

On the Cryptographic Key Secrecy of the Strengthened Yahalom Protocol^{*}

Michael Backes¹ and Birgit Pfitzmann²

¹ Saarland University, Germany

² IBM Zurich Research Laboratory, Switzerland

Abstract. Symbolic secrecy of exchanged keys is arguably one of the most important notions of secrecy shown with automated proof tools. It means that an adversary restricted to symbolic operations on terms can never get the entire key into its knowledge set. Cryptographic key secrecy essentially means computational indistinguishability between the real key and a random one, given the view of a much more general adversary.

We analyze the cryptographic key secrecy for the strengthened Yahalom protocol, which constitutes one of the most prominent key exchange protocols analyzed symbolically by means of automated proof tools. We show that the strengthened Yahalom protocol does not guarantee cryptographic key secrecy. We further show that cryptographic key secrecy can be proven for a slight simplification of the protocol by exploiting recent results on linking symbolic and cryptographic key secrecy in order to perform a symbolic proof of secrecy for the simplified Yahalom protocol in a specific setting that allows us to derive the desired cryptographic key secrecy from the symbolic proof. The proof holds in the presence of arbitrary active attacks provided that the protocol is relying on standard provably secure cryptographic primitives.

1 Introduction

Cryptographic protocols for key establishment are an established technology. Nevertheless, most new networking and messaging stacks come with new protocols for such tasks. Since designing cryptographic protocols is known to be error-prone and, owing to the distributed-system aspects of multiple interleaved protocol runs, security proofs of such protocols are awkward to make for humans, 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 after the first authors [17], e.g., see [24, 1, 29, 31]. 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 $E(E(m))$. A typical cancellation rule is $D(E(m)) = m$ 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 if results obtained over a Dolev-Yao model are also

^{*} An extended version of this paper is available at [7].

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 [17], and in particular key secrecy properties. Symbolically, the secrecy of a key is represented by knowledge sets: The key is secret if the adversary can never get the corresponding symbolic term into its knowledge set. Cryptographically, key secrecy is defined by computational indistinguishability between the real key 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 automated proofs tools, while cryptographic secrecy constitutes a more fine-grained notion of secrecy that is much harder to establish.

The Yahalom protocol [14, 32] is one of the most prominent key exchange protocols. Paulson discovered that the original protocol from [14] is insecure and proposed a strengthened variant [32]. This was extensively investigated, e.g., in [32, 19, 11]. However, all existing security proofs are restricted to the Dolev-Yao model. We first remark that the protocol does not satisfy the definition of cryptographic key secrecy because of the (essentially) well-known fact that computing an encryption with the key as a form of confirmation which becomes known also to the adversary (as is the case for the Yahalom protocol) already makes the key distinguishable from a fresh random key. We show that a simplified version of the protocol obtained by removing the encryption computed with the exchanged key allows for a proof of cryptographic key secrecy, i.e., we show that keys exchanged between two honest users are secret in the strong sense of indistinguishability from random keys. This holds in the presence of arbitrary active attacks, provided that the Dolev-Yao abstraction of symmetric encryption is implemented by a symmetric encryption scheme that is secure against chosen-ciphertext attacks and additionally ensures integrity of ciphertexts. This is the standard security definition of authenticated symmetric encryption [13, 12]. Efficient symmetric encryptions schemes provably secure in this sense exist under reasonable assumptions [18, 23].

We achieve this result by analyzing the simplified version of the strengthened Yahalom protocol based on the *cryptographic library* of Backes, Pfitzmann, and Waidner [9, 10, 6], which corresponds to a slightly extended Dolev-Yao model that can be faithfully realized using provably secure cryptographic primitives in the standard model of cryptography. In combination with a recent result on linking symbolic and cryptographic key secrecy [8], this allows us to perform a proof of secrecy for the simplified Yahalom protocol in a specific, symbolic setting and to derive the desired cryptographic key secrecy from that. This is the first symbolic proof of a cryptographic protocol that can be exploited to derive cryptographic secrecy for the exchanged keys. (Another such proof was conducted concurrently and independently by Canetti and Herzog, cf. below.)

Further Related Work. Early work on linking Dolev-Yao models and cryptography [3, 2, 20, 25] only considered passive attacks, and therefore cannot make general statements about protocols. A cryptographic justification for a Dolev-Yao model in the sense of simulatability [33], i.e., under active attacks and within arbitrary surrounding interactive protocols, was first given in [9] with extensions in [10, 6]. Based on that Dolev-Yao model, the well-known Needham-Schroeder-Lowe and Otway-Rees protocols were

proved in [5, 4]. The former is entirely an authentication proof and hence does not have to reason about secrecy aspects. The latter contains a key secrecy property but this was reformulated by hand into a (considerably weaker) integrity property so that the integrity preservation theorem could be used.

Laud [26] has presented a cryptographic underpinning for a Dolev-Yao model of symmetric encryption under active attacks. His work is directly connected with a formal proof tool, but it is specific to certain confidentiality properties and protocol classes. Herzog et al. [21] and Micciancio and Warinschi [30] have also given a cryptographic underpinning under active attacks. Their results are narrower than that in [9] since they are specific for public-key encryption and certain protocol classes, but consider slightly simpler real implementations. Efforts are also under way to formulate syntactic calculi for dealing with probabilism and polynomial-time considerations, in particular [27, 22] and, as a second step, to encode them into proof tools. This approach can not yet handle protocols with any degree of automation.

