ISSN 0361-7688, Programming and Computer Software, 2022, Vol. 48, No. 8, pp. 781–787. © Pleiades Publishing, Ltd., 2022.

Optimization of ProVerif Programs for AKE Protocols

E. M. Vinarskii[a,b,*] (ORCID: 0000-0002-7328-0942)
and A. V. Demakov[a,] (ORCID: 0000-0001-6573-7925)**

[a] *Ivannikov Institute for System Programming, Russian Academy of Sciences, ul. Solzhenitsyna 25, Moscow, 109004 Russia*

[b] *National Research University Higher School of Economics, Pokrovskii bul'v. 11, Moscow, 109028 Russia*

**e-mail: vinarskii@ispras.ru*

***e-mail: demakov@ispras.ru*

Received January 12, 2022; revised February 16, 2022; accepted March 22, 2022

Abstract—Cryptographic protocols are used to establish secure connection between agents who communicate strictly in accordance with the rules specified by the protocol. To make sure that a newly designed cryptographic protocol is cryptographically strong, various software tools are usually employed. However, an adequate specification of a cryptographic protocol is generally represented as a set of requirements for sequences of transmitted messages, including their format. The fulfillment of all these requirements leads to the fact that the formal specification of a real-world cryptographic protocol becomes cumbersome and, therefore, difficult to analyze by formal methods. ProVerif is one of the intensively developed tools for formal verification of cryptographic protocols. However, ProVerif often fails to analyze large protocols, i.e., it can neither prove the security of the protocol nor refute it. In these cases, either the problem is approximated or equivalent transformations of a program model in the ProVerif language are carried out to simplify the ProVerif model. In this paper, we propose a technique to simplify ProVerif specifications for AKE protocols that use the ElGamal encryption scheme. In particular, we define equivalent transformations that make it possible to construct a ProVerif specification that facilitates the analysis for the ProVerif tool. Experimental results for the Needham–Schroeder and Yahalom cryptoprotocols show that this approach is promising for automatic verification of real-world protocols.

DOI: 10.1134/S0361768822080035

1. INTRODUCTION

Cryptographic protocols (CPs) are widely used in modern networks. They guarantee authentication of users, secrecy of session keys, etc. ProVerif is one of the tools for formal verification of cryptoprotocols, which is widely employed to check security properties of practically important protocols, e.g., TLS 1.3 and ARINC823 [1–3]. Due to a large number of requirements specified for cryptographic protocols, their adequate formal models are very cumbersome, which complicates their formal verification. However, if we use certain features of ProVerif and carry out the analysis only with respect to some predefined security properties (e.g., only with respect to secrecy of shared session keys), then the awkwardness of the protocol and, therefore, the complexity of the analysis can be reduced. For ProVerif, this simplification can be carried out using equivalent transformations (ETs), i.e., transformations that do not affect the adequacy of the model while facilitating the analysis of the cryptoprotocol.

An encryption function is one of the main cryptographic primitives. However, to construct an adequate model that describes the encryption/decryption operation, we need to supplement ProVerif with an equation that relates these two operations; examples of such equations are shown in Figs. 1 and 2. These equations can significantly complicate the ProVerif model and, therefore, the verification of security properties for this model. In this paper, we propose equivalent transformations that simplify the ProVerif model with encryption/decryption operations. We use the following steps to optimize the ProVerif representation of cryptographic protocols in order to facilitate the verification of their security properties.

Currently, the following two models are most widely employed for formal verification of cryptographic protocols.

• **Symbolic model.** Cryptographic primitives are regarded as ideal black boxes modeled by function symbols in a term algebra. All computations are carried out in a fixed theory formed by functional symbols and reduction rules (equations) over them. Messages are terms on these primitives. The attacker can carry out computations only in the framework of the

$M, N ::=$	terms
x, y, z	variable
a, b, c, k	name
$f(M_1,..., M_n)$	constructor application
$P, Q ::=$	processes
$\overline{M}\langle N\rangle.P$	output
$M(x).P$	input
0	nil
$P \mid Q$	parallel composition
$!P$	replication
$(\nu a)P$	restriction
let $x = g(M_1,..., M_n)$ *in* P *else* Q	destructor application
if $M = N$ *then* P *else* Q	conditional
`event`$(M).P$	event

Fig. 1. ProVerify syntax.

fixed theory. Currently, the best symbolic solvers are the Tamarin prover [4–6] and ProVerif [7–9].

