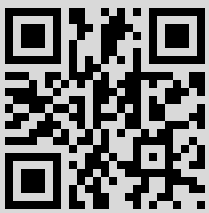
A. M. Semenov, Analysis of Russian key-agreement protocols using automated verification tools, *Mat. Vopr. Kriptogr.*, 2017, Volume 8, Issue 2, 131–142

DOI: https://doi.org/10.4213/mvk229



МАТЕМАТИЧЕСКИЕ ВОПРОСЫ КРИПТОГРАФИИ
2017 Т. 8 № 2 С. 131 – 142

УДК 519.719.2

Analysis of Russian key-agreement protocols using automated verification tools

A. M. Semenov

National Research University Higher School of Economics, Moscow

Получено 06.III.2016

Abstract. We study several Russian key-agreement cryptographic protocols for compliance with specified security properties in view of possible adoption of these protocols as standardized solutions intended to be used in the Russian Federation. We have used a number of automatic cryptographic protocol verification tools available in the Internet such as Proverif, AVISPA-SPAN and Scyther, to simulate examined protocols. We find a number of vulnerabilities and propose ways to fix them.

Keywords: cryptographic protocol, key-agreement cryptographic protocol, automated verification tool

Анализ российских протоколов выработки общего ключа с использованием средств автоматической верификации криптографических протоколов

А. М. Семенов

Национальный исследовательский университет «Высшая школа экономики», Москва

Аннотация. Работа посвящена изучению соответствия ряда российских криптографических протоколов определенному набору свойств безопасности. Анализ проводился в связи с возможной стандартизацией в Российской Федерации этих решений. С помощью доступных в сети Интернет программных средств автоматической верификации криптографических протоколов, таких как Proverif, AVISPA-SPAN и Scyther, проведен анализ указанных протоколов, найден ряд уязвимостей и предложены пути их исправления.

Ключевые слова: криптографический протокол, протокол выработки общего ключа, средства автоматической верификации криптографических протоколов

Citation: *Mathematical Aspects of Cryptography*, 2017, v. 8, № 2, pp. 131–142 (Russian)

1. Properties studied

We examine a class of key-agreement cryptographic protocols developed by Russian experts. These protocols satisfy the basic principles formulated in the report [9].

In our research we use the classification of the security properties introduced by IETF community [1]. In addition, this classification was used in research made by Russian experts [2 – 4, 11].

The key-agreement protocols were tested for compliance with the specified security properties: entity authentication (G1), message authentication (G2), source authentication (G5), key authentication (G7), key confirmation (G8), fresh key derivation (G10), secure capabilities negotiation (G11), sender invariance (G16) and secrecy (G12). Protocol analysis was conducted in accordance with Dolev–Yao intruder model [5]. Terminology is taken from [12] and [1].

Entity authentication (G1) — assuring one entity, through the presentation of evidence and/or credentials of the identity of a second entity involved in a protocol, that the second has actually participated during execution of the current run of the protocol. Usually entity authentication implies that some data can be unequivocally traced back to a certain entity, which implies Data Origin Authentication.

Message authentication (G2) — protocol provides means to ensure confidence that a received message or piece of data has been created by a certain party at some (typically unspecified) time in the past, and that this data has not been corrupted or tampered with, but guarantees of uniqueness or timeliness are not given.

Source authentication (G5) — legitimate group members will be able to authenticate the source and contents of the information or group communication.

Key authentication (G7) — one entity is assured that no other entity aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key.

Key confirmation (G8) — one entity is assured that a second possibly unidentified entity actually has possession of a particular secret key or of all keying material needed to calculate it.

Fresh key derivation (G10) — protocol uses dynamic key management in order to derive fresh session keys.

Secure capabilities negotiation (G11) — key-agreement protocol discovers the cryptographic capabilities and preferences of the entities and negotiates the security parameters, it is important to ensure that the announced capabilities and negotiated parameters have not been forged by an adversary.

Secrecy (G12) — particular data item or information is not made available or disclosed to unauthorized entities and remains unknown to the adversary.

Sender invariance (G16) — entity is assured that the source of the communication has remained the same as the one that started the communication, although the actual identity of the source is not important to the recipient.

Fulfilment of these security properties is checked for the following protocols: Echinacea-2 [11], Echinacea-3 [11], Limonnik-3 [9] (Lemongrass in English) and Crocus [10].