Cortier and Warinschi [16] have shown that symbolically secret nonces are also computationally secret, i.e., indistinguishable from a fresh random value given the view of a cryptographic adversary. Backes and Pfitzmann [8] and Canetti and Herzog [15] have established new symbolic criteria for proving a key cryptographically secret.

2 The Strengthened Yahalom Protocol

The Yahalom protocol [14] and its strengthened variant [32] are four-step protocols for establishing a shared secret encryption key between two users. The protocol relies on a distinguished trusted party T , and it is assumed that every user u initially shares a secret key K_{ut} with T . Expressed in the typical protocol notation as in, e.g., [28], the strengthened Yahalom works as follows.

1. $u \rightarrow v : u, N_u$
2. $v \rightarrow T : v, N_v, (u, N_u)_{K_{vt}}$
3. $T \rightarrow u : N_v, (v, K_{uv}, N_u)_{K_{ut}}, (u, v, K_{uv}, N_v)_{K_{vt}}$
4. $u \rightarrow v : (u, v, K_{uv}, N_v)_{K_{vt}}, (N_v)_{K_{uv}}$.

User u seeks to share a new session key with user v . It generates a nonce N_u and sends it to v together with its identity (first message). Next, v generates a new nonce N_v , creates a new message containing the identity u and the nonce N_u , and encrypts it with the key it shares with T . Then v sends its identity, its nonce N_v , and the encryption to the trusted party (second message). Now T decrypts the encryption yielding the identity of u and the nonce N_u , generates a fresh key K_{uv} for u and v , generates a message according to the protocol description, and sends it to u (third message). Then u decrypts the first encryption and tests whether the contained nonce is the one it sent to v before, i.e., to the identity that is contained in this encryption. If so, it forwards the second encryption to v (fourth message) together with an encryption of N_v with the shared key K_{uv} and terminates the protocol by outputting a handle to the shared secret key K_{uv} to its user. Finally v decrypts the first encryption contained in this message, obtains the shared key K_{uv} , and tests if the message contains its own identity and the contained nonce

was previously sent to T. If so, it further decrypts the second encryption and checks if the obtained nonce matches N_v . It then outputs a handle to the shared key K_{uv} to its user and terminates the protocol. Note that the fourth message of the strengthened Yahalom protocol contains an encryption of the nonce N_v with the shared key K_{uv} . We show below that this encryption destroys cryptographic key secrecy. We subsequently analyze the protocol obtained by removing this encryption, and we show cryptographic key secrecy for this protocol. We only briefly note that the authenticity guarantees of the Yahalom protocol [32] still hold in our setting if we omit this encryption from the fourth message, since our cryptographic implementation is already based on an authenticated symmetric encryption scheme; authenticated encryption is necessary for exploiting the underlying proof of cryptographic soundness of the Dolev-Yao model.

2.1 Why the Strengthened Yahalom Protocol does not offer Cryptographic Key Secrecy

We now briefly sketch why the unmodified strengthened Yahalom protocol cannot achieve cryptographic key secrecy. The argument is general (and obvious for a cryptographer) and applies to *all* key exchange protocols that already use the exchanged key to compute an encryption that becomes known to the adversary. The reason is that symmetric encryptions provide partial information about a symmetric secret key, at least if one also has partial information about the message encrypted. This partial information already allows an adversary to distinguish the exchanged key from a random key that has been chosen independently of the protocol. To distinguish the keys, the adversary first completes a regular execution of the protocol between two honest parties, thus learning the nonce N_v and the encryption $(N_v)_{K_{uv}}$. A bit b is then flipped, and the adversary receives a key K , which equals the (unknown) key K_{uv} if $b = 0$ or a fresh random key if $b = 1$. The adversary then decrypts $(N_v)_{K_{uv}}$ with K yielding N and outputs $b^* := 0$ as its guess on b if $N = N_v$, and $b^* := 1$ otherwise. The probability of a correct guess is then given by $1 - \epsilon$, where ϵ denotes the probability that for a randomly chosen nonce N_v and randomly chosen keys K_{uv}, K , we have that $(N_v)_{K_{uv}}$ decrypted with K yields N_v again, which is negligible. The adversary has thus a non-negligible advantage of distinguishing the keys. Hence cryptographic key secrecy does not hold.

2.2 Protocol Details with the Dolev-Yao-style Cryptographic Library

We now capture the simplified version of the strengthened Yahalom protocol, i.e., without the encryption $(N_v)_{K_{uv}}$ in its fourth step, using the Dolev-Yao-style cryptographic library from [9]. For simplicity, we speak of the Yahalom protocol again in the following. Almost all formal proof techniques for protocols first need a reformulation of the protocol into a more detailed version than the four steps above. These details include necessary tests on received messages, the types and generation rules for values like u and N_u , and a surrounding framework specifying the number of participants, the possibilities of multiple protocol runs, and the adversary capabilities. We now present the protocol details when using the Dolev-Yao-style cryptographic library from [9] as well as general aspects of this framework.

