

MAC Based Lightweight Protocols for Strong Authentication and Key Exchange*

DENIS TRČEK

Department of Digital Communications and Networks

Jožef Stefan Institute

Jamova 39, 1001 Ljubljana, Slovenia

E-mail: denis.trcek@ijs.si

Protocols that provide authentication and key distribution are mainly based on symmetric and asymmetric ciphers. In recent years, some approaches have been introduced that are based on strong one-way hash functions, and further enhancements to these approaches are given in this paper. The lightweight protocols that are presented in this paper are suitable for implementation with simple logic. They enable early recognition of attacks and make distributed session key generation possible. They are intended for use in environments with limited processing capabilities, where relatively short messages are being exchanged, e.g., agents environments.

Keywords: message authentication codes, one-way hash functions, lightweight protocols, authentication and key exchange, MBAKE schemes

1. INTRODUCTION

Authentication is one of the most important security services [1]. The first implementations of this service were based on the exchange of unprotected passwords. However, with the introduction of open system communications, cryptography became a general practice. Authentication that is based on cryptographic techniques is referred to as strong authentication. In 1981, L. Lamport introduced the idea of one-time passwords based on one-way hash functions [2]. More than a decade later, this idea was fully explored with S/Key implementation [3]. In the meantime, another interesting approach was introduced by Gong [4], who showed how to construct authentication and key exchange protocols, using pure, unadulterated hash functions. The next new dimension provided the use of one-way hash functions for constructing Message Authentication Codes or MACs [5, 6]. This work led to an approach, in which entity authentication and key exchange were implemented using MACs. The most notable example here is the KryptoKnight family of protocols (the extensive working document is given in [7], while a stable publication can be found in [8]).

This paper further explores the work described above. Firstly, the presented protocols are completely based on MACs. Secondly, these protocols make it possible for integrity attacks to be recognized as early as possible. This means that if an adversary

Received October 16, 2002; revised May 6 & August 1, 2004; accepted October 7, 2004.

Communicated by Randy Y. C. Chow.

* This work was partially sponsored by the Slovene Ministry of Information Society and Ministry of Education, Science and Sport of RS, contract no. 3311-03-828897.

tweaks some bits in a packet, an attack can be recognized immediately or at least at the end of the run of a protocol, and not later when a session key is deployed for data encryption. Thirdly, the protocols have been designed to retain resistance to most known kinds of attacks [9]. Fourthly, the above-mentioned predecessors enable only trusted entities to generate a session key. The protocols presented here make possible distributed generation of a session key – every entity that is capable of generating and eligible to generate a session key can do this. Fifthly, the protocols are designed with operation in real environments in mind, where entities have to manage multiple secure sessions at the same time. Finally, as one-way hash functions are assumed to be the basis for MACs, this enables efficient implementation. Protocols require simpler logic than those with symmetric or asymmetric algorithms, and fewer computing resources; thus, solutions are especially suitable for mobile devices, agent environments like intelligent agents [10], or CORBA middle-ware objects [11].

The paper is organized as follows. In the second section, a short overview of the field and some background is given. In the third section, the new protocols are presented. In the fourth section, a variation of these protocols is presented, and a conclusion is given in the fifth section. Finally, a formal proof is given in the Appendix.

2. ONE-WAY HASH FUNCTIONS, ENTITY AUTHENTICATION AND KEY EXCHANGE

MACs provide data authentication and integrity. They can be produced with symmetric or asymmetric ciphers, but nowadays they are most frequently produced with one-way hash functions (message authentication codes based on hash functions were introduced independently in [5] and [6]). The basic approach to constructing MACs is the concatenation of a secret key K and a message M . The whole sequence is then submitted as an input to a one-way hash function H . The output is a MAC, which is appended to the message when sent over the network.

