Verification of Mutual Authentication Protocol for MobInfoSec System

Olga Siedlecka-Lamch[3], Imed El Fray[1], Mirosław Kurkowski[2], and Jerzy Pejaś[1](✉)

[1] Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, Szczecin, Poland
{ielfray,jpejas}@zut.edu.pl
[2] Institute of Computer Sciences, Cardinal Stefan Wyszynski University in Warsaw, Warsaw, Poland
mkurkowski@uksw.edu.pl
[3] Institute of Computer and Information Sciences, Częstochowa University of Technology, Częstochowa, Poland
olga.siedlecka@icis.pcz.pl

Abstract. This paper presents a detailed analysis of the mutual authentication protocol developed especially for the system MobInfoSec - for a mobile device to share and protect classified information. MobInfoSec uses fine-grained access rules described by general access structures. In this paper we describe the architecture and functioning of the system, and the requirements imposed on cryptographic authentication protocols, resulting from both: standards, the collection of good practices, as well as directly from the vision of the system. The article contains a description of the protocol's parts and formal analysis of its security.

Keywords: Authentication protocols · One-to-many protocol · Mobile device · Sensitive information · Secure communication channel

1 Introduction

The modern people process tens of gigabytes of information a day, most of which is transferred electronically, by mobile devices. Phones and tablets have become personal, handheld offices by which users download and often send sensitive content: emails, business and banking transactions, etc. Each participant of that mobile/distributed system wants one - security of transmitted and stored data. Security has many aspects: information is only accessible to the defined destination (it should not be stolen or intercepted), the encrypted information can not be lost or deciphered by inappropriate entity (see [7–9]).

Cryptographic protocols provide security relevant to the needs of information systems. This security concerns, inter alia: authentication of entities, session key agreement between the parties, provides confidentiality, integrity, anonymity and non-repudiation. Participants of cryptographic protocol exchange

K. Saeed and W. Homenda (Eds.): CISIM 2015, LNCS 9339, pp. 461–474, 2015.
DOI: 10.1007/978-3-319-24369-6_38

messages through a specific communication channel. Communication channels can be divided into three types: channel point-to-point (or one-to-one) connecting the two participants, broadcast channels (one to many) connecting the sender and multiple recipients and conference channels (many to many) connecting all participants in the protocol and allow for exchange of messages between all participants.

Cryptographic protocols are subject of rigorous analysis, as they represent a critical component of any secure distributed computer system. They are easy to write, but on the basis of the code its very difficult to estimate their security level. Their simple structure is often confusing and leads to false conclusions in their security evaluation. Therefore, an important element in the selection and design of cryptographic protocol is to verify its correctness.

The article shows a mutual authentication protocol designed for system MobInfSec, which enables cryptographic protection of sensitive information in accordance with Originator Controlled access control rules [6,14]. The ORCON rules release a user from the obligation to monitor any information (especially against unauthorized copying). The information is removed when a user is no longer allowed to access it.

A description of the assumptions, architecture, and the rules of functioning of the system MobInfoSec are included in the next chapter. A further section describes the objectives and requirements for the proposed protocol. The next chapter describes its functioning and at the end the verification of its correctness. The whole is closed by summary, containing further directions of research.

2 Objective

The paper gives mutual authentication protocol under the name of SP2SP_Mutual_Auth (Secret Protection module of user A to Secret Protection module of user B mutual authentication) designed for MobInfSec system that allows protection of cryptographic confidential information in accordance with the ORCON rules.

This protocol is initialized by an entity A (called the chairman). The chairman executes (sequentially or simultaneously) n times the one-to-one protocol with every other member B_1, B_2, ..., B_n from the group B. Successful completion of each instance of SP2SP_Mutual_Auth protocol enables to authenticate every pair of users (A, B_i), $i = 1, \ldots, n$, and to establish n independent secure communication channels between them with different key material used by each pair of participants.