Algorithm 1 Evaluation of User Inputs in M_u^{Ya} with $u \neq T$ (Protocol Start)

Input: $(\text{new_prot}, \text{Yahalom}, v)$ at $\text{KE_in}_u?$ with $v \in \{1, \dots, n\} \setminus \{u\}$.

- 1: $n_u^{\text{hnd}} \leftarrow \text{gen_nonce}()$.
 - 2: $\text{Nonce}_u := \text{Nonce}_u \cup \{(n_u^{\text{hnd}}, v, 1)\}$.
 - 3: $u^{\text{hnd}} \leftarrow \text{store}(u)$.
 - 4: $m_1^{\text{hnd}} \leftarrow \text{list}(u^{\text{hnd}}, n_u^{\text{hnd}})$.
 - 5: $\text{send_i}(v, m_1^{\text{hnd}})$.
-

We write “:=” for deterministic assignment, and \downarrow is an error element available as an addition to the domains and ranges of all functions and algorithms. The framework is automata-based, i.e., protocols are executed by interacting machines, and event-based, i.e., machines react on received inputs. By M_i^{Ya} we denote the Yahalom machine for a participant i ; it can act in the roles of both u and v above.

The first type of input that M_i^{Ya} can receive is a start message $(\text{new_prot}, \text{Yahalom}, v)$ from its user denoting that it should start a protocol run with user v . The number of users is called n .¹ User inputs are distinguished from network inputs by arriving at a so-called port $\text{KE_in}_u?$. The “?” for input ports follows the CSP convention, and “KE” stands for key exchange because the user interface is the same for all key exchange protocols. The reaction on this input, i.e., the sending of the first message, is described in Algorithm 1. The command `gen_nonce` generates the nonce. M_u^{Ya} stores the resulting so-called *handle* n_u^{hnd} (a local name that this machine has for the corresponding term) in a set Nonce_u for future comparison together with the identity v and an indicator that this nonce was generated and stored by u in the first step. The set Nonce_u formally consists of triples (n^{hnd}, w, j) where n^{hnd} is a handle, $w \in \{1, \dots, n\} \setminus \{u\}$, and $j \in \{1, 2, 3, 4\}$. A triple (n^{hnd}, w, j) means that M_u^{Ya} stored the handle n^{hnd} in the j -th protocol step in a session with w . The command `store` inputs arbitrary application data into the cryptographic library, here the user identity u . The command `list` forms a list, and the final command `send_i` means that M_u^{Ya} sends the resulting term to v over an insecure channel. The effect is that the adversary obtains a handle to the term and can decide what to do with it (such as forwarding it to M_v^{Ya} or performing Dolev-Yao-style algebraic operations on the term). The superscript ^{hnd} on most parameters denotes that these are handles, i.e., the users obtain local names for the corresponding terms. This is an important aspect of [9] because it allows the same protocol description to be implemented once with Dolev-Yao-style idealized cryptography and once with real cryptography. The four commands we saw so far and their input and output domains belong to the interface (in the same sense as, e.g., a Java interface) of the underlying cryptographic library. This interface is implemented by both the idealized and the real version. In the first case, the handles are local names of Dolev-Yao-style terms, in the second case of real cryptographic bitstrings, on which the adversary can perform arbitrary bit manipulations. We say more about these two implementations below.

¹ The set of users is $\{1, \dots, n\}$ and the Yahalom protocol is designed such that $T \notin \{1, \dots, n\}$ where T denotes the trusted party.

Algorithm 2 Behavior of the Trusted Party M_T^{Ya}

Input: $(v, T, i, m^{\text{hnd}})$ at $\text{out}_T?$ with $v \in \{1, \dots, n\}$.

