A new generic 3-step protocol for authentication and key agreement in embedded systems

Naïm Qachri and Olivier Markowitch

Département d'informatique, Université Libre de Bruxelles,
CP212, boulevard du Triomphe, 1050 Brussels
Belgium

nqachri@ulb.ac.be, olivier.markowitch@ulb.ac.be

Abstract. In this paper we propose a new generic authentication and key agreement protocol intended to be used in wireless environments. The protocol is designed to be implemented on devices with limited computing and storage resources.

Keywords : Authentication, Key agreement protocol, Wireless communication security

1 Introduction

Since now ten years, we have seen many developments to strengthen the security of wireless communication systems. These improvements intended to correct problems related, for example, to defective cryptographic primitives or protocols [8,3] (e.g. the 4-way handshake protocol defined for the Wimedia MAC layer standard [7] used to authenticate and generate session keys). Nowadays, the last versions of the Wifi and Wimax standards include the use of EAP [1] declined in different versions (LEAP - EAP using a Radius Server -, EAP-TTLS, etc...). In practice, EAP is interesting for workstations or desktop computers but does not fit the needed security of particular systems such as handheld devices, short-range communication systems or even domestic Wireless LAN devices. The reason being that many versions of EAP use certificates, public key encryption or exhaustive exchanges of information, that are not viable for lightweight wireless devices.

In this paper, we propose a new generic protocol with low communication that is based exclusively on symmetric encryptions and hashing functions in order to achieve entities authentication and session keys generation.

The existing systems proposed in the literature do not meet all of the needed requirements; indeed, the existing solutions either do not combine authentication and key generation [5,2], use public-key cryptography [9,10], use a trusted entity [5,6], or combine artificially an authentication scheme [4] with a key transport protocol [5]. The ISO/IEC 9798-2 and ISO/IEC 11770-2 protocols differ from the protocol that we are proposing because they do not authenticate the exchanged messages. In particular, because of the lack of messages authentications, the information exchanged during these protocols (that are used to produce the session keys) are not protected against modifications, and are therefore more exposed to denial of services attacks. Moreover, our protocol includes a session key confirmation procedure as well as a synchronization mechanism (in case of repetitive establishments of sessions).

2 Assumptions and notations

We suppose that Alice and Bob (wireless devices that know each other) share a secret key that has been secretly exchanged during an association step that happens before any exchange between the two devices. It is assumed that this association step is secure and therefore does not leak any information about the shared secret. Moreover, it is assumed that the secret, shared by Alice and Bob, is not already used by them with any other devices.

It is also assumed that the devices are tamper resistant (i.e. an attacker cannot physically read or modify the secret stored in the devices).

The notations used in our protocol are the following:
- $h^i(m)$ denotes the hash function h applied successively, i times, on a message m;
- $E_k(m)$ denotes a symmetric bloc encryption of the message m with the secret key k;
- $MAC_k(m)$ denotes the result of a keyed hash function applied on a message m;
- s denotes the secret shared between Alice and Bob;
- r_A and r_B denotes the random nonces chosen and sent by Alice and Bob;
- $LSB_i(m)$ is a function that truncates m to its i *least significant bits.*

3 General description of the protocol

Based on the initial secret (exchanged at the association step), keys for authentication and encryption are generated and shared between Alice and Bob.

The protocol is designed in order to avoid that an attacker, who discovers the secrets of a session of the protocol, can deduce the secrets that will be computed during the following sessions. The secret values of the different sessions are computed on the basis of a chain of hash values. During an initialization step (that takes place after the association step), Alice and Bob realize the computation of n consecutive hashing on the initial shared secret. Those hashed values will be used to authenticate Alice and Bob and to generate the session keys as described hereafter. After the initialization step, when a session must be set, Alice invokes the 3-step main protocol that ensures mutual authentication by the means of challenge-response techniques, session keys generation between Alice and Bob and desynchronization resistance.

Since the produced secret hashed values are in a limited number, when it remains only three hashed values, a new secret is computed from the three remaining hash values and this new secret key is used to create a new chain of hash values that will be used for the next sessions of the protocol.

