On Composable Security for Digital Signatures

Christian Badertscher[1][0000−0002−1353−1922], 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 repository for authentically reading values written by a certain party from certain assumed repositories, 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 asymmetric key pair consisting of a *signing key* and a *verification key*. The signing

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

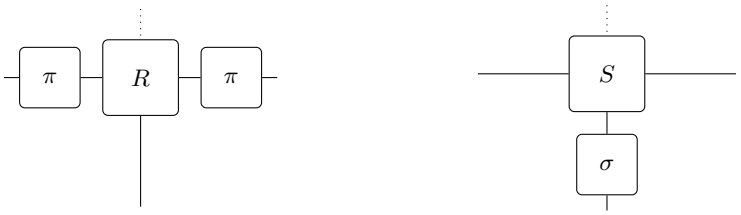

Fig. 1. Left: Execution of protocol π in the real-world model. **Right:** Ideal-world model described by S with simulator σ. In both figures, the dotted lines are "free" interfaces explained below.

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 Formalizing Message Authentication

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 [6,25,22], 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* [22,20] or *hybrid functionalities* [6] 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 indicates a potential attacker's access to R.

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* [22] or *ideal functionality* [6]. The access of the honest parties to S is via direct

interfaces, whereas a potential attacker accesses S via the so-called *simulator* σ. To define security, one considers random experiments in which a *distinguisher* (sometimes called *environment*) is connected to all open interfaces of either of the two settings in Fig. 1. The intuition behind the simulator is that, if the two settings are indistinguishable, then any attack on π (and working with assumed resource R) can be translated via σ into an attack on S, but S is secure by definition. Therefore, using the protocol π with the assumed resource R provides at least the same security guarantees as using the constructed resource 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 for this purpose requires an authenticated repository that can hold one message. This repository is used to transmit the signature verification key. We also assume 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 a request to sign 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. More technically, if a protocol π constructs S from R and protocol π' constructs T from S, then composing the two protocols leads to a construction of T from R.

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 Fig. 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 [10], 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 (or *converter* [22]) 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

[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 [25,17,22], no modification is necessary.

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 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 description 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 [23], which are sets of resources that, for example, fulfill a certain property. As such they are suitable to express an incomplete description of a resource, namely by considering the set of all resources that adhere to such a (partially defined) 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, in a weaker form, 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 chosen messages, 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* [14]. Early instantiations of this concept were given by Rivest, Shamir, and Adleman [26] and by Lamport [18]. The provable-security treatment of DSS was initiated by Goldwasser, Micali, and Rivest [15], 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 [7] and independently Pfitzmann and Waidner [25] 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 [6,8] 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 [6,11,1,8,9,12]. 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.* [5] 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.* [5] 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 [8, 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 [8, 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 [8, 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 [8] 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 [16], 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.* [13], 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 [13]. 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.

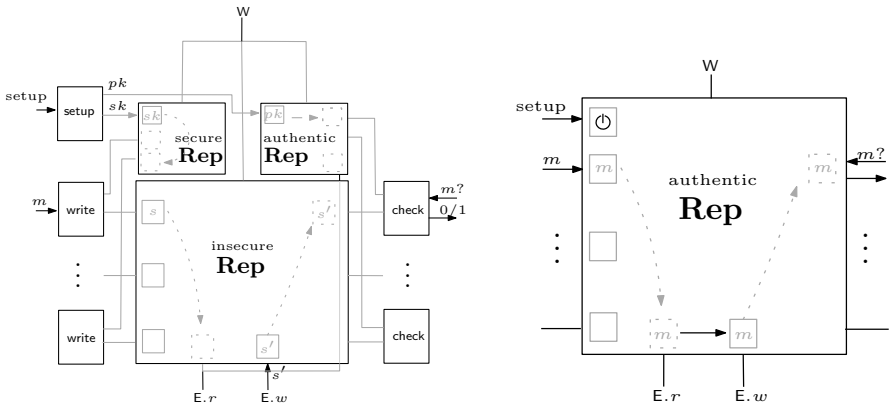

Fig. 2. Illustration of the main construction that characterizes a digital signature scheme. The assumed resources with the protocol (left) and the constructed resource (right). The adversarial interfaces are denoted by $\mathsf{E}.w$ (write) and $\mathsf{E}.r$ (read) and the free interface is denoted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

Finally, our work can be seen as orthogonal to the work of Canetti et al. [12], which extends the model of Canetti [6,8] 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.