- 1: $t_i^{\text{hnd}} \leftarrow \text{list_proj}(m^{\text{hnd}}, i)$ for $i = 1, 2, 3$.
- 2: $t_1 \leftarrow \text{retrieve}(t_1^{\text{hnd}})$. $\{t_1 \approx v\}$
- 3: $\text{type}_{t_2^{\text{hnd}}} \leftarrow \text{get_type}(t_2^{\text{hnd}})$.
- 4: $l^{\text{hnd}} \leftarrow \text{sym_decrypt}(skse_{T,v}^{\text{hnd}}, t_3^{\text{hnd}})$. $\{l^{\text{hnd}} \approx (u, N_u)\}$
- 5: $x_i^{\text{hnd}} \leftarrow \text{list_proj}(l^{\text{hnd}}, i)$ for $i = 1, 2$.
- 6: $x_1 \leftarrow \text{retrieve}(x_1^{\text{hnd}})$. $\{x_1 \approx u\}$
- 7: $\text{type}_{x_2^{\text{hnd}}} \leftarrow \text{get_type}(x_2^{\text{hnd}})$.
- 8: **if** $\text{type}_{t_2^{\text{hnd}}} \neq \text{nonce} \vee \text{type}_{x_2^{\text{hnd}}} \neq \text{nonce} \vee t_1 \neq v \vee x_1 \notin \{1, \dots, n\} \setminus \{v\}$ **then Abort** **end if**
- 9: $skse^{\text{hnd}} \leftarrow \text{gen_symenc_key}()$. $\{skse^{\text{hnd}} \approx K_{uv}\}$
- 10: $l_3^{(1)\text{hnd}} \leftarrow \text{list}(t_1^{\text{hnd}}, skse^{\text{hnd}}, x_2^{\text{hnd}})$. $\{l_3^{(1)\text{hnd}} \approx (v, K_{uv}, N_u)\}$
- 11: $c_3^{(1)\text{hnd}} \leftarrow \text{sym_encrypt}(skse_{T,x_1}^{\text{hnd}}, l_3^{(1)\text{hnd}})$. $\{c_3^{(1)\text{hnd}} \approx (v, K_{uv}, N_u)_{K_{ut}}\}$
- 12: $l_3^{(2)\text{hnd}} \leftarrow \text{list}(x_1^{\text{hnd}}, t_1^{\text{hnd}}, skse^{\text{hnd}}, t_2^{\text{hnd}})$. $\{l_3^{(2)\text{hnd}} \approx (u, v, K_{uv}, N_v)\}$
- 13: $c_3^{(2)\text{hnd}} \leftarrow \text{sym_encrypt}(skse_{T,v}^{\text{hnd}}, l_3^{(2)\text{hnd}})$. $\{c_3^{(2)\text{hnd}} \approx (u, v, K_{uv}, N_v)_{K_{vt}}\}$
- 14: $m_3^{\text{hnd}} \leftarrow \text{list}(t_2^{\text{hnd}}, c_3^{(1)\text{hnd}}, c_3^{(2)\text{hnd}})$. $\{m_3^{\text{hnd}} \approx (N_v, (v, K_{uv}, N_u)_{K_{ut}}, (u, v, K_{uv}, N_v)_{K_{vt}})\}$
- 15: **send_i** (x_1, m_3^{hnd}) .

The treatment of network inputs by protocol machines and by the trusted third party is defined similar to Algorithm 1. Network inputs of user u arrive at port $\text{out}_u?$ and are of the form $(v, u, i, m^{\text{hnd}})$ where v is the supposed sender, i denotes that the channel is insecure, and m^{hnd} is a handle to a list. The port $\text{out}_u?$ is connected to the cryptographic library, whose two implementations represent the obtained Dolev-Yao-style term or real bitstring, respectively, to the protocol in a unified way by the handle m^{hnd} . Due to space constraints, we omit an informal description of how these inputs are processed; this should already be clear from the preceding protocol description. Moreover, we only give the algorithmic description how the trusted third party reacts on network inputs in Algorithm 2 and postpone the algorithmic description how the protocol machines react on network inputs to [7].

We furthermore use the convention that every machine should immediately abort the handling of the current input if a cryptographic command does not yield the desired result, e.g., if a decryption fails.

2.3 Overall Framework and Adversary Model

The framework that determines how machines such as our Yahalom machines and the machines of the idealized or real cryptographic library execute is taken from [33]. The basis is an asynchronous probabilistic execution model with distributed scheduling and with a well-defined Turing-machine refinement for complexity considerations. We already used implicitly above that for term construction and parsing commands to the cryptographic library, so-called local scheduling is defined, i.e., a result is returned immediately. The idealized or real network sending via this library, however, is scheduled by the adversary.

When protocol machines such as M_u^{Ya} are defined there is no guarantee that all these machines are correct. A trust model determines for what subsets \mathcal{H} of $\{1, \dots, n, T\}$ we want to guarantee anything; here these are the subsets that contain at least the trusted party: We prove secrecy of keys shared by u and v whenever $u, v \in \mathcal{H}$ and thus whenever M_u^{Ya} and M_v^{Ya} are correct. Incorrect machines disappear and are replaced by the adversary. Each set of potential correct machines together with its user interface is called a structure, and the set of these structures is called the system. When considering the security of a structure, an arbitrary probabilistic machine H is connected to the user interface to represent all users, and an arbitrary probabilistic 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. This setting implies that any number of concurrent protocol runs with the honest participants and the adversary are considered because H and A can arbitrarily interleave protocol start inputs (`new_prot`, `Yahalom`, v) with the delivery of network messages.

For a set \mathcal{H} of honest participants, the user interface of the Yahalom protocol machines is $S_{\mathcal{H}}^{KE} := \{KE_in_u?, KE_out_u! \mid u \in \mathcal{H} \setminus \{T\}\}$. The ideal and real Yahalom protocol serving this interface differ only in the cryptographic library, i.e., the Yahalom machines either rely on a set $\hat{M}_{\mathcal{H}}^{cry} := \{M_{u,\mathcal{H}}^{cry} \mid u \in \mathcal{H}\}$ of real cryptographic machines or an ideal machine $TH_{\mathcal{H}}^{cry}$ called *trusted host*. With $\hat{M}_{\mathcal{H}}^{Ya} := \{M_u^{Ya} \mid u \in \mathcal{H}\}$, the ideal system is $Sys^{Ya,id} := \{(\hat{M}_{\mathcal{H}}^{Ya} \cup \{TH_{\mathcal{H}}^{cry}\}, S_{\mathcal{H}}^{KE}) \mid \{T\} \subseteq \mathcal{H} \subseteq \{1, \dots, n, T\}\}$, and the real system is $Sys_{SE}^{Ya,real} := \{(\hat{M}_{\mathcal{H}}^{Ya} \cup \hat{M}_{\mathcal{H}}^{cry}, S_{\mathcal{H}}^{KE}) \mid \{T\} \subseteq \mathcal{H} \subseteq \{1, \dots, n, T\}\}$, where SE denotes the symmetric encryption scheme used.