4 The protocol

4.1 Protocol life cycle

The life cycle of the protocol is as the following :

- association step (where a secret s is exchanged)
- initialization step (where some or all of the n hashed values are computed in order to speed up further hash computations in the protocol and i is initialized to 1 by Alice and Bob)
- successive executions of the main protocol
  and/or *resynchronization protocol* (if needed)
- $(n-1)^{th}$ execution of the main protocol
- renewing protocol
- successive executions of the main protocol (a new cycle is launched)...

CC (for *Cycle Counter*) is a variable that counts the number of chains of n hashed values completely used in the life cycle of the protocol since the initialization step (where CC is set to 1). Within a cycle, a session of the main protocol is characterized by a number i. The concatenation $CC \parallel i$ is a unique identifier of a session of the protocol between Alice and Bob.

4.2 Main protocol

When Alice and Bob have to initiate a new session i, during the cycle CC, they execute the following main protocol:

1. **Alice** $\rightarrow$ **Bob** : $ID_{Alice}, i, E_{h^{n-((i-1)\times 3)}(s)}(1, CC, i, r_A, ID_{Alice},\ h(1, CC, i, r_A, ID_{Alice}))$

The message contains the first challenge under the form of a message to decrypt and verify. If the two devices share the same secret, then Bob can decrypt and verify it. The nonce, r_A, sent by Alice, has to be well chosen and contributes to the keys generation in a fair way in regards to Bob.

Once Bob has decrypted the message, he chooses a second random nonce, r_B, and generates three keys by computing the following MAC:

$$(\mathbf{k_{SE}} \parallel \mathbf{k_{SA}} \parallel \mathbf{k_{conf}}) = LSB_q(MAC_{h^{n-((i-1)\times 3)-2}(s)}(CC \parallel i \parallel r_A \parallel r_B \parallel h^{n-((i-1)\times 3)-1}(s)))$$

where $\parallel$ is the concatenation operator. r_A and r_B are the contributions of respectively Alice and Bob in the computation of these three keys. q denotes the sum of the sizes of the different keys generated during the protocol. $CC \parallel i$ is used to avoid replay attacks. $\mathbf{k_{SE}}$ is the key generated to encrypt the communication of the session that will take place between Alice and Bob and $\mathbf{k_{SA}}$ is the key generated to authenticate the packets during this communication.

The MAC algorithm has to be well dimensioned to generate enough bits for the three keys. On the basis of $\mathbf{k_{conf}}$, Bob creates a new challenge and sends it to Alice.

2. Bob → Alice : $ID_{Bob}, i, E_{h^{n-((i-1)\times 3)}(s)}(2, CC, i, r_B, r_A, ID_{Bob}, MAC_{k_{conf}}(2, CC, i, r_B, r_A, ID_{Bob}))$

The challenge has the purpose to ensure that Alice can derive the good key and decrypt the message of Bob. From these keys, Alice can verify that she has derived the same keys than Bob if she is able to verify the MAC on the message. Alice can also authentify Bob, since only Bob knows the secret hashed value used to encrypt the message and authenticate. Furthermore, Bob sends the nonce r_A to give the proof that he has made the correct decryption of the first message.

3. Alice → Bob : $ID_{Alice}, i, MAC_{k_{conf}}(3, ID_{Alice}, CC, i, r_A, r_B)$

In this third message, Alice answers that she has well derived the keys and that the authentication of Bob succeeds, she provides also r_B to give the proof that she has made the correct decryption of the second message. At the end of the main protocol, Alice and Bob increments i.

4.3 Renewing protocol

Within a cycle, on the basis of n hash values, we can realize $\frac{n}{3} - 1$ sessions of the main protocol. We use the last session of the protocol to generate a new secret value shared by Alice and Bob.