2. Automated Verification tools

In order to verify conformity with the security properties listed above we created mathematical models of key-agreement protocols using specification languages: Applied pi-calculus, High-Level Protocol Specification Language (HLPSL) and Security Protocol Description Language (SPDL). As a verification tools we used Proverif [6], AVISPA-SPAN [7] and Scyther [8] based on model checking and logical deduction methods.

Protocols for automated tool Proverif are specified using Applied pi-calculus language. This automated verification tool represents the protocol using Horn clause and analyzes protocols using the resolution method and upper approximation [6].

In a well known automatic verification tool AVISPA-SPAN protocol specification is carried out using High-Level Protocol Specification Language (HLPSL). During the study of security properties using AVISPA-SPAN we used modules OFMC and CL-AtSe. These modules carry out verification by model checking method. OFMC module uses symbolic method to group various states in endless classes, and CL-AtSe, which is based on construction constraints [2, 7].

Automated verification tool Scyther uses Security Protocol Description Language (SPDL) for specification of protocols. SPDL allows to define a set of states and transitions between states of the system. In the analysis of protocols Scyther uses symbolic analysis combined with a bi-directional search, based on partially ordered patterns [8].

3. Echinacea-2 analysis

Protocol Echinacea-2 was first proposed at RusCrypto conference in 2012 [11]. Protocol Echinacea-2 implements the key-agreement scheme with one-factor authentication using digital signature key.

Using the selected automatic verification tools, listed above, an analysis of Echinacea-2 protocol was conducted using the Dolev–Yao intruder model. Consider the developed model of Echinacea-2 protocol in specification language HLPSL for the verifier AVISPA-SPAN.

This protocol is designed to generate a shared key between two entities A and B. Entity B has a long-term digital signature key as defined under section 5.2 of GOST 34.10-2012 and a certificate $Cert_B$. Each of the entities during the initialization of protocol only stores its own identifier I_A and I_B respectively. The elliptic curve parameters agreed and known to both entities before the start of the protocol execution. Optional protocol parameters are omitted in the simulation.

Entity A initiates the protocol by sending to entity B a point K_A on elliptic curve E:

1. State = 0 /\ RCV(start)
=|> State' := 1 /\
K_a' := new() /\
Ka' := MUL(K_a'.P) /\
SND(Ia.Ka')

Entity B receives the message and checks whether $K_A \in E$ which is modeled by specifying a link between the random value K_A generated by entity A and elliptic curve parameter P:

10. State =10 /\ RCV(Ia'.MUL(K_a'.P))

After that entity B calculates point K_B of elliptic curve E, forms a common session keys K_{AB} and M_{AB}, calculates the authentication tag and key confirmation tag:

=|> State' := 20 /\
K_b' := new() /\
Kb' := MUL(K_b'.P) /\
Qab' := MUL(M.Q.K_b'.MUL(K_a'.P)) /\
Tab' := KDF(PI(Qab').Ia'.Ib) /\
Kab' := LOW(Tab') /\
Mab' := HIG(Tab') /\
AutB' := Sign(PI(Kb').PI(MUL(K_a'.P)).Ia')_inv(Yb) /\
TagB' := MAC(H2.PI(Kb').PI(MUL(K_a'.P)).Ib.Ia')_Mab' /\
CertificateB' := CERT(Ib.Yb) /\
SND(Ib.CertificateB'.Kb'.AutB'.TagB')$

Entity A checks the validity of the certificate $Cert_B$ of entity B, checks the digital signature Aut_B and key confirmation tag Tag_B, condition $K_B \in E$ is checked similarly to entity B:

2. State = 1 /\
RCV(Ib'.CertificateB'.MUL(K_b'.P).AutB'.TagB') /\
CertificateB' = CERT(Ib.Yb) /\
AutB' = Sign(PI(Kb').PI(Ka).Ia)_inv(Yb) /\
TagB'=MAC(H2.PI(MUL(K_b'.P)).PI(Ka).Ib'.Ia) \
_HIG(KDF(PI(MUL(M.Q.K_b'.Ka)).Ia.Ib'))=|>
State' := 2 /\
Qba' := MUL(M.Q.K_a.MUL(K_b'.P)) /\
Tba' := KDF(PI(Qba').Ia.Ib') /\
Kba' := LOW(Tba') /\
Mba' := HIG(Tba') /\
TagA' := H3.PI(Ka).PI(MUL(K_b'.P)).Ia.Ib'_Mba' /\
SND(TagA')

When entity A sends key confirmation tag Tag_A, entity B checks key confirmation tag Tag_A and the session terminates successfully. In case of success entities A and B compute a common key $K = K_{AB} = K_{BA}$. Fig. 1 shows the procedure of execution of Echinacea-2 protocol using verification tool AVISPA-SPAN.

