Relating Symbolic and Cryptographic Secrecy

Michael Backes, Birgit Pfitzmann
IBM Zurich Research Lab
{mbc,bpf}@zurich.ibm.com

Abstract

We investigate the relation between symbolic and cryptographic secrecy properties for cryptographic protocols. Symbolic secrecy of payload messages or exchanged keys is arguably the most important notion of secrecy shown with automated proof tools. It means that an adversary restricted to symbolic operations on terms can never get the entire considered object into its knowledge set. Cryptographic secrecy essentially means computational indistinguishability between the real object and a random one, given the view of a much more general adversary. In spite of recent advances in linking symbolic and computational models of cryptography, no relation for secrecy under active attacks is known yet.

For exchanged keys, we show that a certain strict symbolic secrecy definition over a specific Dolev-Yao-style cryptographic library implies cryptographic key secrecy for a real implementation of this cryptographic library. For payload messages, we present the first general cryptographic secrecy definition for a reactive scenario. The main challenge is to separate secrecy violations by the protocol under consideration from secrecy violations by the protocol users in a general way. For this definition we show a general secrecy preservation theorem under reactive simulatability, the cryptographic notion of secure implementation. This theorem is of independent cryptographic interest. We then show that symbolic secrecy implies cryptographic payload secrecy for the same cryptographic library as used in key secrecy. Our results thus enable existing formal proof techniques to establish cryptographically sound proofs of secrecy for payload messages and exchanged keys.

1 Introduction

Proofs of cryptographic protocols are known to be error-prone and, owing to the distributed-system aspects of multiple interleaved protocol runs, awkward to make for humans. Hence automation of such proofs has been studied almost since cryptographic protocols first emerged. From the start, the actual cryptographic operations in such proofs were idealized into so-called Dolev-Yao models, following [14] with extensions in [15, 22], e.g., see [24, 21, 17, 27, 28, 1, 19, 25]. These models replace cryptography by term algebras, e.g., encrypting a message m twice does not yield a different message from the basic message space but the term $\mathsf{E}(\mathsf{E}(m))$. A typical cancellation rule is $\mathsf{D}(\mathsf{E}(m)) = m$ for all m. It is assumed that even an adversary can only operate on terms by the given operators and by exploiting the given cancellation rules. This assumption, in other words the use of initial models of the given equational specifications, makes it highly nontrivial to know whether results obtained over a Dolev-Yao model are also valid over real cryptography. One therefore calls properties and actions in Dolev-Yao models *symbolic* in contrast to *cryptographic*.

Arguably the most important and most common properties proved symbolically are secrecy properties, as initiated in [14]. Symbolically, the secrecy of a payload or a cryptographic object like a secret key is represented by knowledge sets: The object is secret if the adversary can never get the corresponding symbolic term into its knowledge set. Cryptographically, secrecy is typically defined by computational indistinguishability between the real object and a randomly chosen one, given the view of the adversary.

Hence symbolic secrecy captures the absence of structural attacks that make the secret as a whole known to the adversary, and because of its simplicity it is accessible to formal proofs tools, while cryptographic secrecy constitutes a more fine-grained notion of secrecy that is much harder to establish.

There has been significant progress in relating symbolic verification and real cryptographic properties. Nevertheless, secrecy properties in this sense have not yet been considered, and their preservation is not a simple consequence of general simulatability definitions between ideal and real system, nor of specific results from implementing a Dolev-Yao-style cryptographic library in a cryptographically secure way.

1.1 Related Work

Early work on linking Dolev-Yao-style models and real cryptography [3, 2, 18] only considers passive attacks, and can therefore not make general statements about protocols. The same holds for [16].

The Backes-Pfitzmann-Waidner line of work contains a number of related results. Primarily, there is a specific Dolev-Yao-style cryptographic library with a provably secure real implementation [11], and its extensions from public-key to symmetric systems [12, 9]. The notion of "as secure as" proved there, also called reactive simulatability, is indeed a powerful notion that allows for general composition, i.e., the ability to prove protocols with the ideal library and subsequently to plug in the real library. It essentially states that the views of honest users are indistinguishable when they use either the ideal or the real library, and, after composition, when they use a protocol with either the ideal or the real library. This corresponds to the intuitive idea that a replacement of an ideal by a real system is good if anything that can happen to users in the real system could also happen to them in the ideal system.

However, this view of the users does not contain the adversary knowledge set as typically used in symbolic secrecy proofs, and indeed this is a purely symbolic notion that does not exist in an indistinguishable way in the real system. Nor does the user view contain the actual key bitstrings, which are cryptographically secret in the real system, because this is a purely cryptographic notion that does not exist in an indistinguishable way in the ideal system. Hence, although we will essentially prove below that symbolic secrecy implies cryptographic secrecy for this Dolev-Yao-style library and its implementation, this is clearly not a direct consequence of the known as-secure-as relation.

A second class of related results in this line of work are property preservation theorems. So far, they exist for integrity, non-interference, and a polynomial form of liveness [5, 6, 8, 10]. All these theorems are general for the notion of reactive simulatability and build on the indistinguishability of user views. Thus when specialized to the Dolev-Yao-style cryptographic library, they cannot yield the desired type of results as we just saw. In fact, only non-interference is a kind of secrecy property, and it is formulated as the flow of information from one user port to another, irrespective of adversary views.

A third class of related results are protocol proofs above the cryptographic library [7, 4]. The former, for the Needham-Schroeder-Lowe public-key protocol, is entirely an authentication proof. The latter, for the Otway-Rees protocol, contains a secrecy property, but this has been reformulated by hand into an integrity property so that the integrity preservation theorem could be used.

Finally, a much more narrow result (in terms of possible protocols and preserved properties) about an ideal and real cryptographic library, but with a slightly simpler real implementation, is given in [23]. The property preserved here is explicitly only integrity. The Canetti line of work, which also contains abstractions from cryptography, does not contain any Dolev-Yao-style cryptographic library at present, i.e., no system on which the typical kind of symbolic secrecy of terms could even be defined, nor any property preservation theorems.

Hence there is still no theorem that symbolic secrecy properties defined via adversary knowledge sets for a Dolev-Yao-style cryptographic library imply cryptographic secrecy of the corresponding real terms. We will provide such a theorem in this paper.

1.2 Overview of Our Results

The nicest possible theorem would be that for the real and ideal Dolev-Yao-style cryptographic library from [11, 12, 9], all terms that are symbolically secret are also cryptographically secret. However, such a strong statement does not hold (and we believe that this has nothing to do with the specifics of this cryptographic library). First, in many situations, symbolic secrecy does not exclude that partial information about a cryptographic object has become known. This is quite natural given that symbolic secrecy only states that the adversary does not have an *entire* term in its knowledge set. One example is that a public key contains partial information about a secret key, i.e., given the public key, anyone can distinguish the real secret key from a random one, for example by validating that signatures made with the secret key are valid with respect to the public key, and similarly for encryptions (which is even easier if the generation algorithm derives the public key from the secret key alone). The second example is that symmetric authentications and encryptions provide partial information about a symmetric secret key, at least if one also has partial information about the message encrypted or authenticated. Nevertheless, symbolic secrecy never classifies a secret key as known to the adversary just because a corresponding public key or corresponding symmetric encryptions and authentications are known to the adversary. A third and different example is that a payload, i.e., a message input to a protocol by a user, may become known or partially known to the adversary by direct interaction with users (e.g., a chosen-message attack) or by a user reusing this message or a statistically related message in another protocol run. Direct interactions of protocol users and the adversary are typically excluded in symbolic models, and so is the reuse of a secret message in other protocol runs. In a general cryptographic reactive setting, however, this is not excluded a priori. Hence our theorems have to be more specific.

The problems just described are quite different for payloads and for the secrecy of objects generated within the cryptographic library. Hence we prove different theorems for the secrecy of payloads and of cryptographic objects, which in this context means the secret keys typically exchanged in key-exchange protocols.

For payload secrecy, there is not even a general cryptographic secrecy definition yet; definitions are typically specific to the protocols considered and contain an algorithm called a message chooser that selects one particular payload independent of all others and not influenced by the adversary. This overcomes the described problems, but does not easily generalize to arbitrary protocols and to realistic situations with message reuse within a protocol run or across protocol runs, or where the adversary has a priori information about the payload. We introduce a different approach: We let honest users generate payloads as they like, but replace the payloads consistently at the interface to the system under consideration when they occur in certain secret payload positions. The resulting definitions are independent of the cryptographic library and give rise to a general payload secrecy preservation theorem. In addition, we show that symbolic secrecy in the Dolev-Yao sense implies the payload secrecy in this sense for the ideal cryptographic library and consequently for the real cryptographic library.

For the secrecy of secret keys, we essentially restrict ourselves to the typical situation directly after a key-exchange protocol for this key: We require on the symbolic side that no encryptions or authenticators with the exchanged key have yet been made, or at least not become known to the adversary. Then we can indeed show that the cryptographic key is completely indistinguishable from a random key, given the view of the adversary. This is the typical key secrecy definition of cryptography. Although our additional symbolic precondition may exclude some key-exchange protocols that are typically considered secure by symbolic methods, these protocols are in fact imperfect from a cryptographic point of view: A key-exchange protocol in cryptography should be sequentially composable with an arbitrary protocol using this key, e.g., a secure channel. The arbitrary protocol will be proved secure under the assumption that it uses a fresh random key. Hence the key exchange protocol must guarantee that the resulting key can be used wherever a fresh random key can be used. The only way to guarantee this is by indistinguishability from a fresh random key. Indeed, a key that has already been used as an authenticator might potentially end up in a protocol where precisely this authentication can be used for a cross-protocol attack, thus destroying the security of the protocol. Compared with message secrecy, this key-secrecy theorem is relatively easy to state—we simply need the condition on keys to be not only symbolically unknown to the adversary, but also symbolically unused. However, the proof is complex because we have to augment the entire proof of the given cryptographic library with corresponding statements about symbolic key handles and real keys, in addition to the current statements aimed at proving only indistinguishability of the user views.

2 Overview of the Underlying Dolev-Yao-Style Cryptographic Library

In this section, we give an overview of the Dolev-Yao-style model from [11, 12, 9], for which we will prove relations between symbolic and cryptographic payload and key secrecy.