• **Provable security model.** In this model, messages are bit strings, cryptographic primitives are functions of bit strings, and the attacker is any probabilistic Turing machine. The strength of a protocol is expressed in terms of the strength of its primitives. Currently, the most successful tool of this class is CryptoVerif [10].

In this study, we use ProVerif, which implements the symbolic approach to CP verification. As an input, ProVerif receives a model of a security protocol while defining the actions of agents and specifying the desired properties of the protocol. To describe agents, a process algebra is used. ProVerif automatically translates processes into a system of Horn clauses and uses an algorithm based on free choice of clauses from the available set to determine whether a certain statement holds in a given rewriting system (theory). If there is no such inference, then the security property is considered proved. If the inference exists, then the property may not be strong. There is no guarantee of its weakness because Horn clauses can be applied infinitely often, which is an approximation of the cryptoprotocol; as a result, the inference can correspond to a false attack [7, 8].

There are many papers devoted to optimization of CP model representations for software verification tools. In [5], two following approaches to facilitating the proof of lemmas for CP models in the Tamarin prover were proposed: (i) the use of source lemmas, which allow one to limit the size of the model in which solutions are sought, and (ii) the use of oracles, which suggest a path of proving/refuting a lemma. We propose to use a standard mathematical method—equivalent transformations—to simplify CP models. In [7], a method of CP representation by Horn clauses (inference rules) was investigated: the resulting inference rules were supplemented with the inference rules defined for the attacker, and the final inference rules formed a theory. The inference of a term signalizing that the attacker knows the secret is regarded as a threat. When optimizing the ProVerif model of a CP, we first simplify the rewriting system. The following steps are taken to optimize the ProVerif model.

```
fun Encrypt (bitstring, bitstring) : bitstring.
  reduc forall mess : bitstring, key : bitstring
    Decrypt (Encrypt (mess, key), key) = mess
```

Fig. 2. Standard symmetric encryption function.

1. The security properties are limited to the secrecy of a session key.

2. The concept of ProVerif models equivalent with respect to cryptoprotocol formulas is introduced.

3. Equivalent transformations of encryption functions are proposed, and it is proved that the cryptoprotocol obtained by these transformations remains equivalent to the original one in terms of the secrecy of the session key.

4. Experiments on the Needham–Schroeder and Yahalom protocols are carried out to demonstrate that the proposed ETs do facilitate the analysis of the cryptoprotocol.

The paper is organized as follows. Section 2 introduces basic definitions and demonstrates how ProVerif translates a cryptoprotocol into a system of Horn clauses. Section 3 describes the proposed equivalent transformations. Section 4 presents results of the experiments. Section 5 concludes the paper and outlines some directions for further research.

2. PROVERIF REPRESENTATION OF CPs AND HORN CLAUSES

In this section, we briefly describe the syntax of the ProVerif language. Figure 1 shows the grammar that defines the rules for constructing terms and processes in the ProVerif language; the complete description can be found in [7].

Terms in ProVerif may consist of variables, identifiers, constructors, and destructors. Constructors are used to generate terms; destructors are partial functions (not defined over all input data). For instance, destructor *let* $x = g(M_1, ..., M_n)$ *in* P *else* Q tries to compute $g(M_1, ..., M_n)$: in the case of a success, the result g is written in variable x and process P is executed; otherwise, process Q is executed. Action $(va)P$ means that process P has a random secret value a.

2.1. Representation of Encryption in ProVerif

We consider optimization of cryptoprotocols with symmetric encryption. When modeling an encryption scheme in ProVerif, the combination of constructor *Encrypt* and destructor *Decrypt* is generally used. Figures 2 and 3 show the ProVerif representation of encryption schemes.

```
fun Encrypt (bitstring, bitstring, bitstring) : bitstring.
  reduc forall mess : bitstring, key : bitstring, iv : bitstring
    Decrypt (Encrypt(mess, key, iv), key, iv) = mess
```

Fig. 3. Symmetric encryption function with an initialization vector.

