, \mathsf{write}, \ldots, \mathsf{write}, \mathsf{check}, \ldots, \mathsf{check})$ with n copies of write, ℓ copies of converter check and one converter setup. The set of honest interfaces in this section is defined as $\mathcal{P} := \{\mathsf{S}\} \cup \mathcal{W} \cup \mathcal{R}$.

Goal of construction: an authenticated repository. Intuitively, the use of a DSS should allow us to construct from a repository $\mathbf{Rep}^{\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r}$ that allows both the honest users and the attacker to write multiple messages, and a repository that exclusively allows one honest user to write the verification key authentically, a repository in which the attacker has no write access. The reason is that writing a message that will be accepted by honest readers requires to present a valid signature relative to the verification key, thus the attacker would be required to forge signatures. This intuition does, however, not quite hold.

Indeed, when using the insecure repository, the attacker can still copy valid signatures generated by the honest writer to which he has read access via any

of his write interfaces. Since honest readers may later gain read access to those copied signatures, the attacker can indeed control *which* of the messages originating from the honest writer will be visible at those interfaces. The repository that is actually constructed is a specification $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ as in Definition 6. The goal of a digital signature scheme can thus be understood as amplifying the capabilities of authenticated repositories as defined using the specifications above.

To give a more concrete intuition, a particular constructed resource still has an interface S and accepts queries $\textsc{transfer}(\mathsf{S},\mathsf{A}_i)$ and $\textsc{transfer}(\mathsf{S},\mathsf{B}_j)$, in addition to those provided by $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$. Providing input at these interfaces, as indicated by the dead ends drawn in Fig. 6, has no effect, but may influence whether further outputs of the system are still defined (because, e.g., inputs to the system may have been provided in an order such that the behavior of the DSS is not defined).

In the remainder of the section, we prove an equivalence between the validity of the described construction and the definition of existential unforgeability. As the protocol converters described above do not exactly match the algorithms in the traditional definition of a DSS, we also explain how to convert between the two representations of a signature scheme.

4.2 Unforgeability of Signatures implies Validity of Construction

The constructed specification $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ has further (inactive) interfaces beyond those in $\mathcal{I} = \mathcal{W} \cup \mathcal{R} \cup \mathcal{E}_w \cup \mathcal{E}_r$, and behaves equivalently to $\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R},\{\mathcal{E}_r\}_{\mathsf{C}\in\mathcal{E}_w}}$, as long as the assumed order of inputs is respected. The following theorem states that any existentially unforgeable digital signature scheme can be used to construct such an authentic repository from the assumed resources (see also Fig. 2 for a depiction of this statement).

Constructing a specification $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ according to Definition 6 can be a vacuous statement: the specification can be undefined for all possible orders of inputs. The statement we prove in this section, therefore, explicitly specifies for which orders $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ is defined. In particular, the specification is defined for all orders of inputs for which the underlying specifications $\bar{\mathbf{a}}\mathbf{Rep}^{\mathsf{S}}_{\mathcal{W}}$ and $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\mathsf{S}}_{\mathcal{E}_r\cup\mathcal{R}}$ are defined, plus the following natural conditions of a DSS: the keys are generated first and are distributed before anything is signed or verified at a writer or reader interface. As long as these conditions are satisfied, the specification defines the output of the resource.

We now state the formal theorem.

Theorem 1. *Let $n, \ell \in \mathbb{N}$. For any given digital signature scheme $\Sigma = (K, S, V)$, let the converters* $\mathsf{write}, \mathsf{check}$, *and* setup *be defined as in Fig. 8. Then, for the simulator* sim *defined in Fig. 7, there is an (efficient) reduction* $\mathbf{C}$ *described in the proof, that transforms any distinguisher* $\mathbf{D}$ *for systems* $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ *and* $\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell}$, *with* $\mathbf{aRep}_{n,\ell} = \mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ *as described above, into an adversary* $\mathbf{A} := \mathbf{DC}$ *against the game* $\mathbf{G}^{\mathsf{EU\text{-}CMA}}_{\Sigma}$ *such that*

$$\Delta^{\mathbf{D}}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}) \leq \Gamma^{\mathbf{A}}(\mathbf{G}^{\mathsf{EU\text{-}CMA}}_{\Sigma}),$$

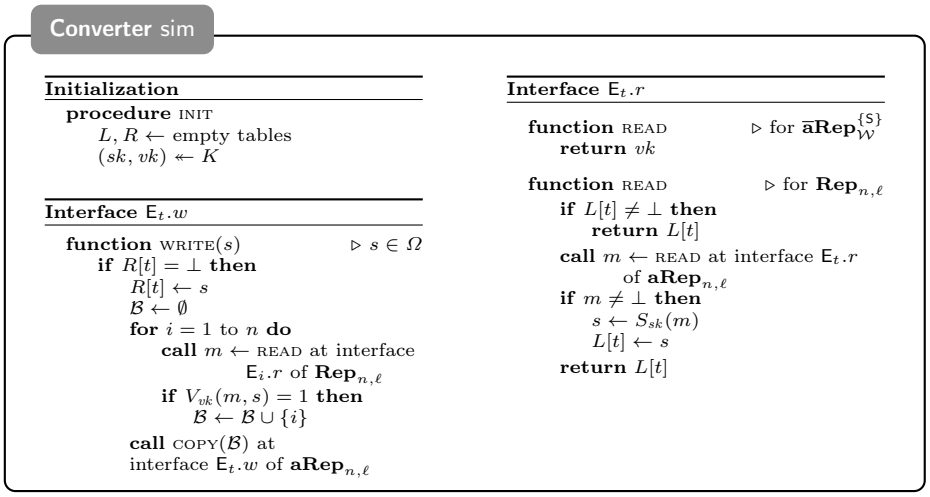

Fig. 7. Simulator for the proof of Theorem 1.

and where $\mathbf{aRep}_{n,\ell}$ *is defined as long as the assumed specification is defined and the following conditions hold:*

- *Command* SETUP *is issued at the* S*-interface before any other command;*
- *Command* TRANSFER$(\mathsf{S}, \mathsf{A}_i)$ *is issued at the* W*-interface corresponding to the first setup repository before* WRITE *is issued at the* A_i*-interface;*
- *Command* TRANSFER$(\mathsf{S}, \mathsf{B}_i)$ *is issued at the* W*-interface corresponding to the second setup repository before* READ *is issued at the* B_i*-interface.*
- *There are no* TRANSFER(X, Y) *queries with* $\mathsf{X} \in \mathcal{E}_w$ *and* $\mathsf{Y} \in \mathcal{E}_r$*, that is, we exclude communication from the adversarial writer to adversarial reader-interfaces.*

In other words, the signature scheme constructs the specification $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ from the assumed specification $\mathbf{R}_{n,\ell}$.

Proof. The proof is given in Sect. B.1.

4.3 Chaining Multiple Construction Steps

The construction proved in Theorem 1 assumes (amongst others) an authenticated repository $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\mathsf{S}}_{\mathcal{E}_r\cup\mathcal{R}}$ and constructs an authenticated repository $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$. A natural question is in which sense multiple such construction steps can be chained, corresponding to signing the verification key of one instance with a different instance of the scheme. For this to work out, we have to "upgrade" the resource $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ to a resource $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$ as needed by Theorem 1, where we can then use any interface $\mathsf{X} \in \mathcal{W}$ as the interface S to transmit the secret key. Of course, we additionally require resources $\bar{\mathbf{a}}\mathbf{Rep}^{\{\mathsf{X}\}}_{\mathcal{W}'}$ for distributing the secret keys and $\mathbf{Rep}^{\emptyset,\mathcal{W}'\cup\mathcal{E}'_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}$ for transmitting the signatures.

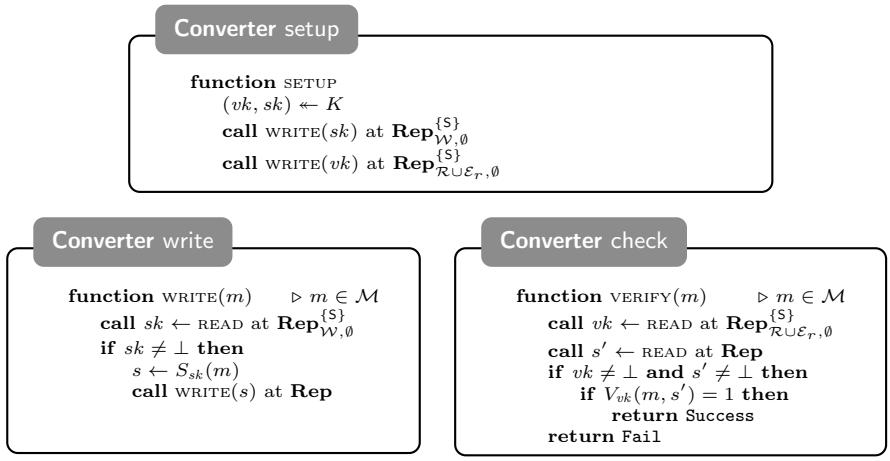

Fig. 8. The three protocol converters derived from a signature scheme $\Sigma = (K, S, V)$.

The chaining is then achieved by the protocol that consists of converters send and receive, sends the messages over an (additional) insecure repository $\mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}$ and authenticates them via $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$. Protocol converter send simply inputs the same message to both resources, whereas receive verifies the messages obtained through the insecure repository at the authenticated repository. This protocol perfectly constructs an authenticated repository with delivery from the two assumed resources.

Theorem 2. *Let $n, \ell \in \mathbb{N}$, and consider a protocol* SND *with converters* send *for all interfaces in $\mathcal{W}$ and converters* receive *for all interfaces in $\mathcal{R}$, defined as described above. Then, for the simulator* sim *described below,*

$$\mathsf{SND}_{\mathcal{P}}\left[\mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}, \mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}\right] \;\equiv\; \mathsf{sim}^{\mathsf{E}}\overline{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}} ,$$

wherever both resources are defined. The constructed resource $\overline{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$ accepts TRANSFER *commands at sub-interfaces corresponding to both assumed resources, and requires, for a given message to be transferred, both those commands to be issued.*

The simulator sim responds to READ queries at the $\mathcal{E}_r$-interfaces corresponding to $\mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}$ or $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ by obtaining the transmitted messages from $\overline{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$. Once COPY has been called at an $\mathcal{E}_w$-interface at $\mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$ and the corresponding message has been input at the same $\mathcal{E}_w$-interface of $\mathbf{Rep}^{\emptyset,\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset}$, sim issues the same COPY command at $\overline{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$.

Together with Theorem 1, this means that sending a message along with a signature constructs an authenticated repository from which the authenticated messages can be read. Several such constructions can then be chained in the expected way.

4.4 Validity of Construction implies Unforgeability of Signatures

In this section, we show that any converters achieving the construction of **aRep** from **Rep** and **Dist** contain a digital signature scheme that is existentially unforgeable under chosen-message attacks. More precisely:

Theorem 3. *Let $n, \ell \in \mathbb{N}$. Consider arbitrary converters* setup, write, *and* check *and define the protocol as* $\mathsf{DSS} := (\mathsf{setup}, \mathsf{write}, \ldots, \mathsf{write}, \mathsf{check}, \ldots, \mathsf{check})$ *(for the honest interfaces) with n copies of* write, *ℓ copies of converter* check *and one converter* setup. *We derive a digital signature scheme $\Sigma = (K, S, V)$ below in Fig. 9 with the following property: given any adversary against the signature scheme that asks at most n queries to* Sign *and ℓ queries to* Forge, *we construct (efficient) distinguishers $\mathbf{D}_i$, $i = 1 \ldots 5$, such that for the systems $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ and $\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell}$, with $\mathbf{aRep}_{n,\ell} = \mathbf{aRep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r,\mathcal{R}}$, for all simulators* sim,

$$\Gamma^{\mathbf{A}}(\mathbf{G}^{\mathsf{EU\text{-}CMA}}_{\Sigma}) \leq \sum_{i=1}^{5} \Delta^{\mathbf{D}_i}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}),$$

and where $\mathbf{aRep}_{n,\ell}$ is defined as long as the assumed specification is defined and under the same additional conditions as in Theorem 1.