2.1 Terms, Handles, and Operations

As described in the introduction, a Dolev-Yao-style model abstracts from cryptographic objects by terms of a term algebra. A specific aspect of the Dolev-Yao-style model in [11] is that participants operate on terms by local names, not by handling the terms directly. This is necessary to give the abstract Dolev-Yao-style model and its realization the same interface, so that either one or the other can be plugged into a protocol. An identical interface is also an important precondition for the security notion of reactive simulatability. One can see protocol descriptions over this interface as low-level symbolic representations as they exist in several other frameworks, and it should be possible to compile higher-level descriptions into them following the ideas first developed in [20]. The local names are called *handles*, and chosen as successive natural numbers for simplicity.

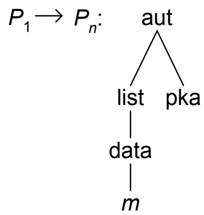
Like all Dolev-Yao-style models when actually used for protocol modeling, e.g., using a special-purpose calculus or embedded in CSP or pi-calculus, the model in [11] has state. An important use of state is to model which participants already know which terms. Here this is given by the handles, i.e., the adversary's knowledge set is the set of terms to which the adversary has a handle.

Another use of state is to remember different versions of terms of the same structure for probabilistic operations such as nonce or key generation. In [11], as probably first in [21], the probabilism is abstracted from by counting, i.e., by assigning successive natural numbers to terms, here globally over all types. This *index* of a term allows us (not the participants) to refer to terms unambiguously.

The users can operate on terms in the expected ways, e.g., give commands to en- or decrypt a message, to generate a key, or to in- or output a payload message. Further, they can input that a term should be sent to another user; in the symbolic representation this only changes the knowledge sets, i.e., in this specific Dolev-Yao-style library it means that the intended recipient and/or the adversary (depending on the security of the chosen channel) obtains a handle to the term.

2.2 Notation

The symbol "$:=$" denotes deterministic and "$\leftarrow$" probabilistic assignment, and "$\stackrel{\mathcal{R}}{\leftarrow}$" denotes the uniform random choice from a set. Messages are strings over an alphabet Σ. The length of a message m is denoted as $\mathsf{len}(m)$, and $\downarrow$ is an error element available as an addition to the domains and ranges of all functions and algorithms. The list operation is denoted as $l := (x_1, \ldots, x_j)$, and the arguments are unambiguously retrievable as $l[i]$, with $l[i] = \downarrow$ if $i > j$. A database D is a set of functions, called entries, each over a finite domain called attributes. For an entry $x \in D$, the value at an attribute att is written $x.att$. For a predicate $pred$ involving attributes, $D[pred]$ means the subset of entries whose



ind	type	arg	hnd_1	...	hnd_n	len
1	pka	()	$\downarrow$	...	$\downarrow$	0
2	ska	(1)	1	...	1	128
3	data	m	2	...	$\downarrow$	1000
4	list	(3)	3	...	$\downarrow$	1032
5	aut	(4,1)	4	...	2	1200

Figure 1: Example of the database representation of terms

attributes fulfill $pred$. If $D[pred]$ contains only one element, the same notation is used for this element. Finally, $NEGL$ denotes the set of all negligible functions, i.e., $g\colon \mathbb{N} \to \mathbb{R}_{\geq 0} \in NEGL$ iff for all positive polynomials Q, $\exists k_0 \forall k \geq k_0\colon\ g(k) \leq 1/Q(k)$

2.3 Details about the State Representation

The overall representation of a state of the Dolev-Yao-style model of [11] is a database D of the existing terms with their type (top-level operator), argument list, handles, index, and lengths as database attributes. The length is needed because encryption cannot completely hide the length of messages. The non-atomic arguments of a term are given by the indices of the respective subterm.

An example is shown in Figure 1. The left side indicates the main action that has happened so far, the sending of an authenticated list with one element, a payload m. The database first contains the symmetric authentication key of type ska together with a public key tag of type pka. (These tags are needed to deal with situations where the adversary can distinguish whether several symmetric authenticators or encryptions have been made with the same key.) In the example, both participants know the secret key, i.e., have a handle to it, while honest participants never have handles to the public key tags. Then the database contains the payload data, the list, and the authenticated message. The example assumes that this message has arrived safely so that P_n has a handle to it, but has not yet been parsed by the recipient. After parsing, the list and m get handles 3 and 4 for P_n, respectively. Note that the handles are indeed local names, i.e., different for the two participants.

In detail, the database attributes of D are defined as follows, where $\mathcal{H}$ denotes the set of user indices.

- $ind \in \mathcal{INDS}$, called index, consecutively numbers all entries in D. The set $\mathcal{INDS}$ is isomorphic to $\mathbb{N}$; it is used to distinguish index arguments from others. The index serves as a primary key attribute of the database, i.e., one can write $D[i]$ for the selection $D[ind = i]$.

- $type \in typeset$ defines the type of the entry. In particular, the type data denotes payloads, skse and ska denote secret encryption and authentication keys, pkse and pka corresponding public tags, and symenc and aut denote symmetric encryptions and authenticators. Other types will be introduced when first used.

- $arg = (a_1, a_2, \ldots, a_j)$ is a possibly empty list of arguments. Many values a_i are indices of other entries in D and thus in $\mathcal{INDS}$. They are sometimes distinguished by a superscript "ind".

- $hnd_u \in \mathcal{HNDS} \cup \{\downarrow\}$ for $u \in \mathcal{H} \cup \{\mathsf{a}\}$ are handles by which a user or adversary u knows this entry. The value $\downarrow$ means that u does not know this entry. The set $\mathcal{HNDS}$ is yet another set isomorphic to $\mathbb{N}$. Handles are always get a superscript "hnd".

- $len \in \mathbb{N}_0$ denotes the "length" of the entry.

2.4 The Real Cryptographic Library

In the real implementation of the cryptographic library in [11, 12, 9], the central database of all terms with handles (local names) for each user is replaced by a different machine for each user u. This machine contains a database D_u with only three main attributes: the handle hnd_u for this user u, the real cryptographic bitstring $word$, and the type $type$. The users can use exactly the same commands as to the ideal library, e.g., en- or decrypt a message etc. These commands now trigger real cryptographic operations. The operations essentially use standard cryptographically secure primitives, but with certain additional tagging, randomization etc. Send commands now trigger the actual sending of bitstrings between machines and/or to the adversary.

2.5 Overall Framework and Adversary Model

So far we described the ideal and real cryptographic library informally. We now give an overview of the underlying system model and introduce some more notation for later use. The underlying machine model is an IO-automata model. Hence the overall ideal Dolev-Yao-style library, with its database D, is represented as a machine. It is called trusted host. Actually there is one possible trusted host $\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}$ for every subset $\mathcal{H}$ of a set $\{1, \ldots, n\}$ of users, denoting the possible honest users. It has ports $\mathsf{in}_u?$ for inputs from and $\mathsf{out}_u!$ for outputs to each user $u \in \mathcal{H}$ and for $u = \mathsf{a}$, denoting the adversary. The use of ports for attaching different channels to a machine and their naming follows the CSP convention, e.g., the cryptographic library obtains messages at $\mathsf{in}_u?$ that have been output by a user machine at $\mathsf{in}_u!$.

Using the notation of [11], the ideal cryptographic library is a *system* $Sys_{n,L}^{\mathsf{cry,id}}$ that consists of several *structures* $(\{\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}\}, S_{\mathcal{H}}^{\mathsf{cry}})$, one for each value of $\mathcal{H}$. Each structure consists of a set of machines, here only containing the machine $\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}$, and a set $S_{\mathcal{H}}^{\mathsf{cry}} := \{\mathsf{in}_u?, \mathsf{out}_u! \mid u \in \mathcal{H}\}$ denoting those ports of $\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}$ that the honest users connect to. The set $S_{\mathcal{H}}^{\mathsf{cry}}$ is called *service ports* or informally the user interface. Formally, the system is $Sys_{n,L}^{\mathsf{cry,id}} := \{(\{\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}\}, S_{\mathcal{H}}^{\mathsf{cry}}) \mid \mathcal{H} \subseteq \{1, \ldots, n\}\}$, where L denotes a tuple of length functions needed to compute the "length" of the abstract terms in the database. The parameters n and L will not matter any further and are hence omitted in the following.

In the real implementation of the cryptographic library, the same interface is served by a set $\hat{M}_{\mathcal{H}}^{\mathsf{cry}} := \{\mathsf{M}_u^{\mathsf{cry}} \mid u \in \mathcal{H}\}$ of real cryptographic machines. The corresponding system is called $Sys_{\mathcal{E},\mathcal{S},\mathcal{A},\mathcal{SE}}^{\mathsf{cry,real}} := \{(\hat{M}_{\mathcal{H}}^{\mathsf{cry}}, S_{\mathcal{H}}^{\mathsf{cry}}) \mid \mathcal{H} \subseteq \{1, \ldots, n\}\}$, where $\mathcal{E}$, $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{SE}$ denote the cryptographic schemes used for asymmetric encryption, signatures, symmetric authentication, and symmetric encryption, respectively.

2.6 Configurations, Runs, and Views

When considering the security of a structure $(\hat{M}, S)$, an arbitrary probabilistic machine H is connected to the user interface to represent all users, and an arbitrary machine A is connected to the remaining free ports (typically the network) and to H to represent the adversary. In polynomial-time security proofs, H and A are polynomial-time. The resulting tuple $(\hat{M}, S, \mathsf{H}, \mathsf{A})$ is called a *configuration*, and the set of all configurations of a system Sys is called $\mathsf{Conf}(Sys)$. A configuration is runnable, i.e., for each value k of a security parameter one gets a well-defined probability space of *runs*. The *view* of a machine in a run is the restriction to all in- and outputs this machine sees and its internal states. Formally, the possible runs run_{conf} in a configuration $conf$ and the view $view_{conf}(\mathsf{M})$ of a machine M in $conf$ are a *family of random variables* with one element for each security parameter value k. The notation $r \in run_{conf}$ abbreviates that r is a possible run of $conf$, i.e., it belongs to the carrier set of an arbitrary random variable in run_{conf}.