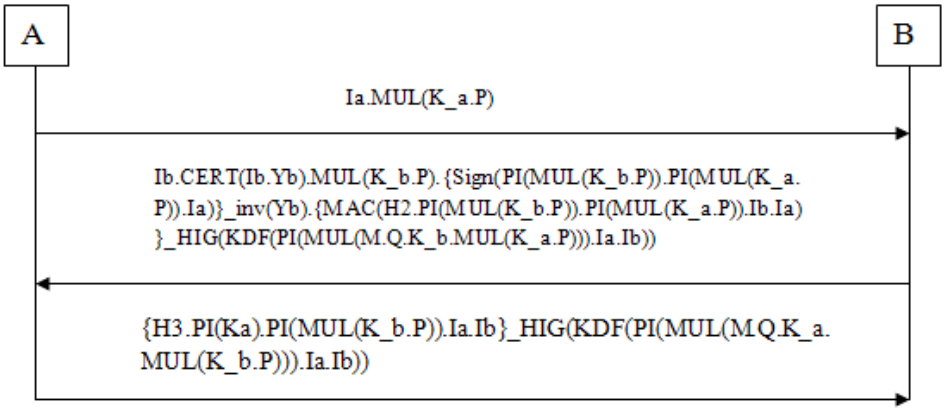


Fig. 1. Execution of Echinacea-2 protocol in AVISPA-SPAN

The adversary in this case knows all functions which are used in the protocol and parameters of the elliptic curve. AVISPA-SPAN showed that protocol is unsafe due to the fact that entity authentication may be conducted only for entity B.

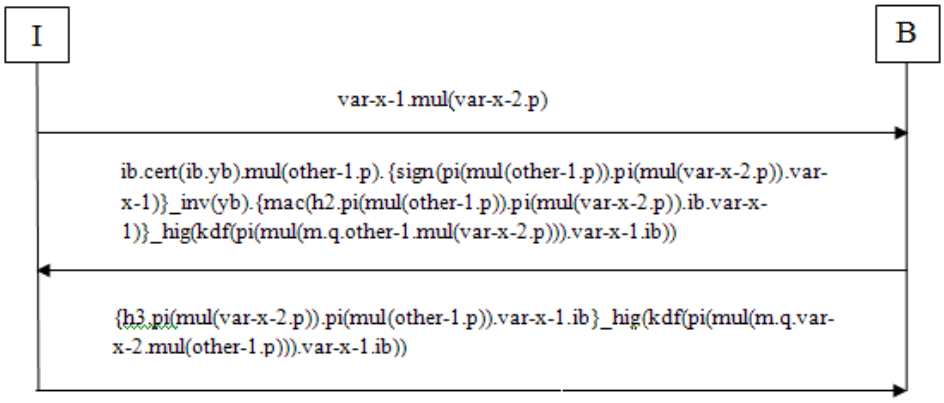


Fig. 2. AVISPA-SPAN results for Echinacea-2 protocol

The table below shows Scyther tool analysis results for Echinacea-2 protocol.

Entity	Claim	Status
A	Secret(Key)	OK
A	Alive (defined in [14])	OK
A	Weakagree (defined in [14])	OK
A	Nisynch (defined in [13])	OK
A	Niagree (defined in [13])	OK
B	Secret(Key)	OK
B	Alive (defined in [14])	FAIL
B	Weakagree (defined in [14])	FAIL
B	Nisynch (defined in [13])	FAIL
B	Niagree (defined in [13])	FAIL

The results of analysis of Echinacea-2 protocol model by the automated verification tools have not revealed a possibility that the adversary can find out the secret key, which is being generated during protocol execution. All verification tools demonstrated fulfillment of the properties G5, G7, G8, G10, G12, G16 for Echinacea-2 protocol.

4. Echinacea-3 analysis

Echinacea-3 protocol was first proposed at RusCrypto conference in 2012 [11]. The protocol implements the key-agreement scheme with mutual authentication using digital signature keys.

Echinacea-3 protocol, as opposed to the protocol Echinacea-2 uses mutual authentication, which prevents detection of violations of the formal properties. Both Echinacea-2 and Echinacea-3 protocols are vulnerable to denial of service attacks (DDoS attacks). Since the message transmitted in the first round of Echinacea-2 and Echinacea-3 protocols does not require confirmation of identifier of the sender these protocols are opposed to Limonnik-3 protocol, which requires the presence of the certificate, and Crocus protocol, where the digital signature is required.

