Security Protocol Design: A Case Study Using Key Distribution Protocols

Anca Jurcut[*], Tom Coffey[*], Reiner Dojen[*], Robert Gyorodi[**]

[*] Department of Electronic & Computer Engineering,
University of Limerick,
Ireland.
E-Mail: anca jurcut @ul.ie, tom.coffey@ul.ie, reiner.dojen@ul.ie

[**]Department of Computer Science,
University of Oradea,
Romania
E-Mail: rgyorodi@uoradea.ro

***Abstract* – Nowadays security protocols are a key component in providing security services for fixed and mobile networks. These services include data confidentiality, radio link encryption, message integrity, mobile subscriber authentication, electronic payment, certified e-mail, contract signing and non-repudiation.**

This paper is concerned with design of effective security protocols. Security protocols are introduced and some common attacks against security protocols are discussed. The vulnerabilities that lead to the attacks are analyzed and guidelines for effective security protocol design are proposed. The presented guidelines are applied to the Andrew Secure RPC protocol and its adapted versions. It is demonstrated that compliance with the guidelines successfully avoids freshness and parallel session attacks.

Keywords: ***Security protocols, protocol weaknesses, protocol attacks, freshness attacks, parallel session attacks, design guidelines for security protocols***

I. INTRODUCTION

With the ever increasing use of distributed applications for meeting customer demands, there is a commensurate increase in the reliance of electronic communication over networks. Since many of these applications require the exchange of highly sensitive information over computer networks, the need for security protocols to ensure the protection and integrity of data is critical.

Many published security protocols have been subsequently been found to contain security weaknesses [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], which can be exploited in an attack on the protocol.

In this paper we introduce security protocols and the most common attacks that exploit weaknesses in the design of security protocols. As an example, the Andrew Secure RPC key distribution protocol [12], its adapted versions and two published attacks against these protocols: a freshness attack by Burrows et al. [5] and a parallel session attack by Lowe [7], respectively are discussed. The reasons for freshness and parallel session attacks to succeed and the vulnerabilities which lead to these attacks are analyzed. Additionally, guidelines for security protocol design, in order to prevent freshness and parallel session attacks are proposed. Furthermore, the proposed guidelines are applied to the Andrew Secure RPC protocol and its adapted versions, in order to avoid the two types of attack.

II. SECURITY PROTOCOLS

A security protocol is a communication protocol that is based on a cryptographic system. A protocol is a prescribed sequence of interactions between communicating parties designed to achieve certain goals. These parties may be users, hosts, mobile devices or processes and they are referred to as principals. An honest principal follows a particular communication protocol, while a dishonest principal (or attacker, intruder, spy, enemy, adversary, etc.) tries to manipulate the protocol to achieve an unfair advantage.

The main difficulty in the development of effective security protocols is to address the vast possibilities of an adversary to gain information. In contrast to communications protocols, where the main issues are reachability of all legal states and avoidance of infinite loops, security protocol verification deals with the gain of information by an adversary.

The adversary can be either passive (just listening to communication) or active (modifying message content, message order, dropping messages, etc). An active adversary is more dangerous than a passive eavesdropper. Further, an adversary might use

simultaneous protocol runs to gain information. One has to keep in mind that an adversary will not play fair by any rules. Hence, a security protocol should attempt to achieve its goals under all possible circumstances.

Basic cryptographic protocols allow agents to authenticate each other, to establish fresh session keys for confidential communication, and to ensure the authenticity of data and services. Building on such basic cryptographic protocols, more advanced services like non-repudiation, fairness, electronic payment, and electronic contract signing are achieved.

III. ATTACKS ON SECURITY PROTOCOLS

An attack on a protocol is a sequence of actions performed by an attacker, by means of any hardware or software tool, in order to subvert protocol goals. The attacker is a dishonest participant (also called 'adversary' or 'intruder') that may be a member of the system, i.e., a legitimate user, or external to the system.

A security protocol should enforce the data exchange between honest participants, while the dishonest ones should be denied any benefit of it. However, security protocols can contain weaknesses which exploit the conceptual structure of the protocols and are independent on the cryptographic systems. These weaknesses make them vulnerable to a range of attacks such as freshness, parallel session and type-flaw attacks.