The encryption scheme shown in Fig. 2 is called the *standard* scheme. For this scheme, ProVerif generates the following Horn clauses:

• $attacker(mess) \wedge attacker(key) \Rightarrow attacker(Encrypt(mess, key))$;

• $attacker(Encrypt(mess, key)) \wedge attacker(key) \Rightarrow attacker(mess)$.

The encryption scheme shown in Fig. 3 is called *encryption with an initialization vector*. This scheme generates the following Horn clauses:

• $attacker(mess) \wedge attacker(key) \wedge attacker(iv) \Rightarrow attacker(Encrypt(mess, key, iv))$;

• $attacker(Encrypt(mess, key, iv)) \wedge attacker(key) \wedge attacker(key) \Rightarrow attacker(mess)$

2.2. ElGamal Encryption Scheme

In this study, we assume that computations are carried out in group G, where g is a generating element in G [11]. Figure 4 shows ProVerif equation $(g^x)^y = (g^y)^x$ for group G. The ElGamal encryption scheme [11] allows a symmetric encryption key to be transferred to another party in encrypted form. Let us describe the ProVerif representation of the ElGamal scheme. Suppose that the cryptosystem consists of client $Clnt(x, g^x)$ and server $Serv(y, g^y)$, where x (y) is a long-term private key of the client (server), while g^x (g^y) is a long-term public key of the client (server). Suppose that k is a symmetric key generated on the client side that needs to be safely passed to the server. Then, the client generates a random $a \in_U G$, computes $s = k \cdot (g^y)^a$, and sends the following message to the server:

$$Clnt \xrightarrow{(g^a, s)} Serv.$$

The server receives s and computes $k = s \cdot (g^a)^{-y} = k \cdot g^{ya} \cdot g^{-ya} = k$. Thus, the server obtains the symmetric encryption key k sent by the client.

The ElGamal encryption scheme in the ProVerif language can be implemented using constructor *ModMult* and destructor *ModDiv*0 (see Fig. 5). This scheme is represented by the following Horn clauses, where *vCurveN* is the module of an elliptic curve:

• $attacker(key) \wedge attacker(g^y) \wedge attacker(vCurveN) \Rightarrow attacker(ModMult(key, g^y, vCurveN))$;

• $attacker(ModMult(a_0, a_1, a_2)) \wedge attacker(a_1) \wedge attacker(a_2) \Rightarrow attacker(a_0)$.

2.3. Security Properties of Cryptographic Protocols

In this subsection, we consider the secrecy property (reachability property [7]) of the key. In ProVerif, this property is defined by formula $\varphi_{sec} = attacker(key)$.

To check security property $\varphi_{sec} = attacker(key)$, ProVerif constructs a system of Horn clauses and tries to deduce term $attacker(key)$. If the attacker managed to deduce term $attacker(key)$, then this should be interpreted as "the attacker may know *key*." Otherwise, if the attacker fails to deduce $attacker(key)$, then "the attacker does NOT know *key*." This approximation is due to the fact that each Horn clause can be applied infinitely often, which can lead to false positives. However, the problem of checking the satisfiability of term $attacker(key)$ is decidable (see [7]), and the algorithm proposed in [7] proved to be effective in checking practically important protocols (see [4–6]).

Suppose that π is a cryptoprotocol; then, $\mathcal{M}_\pi$ is a model of π in the ProVerif language $\mathcal{M}_\pi$ and $P(\mathcal{M}_\pi)$ is a system of Horn clauses constructed based on model $\mathcal{M}_\pi$. By $\mathcal{N}_{pub}(\mathcal{N}_{priv})$, we denote a set of public (private) names. Thus, the system of Horn clauses constructed based on protocol π with respect to the capabilities of the attacker is $\mathcal{R}_\pi = \mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}} = P(\mathcal{M}_\pi) \cup [\{attacker(a[]) \mid a \in \mathcal{N}_{pub}\} \cup \{(Rn), (Rh), (Rl), (Rs)\}]$, where inference rules $\{(Rn),(Rh),(Rl),(Rs)\}$ are shown in Fig. 6 and term $message(x, y)$ means that message y was sent via channel x. System $\mathcal{R}_\pi$ is referred to as the inference rules generated by the model of protocol π, or simply the *theory* of π.