The security analysis of SP2SP_Mutual_Auth protocol is based on its specification written in HLPSL and ProToc languages and next used in well-known tools of automatic protocol verification like AVISPA, VerICS and PathFinder (see [1,10,13]. We assume that the adversaries inside our system have many capabilities of the standard Dolev-Yao intruder, namely, they are able, within their bounded storage capacity, to compose, decompose, overhear, and intercept messages as well as update values with fresh ones. Hence, it is commonly

believed that Dolev-Yao intruder is the most powerful attacker because following the seminal work of Dolev and Yao [4], the communication media are assumed to be under absolute control of the intruder. This intruder can in particular destroy all transmitted messages.

The results of our security analysis have showed the correctness of the proposed protocol for the different states of $SP.B_i$ and $SP.A_i$ tokens when performing cryptographic operations and during authentication.

3 Architecture of MobInfoSec System

MobInfoSec system can be seen as a set of cooperating applications, including trusted parts, that are distributed in different locations and communicate with each other. More detailed assumptions, the architecture and functioning of the system MobInfoSec are described in [5]. At this point, we focus only on a description of those elements that have an impact on SP2SP_Mutual_Auth protocol and its security analysis.

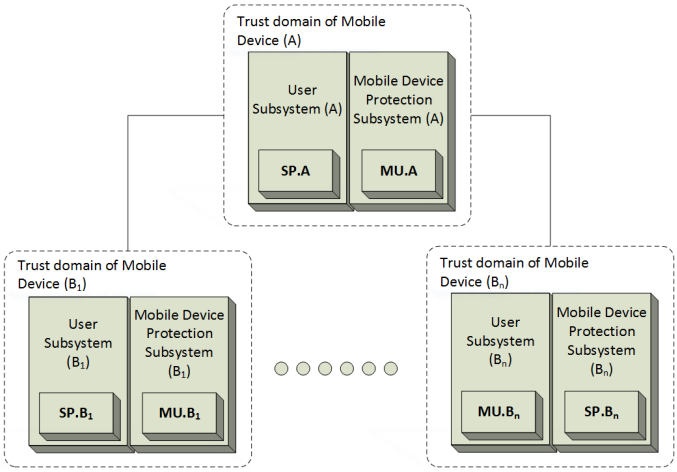

Fig. 1. Trust domains concept for different mobile devices

Applications can be closed in the domain of trust. A single domain is created around a trusted application or a group of trusted applications. We assume that communication between applications within the same trust domain is secure, could be caused by placing them in one location or the use of ready-made security technologies (eg. SSL).

Communication between domains therefore remains an open problem, namely: communication between components located in different domains, which will require the compilation of trusted paths and channels. Paths/channels created by the use of strong cryptography, allows applications from different domains to trust each other and accept their decisions. The main task is to define domains of trust and determine trusted paths and channels [5, 7].

4 Basic Authentication Protocols

The aim of the cryptographic authentication protocol is to identify a particular or all of the participants of communication. Typically, the additional effect of the protocol is the key - established between the participants of the protocol, which will be used to build a trusted channel or a trusted path.

Another goal is to provide for messages exchanged during protocol execution the following (all or some) security features: confidentiality, authentication, integrity or non-repudiation. As mentioned earlier authentication protocols can be divided into unilateral, mutual and multilateral protocols of authentication of entities.

In the case of MobInfoSec system the secure communications between multiple entities is required. The main point is to ensure the proper authentication between the SP components which are located in different trust domains. One of the domains (a chairman) is the initiator of communication and should be mutually authenticate with each of the other domains of trust and establish a secure communication channel.

There are few potentially useful protocols (Table 1) that can be considered in MobInfoSec ([11,15,16]). All of them guarantee mutual authentication, key integrity, key authentication, key control or confirmation, but only the first has a formal security proof. None of them provides a common communication channel or the secret handling. No matching of existing solutions to the system assumptions forced to design a new mutual authentication protocol.

5 Mutual Authentication Protocol

Cryptographic protocol design must be preceded by a clear definintion of project objectives and a determination of the impact of violations of these assumptions on achieving planned objectives of security. Examination of protocol defects that were not included in the assumptions (e.g. due to the lack of knowledge related to a given defect) motivates and allows to understand the various design features of the protocol, and the knowledge of successful (and known) attacks helps designers to avoid standard attacks.