A *freshness attack* is one of the most common attacks on authentication and key-establishment protocols. If the messages exchanged in an authentication protocol do not carry appropriate freshness identifiers, then an intruder can get himself authenticated by replaying messages copied from a legitimate authentication session. The intruder, according to the Dolev Yao model [13], has complete control over the communication channels. The intruder knows the session keys and short-time secrets generated in previous runs of the protocol. In a system in which sequential runs of the protocols are performed it is assumed that the intruder has the capacity to replay messages from an old session run of the protocol that might contain a possibly compromised session key. Examples of freshness attacks on security protocols include [1, 5, 8]. Freshness attacks can be foiled by the use of nonces, run identifiers and timestamps.

A *parallel session attack* requires the parallel execution of multiple protocol runs, where the intruder uses messages from one session (*the reference session*) to synthesize messages in the other session (*the attack session*). Examples of parallel session attacks on security protocols include [1, 2, 3, 7, 8, 10]. There are several forms of parallel session attacks such as *oracle attacks*, *man-in-the-middle attacks or multiplicity attacks.* In an *oracle attack* the intruder starts a new run of the protocol and uses one of the principals as an oracle. Thus, the intruder has the appropriate answers for the challenges in the main run of the protocol. *A man-in-the-middle attack* occurs when two principals believe they are mutually authenticated, when in fact the intruder masquerades as one principal in one session and as the other principal in another. There are two runs of the protocol, but the honest principals think that there is only one. *A multiplicity attack* is a parallel session attack that can occur when the principals disagree on the number of runs they think they have successfully completed with each other.

A type flaw attack involves the replacement of a message component with another message of a different type by the intruder (e.g. accepts a nonce as a key). Examples of type flaw attack on security protocols include [7, 9, 15].

IV. CASE STUDY USING KEY DISTRIBUTION PROTOCOLS

In this section we present the Andrew Secure RPC key distribution protocol [12], its adapted versions and two published attacks against these protocols by Burrows et al. [5] and Lowe [7], respectively.

A. The Andrew Secure RPC protocol and the freshness attack

The Andrew Secure RPC protocol was proposed by M. Satyanarayanan [12] for the distribution of a fresh shared key between two principals, who already share a key and to perform an authentication handshake (cf. Figure 1).

1. **A → B : A, {Na}Kab**
2. **B → A : {succNa, Nb}Kab**
3. **A → B : {succNb}Kab**
4. **B → A : {K'ab, N'b}Kab**

Figure 1. The Andrew Secure RPC protocol

Table 1 details the notation used for protocol descriptions throughout this paper.

Table 1. Notation used for protocol descriptions

Notation	Description
A, B	Legitimate principals
I(A) / I(B)	Intruder masquerading as principal A / B
Kab, K'ab	Symmetric key shared between A and B
Na, Nb, N'b	Nonce (once off random number) generated by A and B
succNa, succNb	Successor to nonce
{m}k	Cryptographic expression - message m encrypted with key k
i, ii,…	Denotes the run of the protocol

The protocol has two independent components. In the first three messages, principals A and B perform a handshake using a key they already share, Kab. The handshake is intended to allow principal A to obtain a new session key, K'ab from principal B. Principal A initiates the protocol by sending a request to B. This request consists of A's identity and a nonce, Na, encrypted with their shared key Kab. In the second message B replays to A's challenge, by sending to A, a successor to its nonce along with a new nonce, Nb, generated by B. This message is also encrypted with the shared key Kab. In the third message A responses to B's challenge by sending the successor to *B's* nonce Nb encrypted with their shared key Kab. In the fourth and final message, B sends the new session key K'ab to A along with a new generated nonce N'b, all encrypted with the key Kab. The nonce N'b sent in message 4 is to be used in the further communication.

