Protocols security analysis using modern tools of verification

Liudmila Babenko
IT security department
Southern Federal University
Taganrog, Russian Federation
lkbabenko@sfedu.ru

Ilya Pisarev
IT security department
Southern Federal University
Taganrog, Russian Federation
ilua.pisar@gmail.com

Abstract

Cryptographic protocols are the core of any protected system. With the help of them, data are transmitted that need protection from third parties. The paper examines the features of analyzing the security of cryptographic pro-tocols using various modern verification tools. Basic verification goals, such as party authentication, data confidentiality, integrity control, have been set. A simplified cryptographic protocol of electronic voting based on blind intermediaries is described. Present-day effective means of verification of Avispa, SPIN are presented. The voting protocol in the CAS + language is described in the Avispa tool, the verification objectives in the form of data secrecy and authentication of the parties are indicated, the scheme of the transmitted data is shown in the presence of an attacker. The protocol of voting in the language Promela in the tool SPIN is described, verification objectives in the form of authentication and verification of the correct completion of the protocol are established in the presence of an active attack by the attacker. The analysis of the protocol's safety with the help of the given means of verification was carried out. The possibilities and limitations of each tool are shown when verifying compliance with verification objectives. Recommendations on the use of each of the means are given.

1 Introduction

The security of secure systems directly depends on the quality of the cryptographic protocols used in them. Various verifiers are used to analyze the security of protocols. In this paper, the most popular verifier Avispa [Avi06], specifically designed for security analysis of cryptographic protocols, and the more versatile tool SPIN [Ben08] are considered. However, it is not always possible to produce a full analysis of the protocol using one of the verifiers provided. Each of the presented tools has its own peculiarities and is used depending on the chosen verification objectives. Thus, the goal of this paper is to identify the features of the protocol safety analysis using the verification tools provided and to make recommendations on the use of each of them.

In: Marco Schaerf, Massimo Mecella, Drozdova Viktoria Igorevna, Kalmykov Igor Anatolievich (eds.): Proceedings of REMS 2018 – Russian Federation & Europe Multidisciplinary Symposium on Computer Science and ICT, Stavropol – Dombay, Russia, 15–20 October 2018, published at http://ceur-ws.org

2 Verification Goals

When analyzing the security of cryptographic protocols, the use of the Dolev-Yao threat model is assumed [Dol83]. This involves the presence of an attacker who can monitor the data transfer between the parties, block it, intercept and modify all messages. The protocol must be resistant to this kind of attacker and ensure the secrecy, authenticity and integrity of the data. Thus, the goals of verification of the security of cryptographic protocols are:

1. Privacy of transmitted data
2. Integrity of transmitted data
3. Authentication of the parties

The purpose of the confidentiality of the transmitted data is set so that the attacker can not know the contents of the transmitted data between the legal parties. To achieve these goals, asymmetric and symmetric encryption is used, depending on the type of data. Typically, asymmetric ciphers are used to transfer a small amount of data, such as keys for other ciphers, identifiers, random numbers, since asymmetric encryption is limited by the smallest bit of the cipher itself. Symmetric ciphers are used to encrypt a large amount of data, but unlike asymmetric ciphers using a public key for encryption and secret for decryption, a common secret key is used. This makes it necessary to use a pre-shared secret encryption key, or to generate it using the protocols for generating a common session key, such as Diffie-Hellman.

The purpose of the integrity of the transmitted data is to ensure that the attacker can not modify the data transmitted between the legal parties. If an integrity violation is detected, either a repeated data request or the end of the current session is performed. To ensure the integrity of the data, the authentication codes of the MAC message, which are one of the symmetric cipher encryption modes, HMAC (hash-based message authentication code), the mechanism for exchanging data using a private key and hash functions, signature with a secret key and signature verification with using the public key of the asymmetric cipher.

The purpose of party authentication is to ensure that the parties are confident that they are interacting in the current session that the messages contain data of the present time and the parties are sure that they communicate with each other. To ensure authentication, a request-response scheme can be used in which one side sends a random number to the other party and the other party in the response message sends the same number or some function performed on this number and which the original side knows. One of the types of authentication attacks is replay-attack [Syv94]. It consists in reusing the previously transmitted legal message. To prevent this, you can use both random numbers in conjunction with the regeneration of the session interaction keys at the beginning of each session, and the use of timestamps in the message.