3 The Key Secrecy Property

In the following, we formalize the key secrecy property of the ideal and real Yahalom protocols. The property is an instantiation of a general symbolic key secrecy definition for arbitrary protocols based on the ideal cryptographic library from [8], which is based on the typical notion that a term is not an element of the adversary's knowledge set. In the given Dolev-Yao-style library, the adversary's knowledge set is the set of all terms to which the adversary has a handle.

3.1 Overview and States of the Ideal Cryptographic Library

The ideal cryptographic library administrates Dolev-Yao-style terms and allows each user to operate on them via handles, i.e., via local names specific to this user. The handles also contain the information that knowledge sets give in other Dolev-Yao formalizations: The set of terms that a participant u knows, including $u = a$ for the adversary, is the set of terms with a handle for u . The terms are typed; for instance, decryption only succeeds on ciphertexts and projection only on lists. Moreover, the terms are globally numbered by a so-called index. Each term is represented by its type (i.e., root node) and its first-level arguments, which can be indices of earlier terms. This enables easy distinction of, e.g., which of many nonces is encrypted in a larger term. These global indices are never visible at the user interface. The indices and the handles for each participant are generated by one counter each.

The data structure storing the terms in [9] is a database D . Generally, 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 attributes fulfill $pred$. If $D[pred]$ contains only one element, we use the same notation for this element. Adding an entry x to D is abbreviated $D := x$. Moreover, we write the list operation as $l := (x_1, \dots, x_j)$, and argument retrieval as $l[i]$ with $l[i] = \downarrow$ if $i > j$.

In the specific term database D , each entry x can have the following arguments (with domains omitted for brevity): $x.ind$ is the global index of an entry, which is used as a primary key attribute of the database, i.e., we write $D[i]$ for the selection $D[ind = i]$. $x.type$ denotes the *type* of x and $x.arg = (a_1, a_2, \dots, a_j)$ is a possibly empty list of arguments. $x.hnd_u$ for $u \in \mathcal{H} \cup \{\mathbf{a}\}$ are handles, where $x.hnd_u = \downarrow$ means that u does not know this entry. Finally, $x.len$ denotes the length of the entry. The machine $\text{TH}_{\mathcal{H}}$ has a counter $size \in \mathcal{INDS}$ for the current size of D and counters $curhnd_u$ (current handle) for the handles, all initialized with 0.

In order to capture that keys shared between users and the trusted party have already been generated and distributed, we assume that suitable entries for the keys already exist in the database. We denote the handle of u to the secret key shared with v , where either $u \in \{1, \dots, n\}$ and $v = \mathbf{T}$ or vice versa, by $skse_{u,v}^{hnd}$.

3.2 The Real Cryptographic Library

In the real cryptographic library, each user has its own machine. 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 the same commands as with the ideal library, e.g., en- or decrypt a message etc. These commands now trigger real cryptographic operations. The operations 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.

3.3 Definition of the Key Secrecy Property

The first step towards defining symbolic key secrecy is to consider one state of the ideal Dolev-Yao-style library and to 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 the operations that the Yahalom protocol performs on new keys are allowed in this sense. For Condition (3) in the definition, note that the arguments of a ciphertext term are (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 1. (*Symbolically Secret Encryption Keys [8]*) Let $\{\mathbf{T}\} \subseteq \mathcal{H} \subseteq \{1, \dots, n, \mathbf{T}\}$, a database state D of $\text{TH}_{\mathcal{H}}^{cry}$, and a pair $(u, l^{hnd}) \in \mathcal{H} \times \mathcal{INDS}$ of a user and a handle be given. Let $i := D[hnd_u = l^{hnd}].ind$ be the corresponding database index. The *term under* (u, l^{hnd}) (1) is a *symmetric encryption key* iff

$D[i].type = \text{skse}$, (2) is symbolically unknown (to the adversary) iff $D[i].hnd_a = \downarrow$, (3) has not been used for encryption, or short is unused, iff for all indices $j \in \mathbb{N}$ we have $D[j].type = \text{symenc} \Rightarrow D[j].arg[2] \neq i - 1$, and (4) is a symbolically secret key iff it has the three previous properties. \diamond

A secret-key belief function is a general way to designate the keys whose secrecy should be proved. The underlying theory from [8] is based on such functions. We instantiate them for the Yahalom protocol and thus essentially for all individual key exchange protocols. A secret key belief function 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. For the Yahalom protocol, we define secret-key belief functions $\text{seckey_initiator_Ya}$ for the initiator and $\text{seckey_responder_Ya}$ for the responder that designate the exchanged keys.

Definition 2. (*Secret-key Belief Functions for the Yahalom Protocol*) A secret-key belief function for a set \mathcal{H} is a function seckey that maps each view $view$ of the user to an element of $(\mathcal{H} \times \mathcal{HND} \times \{\text{skse}\})^*$.