Principles of engineering design of cryptographic protocols are associated with the three phases [3]:

- analysis of the protocol requirements phase,
- detailed protocol design phase,
- proving protocol security phase.

Below, we present SP2SP_Mutual_Auth one-to-many group mutual authentication protocol. In this protocol the group members are authenticated by the chairman "one-by-one". That is, n authentication messages are required to authenticate n group members. Then, these members share individual keys for the communication with the group chairman.

Table 1. Authentication protocols comparison

No.	Property	Protocol (variant)		
		Transport RSA (EN 14890) [15]	Key transport ISO/IEC 11770-3 Mechanism 5 [16]	Lim-Lee key agreement protocol 5 [11]
1.	Mutual authentication	+	+	+
2.	Multi-party authentication (one to many and many to one)	-	-	-
3.	Key integrity	+	+	+
4.	Key authentication	+	+	+
5.	Personal (independent) communication channels	+[1)]	+[1)]	+[1)]
6.	Common communication channel	-	-	-
7.	Forward secrecy	-	-	-
8.	Backward secrecy	N/A [2)]	N/A	N/A
9.	Liveness	+ [3)]	+[3)]	+
10.	Key control	+	+[4)]	+
11.	Key freshness	+[3)]	+[3)]	+[3)]
12.	Key confirmation	+	+	+
13.	Formal security proof	+	+[6)]	+

Legend

+ Means that the protocol has indicated property, perhaps after meeting additional requirements presented in footnote

[1)] It applies also to the case when the protocol is used to authenticate the initiator of the protocol with other members of the participants group

[2)] N/A - not applicable

[3)] Applies to all participants of the protocol

[4)] Applies only to the initiator of the protocol

[5)] If session key is used by all members of the group

[6)] Lack of information about the existence of a formal security proof

5.1 Assumptions and Notations

For the proposed protocols the following assumptions have been made:

- types of connection and transmission medium (LAN, WLAN, WWAN, ...) used by the parties involved in the protocol are not significant,
- the term 'address' of the device means: complete and current information that helps to communicate with the device through the selected communication medium; method of processing and distribution of this information is beyond the scope of this paper,
- All parties involved in the described authentication protocol are equipped with SP, under their exclusive control,
- SP provides an interface compatible with the `PKCS#11`.

The protocol specification uses the following symbols and markings:

AUT	authentication
C	Certificate (X509 format or CVC card format)
CS	Signing certificate that contains the public part of the key used to submit and verify the signature
CS_AUT	Signing certificate used to authenticate other certificates; the private, key complementary to the public key contained in CS_AUT, may be used to sign other certificates
C.CAX.CS_AUT	A certificate issued by the main office RCA to intermediate CA certification authority used by this office to authenticate the public key of X
C.X.AUT	The certificate containing the public key of the entity X used in the authentication procedure
D[key](msg)	Decrypt a message <msg> with the key <key>
DS[key](msg)	Digital signature of messages <msg> with the key <key>
E[key](msg)	Encrypt a message <msg> with the key <key>
h(msg)	The message digest calculated for message <msg> using a hash function h
ID	id
MU.A	User Subsystem Authentication Module installed in the mobile device under the control of the user A
MDC.A	Protected Data Magazine of User A
MK.A	Keys Magazine of User A
PrK	The private key
PRND	Complementary random number
PuK	Public key
PuK.CAX.CS_AUT	The public key contained in the certificate C.MAX.CS_AUT used for authentication of public keys
Q||Z	Concatenation information of Q and Z
SK	Secret key (symmetric)
RCA	Root CA.
SP.A	Secret Protection Module installed in the mobile device user A
SP.SU	Secret Protection Module installed on the Authentication Server
SU	The authentication server (one of the functions of a Trusted Third Party Distribution and Authentication Users and Devices Server)
UM.A	Mobile device under the control of user A
X	Protocol participant
XC	Mutual certificate