Three basic MAC architectures were introduced in [6]. The first is the secret suffix technique, where the MAC is produced by appending a secret key to a message. Let comma denote concatenation throughout this paper. Then the whole sequence, sent over the network, is $[M, H(M, K)]$. Next is the secret prefix technique, where an MAC is produced by appending a message to a secret key; thus, the whole sequence, sent over the network, is $[M, H(K, M)]$. The third technique is the envelope method, where an MAC is produced by inserting the message between the keys; thus, the whole sequence, sent over the network, is $[M, H(K, M, K)]$.

These kinds of MACs soon gained widespread use and, consequently, became objects of intensive study. It soon became clear that the cryptographic strengths of the above techniques are not equal and in some cases may be insufficient. New MAC techniques have been proposed, e.g., in [12-15]. Some new MAC constructions are $H(K_1, H(K_2, M))$ and $H(K, H(K, M))$.

Nevertheless, whatever the construction of an MAC, the conversion of protocols presented here is straightforward. One just has to take into account the new formation rules for MACs. For reasons of clarity and simplicity, constructs introduced by [6] will be used, and the cryptographic strength of the MAC architecture will be taken for granted.

In addition, the following properties of strong one-way hash functions are assumed:

1. Pre-image resistance: for a given y , it is computationally infeasible to find such x that $f(x) = y$.
2. Second pre-image resistance: for a given x_1 , it is computationally infeasible to find x_2 with the same hashed value, i.e., $f(x_1) = f(x_2)$.
3. Collision resistance: it is computationally infeasible to find a pair (x_1, x_2) with the same hashed value, i.e., $f(x_1) = f(x_2)$.

Examples of one-way hash functions that currently meet these criteria are SHA-124 [16], and RIPEMD-160 [17].

3. PURE MAC BASED SCHEMES FOR AUTHENTICATION AND KEY EXCHANGE

The first two schemes presented in this paper will be based on time-stamps. As they can be based alternatively on nonces, this will be demonstrated for one of them in the next section. The schemes will be called MBAKE schemes (MAC based authentication with key exchange). The notation, used throughout this paper, will be as follows:

1. I denotes context identification, as explained below;
2. A , B , and C denote principals, where C is a privileged principal, trusted by both of the other parties;
3. K_A stands for a symmetric key, known to A and C (similarly, K_B stands for a symmetric key, known to B and C);
4. K_S is a random value that presents a symmetric session key shared between principals A and B ;
5. MAC denotes the output of a one-way hash function H with a key K and a message M , i.e., $H(K, M)$; note that the comma in this case only delimits arguments and does not mean the order of concatenation;
6. “_” stands for the NULL character;
7. \oplus stands for the XOR (exclusive-or) operation.

An important construct in some key-distribution protocols, used in [8] (and known in other authentication and key distribution systems, e.g., Kerberos [18]), is the so-called ticket. By definition, a ticket must include a distributed session key K_S , a key stamp (which might be a unique key version number, time-stamp, or nonce), and an identifier for each party intended to use the key.

It can be immediately concluded that a ticket, as known from the above mentioned reports, has to be modified if one wants to build a protocol solely using MACs. If every other kind of encryption has to be omitted, then session keys have to be excluded from tickets. Thus, keyless tickets are introduced in our architectures. They have the following syntax that explicitly reflects ticket semantics (bold symbols denote terminal symbols):

```

ticket          ::= context-id, sender, relay,
                  final-receiver, unique-id, validity
context-id      ::= message-seq, OID
message-seq     ::= 1 | 2 | 3 | 4
unique-id       ::= nonce | time-stamp

```

Context identification (context-id) serves to identify the protocol that is being used and the sequence number of the step within this protocol (more precisely, a sequence number when a packet is formed). For protocol identification, the use of object identifiers (OIDs) is proposed, which are defined in [19]. Such an explicit treatment is the basis for proper interpretation of messages and prevention of attacks based on the exploitation of modified context. Moreover, in a real environment, various kinds of protocols may be deployed. Enabling the possibility of their interleaving might lead to serious threats, like reflection attacks.