3 Simplified E-Voting Protocol Based On Blind Intermediaries

As a protocol for verification, a simplified e-voting protocol is used, the extended version of which is used in the electronic voting system based on blind intermediaries [Bab17].Any connection between the client and the component begins with the initialization of the session key. All transmitted data between the parties is encrypted with a symmetric AES cipher, in the CBC mode with 128 bits IV (initialization vector) and HMAC-SHA256 256 bits with a 128 bit key for integrity checking control. The key of the session is the combination of [key, IV, HMAC key], its total length is 512 bits. The session key is generated using the Diffie-Hellman protocol on elliptical curves using ephemeral keys and signing the secret parts.

(1) A -> B: “Hello”
(2) B -> A: DHB, $Sign_{Bsk}$(DHB), B_Certificate
(3) A: verifies the certificate and signature
(4) A -> B: DHA
(5) Both parties generate a key k for symmetric encryption

It is important that two components authenticate each other, so both of them make the secret part signatures. When connecting to the user, it is important only to authenticate the server, so the signature is required only from the server. Denote ECDHE (A, B) = k as generating the session key k via the Diffie-Hellman protocol between parties A and B. When voting, the following protocol is used, based on the principle of blind intermediaries

[Bab17], simplified scheme of which shown in Figure 1. It provides the ability to authenticate the user on his personal data while preserving the anonymity of the vote. E-voting protocol:

(1) ECDHE (V, AS) = vas;
(2) ECDHE (V, VS) = vvs;
(3) ECDHE (AS, VS) = asvs;
(4) AS-> V: $E_{vas}(N_{as})$, HMAC1
(5) VS-> V: $E_{vvs}(N_b,N_{vs})$, HMAC2
(6) VS -> AS: $E_{asvs}(N_{asvs})$, HMAC3
(7) V -> AS: E_{vas} (N_{as}, authData, $E_{vvs}(N_{vs}$, vote), HMAC4), HMAC5
(8) AS-> VS: E_{asvs} (N_{asvs}, $E_{vvs}(N_{vs}$, vote), HMAC4)), HMAC6

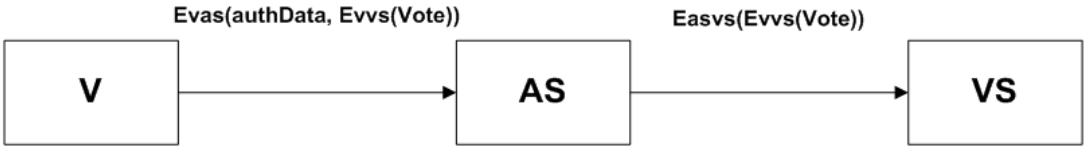

Figure 1: Simplified scheme of blind intermediaries principle

Nb is N blinded (number of blinding), a non-random random number, which is regenerated each time. It is introduced in order to add some data before the semantic random number for making full search more complicated (in particular, it is necessary to select two encryption keys for message 7 in order to find authData). Session keys are generated by (1)-(3). Randomly generated random numbers are sent as shown in (4)-(6) to authenticate the parties and to enhance the sparseness of subsequent data in the message when CBC encryption mode is applied. HMAC-SHA256 with a 128-bit key is used for integrity monitoring of all transmitted messages. The message (7) uses the principle of blind intermediaries [Bab17]. The voter encrypts his vote vote on the session key with VS, applies his personal data to the ciphertext, and encrypts it on the session key with AS. AS hashes the sent personal data, searches for the hash in the database and, and, if detected, redirects the message to the VS component. VS memorizes the vote.