Cryptoprotocol π is said to *satisfy the secrecy property* $\varphi_{sec} = attacker(key)$ if and only if $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}} \models \varphi_{sec}$, i.e., event $attacker(key)$ is NOT inferable in theory $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$. Let us define theories of cryptoprotocols that are equivalent with respect to formulas.

```
equation forall a1 : bitstring, a2 : bitstring;
  Exp(Exp(g, a1), a2) = Exp(Exp(g, a2), a1).
```

Fig. 4. Equation in the ProVerif language.

```
fun ModMult (bitstring, bitstring, bitstring) : bitstring.
  reduc forall a0 : bitstring, a1 : bitstring, a2 : bitstring;
    ModDiv0(ModMult(a0, a1, a2), a1, a2) = a0.
```

Fig. 5. Equation in the ProVerif language.

attacker($b_0[x]$) (Rn)

For each public constructor f of arity n,
attacker(x_1) $\wedge \ldots \wedge$ attacker(x_n) $\Rightarrow$ attacker($f(x_1, \ldots, x_n)$) (Rf)

For each public destructor g,
for each rewrite rule $g(M_1, \ldots, M_n) \to M$ in def(g),
attacker(M_1) $\wedge \ldots \wedge$ attacker(M_n) $\Rightarrow$ attacker(M) (Rg)

message(x, y) $\wedge$ attacker(x) $\Rightarrow$ attacker(y) (Rl)

attacker(x) $\wedge \ldots \wedge$ attacker(y) $\Rightarrow$ attacker(x, y) (Rs)

Fig. 6. Horn clauses for the attacker.

Definition. Suppose that φ is a formula, while π_1 and π_2 are cryptoprotocols; then, $\pi_1 \sim_\varphi \pi_2$ for $\mathcal{R}_{\pi_1,\mathcal{N}_{pub},\mathcal{N}_{priv}} \models \varphi$ if and only if $\mathcal{R}_{\pi_2,\mathcal{N}_{pub},\mathcal{N}_{priv}} \models \varphi$.

3. EQUIVALENT TRANSFORMATIONS

In this section, we describe some equivalent transformations of the ProVerif model that make it possible to simplify the model of protocol π over public names $\mathcal{N}_{pub}$ and private names $\mathcal{N}_{priv}$ ($\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$). Our experiments (see Section 4) show that, for an optimized ProVerif model, the equivalent transformations described here facilitate the verification of the secrecy property for keys. For simplicity, we hereinafter write $\mathcal{R}_\pi$ instead of $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$.

3.1. Optimization in Encryption Scheme Representation

Let us prove that, if we check the security of a cryptoprotocol in ProVerif with respect to the secrecy properties φ_{sec} defined in Subsection 2.3, then an encryption scheme with an initialization value is equivalent to the standard encryption scheme.

Suppose that $\pi^{e_{iv}}$ is a cryptoprotocol that uses an encryption scheme with initialization vector iv; then, $\mathcal{M}_\pi^{e_{iv}}$ contains constructors/destructors shown in Fig. 2. The corresponding system of Horn clauses for cryptoprotocol π is denoted by $P^{e_{iv}}(\mathcal{M}_\pi)$; the theory of π, by $\mathcal{R}_\pi^{e_{iv}}$.

By $\mathcal{M}_\pi^{\bar{e}}$, we denote a model of cryptorotocol π in which all constructors $Encrypt_{iv}$ (Fig. 3) are replaced by constructor *Encrypt* (Fig. 2) and all destructors $Decrypt_{iv}$ are replaced by destructor *Decrypt*. The corresponding theory is denoted by $\mathcal{R}_\pi^{\bar{e}}$. Then, the following statement that theories $\mathcal{R}_\pi^{e_{iv}}$ and $\mathcal{R}_\pi^{\bar{e}}$ are equivalent with respect to formula $\varphi_{sec} = attacker(key)$ holds.

Proposition 1. Suppose that $iv \in \mathcal{N}_{pub}$; then, $\mathcal{R}_\pi^{e_{iv}} \sim_{\varphi_{sec}} \mathcal{R}_\pi^{\bar{e}}$, i.e., $\mathcal{R}_\pi^{e_{iv}} \models \varphi_{sec}$ if and only if $\mathcal{R}_\pi^{\bar{e}} \models \varphi_{sec}$.