Burrows et al. have analyzed this protocol using the BAN Logic and claimed a freshness attack on this protocol [5], which exploits the fact that in message 4 there is nothing that principal A believes to be fresh. An intruder could replay an old message as the last message in the protocol and could convince A to use an old, possibly compromised session key. In other words, an intruder, denoted by I, might find an old session key and might replay the fourth message in which that key was established. Thus, the intruder can impersonate B. The details of the attack are given in Figure 2.

i.1. A → B : A, {Na}Kab
i.2. B → A : {succNa, Nb}Kab
i.3. A → B : {succNb}Kab
i.4. B → A : {K'ab, N'b}Kab
ii.1. A → B : A, {Ma}Kab
ii.2. B → A : {succMa, Mb}Kab
ii.3. A → B : {succMb}Kab
ii.4. B → I(A) : {K''ab, M'b}Kab
ii.4. I(B) → A : {K'ab, N'b}Kab

Figure 2. Freshness Attack on Andrew Secure RPC

B. The adapted versions of the Andrew Secure RPC protocol and the parallel session attack

Burrows et al. modified the Andrew Secure RPC protocol [5], by adding the nonce Na to the cryptographic expression in message 4, to prevent the freshness attack detailed above. This version of the protocol is known as the BAN modified Andrew Secure RPC protocol and the steps are given in Figure 3.

1. **A → B : A, {Na}Kab**
2. **B → A : {succNa, Nb}Kab**
3. **A → B : {succNb}Kab**
4. **B → A : {K'ab, N'b, Na}Kab**

Figure 3. BAN Modified Andrew Secure RPC Protocol

The concrete version of the Andrew Secure RPC proposed by Burrows et. al [5] is an optimized version of the modified version of the Andrew Secure RPC protocol. The concrete version needs less encryption. The messages exchanged by the principals are the given in Figure 4.

1. **A → B : A, Na**
2. **B → A : {Na, K'ab}Kab**
3. **A → B : {Na}K'ab**
4. **B → A : Nb**

Figure 4. BAN Concrete Andrew Secure RPC Protocol

In the BAN Concrete Andrew Secure RPC protocol, if A wants to establish a new key with B, A sends its identity and a nonce Na to B (message 1). B responds by choosing a new key K'ab, and returning it to A, along with the nonce, encrypted with the existing key (message 2). A decrypts this message to learn the new key, and then sends a message back to B, encrypted with the new key, to demonstrate knowledge of the new key to B (message 3). Finally, B returns a new nonce Nb to A, to confirm that A is free to proceed (message 4).

This protocol is subject to a parallel session attack [7] as outlined in Figure 5. The intruder I will start a second session with principal A, where I masquerades as principal B trying to engage in a session with A. In this second session, the intruder uses principal A as an oracle to encrypt messages required in the first session. The intruder replays A's nonce (from the first run) along with B's identity to A. In the next step, A sends to B (masqueraded by the intruder) a new session key and the nonce received in the previous step. This message is encrypted with their shared key. The intruder simply replays this message to A in the first run. The third message from the first run is replayed by the intruder in the second run. In the final message the principals simply exchange nonces. At the end of the attack A is convinced that to have engaged in two protocol runs with B: In one run A has updated the session key and in the other B has updated the session key. However, B is absent in both protocol runs.

i.1. A → I(B) : A, Na
ii.1. I(B) → A : B, Na
ii.2. A → I(B) : {Na, K'ab}Kab
i.2. I(B) → A : {Na, K'ab}Kab
i.3. A → I(B) : {Na}K'ab
ii.3. I(B) → A : {Na}K'ab
i.4. I(B) → A : Ni
ii.4. A → I(B) : Nb

Figure 5. The attack on BAN concrete Andrew Secure RPC protocol

In order to prevent the above described attack, Lowe [7] modified the BAN concrete Andrew Secure RPC protocol by adding the identity of the sender B in

message 2 as detailed in Figure 6. This will prevent the attack, as message ii.2 will include the identity A, so the intruder will be unable to replay the message as message i.2, where the identity B would be required. The adapted version of the protocol is known as the Lowe modified BAN concrete Andrew Secure RPC protocol.

1. **A → B : A, Na**
2. **B → A : {Na, K'ab, B}Kab**
3. **A → B : {Na}K'ab**
4. **B → A : Nb**

Figure 6. Lowe Modified BAN Concrete Andrew Secure RPC Protocol

The outlined parallel session attack can also be mounted on the Andrew Secure RPC protocol and the Ban modified Andrew Secure RPC protocol. Both protocols can be corrected by adopting the same solution as Lowe proposed in the case of BAN concrete Andrew Secure RPC protocol: the addition of the identity of the sender B in message 2.