Proof. The proof is given in Sect. B.2. □

As a corollary, one can specifically deduce that if there exists a simulator sim such that systems $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ and $\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell}$ are indistinguishable, then the constructed signature scheme Σ is existentially unforgeable under chosen message attacks.

Obtaining the signature scheme from the converters. The key generation, signing, and verification functions are derived from the converters setup, write, and check that construct **aRep** from [**Dist**, **Rep**] as follows: The key generation K consists of evaluating the function setup.SETUP, the two values written to the resource **Dist** are considered as the corresponding key pair. The secret key is the value that is written to the first sub-system of **Dist**. The signing algorithm $S_{sk}(m)$ consists of evaluating the function write.WRITE(m). The signature for message m is defined as the value that is written to the repository. Any request to obtain a value from resource **Dist** is answered by providing the signing key sk. The verification algorithm $V_{vk}(m, s)$ consists of evaluating the function check.VERIFY(m) and the candidate signature s is provided as the actual value in the repository and the verification key vk is given as the value in **Dist**. The formal description of the algorithms appear in Fig. 9.

4.5 Digital Signatures with Message Recovery

So far we have focused on repositories that offer the capability to check whether a given value has been written to the buffer and denoted them by **aRep**. Now,

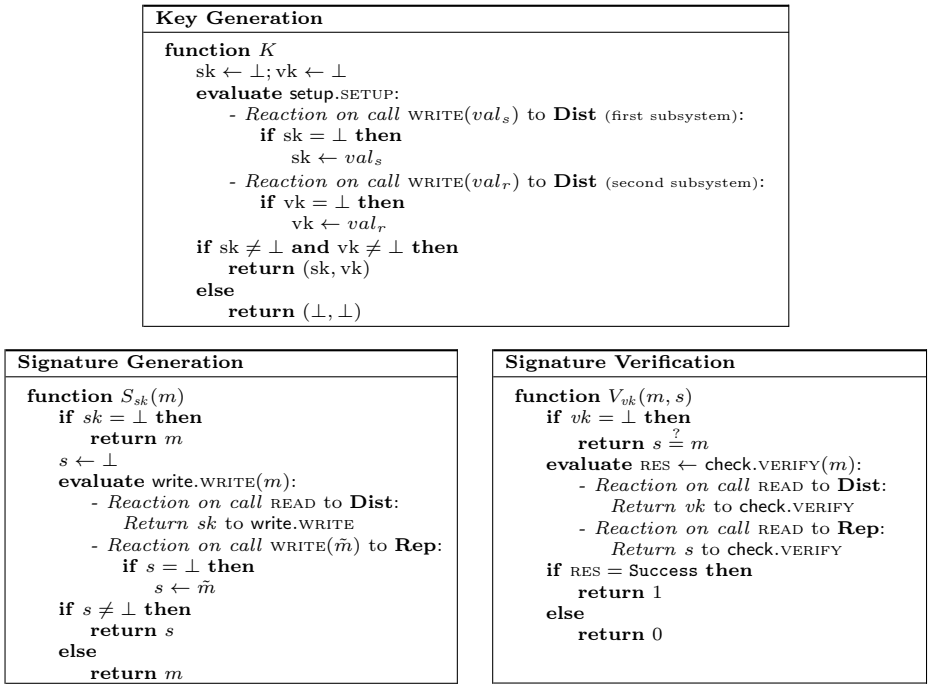

Fig. 9. Signature scheme (K, S, V) extracted from converters setup, write, and check.

we consider repositories that offer the capability to retrieve the value that has been transferred to an interface. In other words, the goal of this section is to show how to construct the specification $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$.

While the construction of **aRep** from **Rep** and **Dist** is achieved by traditional signature schemes, the construction of **ā**Rep from the same assumed resources is achieved by signature schemes with message recovery. Intuitively, converter check is replaced by a converter read whose task is to recover and output the message (and not simply check the authenticity of a given message). It is easy to see that any signature scheme $\Sigma_{\text{rec}} = (K, \Sigma, R)$ can be used to derive converters that achieve the construction (similar to the previous section). For the other direction, we have:

Theorem 4. *Let $n, \ell \in \mathbb{N}$. Consider arbitrary converters* setup, write, *and* read *and define the protocol as* DSS := (setup, write, ..., write, read, ..., read) *(for the honest interfaces) with n copies of* write, *ℓ copies of converter* read *and one converter* setup. *One can derive a digital signature scheme $\Sigma_{\text{rec}} = (K, S, R)$* with message recovery *with the following property: given any adversary against the signature scheme that asks at most n queries to* Sign *and ℓ queries to* forge, *we derive (efficient) distinguishers $\mathbf{D}_i$, $i = 1 \ldots 5$, such that for the systems*

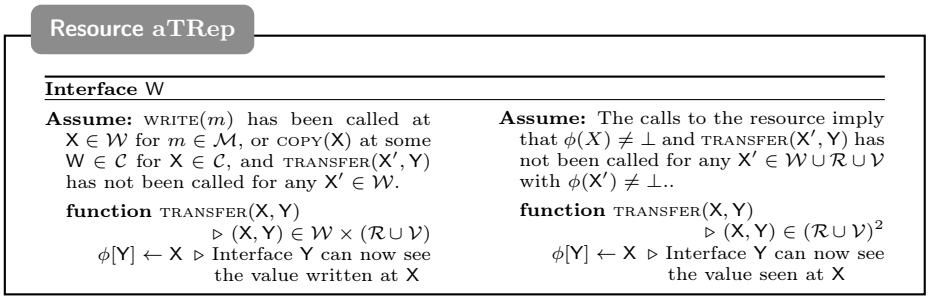

Fig. 10. Specification of a repository resource with transferable rights. Only the modifications with respect to Fig. 4 are shown; the other functions are as described there.

$\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ *and* $\mathsf{sim}^{\mathsf{E}}\bar{\mathbf{a}}\mathbf{Rep}_{n,\ell}$*, with* $\bar{\mathbf{a}}\mathbf{Rep}_{n,\ell} = \bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\mathcal{W}}_{\mathcal{E}_r\cup\mathcal{R}}$*, for all simulators* sim*,*

$$\Gamma^{\mathbf{A}}(\mathbf{G}^{\mathsf{EU\text{-}CMA}}_{\Sigma_{\mathrm{rec}}}) \leq \sum_{i=1}^{5} \Delta^{\mathbf{D}_i}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\bar{\mathbf{a}}\mathbf{Rep}_{n,\ell}),$$

and where $\bar{\mathbf{a}}\mathbf{Rep}_{n,\ell}$ *is defined as long as the assumed specification is defined and under the same additional conditions as in Theorem 1.*

Proof. We omit the proof and simply mention that it follows the same line of argumentation as the proof of Theorem 3. Algorithms K and S are derived in the same way as in Sect. 4.4 and the recovery algorithm $R_{vk}(s)$ is derived from converter read by evaluating the function READ (and appropriately providing s and vk) and to return whatever this function returns. □

5 On the Transferability of Verification Rights

Universal verification is arguably an important property of signatures. Anybody possessing the public key and a valid signature string s for some message m can verify the signature. This implies furthermore that signatures are naturally transferable, which is essential for their key role in public-key infrastructures or signing electronic documents. In this section, we demonstrate that our definition directly implies transferability by constructing a message repository in which information can be forwarded among readers. The high-level idea is to apply a converter to the free interface that instead copies the desired message from the sender buffer, where it was input originally, to the targeted reader buffer.

The role of the free interface. Recall that the role of the free interface in the repository resources is to transfer the contents from certain write-buffers to certain read-buffers. The transferability of signatures then simply means that values can also be transferred from read-buffers to other read-buffers; this can easily be achieved by translating the transfer-requests appropriately.

The core idea, then, is to observe that the new repository and the old repository only differ by attaching a converter at interface W. We assign a

new name to this resource and define $\mathbf{aTRep} = \mathsf{relay}^{\mathsf{W}}\mathbf{aRep}$ (and analogously $\bar{\mathbf{a}}\mathbf{TRep} := \mathsf{relay}^{\mathsf{W}}\bar{\mathbf{a}}\mathbf{Rep}$) with a converter relay that always remembers the existing assignments of reader to writer interfaces and on a transfer-query for two reader interfaces, it simply connects the corresponding writer-interface. The resource **aTRep** is additionally formally described in Fig. 10.

The converter. Converter relay distinguishes two types of inputs: transfer commands from a writer to a reader $\textsc{transfer}(\mathsf{X}, \mathsf{Y})$ are forwarded to the connected repository. Transfer commands between two readers, $\textsc{transfer}(\mathsf{R}_1, \mathsf{R}_2)$ are translated to transfer commands $\textsc{transfer}(\mathsf{X}, \mathsf{R}_2)$, where X denotes the writer interface where the value readable at R_1 was first input.

A simple black-box construction. Any protocol that constructs **aRep** from **Rep** (and **Dist**) also constructs $\mathsf{relay}^{\mathsf{W}}\mathbf{aRep}$ from $\mathsf{relay}^{\mathsf{W}}\mathbf{Rep}$ (and **Dist**), where the assumed resource $\mathsf{relay}^{\mathsf{W}}\mathbf{Rep}$ is an insecure repository that also allows information transfer between two receivers, i.e., sending a signature from one receiver to another. This is easy to see: assume there was a distinguisher **D** for systems $\mathsf{sim}^{\mathsf{E}}\mathsf{relay}^{\mathsf{W}}\mathbf{aRep}$ and $[\mathsf{relay}^{\mathsf{W}}\mathbf{Rep}, \mathbf{Dist}]$, and we are going to construct a distinguisher $\mathbf{D}'$ for the underlying two resources without the converter relay attached. (Note that sim is the same simulator as in Theorem 1.) Distinguisher $\mathbf{D}'$ simply behaves as **D** but additionally emulates relay for queries at the free interface.

6 Application 1: Implementing a Registration Service

The goal of this section is to construct a resource that allows several parties to send messages authentically to a population of receivers, via one *issuer* that authenticates the messages. This happens in public-key infrastructures, where the issuer, which is also denoted by *certification authority* in that context and can authenticate messages, acts as a relay. This is the setup of a (simple) public-key infrastructure and its use in Internet protocols, where the senders correspond to the *submitters* of public keys to the CA (registration), and the receivers are the consumers those public keys to authenticate messages. For the remainder of the section, we will therefore refer to the senders as submitters and the receivers as consumers (although the resource can of course also be used in other protocols).

The registration resource Reg. We denote the set of interfaces for the submitters by $\mathcal{S} := \{\mathsf{S}_1, \ldots, \mathsf{S}_\ell\}$, the consumers by $\mathcal{C} := \{\mathsf{C}_1, \ldots, \mathsf{C}_\ell\}$, and the interfaces for the issuer by I. The adversarial interface is denoted by E. The registration resource **Reg** offers the capability to input a value x at any submitter interface. Once this value has been transferred to the issuer, he can acknowledge the value by calling $\textsc{issue}$ at its interface. Once this happened, the value x, together with the information which submitter has input the value, can be made available at any consumer interface and, in addition, it can be transferred between any two consumer interfaces (or submitter interfaces). The formal description of the behavior of **Reg** appears in Fig. 11.