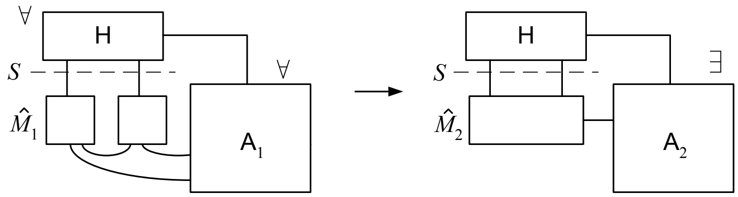


Figure 2: Simulatability example: The two views of H must be indistinguishable

2.7 Reactive Simulatability

The security proof of [11] states that the real library is *at least as secure as* the ideal library. This is captured using the notion of *reactive simulatability*, which is the cryptographic notion of secure implementation. For reactive systems, it means that whatever might happen to an honest user in a (typically real) system Sys_1 can also happen in a (typically more ideal) system Sys_2 given as a specification: For every user H and every real structure and real adversary this user may interact with, there exists a corresponding ideal structure and ideal adversary such that the view of H is computationally indistinguishable in the two configurations. This is illustrated in Figure 2. Indistinguishability is a well-known cryptographic notion from [29].

Definition 2.1 *(Computational Indistinguishability)* Two families $(\mathsf{var}_k)_{k\in\mathbb{N}}$ and $(\mathsf{var}'_k)_{k\in\mathbb{N}}$ of random variables on common domains D_k are *computationally indistinguishable* ("$\approx$") iff for every algorithm D (the distinguisher) that is probabilistic polynomial-time in its first input,

$$|P(\mathsf{D}(1^k, \mathsf{var}_k) = 1) - P(\mathsf{D}(1^k, \mathsf{var}'_k) = 1)| \in NEGL,$$

(as a function of k). ◇

Intuitively, given the security parameter and an element chosen according to either var_k or var'_k, D tries to guess which distribution the element came from.

Definition 2.2 *(Reactive Simulatability)* For two systems Sys_1 and Sys_2, one says $Sys_1 \geq_{\mathsf{sec}} Sys_1$ (*at least as secure as*) iff for every polynomial-time configuration $conf_1 = (\hat{M}_1, S, \mathsf{H}, \mathsf{A}_1) \in \mathsf{Conf}(Sys_1)$, there exists a polynomial-time configuration $conf_2 = (\hat{M}_2, S, \mathsf{H}, \mathsf{A}_2) \in \mathsf{Conf}(Sys_2)$ (with the same H) such that $view_{conf_1}(\mathsf{H}) \approx view_{conf_2}(\mathsf{H})$. The relation $\geq_{\mathsf{sec}}$ is also called *simulatability*. *Universal* simulatability, written $\geq_{\mathsf{sec}}^{\mathsf{univ}}$, means that A_2 does not depend on H (only on $\hat{M}_1$, S, and A_1), and *black-box* simulatability that A_2 consists of a simulator Sim that depends only on $(\hat{M}_1, S)$ and uses A_1 as a blackbox submachine. ◇

Clearly, black-box simulatability implies universal simulatability; the cryptographic library has been proven with blackbox simulatability. An essential feature of this definition of simulatability is a composition theorem [26, 13], which roughly says that one can design and prove a larger system based on the ideal system Sys_{id}, and then securely replace Sys_{id} by the real system Sys_{real}.

3 Secrecy of Payload Messages

Since we work in a reactive environment and since we quantify over all users, we cannot simply define the secrecy of payloads by demanding that the adversary does not learn them at all since the users themselves might send him the payloads. Thus we have to capture that the adversary does not learn any information about the payloads *from the system*. E.g., even a secure channel would clearly not offer secrecy in the strict sense that the adversary does not learn the transmitted payloads at all, since the honest sender or recipient might send the same payloads to the adversary. We therefore have to separate information that leaks by user behavior from information that leaks in the system. We first present a general cryptographic definition that captures this separation. We then prove that this type of payload secrecy is preserved by "as secure as". Finally, we define a symbolic payload secrecy notion for protocols over the ideal Dolev-Yao-style cryptographic library that also comprises this separation, and we prove that this symbolic payload secrecy implies cryptographic payload secrecy for the protocol using the real cryptographic library.

3.1 General Cryptographic Message Secrecy

To capture the separation between information leakage by a protocol and information leakage by the users in a reactive framework, we define a replacement machine R that replaces message parts that are supposed to be secret by random ones at the system interface. If the system leaks no information about these message parts, then this replacement will not be distinguishable, no matter what information the honest users leak about the real messages. The replacement must be done consistently for different in- and outputs that should represent the same message; hence we have selection functions for these message parts both in inputs and in outputs. For instance, for a two-party secure channel with inputs (send, m) and outputs $(\mathsf{receive}, m)$, the selection functions for inputs and outputs would both select m, i.e., the second list element. On input a command containing a selected payload m, the replacement machine replaces m by a random payload n of the same length, stores the tuple (m, n) in a set T called a replacement table, and outputs the command with the replaced parameters. To ensure indistinguishable behavior to the users, the replacement machine further uses table-lookup in T to transform messages received from the network back into their original form.

We start the formal definitions by defining suitable selection functions.

Definition 3.1 *(Payload Selection Function)* A *payload selection function* is a function that assign every string l a potentially empty set of non-overlapping substrings of l. ◇

We now formally introduce the replacement machine. The selection functions of secret input and output parts are called f and g. In order to wrap a structure with service ports S by a replacement machine, we give the replacement machine these ports so that the overall user interface remains unchanged, see Figure 3, and we use a consistently renamed version of the port set to link the replacement machine and the original machines. The complement of a port set, i.e., the ports the connecting machines need, is denoted by S^C.

Definition 3.2 *(Replacement Machine)* Let a port set S and payload selection functions f, g be given. Let $L\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ be arbitrary. The *replacement machine* $\mathsf{R}_{S,f,g,L}$ for S, f, g, and L is defined as follows: It has the port set S and a renamed version S' of S^C. It has an initially empty set T called replacement table and the following transition rules:

- On input a message l at a port in S, let $\{m_1, \ldots, m_n\} := f(l)$. Replace every payload m_i for which there exists exactly one n_i with $(m_i, n_i) \in T$ by n_i in l. For the remaining payloads m_i set $n_i \xleftarrow{\mathcal{R}} \{0,1\}^{\mathsf{len}(m_i)} \setminus \{n \mid \exists m : (m,n) \in T\}$, $T := T \cup \{(m_i, n_i)\}$, and replace m_i by n_i in l. Output the resulting string l' to the underlying system at the corresponding port.
- On input a message l at a port in S', let $\{n_1, \ldots, n_j\} := g(l)$. Replace every payload n_i for which there exists exactly one m_i with $(m_i, n_i) \in T$ by m_i in l. Output the resulting string l' to the honest user at the corresponding port.

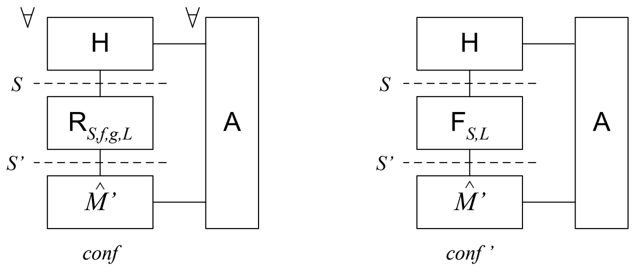


Figure 3: Sketch of the definition of reactive message Secrecy. The view of H should be indistinguishable in both configurations.

We further define that $\mathsf{R}_{S,f,g,L}$ accepts $L(k)$ inputs at each port in $S \cup S'$ with k being the security parameter and that it reads the first $L(k)$ bits of each input. $\diamond$

It is easily provable that $\mathsf{R}_{S,f,g,L}$ is polynomial-time if L is polynomially bounded since only a polynomial number of inputs of polynomial length are processed, hence only a polynomial number of entries is created in T and the selection of payloads n_i is therefore easy to achieve in polynomial time. Moreover, it is clear by definition that for every n there exists at most one m such that $(m, n) \in T$, and vice versa.

Reactive payload secrecy for an arbitrary system is now captured by requiring that no user can distinguish whether it is interacting with an arbitrary adversary, the system and a replacement machine, or with the same adversary, the system and a machine F that simply forwards messages between the user and the system without modifying them. This is illustrated in Figure 3. We first formally introduce the forwarding machine and then give the definition of payload secrecy formally.

Definition 3.3 *(Forwarding Machine)* Let a port set S and a function $L\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ be given. The *forwarding machine* $\mathsf{F}_{S,L}$ for S and L is defined as follows: It has the port set S and a renamed version S' of S^C. On input a message l at a port in S or S', it forwards l to the corresponding port in S' or S, respectively. $\mathsf{F}_{S,L}$ accepts $L(k)$ inputs at each port in $S \cup S'$ with k being the security parameter and reads the first $L(k)$ bits of each input. $\diamond$

Definition 3.4 *(Reactive Payload Secrecy)* Let a system Sys, a structure $(\hat{M}, S) \in Sys$, and payload selection functions f and g be given. Let $(\hat{M}', S')$ be the structure where the port names of ports in S are consistently replaced on the machines $\hat{M}$ as for the port set S' in $\mathsf{R}_{S,f,g,L}$, see Figure 3. Then we say that *the payload messages selected by f and g are*

- *perfectly secret in* $(\hat{M}, S)$, written$(\hat{M}, S) = [f, g](\hat{M}, S)$, iff for all functions $L\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ and for all configurations $conf = (\hat{M}' \cup \{\mathsf{R}_{S,f,g,L}\}, S, \mathsf{H}, \mathsf{A})$ and $conf' = (\hat{M}' \cup \{\mathsf{F}_{S,L}\}, S, \mathsf{H}, \mathsf{A})$ (i.e., with the same user H and adversary A), we have $view_{conf}(\mathsf{H}) = view_{conf'}(\mathsf{H})$.

- *computationally secret in* $(\hat{M}, S)$, written $(\hat{M}, S) \approx [f, g](\hat{M}, S)$, iff the above holds for all polynomially bounded functions L, polynomial-time users H, polynomial-time adversaries A, and with equality of views replaced by indistinguishability of views.

- *perfectly respectively computationally secret in Sys*, written $Sys = [f, g]Sys$ respectively $Sys \approx [f, g]Sys$, iff $(\hat{M}, S) = [f, g](\hat{M}, S)$ respectively $(\hat{M}, S) \approx [f, g](\hat{M}, S)$ holds for all $(\hat{M}, S) \in Sys$.

$\diamond$

Clearly, perfect secrecy of payloads implies computational secrecy.