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 converters 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 [15]:

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 [8], 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. We then proceed to discuss two concrete applications.

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 [19]. 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 $\mathsf{Pr}[\mathbf{DR} = 1]$ 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)* [21] or *bad event* [4]. This concept is used to define a similarity between two systems, the *game equivalence* [19] or *equivalence until bad* [4], 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* [19,4], 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 [22].

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 [22,23].

Specifications and relaxed specifications. As discussed in the introduction, we consider *specifications* [23] 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 $\mathbf{S}$ [23].

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 [15], 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,*

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

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

```
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 $\Sigma_{\text{rec}} = (K, S, R)$ *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 $\textsc{transfer}(\mathsf{A}, \mathsf{B})$, 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 $\mathbf{Rep}^{\mathcal{W}}_{\mathcal{R},\mathcal{V}}$.*

[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.

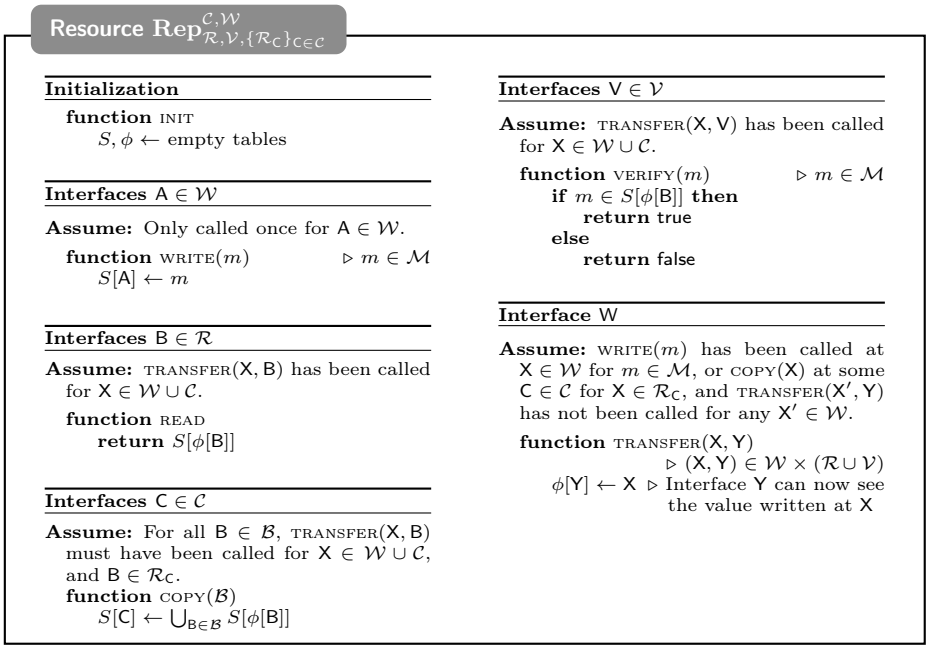

Fig. 4. Specification of a repository resource. For ease of notation, we treat values $m \in \mathcal{M}$ and singular sets $\{m\}$ for $m \in \mathcal{M}$ interchangeably.