The secret-key belief functions $\text{seckey_initiator_Ya}$ and $\text{seckey_responder_Ya}$ of the Yahalom protocol map each element $(\text{ok_initiator}, \text{Yahalom}, v, \text{skse}^{\text{hnd}})$ respectively $(\text{ok_responder}, \text{Yahalom}, v, \text{skse}^{\text{hnd}})$ of $view$ arriving at port $\text{KE_out}_u?$ in the users view to $(u, \text{skse}^{\text{hnd}}, \text{skse})$ if $u \in \mathcal{H}$, and to ϵ otherwise. Elements of $view$ that are not of this form are also mapped to ϵ . \diamond

We now 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 3. (*Symbolic Key Secrecy Generally and for the Yahalom Protocol*) Let a user H^* suitable for a structure $(\{\text{TH}_{\mathcal{H}}^{\text{cry}}\}, S_{\mathcal{H}}^{\text{cry}})$ of the cryptographic library $Sys^{\text{cry}, \text{id}}$ and a secret-key belief function seckey for \mathcal{H} be given. The ideal cryptographic library with this user keeps the keys in seckey strictly symbolically secret iff for all configurations $conf = (\{\text{TH}_{\mathcal{H}}^{\text{cry}}\}, S_{\mathcal{H}}^{\text{cry}}, H, A)$ of this structure, every $v \in view_{conf}(H)$, and every element $(u_i, l_i^{\text{hnd}}, t_i)$ of the set $\text{seckey}(v)$, the term under (u_i, l_i^{hnd}) is a symbolically secret key of type t_i , and $D[hnd_{u_i} = l_i^{\text{hnd}}].ind \neq D[hnd_{u_j} = l_j^{\text{hnd}}].ind$ for all $i \neq j$.

The ideal Yahalom protocol keeps the exchanged keys of honest users strictly symbolically secret iff the ideal cryptographic library keeps the keys in $\text{seckey_initiator_Ya}$ and $\text{seckey_responder_Ya}$ strictly symbolically secret with all users H^* that are the combination of the machines M_u^{Ya} for $u \in \mathcal{H}$ and a user H of those machines. \diamond

General cryptographic key secrecy requires that no polynomial-time adversary can distinguish the keys designated by the function seckey from fresh keys. The cryptographic key secrecy of the Yahalom protocol is the instantiation for $\text{seckey_initiator_Ya}$ and $\text{seckey_responder_Ya}$ and the configurations of the Yahalom protocol.

Definition 4. (*Cryptographic Key Secrecy Generally and for the Yahalom Protocol*) Let a polynomial-time configuration $conf = (\hat{M}_{\mathcal{H}}^{\text{cry}}, S_{\mathcal{H}}^{\text{cry}}, H, A)$ of the real cryptographic library $Sys_{SE}^{\text{cry}, \text{real}}$ and a secret-key belief function seckey for \mathcal{H} be given. Let gen_{SE}

denote the key generation algorithm. This configuration *keeps the keys in seckey cryptographically secret* iff for all probabilistic-polynomial time algorithms Dis (the distinguisher), we have

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

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

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

and $keys_{fresh} := (sk'_i)_{i=1, \dots, n}$ with $sk'_i \leftarrow gen_A(1^k)$ if $t_i = ska$, else $sk'_i \leftarrow \epsilon$.

A polynomial-time configuration $(\hat{M}_{\mathcal{H}}^{cry} \cup \hat{M}_{\mathcal{H}}^{Ya}, S_{\mathcal{H}}^{KE}, H, A)$ of the real Yahalom protocol $Sys^{Ya, real}$ *keeps the exchanged keys of honest users cryptographically secret* iff the configuration $(\hat{M}_{\mathcal{H}}^{cry}, S_{\mathcal{H}}^{cry}, \{H\} \cup \hat{M}_{\mathcal{H}}^{Ya}, A)$ keeps the keys in `seckey_initiator_Ya` and `seckey_responder_Ya` cryptographically secret. \diamond

Theorem 1. (*Security of the Yahalom Protocol*) The ideal Yahalom system $Sys^{Ya, id}$ from Section 2.3 keeps the exchanged keys of honest users strictly symbolically secret, and all polynomial-time configurations of the real system $Sys^{Ya, real}$ keep the exchanged keys of honest users cryptographically secret. \square

4 Proof of the Cryptographic Realization from the Idealization

As discussed in the introduction, the idea of our approach is to prove Theorem 1 for the protocol using the ideal Dolev-Yao-style cryptographic library. Then the result for the real system follows automatically. The notion that a system Sys_1 securely implements another system Sys_2 in the sense of reactive simulatability [33], is written $Sys_1 \geq_{sec}^{poly} Sys_2$ (in the computational case). The main result of [9, 10, 6] is therefore

$$Sys^{cry, real} \geq_{sec}^{poly} Sys^{cry, id}. \quad (1)$$

If symmetric encryption is present, this result is additionally subject to the condition that the surrounding protocol, in our case the Yahalom protocol, does not raise a so-called commitment problem for symmetric encryption. It is a nice observation that this condition can be immediately concluded from the overall proof; the formal argument is contained in the long version [7]. For technical reasons, one further has to ensure that the protocol does not create encryption cycles (such as encrypting a key with itself); this is needed even for much weaker properties than simulatability, see [3]. This property clearly holds for the Yahalom protocol.