4 Avispa

Avispa is a tool for automated security analysis of cryptographic protocols [Avi06]. With the help of Avispa, in the context of the developed protocols, it is possible to verify: the authentication of the parties, the secrecy of data and protection against replay attacks. The possibility of verification of integrity when using the HMAC mechanism is not provided. The protocol will be analyzed after the phase of development of the common session key between the parties. The protocol will be described in the CAS + [Sai11] language, then translated using the Avispa translator into HLPSL [Ohe05]. The instrument allows verification in one of 4 modes: OFMC [Bas05], ATSE [Tur06], SATMC [Arm04], TA4SP [Glo06]. The CAS + language is quite simple and allows you to quickly describe your protocol. However, Avispa works with the HLPSL language, which is more complicated. When translating a description to CAS + in HLPSL, errors may occur that need to be corrected manually. The description of the protocol in CAS + is given below.

```
1 protocol EVoting;
2 identifiers
3     V,AS,VS: user;
4     Nas,Nvs,Nasvs,AuthData,Vote : number;
5     Kvas,Kvvs,Kasvs: symmetric_key;
6
7 messages
8 1. VS -> V   : {Nvs}Kvvs
```

```
 9 2. VS -> AS  : {Nasvs}Kasvs
10 3. AS -> V   : {Nas}Kvas
11 4. V -> AS   : {Nas,AuthData,{Nvs,Vote}Kvvs}Kvas
12 5. AS -> VS  : {Nasvs,{Nvs,Vote}Kvvs}Kasvs
13
14 knowledge
15      V:V,AS,VS,Nas,Nvs,AuthData,Kvas,Kvvs,Vote
16      AS:V,AS,VS,Nvs,Nasvs,AuthData,Kvas,Kasvs
17      VS:V,AS,VS,Nasvs,Nvs,Kvvs,Kasvs,Vote
18
19 session_instances
20      [V:v,AS:as,VS:vs,Kvas:kvas,Kvvs:kvvs,Kasvs:kasvs]
21      [V:v,AS:as,VS:vs,Kvas:kvas,Kvvs:kvvs,Kasvs:kasvs];
22
23 intruder_knowledge
24      v,as,vs;
25
26 goal
27      secrecy_of Nvs [V,VS];
28      secrecy_of Nasvs [AS,VS];
29      secrecy_of Nas [V,AS];
30      secrecy_of Nv [V,VS];
31      secrecy_of AuthData [V,AS];
32      secrecy_of Vote [V,VS];
33      AS authenticates V on Nas;
34      VS authenticates AS on Nasvs;
35      VS authenticates V on Nvs;
```

In the area (2) - (5) are indicated the role sides, encryption keys, data. In the area (7) - (12) is described the transfer of messages between roles, what data is transmitted and on which key are encrypted. In the area (14) - (17), the knowledge of the parties is described. In the area (14) - (21) are indicated the sessions and knowledge of the attacker. In the area (26) - (35), the purposes of verification for the authentication of the parties and secrecy of the data are established. After checking the protocol, if an attack is detected, an attack route is displayed (in this case, an attack on secrecy, in case the attacker knows the Kvas key) as in Figure 2. Then you can see the scheme of interaction in the presence of an attacker as in Figure 3. As can be seen from the field of intruder knowledge, after a successful attack, an attacker managed to acquire nonce-5 and nonce-3 values, which are a random number of Nas and authData user authentication data.

Thus, the Avispa tool allows you to check the protocol for attacks on privacy and authentication, including the presence of replay-attack. The description of the protocols can be performed in a very simple CAS + language, but it still requires translation into HLPSL, which is more complicated. To detect attacks, you must manually assign scan targets for each element of the cryptographic protocol, such as a random number or a transmitted identifier. This tool can be used for initial analysis of the cryptographic protocol and allows to detect typical attacks with the subsequent study of the graphical representation of the data transmitted between the parties.

5 SPIN

SPIN (Simple Promela Interpreter) [Hol03] is a formal verifier for multithreaded systems, using model checking and linear temporal logic. To be able to verify, it is necessary to describe your model in the special language Promela (PROcess MEta LAnguage), set the verification variables that must change depending on the progress of the model execution, specify the verification objectives using the LTL formula associated with the verification variables. The main structural units for modeling are the channels, processes, data transferred and the LTL formula.

LTL [Ger95] is an extended condition in which, in addition to the classical operators, and, or, time operators are not added. Thus, it is possible to make a more extensive condition for testing the model, which will take into account different points in time. The complete list of operators is given below.