Assumed resources and the protocol. We assume a network resource **Net** which allows any party interface to send (by calling SEND) and receive messages (by calling RECEIVE), and allows the attacker to read all messages and send any message (note that the honest parties in a network have no means to verify who sent the message). In addition, we assume authentic communication as a setup. More formally, let $\mathbf{Ch}_{\mathsf{I}\leftarrow}$ be a system that has interface set $\{\mathsf{I}\}\cup\mathcal{S}\cup\{\mathsf{E}_1,\ldots,\mathsf{E}_\ell\}$. Each interface except the issuer offers the capability to send one message, i.e., to call SEND(m), which can be fetched at the issuer interface (they are authentic in the sense that the message cannot be modified and the resource indicates to the receiver who is the sender of the message). The issuer interface I can be thought of as being divided into 2ℓ sub-interfaces, and each sub-interface offers the capability to obtain the message from the corresponding sender (and hence identifies the sender reliably). Also, let the system $\mathbf{Ch}_{\mathsf{I}\rightarrow}$ be defined similarly, but which allows the issuer to send two messages in an authenticated manner to *each* submitter and one to each consumer.[5]

We can now describe the protocol that implements a registration service based on the above setup. The issuer's converter, upon ISSUE, takes all values x received on the incoming authenticated channel and acknowledges them by signing value (x,λ), where λ is a unique identifier that the issuer assigns to its sub-interface from which x was received.[6] The issuer sends the signed value back via the outgoing authentic channel. The protocol for the submitters, upon REGISTER(x) simply send x to the issuer over the authentic channel. Finally, the consumer converter reads inputs from the insecure network. When reading a new input, they verify the received value-signature pair and output the associated value only if the signature can be verified.

Theorem 5. *Let $\mathcal{S},\mathcal{C}$ be the above sets and I an interface name (different from all remaining interfaces). The protocol of Fig.* 12 *constructs resource* **Reg** *from the described set of authenticated channels. More specifically, for converters* issue *and* reg*, the simulator* $\tilde{\mathsf{sim}}$ *defined in Fig. 18 in Appendix C, it holds that for all distinguishers* $\mathbf{D}$ *there is an attacker against the signature scheme (with essentially the same efficiency), i.e.,*

$$\Delta^{\mathbf{D}}(\mathsf{issue}^{\mathsf{I}}\ \mathsf{reg}^{\mathsf{S}_1}\ldots\mathsf{reg}^{\mathsf{S}_\ell}\mathsf{rel}^{\mathsf{C}_1}\ldots\mathsf{rel}^{\mathsf{C}_\ell}[\mathbf{Ch}_{\mathsf{I}\rightarrow},\mathbf{Ch}_{\mathsf{I}\leftarrow},\mathbf{Net}],\tilde{\mathsf{sim}}^{\mathsf{E}}\mathbf{Reg}^{\mathsf{I}}_{\mathcal{S},\mathcal{C}}) \leq \Gamma^{\mathbf{A}}(\mathbf{G}^{\mathsf{EU\text{-}CMA}}_{\Sigma}).$$

Proof. Due to the abstract nature of repositories, we can easily represent the real world by a wrapped repository, and the ideal world as a wrapped authenticated repository and conclude the statement by invoking the results from the previous section. The proof is given in Appendix C. □

[5] This setup reflects that we need to distribute the issuer's verification key to each participant and in addition one signature to each submitter.

[6] In an application, this identifier could be the name of a server or a company. For concreteness we assume the identifier of the ith sub-interface to simply be the number i.

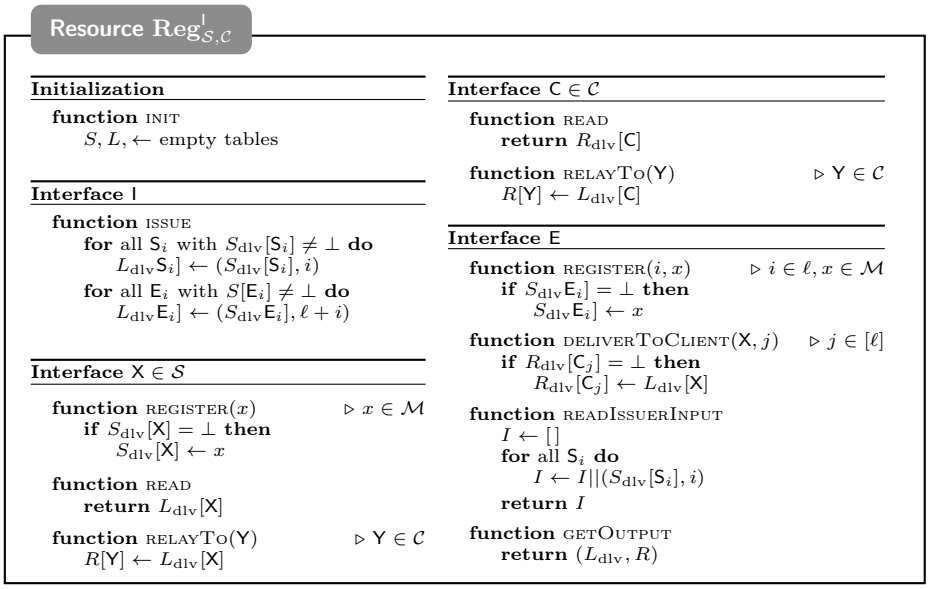

Fig. 11. The registration service resource.

7 Application 2: Authenticating Sessions using Assertions

Unilaterally secure channels. Establishing secure sessions in the internet is a crucial task. The most widely known solution to establish secure session is TLS, that, in a first handshake phase, establishes a shared key between client and server. Subsequently, this key is used to authenticate and encrypt the communication. In TLS, the server is usually authenticated, whereas the client is not. This results in an only unilateral authenticity guarantee: while the client is guaranteed that its messages are received by the intended server, the server does not know whether he is communicating with a legitimate client or with an attacker. This security guarantees for unilaterally secure channels is captured by the resource $\mathbf{NET}^{n}_{\mathsf{uni}}$ and the guarantee provided by the mutually authenticated secure channel is captured by the resource $\mathbf{NET}^{n,\mathsf{IdP}}_{\mathsf{mut}}$. The description appears in Fig. 13.

Modeling session authentication of SSO schemes. In a typical single-sign-on use case, the clients have a unilaterally authenticated session with the service provider. Aside of that they have establish a mutually authenticated and secure session with the identity provider. In practice, such a session is authenticated using a secure channel protocol involving an authentication based on passwords, hardware tokens, or one-time codes. In short, there is a secure channel between the identity provider and the client denoted by $\mathbf{SEC}_{\mathsf{IdP},\mathsf{C}_1}$.

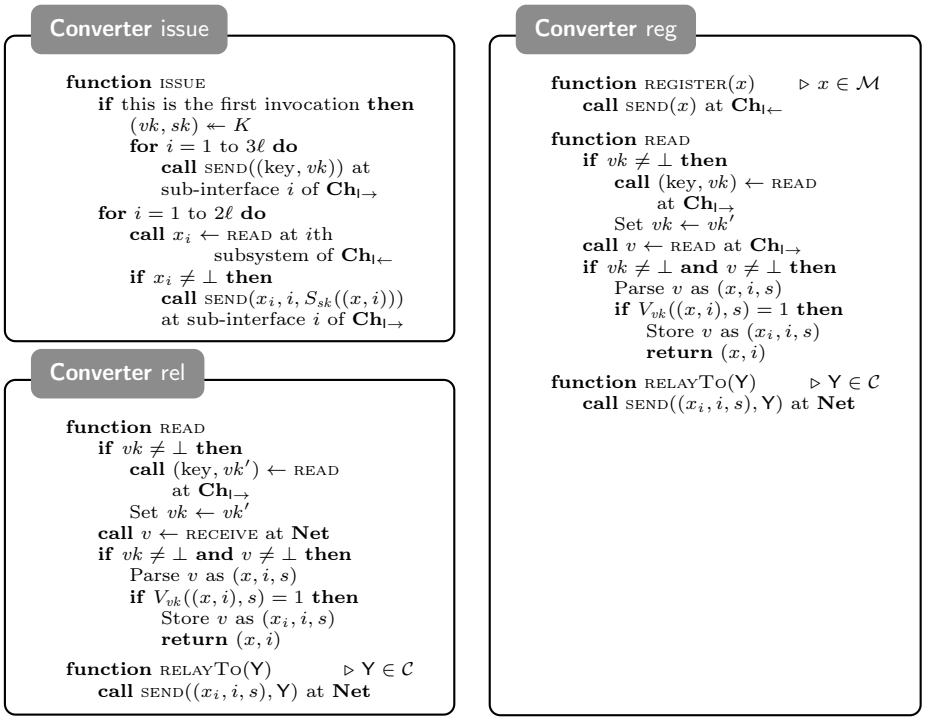

Fig. 12. The protocol converter for the issuer (upper left), the protocol converter for the submitter (upper right) to register a public value x (right), and the converter for the consumers (lower left) that can read and relay values.

Aside of this assumed channel, we again need a mechanism to distribute the verification key of the identity provider using an authenticated channel between the identity provider and the service provider. which we denote by $\mathbf{Ch}_{\mathsf{IdP}\to\mathsf{SP}}$.

The client realizes a mutually authenticated secure channel $\mathbf{NET}^{n,\mathsf{IdP}}_{\mathsf{mut}}$ by relaying a signed message (and its signature string), i.e., *the assertion*, from the identity provider to the service provider (this is usually denoted as IdP-initiated scenario in SSO terms) and have the signature verified by the service provider. In more detail, the protocol converter assert for the identity provider distributes its verification key and signs a specific token and sends it to the client via the assumed secure channel. The client converter fwd forwards this token to the service provider. Finally, the converter of the service provider, denoted filter, only starts outputting messages once the token is received and verified as the first message from $\mathbf{NET}^{n}_{\mathsf{uni}}$. Note that we treat all interfaces SP_i as sub-interfaces of one service provider interface SP. We establish the following theorem:

Theorem 6. *The protocol in Fig. 15 consisting of the service provider protocol* filter*, the identity provider protocol* assert*, and the client protocol* fwd*, constructs the mutually secure network* $\mathbf{NET}^{n,\mathsf{IdP}}_{\mathsf{mut}}$*, from the assumed, unilaterally secure,*

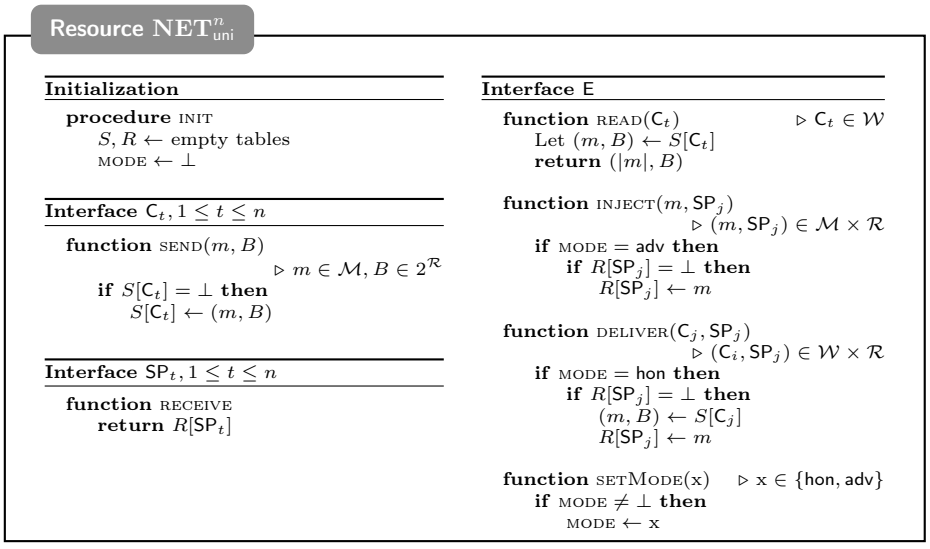

Fig. 13. The unilaterally secure network resource, where an adversarial interface E can choose whether the network runs in *secure* mode (MODE $=$ hon) or *adversarial* mode (MODE $=$ adv).

setting $[\mathbf{Ch}_{\mathsf{IdP}\to\mathsf{SP}}, \mathbf{SEC}_{\mathsf{IdP},\mathsf{C}_1}, \mathbf{NET}^n_{\mathsf{uni}}]$. *More specifically, for any distinguisher* $\mathbf{D}$*, there is an attacker* $\mathbf{A}$ *against the underlying signature scheme (with essentially the same efficiency), such that*

$$\Delta^{\mathbf{D}}(\mathsf{fwd}^{\mathsf{C}_1}\mathsf{assert}^{\mathsf{IdP}}\mathsf{filter}^{\mathsf{SP}}\,[\mathbf{Ch}_{\mathsf{IdP}\to\mathsf{SP}}, \mathbf{SEC}_{\mathsf{IdP},\mathsf{C}_1}, \mathbf{NET}^n_{\mathsf{uni}}]\,, \tilde{\mathsf{sim}}\mathbf{NET}^{n,\mathsf{IdP}}_{\mathsf{mut}}) \le \Gamma^{\mathbf{A}}(\mathbf{G}^{\mathsf{EU\text{-}CMA}}_{\Sigma}).$$

Proof. The proof idea is the same as in Theorem 5. As in the previous section, the abstract nature of repositories, allows us to represent the real world by a wrapped repository, and the ideal world as a wrapped authenticated repository and conclude the statement by invoking the results from the previous section. The proof is given in Appendix D. □

The approach of sending assertions to upgrade a unilaterally authenticated channel to full authentication is used, for instance, in the widely used SAML protocol [15], our treatment in this section can be seen as a proof of an abstract version of SAML.