Once we have shown that the considered keys are symbolically secret and that the commitment problem does not occur for the Yahalom protocol, we can exploit the key-secrecy preservation theorem of [8]: If for certain honest users H and a secret-key belief function `seckey` the ideal cryptographic library keeps the keys in `seckey` strictly symbolically secret, then every configuration of H with the real cryptographic library keeps the keys in `seckey` cryptographically secret.

5 Proof in the Ideal Setting

It remains to prove the ideal part of Theorem 1. The proof idea is the following: If an honest user u successfully terminates a session run with another honest user v , we first show that the established key has been created by the trusted party. Then we exploit that the trusted party and the honest users only send this key within an encryption generated with a key shared between u and T respectively v and T , and we conclude that the adversary never gets a handle to the key. The main challenge is to find suitable invariants on the state of the ideal Yahalom system $\text{Sys}^{\text{Ya, id}}$. We now present these invariants. Their proof and the proof of Theorem 1 is postponed to [7] due to space constraints.

The first invariants, *correct nonce owner* and *unique nonce use*, are easily proved. They essentially state that handles n^{hnd} where $(n^{\text{hnd}}, \cdot, \cdot)$ is contained in a set Nonce_u point to entries of type nonce and that no nonce is in two such sets.

Invariant 1 (*Correct Nonce Owner*) For all $u \in \mathcal{H} \setminus \{\mathsf{T}\}$, $v \in \{1, \dots, n\}$, $j \in \{1, 2, 3, 4\}$ and $(n^{\text{hnd}}, v, j) \in \text{Nonce}_u$, we have $D[\text{hnd}_u = n^{\text{hnd}}] \neq \downarrow$ and $D[\text{hnd}_u = n^{\text{hnd}}].\text{type} = \text{nonce}$.

Invariant 2 (*Unique Nonce Use*) For all $u, v \in \mathcal{H} \setminus \{\mathsf{T}\}$, all $w, w' \in \{1, \dots, n\}$, all $j, j' \in \{1, 2, 3, 4\}$, and all $i \leq \text{size}$: If $(D[i].\text{hnd}_u, w, j) \in \text{Nonce}_u$ and $(D[i].\text{hnd}_v, w', j') \in \text{Nonce}_v$, then $(u, w) = (v, w')$.

The invariant *encrypted-key secrecy* states that a key shared between honest u and v as well as all lists containing this key can only be known to u , v , and T . Moreover, such lists only occur within “suitable” symmetric encryptions.

Invariant 3 (*Encrypted-Key Secrecy*) For all $u, v \in \mathcal{H} \setminus \{\mathsf{T}\}$ and all $i \leq \text{size}$ with $D[i].\text{type} = \text{symenc}$: Let $l^{\text{ind}} := D[i].\text{arg}[1]$, $\text{pkse}^{\text{ind}} := D[i].\text{arg}[2]$, $x_t^{\text{ind}} := D[l^{\text{ind}}].\text{arg}[t]$, and $x_t := D[x_t^{\text{ind}}].\text{arg}[1]$ for $t = 1, 2, 3$. If $D[l^{\text{ind}}].\text{type} = \text{list} \wedge \text{pkse}^{\text{ind}} = \text{pkse}_u \wedge x_1 = v \wedge D[x_t^{\text{ind}}].\text{type} = \text{skse}$ for some $t \in \{1, 2, 3\}$ then

- a) $D[x_t^{\text{ind}}].\text{hnd}_w = \downarrow$ and $D[l'^{\text{ind}}].\text{hnd}_w = \downarrow$ for $(\mathcal{H} \setminus \{u, v, \mathsf{T}\}) \cup \{a\}$ and for all l'^{ind} with $x_t^{\text{ind}} \in D[l'^{\text{ind}}].\text{arg}$.
- b) For all $l', k \leq \text{size}$ such that $D[l'].type = \text{list} \wedge x_t^{\text{ind}} \in D[l'].arg$, we have that $l' \in D[k]$ only if $D[k].type = \text{symenc}$ and $D[k].arg[2] \in \{\text{pkse}_u, \text{pkse}_v\}$.

The invariant *correct encryption owner* finally states that certain protocol messages can only be constructed by the “intended” users or by the trusted party, respectively.

Invariant 4 (*Correct Encryption Owner*) For all $u \in \mathcal{H} \setminus \{\mathsf{T}\}$ and all $i \leq \text{size}$ with $D[i].type = \text{symenc}$: Let $l_k^{\text{ind}} := D[i].arg[2k - 1]$ and $\text{pkse}_k^{\text{ind}} := D[i].arg[2k]$ for $1 \leq k \leq \frac{|D[i].arg|}{2}$ (entries of type symenc have an even number of arguments by construction). Let further $x_{k,t}^{\text{ind}} := D[l_k^{\text{ind}}].arg[t]$ and $x_{k,t,u}^{\text{hnd}} := D[x_{k,t}^{\text{ind}}].\text{hnd}_u$ for $t = 1, 2, 3, 4$, and $x_{k,t} := D[x_{k,t}^{\text{ind}}].arg[1]$ for $t = 1, 2$.