Regarding the rest of the fields, sender is the originating principal of the message, while relay is a principal that passes a particular packet or its part (possibly after processing) to the final receiver. If the value is assigned to a NULL character, then this means exclusion of intermediate receivers, i.e., relays (whenever possible without degradation of security, a NULL character should be used to keep the packet length short as possible). The final receiver is the intended recipient of the packet or its part. Time stamps or nonces assure freshness of the message, while validity defines the period of use of the session key. Validity represents the period in which a packet is considered to be acceptable for further processing, but for the sake of simplicity, validity will be omitted completely from the rest of this paper. This omission does not degrade the paper or the completeness of the work, but enables better presentation. Finally, the derivations of sender, relay, final receiver, validity, nonces, and time-stamps are obvious and, therefore, are not derived further in the above grammar.

Two kinds of packets are exchanged over the network: one with and one without a session key. In the packet without a session key, an MAC provides authentication and integrity. The freshness of the packet is provided by the time-stamp or nonces within the keyless ticket. Authentication is provided through the use of a shared secret key for production of the MAC.

The second kind of packet enables distribution of a session key. In this type of packet, the session key to be distributed is XORed with the MAC. This approach prevents disclosure of a session key to an attacker because applying a one-way hash function to a concatenation of a string and a secret key can be treated as a random generator with a shared secret as a seed. This is a known cryptographic assumption.

However, it has to be assured that both the MAC and the session key can be checked separately. The reason is straightforward. Although an attacker cannot calculate the key, he/she can modify a few bits in a key distribution packet. Two groups of variables are not known to the recipient in advance – the ticket and the session key. Therefore, additional data, packets, or protocol steps are needed to assure their integrity. If all goes well, a recipient can be assured of the freshness of the session key through the freshness of the keyless ticket. In addition, a recipient can be assured of authenticity and integrity of the session key through the MAC.

To summarize the basic principles and explicit meaning of the messages for the design of MBAKE based protocols, the following explanations are given (other details follow in the rest of this section):

- Authentication and integrity of packets is achieved through the inclusion of a shared secret and the use of MACs.
- Prevention of replays is achieved through the use of timestamps or nonces within tickets.
- Authentication of the exchanged session key K_S is achieved by XORing it with an MAC, where this very MAC must never be sent alone over the network.
- Integrity of the key K_S is also achieved with an additional new MAC, produced with this very session key (this additional MAC can be delivered through a new protocol step).
- Confidentiality of the shared secret key K_S in the additional new MAC is assured by the nature of a hash function, where K_S is included in its arguments.

Finally, some architectural issues concerning authentication and key exchange protocols have to be addressed. In this area, we distinguish so called push and pull scenarios. In push scenarios (e.g., Kerberos), if principal A wants to communicate with principal B , he/she must obtain a session key from principal C prior to communicating with B . In pull scenarios (e.g., in X9.17 [20]), A contacts B first, and then B gets the session key from C .

The scenarios presented in this paper are so-called relay scenarios, where the majority of messages originating from A and intended for B are relayed through C . This has an advantage in that C has the possibility of controlling the session set-up and can react accordingly.

3.1 MBAKE Relay Scenario I – Centralized Key Generation

The protocol with centralized key generation proceeds as follows:

1. The scenario starts with the sending of a packet from A to C . The packet is a keyless ticket with a corresponding MAC. After receiving this message, principal C can check the origin and integrity of the packet. From the ticket and its MAC, it concludes that principal A wants to communicate with principal B , and that the packet is fresh.
2. Principal C constructs new keyless tickets, along with appropriate MACs – one for principal B and another for principal A . Each of these MACs is XORed with a new random session key for communication between A and B .
3. B receives the packet. As the session key is a random value, an attacker can tweak a few bits in this packet. Thus, B cannot be assured of its integrity. He/she extracts the session key and forms a keyless ticket for principal A , where MAC is produced using the new session key. This packet is appended to the part of the packet received from C and intended for B , which B cannot process, as he/she does not know key K_A .
4. Both packets are finally sent to principal A , who extracts the session key from the packet, formed by C . A uses this key to check a keyless ticket, produced by B , using the session key. If all goes well, A knows that C has generated the session key, which is fresh. He/she knows also that B possesses the same session key.