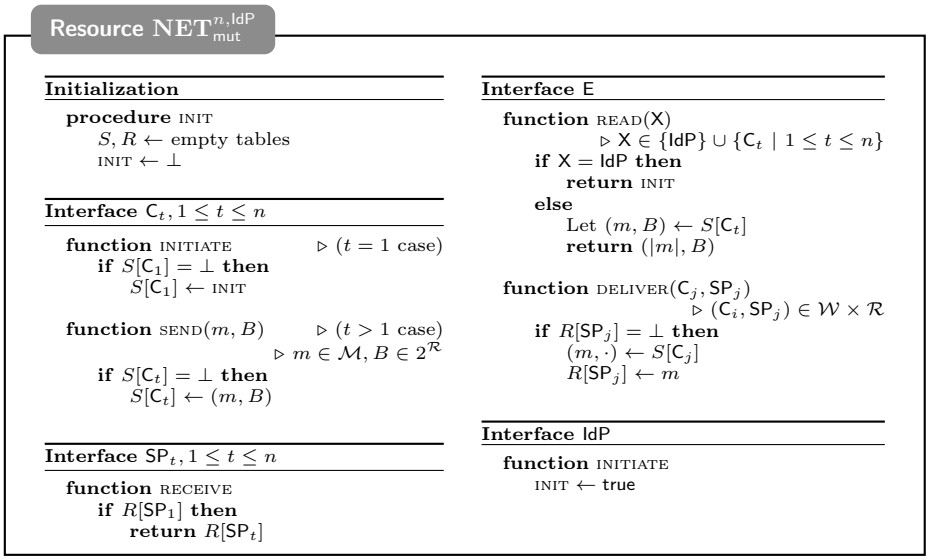

Fig. 14. The idealization of a mutually secure network between a client and a service provider established in the IdP-initiated scenario. Note that the adversarial interface E can only deliver messages between the client and the service provider.

References

1. Backes, M., Hofheinz, D.: How to break and repair a universally composable signature functionality. In: Zhang, K., Zheng, Y. (eds.) Information Security. LNCS, vol. 3225, pp. 61–72. Springer, Heidelberg (2003)
2. Backes, M., Pfitzmann, B., Waidner, M.: A universally composable cryptographic library. Cryptology ePrint Archive, Report 2003/015 (January 2003), `http://eprint.iacr.org/2003/015`
3. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) Advances in Cryptology — EUROCRYPT. LNCS, vol. 4004, pp. 409–426. Springer (2006)
4. Camenisch, J., Enderlein, R., Krenn, S., Küsters, R., Rausch, D.: Universal composition with responsive environments. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology — ASIACRYPT 2016. LNCS, vol. 10032, pp. 807–840. Springer (2016)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (October 2001), `http://eprint.iacr.org/2000/067`, version of October 2001
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings of the 42nd Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)
7. Canetti, R.: Universally composable signature, certification and authentication. In: Proceedings of CSFW 2004 (2004)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (December 2005), `http://eprint.iacr.org/2000/067`, version of December 2005

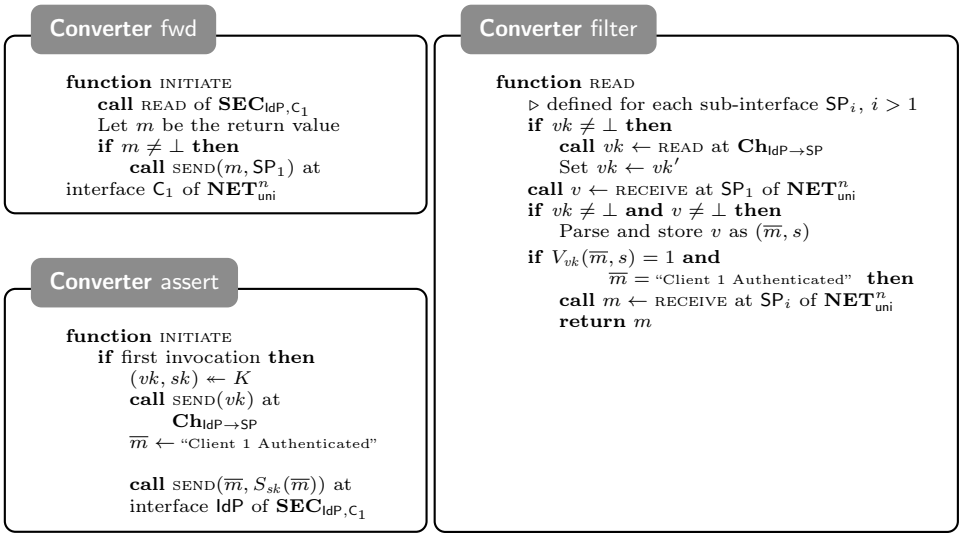

Fig. 15. The converters to authenticate a unilaterally authenticated session with the help of an identity provider.

9. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Theory of Cryptography 2006. Lecture Notes in Computer Science, Springer (2006)
10. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) Advances in Cryptology — CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 265–281. Springer (2003), http://www.springerlink.com/content/mdkp414ew1kjv918/
11. Canetti, R., Shahaf, D., Vald, M.: Universally composable authentication and key-exchange with global PKI. In: Cheng, C., Chung, K., Persiano, G., Yang, B. (eds.) Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9615, pp. 265–296. Springer (2016), http://dx.doi.org/10.1007/978-3-662-49387-8_11
12. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: Canetti, R. (ed.) Theory of Cryptography. LNCS, vol. 4948, pp. 424–441 (2008)
13. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (November 1976)
14. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen message attacks. SIAM Journal on Computing 17(2), 281–308 (April 1988)
15. Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., Maler, E.: Profiles for the Security Assertions Markup Language (SAML). OASIS Standard (March 2015)
16. Küsters, R., Tuengerthal, M.: Joint state theorems for public-key encryption and digitial signature functionalities with local computation. In: In Proc. 21st IEEE Computer Security Foundations Symposium (CSF'08 (2008)

17. Lamport, L.: Constructing digital signatures from a one-way function. Tech. Rep. CSL-98, SRI International, Menlo Park, California (October 1979)
18. Maurer, U.: Indistinguishability of random systems. In: Knudsen, L. (ed.) Advances in Cryptology — EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 110–132. Springer-Verlag (May 2002)
19. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Theory of Cryptography (2004)
20. Maurer, U., Renner, R.: Abstract cryptography. In: ICS. pp. 1–21 (2011)
21. Maurer, U., Renner, R.: From indifferentiability to constructive cryptography (and back) from indifferentiability to constructive cryptography (and back). In: Hirt, M., Smith, A. (eds.) Theory of Cryptography. LNCS, vol. 9985, pp. 3–24. Springer (2016)
22. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy. pp. 184–200. IEEE (2001)
23. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (February 1978)

Appendix

A Details of Section 3

A.1 Modeling Networks through Repositories

Recall that the generality of the construction statements involving repositories stems from the capability to transfer information is assigned to the free interface (cf. Sect. 1 and Sect. 2). However, in concrete networks these capabilities are assigned to specific parties, and in many cryptographic statements make the worst-case assumption of assigning this control over the entire network to the attacker. Such a *network resource* offers the honest interfaces to send a message m to a set B of recipients. Any message that is sent over the network is first received at an adversarial interface who can also decide whether to relay the message or to inject a new message to a receiver. This network resource is depicted in Fig. 16. It is straightforward to show that such a resource can be obtained by wrapping an insecure repository **Rep**. Intuitively, the wrapper system **W** assigns certain capabilities available at the free interface W to the attacker's interface (the delivery of messages to the receivers) and assigns certain capabilities to the honest interfaces (to initiate sending a message). Formally, we have the following statement:

Lemma 1. *For the wrapper system* **W** *defined in the proof below, the buffer resource defined in Fig. 4, and the network resource defined in Fig. 16 and all distinguishers* **D**

$$\mathbf{W}(\mathbf{Rep}_{n,\ell}) = \mathbf{Net}_{n,\ell}$$

Proof (Sketch). We first describe the wrapper system.

Upon SEND(m, B) **at interface** A_i**:** The system **W** calls WRITE(m) at interface A_i of $\mathbf{Rep}_{n,\ell}$ and subsequently calls TRANSFER$(\mathsf{A}_i, \mathsf{E}_i.r)$ at interface W of $\mathbf{Rep}_{n,\ell}$. It further stores the destination set B internally.

Upon RECEIVE **at interface** B_i**:** If a message was injected to interface B_i, then the system **W** calls READ at interface B_i of $\mathbf{Rep}_{n,\ell}$ and outputs whatever the repository outputs. Otherwise, it outputs $\perp$

Upon READ(A_t) **at interface** E**:** If there was a message input at A_t, then the system **W** calls READ at interface $\mathsf{E}_t.r$ of $\mathbf{Rep}_{n,\ell}$ to receive m, and returns (m, B), where B is the stored destination set of message m input at interface A_t before. Otherwise, output $\perp$.

Upon INJECT(m, B_t) **at interface** E**:** The system **W** calls WRITE(m) at interface $\mathsf{E}_t.w$ of $\mathbf{Rep}_{n,\ell}$ and subsequently calls TRANSFER$(\mathsf{E}_t.w, \mathsf{B}_t)$ at interface W of $\mathbf{Rep}_{n,\ell}$.

The equivalence of $\mathbf{W}(\mathbf{Rep}_{n,\ell})$ and $\mathbf{Net}_{n,\ell}$ follows by inspection. In particular, the wrapper obeys all conditions required by the repository specification. □

A.2 Authenticating a Network

We can derive a simple corollary and a straightforward modular proof that shows how to authenticate an insecure network. We therefore define three basic network converters, analogous to the three signature converters $\mathsf{setup}, \mathsf{write}$, and read. While the first converter stays identical, the second converter, which we call snd, accepts calls $\textsc{send}(m, B)$ at its outer interface, evaluates $\textsc{write}(m)$ identical to what converter write does, receives the recoverable signature s, and issues $\textsc{send}, s, B)$ to the network. Finally, we define converter rcv which is identical to read, but where the command to read a message renamed to $\textsc{receive}$ (instead of $\textsc{read}$ in case of read).