We should also note the absence of a formal verification of the sender identifier at the first round of the Echinacea-2 and Echinacea-3 protocols. This allows the adversary to establish connections with legal entities. Such verification of entity identifier must be carried out while receiving messages and should explicitly be taken into account in protocols description and in its modeling. The presence of such defects can lead to a formal violation of the specific security properties.

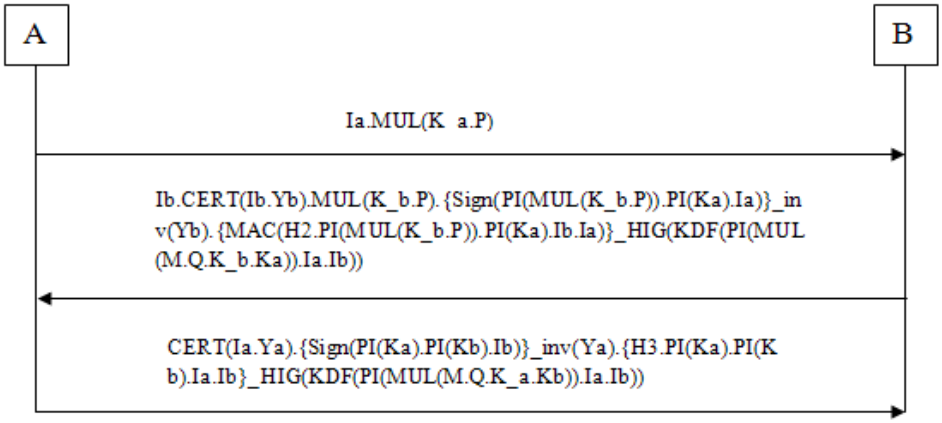


Fig. 3. Execution of Echinacea-3 protocol in AVISPA-SPAN

The table below shows Scyther tool analysis results for Echinacea-3 protocol.

Entity	**Claim**	**Status**
A	Secret(Key)	OK
A	Alive (defined in [14])	OK
A	Weakagree (defined in [14])	OK
A	Nisynch (defined in [13])	OK
A	Niagree (defined in [13])	OK
B	Secret(Key)	OK
B	Alive (defined in [14])	OK
B	Weakagree (defined in [14])	OK
B	Nisynch (defined in [13])	OK
B	Niagree (defined in [13])	OK

5. Limonnik-3 analysis

For the first time Limonnik-3 (Lemongrass in English) protocol was proposed at RusCrypto conference in 2011 [9]. This protocol implements the key-agreement scheme with the possibility of using two different elliptic curves, with two-factor authentication scheme based on the Diffie–Hellman key exchange. Limonnik-3 protocol scheme is shown at Fig. 4.

Entity authentication is performed at the 1st and the 2nd rounds of the protocol, by means of certificates previously issued to legal entities by certifying center.

The ability to validate the shared key is provided by checking the authentication tag by each of the entities of the Limonnik-3 protocol. Automated verification tools did not reveal opportunities for an adversary to obtain the shared secret key computed during the execution of the protocol.

Similarly to Echinacea-3 and Echinacea-2 protocols the shared secret key in Limonnik-3 is generated with secret information agreed before the start of the protocol, which is known to all legal entities, and key generated directly between two entities that interact in the protocol.

An adversary can impersonate a legal entity at the 1st round of the protocol using I_a and $Cert_A$ to initialize protocol execution, but still cannot compute the shared key with legal entity.

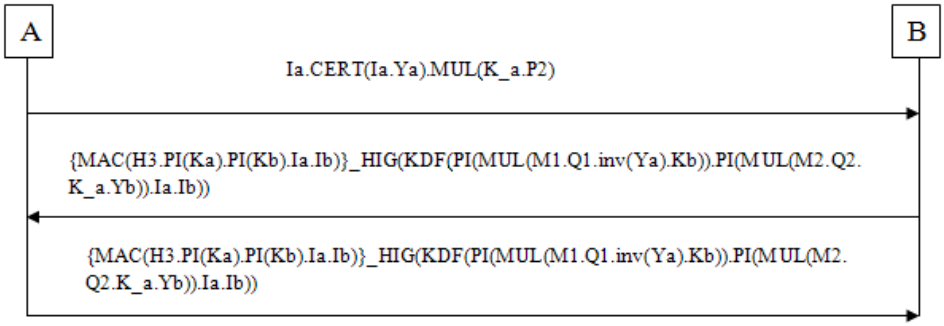


Fig. 4. Execution of Limonnik-3 protocol in AVISPA-SPAN

The table below shows Scyther tool analysis results for Limonnik-3 protocol.