We assume that for a given set of entities $P = A, B_1, ..., B_n$, where $n \geq 1$, the entity A is the preferred entity responsible for initiating the protocol. The aim of the protocol is the mutual authentication with each entity B_i, $i = 1, \ldots, n$, and generation of a key material necessary to ensure the confidentiality and authen-

ticity of information exchanged between the parties. Additionaly we assume that:

- SP tokens have the same root CA certificate; the case of several certificates is not included in the protocol; however, if we assume that the two root CAs have issued cross certificates to each other, the protocol - after the introduction of minor modifications - will also work correctly;
- token SP.X of the entity X stores in its memory: the root CA's certificate C.RCA.AUT, the intermediate CA's certificate C.CASP.X.CS_AUT CA issued by root CA, and certificate C.SP.X.AUT of the token SP.X.

5.2 Protocol description

The SP2SP_Mutual_Auth$(A, B_1, ..., B_n)$ protocol provides authentication of each pair of entities (A, B_i), $i = 1, \ldots, n$ and consists of the following states:

State 0: `SP.B_i` and `SP.A` tokens do not possess the public keys of the opposite side.

1. Authentication `MU.A` module initiates the protocol, establishes a connection to authentication module `MU.B_i` of mobile device `B_i` and and carries protocol in the initial state.
2. If token `SP.B_i` do not have public key `PuK.CASP.A.CS_AUT`, then:
   2.1 `MU.A` requests through `MU.B_i` selection and verification of key `PuK.RCA.AUT` by `SP.B_i`:
   `MU.A -> MU.B_i -> SP.B_i: select and verify(PuK.RCA.AUT)`
   2.2 `SP.B_i` chooses and verifies `PuK.RCA.AUT` key, and then returns the confirmation:
   `SP.B_i -> MU.B_i -> MU.A: conf.OK`
   2.3 `MU.A` ask `SP.A` to read certificate `C.CASP.A.CS_AUT`:
   `MU.A -> SP.A: get certificate (C.CASP.A.CS_AUT)`
   2.4 `SP.A` gets certificate `C.CASP.A.CS_AUT` and returnt it to the `MU.A`:
   `SP.A -> MU.A: C.CASP.A.CS_AUT`
   2.5 `MU.A` requests through `MU.B_i` that `SP.B_i` verified certicate `C.CASP.A.CS_AUT`:
   `MU.A -> MU.B_i -> SP.B_i: verify certificate (C.CASP.A.CS_AUT)`
   2.6 `SP.B_i` verifies certificate `C.CASP.A.CS_AUT`, saves the public key `PuK.CASP.A.CS_AUT` and sends confirmation to the `MU.A`:
   `SP.B_i -> MU.B_i -> MU.A: conf.OK`
3. `MU.A` through `MU.B_i` requests `SP.B_i` to choose and verify the key `PuK.CASP.A.CS_AUT`:
   `MU.A -> MU.B_i -> SP.B_i: select and verify(PuK.CASP.A.CS_AUT)`
4. `SP.B_i` chooses and verifies `PuK.CASP.A.CS_AUT` key, and then returns the confirmation:
   `SP.B_i -> MU.B_i -> MU.A: conf.OK`
5. `MU.A` asks `SP.A` to read its authentication certificate `C.SP.A.AUT`:
   `MU.A -> SP.A: get certificate (C.SP.A.AUT)`

6. SP.A gets C.SP.A.AUT certificate and returns it to the MU.A:
   SP.A -> MU.A: C.SP.A.AUT
7. MU.A through MU.B_i requests SP.B_i to verify the certificate C.SP.A.AUT of token SP.A:
   MU.A -> MU.B_i -> SP.B_i: verify certificate (C.SP.A.AUT)
8. SP.B_i verifies the C.SP.A.AUT certificate, stores the public key PuK.SP.A.AUT and sends a confirmation to MU.A:
   SP.B_i -> MU.B_i -> MU.A: conf.OK