Corollary 1. *Let $n, \ell \in \mathbb{N}$. Let DSS and sim be as defined in Theorem 4 and let DSN be defined as DSS, but where the converters write and read are replaced by snd and rcv respectively, as defined above. For the simulator sim' defined in the proof, it holds that for any distinguisher $\mathbf{D}$, we derive a new distinguisher $\mathbf{D}'$ (with essentially the same efficiency) such that*

$$\Delta^{\mathbf{D}}(\mathsf{DSN}_{\mathcal{P}}\left[\mathbf{Dist}, \mathbf{Net}_{n,\ell}\right], \mathsf{sim}'^{\mathsf{E}}\mathbf{AUTH}_{n,\ell}) \\ = \Delta^{\mathbf{D}'}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}, \mathsf{sim}^{\mathsf{E}}\bar{\mathbf{a}}\mathbf{Rep}_{n,\ell}).$$

Proof (Sketch). The proof follows from a couple of simple observations. First, from the previous lemma, we can construct a wrapper system $\mathbf{W}$ such that

$$\mathbf{W}(\left[\mathbf{Dist}, \mathbf{Rep}_{n,\ell}\right]) = \left[\mathbf{Dist}, \mathbf{Net}_{n,\ell}\right].$$

In fact, the wrapper system is identical to the one above, but additionally relays back and forth inputs and outputs to the first subsystem $\mathbf{Dist}$ without modifications. Second, we can design the simulator sim' in a way such that

$$\mathbf{W}(\mathsf{sim}^{\mathsf{E}}\bar{\mathbf{a}}\mathbf{Rep}_{n,\ell}) = \mathsf{sim}'^{\mathsf{E}}\mathbf{AUTH}_{n,\ell}$$

holds. This is easy to achieve: sim' essentially behaves like the combination of the wrapper and the simulator sim on inputs at interface E and calls $\textsc{deliver}(\mathsf{A}_i, \mathsf{B}_j)$ of $\mathbf{AUTH}$ whenever sim would copy a value from interface $\mathsf{E}_i.r$ to interface $\mathsf{E}_j.w$. Since the real and ideal systems only differ in the behavior at interface E, we can establish the sequence of hybrid steps as follows:

$$\mathsf{DSN}_{\mathcal{P}}\left[\mathbf{Dist}, \mathbf{Net}_{n,\ell}\right] = \mathsf{DSN}_{\mathcal{P}}\mathbf{W}(\left[\mathbf{Dist}, \mathbf{Rep}_{n,\ell}\right]) \\ = \mathbf{W}(\mathsf{DSS}_{\mathcal{P}}\left[\mathbf{Dist}, \mathbf{Rep}_{n,\ell}\right]) \approx \mathbf{W}(\mathsf{sim}^{\mathsf{E}}\bar{\mathbf{a}}\mathbf{Rep}_{n,\ell}) = \mathsf{sim}^{\mathsf{E}}\mathbf{AUTH}_{n,\ell},$$

where the second equality holds, by definition of DSN. This proves the claim. □

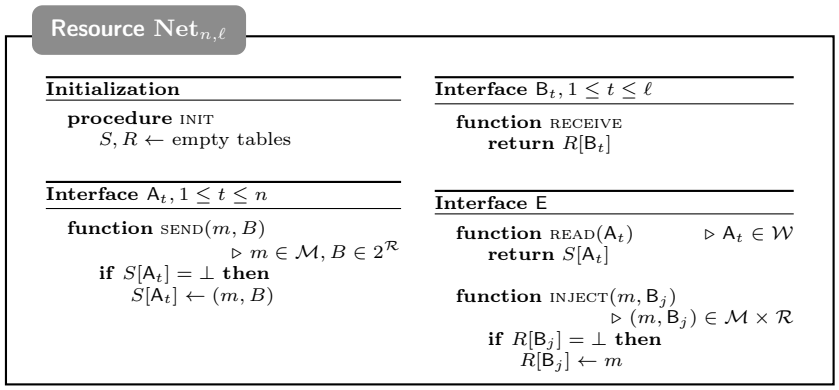

Fig. 16. The insecure network resource, where an adversarial interface E controls message delivery and where each interface can specify a destination set for its message.

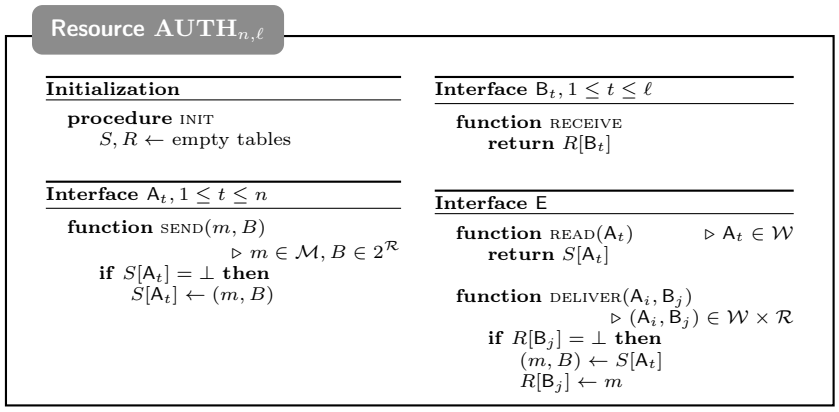

Fig. 17. The authenticated network resource, where an adversarial interface E controls message delivery but cannot inject arbitrary messages.

B Details of Section 4

B.1 Proof of Theorem 1

Proof. Consider the simulator given in Fig. 7. We introduce the shorthand notation $\mathbf{S}_{n,\ell} := \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}$. The simulator sim accesses all the capabilities provided at the adversarial interfaces (by merging them into the single interface E) and provides at its outer interface the simulated capabilities (accessible at the appropriate interfaces) of the real system. We now argue that all queries of a distinguisher $\mathbf{D}$ are answered consistently by the systems $\mathbf{S}_{n,\ell}$ and $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$, we observe that in random experiments $\mathbf{D}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})$ and $\mathbf{D}(\mathbf{S}_{n,\ell})$, the behavior is identical, unless a query $\textsc{write}(s')$ is made at interface E for some signature s' that satisfies $V_{vk}(m', s') = 1$ for some message m' that is subsequently part of a call $\textsc{verify}(m')$ at the interface where s' was transferred to. Let us denote this event in the real-world random experiment by $\mathcal{E}$.

If we prove that the systems behave equivalently until event $\mathcal{B}$ occurs, then we can bound the distinguishing advantage of $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ and $\mathbf{S}_{n,\ell}$ by bounding the probability of event $\mathcal{E}$. As discussed above, this probability can be bounded by the success probability of an (efficient) adversary $\mathbf{A}$ in breaking the security of the DSS.

We first elaborate on the other queries being handled consistently by the two systems; for the structure, recall the depictions in Fig. 5 and Fig. 6. We now argue for each query individually.

$\mathsf{S}.\textsc{setup}$**:** This query has no effect in the ideal system, but in the real system it makes the key sk available in $\bar{\mathbf{a}}\mathbf{Rep}^{\{\mathsf{S}\}}_{\mathcal{W}}$ and the key vk available in $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\{\mathsf{S}\}}_{\mathcal{E}_r\cup\mathcal{R}}$. No output to $\mathbf{D}$.

$\mathsf{W}.\textsc{transfer}(\mathsf{S},\mathsf{A})$ **for** $\mathsf{A} \in \mathcal{W}$**:** Requires that $\mathsf{S}.\textsc{setup}$ has been called. No effect in the ideal system, but in the real system it makes the key sk accessible to writer A. No output to $\mathbf{D}$.

$\mathsf{W}.\textsc{transfer}(\mathsf{S},\mathsf{X})$ **for** $\mathsf{X} \in \mathcal{E}_r$ **at** $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\{\mathsf{S}\}}_{\mathcal{E}_r\cup\mathcal{R}}$**:** Requires that $\mathsf{S}.\textsc{setup}$ has been called. No effect in the ideal system, but in the real system it makes the key vk accessible to adversarial reader X. No immediate output to $\mathbf{D}$.

$\mathsf{Y}.\textsc{copy}(\mathsf{X})$ **for** $\mathsf{X} \in \mathcal{E}_r$ **and** $\mathsf{Y} \in \mathcal{E}_c$ **at** $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\{\mathsf{S}\}}_{\mathcal{E}_r\cup\mathcal{R}}$**:** Requires that before query $\mathsf{W}.\textsc{transfer}(\mathsf{S},\mathsf{X})$ has been made. No effect in the ideal system, but in the real system it copies the key vk to interface Y. No immediate output to $\mathbf{D}$.

$\mathsf{W}.\textsc{transfer}(\mathsf{X},\mathsf{B})$ **for** $\mathsf{B} \in \mathcal{R}$ **at** $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\{\mathsf{S}\}}_{\mathcal{E}_r\cup\mathcal{R}}$**:** Requires that either $\mathsf{S}.\textsc{setup}$ resp. $\mathsf{X}.\textsc{copy}$ has been called. No effect in the ideal system, but in the real system it makes the key vk accessible to reader B. No output to $\mathbf{D}$.

$\mathsf{A}.\textsc{write}(m)$ **for** $\mathsf{A} \in \mathcal{W}$**:** Requires that $\mathsf{W}.\textsc{transfer}(\mathsf{S},\mathsf{A})$ has been called. Enters m into the resource in the ideal system, obtains vk, computes the signature, and enters it into $\mathbf{Rep}^{\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r}$ in the real system. No immediate output to $\mathbf{D}$.

$\mathsf{X}.\textsc{read}$ **for** $\mathsf{X} \in \mathcal{E}_r$ **at** $\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\{\mathsf{S}\}}_{\mathcal{E}_r\cup\mathcal{R}}$**:** Requires that $\mathsf{W}.\textsc{transfer}(\mathsf{S},\mathsf{X})$ has been called. In the real system, returns the verification key vk. In the ideal system, sim returns the simulated verification key. This has the same distribution.

$\mathsf{W}.\textsc{transfer}(\mathsf{A},\mathsf{X})$ **for** $\mathsf{X} \in \mathcal{R} \cup \mathcal{E}_r$**:** Requires that $\mathsf{A}.\textsc{write}(m)$ has been called. In the real system, makes the generated signature available to X. In the ideal system, makes m available to X. No immediate output to $\mathbf{D}$.

$\mathsf{X}.\textsc{read}$ **for** $\mathsf{X} \in \mathcal{E}_r$ **at** $\mathbf{Rep}^{\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r}$**:** Requires that $\mathsf{W}.\textsc{transfer}(\mathsf{A},\mathsf{X})$ has been queried for $\mathsf{A} \in \mathcal{W}$. (Adversarial writers in $\mathcal{E}_w$ are explicitly excluded.) In the real system, outputs the signature that has been made available in the repository. In the ideal system, the simulator sim checks whether the query has been made before, answers consistently in that case, and otherwise generates a new signature using the internal key. This computation is done exactly in the same way, and therefore the returned signature also has the same distribution.

$\mathsf{X}.\textsc{write}(s)$ **for** $\mathsf{X} \in \mathcal{E}_w$**:** In the real system, this enters the value s into the repository $\mathbf{Rep}^{\mathcal{W}\cup\mathcal{E}_w}_{\mathcal{R}\cup\mathcal{E}_r}$ and has no immediate output. In the ideal system, the simulator sim processes the message, and checks for which messages that it has already received, the signature verifies. In case no message can be verified, the simulator inserts $\perp$ at its copier interface. In the other case, for each message, received at reader-subinterface i that verifies successfully with the given signature string, the simulator calls $\textsc{copy}(i)$ at X to insert this message into the copier buffer (since all these messages can be verified w.r.t. the signature string s.)

$\mathsf{W}.\textsc{transfer}(\mathsf{X},\mathsf{B})$**:** This is only valid if $\mathsf{X}.\textsc{write}(s)$ was called before, has the same effect in both cases.