Entity	**Claim**	**Status**
A	Secret(Key)	OK
A	Secret(Rand-ka)	OK
A	Secret(Rand-kb)	OK
A	Alive (defined in [14])	OK
A	Weakagree (defined in [14])	OK
A	Nisynch (defined in [13])	OK
A	Niagree (defined in [13])	OK
B	Secret(Key)	OK
B	Secret(Rand-ka)	OK
B	Secret(Rand-kb)	OK
B	Alive (defined in [14])	OK
B	Weakagree (defined in [14])	OK
B	Nisynch (defined in [13])	OK
B	Niagree (defined in [13])	OK

6. Crocus analysis

Crocus protocol was first proposed at the RusCrypto conference in 2012 [10]. This protocol is a public key-agreement protocol. It allows to implement secure exchange of request-reply messages where connection between the entities is set at the time of transmission of the request and the reply to the request.

Crocus protocol scheme is shown at Figure 5. The analysis of this protocol has shown that it allows to implement secure exchange of request-reply messages and generation of the shared encryption key between two legal entities.

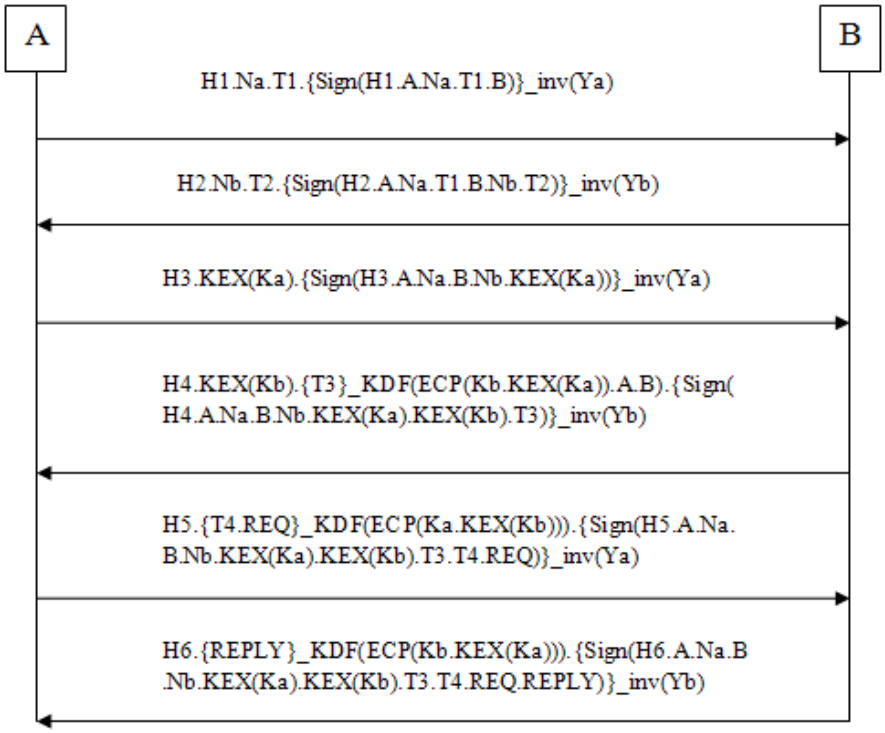


Fig. 5. Execution of Crocus protocol in AVISPA-SPAN

The results of analysis of Crocus protocol model by the automated verification tools revealed no possibility that the adversary can find out the secret key which is generated during protocol execution or decrypt the encrypted parts of messages, in particular the request and reply messages. Secret keys are generated at the start of a new session protocol.

Authentication of entities is provided through the use of digital signatures, according to the results of testing revealed that the adversary is not able to forge messages or to impersonate a legal entity.

The table below shows Scyther tool analysis results for Crocus protocol.

Entity	Claim	Status
A	Secret(REQUEST)	OK
A	Secret(REPLY)	OK
A	Secret(Rand-ka)	OK
A	Secret(Key)	OK
A	Alive (defined in [14])	OK
A	Weakagree (defined in [14])	OK
A	Nisynch (defined in [13])	OK
A	Niagree (defined in [13])	OK
B	Secret(REQUEST)	OK
B	Secret(REPLY)	OK
B	Secret(Rand-kb)	OK
B	Secret(Key)	OK
B	Alive (defined in [14])	OK
B	Weakagree (defined in [14])	OK
B	Nisynch (defined in [13])	OK
B	Niagree (defined in [13])	OK

7. Results

In the analysis of security properties according to the selected classification of IETF the property is considered as fulfilled if it holds for all messages sent at all rounds of the protocol. The property is considered as partially fulfilled if it holds for some of the messages and protocol rounds, not for all. In the table below

symbol “+” means that security property is fulfilled,

symbol “+/–"means that security property is partially fulfilled.