SPAN 1.6 - Protocol Verification : ShortVOTING.hlpsl

File

```
% OFMC
% Version of 2006/02/13
SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
PROTOCOL
  /home/span/span/testsuite/results/ShortVOTING.if
GOAL
  secrecy_of_nas
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.01s
  visitedNodes: 1 nodes
  depth: 1 plies
ATTACK TRACE
i -> (vs,3): start
(vs,3) -> i: {Nvs(1)}_kvvs,{Nasvs(1)}_kasvs
i -> (as,3): {Nasvs(1)}_kasvs
(as,3) -> i: {Nas(2)}_kvas
i -> (i,17): Nas(2)
i -> (i,17): Nas(2)
```

Figure 2: Avispa output

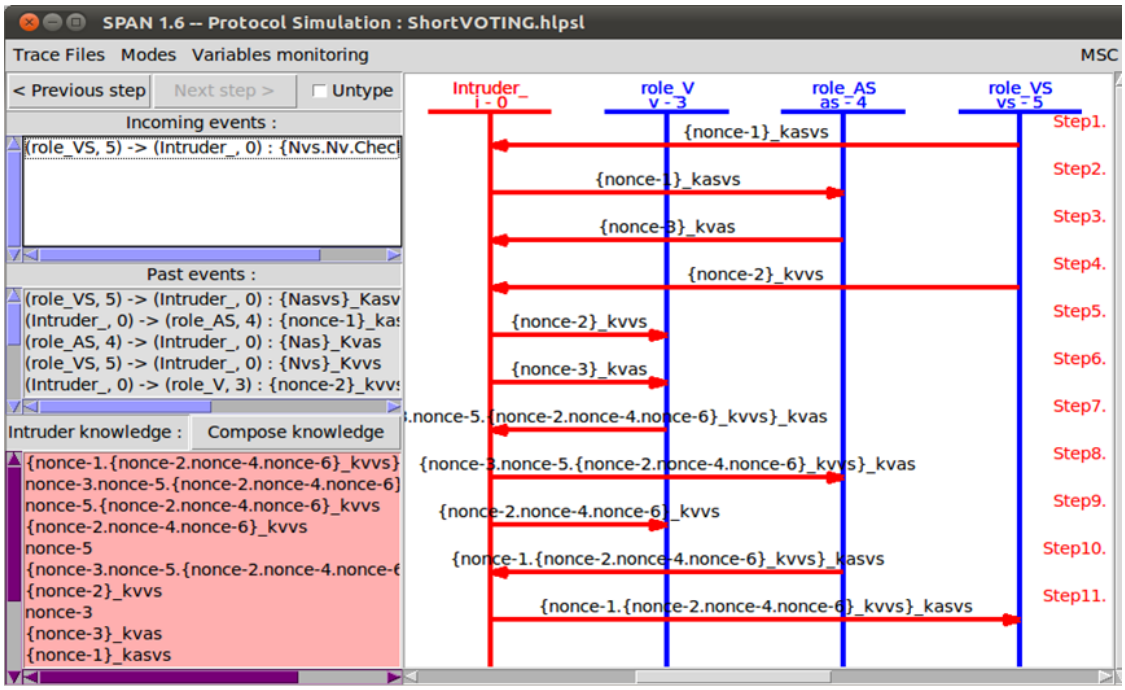

Figure 3: Interaction scheme in "intruder simulation" mode

Unary Operators (unop):
[] (the temporal operator always)
<> (the temporal operator eventually)
! (the boolean operator for negation)
Binary Operators (binop):
U (the temporal operator strong until)
W (the temporal operator weak until (only when used in inline formula)
V (the dual of U): (p V q) means !(!p U !q))
&& (the boolean operator for logical and)
|| (the boolean operator for logical or)
/\ (alternative form of &&)
\/ (alternative form of ||)
-> (the boolean operator for logical implication)
<-> (the boolean operator for logical equivalence)