We introduce the reduction system $\mathbf{C}$ that emulates the real world view towards any distinguisher $\mathbf{D}$ by accessing the oracles of $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$ such that for any distinguisher $\mathbf{D}$, $\Delta^{\mathbf{D}}(\mathbf{CG}_{\Sigma}^{\mathsf{EU\text{–}CMA}}, \mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}) = 0$. During that emulation, $\mathbf{C}$ tries to extract a forgery from the interaction with the distinguisher. The main challenge for the reduction is to make sure that it does only query the signing oracle on messages that definitely cannot be forgery candidates. Formally, the reduction system $\mathbf{C}$ emulates one setup interface, n writer-interfaces A_t, ℓ receiver-interfaces B_t, and $n + \ell$ adversarial read and write interfaces $\mathsf{E}_t.r/w$ and one free interface W to $\mathbf{D}$. $\mathbf{C}$ first initializes two empty tables R and L. Then, $\mathbf{C}$ receives the verification key vk from $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$ and stores it internally. Furthermore, it answers all queries by $\mathbf{D}$ accordingly to the description of the real system, with the only difference that signatures are computed through the game. When $\mathbf{C}$ detects a signature forgery, it outputs this forgery to the game.

For any distinguisher $\mathbf{D}$, we define the the adversary $\mathbf{A} := \mathbf{DC}$ against the game $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$ and conclude that

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}, \mathbf{S}_{n,\ell}) &\leq \mathsf{Pr}^{\mathbf{D}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[\mathcal{E}] = \mathsf{Pr}^{\mathbf{DCG}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[\mathcal{E}] \\ &= \mathsf{Pr}^{\mathbf{DCG}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[\textsc{won} = 1] = \mathsf{Pr}^{\mathbf{AG}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[\textsc{won} = 1].\end{aligned}$$

□

B.2 Proof of Theorem 3

The theorem directly follows from the following two lemmas: Lemma 2 states that if the output of the key generation algorithm K as defined in Sect. 4.4 is not $(\perp, \perp)$, then any adversary against the derived signature scheme can be transformed into a distinguisher for the real and ideal systems (for any simulator).

Lemma 2. *Let $n, \ell \in \mathbb{N}$, let* DSS *be as defined in Theorem 3 for arbitrary converters* setup, write, *and* check, *and let the digital signature scheme $\Sigma = (K, S, V)$ be defined as in Sect. 4.4. We present an (efficient) reduction that transforms any adversary for $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$, that asks at most n queries to* Sign *and ℓ queries to* Forge*, into a distinguisher $\mathbf{D}(\mathbf{A})$, such that for all simulators* sim*,*

$$\Gamma^{\mathbf{A}}(\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}) \leq \mathsf{Pr}[(\perp, \perp) \leftarrow K] + \Delta^{\mathbf{D}(\mathbf{A})}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}).$$

Proof. We define the shorthand notation $\mathbf{S}_{n,\ell} := \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}$. We define a reduction system $\mathbf{C}$ that is given access to the interfaces of either system $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ or $\mathbf{S}_{n,\ell}$, and provides one additional outside interface. At that outside interface, $\mathbf{C}$ simulates the oracles Sign and Forge of game $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$. First, the system $\mathbf{C}$ initializes an internal variable won to 0. Then, it activates all interfaces of its connected system and queries setup at interface C. Subsequently, $\mathbf{C}$ calls transfer$(\mathsf{C}, \mathsf{E}_1.r)$, transfer$(\mathsf{C}, \mathsf{B}_i)$, and transfer$(\mathsf{C}, \mathsf{A}_j)$ for $i = 1 \ldots \ell$, $j = 1 \ldots n$. Then, $\mathbf{C}$ queries read to (the possibly simulated) resource **Dist** at interface $\mathsf{E}_1.r$ and outputs at the outside interface whatever is output by $\mathsf{E}_1.r$.

It further answers the following queries by an adversary $\mathbf{A}$:

On the ith query Sign(m)**:** Upon this query, $\mathbf{C}$ queries write(m) at interface A_i and subsequently transfer$(\mathsf{A}_i, \mathsf{E}_i.r)$. Then, retrieve the value s by querying read at interface $\mathsf{E}_i.r$. Finally, $\mathbf{C}$ outputs the pair (m, s) at its outer interface. If no value is output at interface $\mathsf{E}_i.r$, then $\mathbf{C}$ outputs (m, m).

On the ith query Forge(m, s)**:** On input a possible forgery $\mathbf{C}$ queries write(s) at interface $\mathsf{E}_i.w$ of its connected system. Then, $\mathbf{C}$ queries transfer$(\mathsf{E}_i.w, \mathsf{B}_i)$ to give B_i access to the signature string s. Next, $\mathbf{C}$ queries verify(m) at interface B_i to receive either the the indication `Success` or `Fail`. If the message m is successfully verified and has not been queried to Sign before, then won is set to 1. In any case won is output at the outside interface.

We first note that in the key generation process of $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$ for Σ defined above, the function setup.setup defines the signing and verification key. It is this function that defines the values that converter setup writes to the distribution resource in the real world upon setup at interfaces C. So the probability distribution of the pair (vk, sk) output by K is identical to the distribution of the values written to **Dist** in system $\mathbf{C}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})$.

In the random experiment of any adversary that asks at most n queries to its signature oracle and ℓ queries to its forgery oracle, the input-output behavior of $\mathbf{C}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})$ and $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$ are identical given that $vk \neq \perp$ and $sk \neq \perp$ during

the key generation process. This follows from the definition of the algorithms of Σ: The signing algorithm S and the verification algorithm V execute the same converter functions as are executed in the system $\mathbf{C}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})$ on an input by the adversary. And in both cases, if no signature is generated for some message m it is set to the default value m. This is sufficient to build a distinguisher based on an adversary $\mathbf{A}$.

Claim: *Let $n, \ell \in \mathbb{N}$. From any game winner $\mathbf{A}$ for $\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}$, that asks at most n queries to* SIGN *and ℓ queries to* FORGE*, we construct a distinguisher $\mathbf{D}(\mathbf{A})$ such that for any simulator* sim*,*

$$\Gamma^{\mathbf{A}}(\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}) \leq \Delta^{\mathbf{D}(\mathbf{A})}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}) + \mathsf{Pr}[(\bot,\bot) \leftarrow K].$$

Proof: Consider the random experiment in which an adversary $\mathbf{A}$ interacts with system $\mathbf{C}(\mathbf{T})$, where $\mathbf{T} \in \{\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}, \mathbf{S}_{n,\ell}\}$. For this random experiment, let W be the random variable that takes on the value of WON at the end of the experiment. Let further be F the binary random variable that takes on the value 1 if, after the invocation of setup, at least one of the values stored in **Dist** is undefined (and $F = 0$ otherwise).

The actual distinguisher $\mathbf{D}(\mathbf{A})$ connected to a system $\mathbf{T} \in \{\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}, \mathbf{S}_{n,\ell}\}$ works as follows: it lets $\mathbf{A}$ interact with system $\mathbf{C}(\mathbf{T})$ and, after $\mathbf{A}$ has finished, outputs the value of WON as its decision bit[7]

We can therefore conclude that

$$\begin{aligned}
\mathsf{Pr}[\mathbf{D}(\mathbf{A})(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}) = 1] &= \mathsf{Pr}^{\mathbf{AC}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[W = 1] \\
&= \mathsf{Pr}^{\mathbf{AC}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[F = 0] \cdot \mathsf{Pr}^{\mathbf{AC}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[W = 1 | F = 0] \\
&+ \mathsf{Pr}^{\mathbf{AC}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[F = 1] \cdot \mathsf{Pr}^{\mathbf{AC}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[W = 1 | F = 1] \\
&\leq \mathsf{Pr}^{\mathbf{AG}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[F = 0] \cdot \mathsf{Pr}^{\mathbf{AG}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[W = 1 | F = 0] \\
&\quad + \mathsf{Pr}^{\hat{\mathbf{A}}\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[F = 1] \cdot \underbrace{\mathsf{Pr}^{\hat{\mathbf{A}}\mathbf{G}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[W = 1 | F = 1]}_{=1} \\
&\leq \mathsf{Pr}^{\mathbf{AG}_{\Sigma}^{\mathsf{EU\text{–}CMA}}}[\mathrm{WON} = 1] + \mathsf{Pr}[(\bot,\bot) \leftarrow K], \qquad (3)
\end{aligned}$$

where $\hat{\mathbf{A}}$ is the adversary that wins the game with probability 1 by a single query (m, m) to oracle Forge in case $F = 1$. Note that by definition, the scheme does not provide any security whenever this condition occurs.

On the other hand, $\mathsf{Pr}[\mathbf{D}(\mathbf{A})(\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell}) = 1] = 0$, since no new message can be written at any interface $\mathsf{E}_i.w$ of $\mathbf{aRep}_{n,\ell}$. This yields the claim and concludes the lemma. ■ □

The second lemma states that for the key generation algorithm K defined in Sect. 4.4, the probability that $(\bot,\bot)$ is returned is a lower bound for the advantage in distinguishing the real and ideal systems.

[7] This means that the output bit 1 indicates that the connected system is the real system.

Lemma 3. *Let $n, \ell \in \mathbb{N}$, let $\mathbf{R}_{n,\ell}$ be as defined above for converters* setup, write, *and* check *and let the digital signature scheme $\Sigma = (K, S, V)$ be defined as in Sect. 4.4. We construct (efficient) distinguishers $\mathbf{D}_i$, $i = 1 \dots 5$, such that for all simulators* sim,

$$\mathsf{Pr}[(\bot, \bot) \leftarrow K] \leq \sum_{i=1}^{4} \Delta^{\mathbf{D}_i}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}, \mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell})$$

In particular, if there exists a simulator sim *such that $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ and $\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell}$ are indistinguishable, then the output of the key generation algorithm of Fig. 9 is defined with overwhelming probability.*

Proof. Let us consider an execution of algorithm K and let us define the events $\mathcal{E}_1 := sk = \bot$ and $\mathcal{E}_2 := vk = \bot$ and let $\mathcal{E} := \mathcal{E}_1 \cup \mathcal{E}_2$. By definition of algorithm K, we immediately observe that $\mathsf{Pr}[\mathcal{E}_i]$ is equal to the probability that, in the real system $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ upon calling SETUP at C, converter setup does not define the respective values by an appropriate write-query to **Dist**. We now show that the occurrence of either event leads to lower bounds on the security condition.