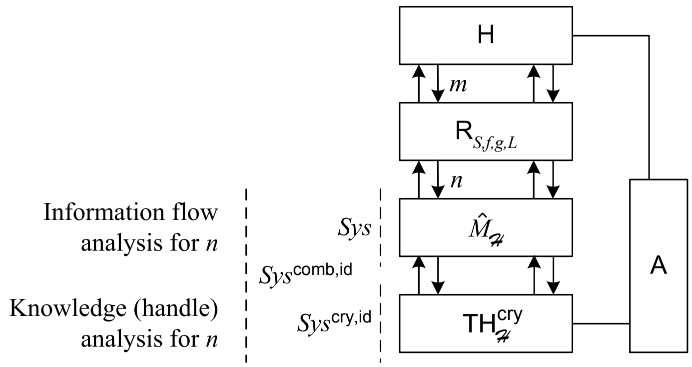


Figure 4: Symbolic payload secrecy in a protocol Sys

3.2 Payload Secrecy Preservation under Simulatability

We now show that if a system Sys_1 is as secure as a system Sys_2 in the sense of universal simulatability, then secrecy of payloads selected by payload selection functions f and g in Sys_2 implies the secrecy of the same payloads in Sys_1. This is a basis for proving payload secrecy for ideal systems and deriving it automatically for corresponding real systems.

Theorem 3.1 *(General Preservation Theorem for Payload Secrecy)* Let systems Sys_1, Sys_2 and payload selection functions f and g be given, and let $Sys_1 \geq_{\mathsf{sec}}^{\mathsf{univ}} Sys_2$. Then $Sys_2 \approx [f,g]Sys_2$ implies $Sys_1 \approx [f,g]Sys_1$. $\square$

The proof is postponed to the appendix. The preservation theorem constitutes a powerful tool for rigorously showing the secrecy of specific payloads in arbitrary reactive systems based on simple, usually even deterministic abstractions. Specifically for protocols over the ideal Dolev-Yao-style cryptographic library we can go even further and link the cryptographic secrecy notion to the original idea of symbolic secrecy.

3.3 Symbolic Message Secrecy and its Cryptographic Implications

For Dolev-Yao models, the original notion of the symbolic secrecy of a payload message is that the adversary does not get this payload into its knowledge set, i.e., in the current setting, that it does not get a handle to this payload. This is captured by the following definition, which considers a protocol that runs on top of the cryptographic library, corresponding to the usual scenario for symbolic secrecy analysis. The protocol is represented by a system Sys; typically such a system allows many interleaved executions of one or more protocols in the narrow sense. Even for symbolic secrecy we need the replacement machines in the general reactive setting with arbitrary protocol users H because we have to factor out the case that H hands the same payload directly to the adversary A, or sends it via other protocol executions. The situation is illustrated in Figure 4.

Definition 3.5 *(Symbolic Payload Secrecy in Protocols)* Let a polynomial-time system $Sys = \{(\hat{M}_{\mathcal{H}}, S_{\mathcal{H}} \cup S_{\mathcal{H}}^{\mathsf{cry}\,C}) \mid \mathcal{H} \subseteq \{1,\ldots,n\}\}$ be given, i.e., a system that can use the cryptographic library $Sys^{\mathsf{cry,id}}$, and where the free ports of $\hat{M}_{\mathcal{H}}$, i.e., the ports that are connected to other machines, are $S_{\mathcal{H}} \cup S_{\mathcal{H}}^{\mathsf{cry}\,C}$ for all $\mathcal{H}$. We assume further that the states of the machines in Sys are given by individual variables and their state transitions by programs over these variables, so that we can speak of a static information-flow analysis. Moreover, let payload selection functions f and g be given.

Let $Sys^{\mathsf{comb,id}} := \{(\hat{M}_{\mathcal{H}} \cup \{\mathsf{TH}^{\mathsf{cry}}_{\mathcal{H}}\}, S_{\mathcal{H}}) \mid \mathcal{H} \subseteq \{1, \ldots, n\}\}$ denote the composition of Sys and $Sys^{\mathsf{cry,id}}$, and for every $\mathcal{H}$ let $(\hat{M}'_{\mathcal{H}} \cup \{\mathsf{TH}^{\mathsf{cry}}_{\mathcal{H}}\}, S'_{\mathcal{H}})$ denote the structure where the port names of ports in $S_{\mathcal{H}}$ are consistently replaced on the machines in $\hat{M}_{\mathcal{H}}$ as for the port set $S'_{\mathcal{H}}$ in $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$.

Assume that for all functions $L\colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ and all configurations $(\hat{M}'_{\mathcal{H}} \cup \{\mathsf{TH}^{\mathsf{cry}}_{\mathcal{H}}, \mathsf{R}_{S_{\mathcal{H}},f,g,L}\}, S_{\mathcal{H}}, \mathsf{H}, \mathsf{A})$ the following holds throughout all runs, where D denotes the cryptographic term database and T the replacement table in $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$:

- Within $\hat{M}_{\mathcal{H}}$, static information flow from any input n with $(m, n) \in T$ for some m only takes place by propagation of n itself.
- If $\hat{M}_{\mathcal{H}}$ passes such a value n (i.e., one that arose from information flow as in the previous item) to $\mathsf{TH}^{\mathsf{cry}}_{\mathcal{H}}$, then only as the argument of a command store.
- If $\hat{M}_{\mathcal{H}}$ passes such a value n to $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$, then only as a message part selected by g, and vice versa, i.e., g only selects such values n for replacement.
- A term $D[i]$ resulting from such a command $\mathsf{store}(n)$ never gets an adversary handle, i.e., $D[i].hnd_{\mathsf{a}} = \downarrow$.

Then we say that *the payloads selected by f and g are symbolically secret in $Sys^{\mathsf{comb,id}}$*. ◇

The condition that $\hat{M}_{\mathcal{H}}$ has no free ports except those connected to its user or the cryptographic library means that the protocol does not communicate with the adversary except via the send commands within the cryptographic library, i.e., by the Dolev-Yao-style model.

Theorem 3.2 *(Symbolic and Computational Payload Secrecy in Protocols)* Let systems Sys and $Sys^{\mathsf{comb,id}}$ and payload selection functions f and g be given as in Definition 3.5. If the payloads selected by f and g are symbolically secret in $Sys^{\mathsf{comb,id}}$, they are computationally secret in $Sys^{\mathsf{comb,id}}$. □

The proof is postponed to the appendix. The complexity of the symbolic information-flow analysis underlying symbolic payload secrecy depends on the protocol language. Some simple high-level protocol expressions do not allow any information flow on payload messages except by direct assignments $x := y$, in particular the classical arrow notation without branching. Then the first condition is fulfilled for all protocols expressed in this language, and typically so is the second condition because of typing. Other languages may allow branches and thus indirect information flow, but still no direct operators on payload messages. Combining such an information-flow analysis with an analysis of the knowledge sets of a Dolev-Yao model (here represented by the possible adversary handles) that can arise by executing the protocol, is a standard problem addressed by symbolic proof tools for cryptographic protocols. The analysis might be made even more symbolic by replacing $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$ by a symbolic machine that chooses new names instead of random values, but that makes further conditions on the language used at the protocol interface to the user which we did not want to impose here for consistency with the cryptographic definitions.

Combining the results of Theorem 3.2, Theorem 3.1, and the fact that the real cryptographic library is as secure as the ideal one [11, 12, 9] yields the following corollary, which links symbolic secrecy to the cryptographic secrecy of the same protocol with a real cryptographic implementation.

Corollary 3.1 *With the notation of Definition 3.5, let the payload messages selected by f and g be symbolically secret in $Sys^{\mathsf{comb,id}}$. Then the payloads selected by f and g are computationally secret in the system $Sys^{\mathsf{comb,real}} := \{(\hat{M}_{\mathcal{H}} \cup \hat{M}^{\mathsf{cry}}_{\mathcal{H}}, S_{\mathcal{H}}) \mid \mathcal{H} \subseteq \{1, \ldots, n\}\}$ where $\hat{M}^{\mathsf{cry}}_{\mathcal{H}}$ denotes the set of machines of the real cryptographic library for a set $\mathcal{H}$.* □

4 Key Secrecy

In this section, we investigate the relationship of the secrecy of symmetric keys in the symbolic and the cryptographic approach. We define symbolic key secrecy for the ideal Dolev-Yao-style cryptographic library and cryptographic key secrecy for the real library, and we show that symbolic key secrecy implies cryptographic secrecy of the corresponding keys.

The symbolic secrecy definition is based on the typical notion that a term is not an element of the adversary's knowledge set. Recall that in the given Dolev-Yao-style library, the adversary's knowledge set is the set of all database entries (representing terms) to which the adversary has a handle. However, as explained in the introduction, we cannot hope to show the strong notion of cryptographic key secrecy, i.e., that the real cryptographic adversary cannot distinguish a real key from a fresh random key, for all keys without an adversary handle, but only for keys that are also unused, i.e., no corresponding encryption or authenticator has an adversary handle.

Furthermore, we have to be careful with the notion of correspondence between ideal and real keys for the secrecy preservation theorem. Originally, runs of either the ideal system or the real system are defined separately, and a per-key correspondence exists only in the simulatability proof. We start by using this correspondence. Then we define a more abstract correspondence notion without reference to the proof by characterizing the keys to be secret as a function of the user view, which exists in each system and should be indistinguishable between them.

4.1 Symbolic and Cryptographic Key Secrecy

As a first step towards defining symbolic key secrecy, we consider one state of the ideal Dolev-Yao-style library and define that a handle points to a symmetric key, that the key is symbolically unknown to the adversary, and that it has not been used for encryption or authentication. These are the symbolic conditions under which we can hope to prove that the corresponding real key is indistinguishable from a fresh random key for the adversary. Note that such a key may have been treated in the ways usual in key exchange protocols, e.g., an honest user may have put it into a list, encrypted the list, and sent it to another honest user.

For the third condition in the following definition, note that the arguments of a symmetric authenticator and a symmetric encryption with a key of an honest user are of the form (l, pk) where l is the plaintext index and pk the index of the public tag of the secret key, with $pk = sk - 1$ for the secret key index.

Definition 4.1 *(Symbolically Secret Keys)* Let $\mathcal{H} \subseteq \{1, \ldots, n\}$, a database state D of $\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}$, and a pair $(u, l^{\mathsf{hnd}}) \in \mathcal{H} \times \mathcal{HNDS}$ of a user and a handle be given. Let $i := D[hnd_u = l^{\mathsf{hnd}}].ind$ be the corresponding database index. We say that the *term under* (u, l^{hnd})

- *is a symmetric key* iff $D[i].type \in \{\mathsf{ska}, \mathsf{skse}\}$.
- *is symbolically unknown to the adversary*, or short *symbolically unknown*, iff $D[i].hnd_{\mathsf{a}} = \downarrow$.
- *has not been used for encryption/authentication*, or short *is unused*, iff for all indices $j \in \mathbb{N}$ we have
$$D[j].type \in \{\mathsf{aut}, \mathsf{symenc}\} \Rightarrow D[j].arg[2] \neq i - 1.$$
- *is a symbolically secret key* iff it has the three previous properties.