State 1: Token SP.B_i has the public key C.SP.A.AUT of token SP.A

9. If SP.A does not have PuK.CASP.B_i.CS_AUT public key, then:
   9.1 MU.A requests the SP.A to select and verify key PuK.RCA.AUT:
   MU.A -> SP.A: select and verify(PuK.RCA.AUT)
   9.2 SP.A chooses and verifies PuK.RCA.AUT key, and then returns the confirmation:
   SP.A -> MU.A: conf.OK
   9.3 MU.A through MU.B_i requests SP.B_i to read C.CASP.B_i.CS_AUT certificate:
   MU.A -> MU.B_i -> SP.B_i: get certificate (C.CASP.B_i.CS_AUT)
   9.4 SP.B_i gets C.CASP.B_i.CS_AUT certificate and returns it to the MU.A:
   SP.B_i -> MU.B_i -> MU.A: C.CASP.B_i.CS_AUT
   9.5 MU.A requests to SP.A verified C.CASP.B_i.CS_AUT certificate:
   MU.A -> SP.A: verify certificate (C.CASP.B_i.CS_AUT)
   9.6 SP.A verifies the certificate C.CASP.B_i.CS_AUT, stores the public key PuK.CASP.B_i.CS_AUT and sends a confirmation to MU.A:
   SP.A -> MU.A: conf.OK
10. MU.A requests the SP.A to select and verify key PuK.CASP.B_i.CS_AUT:
    MU.A -> SP.A: select and verify(PuK.CASP.Bi.CS_AUT)
11. SP.A chooses and verifies PuK.CASP.B_i.CS_AUT key, and then returns the confirmation:
    SP.A -> MU.A: conf.OK
12. MU.A through MU.B_i requests SP.B_i to read authentication certificate C.SP.B_i.AUT:
    MU.A -> MU.B_i -> SP.B_i: get certificate (C.SP.B_i.AUT)
13. SP.B_i gets C.SP.B_i.AUT certificate and returns it to the MU.A:
    SP.B_i -> MU.B_i -> MU.A: C.SP.Bi.AUT
14. MU.A requests to SP.A verified the certificate C.SP.B_i.AUT of token SP.B_i:
    MU.A -> SP.A: verify certificate (C.SP.B_i.AUT)
15. SP.A verifies the certificate C.SP.B_i.AUT , stores the public key PuK.SP.B_i.AUT and sends a confirmation to MU.A:
    SP.A -> MU.A: conf.OK

State 2: Token SP.A has the public key C.SP.B_i.AUT of token SP.B_i

16. MU.A through MU.B_i requests SP.B_i to activate (select) PuK.SP.A.AUT and PrK.SP.B_i.AUT keys:
    MU.A -> MU.B_i -> SP.B_i:
    acivate security key (PuK.SP.A.AUT, PrK.SP.B_i.AUT)
17. SP.B_i activates PrK.SP.B_i.AUT and PuK.SP.A.AUT keys and sends confirmation to the MU.A:
    SP.B_i -> MU.B_i -> MU.A: conf.OK
18. MU.A requests that SP.A activated (selected) PuK.SP.B_i.AUT and PrK.SP.A.AUT keys:
    MU.A -> SP.A: acivate security key (PuK.SP.B_i.AUT, PrK.SP.A.AUT
19. SP.A activates PrK.SP.A.AUT and PuK.SP.B_i.AUT keys and sends confirmation to the MU.A: SP.A -> MU.A: conf.OK

State 3: SP.A and SP.B_i tokens activated their keys, which are necessary during performing cryptographic operations; moreover, the public key of the SP.A is now known by the SP.B_i and *vice versa*, and can be trusted by both sides.