Property	Crocus	Echinacea-2	Echinacea-3	Limonnik-3
G1	+	+/–	+/–	+
G2	+	+/–	+/–	+/–
G5	+	+	+	+
G7	+	+	+	+
G8	+	+	+	+
G10	+	+	+	+
G11	+/–	+/–	+/–	+/–
G12	+	+	+	+
G16	+	+	+	+

Partial fulfillment of the security property G11 is due to the fact that the fulfillment of this property does not require the obvious check of quality of generating parameters. It may be checked only at the final implementation of a protocol.

Infringement of security property G1 for Echinacea-2 is associated with the protocol structure. As for Echinacea-3 the partial fulfillment of G1 may be fixed by using digital signature at the first round of the protocol.

Security property G2 fulfills partly in view of the absence of authentication in the first transmission of Echinacea-2, Echinacea-3 and Limonnik-3 protocols. Using of MAC for Echinacea-2 and digital signatures for Echinacea-3 and Limonnik-3 at the first round may solve this lack.

Proverif, AVISPA-SPAN and Scyther have not revealed a possibility for an adversary to find out the secret key or impersonate a legal entity for all protocol rounds and compute a common key with legal entity.

8. Conclusion

We show that the examined protocols satisfy the basic security requirements for cryptographic protocols, and may be considered as safe cryptographic solutions. The most efficient of considered protocols are Crocus and Limonnik-3.

Analysis of protocol models Echinacea-2 and Echinacea-3 revealed that these protocols require further modification to eliminate discovered issues. The results presented in this work may be useful in justifying the choice of one of the considered cryptographic protocols as standardized solutions.

References

[1] *Properties (Goals)*, http://www.avispa-project.org/delivs/6.1/d6-1/node3.html.

[2] Cheremushkin A. V., *Cryptographic protocols: basic properties and vulnerability*: Publ. centre "Akademija", 2009 (in Russian).

[3] Nesterenko A. Yu., "A new key agreement protocol based on Diffie–Hellman scheme", *Sistemy vysokoi dostupnosti*, **8**:2 (2012), 81–90 (in Russian).

[4] Nesterenko A. Yu., "On an approach to the construction of secure connections", *Mathematical aspects of cryptography*, **4**:2 (2013), 101–111.

[5] Dolev D., Yao A. C., "On the security of public key protocols", *IEEE Trans. Inf. Theory*, **29**:2 (1983), 198–208.

[6] Blanchet B., "An efficient cryptographic protocol verifier based on Prolog rules". In: "*Proc. 14th IEEE Computer Security Foundation Workshop (CSFW)*", 2009, 82–96.

[7] Armando A. et al., "The AVISPA tool for the automated validation of Internet security protocols and applications", *Lect. Notes Comput. Sci.*, **3576** (2005), 281–285.

[8] Cremers C. J. F., *Scyther — Semantics and Verification of Security Protocols*, Ph. D. Thesis, Eindhoven Univ. Technology, 2006.

[9] Matyukhin D. V., "On some properties of common key establishment schemes using infrastructure of public keys in a context of development of standardized cryptographic solutions" (2011) (in Russian), http://www.ruscrypto.ru/accotiation/archive/rc2011/.

[10] Nesterenko A. Yu., "On a protocol of common key computation" (2012) (in Russian), http://www.ruscrypto.ru/accotiation/archive/rc2012/.

[11] Grebnev S. V., "On the possibility of standardization of a key establishment protocol" (2014) (in Russian), http://www.ruscrypto.ru/accotiation/archive/rc2014/.

[12] Boyd C., Mathuria A., *Protocols for Authentication and Key Establishment*: Springer Science & Business Media, 2003.

[13] Cremers C., Mauw S., *Operational Semantics and Verification of Security Protocols. Information Security and Cryptography*, Heidelberg etc.: Springer, 2012.

[14] Lowe G., "A hierarchy of authentication specifications". In: "*Proc. 10th IEEE Computer Security Foundation Workshop (CSFW)*", Piscataway, CA: IEEE, 1997, 31 – 44.