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 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 $\mathbf{Rep}^{\mathcal{E}_c,\mathcal{W}}_{\mathcal{R}\cup\mathcal{E}_r,\emptyset,\{\mathcal{E}_r\}_{c\in\mathcal{E}_c}}$, 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 the full version of this work [3]. 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*

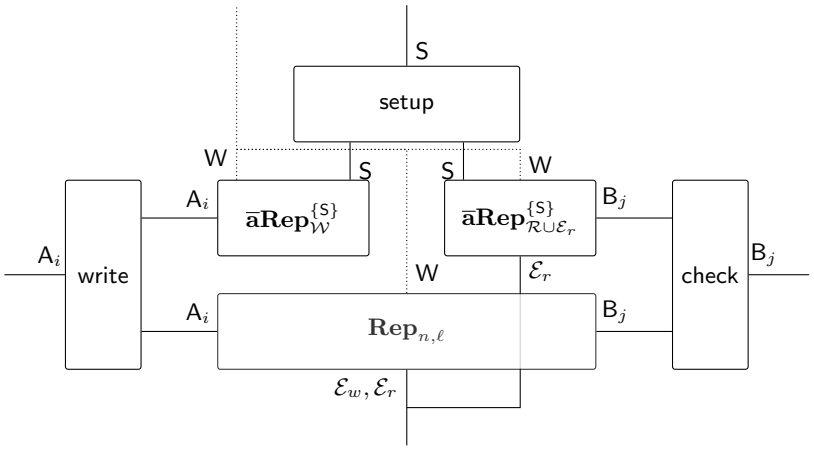

Fig. 5. The real-world setting of the signature construction.

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 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

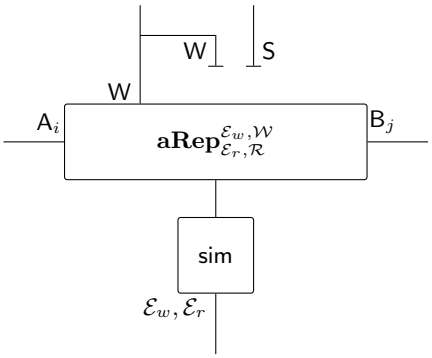

Fig. 6. The ideal-world setting of the signature construction. Inputs at the interfaces whose corresponding lines stop before the box (interfaces W and S in this example) have no effect on the behavior, therefore they are ignored in the specification.

confidential) transmission of a value, we assume another resource as specified by $\overline{\mathbf{a}}\mathbf{Rep}^{\mathcal{E}_c,\mathsf{S}}_{\mathcal{E}_r\cup\mathcal{R}}$ 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 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 $\text{TRANSFER}(\mathsf{S},\mathsf{A}_i)$ and $\text{TRANSFER}(\mathsf{S},\mathsf{B}_j)$, 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 whose proof appears in the full version [3].

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*

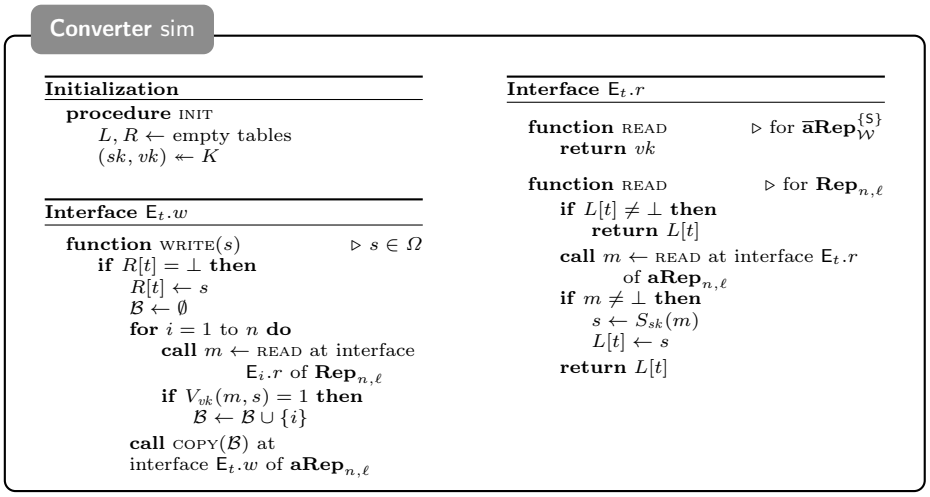

Fig. 7. Simulator for the proof of Theorem 1.

$\mathsf{sim}^{\mathsf{E}}\mathbf{aRep}_{n,\ell}$, *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*

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

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(S, 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(S, 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.*

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.

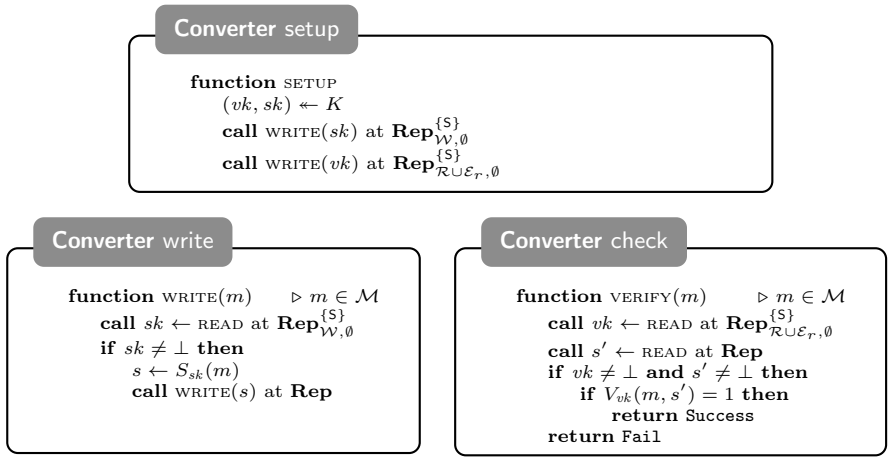

Fig. 8. The three protocol converters derived from a signature scheme $\Sigma = (K, S, V)$.

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, we state the following theorem proven in the full version [3].

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*,*