Proof.

$\Rightarrow$ Suppose that $\mathcal{R}_\pi^{e_{iv}} \models \boldsymbol{\varphi_{sec}}$ holds, while $\mathcal{R}_\pi^{\bar{e}} \models \boldsymbol{\varphi_{sec}}$ does not. Then, term ***attacker*(*key*)** is inferable in theory $\mathcal{R}_\pi^{\bar{e}}$. Let us consider this inference $\boldsymbol{\rho = r_1, \ldots, r_n}$ (successive application of Horn clauses) in theory $\mathcal{R}_\pi^{\bar{e}}$. Inference $\boldsymbol{\rho}$ necessarily contains Horn clauses generated by constructors ***Encrypt*** and destructors ***Decrypt***, because, otherwise, similar inference $\boldsymbol{\rho}$ would exist in theory $\mathcal{R}_\pi^{e_{iv}}$.

Let us consider this inference $\boldsymbol{\rho = r_1, \ldots, r_n}$, where $\boldsymbol{r_j}$ is an application of Horn clause ***attacker*(*mess*)** $\wedge$ ***attacker*(*key*)** $\Rightarrow$ ***attacker*(*Encrypt*(*mess*, *key*))**. Relation $iv \in \mathcal{N}_{pub}$ implies that the attacker knows term ***attacker*(*iv*)**; therefore, inference $\boldsymbol{\rho}$ may contain Horn clause ***attacker*(*mess*)** $\wedge$ ***attacker*(*key*)** $\wedge$ ***attacker*(*iv*)** $\Rightarrow$ ***attacker*(*Encrypt*(*mess*, *key*, *iv*))**. Similarly, if inference $\boldsymbol{\rho}$ contains an application of Horn clause ***attacker*(*Encrypt*(*mess*, *key*))** $\wedge$ ***attacker*(*key*)** $\Rightarrow$ ***attacker*(*mess*)**, then Horn clause ***attacker*(*Encrypt*(*mess*, *key*, *iv*))** $\wedge$ ***attacker*(*iv*)** $\wedge$ ***attacker*(*key*)** $\Rightarrow$ ***attacker*(*mess*)** can also be applied.

Thus, inference $\bar{\boldsymbol{\rho}}$ also exists in theory $\mathcal{R}_\pi^{e_{iv}}$, i.e., $\mathcal{R}_\pi^{e_{iv}} \models \varphi_{sec}$ does not hold, which is a contradiction.

$\Leftarrow$ Suppose now that $\mathcal{R}_\pi^{\bar{e}} \models \varphi_{sec}$; then, the fact that $\mathcal{R}_\pi^{e_{iv}} \models \varphi_{sec}$ is proved in a similar way.

Similar statements can be proved for other encryption schemes. Hence, in the following discussion, we

```
processC(skC, pkC) =
...
new k1 : bitstring :
new a1 : bitstring :
let g_pkS_al = Exp(pkS, a1) in
let cms_kl = ModMult(k1, g_a1) in
let g_al = Exp(vBasePoint, a1) in
let enc_messl = Encrypt(mess1, k1) in
out(c, (g_a1, cms_k1, enc_mess1));
...
new k2 : bitstring :
new a2 : bitstring :
let g_pkS_a2 = Exp(pkS, a2) in
let cms_k2 = ModMult(k2, g_a2) in
let g_a2 = Exp(vBasePoint, a2) in
let enc_mess2 = Encrypt(mess2, k2) in
out(c, (g_a2, cms_k2, enc_mess2));
...

processS(skS, pkS) =
...
in(c, (g_a1 : bitstring, cms_k1 : bitstring, enc_mess1 : bitstring));
let k1 = ModDiv0(cms_k1, Exp(g_a1, skS)) in
let mess1 = Decrypt(enc_mess1, k1) in
...
in(c, (g_a2 : bitstring, cms_k2 : bitstring, enc_mess2 : bitstring));
let k2 = ModDiv0(cms_k2, Exp(g_a2, skS)) in
let mess2 = Decrypt(enc_mess2, k2) in
...
```