Unlike other tools, SPIN has not been sharpened for security analysis of cryptographic protocols. There are no concepts of encryption, roles, knowledge of a certain role. Processes are the main units of interaction. It is processes that can be considered roles. Processes can interact through channels into which one process can put information, and the other can take. Thus, to verify the security of the protocols, you first need to model your protocol, the cryptographic functions used in it. Since this tool is not tied to any context of checks, it is possible to create your own more extensive checks. For example, you can check the values of HMAC, which in other tools is not possible, conduct an active attack on the substitution of data and look at the change in the course of the protocol. It is also possible to check various situations of non-public execution of the protocol, for example, if the message integrity is violated, check the session break or reconnect. A simplified e-voting protocol has been simulated to detect authentication attacks and verify that the connection is complete when an authentication random number is substituted. The description of the protocol in the Promela language is given below.

```
1 mtype = {Nas, Nb, Nvs, Nasvs, Nv, authData, checkData, NasBad, Vote};
2
3 #define Get1(x,y)    if \
4                      :: ((x!=Nas) || (y!=Nas)) -> gottenErr1=1 \
5                      :: else skip \
6                      fi
7
8 chan dyvvs = [0] of {mtype, mtype, mtype};                 /*V-VS-3*/
9 chan dyasvs = [0] of {mtype, mtype, mtype, mtype};        /*AS-VS-4*/
10 1chan dyvas = [0] of {mtype, mtype, mtype, mtype, mtype}; /*V-AS-5*/
11
12 bit gottenErr1 = 0;
13
14 proctype V (chan vas, vvs)
15 {
16     mtype Nas_, Nb_, Nvs_; /* Dont know */
17     mtype authData_, Vote_ /* Know */
18
19     authData_ = authData;
20     Vote_ = Vote;
21
22     vas ? Nas_, 0, 0, 0, 0;
23     vvs ? Nb_, Nvs_, 0;
24     vas ! Nas_, authData_, Nvs_, Vote_, 0;
25 }
26
27 proctype AS (chan vas, asvs)
```

```
28 {
29      mtype Nasvs_, authData_, Nvs_, Vote_, Nb_, NasRet_;/* Dont know */
30      mtype Nas_; /* Know */
31
32      Nas_ = Nas;
33
34      vas ! Nas_, 0, 0, 0, 0;
35      asvs ? Nasvs_, 0, 0, 0;
36
37      vas ? NasRet_, authData_, Nvs_, Vote_, 0;
38      atomic {
39      Get1(NasRet_, Nas_);
40      }
41
42      if
43      :: NasRet_ == Nas_ ->
44      asvs ! Nasvs_, Nvs_, Vote_, 0;
45      :: else
46      fi
47 }
48
49 proctype VS (chan vvs, asvs)
50 {
51      mtype Vote_, NvsRet_, NasvsRet_; /* Dont know */
52      mtype Nasvs_, Nvs_, Nb_; /* Know */
53
54      Nasvs_ = Nasvs;
55      Nvs_ = Nvs;
56      Nb_ = Nb;
57
58      vvs ! Nb_, Nvs_, 0;
59      asvs ! Nasvs_, 0, 0, 0;
60 }
```

The line (1) denotes the types of data used. In the lines (3) - (6), a macro is defined, which are located at the message transfer points to verify the correctness of the authentication. In the lines (8) - (10) the channels for interaction of the sides are indicated. 3 channels for interaction of V-VS, AS-VS, V-AS are defined. In line (12), the error flag is shown. Further, a process is defined that can be considered as roles for the client, the authentication server, and the voting server. In each of the processes, the original knowledge of the party, the order and the data itself transferred between the parties are described and a macro is also specified to verify the correctness of the returned random number Nas. Also, after the macros, the checks of the returned random numbers are indicated. If the verification fails, the protocol is terminated. The following describes an attacker who replaces the value of Nas to commit an attack on the authentication of the parties. In lines (74) - (82), the starting point init is specified, in which the processes are indicated. The line (84) specifies the macro for the LTL formula, which is the error flag. In lines (86) - (94), an LTL formula is specified to validate the model, which has already been converted to never claim. In order for verification to be successful it is necessary that never in the future the check flag take the value 1. After verification, if an error is detected according to the LTL formula, a corresponding message is displayed as in Figure 4. You can see the scheme of interaction between the parties, from which you can understand what went wrong. In this case, in Figure 5, you can see that there is 4 side - an attacker. He intercepted the message and changed the value of Nas to NasBad, which resulted in a violation of authentication. As can be seen, at the time of receiving the response message from the AS, the subsequent message was not sent. This is the correct case of a non-public execution of the protocol that was described in the protocol model. In this way, the SPIN tool potentially allows verification of any previously described verification objective by compiling appropriate verification rules. In this paper, an example of party authentication was given. The tool allows you to create your own model with your checks, which greatly increases the scale of the