V. GUIDELINES FOR SECURITY PROTOCOL DESIGN

In this section we present a set of guidelines for security protocol design. Adopting these design guidelines will result in security protocols that are invulnerable to freshness and parallel session attacks. These guidelines can also be used to identify weaknesses in existing protocols.

A. Preventing Freshness Attacks

Consider two principals A and B, involved in a protocol P. In an arbitrary step of the protocol P, A sends a message m containing a cryptographic expression to B. If this cryptographic expression does not have at least one component fresh for the recipient, then message m can be replayed. An intruder (I) can mislead the recipient (B) to accept old and possibly compromised data as a component of the cryptographic expression from message m. Further, if message m is used for authentication, then the intruder can get authenticated by substituting a previously recorded message (from A to B) for m.

Since the cryptographic expression from that message does not contain any component which the recipient B recognizes as being fresh, B cannot detect the message replay.

a) Examples of freshness attacks

Consider the Andrew Secure RPC protocol, as outlined in Figure 1. The message transmitted in step 4 of this protocol consist of a cryptographic expression ({K'ab,N'b}Kab) encrypted with a symmetric key. This cryptographic expression does not contain any fresh component for the recipient (principal A), since N'b is a new nonce generated by the sender (principal B) in step 4 of the protocol. Therefore principal A has no assurance that key K'ab is fresh. Hence, an intruder can substitute a previously recorded message 4 (from B to A) and mislead principal A to accept an old and possibly compromised session key.

Similar flaws have been the cause of freshness attacks on other security protocols such as an attack on the Kao-Chow v1.protocol, claimed by Dojen, Lasc and Coffey in 2008 [11]; the attack on the Needham Schroeder symmetric key protocol, claimed by Denning and G. Sacco [14] and the attack on the Neumann Stubblebine protocol, claimed by Hwang et al. [15].

b) Guidelines for preventing freshness attacks

In order to avoid freshness attacks, one of the following guidelines for security protocols design should be respected:

- Each cryptographic expression of a protocol should contain a nonce generated by the recipient in a previous step of the protocol.
- Under the assumption of synchronized clocks, each cryptographic expression of a protocol should contain a timestamp.

B. Preventing parallel session attacks

Consider two principals A and B, involved in a challenge-response handshake of a protocol P. A *challenge-response* handshake of a protocol is a process where fresh data (random nonces) is generated by one of the principals (here we assume principal A), acting as a verifier, and then delivered as part of a cryptographic expression to the other principal (B), acting as a prover.

If none of the cryptographic expressions belonging to the challenge-response handshake of the protocol P contain appropriate identities of principals, then it is not possible to establish the source of messages transmitted. This kind of weakness can be exploited by a parallel session attack that enables an intruder to masquerade as a legitimate principal. As the source of the cryptographic expression cannot be established an intruder can use the cryptographic expression from a parallel run to deceive the recipient in the original protocol run.

b) Examples of parallel session attacks

Considering the Ban concrete Andrew Secure RPC protocol as detailed in Figure 4. One of the challenge-response handshakes of this protocol consist of the following two messages:

1. A → B : A, Na
2. B → A : {Na, K'ab}Kab

The message transmitted in step 2 of this challenge-response of the protocol, consists of a cryptographic expression encrypted with a symmetric key. This cryptographic expression does not contain an identity of any of the principals (A or B). Therefore, the parallel session attack as outlined in Figure 5 succeeds, enabling an intruder to masquerade as a legitimate principal. The weakness exploited by this attack is the fact that it is impossible to establish the source of message 2 of the protocol. Thus, it is impossible to tell whether this message is intended to be sent by A to B or vice versa.

The same weakness exists in the Andrew Secure RPC and the Ban modified Andrew Secure RPC protocols and can be exploited by corresponding parallel session attack.