20. MU.A requests the SP.A to generate a random number and return it together with its ID:
    MU.A -> SP.A: get challenge
21. SP.A generates RND.SP.A and, together with its identifier SN.SP.A sends all to MU.A:
    SP.A -> MU.A: RND.SP.A||SN.SP.A
22. MU.A through MU.B_i requests to authenticate SP.B_i:
    MU.A -> MU.B_i -> SP.B_i: authenticate(RND.SP.A || SN.SP.A)
23. SP.B_i generates a random key K.SP.B_i, random padding PRND.SP.B_i, prepares preToken.SP.B_i, signs the concatenated data using its private key, encrypts it and then through MU.B_i return it to MU.A:
    SP.B_i -> MU.B_i -> MU.A:
    E[PuK.SP.A.AUT](DS[PrK.SP.B_i.AUT](preToken.SP.B_i))
    where:
    preToken.SP.B_i = textA.SP.B_i || PRND.SP.B_i || K.SP.B_i
    || h(PRND.SP.B_i || K.SP.B_i || RND.SP.A || SN.SP.A) || textB.SP.B_i
24. MU.A sends to SP.A the request for verification of authentication token:
    MU.A -> SP.A:
    verify(E[PuK.SP.A.AUT](DS[PrK.SP.B_i.AUT](preToken.SP.B_i)))
25. SP.A decrypts E[PuK.SP.A.AUT](DS [PrK.SP.B_i.AUT](preToken.SP.B_i)) and after the verification by SP.A the signature of SP.B_i (after confirmation of compliance with the previously sent random challenge RND.SP.A) sends confirmation to the MU.A: SP.A -> MU.A: conf.OK

State 4: SP.B_i token has been authenticated to the SP.A.

26. MU.A requests through MU.B_i that SP.B_i generated a random number and send it together with its ID:
    MU.A -> MU.B_i -> SP.B_i: get challenge
27. SP.B_i generates RND.SP.B_i challenge and, together with its identifier SN.SP.B_i sends all to MU.A:
    SP.B_i -> MU.B_i -> MU.A: RND.SP.B_i || SN.SP.B_i
28. MU.A requests authentication by SP.A:
    MU.A -> SP.A: authenticate(RND.SP.B_i || SN.SP.B_i)
29. SP.A generates a random key K.SP.A, random padding PRND.SP.A, prepares preToken.SP.A, signs the concatenated data using its private key, encrypts it and then sends to MU.A:
    SP.A -> MU.A: E[PuK.SP.B_i.AUT](DS[PrK.SP.A.AUT](preToken.SP.A))
    where:
    preToken.SP.A = textA.SP.A || PRND.SP.A || K.SP.A
    || h(PRND.SP.A || K.SP.A || RND.SP.B_i || SN.SP.Bi) || textB.SP.A
30. MU.A sends via MU.B_i request to SP.B_i for verification of authentication token:
    MU.A -> MU.B_i ->
    verify(SP.Bi: E[PuK.SP.B_i.AUT](DS[PrK.SP.A.AUT](preToken.SP.A)))
31. SP.B_i decrypts E[PuK.SP.B_i.AUT](DS[PrK.SP.A.AUT](preToken.SP.A)) and after the verification by SP.B_i the signature of SP.A (after confirmation of compliance with the previously sent random challenge RND.SP.B_i), token SP.B_i sends confirmation to MU.A:
    SP.B_i -> MU.B_i -> MU.A: conf.OK

    State 5: SP.A token has been authenticated now to the opposite party, i.e. to SP.B_i.

32. After performing of the protocol SP.B_i and SP.A have a confidential key material K.SP.A and K.SP.B_i. On this basis both parties calculate the symmetric difference:
    K.SP.A/SP.B_i = K.SP.A $\oplus$ K.SP.A
    and then create session keys to ensure confidentiality and authentication of the message. It being understood that:
    K.SP.A/SP.B_i = Ka(ENC) || Kb (ENC) || Ka (MAC) || Kb (MAC)
    Other more general methods for generating keys to ensure the confidentiality and authentication can be found in [15] (see chap. 8.10).

    State 6: SP.B_i and SP.A tokens are able to set the trusted channel.

5.3 Protocol's verification

Protocol analysis was performed using known automatic verification tools: Avispa [1], VerICS [10] and PathFinder [12,13]. In the case of communication over an insecure channel, in an open way, without any encryption, and in which

the goal for subprotocol is not to maintain the confidentiality of the new data (keys, nonces) or user authentication, then automatic verification tools could not be used.

The need to introduce this type of communication results from the fact that the parties haven't established yet a secure communication channel. In these cases, the intruder can perform only flooding-type attack or disrupt communications. They are, however, risks faced by all communication systems. Justification of subprotocols' correctness, in such cases, is based on the analysis of the correctness of data transfer scheme, in order to achieve the objectives and the assumption of a trusted repository and duly signed certificates.