Fig. 7. Scheme of the protocol with two symmetric session keys.

```
processC(skC, pkC) =
...
new k : bitstring :
new a : bitstring :
let g_pkS_a = Exp(pkS, a) in
let cms_k = ModMult(k, g_a) in
let g_a = Exp(vBasePoint, a) in
let enc_mess = Encrypt(mess1, k) in
out(c, (g_a, cms_k, enc_mess1));
...
new k2 : bitstring :
new a2 : bitstring :
let g_pkS_a2 = Exp(pkS, a2) in
let cms_k2 = ModMult(k2, g_a2) in
let enc_mess2 = Encrypt(mess2, k) in
let g_a2 = Exp(vBasePoint, a2) in
out(c, enc__mess2);
...

processS(skS, pkS) =
...
in(c, (g_a : bitstring, cms_k : bitstring, enc_mess : bitstring));
let k = ModDiv0(cms_k, Exp(g_a, skS)) in
let mess1 = Decrypt(enc_mess1, k) in
...
in(c, enc_mess2 : bitstring);
let mess2 = Decrypt(enc_mess2, k) in
...
```

Fig. 8. Scheme of the protocol with one symmetric session key.

confine ourselves to the theories that use only standard encryption functions.

3.2. Using One Encryption Key for the ElGamal Encryption Scheme

In this section, we propose equivalent transformations that make it possible to simplify models of cryptoprotocols that use symmetric encryption based on the ElGamal scheme. Let one side of cryptoprotocol π sends the other side encrypted messages $mess_1$ on symmetric key k_1 and encrypted messages $mess_2$ on symmetric key k_2. We consider the following secrecy property of cryptoprotocol π: $\varphi_{sec_1} = attacker(k_1)$.

Let us show that the scheme with two symmetric session keys k_1, k_2 shown in Fig. 7 is equivalent to the scheme with one symmetric session key k_1 shown in Fig. 8. The theory constructed on $\mathcal{M}^{k_1,k_2}(\pi)$, is denoted by $\mathcal{R}_\pi^{k_1,k_2}$; the theory constructed on $\mathcal{M}^{k_1}(\pi)$, by $\mathcal{R}_\pi^{k_1}$. It is required to prove that theories $\mathcal{R}_\pi^{k_1,k_2}$ and $\mathcal{R}_\pi^{k}$ are equivalent with respect to formula $\varphi_{sec} = attacker(k)$.

Proposition 2. $\mathcal{R}_\pi^{\boldsymbol{k_1},\boldsymbol{k_2}} \sim_{\boldsymbol{\varphi_{sec_1}}} \mathcal{R}_\pi^{\boldsymbol{k}}$.

Proof. Note that both the theories are defined over the same namespace $\mathcal{N}_{pub} \cup \mathcal{N}_{priv}$.

Suppose that $\mathcal{R}_\pi^{k_1} \vDash \varphi_{sec}$ holds, while $\mathcal{R}_\pi^{k_1,k_2} \vDash \varphi_{sec_1}$ does not. Then, term $attacker(k_1)$ is inferable in theory $\mathcal{R}_\pi^{k_1,k_2}$. Let us consider this inference $\rho = r_1, ..., r_n$ (successive application of Horn clauses) in theory $\mathcal{R}_\pi^{k}$. The Horn clauses generated by theories $\mathcal{R}^{k}(\pi)$ and $\mathcal{R}^{k_1,k_2}(\pi)$ differ in that theory $\mathcal{R}^{k_1,k_2}(\pi)$ contains terms $attacker(Exp(g, a_2))$, $attacker(ModMult(k_2, Exp(g, a_2)))$, and $attacker(Encrypt(mess2, k_2))$, which are absent in theory $\mathcal{R}^{k_1}(\pi)$.

Thus, since inference $\boldsymbol{\rho}$ exists in theory $\mathcal{R}^{\boldsymbol{k_1},\boldsymbol{k_2}}(\pi)$ but does not exist in theory $\mathcal{R}^{\boldsymbol{k_1}}(\boldsymbol{\pi})$, $\boldsymbol{\rho}$ contains terms $\boldsymbol{attacker(Exp(g, a_2))}$, $\boldsymbol{attacker(ModMult(k_2, Exp(g, a_2)))}$, and $\boldsymbol{attacker(Encrypt(mess_2, k_2))}$, because otherwise this inference would also exist in theory $\mathcal{R}^{\boldsymbol{k_1}}(\pi)$.