To achieve this, let us consider the following real-world random experiment (with system $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$). For further reference, we denote the experiment by *Exp*. First, we call SETUP at interface C. Subsequently, we call TRANSFER$(\mathsf{C}, \mathsf{E}_1.r)$, TRANSFER$(\mathsf{C}, \mathsf{B}_i)$, and TRANSFER$(\mathsf{C}, \mathsf{A}_j)$ for $i = 1 \dots \ell$, $j = 1 \dots n$ to distribute the setup values (,i.e., keys). Then, we choose a uniformly random message $\overline{m}$ and input it at interface A_1. Let us denote by S the random variable that takes on the value of the output of write on this write-query. Afterward, we call TRANSFER$(\mathsf{A}_1, \mathsf{B}_1)$.[8] Finally, we call VERIFY$(\overline{m})$ at interface B_1. Let R be the random variable that takes on the value output by check. We decompose the probability of event $\mathcal{E}$ in *Exp* as follows:

$$\mathsf{Pr}^{Exp}[\mathcal{E}] = \underbrace{\mathsf{Pr}^{Exp}[\mathcal{E}] \cdot \mathsf{Pr}^{Exp}[S = \bot \mid \mathcal{E}]}_{\alpha} \tag{4}$$

$$+ \underbrace{\mathsf{Pr}^{Exp}[\mathcal{E}] \cdot \mathsf{Pr}^{Exp}[S \neq \bot \mid \mathcal{E}] \cdot \mathsf{Pr}^{Exp}[R = \mathtt{Fail} \mid \mathcal{E}, S \neq \bot]}_{\beta} \tag{5}$$

$$+ \sum_{i=1}^{2} \underbrace{\mathsf{Pr}^{Exp}[\mathcal{E}_i] \cdot \mathsf{Pr}^{Exp}[S \neq \bot \mid \mathcal{E}_i] \cdot \mathsf{Pr}^{Exp}[R = \mathtt{Success} \mid \mathcal{E}_i, S \neq \bot]}_{\gamma_i}. \tag{6}$$

Let $\mathbf{D}_1$ be the distinguisher that implements the strategy of *Exp* with the following exception: instead of calling VERIFY$(\overline{m})$ in the last step, $\mathbf{D}_1$ flips a uniform bit b and only if $b = 0$ it calls VERIFY$(\overline{m})$; if $b = 1$ $\mathbf{D}_1$ calls VERIFY(m') for with m' chosen uniformly at random from $\mathcal{M} \setminus \{\overline{m}\}$. $\mathbf{D}_1$ outputs 1 as its

[8] Note that this input bypasses any adversarial influence, as the output by the honest writer is directly given to the honest reader.

decision bit if either $R = \texttt{Success}$ and $b = 1$ or if $R = \texttt{Fail}$ and $b = 0$. In any other case, $\mathbf{D}_1$ outputs 0.

We see that the random variable S in the experiment between $\mathbf{D}_1$ and $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ is identically distributed as in experiment *Exp*. Hence, in the real world, with probability α, converter check attached at interface B_1 does not learn any information about $\overline{m}$, because $\overline{m}$ was chosen after the SETUP query and no information is transmitted within the repository since $S = \perp$. In this case, the probability that $\mathbf{D}_1$ outputs 1 is $\frac{1}{2}$. We observe that in the ideal world, i.e., in $\mathbf{D}_1(\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell})$, the output $R = \texttt{Success}$ is observed whenever $b = 0$ and the output $R = \texttt{Fail}$ is observed whenever $b = 1$. Hence,

$$\Delta^{\mathbf{D}_1}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}) \geq \frac{\alpha}{2}.$$

Let $\mathbf{D}_2$ be the distinguisher that implements the strategy in experiment *Exp* and which outputs 1 if and only if $R = \texttt{Fail}$. We directly see that in the experiment between $\mathbf{D}_2$ and $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$, converter check outputs $\texttt{Fail}$ with probability β, whereas in the random experiment $\mathbf{D}_1(\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell})$ with any simulator, the probability of an output $\texttt{Fail}$ at interface B_1 is 0 by definition:

$$\Delta^{\mathbf{D}_1}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}) \geq \beta.$$

We now show that the third term constitutes a security issue in both cases:

Event $\mathcal{E}_1$: We define a third distinguisher $\mathbf{D}_3$ as follows: it first chooses a uniformly random message $\overline{m}$, activates all interfaces, and then calls WRITE($\overline{m}$) at interface A_1, subsequently retrieves the value of S by calling TRANSFER($\mathsf{A}_1, \mathsf{E}_1.r$) and after that calls READ at interface $\mathsf{E}_1.r$. Only then, $\mathbf{D}_3$ calls SETUP and distributes the setup values as in *Exp*. Then, $\mathbf{D}_3$ calls WRITE(S) at interface $\mathsf{E}_1.w$ and calls TRANSFER($\mathsf{E}_1.w, \mathsf{B}_1$) to give the receiver interface access to the value S.

Finally, $\mathbf{D}_3$ queries VERIFY($\overline{m}$) at interface $\mathsf{B}_1.r$. $\mathbf{D}_2$ outputs 1 if the output is $\texttt{Success}$ and outputs 0 otherwise. We have in particular

$$\mathsf{Pr}^{\mathbf{D}_3(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[\mathcal{E}_1 \mid S \neq \perp] = \mathsf{Pr}^{Exp}[\mathcal{E}_1]$$
$$\mathsf{Pr}^{\mathbf{D}_3(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[S \neq \perp] = \mathsf{Pr}^{Exp}[S \neq \perp \mid \mathcal{E}_1],$$

since in both experiments, converter write is invoked when the value val_s stored in **Dist** for converter write is $\perp$. Note that this in particular means, this behavior has to be in the specification, since the behavior of *Exp* obeys all the conditions. We conclude that the probability of an output other than $\texttt{Fail}$ at interface B_1 in the real world is exactly γ_1. On the other hand, in the ideal world, i.e., in $\mathbf{D}_2(\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell})$, there cannot be any output other than $\texttt{Fail}$ for this distinguishing strategy. We get

$$\Delta^{\mathbf{D}_3}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}) \geq \gamma_1.$$

Event $\mathcal{E}_2$: We define a fourth distinguisher $\mathbf{D}_4$ as follows: instead of querying SETUP, $\mathbf{D}_4$ internally runs setup.SETUP to simulate the public and the private

values. Let the private value be denoted as sk. Then, $\mathbf{D}_4$ chooses a message $\overline{m}$ uniformly at random and simulates the output S of converter write by evaluating $\mathsf{write}.\textsc{write}(\overline{m})$ using sk as the emulated value stored in $\mathbf{Dist}$. $\mathbf{D}_4$ then activates all interfaces of its connected system and queries $\textsc{write}(S)$ at interface $\mathsf{E}_1.w$ and $\textsc{transfer}(\mathsf{E}_1.w, \mathsf{B}_1)$. Finally, $\mathbf{D}_4$ queries $\textsc{verify}(\overline{m})$ at interface B_1 and outputs 1 as its decision bit if and only if the output R is $\texttt{Success}$. We observe that in particular,

$$\mathsf{Pr}^{\mathbf{D}_4(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[\mathcal{E}_2] = \mathsf{Pr}^{Exp}[\mathcal{E}_2]$$
$$\mathsf{Pr}^{\mathbf{D}_4(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell})}[R = \texttt{Success} \mid \mathcal{E}_2, S \neq \bot] = \mathsf{Pr}^{Exp}[R = \texttt{Success} \mid \mathcal{E}_2, S \neq \bot].$$

The first equation follows from the fact that $\mathbf{D}_4$ internally imitates the setup-process and hence the probability that the (public) value is equal to $\bot$ is the same as in *Exp*. The second equality follows since the value that converter check retrieves from $\mathbf{Dist}$ is $\bot$ in both systems and hence the views are identical (and again this shows that the behavior for this resource has to be in the specification). We conclude that the probability of an output $\texttt{Success}$ at interface $\mathsf{B}_1.r$ in the real world is exactly γ_2. On the other hand, in the ideal world, i.e., in $\mathbf{D}_4(\mathsf{sim}^{\mathsf{E}}\mathbf{aRep})$, there cannot be any output other than $\texttt{Fail}$, since the message $\overline{m}$ has never been written to the repository. We get

$$\Delta^{\mathbf{D}_4}(\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell},\ \mathsf{sim}^{\mathsf{E}}\,\mathbf{aRep}_{n,\ell}) \geq \gamma_2.$$

This concludes the proof. □

C Details of Section 6

C.1 Proof of Theorem 5

Proof. Let us introduce some shorthand notation. Let

$$\mathbf{R} := \mathsf{issue}^{\mathsf{I}}\ \mathsf{reg}^{\mathsf{S}_1} \ldots \mathsf{reg}^{\mathsf{S}_\ell} \mathsf{rel}^{\mathsf{C}_1} \ldots \mathsf{rel}^{\mathsf{C}_\ell}\ [\mathbf{Ch}_{\mathsf{I}\rightarrow}\mathbf{Ch}_{\mathsf{I}\leftarrow}, \mathbf{Net}]$$

and let

$$\mathbf{S} := \tilde{\mathsf{sim}}^{\mathsf{E}}\mathbf{Reg}_{\mathcal{S},\mathcal{C}}.$$

Let us further define the party set

$$\mathcal{P} := \{\mathsf{I}_0, \mathsf{I}_1, \ldots, \mathsf{I}_n, \mathsf{S}_1, \ldots, \mathsf{S}_\ell, \mathsf{C}_1, \ldots, \mathsf{C}_\ell\}$$

Let further $\mathsf{DSS} := (\mathsf{setup}, \mathsf{write}, \ldots, \mathsf{write}, \mathsf{read}, \ldots, \mathsf{read})$ with one converter setup, n copies of write (for the issuer), and 2ℓ copies of converter read (for the submitters and consumers). We design a wrapper system $\mathbf{W}$ such that

$$\mathbf{R} = \mathbf{W}(\mathsf{DSS}_{\mathcal{P}}\left[\mathbf{Dist}, \mathsf{relay}^{\mathsf{W}}\mathbf{Rep}^{\emptyset,\{\mathsf{I}_1,\ldots,\mathsf{I}_n\}\cup\mathcal{E}_w}_{\mathcal{S}\cup\mathcal{C}\cup\mathcal{E}_r,\emptyset,\emptyset}\right])$$

and

$$\mathbf{S} = \mathbf{W}(\mathsf{sim}^{\mathsf{E}}\mathsf{relay}^{\mathsf{W}}\bar{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_w,\{\mathsf{I}_1,\ldots,\mathsf{I}_n\}}_{\mathcal{S}\cup\mathcal{C}\cup\mathcal{E}_r,\emptyset,\mathcal{S}\cup\mathcal{C}}),$$

where sim is the simulator from Theorem 1, and relay is the converter from Sect. 5 that implements information transfer between receiver-interfaces of the repository. These two equations are sufficient to conclude the statement, since a distinguisher for $\mathbf{R}$ and $\mathbf{S}$ can be translated into a distinguisher for the wrapped systems (by emulating the wrapper).

Let us construct such a wrapper $\mathbf{W}$. The wrapper has to answer to the following queries by a distinguisher $\mathbf{D}$:

On ISSUE **at interface** I**:** The wrapper system first calls SETUP at interface I_0 of the sub-system and completes the setup by calling TRANSFER$(\mathsf{I}_0, \mathsf{Y})$ for reader and writer interfaces of the repository.
Next, the wrapper calls WRITE(x_i, i) at each interface I_i, where x_i is the value that the issuer received from party with interface number i. Again, wrapper $\mathbf{W}$ distributes all values by calling TRANSFER$(\mathsf{I}_i, \mathsf{X})$ at interface W for all reader interfaces X of the repository.

On REGISTER(x) **at interface** S**:** The wrapper internally stores the pair (x, i) for later reference where i is the interface number of interface S. The wrapper accepts only one such query per interface.

On READ **at interface** $\mathsf{X} \in \mathcal{S} \cup \mathcal{C}$**:** If a value has been transferred to interface X already, then this input is translated into a call READ to the reader interface X of the repository (which returns a registered pair). Otherwise, the return value is defined to be $\perp$.

On RELAYTO(Y) **at interface** $\mathsf{X} \in \mathcal{S} \cup \mathcal{C} \cup \{\mathsf{E}_1, \ldots, \mathsf{E}_l\}$**:** As long as the SETUP has not been called at the subsystem, the wrapper does not relay any input. If SETUP has been called already, then the query is translated to a call TRANSFER(X, Y) at interface W of the repository (recall that we can transfer values between any two entities of the repository).

On SEND(m) **at interface** E_i **(for system** $\mathbf{Ch}_{\mathsf{I}\leftarrow}$**):** The wrapper internally stores the pair (x, k) for later reference where k is the interface number of interface E_i. The wrapper accepts only one such query per interface

On READ **at interface** E_i **(for system** $\mathbf{Ch}_{\mathsf{I}\rightarrow}$**):** This input is translated into a call READ to the reader interface $\mathsf{E}_i.r$ of the repository if a value has been transferred to interface $\mathsf{E}_i.r$ (and returns a signature; recall that no converter is attached at the dishonest interfaces E_i).