The protocol can conditionally be completed at this point, but the problem is that *B* is still not assured of the integrity of the session key and does not know whether *A* has received a fresh and good session key.

If the context of the communication that follows is not known to both parties (e.g., the exchange of text messages which would result in meaningless messages if the session key has been attacked) then another optional step is needed. In this step, *A* generates a keyless ticket and produces its MAC with the newly received session key. It then sends it to principal *B* through principal *C*. *C* can check the session key and simply forward the packet to *B*. With this step, all parties are aware of the fact that *A* and *B* share the same session key, which is fresh and was generated by principal *C*. This scheme is shown in Fig. 1, where the dashed line denotes the optional step.

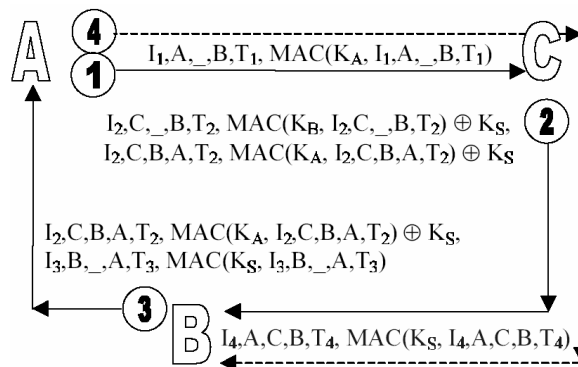


Fig. 1. Relay scenario I.

3.2 MBAKE Relay Scenario II – Distributed Key Generation

In relay scenario I, only the trusted center is eligible and able to generate and distribute the key. It is possible to modify this scenario in such a way that provides the possibility for key generation to principal *A* or *B*. The result is relay scenario II, which has three steps (see Fig. 2):

1. In the first step, principal *A* generates a keyless ticket, and the corresponding MAC is XORed with a random session key generated by principal *A*. Additionally, principal *A* forms a keyless ticket along with the corresponding MAC, using the proposed new session key. This packet is intended to be further sent to *B* through *C* so that all parties can be assured that *A* and *B* share the same session key, which is fresh and valid.
2. The whole packet is sent to principal *C*, who re-packs the first part of the packet by extracting the session key and producing a new keyless ticket, along with the corresponding MAC, which is XORed with the session key. Of course, the second part (which will be also relayed to principal *B*) serves to check the integrity of the received session key. Together with the first part of the packet, this assures authenticity and freshness of the session key.

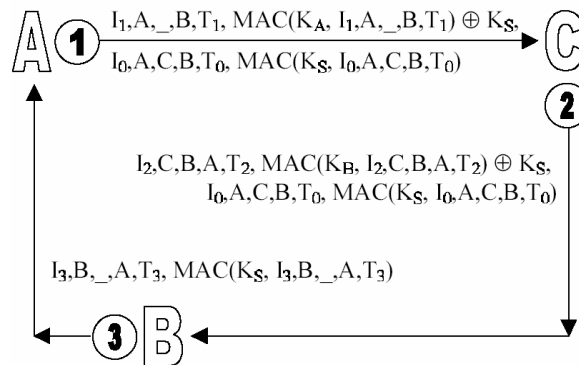


Fig. 2. Relay scenario II.

- When *B* receives the packet, he/she extracts the key from the first part of the packet and uses it to verify its integrity against the second part of the packet. If all goes well, *B* can be assured of the authenticity, integrity, and validity of the session key. Finally, *B* produces a keyless ticket for *A*, where an MAC is created by using a newly extracted session key.
- When *A* receives this packet, he/she can check the integrity of the key. If all goes well, *A* knows that *B* has a fresh and valid session key, known also to *C*.