Similar flaws have been the cause of numerous attacks on other protocols, for example: the attack on the Lowe's modified version of the Wide-Mouthed Frog protocol discovered by Dojen, Jurcut, Coffey and Gyorodi [10]; the attack on KSL protocol, claimed by Lowe [7]; an attack on the Neumann Stubblebine protocol, claimed by Hwang et al. [15], two attacks on the SPLICE protocol [16], [17]; the attack on the Needham-Shroeder public key protocol, claimed by Lowe [6]; the attack on CCITT X.509(3) protocol, discovered by Burrows et al. [5].

b) Guidelines for preventing parallel session attacks

In order to avoid parallel session attacks in challenge-response handshakes, the following guidelines for the design of security protocols are proposed:

- Case 1 - a symmetric key is used to encrypt the messages: At least one of the cryptographic expressions, belonging to a challenge-response handshake of a protocol should contain the *sender identity*.
- Case 2 – a public key is used to encrypt the messages: At least one of the cryptographic expressions, belonging to a challenge-response handshake of a protocol should contain the *sender identity*.
- Case 3 – a private key is used to encrypt the messages: Both of the cryptographic expressions belonging to a challenge-response handshake of a protocol should contain the *recipient identity*.

V. APPLYING THE PROPOSED GUIDELINES FOR SECURITY PROTOCOL DESIGN

In this section the proposed guidelines for security protocol design are applied to the Andrew Secure RPC and Ban concrete Andrew Secure RPC protocols, in order to detect and avoid the freshness and parallel session attacks outlined in previous sections.

A. Applying Freshness Attack Guidelines

The Andrew Secure RPC protocol (Figure 1) does not respect the proposed guidelines: the cryptographic expression transmitted in a response step (step 4) of this protocol does not contain any fresh component for the recipient (principal A); therefore a freshness attack (Figure 2) can be mounted. Following the proposed design *guidelines for preventing freshness attacks*, the nonce Na is added to the cryptographic expression transmitted in the response step 4 of the protocol (Figure 3). As the recipient A is now able to recognize the cryptographic expression as being fresh, the attack is prevented. As the nonce Na is generated by A as part of the current protocol run, A is able to bind the message to the current protocol run. Messages from previous protocol runs cannot be replayed, as these will not contain the correct nonce.

An alternative solution to prevent the discussed freshness attack is the addition of a timestamp to the encrypted message 4 of the protocol. However, this requires that synchronized clocks are available. This solution also prevents the attack, as the recipient can check that the timestamp is recent. If an intruder attempts to replay message 4 from a previous run, the recipient will be able to detect this replay as the timestamp will be expired.

B. Applying Parallel Session Attack Guidelines

The Ban concrete Andrew Secure RPC protocol (Figure 4) does not respect the proposed guidelines to avoid parallel session attacks: the cryptographic expression used to a challenge-response handshake encrypted with a symmetric key does not contain the *sender identity*. Therefore, a parallel session attack can be mounted (Figure 5). Following the proposed design *guidelines for preventing parallel session attacks*, the identity of the sender B is added in the encrypted expression from message 2 of the protocol (Figure 6). This will prevent the attack from working, as the sender identity identifies the intended use of the message 2, i.e. it is intended to be sent from B to A. Thus, an intruder will be unable to use principal A as an oracle and to replay that message as message i.2 (Figure 5).

As an example to confirm the applicability of the proposed guidelines, for the case when the key used for the encryption of the messages belonging to a challenge-response handshake of a protocol is public, consider Needham-Shroeder public key protocol and its adapted version by Lowe [6]. In the case when the key used for the encryption is private, consider the CCITT X.509 (3) protocol and its adapted version by BAN [5] as an example.

VI. CONCLUSION

This paper introduced security protocols and common attacks that exploit weaknesses in the design of

security protocols. As a case study, the Andrew Secure RPC key distribution protocol by Satyanarayanan and its adapted versions by Burrows et al. and Lowe were reviewed. Two different types of attack - freshness and parallel session attacks - that exploit known weaknesses in the design of these protocols were discussed. The causes of these weaknesses that made the protocols vulnerable to the attacks were analyzed. Based on this analysis guidelines for security protocol design were presented. Protocols that conform to these guidelines will be secure against the presented types of attack.