On SEND(m) **at interface** E **(for system Net):** This input directly corresponds to an adversarial write-operation to the repository. Hence, the wrapper calls WRITE(m) at some interface $\mathsf{E}_i.w$ to write the value m into the repository.

It is straightforward to verify that the wrapper can mimic the real world by translating the queries into queries to the repository and thereby obeys the conditions required by the specifications (we assume that no further conditions occur, i.e., that the specification does not demand any further requirements from the order of inputs): in fact, the repository captures an abstract application of the

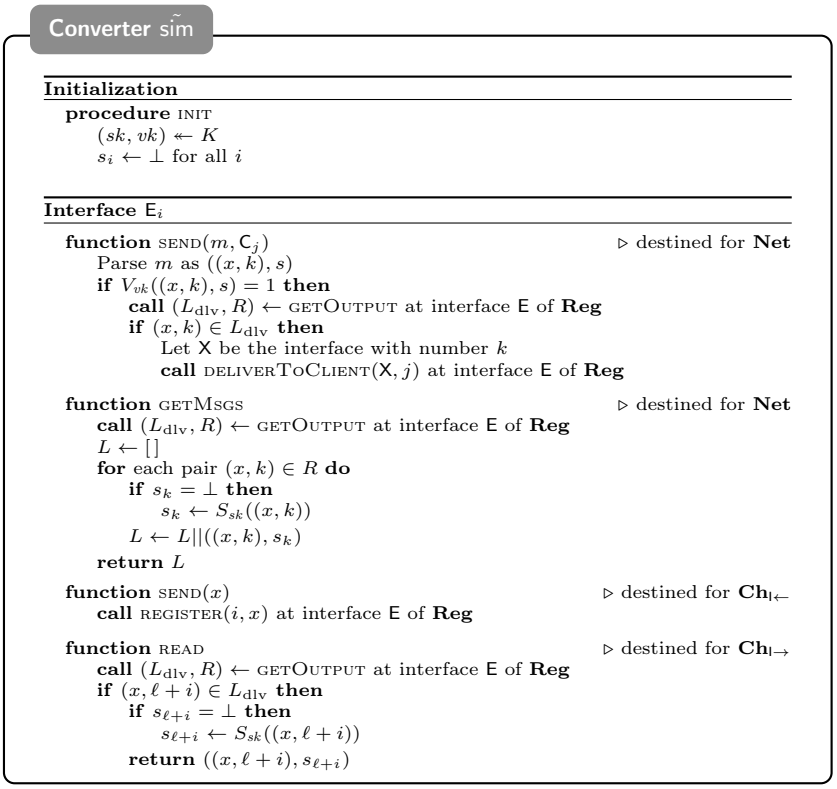

Fig. 18. Simulator for the construction of a registration service in Theorem 5.

signature scheme to sign messages and distribute the signature strings. Since this is all what happens in this application of signatures to establish a registration service, it is not surprising that we can represent the real world as an application of this repository.

On the other hand, replacing the subsystem by the authenticated repository (and the simulator) rules out that the adversary can write arbitrary values into the repository. This, translated to our setting, simply means that the adversary can not send any message m to any of the consumers (or servers), which was not sent to the issuer first. This is exactly what simulator $\tilde{\mathsf{sim}}$ verifies: it verifies whether the signature matches and only then relays an input. The relay-command stands in a one-to-one correspondence to the copy-commands of simulator sim from Theorem 1: it verifies the signature and only then writes it into the repository by using the copy-command (hence relaying a signature).

Thus, the only way to distinguish **R** and **S** is to distinguish the sub-systems of the wrapper **W**, which is upper bounded by Theorem 1 and the result from Sect. 5. The theorem follows. □

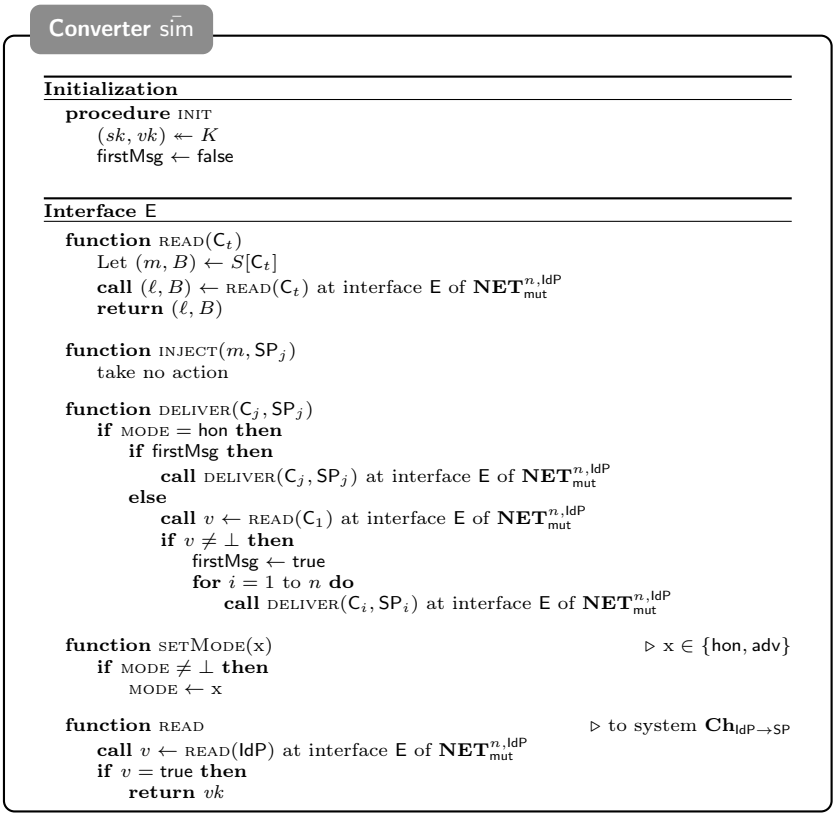

Fig. 19. Simulator for the session-authentication construction in Theorem 6.

D Details of Section 7

D.1 Proof of Theorem 6

Proof (Sketch). We show that attaching the protocol fwd, assert and filter to interfaces C_1, IdP, and SP (where SP has n sub-interface SP_i) of the assumed setting can be represented as a wrapped repository as in the proof of Theorem 5. More specifically, we obtain for a wrapper $\mathbf{W}$ defined below,

$$\begin{aligned}&\mathsf{fwd}^{\mathsf{C}_1}\mathsf{assert}^{\mathsf{IdP}}\mathsf{filter}^{\mathsf{SP}}\,[\mathbf{Ch}_{\mathsf{IdP}\to\mathsf{SP}}, \mathbf{SEC}_{\mathsf{IdP},\mathsf{C}_1}, \mathbf{NET}^{n}_{\mathsf{uni}}]\\ &= \mathbf{W}(\mathsf{setup}^{\mathsf{IdP}}\mathsf{write}^{\mathsf{IdP}}\mathsf{read}^{\mathsf{SP}_1}\,\Big[\mathbf{Dist}, \mathbf{Rep}^{\emptyset,\{\mathsf{IdP},\mathsf{E}\}}_{\{\mathsf{SP}_1\},\emptyset,\emptyset}, \mathbf{NET}^{n-1}_{\mathsf{uni}}\Big]),\end{aligned}$$

and

$$\begin{aligned}&\bar{\mathsf{sim}}^{\mathsf{E}}\mathbf{NET}^{n,\mathsf{IdP}}_{\mathsf{mut}}\\ &= \mathbf{W}(\Big[\mathsf{sim}^{\mathsf{E}}\bar{\mathbf{a}}\mathbf{Rep}^{\{\mathsf{E}\},\{\mathsf{IdP}\}}_{\{\mathsf{SP}_1\},\emptyset,\emptyset}, \mathbf{NET}^{n-1}_{\mathsf{uni}}\Big]),\end{aligned}$$

where the signature converters and simulator sim are again as in Sect. 4.1. We now sketch the behavior of the wrapper system $\mathbf{W}$ on any possible input by a distinguisher that fulfills the above two equations.

On INITIATE **at interface** IdP**:** The wrapper calls the function SETUP at interface IdP of the sub-system and subsequently calls TRANSFER$(\mathsf{IdP}, \mathsf{SP}_1)$ and TRANSFER$(\mathsf{IdP}, \mathsf{E})$ at interface W of **Dist** to complete the setup as it would happen in the real world via $\mathbf{Ch}_{\mathsf{IdP}\to\mathsf{SP}_1}$.
Then, the wrapper calls the function WRITE(m) at interface IdP of resource $\mathbf{Rep}^{\emptyset,\{\mathsf{IdP},\mathsf{E}\}}_{\{\mathsf{SP}_1\},\emptyset,\emptyset}$, where $\overline{m}$ = "Client 1 Authenticated" as in the protocol assert.

On READ **at interface** E **(for system** $\mathbf{Ch}_{\mathsf{IdP}\to\mathsf{SP}}$**):** The wrapper calls READ at the same interface of sub-system **Dist** only if a value has been transferred before to that interface, and otherwise outputs $\perp$.

On READ **at interface** SP_1**:** The wrapper checks whether the service provider has received the assertion. Only then, the wrapper will produce output for the service provider at the sub-interfaces SP_i for $i > 1$.

On INJECT(m, SP_1) **at interface** E **(for system** $\mathbf{NET}^n_{\mathsf{uni}}$**):** If the distinguisher set the mode of $\mathbf{NET}^n_{\mathsf{uni}}$ such that MODE $= \mathsf{adv}$, then the wrapper calls the function WRITE(m) at interface E of resource $\mathbf{Rep}^{\emptyset,\{\mathsf{IdP},\mathsf{E}\}}_{\{\mathsf{SP}_1\},\emptyset,\emptyset}$ and subsequently calls TRANSFER$(\mathsf{E}, \mathsf{SP}_1)$ at interface W of $\mathbf{Rep}^{\emptyset,\{\mathsf{IdP},\mathsf{E}\}}_{\{\mathsf{SP}_1\},\emptyset,\emptyset}$ to mimic the transfer of the first message to SP_1 (which is injected at E in this case).

On INITIATE **at interface** C_1**:** The wrapper calls TRANSFER$(\mathsf{IdP}, \mathsf{SP}_1)$ at interface W of $\mathbf{Rep}^{\emptyset,\{\mathsf{IdP},\mathsf{E}\}}_{\{\mathsf{SP}_1\},\emptyset,\emptyset}$ to mimic the transfer of the first message to SP_1 (which is the value that C_1 received from IdP).

On any other input at C_i **or** SP_i**,** $i > 1$**, or** E **(for** $\mathbf{NET}^n_{\mathsf{uni}}$**):** Any such input is directly given as input to the respective interface of the subsytem $\mathbf{NET}^{n-1}_{\mathsf{uni}}$ and perfectly mimics the transmissions and injections of any other message in the session between the client interfaces C_i and the service provider interfaces SP_i. Furthermore, the wrapper only produces outputs at SP_i if the service provider received an assertion.

First, system $\mathbf{W}$ obeys the condition on the sequence of inputs as required by the specification in Sect. 4.1 and we can therefore conclude that replacing the subsystem is sound (note that we assume that there are no further conditions beyond this). For the remaining argument, observe that the main difference of the real and the ideal worlds are that the mode cannot MODE $= \mathsf{adv}$ is ineffective in the ideal world: Resource $\mathbf{NET}^{n,\mathsf{IdP}}_{\mathsf{mut}}$ only has one mode of operation where the adversary cannot inject messages. Now, when the wrapper $\mathbf{W}$ is connected to an authenticated repository (with the special simulator sim attached), then no write-instruction can be issued at interface E (the adversary can at most copy what the issuer's input). This is the same behavior as simulator $\overline{\mathsf{sim}}$ implements. The remaining cases are straightforward to verify. □