In this case, terms $\boldsymbol{attacker(Exp(g, a_2))}$ and $\boldsymbol{attacker(ModMult(k_2, Exp(g, a_2)))}$ can be inferred by the attacker based on inference rule **(*Rh*)** from Fig. 6. Similarly, if inference $\boldsymbol{\rho}$ contains term $\boldsymbol{attacker(k_1)}$, then there is term $\boldsymbol{attacker(Encrypt(mess_2, k_2))}$ and there is inference $\overline{\boldsymbol{\rho}} = \boldsymbol{\rho[k_1/k, k_2/k]}$. Therefore, $\mathcal{R}^{\boldsymbol{k_1}}(\pi) \vDash \boldsymbol{attacker(k_1)}$ does not hold, which is a contradiction.

Now, let us consider another simplification of the ElGamal encryption scheme. Suppose that the client uses symmetric encryption key $\boldsymbol{k_C}$ passed to the server by using the ElGamal scheme, while the server uses symmetric encryption key $\boldsymbol{k_S}$ passed to the client by using the same scheme. We propose equivalent transformations that make it possible to obtain a model that uses only one symmetric encryption key. Let us denote

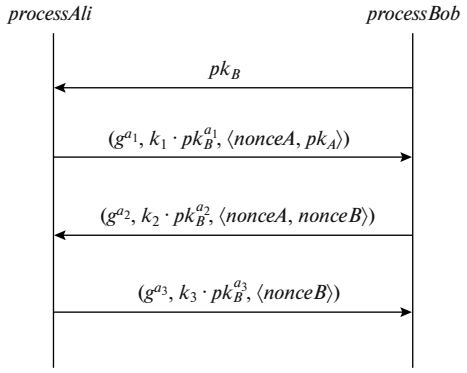

Fig. 9. Scheme of the Needham–Schroeder protocol with three symmetric session keys.

a theory constructed on $\mathcal{M}^{k_C,k_S}(\pi)$ by $\mathcal{R}_\pi^{k_C,k_S}$ and a theory constructed on $\mathcal{M}^{k_C}(\pi)$ by $\mathcal{R}_\pi^{k_C}$. Then, the following statement holds, which is proved in the same way as Proposition 2.

Proposition 3. $\mathcal{R}_\pi^{k_C,k_S} \sim_{\varphi_{secC}} \mathcal{R}_\pi^{k_C}$.

Thus, we have shown that checking the secrecy property of a session key in a protocol with the ElGamal encryption scheme that uses several session keys can be reduced to checking the secrecy property of the session key in a protocol that uses only one session key. We have also shown that encryption schemes with initialization vectors are equivalent to the standard encryption scheme for the ProVerif model. In Section 4, we demonstrate how these results can be used to optimize ProVerif code.

4. EXPERIMENTAL RESULTS

This section describes our experiments with the Needham–Schroeder and Yahalom protocols, which confirm the effectiveness of the proposed equivalent transformations for ProVerif code of cryptoprotocols.

We used a modification of the Needham–Schroeder protocol shown in Fig. 9. The original scheme uses three session keys. We checked the secrecy property of a session key, $\varphi_{sec} = attacker(key)$. This scheme was modified as follows. First, Theorem 1 was used for reduction to the standard encryption function; then, Theorems 2 and 3 were applied to obtain equivalent ProVerif models with fewer inference rules. Next, we ran ProVerif to check three versions of the models: with three session keys, with two session keys, and with one session key. All source codes of the experiments are available online [12]. As a metric, we used the number of rules used by ProVerif to prove the secrecy formula of the session key, $\varphi_{sec} = attacker(key)$.

As a result, ProVerif used 45800 rules for three session keys, 3000 rules for two session keys, and 200 rules for one session key. Thus, we can see a significant gain when using the optimizations proposed in this paper.