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

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.

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, we consider repositories that offer the capability to retrieve the value that has been

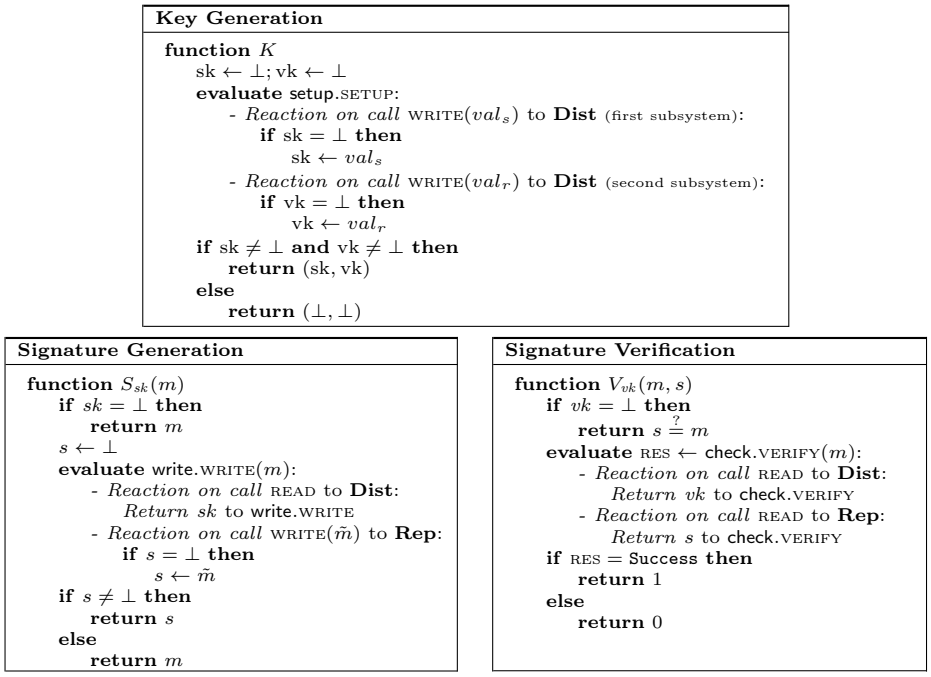

Fig. 9. Signature scheme (K, S, V) extracted from converters setup, write, and check.

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 $\bar{\mathbf{a}}\mathbf{Rep}$ 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* $\mathsf{DSS} := (\mathsf{setup}, \mathsf{write}, \ldots, \mathsf{write}, \mathsf{read}, \ldots, \mathsf{read})$ *(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 $\mathsf{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ 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,

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

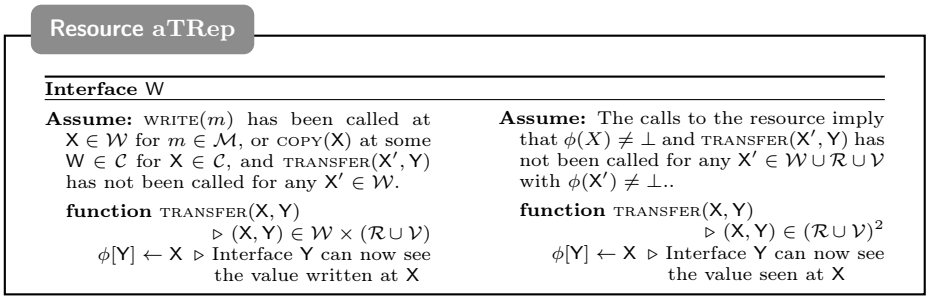

Fig. 10. Specification of a repository resource with transferable rights. Only the modifications with respect to Fig. 4 are shown; the other functions are as described there.

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 $\mathbf{aTRep}$ 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 $\textsc{send}$) and receive messages (by calling $\textsc{receive}$), 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 described above (and formally specified as pseudo-code in the full version) constructs resource* $\mathbf{Reg}$ *from the described set of authenticated channels. More specifically, for converters* issue *and* reg*, there is a simulator* $\tilde{\mathsf{sim}}$ *(formally specified as pseudo-code in the full version of this work), such 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 the full version [3]. □

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,

[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 be the number i.

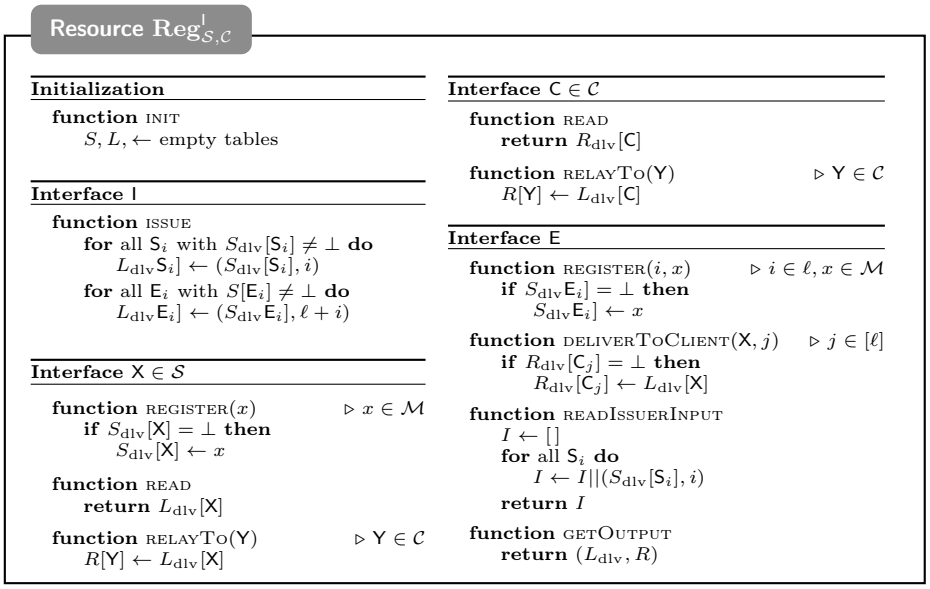

Fig. 11. The registration service resource.