At this point, the protocol is completed; no further steps are needed, even if the context of the communication that follows is not known to both parties, because the integrity of the key is assured (see also the proof in the Appendix).

4. NONCE BASED MBAKE

The MBAKE relay scenarios can be extended to obtain nonce-based schemes by adding additional messages. Simple replacement of time-stamps with nonces may be acceptable, but it is wiser to assure as much explicitness in the protocol as possible. Due to the simple replacement of time-stamps with nonces, the question of assurance of on-line communication arises. Put another way, it is wise to make sure that entities are communicating in synchronization, where they can start validity counters and narrow the gap between the creation of a packet and its further processing, thus reducing exposure to attacks. This synchronization will be demonstrated for relay scenario II, and the whole protocol will be divided into a synchronization round and a key delivery round.

The synchronization round proceeds as follows (see Fig. 3):

- Principal *A* forms a keyless ticket with nonces N_1 , N_2 , and N_3 , denoted together by N_{13} , and sends a packet to principal *C*. These nonces enable synchronization to assure freshness of the subsequently exchanged session key.
- Similarly, principal *C* repacks the data and forms a new packet with a keyless ticket, containing nonces N_2 and N_3 , denoted together by N_{23} , and a corresponding MAC.

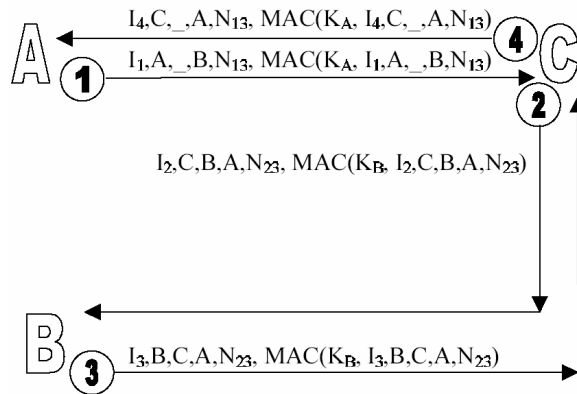


Fig. 3. Nonce based relay scenario II – synchronization phase.

3. In the third step, *B* generates a keyless ticket with the original nonce N_{23} , appends the corresponding MAC, and returns it to *C*.
4. In the fourth step, nonce N_{13} is delivered back to the originator, who can be assured that appropriate nonces are shared with *B* and *C*, which can ensure the timeliness of messages during the key exchange phase.

After this run, entities *A*, *B*, and *C* share the necessary nonces. In cases where four nonces are needed, $N_{1,3}$ should be replaced by $N_{1,4}$, and $N_{2,3}$ by $N_{2,4}$. These nonces are used in the key exchange phase that is presented in Fig. 4. The role of these nonces is the same as the role of time stamps in relay scenario II; therefore, the same explanation of the protocol applies.

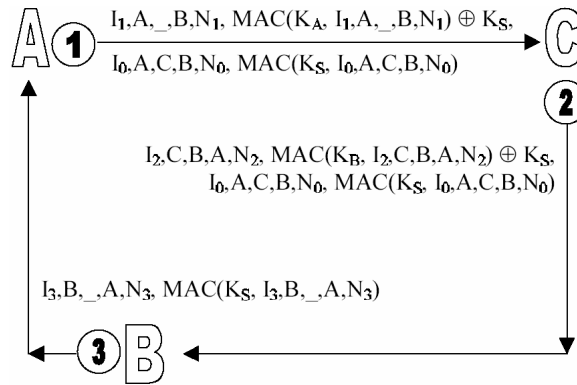


Fig. 4. Nonce based relay scenario II – key exchange phase.