$\diamond$

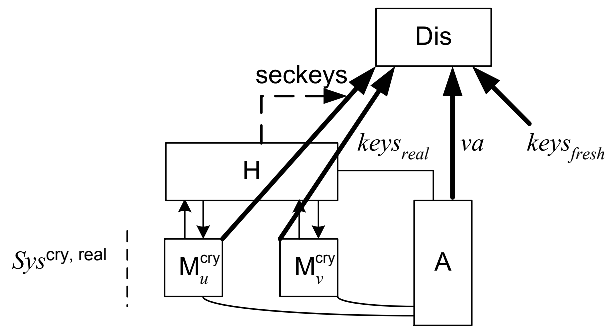


Figure 5: Cryptographic key secrecy

Essentially we want to show that symbolically secret keys are also cryptographically secret. However, the only direct correspondence between one particular symbolic key and one particular real key exists in a so-called combined system within the proof of the cryptographic library. Hence we will show both a close per-key relation for the combined system (Lemma 4.1) and a more abstract theorem that considers each of the real and ideal systems as a whole (Theorem 4.1). For the latter, we introduce a function seckeys based on the user view that indicates the keys that the users consider secret. We show that if this consideration is always correct in the ideal system in the symbolic sense, then it is also always correct in the real system in the cryptographic sense. In practical situations, such a function seckeys might denote "the second key that was exchanged between users u and v", or "all keys that were the results of a successful key-exchange protocol KX". In particular, the latter type of function seckeys is the symbolical formulation of secrecy goals on key exchange protocols. Formally, the function seckeys maps the user view to a set of triples (u, l^{hnd}, t) of a user, a handle, and a type, pointing to the supposedly secret keys.

Definition 4.2 *(Secret-key Belief Function)* A *secret-key belief function* for a set $\mathcal{H}$ (intuitively the indices of honest participants) is a function seckeys with domain Σ^* and range $(\mathcal{H} \times \mathcal{HNDS} \times \{\mathsf{ska}, \mathsf{skse}\})^*$. $\diamond$

We first define symbolic key secrecy for such a function. In addition to the conditions for individual keys, we require that all elements point to different terms, so that we can expect the corresponding list of cryptographic keys to be entirely random.

Definition 4.3 *(Symbolic Key Secrecy for the Ideal Cryptographic Library)* Let a user H suitable for a structure $(\{\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}\}, S_{\mathcal{H}}^{\mathsf{cry}})$ of the cryptographic library $Sys^{\mathsf{cry,id}}$ and a secret-key belief function seckeys for $\mathcal{H}$ be given. We say that the cryptographic library with this user *keeps the keys in* seckeys *strictly symbolically secret* iff for all configurations $conf = (\{\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}\}, S_{\mathcal{H}}^{\mathsf{cry}}, \mathsf{H}, \mathsf{A})$ of this structure, every $v \in view_{conf}(\mathsf{H})$, and every element $(u_i, l_i^{\mathsf{hnd}}, t_i)$ of the list $\mathsf{seckeys}(v)$, the term under $(u_i, l_i^{\mathsf{hnd}})$ is a symbolically secret key of type t_i, and $D[hnd_{u_i} = l_i^{\mathsf{hnd}}].ind \neq D[hnd_{u_j} = l_j^{\mathsf{hnd}}].ind$ for all $i \neq j$. $\diamond$

This definition lends itself to automated proof tools because it is entirely symbolic and belongs to the typical class of secrecy properties proven with such tools. The typical formulation is that no ideal adversary can obtain certain designated terms into its symbolic knowledge set. In the given model, the knowledge sets are defined by the possession of handles to terms.

We define cryptographic key secrecy similar to cryptographic definitions for key-exchange protocols: We demand that no polynomial-time adversary can distinguish the keys designated by the function seckeys from fresh keys. This is illustrated in Figure 5.

Definition 4.4 *(Cryptographic Key Secrecy for the Real Cryptographic Library)* Let a polynomial-time configuration $conf = (\hat{M}_{\mathcal{H}}^{\mathsf{cry}}, S_{\mathcal{H}}^{\mathsf{cry}}, \mathsf{H}, \mathsf{A})$ of the real cryptographic library $Sys_{\mathcal{E},\mathcal{S},\mathcal{A},\mathcal{SE}}^{\mathsf{cry,real}}$ and a secret-key belief function seckeys for $\mathcal{H}$ be given. Let $\mathsf{gen}_{\mathsf{A}}$ and $\mathsf{gen}_{\mathsf{SE}}$ denote the key generation algorithms of $\mathcal{A}$ and $\mathcal{SE}$, respectively. We say that this configuration *keeps the keys in* seckeys *cryptographically secret* iff for all probabilistic-polynomial time algorithms Dis (the distinguisher), we have

$$|\Pr[\mathsf{Dis}(1^k, va, keys_{real}) = 1] - \Pr[\mathsf{Dis}(1^k, va, keys_{fresh}) = 1]| \in NEGL$$

(as a function of the security parameter k), where the used random variables are defined as follows: For $r \in run_{conf}$, let $va := view_{conf}(\mathsf{A})(r)$ be the view of the adversary, let $(u_i, l_i^{\mathsf{hnd}}, t_i)_{i=1,\ldots,n} := \mathsf{seckeys}(view_{conf}(\mathsf{H})(r))$ be the user-handle-type triples of presumably secret keys, and let the keys be $keys_{real} := (sk_i)_{i=1,\ldots,n}$ with

$$sk_i \;:=\; D_{u_i}[hnd_{u_i} = l_i^{\mathsf{hnd}}].word \text{ if } D_{u_i}[hnd_{u_i} = l_i^{\mathsf{hnd}}].type = t_i, \text{ else } \epsilon;$$

and $keys_{fresh} := (sk_i')_{i=1,\ldots,n}$ with

$$\begin{aligned} sk_i' &\leftarrow \mathsf{gen}_{\mathsf{A}}(1^k) \text{ if } t_i = \mathsf{ska}, \\ sk_i' &\leftarrow \mathsf{gen}_{\mathsf{SE}}(1^k) \text{ if } t_i = \mathsf{skse}, \end{aligned}$$

and $sk_i' \leftarrow \epsilon$ otherwise. $\diamond$

4.2 Preservation of Key Secrecy

We can now state our main key-secrecy theorem: If for certain honest users H and a secret-key belief function seckeys, the ideal cryptographic library keeps the keys in seckeys symbolically secret, then every configuration of H with the real cryptographic library keeps the keys in seckeys cryptographically secret.

Theorem 4.1 *(Symbolic Key Secrecy Implies Cryptographic Key Secrecy)* Let a polynomial-time honest user H of a structure $(\{\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}\}, S_{\mathcal{H}}^{\mathsf{cry}})$ of the ideal cryptographic library $Sys^{\mathsf{cry,id}}$ and a secret-key belief function seckeys for $\mathcal{H}$ be given such that the cryptographic library with this user keeps the keys in seckeys strictly symbolically secret. Then every polynomial-time configuration $(\hat{M}_{\mathcal{H}}^{\mathsf{cry}}, S_{\mathcal{H}}^{\mathsf{cry}}, \mathsf{H}, \mathsf{A})$ of the real cryptographic library $Sys_{\mathcal{E},\mathcal{S},\mathcal{A},\mathcal{SE}}^{\mathsf{cry,real}}$ (with the same user H) keeps the keys in seckeys cryptographically secret. $\square$

This theorem makes statements about adversary handles and real keys, which only exist in either the ideal or the real cryptographic library, respectively. Hence the theorem cannot be proved solely as a consequence of the as-secure-as relation, in other words reactive simulatability, between these two systems, because reactive simulatability only concerns the indistinguishability of the views of the honest users H. We therefore extend the simulatability proof from [11, 12, 9] to the desired property. The basic proof structure is that a combined system $\mathsf{C}_{\mathcal{H}}^*$ is defined that essentially contains all elements of both the real and the ideal system. In particular, it contains a database structured like D but with an additional attribute $word$ for real bitstrings corresponding to the terms, as they are generated by the simulator. A second combined system $\mathsf{C}_{\mathcal{H}}$ contains the real bitstrings as generated by the real machines. An important invariant of $\mathsf{C}_{\mathcal{H}}^*$ is word secrecy, which states that no information flows from certain variables into others that are or may later become known to the adversary. We use the following word-secrecy lemma as a basis for our key secrecy proof.

Lemma 4.1 *(Word Secrecy with Symmetric Keys [11, 12, 9])* Let H and A be machines such that $(\hat{M}_{\mathcal{H}}^{\mathsf{cry}}, S_{\mathcal{H}}^{\mathsf{cry}}, \mathsf{H}, \mathsf{A})$ is a polynomial-time configuration of the real cryptographic library $Sys_{\mathcal{E},\mathcal{S},\mathcal{A},\mathcal{SE}}^{\mathsf{cry,real}}$. Then the following invariant holds in runs of the configuration $(\{\mathsf{C}_{\mathcal{H}}^*\}, S_{\mathcal{H}}^{\mathsf{cry}}, \mathsf{H}, \mathsf{A})$ except with negligible probability: Given a state $D_{\mathsf{C}_{\mathcal{H}}^*}$ of the database of the combined system, let the set Pub_Var of "public" variables contain

- all words $D_{\mathsf{C}_{\mathcal{H}}^*}[i].word$ with $D_{\mathsf{C}_{\mathcal{H}}^*}[i].hnd_{\mathsf{a}} \neq \downarrow$, i.e., the real messages where the adversary has learned the corresponding term symbolically,
- the state of A and H, and the $\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}$-part of the state of $\mathsf{C}_{\mathcal{H}}^*$,
- the secret keys of public-key schemes where the public keys are known to the adversary, i.e., the words $D_{\mathsf{C}_{\mathcal{H}}^*}[i].word$ with $D_{\mathsf{C}_{\mathcal{H}}^*}[i-1].type \in \{\mathsf{pke}, \mathsf{pks}\}$ and $D_{\mathsf{C}_{\mathcal{H}}^*}[i-1].hnd_{\mathsf{a}} \neq \downarrow$, and[1]
- the symmetric secret keys for which an encryption or authenticator is public, i.e., the words $D_{\mathsf{C}_{\mathcal{H}}^*}[i].word$ where an index j exists with $D_{\mathsf{C}_{\mathcal{H}}^*}[j].hnd_{\mathsf{a}} \neq \downarrow$ and $D_{\mathsf{C}_{\mathcal{H}}^*}[j].type \in \{\mathsf{aut}, \mathsf{symenc}\}$ and $D_{\mathsf{C}_{\mathcal{H}}^*}[j].arg[2] = i - 1$.