- a) If $\text{pkse}_k^{\text{ind}} = \text{pkse}_u$, $x_{k,1} \in \mathcal{H}$, $D[x_{k,2}^{\text{ind}}].type = \text{skse}$, and $(x_{k,3,u}^{\text{hnd}}, x_{k,1}, j) \in \text{Nonce}_u$ for some $j \in \{1, 3\}$ and some $k \in \{1, \dots, \frac{|D[i].arg|}{2}\}$, then $D[i]$ was created by $M_{\mathsf{T}}^{\text{Ya}}$ in Step 11 of Algorithm 2.

- b) If $pkse_k^{\text{ind}} = pkse_u$, $x_{k,1} \in \mathcal{H}$, $x_{k,2} = u$, $D[x_{k,3}^{\text{ind}}].type = skse$, and $(x_{k,4,u}^{\text{hd}}, x_{k,1}, j) \in \text{Nonce}_u$ for some $j \in \{2, 4\}$ and some $k \in \{1, \dots, \frac{|D[i].arg|}{2}\}$, then $D[i]$ was created by M_1^{Ya} in Step 13 of Algorithm 2.

6 Conclusion

We have analyzed the key secrecy property of the strengthened Yahalom protocol. After showing that the protocol does not guarantee cryptographic key secrecy in the sense of computational indistinguishability of the exchanged key from a random one, we have proven cryptographic key secrecy of a slightly simplified version of the protocol via a deterministic, provably secure abstraction of a real cryptographic library. Together with composition and preservation theorems from the underlying model, this library allowed us to perform the actual proof effort in a deterministic setting corresponding to a slightly extended Dolev-Yao model. Besides establishing the cryptographic security of the strengthened Yahalom protocol, our result also serves as an exemplification of the potential of the cryptographic library and the recent secrecy preservation theorem for symbolic, cryptographically sound proofs of security protocols.

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th ACM CCS*, pages 36–47, 1997.
2. M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. 4th 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 TCS*, volume 1872 of *LNCS*, pages 3–22. Springer, 2000.
4. M. Backes. A cryptographically sound Dolev-Yao style security proof of the Otway-Rees protocol. In *Proc. 9th ESORICS*, volume 3193 of *LNCS*, pages 89–108. Springer, 2004.
5. M. Backes and B. Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. In *Proc. 23rd FSTTCS*, pages 1–12, 2003.
6. M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE CSFW*, pages 204–218, 2004.
7. M. Backes and B. Pfitzmann. Cryptographic key secrecy of the strengthened Yahalom protocol via a symbolic security proof. Research Report 3601, IBM Research, 2005. <http://domino.research.ibm.com/library/cyberdig.nsf/index.html>.
8. M. Backes and B. Pfitzmann. Relating symbolic and cryptographic key secrecy. In *Proc. 26th IEEE S & P*, 2005. Extended version accepted for *IEEE Transactions on Dependable and Secure Computing*.
9. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proc. 10th ACM CCS*, pages 220–230, 2003. Full version in IACR ePrint Archive 2003/015, Jan. 2003.
10. M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. In *Proc. 8th ESORICS*, volume 2808 of *LNCS*, pages 271–290. Springer, 2003.
11. D. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 2004.

12. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Proc. ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, 2000.
13. M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient constructions. In *Proc. ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer, 2000.
14. M. Burrows, M. Abadi, and R. Needham. A logic for authentication. Technical Report 39, SRC DIGITAL, 1990.
15. R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols (the case of encryption-based mutual authentication and key exchange). Cryptology ePrint Archive, Report 2004/334, 2004.
16. V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th ESOP*, pages 157–171, 2005.
17. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
18. V. D. Gligor and P. Donescu. Fast encryption and authentication: Xcbc encryption and xecb authentication modes. In *Proc. 8th FSE*, pages 82–108, 2001.
19. J. Guttman. Key compromise and the authentication tests. In *Proc. MPFS*, volume 17 of ENTCS, pages 1–21, 2001.
20. J. D. Guttman, F. J. Thayer Fabrega, and L. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In *Proc. 8th ACM CCS*, pages 186–195, 2001.
21. J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In *Advances in Cryptology: CRYPTO 2003*, volume 2729 of *LNCS*, pages 548–564. Springer, 2003.
22. R. Impagliazzo and B. M. Kapron. Logics for reasoning about cryptographic constructions. In *Proc. 44th FOCS*, pages 372–381, 2003.
23. C. Jutla. Encryption modes with almost free message integrity. In *Advances in Cryptology: EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 529–544. Springer, 2001.
24. R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.
25. P. Laud. Semantics and program analysis of computationally secure information flow. In *Proc. 10th ESOP*, pages 77–91, 2001.
26. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. 25th IEEE S & P*, pages 71–85, 2004.
27. P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM CCS*, pages 112–121, 1998.
28. G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–135, 1995.
29. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. 2nd TACAS*, volume 1055 of *LNCS*, pages 147–166. Springer, 1996.
30. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st TCC*, volume 2951 of *LNCS*, pages 133–151. Springer, 2004.
31. L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Cryptology*, 6(1):85–128, 1998.
32. L. Paulson. Relations between secrets: Two formal analyses of the yahalom protocol. *Journal of Computer Security*, 9(3):197–216, 2001.
33. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE S & P*, pages 184–200, 2001. Extended version of the model (with Michael Backes) IACR Cryptology ePrint Archive 2004/082.