It is worth emphasizing that despite its good quality of design, every cryptographic protocol is exposed to attacks if good practices are not followed, and the MBAKE family is no exception. Therefore, each entity has to

- use unique keyless tickets and keep a record of already used keyless tickets,
- check all elements of exchanged messages that can be checked in each step,
- have a quality time-base or random numbers generator to assure unique nonces/session keys, and
- generate random values that are large enough to make attacks infeasible.

A cautious reader might have noticed a bit varying connotations of some fields; examples are relay and final recipient fields in the second step of Fig. 1 and the second step of Fig. 3. In the first case, the complete second part of the packet is simply handed over to principal *A* through principal *B*, while in the second case, only a part of the second message (nonce N_{23}) is repacked and send back to *A* through *C*. Thus, based on the context-id, the connotation of the fields can be clearly comprehended, and that is why it is necessary that tickets be unique.

5. CONCLUSIONS

Using strong, one-way hash function based MACs, it is possible to build effective authentication and key exchange protocols that have many desired properties, inherited from one-way hash functions.

Because of the continuing need for lightweight authentication and key-exchange protocols, further research in this area has been done, and the protocols presented in this paper have been developed. These compact protocols, called MBAKE, are based exclusively on MACs. Therefore, advances in crypto research of MACs can be directly applied to these protocols. These protocols also make distributed session key generation and early detection of attacks possible.

Further work in this area will focus on MAC based protocols that can be linked to the public key infrastructure without introducing extensive processing requirements. This would make them suitable also for certain wireless applications, where current public key infrastructure protocols significantly limit the wider deployment of security services. Further work will also address the application of the presented protocols in micro-payments schemes.

REFERENCES

1. M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," Research Report 39, Digital Equipment Corporation Systems Research Centre, 1989.
2. L. Lamport, "Password authentication with insecure communication," *Communications of ACM*, Vol. 24, 1981, pp. 770-772.
3. N. M. Haller, "The S/Key one-time password system," in *Proceedings of ISOC Symposium on Networks and Distributed Systems Security (SNDSS)*, 1994, pp. 151-157.
4. L. Gong, "Using a one way functions for authentication," *ACM SIGCOMM Computer Communication Review*, Vol. 19, 1989, pp. 8-11.
5. J. M. Galvin, K. McCloghrie, and J. R. Davin, "Secure management of SNMP networks," in *Proceedings of IFIP TC6/WG 6.6 2nd International Symposium on*

- Integrated Network Management, II*, 1991, pp. 703-714.
6. G. Tsudik, "Message authentication with one-way hash functions," *ACM Computer Communications Review*, Vol. 22, 1992, pp. 29-38.
 7. P. Jansson, *et al.*, *KryptoKnight Protocol Cookbook*, Working Document, IBM Research, 1994.
 8. R. Bird, *et al.*, "The KryptoKnight family of light-weight protocols for authentication and key distribution," *IEEE Transactions on Networking*, Vol. 3, 1995, pp. 31-41.
 9. M. Abadi, R. M. Needham, "Prudent engineering practice for cryptographic protocols," *IEEE Transactions on Software Engineering*, Vol. 22, 1996, pp. 6-15.
 10. Foundation for Intelligent Physical Agents, FIPA Security SIG Request for Information, F-OUT-00065 Deliverable, Concord 2001.
 11. J. Siegel, *CORBA 3 Fundamentals and Programming*, 2nd edition, John Wiley & Sons, New York, 2000.
 12. B. Preneel and P. C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," in *Advances in Cryptology – CRYPTO '95*, LNCS 963, 1995, pp. 1-14.
 13. M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash function for message authentication," in *Advances in Cryptology – CRYPTO '96*, LNCS 1109, 1996, 1-15.
 14. S. Kent and R. Atkinson, IP Authentication Header, RFC 2402, IETF, 1998.
 15. S. Kent and R. Atkinson, IP Encapsulating Security Payload, RFC 2406, IETF, 1998.
 16. D. Eastlake and P. Jones, Secure Hash Algorithm – 1, RFC 3174 Standard, IETF, 2001.
 17. ISO/IEC, Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions, ISO 10118-3, 1998.
 18. B. C. Neumann and T. Ts'o, "Kerberos: an authentication service for computer networks," *IEEE Communication Magazine*, Vol. 32, 1994, pp. 33-38.
 19. ITU-T, Specification of Abstract Syntax Notation One (ASN.1), Recommendation X.208, 1988.
 20. American National Standard Institute, X9.17: Financial Institution Key Management, 1985.