The renewing is made when only three hashed values ($h^3(s), h^2(s)$ and $h(s)$) remain before the secret is reached. Alice and Bob run again the protocol (see Figure 1) during which the new secret is computed from $h(s)$ and s.

$$\mathbf{s_{new}} = MAC_{s_{old}}(CC \parallel r_A \parallel r_B \parallel h(s_{old}))$$

The shared secret hash values and the CC are computed for the further sessions of the next cycle of the main protocol:

$$h(s_{new}), h^2(s_{new}), h^3(s_{new}), ..., h^n(s_{new}) \quad \text{and} \quad CC \leftarrow CC + 1; \quad i \leftarrow 1$$

4.4 Resynchronization protocol

If the two devices are desynchronized (i.e. if Alice and Bob consider a different value of i), they reveals their session values i and i'. In that case, the current session of the main protocol is aborted and a new session protocol is launched with a session value $max(i, i') + 1$.

We make the assumption that, in case of desynchronization, the devices cannot have different CC's, because it would mean that one of the two devices has done the renewing protocol without having incremented the variable CC

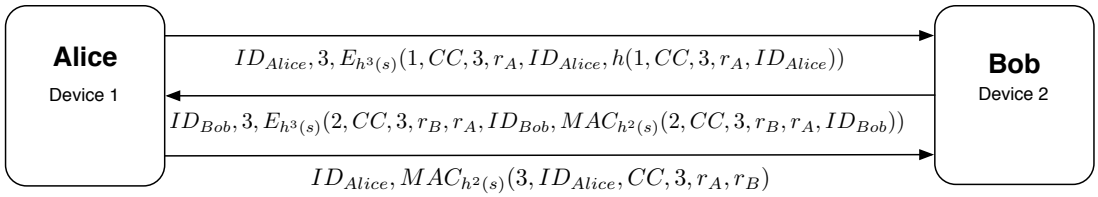

Fig. 1. The renewing protocol

5 Work in progress

We are currently studying the formal security of the protocol with SVO logic. We will then study and compare the performances of some possible constructions of the protocol with concrete cryptographic primitives. We will also study the possibility to adapt our protocol in a multi-party framework.

References

1. Aboba, B., Blunk, L., Vollbrecht, J., and Carlson, J. *Extensible authentication protocol* (EAP). RFC 3748, June 2004.
2. Diffie, W., and Hellman, M. E. *New directions in cryptography.* In *IEEE Transactions on Information Theory* (1976), vol. 22, pp. 644–654.
3. Fluhrer, S., Mantin, I., and Shamir, A. *Weaknesses in the key scheduling algorithm of rc4.* In *Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography* (2001), Springer Berlin / Heidelberg, Ed., pp. 1–24.
4. Lamport, L. *Password authentication with insecure communication. Communications of the ACM 24*, 11 (November 1981), 770–772.
5. Needham, R., and Schroeder, M. *Using encryption for authentication in large networks of computers. Communications of the ACM 21*, 12 (December 1978), 993–999.
6. Neuman, B. C., and Ts'o, T. *Kerberos: an authentication service for computer networks. IEEE Communications 32*, 9 (September 1994), 33–38.
7. Qachri, N., and Roggeman, Y. *The flaws and critics about the security layer for the wimedia mac standard.* In *30-th symposium on Information Theory in the Benelux* (may 2009), pp. 89–96.
8. Stubblefield, A., Ioannidis, J., and Rubin, A. D. *Using the fluhrer, mantin, and shamir attack to break WEP.* Tech. Rep. TD-4ZCPZZ, AT&T Labs, 2001.
9. Wang, F., and Zhang, Y. *A new provably secure authentication and key agreement mechanism for sip using certificateless public-key cryptography.* In *2007 International Conference on Computational Intelligence and Security* (December 2007), IEEE, Ed., pp. 809–814.
10. Zou, X., Thukral, A., and Ramamurthy, B. *An authenticated key agreement protocol for mobile ad hoc networks.* In *Mobile Ad-hoc and Sensor Networks. Second International Conference, MSN 2006. Proceedings (Lecture Notes in Computer Science Vol. 4325)*. Springer-Verlag, January 2006.