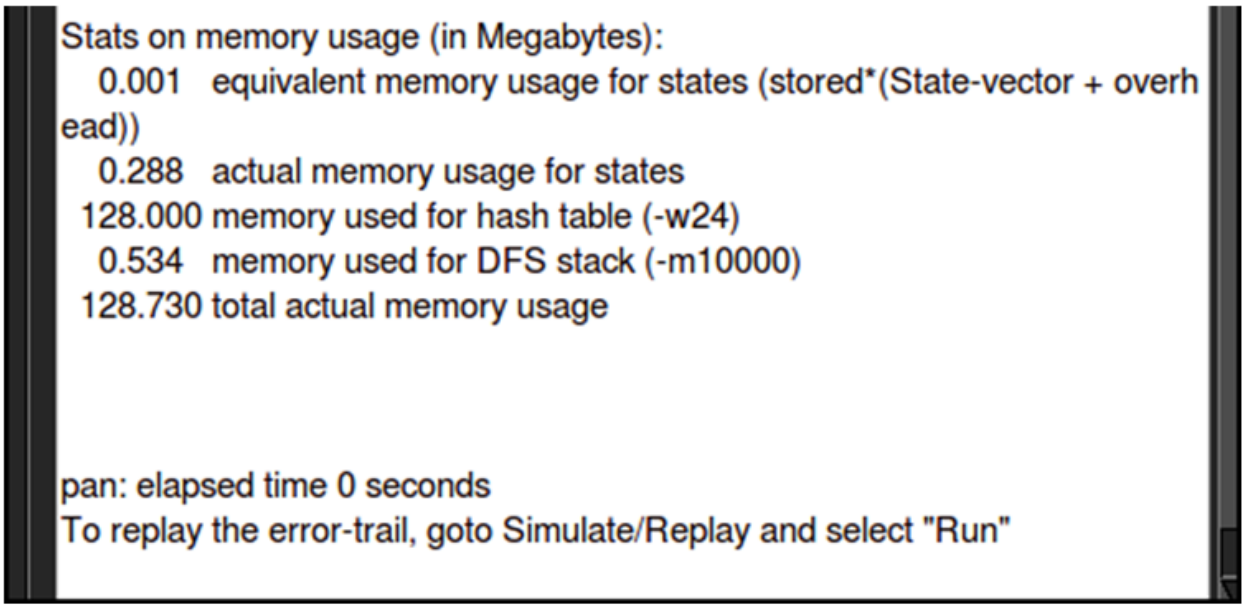

Figure 4: SPIN output

possible verification.

However, this is laborious process, since it is necessary to correctly create the model, including the model of the attacker, to identify the critical locations of the model in which it is necessary to verify certain requirements and compile an LTL formula according to which the model will be verified. Also, after verification, it is necessary to correctly interpret the graphical results to determine the errors. This tool can be used for the final analysis of the cryptographic protocol, allows to detect atypical errors according to the established requirements.

6 Conclusion

In the course of the work, the verifiers Avispa, SPIN were considered. Basic verification goals, such as party authentication, data confidentiality, integrity control, have been set. A simplified cryptographic protocol of electronic voting based on blind intermediaries is described. The means of verification of Avispa, SPIN are described. The analysis of the protocol's safety with the help of the given means of verification was carried out. It is established that for the primary and rapid analysis for the presence of attacks on authentication and secrecy, the Avispa tool is suitable. It allows you to simply describe the protocol in a special language and verify the cryptographic protocols. For the final analysis of the protocol, the SPIN tool can be used, with the help of which, according to the established checking rules and the created protocol model, it is possible to check more exact requirements, for example, the case of a correct completion of the protocol, after an attack by the intruder.

Acknowledgements

The work was supported by the Ministry of Education and Science of the Russian Federation, grant No 2.6264.2017/8.9.

References

[Avi06] The AVISPA team. The High Level Protocol Specification Language. *http://www.avispa-project.org/*, 2006.

[Ben08] M. Ben-Ari. Principles of Spin. *Springer*, Verlag, 2008.