6. APPENDIX

In this appendix, BAN logic will be used to prove the correctness of the protocol shown in Fig. 2. The reason for choosing this protocol is its simplicity with regard to the length and number of exchanged messages. Further, the protocol shown in Fig. 4 is a direct derivation of the above-mentioned protocol. Based on these examples, proving the properties of the first protocol should be straightforward.

In the following proofs, only context-ids, time stamps, and nonces will be used, while fields for the sender, relay, and final recipient will be omitted. This is done intentionally in order to simplify the expressions and make the proofs more readable. However, one should note that due to the uniqueness requirement, context-id already defines the rest of the fields and their sequence. Thus, in the synchronization phase, where nonces are used, their freshness can be determined correctly because of the above stated requirement.

6.1 Assumptions

A brief explanation of the following assumptions has to be given first. The assumptions about freshness imply an obligation to include and check timestamps. The assumptions about shared secret keys imply a requirement that these keys are somehow delivered secretly in advance. The assumptions about jurisdiction imply that a principal is fully trusted to determine a session key. Finally, it is understood that principal A believes that the session key, which he/she has produced, is intended for a secret channel between A and B :

$$\begin{aligned}
& C \models \#(T_1), B \models \#(T_2), A \models \#(T_3), \\
& C \models \#(T_0), B \models \#(T_0), \\
& C \models (A \mapsto A \xleftarrow{K} B), B \models (C \mapsto A \xleftarrow{K} B), \\
& C \models (C \xleftarrow{K_A} A), A \models (C \xleftarrow{K_A} A), \\
& C \models (C \xleftarrow{K_B} B), B \models (C \xleftarrow{K_B} B), \\
& A \models (A \xleftarrow{K_S} B).
\end{aligned}$$

6.2 Protocol Idealization

When BAN logic was invented, protocols like the ones described in this paper did not exist. Therefore, exact syntactical treatment of such protocols is a bit problematic. This has to be taken into account during the idealization process, which is a very subtle and sensitive step in BAN logic. Our reasoning for idealization will be based on the fact that strong one-way hash functions are cryptographic primitives. Basically, $\text{MAC}(K, X)$ will be denoted by a construct $\{X\}_K$, where K is a shared secret key for deriving an MAC. The same notation will be used in the XOR operation.

The idealized protocol is given below. The first halves of the first two messages enable authenticated session key exchange, while the second halves of these messages enable integrity checking, and this is reflected in the idealization. Despite the fact that the third message is syntactically analogous to the second parts of the first two messages, it conveys additional meaning. This is confirmation of possession of the shared secret key by the initiating principal:

Message 1

$$\{A \xleftarrow{K_S} B, I_1, T_1\}_{K_A}, \{I_0, T_0\}_{K_S}$$

Message 2

$$\{A \xleftarrow{K_S} B, I_2, T_2\}_{K_B}, \{I_0, T_0\}_{K_S}$$

Message 3

$$\{A \xleftarrow{K_S} B, I_3, T_3\}_{K_S}$$

6.3 Proof Derivation

In this proof, MMR will stand for message meaning rules, NVR for the nonce-verification rule, JR for the jurisdiction rule, CSR for composition and separation rules, CP for postulates about the visibility of components, and FP for postulates about the freshness of formulae (all these postulates can be found in [1]).