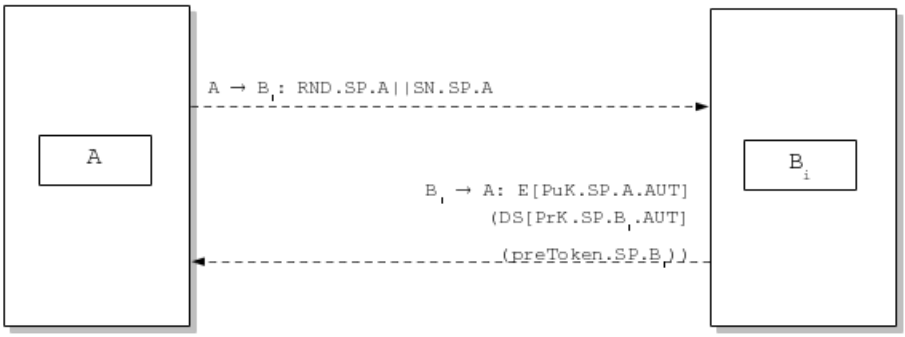

Fig. 2. Simplified diagram of subprotocol: State 3

Accordingly, only two parts of the protocol (subprotocols of states 3 and 4) are designed to maintain the confidentiality of the new data (keys, nonces), and providing authentication of users. These two subprotocols provides mutual entity authentication. Because the messages sent are independent of each other, it is important to note that the subprotocol of state 4 is unambiguously symmetrical (similar) to the subprotocol of state 3. Therefore, in the description of these subprotocols we will focus only on the subprotocol of state 3. It is easy to observe that in other subprotocols there is no possibility of possessing by Intruder important data, so correctness of the whole protocol from security and authentication point of view is assured. The analysis of the correctness and the security of subprotocol of state 3 is made on the basis of the data transmission diagram (Fig. 2) and the experimental results obtained by using formal methods and aforementioned automatic tools.

In conducted studies the Dolev-Yao Intruder model was used, which is widely considered in the literature. According to this model the Intruder has full access to the network and transmitted data, he can decompose and compose transmitted data according to held by him cryptographic keys. The only assumption that limits privileges of the Intruder is the perfect cryptography assumption - the

inability to decode the corresponding ciphertext without knowing the encryption key.

Specifications were made according to the syntax of HLPSL and ProToc languages and the data transmission of tested subprotocol. Users participating in the protocol and security goals guaranteed by this protocol were also modeled. Specifications in HLPSL is extensive, and does not introduce no additional information in relation to the specifications in ProToc, which is demonstrated below (see Listing 1).

From the viewpoint of the tested security properties, all modules of AVISPA tool reported the SP2SP_Mutual_Auth protocol correctness for a limited number of sessions, and one of AVISPA module reported also the correctness to an unlimited number of sessions.

VerICS tool generated 18 hypothetical runs, and for each of them built an automata model, which was then encoded into the Boolean formula. The formula was verified by the SAT solver MiniSAT. The result showed that in the surveyed space and with the adopted assumptions, protocol is correct and no errors were found in its structure.

For generated runs the PathFinder tool created chains of states. An attempt to construct a tree of runs containing a path of attack failed. This proves the correctness of subprotocol.

Listing 1. Authentication protocol specification in ProToc language

```
BEGIN
    Users(2)
    Players(3)
    Steps(2)
    Intruder(DY)

    Protocol:
        A; N_A, i(A); N_A; N_A|i(A); B;
        B; +K_A, -K_B, K_AB, N_A, i(A); K_AB;
        <<K_AB, h(K_AB, i(A))>_-K_B>_+K_A;
    Session
        (A,B,I)
    Goals
        Authentication(B,A)
        Secrecy(K_AB)
End
```

Verification for selected parts of the protocol for all assumed parameters, fared well - the protocol is correct and secure.

All results and times are listed in the table 2.