Then no information from other variables has flown into Pub_Var in the sense of information flow in programming languages, i.e., static program analysis. □

Lemma 4.1 gives the tight correspondence of symbolic secrecy and cryptographic secrecy for individual keys that was mentioned in the introductory sections. However, such per-key considerations only work for information-theoretic security; this is why the lemma is formulated for the combined system $\mathsf{C}_{\mathcal{H}}^*$ which contains some simulated aspects instead of the combined system $\mathsf{C}_{\mathcal{H}}$ with the completely real bitstrings; for $\mathsf{C}_{\mathcal{H}}$ we only show more abstract key secrecy similar to Definition 4.4.

The proof of Theorem 4.1 based on the word-secrecy lemma is postponed to the appendix.

5 Conclusion

For the first time, we have related symbolic secrecy as used in all usual automated proof tools for cryptographic protocols with real cryptographic secrecy notions. Symbolic secrecy is essentially defined by the absence of terms from an adversary's knowledge set, cryptographic secrecy by the indistinguishability of the real secret bitstrings from fresh random bitstrings of the same type, given the view of a real, cryptographic adversary. We based our results on the Dolev-Yao-style ideal cryptographic library from [11, 12, 9] and its provably secure implementation. We pointed out why symbolic secrecy does not imply cryptographic secrecy for all terms and in all situations and therefore investigated two particularly important cases separately, message (payload) secrecy and key secrecy. For the former, we came up with a general cryptographic secrecy definition that separates information leakage about a payload by the users themselves from information leakage in the system, and we showed that symbolic key secrecy of the protocol implies that no information leaks in the protocol. For key secrecy, we defined realistic, symbolically verifiable conditions beyond the absence of a key from the adversary's knowledge set and showed that these conditions imply full cryptographic secrecy of the corresponding real key.

References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 36–47, 1997.

[1] These secret keys are included because information from them flows into the public keys, but they do not get adversary handles when the public keys are published.

[2] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. 4th International Symposium on Theoretical Aspects of Computer Software (TACS)*, pages 82–94, 2001.

[3] M. Abadi and P. Rogaway. Reconciling two views of cryptography: The computational soundness of formal encryption. In *Proc. 1st IFIP International Conference on Theoretical Computer Science*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2000.

[4] M. Backes. A cryptographically sound Dolev-Yao style security proof of the Otway-Rees protocol. In *Proc. 9th European Symposium on Research in Computer Security (ESORICS)*, volume 3193 of *Lecture Notes in Computer Science*, pages 89–108. Springer, 2004.

[5] M. Backes and C. Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 675–686. Springer, 2003.

[6] M. Backes and B. Pfitzmann. Computational probabilistic non-interference. In *Proc. 7th European Symposium on Research in Computer Security (ESORICS)*, volume 2502 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2002.

[7] M. Backes and B. Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 1–12, 2003. Full version in IACR Cryptology ePrint Archive 2003/121, Jun. 2003, `http://eprint.iacr.org/`.

[8] M. Backes and B. Pfitzmann. Intransitive non-interference for cryptographic purposes. In *Proc. 24th IEEE Symposium on Security & Privacy*, pages 140–152, 2003.

[9] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW)*, 2004. Full version in IACR Cryptology ePrint Archive 2004/059, Feb. 2004, `http://eprint.iacr.org/`.

[10] M. Backes, B. Pfitzmann, M. Steiner, and M. Waidner. Polynomial fairness and liveness. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW)*, pages 160–174, 2002.

[11] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proc. 10th ACM Conference on Computer and Communications Security*, pages 220–230, 2003. Full version in IACR Cryptology ePrint Archive 2003/015, Jan. 2003, `http://eprint.iacr.org/`.

[12] M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. In *Proc. 8th European Symposium on Research in Computer Security (ESORICS)*, volume 2808 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2003. Extended version in IACR Cryptology ePrint Archive 2003/145, Jul. 2003, `http://eprint.iacr.org/`.

[13] M. Backes, B. Pfitzmann, and M. Waidner. A general composition theorem for secure reactive systems. In *Proc. 1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2004.

[14] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[15] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *Proc. 24th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 34–39, 1983.

[16] J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In *Advances in Cryptology: CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 548–564. Springer, 2003.

[17] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[18] P. Laud. Semantics and program analysis of computationally secure information flow. In *Proc. 10th European Symposium on Programming (ESOP)*, pages 77–91, 2001.

[19] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[20] G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 18–30, 1997.

[21] C. Meadows. Using narrowing in the analysis of key management protocols. In *Proc. 10th IEEE Symposium on Security & Privacy*, pages 138–147, 1989.

[22] M. Merritt. *Cryptographic Protocols*. PhD thesis, Georgia Institute of Technology, 1983.

[23] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[24] J. K. Millen. The interrogator: A tool for cryptographic protocol security. In *Proc. 5th IEEE Symposium on Security & Privacy*, pages 134–141, 1984.

[25] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Cryptology*, 6(1):85–128, 1998.

[26] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE Symposium on Security & Privacy*, pages 184–200, 2001. Extended version of the model (with Michael Backes) IACR Cryptology ePrint Archive 2004/082, `http://eprint.iacr.org/`.

[27] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *Proc. 8th IEEE Computer Security Foundations Workshop (CSFW)*, pages 98–107, 1995.

[28] S. Schneider. Security properties and CSP. In *Proc. 17th IEEE Symposium on Security & Privacy*, pages 174–187, 1996.

[29] A. C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

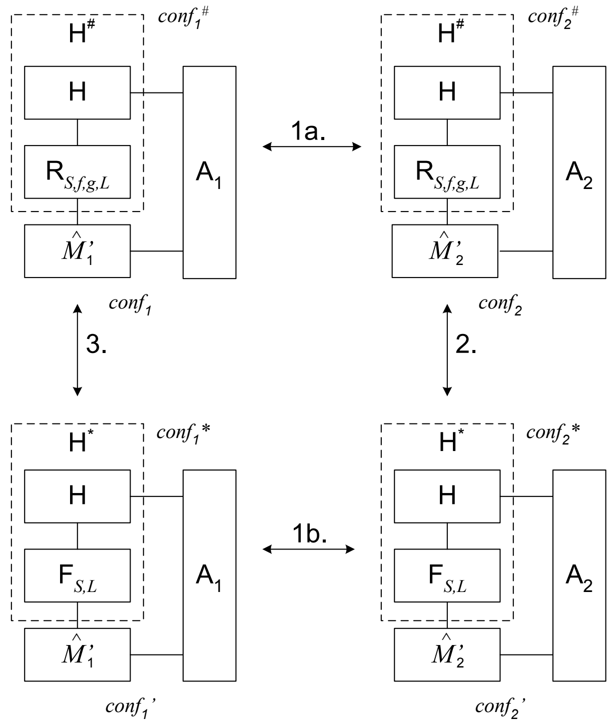


Figure 6: Overview of the proof of the general preservation theorem for message secrecy

A Postponed Proofs

A.1 Proof of Payload Secrecy Preservation

Proof. (Theorem 3.1) Let $(\hat{M}_1, S) \in Sys_1$ denote an arbitrary structure, f and g the given payload selection functions, and L a polynomially bounded function. Let further $(\hat{M}'_1, S')$ be the structure where the port names of ports in S are consistently replaced on the machines as for the port set S' in $\mathsf{R}_{S,f,g,L}$.

Let $conf_1 = (\hat{M}'_1 \cup \{\mathsf{R}_{S,f,g,L}\}, S, \mathsf{H}, \mathsf{A}_1)$ for an arbitrary polynomial-time user H and an arbitrary polynomial-time adversary A_1, and let $conf'_1 = (\hat{M}'_1 \cup \{\mathsf{F}_{S,L}\}, S, \mathsf{H}, \mathsf{A}_1)$ for the same H and A_1. We have to show that $view_{conf_1}(\mathsf{H}) \approx view_{conf'_1}(\mathsf{H})$. The proof is conducted in three steps, which are illustrated in Figure 6.

1. First, we combine the user H and the replacement machine $\mathsf{R}_{S,f,g,L}$ yielding a new machine $\mathsf{H}^{\#}$. Combination of H with the forwarding machine $\mathsf{F}_{S,L}$ similarly yields a machine H^*. Since H is polynomial-time by assumption and since $\mathsf{F}_{S,L}$ and $\mathsf{R}_{S,f,g,L}$ are polynomial-time because L is polynomially bounded, $\mathsf{H}^{\#}$ and H^* constitute valid polynomial-time users for interacting with the structure $(\hat{M}'_1, S')$. Moreover, there exists a combination lemma in the underlying model stating that the view of a machine before the combination is identical to the view of the same machine considered as a submachine of the combined machine. This yields

$$view_{conf_1}(\mathsf{H}) = view_{conf_1^{\#}}(\mathsf{H}) \quad \text{and} \quad view_{conf'_1}(\mathsf{H}) = view_{conf_1^*}(\mathsf{H}),$$

   where $conf_1^{\#} = (\hat{M}'_1, S', \mathsf{H}^{\#}, \mathsf{A}_1)$, $conf_1^* = (\hat{M}'_1, S', \mathsf{H}^*, \mathsf{A}_1)$, and the views of H in $conf_1^{\#}$ and $conf_1^*$ are defined in the aforementioned sense as a well-defined function on the view of $\mathsf{H}^{\#}$ and H^*, respectively.