After the first step of the protocol, the following can be derived (using an assumption and CP):

$$\frac{C \models (C \xleftarrow{K_A} A), C \triangleleft \{A \xleftarrow{K_S} B, I_1, T_1\}_{K_A}}{C \triangleleft (A \xleftarrow{K_S} B, I_1, T_1)} \quad (1)$$

Application of MMR to the assumption and 1 results in

$$\frac{C \models (C \xleftarrow{K_A} A), C \triangleleft \{A \xleftarrow{K_S} B, I_1, T_1\}_{K_A}}{C \models A \vdash (A \xleftarrow{K_S} B, I_1, T_1)} \quad (2)$$

Beliefs 1 and 2 are acceptable, because C has verified the integrity of session key K_S using the second part of the received message. Therefore, principal C believes that K_S is indeed the session key that has been sent by $C A$.

Using an assumption and applying FP, one can further conclude that

$$\frac{C \models \#(T_1)}{C \models \#(A \xleftarrow{K_S} B, I_1, T_1)} \quad (3)$$

Using CSR, we can write the following:

$$\frac{C \models \#(A \xleftarrow{K_S} B), C \models A \vdash (A \xleftarrow{K_S} B)}{C \models A \models (A \xleftarrow{K_S} B)} \quad (4)$$

Now, using JR and the above result, we can state the following:

$$\frac{C \models (A \mid\Rightarrow (A \xleftarrow{K_S} B)), C \models A \models (A \xleftarrow{K_S} B)}{C \models (A \xleftarrow{K_S} B)} \quad (5)$$

After the exchange of the second message and using reasoning analogous to that in 1, 2, 3 and 4, we can write

$$\frac{B \models (C \mid\Rightarrow (A \xleftarrow{K_S} B)), B \models C \models (A \xleftarrow{K_S} B)}{B \models (A \xleftarrow{K_S} B)} \quad (6)$$

Note that the assumption used above is acceptable, because principal C would not become involved in the second step of the protocol, if belief 5 could not be obtained.

After the third step, the following statements hold true:

$$\frac{A \models (A \xleftarrow{K_S} B), A \triangleleft \{A \xleftarrow{K_S} B, I_3, T_3\}_{K_S}}{A \triangleleft (A \xleftarrow{K_S} B, I_3, T_3)} \quad (7)$$

$$\frac{A \models (A \xleftrightarrow{K_S} B), A \triangleleft \{A \xleftrightarrow{K_S} B, I_3, T_3\}_{K_S}}{A \models B \vdash (A \xleftrightarrow{K_S} B, I_3, T_3)} \quad (8)$$

Using 7, an assumption, and FP, we obtain

$$\frac{A \models \#(T_3)}{A \models \#(A \xleftarrow{K_S} B, I_3, T_3)} \quad (9)$$

Applying CSR and NVR to 8 and 9, the final belief follows:

$$\frac{A \models \#(A \xleftrightarrow{K_S} B), A \models B \vdash (A \xleftrightarrow{K_S} B)}{A \models B \models (A \xleftrightarrow{K_S} B)} \quad (10)$$

Principal A believes that principal B believes that K_S is the shared secret key between these two principals; thus, the goal of authentication is fulfilled. \square



Denis Trček received his Ph.D. from the University of Ljubljana. His area of research and interest include information systems and electronic business with emphasis on security. He has taken part in various European research projects (NetLINK CEE, COST projects, etc.). He has been involved in application-oriented projects for e.g. Slovene National gallery (implementation of IS), internet banking services for the biggest Slovene bank Nova ljubljanska banka, and introduction of smart cards into the health sector (a nationwide project). His bibliography includes over 70 titles, including international journals with SCI impact factors and monographs. He was an invited speaker at international security events, e.g. PKI Invitational Workshop, organized by US Vice President Al Gore's Committee for Security on Information Superhighway, NIST and MITRE Corp. in September 1995, Washington DC. He is also a member of program committees of international conferences, serves on editorial board of ACTA International Journal of Computers & Applications, and is a member of SDS and IEEE.