We conducted similar experiments with the ProVerif model of the Yahalom protocol (the scheme is shown in Fig. 10). We checked the secrecy property of a session key, $\varphi_{sec} = attacker(key)$. All source codes of the experiments are available online [12]. We modified this scheme in accordance with Propositions 2 and 3 to obtain the ElGamal encryption scheme with one session key. Overall, ProVerif used 2600 rules for two session keys and 200 rules for one session key. Thus, again, we can see a significant gain from the optimizations proposed in this paper when proving the secrecy property of the session key by using ProVerif.

5. CONCLUSIONS

In this paper, we have described some optimization techniques for ProVerif models of cryptographic protocols. We have proposed several transformations of the ProVerif model and proved that they are equivalent with respect to the secrecy formula of the session key. The effectiveness of these transformations has been tested on implementations of the Needham–Schroeder and Yahalom protocols.

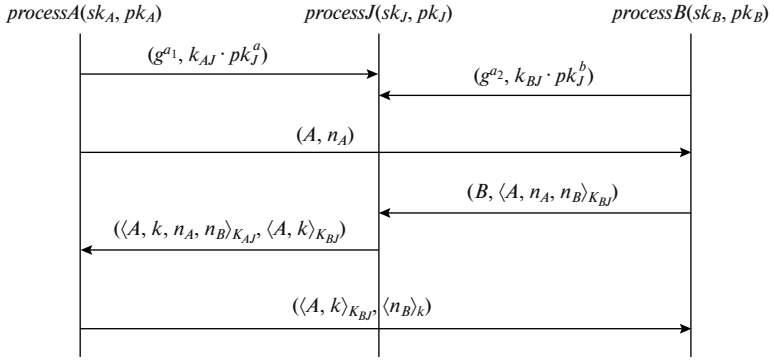

Fig. 10. Scheme of the Yahalom protocol with two symmetric session keys.

In our future works, we intend to develop various optimization techniques for other ProVerif constructs and experiment on practically important protocols.

6. FUNDING

This work was supported by the Ministry of Education and Science of the Russian Federation, grant no. 075-15-2020-788.

REFERENCES

1. Blanchet, B., Symbolic and computational mechanized verification of the ARINC823 avionic protocols, *Proc. 30th IEEE Computer Security Foundations Symp. (CSF),* 2017, pp. 68–82.
2. Bhargavan, K., Blanchet, B., and Kobeissi, N., Verified models and reference implementations for the TLS 1.3 standard candidate, Research Report RR-9040, Inria, 2017.
3. Bhargavan, K., Blanchet, B., and Kobeissi, N., Verified models and reference implementations for the TLS 1.3 standard candidate, *Proc. IEEE Symp. Security and Privacy (S&P),* pp. 483–503.
4. Meier, S., Schmidt, B., et al., The TAMARIN prover for the symbolic analysis of security protocols, *Proc. 25th Int. Conf. Computer Aided Verification,* 2013, pp. 696–701.
5. Meier, S., Advancing automated security protocol verification, *PhD Thesis,* ETH Zurich, 2013.
6. Schmidt, B., Formal analysis of key exchange protocols and physical protocols, *PhD Thesis,* ETH Zurich, 2012.
7. Blanchet, B., Modeling and verifying security protocols with the applied pi calculus and ProVerif, *Found. Trends Privacy Secur.,* 2016, vol. 1, nos. 1–2, pp. 1–135.
8. Blanchet, B., Automatic verification of correspondences for security protocols, *J. Comput. Secur.,* 2009, vol. 17, no. 4, pp. 363–434.
9. Blanchet, B., Automatic verification of security protocols in the symbolic model: The verifier ProVerif, *Lect. Notes Comput. Sci.,* 2012, vol. 8604, pp. 54–87.
10. Blanchet, B., CryptoVerif: A computationally sound mechanized prover for cryptographic protocols, *Proc. Dagstuhl Seminar on Formal Protocol Verification Applied,* 2007.
11. Elgamal, T., A public-key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inf. Theory,* 1985, vol. 31, no. 4, pp. 469–472.
12. Vinarskii, E., Proverif_code_optimisation. https://github.com/vinevg1996/proverif_code_optimisation. Accessed October 24, 2021.

Translated by Yu. Kornienko