Now $Sys_1 \geq_{\mathsf{sec}}^{\mathsf{univ}} Sys_2$ implies there exist configurations $conf_2^{\#} = (\hat{M}_2', S', \mathsf{H}^{\#}, \mathsf{A}_2)$ and $conf_2^* = (\hat{M}_2', S', \mathsf{H}^*, \mathsf{A}_2')$ such that $view_{conf_1^{\#}}(\mathsf{H}^{\#}) \approx view_{conf_2^{\#}}(\mathsf{H}^{\#})$ and $view_{conf_1^*}(\mathsf{H}^*) \approx view_{conf_2^*}(\mathsf{H}^*)$. Universal simulatability further implies that both A_2 and A_2' may only depend on the machines of the structure and on the adversary in $conf_1$ and $conf_1'$, respectively. Since the machines $\hat{M}_1'$ of the structure and the adversary A_1 are identical in both configurations, we obtain $\mathsf{A}_2 = \mathsf{A}_2'$. Projecting the view of $\mathsf{H}^{\#}$ and H^* to the view of its submachine H in the considered four configurations then yields

$$view_{conf_1^{\#}}(\mathsf{H}) \approx view_{conf_2^{\#}}(\mathsf{H}) \quad \text{and} \quad view_{conf_1^*}(\mathsf{H}) \approx view_{conf_2^*}(\mathsf{H}),$$

where we have exploited that applying a function (here the projection) to families of indistinguishable random variables yields families of indistinguishable random variables again. Finally, the combination lemma yields

$$view_{conf_2^{\#}}(\mathsf{H}) = view_{conf_2}(\mathsf{H}) \quad \text{and} \quad view_{conf_2^*}(\mathsf{H}) = view_{conf_2'}(\mathsf{H}),$$

where $conf_2 = (\hat{M}_2' \cup \{\mathsf{R}_{S,f,g,L}\}, S, \mathsf{H}, \mathsf{A}_2)$ and $conf_2' = (\hat{M}_2' \cup \{\mathsf{F}_{S,L}\}, S, \mathsf{H}, \mathsf{A}_2)$.

2. Now by assumption, we have $Sys_2 \approx [f,g]Sys_2$, hence in particular $(\hat{M}_2, S) \approx [f,g](\hat{M}_2, S)$ for the structure $(\hat{M}_2, S) \in Sys_2$ that satisfies that replacing the port names of ports in S as for the port set S' in $\mathsf{R}_{S,f,g,L}$ yields the machines $\hat{M}_2'$. Then the definition of payload secrecy applied to the configurations $conf_2$ and $conf_2'$ in particular implies

$$view_{conf_2}(\mathsf{H}) \approx view_{conf_2'}(\mathsf{H}).$$

Note that $conf_2$ and $conf_2'$ are valid choices with respect to the definition of messages secrecy, since universal simulatability implied that the adversaries in both configurations are identical.

3. Finally, we exploit the transitivity of $\approx$ applied to the view of H in all eight configurations, which yields $view_{conf_1}(\mathsf{H}) \approx view_{conf_1'}(\mathsf{H})$. This finishes the proof.

■

A.2 Proof of Symbolic versus Cryptographic Payload Secrecy

Proof. (Theorem 3.2.) With the notation of Definition 3.4, let $conf = (\hat{M}_{\mathcal{H}}' \cup \{\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}, \mathsf{R}_{S_{\mathcal{H}},f,g,L}\}, S_{\mathcal{H}}, \mathsf{H}, \mathsf{A})$ for a set $\mathcal{H}$ and an arbitrary user H and adversary A, and let $conf' = (\hat{M}_{\mathcal{H}}' \cup \{\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}, \mathsf{F}_{S_{\mathcal{H}},L}\}, S_{\mathcal{H}}, \mathsf{H}, \mathsf{A})$, i.e., we have a configuration of the protocol over the ideal cryptographic library with and without payload replacement. We have to show that the views of H are equal in the two configurations. The proof is by induction over the steps of the runs. We show the following stronger invariants, where D denotes the state of the database of $\mathsf{TH}_{\mathcal{H}}^{\mathsf{cry}}$ in $conf$ and D' that in $conf'$:

1. The joint view of H and A is identical in $conf$ and $conf'$.
2. If $D[i].type \neq \mathsf{data}$ or $D[i].arg[1] \neq n$ for all $n \in \{n \mid \exists m : (m,n) \in T\}$, then $D[i] = D'[i]$.
3. If $D[i].type = \mathsf{data}$ and there exists $(m,n) \in T$ such that $D[i].arg[1] = n$, then $D[i] = D'[i]$ except that $D'[i].arg[1] = m$.
4. If the values v and v' of a variable of $\hat{M}_{\mathcal{H}}'$ are different in $conf$ and $conf'$, then $v = n$ and $v' = m$ for a pair $(m,n) \in T$.

To prove this, we consider an arbitrary prefix of a run in each configuration where the invariants are fulfilled, and an arbitrary next step in both configurations. By a step we typically mean a machine transition, except that we consider individual program execution steps within $\hat{M}'_{\mathcal{H}}$. For simplicity, we assume that an input to $\hat{M}'_{\mathcal{H}}$ is first stored in a variable and outputs of $\hat{M}'_{\mathcal{H}}$ come directly from a variable.

- A message between H and A clearly retains the invariants.
- So does an input from H to $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$ or $\mathsf{F}_{S_{\mathcal{H}},L}$, but it may lead to different outputs n from $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$ and m from $\mathsf{F}_{S_{\mathcal{H}},L}$ for the next step, where $(m,n) \in T$.
- Such different inputs from $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$ or $\mathsf{F}_{S_{\mathcal{H}},L}$ to $\hat{M}'_{\mathcal{H}}$ may lead to different variable values in $\hat{M}'_{\mathcal{H}}$, but this difference does not invalidate Invariant 4. Equal inputs clearly retain the invariants.
- Steps within $\hat{M}'_{\mathcal{H}}$ retain Invariant 4; in particular the program execution remains synchronized between conf and conf' because no information flow except by assignments is allowed from the unequal variables.
- Inputs from $\hat{M}'_{\mathcal{H}}$ to the cryptographic library can only differ in arguments of the command store by a precondition; then a payload n is stored in conf and m in conf' with $(m,n) \in T$. Hence Invariant 3 is maintained.
- Outputs from the cryptographic library to the adversary A can only differ if a corresponding input command operates on an entry of type data and makes an output to A. This is only the case for a command $\mathsf{retrieve}$ input by A. However, a differing entry has a value n in conf and m in conf' with $(m,n) \in T$ by Invariants 2 and 3. Such an entry in conf has no adversary handle by a precondition, and by Invariant 3 also not in conf'. Hence no such output can happen, and Invariant 1 is retained.
- An output from the cryptographic library to $\hat{M}'_{\mathcal{H}}$ can only differ if it is the result of a command $\mathsf{retrieve}$ on differing data. By similar arguments as in the previous cases, Invariant 4 is retained.
- If an output value from $\hat{M}'_{\mathcal{H}}$ is o to $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$ in conf and o' to $\mathsf{F}_{S_{\mathcal{H}},L}$ in conf', then $\mathsf{F}_{S_{\mathcal{H}},L}$ forwards o'. We want to show that so does $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$. By a precondition o and o' differ at most in fields selected by the function g, and the field value is then n in o and m in o' with $(m,n) \in T$. Hence $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$ replaces these fields by m, making them equal to the corresponding fields in o'. Conversely, every field of o that is selected by g is such a value n that arose by direct assignment of a value n from the replacement table T, and thus the corresponding value in conf' is m, so that the replacement is correct.
- Outputs from $\mathsf{R}_{S_{\mathcal{H}},f,g,L}$ or $\mathsf{F}_{S_{\mathcal{H}},L}$ to H are always equal, as we just saw, and thus retain Invariant 1.

Putting everything together, we have shown that $\mathit{view}_{\mathit{conf}}(\mathsf{H}) = \mathit{view}_{\mathit{conf}'}(\mathsf{H})$. Hence the payload messages selected by f and g are perfectly secret. ■

A.3 Proof of Symbolic versus Cryptographic Key Secrecy

Before actually proving Theorem 4.1, we give an overview of the underlying simulatability proof from [11, 12, 9] that we extend. Figure 7 gives an overview of the original proof. The top row shows the real configuration and the ideal configuration with the simulator. The basic proof structure is that a combined system $\mathsf{C}_{\mathcal{H}}$ (lower right in Figure 7) is defined that essentially contains all elements of both the real and the ideal system. In particular, it contains a database structured like D but with an additional

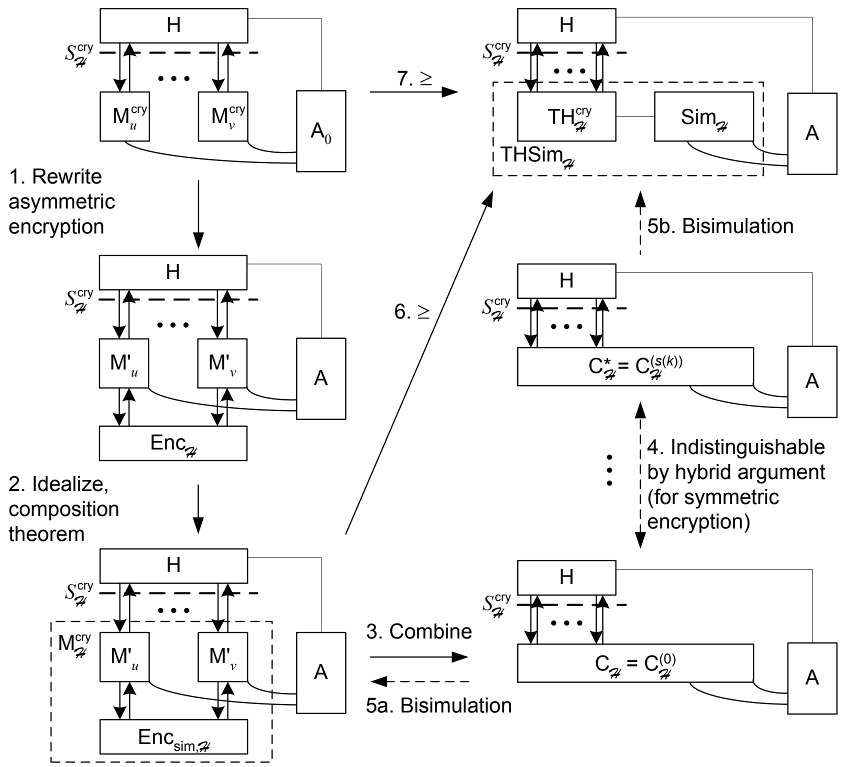


Figure 7: Overview of the proof of correct simulation for the cryptographic library

attribute $word$ for the real bitstrings corresponding to the terms. Then bisimulations are proved between $\mathsf{C}_{\mathcal{H}}$ and the real machines, and between $\mathsf{C}_{\mathcal{H}}$ and the trusted host with the simulator (Steps 5a and 5b of Figure 7). A bisimulation, however, cannot deal with computational indistinguishability. Hence at the beginning of the proof, the real asymmetric encryptions are replaced by simulated ones as made in the simulator (there, all ciphertexts where the plaintext is symbolically secret contain a fixed plaintext string instead), using a low-level idealization of asymmetric encryption and the composition theorem (Steps 1 and 2 of Figure 7). Symmetric encryption cannot be treated with such a simple one-step replacement. The successive exchange of real encryptions for simulated encryptions is therefore done by a so-called hybrid argument (Step 4 in Figure 7) that considers multiple indexed combined systems $\mathsf{C}^{(i)}_{\mathcal{H}}$, each replacing the encryptions with one key. The bisimulation mappings from the initial and final combined systems to the real and ideal system, respectively, are called derivations because they essentially extract the relevant elements from the combined systems unchanged.