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 [24]: 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 guarantee 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 $\mathbf{NET}^{n,\mathsf{IdP}}_{\mathsf{mut}}$ as described in Fig. 12.

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 $\mathbf{SEC}_{\mathsf{IdP},\mathsf{C}_1}$. 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 proto-

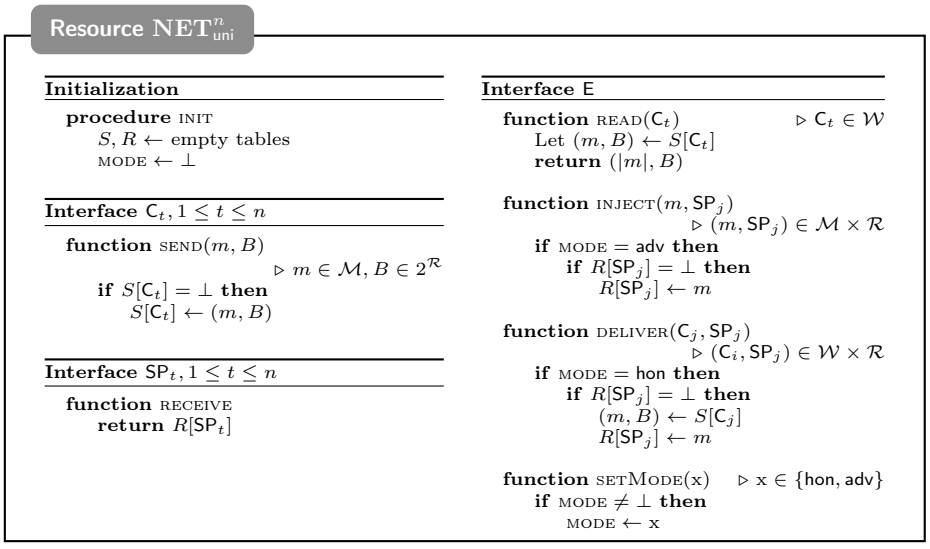

Fig. 12. The unilaterally secure network resource: the adversarial interface E can choose whether the network runs in *secure* (MODE $=$ hon) or *adversarial* mode (MODE $=$ adv).

col 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 described above (and formally specified as pseudo-code in the full version), 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, setting* $[\mathbf{Ch}_{\mathsf{IdP}\to\mathsf{SP}}, \mathbf{SEC}_{\mathsf{IdP},\mathsf{C}_1}, \mathbf{NET}^n_{\mathsf{uni}}]$*. More specifically, there is a simulator* $\tilde{\mathsf{sim}}$ *(formally specified as pseudo-code in the full version of this work) such that for any distinguisher* $\mathbf{D}$*, there is an attacker* $\mathbf{A}$ *against the underlying signature scheme (with essentially the same efficiency), i.e.,*

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

Proof. The proof is given in the full version [3] and follows a similar idea to the one of the previous section. □

The approach of sending assertions to upgrade a unilaterally authenticated channel to full authentication is used, for instance, in the widely used SAML protocol [16]. This section can be seen as a proof of an abstract version of SAML.

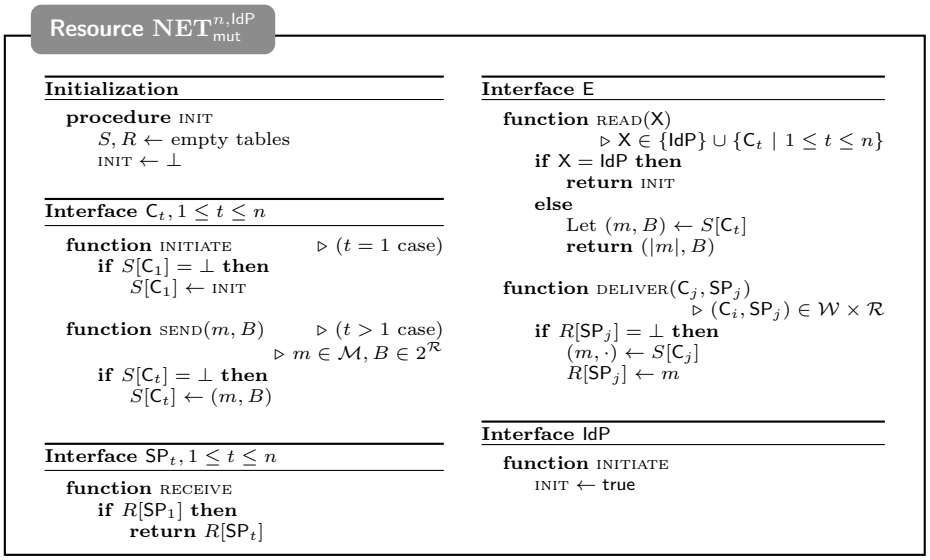

Fig. 13. In a mutually secure network between a client and a service provider, the adversary can only deliver messages between the client and the service provider.

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. Badertscher, C., Maurer, U., Tackmann, B.: On composable security for digital signatures. Cryptology ePrint Archive, Report 2018/015 (2018), `https://eprint.iacr.org/2018/015` (Full Version of this paper)
4. 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)
5. 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)
6. 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
7. 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)
8. Canetti, R.: Universally composable signature, certification and authentication. In: Proceedings of CSFW 2004 (2004)

9. 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
10. 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)
11. 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/
12. 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
13. 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)
14. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (November 1976)
15. 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)
16. 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)
17. 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)
18. Lamport, L.: Constructing digital signatures from a one-way function. Tech. Rep. CSL-98, SRI International, Menlo Park, California (October 1979)
19. 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)
20. Maurer, U.: Constructive cryptography - a new paradigm for security definitions and proofs. In: TOSCA 2011. pp. 33–56 (2011)
21. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Theory of Cryptography (2004)
22. Maurer, U., Renner, R.: Abstract cryptography. In: ICS. pp. 1–21 (2011)
23. Maurer, U., Renner, R.: From indifferentiability to constructive cryptography (and back). In: Hirt, M., Smith, A. (eds.) Theory of Cryptography. LNCS, vol. 9985, pp. 3–24. Springer (2016)
24. Maurer, U., Tackmann, B., Coretti, S.: Key exchange with unilateral authentication: Composable security definition and modular protocol design. Cryptology ePrint Archive, Report 2013/555 (September 2013)
25. 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)
26. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)