Table 2. Summary of the results

Subprotocol	AVISPA				VerICS	PathFinder
	OFMC	CL-AtSe	SATMC	TA4SP		
State 3	SAFE 70 ms.	SAFE <10 ms.	SAFE 30 ms.	SAFE 661 ms.	SAFE 15 ms.	SAFE <10 ms.
State 4	SAFE 70 ms.	SAFE <10 ms.	SAFE 30 ms.	SAFE 661 ms.	SAFE 15 ms.	SAFE <10 ms.

6 Summary

In the paper a mutual authentication protocol was presented, it was designed specifically for MobInfoSEc system, to guarantee secure communication for mobile devices. The protocol provides mutual authentication between each pair of participants of communication, establishing a common key material, and thus setting up a secure communication channel.

The most important security properties of the described protocol were tested using three different automatic verification tools. During the verification perfect cryptography and the Dolev-Yao intruder model were assumed. The SP2SP_Mutual_Auth$(A, B_1, ..., B_n)$ protocol has passed verification and achieved its objectives.

Acknowledgments. This scientific research work is supported by NCBiR of Poland (grant No PBS1/B3/11/2012) in 2012-2015.

References

1. Armando, A., et al.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
2. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer-Verlag, Heidelberg (2003)
3. Dong, L., Chen, K.: Cryptographic Protocol Security Analysis Based on Trusted Freshness. Springer-Verlag, Heidelberg (2012)
4. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(2), 198–207 (1983)
5. El Fray, I., Hyla, T., Kurkowski, M., Maćków, W., Pejaś, J.: Practical authentication protocols for protecting and sharing sensitive information on mobile devices. In: Kotulski, Z., Księżopolski, B., Mazur, K. (eds.) CSS 2014. CCIS, vol. 448, pp. 153–165. Springer, Heidelberg (2014)
6. Hyla, T., Pejaś, J.: Certificate-based encryption scheme with general access structure. In: Cortesi, A., Chaki, N., Saeed, K., Wierzchoń, S. (eds.) CISIM 2012. LNCS, vol. 7564, pp. 41–55. Springer, Heidelberg (2012)
7. Hyla, T., Pejaś, J., El Fray, I., Maćków, W., Chocianowicz, W., Szulga, M.: Sensitive information protection on mobile devices using general access structures. In: ICONS 2014, The Ninth International Conference on Systems, pp. 192–196. IARIA (2014)

8. Hyla, T., Pejaś, J.: A practical certificate and identity based encryption scheme and related security architecture. In: Saeed, K., Chaki, R., Cortesi, A., Wierzchoń, S. (eds.) CISIM 2013. LNCS, vol. 8104, pp. 190–205. Springer, Heidelberg (2013)
9. Hyla, T., Maćków, W., Pejaś, J.: Implicit and explicit certificates-based encryption scheme. In: Saeed, K., Snášel, V. (eds.) CISIM 2014. LNCS, vol. 8838, pp. 651–666. Springer, Heidelberg (2014)
10. Kurkowski, M., Penczek, W.: Verifying Security Protocols Modeled by Networks of Automata. Fund. Inform. **79**(3–4), 453–471 (2007)
11. Lim, C.H., Lee, P.J.: Several practical protocols for authentication and key exchange. Information Processing Letters **53**, 91–96 (1995)
12. Kurkowski, M., Siedlecka-Lamch, O., Szymoniak, S., Piech, H.: Parallel bounded model checking of security protocols. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2013, Part I. LNCS, vol. 8384, pp. 224–234. Springer, Heidelberg (2014)
13. Siedlecka-Lamch, O., Kurkowski, M., Piech, H.: A new effective approach for modeling and verification of security protocols. In: Proceedings of 21th international Workshop on Concurrency. Specification and Programming (CS&P 2012), pp. 191–202. Humboldt University Press, Berlin (2012)
14. Chen, Y.-Y., Lee, R.B.: Hardware-assisted application-level access control. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 363–378. Springer, Heidelberg (2009)
15. prEN 14890–1 Application Interface for smart cards used as Secure Signature Creation Devices - Part 1: Basic services (2012)
16. ISO/IEC 11770–3:2008 Information technology - Security techniques - Key management - Part 3: Mechanisms using asymmetric techniques (2008)