An important invariant of the combined system $\mathsf{C}^*_{\mathcal{H}}$ is word secrecy, which states that no information flows from certain variables into others that are or may later become known to the adversary. It was stated in Lemma 4.1. We are now ready to present the proof of the key secrecy theorem.

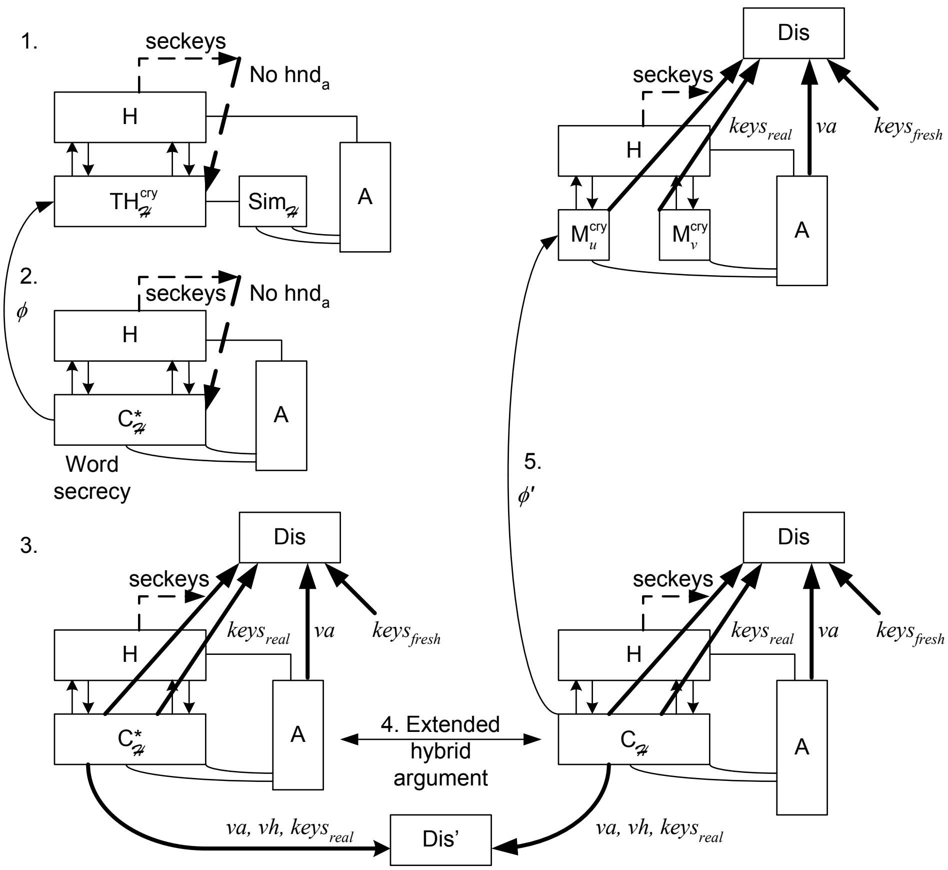


Figure 8: Key secrecy in ideal, combined, and real cryptographic library

Proof. (Theorem 4.1.) We fix a polynomial-time user H and a polynomial-time adversary A suitable for the real cryptographic library and thus for all the configurations shown in Figures 7 and 8. We assume that the ideal cryptographic library keeps the keys in seckeys strictly symbolically secret.

Symbolic secrecy in $\mathsf{C}^*_{\mathcal{H}}$. The derivation of the ideal system from the combined one, i.e., the bisimulation ϕ in Figure 8, maps all state elements of the ideal system identically. By the bisimulation property this derivation is invariant and the view of H equal in runs of the ideal or combined system except on a negligible error set. As strict symbolic key secrecy is defined in terms of state elements of the ideal system and the view of H, it is also fulfilled in the combined system $\mathsf{C}^*_{\mathcal{H}}$ except with negligible probability ϵ (a function of k), i.e., the terms designated by seckeys are different secret keys of the correct types, do not have adversary handles, and are unused.

Cryptographic secrecy in $\mathsf{C}^*_{\mathcal{H}}$ via word secrecy. It follows immediately that, still in $\mathsf{C}^*_{\mathcal{H}}$, the word attributes of terms designated by seckeys are not in the set Pub_Var, except with probability ϵ. These words are exactly the random variable $keys_{real}$, if we define this random variable for each combined system by $sk_i := D[hnd_{u_i} = l_i^{\mathsf{hnd}}].word$ if $D[hnd_{u_i} = l_i^{\mathsf{hnd}}].type = t_i$, else ϵ. By word secrecy for $\mathsf{C}^*_{\mathcal{H}}$, no static information flow therefore takes place from $keys_{real}$ into variables in Pub_Var, and thus in particular into the view va of A, except with probability ϵ.

We now fix a distinguisher Dis as in the definition of cryptographic key secrecy, i.e., it gets inputs $(1^k, va, keys_{real})$ or $(1^k, va, keys_{fresh})$, where the keys in $keys_{fresh}$ are by definition generated by the same algorithms as those in $keys_{real}$, but independently of the system run. Total absence of information flow would imply that va contains no Shannon information about $keys_{real}$, and thus the two distributions would be perfectly indistinguishability. In reality, the distinguisher Dis may only have an advantage over this situation in the runs in the error set, and thus its advantage is negligible.

Cryptographic secrecy in $\mathsf{C}_{\mathcal{H}}$ via hybrid argument. Next we show that the advantage of Dis is still negligible for the combined system $\mathsf{C}_{\mathcal{H}}$, which contains real instead of simulated symmetric encryptions. Assume for contradiction that it were not. We then construct a machine Dis', called *extended system distinguisher*, that can distinguish the views vh of H and va of A and additionally $keys_{real}$. From its inputs Dis' computes $l := \mathsf{seckeys}(vh)$. Given the key types in l, it can generate a suitable list $keys_{fresh}$. It then runs Dis on the adversary view va and either $keys_{real}$ or $keys_{fresh}$. The result for the two types of keys is, by the assumption, significantly different for $\mathsf{C}_{\mathcal{H}}$ but not for $\mathsf{C}^*_{\mathcal{H}}$. This allows Dis' to distinguish $\mathsf{C}_{\mathcal{H}}$ and $\mathsf{C}^*_{\mathcal{H}}$ with not negligible advantage.

Our result does not yet contradict the indistinguishability of $\mathsf{C}_{\mathcal{H}}$ and $\mathsf{C}^*_{\mathcal{H}}$ from the original proof because our extended distinguisher also gets $keys_{real}$ as input. We therefore have to extend the hybrid argument to extended distinguishers. The framework of the hybrid argument can remain identical; we only need to show that Dis' cannot distinguish any two neighboring hybrid systems. Two such hybrids differ only in making either real or simulated encryptions with one particular symmetric key $sk^{(i)}$, which is defined as the i-th key used for encryption. The proof uses a machine $\mathsf{SymComb}$ that contains one symmetric encryption key sk^* and a bit b and, depending on b, makes either real or simulated encryptions with sk^*, and in the latter case answers decryption requests by table look-up. A lemma in [9] states that the two cases of b are indistinguishable. We want to show for contradiction that if Dis' can distinguish two hybrids, one can also distinguish the two cases of b in $\mathsf{SymComb}$. This would be trivial if the key $sk^{(i)}$ in the hybrids were only used for en- and decryption; one could simply realize the two hybrids by a fixed part $\mathsf{C'}^{(i)}_{\mathcal{H}}$ in combination with $\mathsf{SymComb}$ with either $b = 0$ or $b = 1$. The proof still essentially works like this, but the key $sk^{(i)}$ might also be put into lists, sent around, etc. This cannot be done with the internal key sk^* from $\mathsf{SymComb}$. Hence $\mathsf{C'}^{(i)}_{\mathcal{H}}$ keeps its own key $sk^{(i)}$ for these purposes. In [9] it is shown that this use of two different keys instead of one is perfectly indistinguishable for normal (non-extended) distinguishers. (This only holds because of the precise order in which the different keys are treated in the successive hybrids.) The proof of perfect indistinguishability shows that no information about the "outer" $sk^{(i)}$ used by $\mathsf{C'}^{(i)}_{\mathcal{H}}$ flows into the view of H and A. These proof parts are still true, but we have to add a third part showing that no information about $sk^{(i)}$ flows into the additional input $keys_{real}$ for the extended distinguisher Dis'.

As $keys_{real}$ consists of keys generated by the honest users, and thus with the correct key generation algorithms, no information about $sk^{(i)}$ flows into the list $keys_{real}$ unless $sk^{(i)}$ is one of the keys in $keys_{real}$. However, by the definition of the hybrids, $sk^{(i)}$ is a used key, and by the correctness of $\mathsf{seckeys}$, the list $keys_{real}$ only consists of unused keys. Hence this can indeed be excluded. This finishes the proof that the hybrid argument is still correct for the more powerful distinguishers Dis', and thus the proof that cryptographic key secrecy holds for the most real combined system $\mathsf{C}_{\mathcal{H}}$.

Cryptographic secrecy in the real system. Finally, the derivation of the real system from the combined one, i.e., the bisimulation ϕ' in Figure 7, maps all the user handles and all the word attributes corresponding to them identically, and thus in particular the list $keys_{real}$. By the bisimulation property this derivation is invariant and the view of A equal in runs of the combined or real system except on a negligible error set. Hence the advantage of Dis can only differ by a negligible function, as its inputs only depend on these invariant values. Thus the advantage of Dis is also negligible on the real system. ■