The applicability of the proposed guidelines for security protocol design is demonstrated by example of the Andrew Secure RPC protocol and its adapted versions. It is shown that compliance with these guidelines successfully detects and avoids the freshness and parallel session attacks.

Acknowledgements

This work was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET Embark Initiative) and Science Foundation Ireland - Research Frontiers Programme (SFI RFP07 CMSF 631).

REFERENCES

[1] Ayesha Altaf, M. Younus Javed, Attiq Ahmed, "Security Enhancements for Privacy and Key Management Protocol in IEEE 802.16e-2005", Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008, pp.335-339

[2] Mahdi Asadpour, Behnam Sattarzadeh, and Rasool Jalili, "Enhanced Smart-Card-Based Authentication Scheme Providing Forward-Secure Key Agreement," The International Conference on New Technologies, Mobility and Security (NTMS'2007), Springer, Paris-France, April 30 - May 3, 2007, pp. 447-458

[3] Y. Espelid, L.-H. Netland, A. N. Klingsheim, and K. J. Hole. A proof of concept attack against Norwegian internet banking systems. In Proc. of the 12th International Conference on Financial Cryptography and Data Security (FC08), Cozumel, Mexico, January 28-31, 2008, pp. 197-201

[4] Dazhi Sun and Zhenfu Cao. New cryptanalysis paradigm on a nonce-based mutual authentication scheme. International Journal of Network Security, Vol. 6, No. 1, 2008, pp. 116-120

[5] M. Burrows, M. Abadi, R. Needham. A logic of authentication. ACM Transactions on Computer Systems TOCS, Vol. 8, No. 1, 1990, pp.18-36

[6] Gavin Lowe. "An attack on the Needham-Schroeder public key authentication protocol", Information Processing Letters, Vol. 56, No. 3, November 1995, pp. 131–136

[7] Gavin Lowe, "Some new attacks upon security protocols", In IEEE Computer Society Press, editor, In Proceedings of the Computer Security Foundations Workshop VIII, 1996, pp. 162-169

[8] Junghyun Nam, Seungjoo Kim, Sangjoon Park, Dongho Won, "Security Analysis of a Nonce-Based User Authentication Scheme Using Smart Cards", IEICE Transactions Fundamentals, vol. E90-A, no. 1, January 2007, pp.299-302

[9] J. Heather, G. Lowe, and S. Schneider, "How to prevent type flaw attacks on security protocols", IEEE Computer Society, 2000, pp. 255–268.

[10] Reiner Dojen, Anca Jurcut, Tom Coffey, Cornelia Györödi: On Establishing and Fixing a Parallel Session Attack in a Security Protocol. Intelligent Distributed Computing, Systems and Applications. Springer Berlin / Heidelberg, Vol. 162, September 2008, pp. 239-244

[11] Dojen, R., Lasc, I. and Coffey, T. , "Establishing and Fixing a Freshness Flaw in a Key-Distribution and Authentication Protocol", In: IEEE International Conference on Intelligent Computer Communication and Processing, August 2008, pp. 185-192.

[12] M. Satyanarayanan, *Integrating security in a large distributed system*. ACM Transactions on Computer Systems, Vol7, No. 3, 1989, pp. 247–280

[13] D. Dolev and A. C. Yao. On the Security of Public Key Protocols. IEEE. T. on Information Theory, Vol. 29, No. 2, 1983, pp.198-208

[14] D. Denning and G. Sacco, "Timestamps in key distributed protocols", Communication of the ACM, Vol. 24, No. 8, 1981, pp. 533–535

[15] Tzonelih Hwang, Narn-Yoh Lee, Chuang-Ming Li, Ming-Yung Ko, and Yung-Hsiang Chen, "Two attacks on Neumann-Stubblebine authentication protocols", Information Processing Letters, no. 53, 1995, pp. 103–107

[16] Tzonelih Hwang and Yung-Hsiang Chen. On the security of splice/as : The authentication system in wide internet. *Information Processing Letters*, Vol. 53, 1995, pp. 97-101

[17] J. Clark and J. Jacob. On the security of recent protocols. *Information Processing Letters*, Vol. 56, No. 3, 1995, 151-155.