[Dol83] D. Dolev, A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 1983. IT-29: 198–208, doi:10.1109/tit.1983.1056650

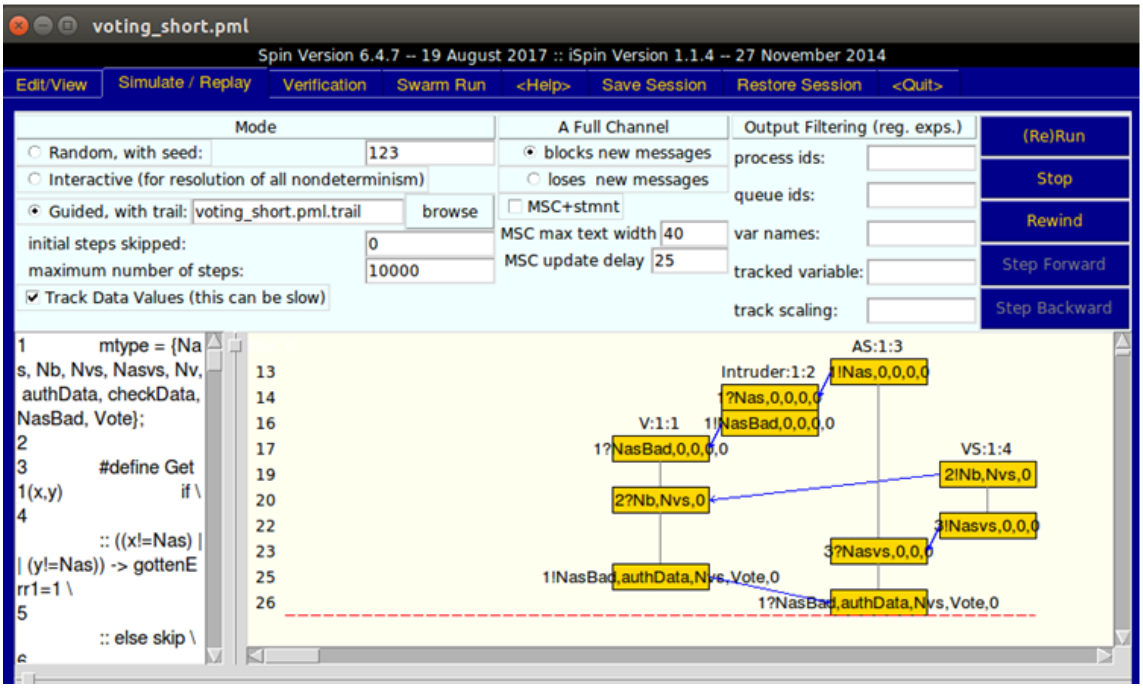

Figure 5: Interaction scheme in the presence of intruder

[Syv94] P. Syverson. A taxonomy of replay attacks [cryptographic protocols]. *In Computer Security Foundations Workshop VII*, 1994. CSFW 7. Proceedings (pp. 187-191). IEEE.

[Bab17] L. Babenko, I. Pisarev, O. Makarevich. A model of a secure electronic voting system based on blind intermediaries using russian cryptographic algorithms. *In Proceedings of the 10th International Conference on Security of Information and Networks*, (pp. 45-50). ACM. 2017.

[Sai11] R. Saillard, T. Genet. CAS+. March 21, 2011

[Ohe05] D. Von Oheimb. The high-level protocol specification language HLPSL developed in the EU project AVISPA. *In Proceedings of APPSEM 2005 workshop*, 2005. (pp. 1-17).

[Bas05] D. Basin, S. Mödersheim, L. Vigano. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3), 181-208, 2005.

[Tur06] M. Turuani. The CL-Atse protocol analyser. *In International Conference on Rewriting Techniques and Applications*, (pp. 277-286). Springer, Berlin, Heidelberg, 2006.

[Arm04] A. Armando, L. Compagna. SATMC: a SAT-based model checker for security protocols. *In European Workshop on Logics in Artificial Intelligence*, (pp. 730-733). Springer, Berlin, Heidelberg, 2004.

[Glo06] Y. Glouche, T. Genet, O. Heen, O. Courtay. A security protocol animator tool for AVISPA. *In ARTIST2 workshop on security specification and verification of embedded systems*, Pisa, 2006.

[Hol03] G. Holzmann. Spin model checker, the: primer and reference manual. *Addison-Wesley Professional*, 2003.

[Ger95] R. Gerth, D. Peled, M. Y. Vardi, P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. *In Protocol Specification, Testing and Verification XV*, (pp. 3-18). Springer, Boston, MA, 1995.