Analysis and Modification of ASK Mobile Security Protocol

Il-Gon Kim, Hyun-Seok Kim, Ji-Yeon Lee, Jin-Young Choi
Dept. of Computer Science and Engineering, Korea University, Seoul, 136-701 KOREA
{igkim, hskim, jylee, choi}@formal.korea.ac.kr

Abstract

Generally, security protocols have been designed and verified using informal techniques. In the result, it is now well recognized that many security protocols which were previously proposed have found to be vulnerable later on. In this paper, we model and verify of the ASK protocol, which is a complicated mobile security protocol. After showing the vulnerability of the ASK protocol using formal verification approach, we propose a modification, and then show that the new protocol is secure against replay attack.

1 Introduction

Generally, security protocols have been designed and verified using informal techniques. In the result, it is now well recognized that many security protocols which were previously proposed have found to be vulnerable later on[9]. To solve this problem, formal methods have been widely used to specify security protocols and verify security properties, such as confidentiality, authentication and non-repudiation, to guarantee correctness[1][8][11]. Especially, the use of Caper/FDR approach has been very successful over the past few years and has discovered many attacks against protocols that were thought to be correct[9].

Many formal verification approaches are based on fixed wire-based security protocols. Relatively few studies have been devoted to formal analysis of the safety of wireless-based mobile security protocols.

Currently, many new mobile security protocols are proposed in the literature of communication protocols with the development of wireless network and rapid spread of low-power devices such as mobile phones. However, the design of mobile security protocols is very difficult due to some constraints of computation overhead, battery consumption, and low-bandwidth in wireless networks. Furthermore, the security in communication protocols based on wireless networks is more vulnerable than fixed wire-based protocols. Therefore, it is very important to guarantee the safety and reduce redundant communication steps, considering computation overhead and network speed of mobile security protocols during the development of them.

In this paper, we verify the safety of the ASK protocol, which is a complicated protocol for authentication and key agreement. Next, we identify the weakness of the ASK protocol using Casper/FDR. Then we propose a modified ASK protocol which is strong against man-in-the-middle attack and reduces a redundant communication step.

The remainder of this paper is organized as follows. In section 2, we give a brief overview of CSP/Casper and FDR approach. In section 3, we design, verify and modify the ASK protocol. Finally, section 4 concludes this paper.

2 Casper and FDR

2.1 Casper(A Compiler for the Analysis of Security Protocols)

Over the last few years, a method for analyzing security protocol that firstly model communication security protocol using CSP[6], then verifies its secrecy, authentication and other properties usingFDR[9][11][10].

In this method, the main difficulty is specifying the security protocol's behavior using CSP. Creating the description of the security model with CSP is a very error-prone and difficult task. To simplify the expression of the security protocol, and render this process more error free, Casper was developed by Gavin Lowe[10]. This tool enables a non-expert who is unfamiliar with CSP to express the security protocol's behavior more easily, without being familiar with the notation used by CSP notation, using various key types, messages, security properties and intruder knowledge descriptions contained in Casper. In brief, Casper is a compiler that translates a more simple and concise description of a security communication model into CSP code. The security process is described by means of 8 section headers, including "#Free variables", "Processes", "#Protocol", "#Specification", "#Variables", "#Functions", "#System", and "#Intruder Information".

2.2 FDR(Failure Divergence Refinement)

FDR is a model checking tool for state machine, with foundations in the theory of concurrency based on CSP. This tool checks whether a security model described with CSP_M(Machine Readable CSP) satisfies certain security properties such as secrecy and authentication. If the security model doesn't satisfy these properties, FDR shows counterexample event traces and helps to analyze which attack scenario would be most likely to happen. For the equivalence checking of the specification model(safety property model) and implementation model(security system model paralleled with intruder model), FDR supports three refinement checking methods. First, traces refinement can show safety property. Second, failures refinement can represent system's deadlock property. Finally, failure and divergence can detect the livelock property. For detailed information about the CSP, Casper and FDR, the reader is referred to [6][5][10].

Fig.1 shows an overall procedure to design and verify of security protocols using Casper/FDR approach.

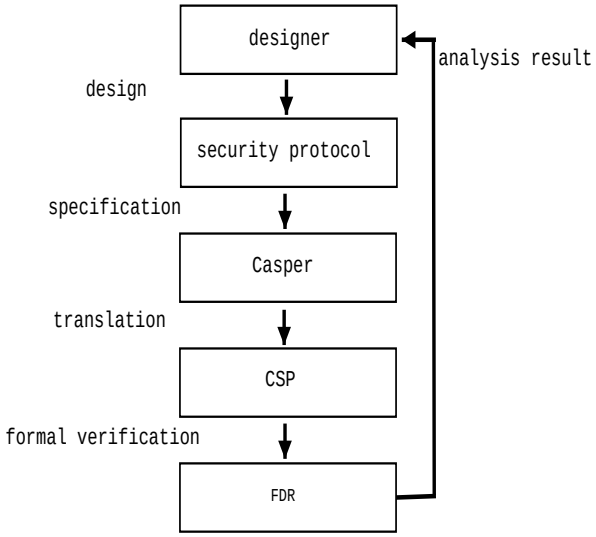

Figure 1. Framework for modelling and verification of security protocols using Casper/FDR

First, a designer of a security protocol designs it considering the security and network speed. In this step, the designer combines message variables and use cryptographic algorithms to protect important information such as a session key. Second, we specify the security properties and message sequences of security protocols written in Casper script. Third, we obtain CSP code using automatic translation function of Casper. Fourth, we input the CSP code into FDR model checking tool and run it. If a security protocol doesn't satisfy security properties such as confidentiality and authentication , which are described in Casper script, then FDR tool shows a counter example which represents the reason against vulnerability. Lastly, the designer modifies the weakness until the newly designed security protocol meets security requirements.

Message 1. S → V : CertV

Message 2. S → U : CertU

Message 3. V → U : Kv+

[U computes : K1 = {Kv+}Ku-]

Message 4. U → V : Ku+

[V computes : K2 = {Ku+}Kv-]

Message 5. V → U : {CertV, Rv}K

[U and V compute : SK = {K}Rv]

Message 6. U → V : {CertU, Rv}K

Figure 2. ASK protocol

3 Formal Verification and Modification of the ASK Protocol

In this section, we verify the safety of the ASK mobile security protocol with Casper/FDR. Then, we confirm the security weakness of the ASK protocol. Next, we verify the correctness of a modified ASK protocol proposed in this paper.

3.1 ASK protocol

The ASK(Aydos, Sunar, and Koc) protocol was first proposed by *Aydos et al*[2]. The purpose of this protocol is to provide authentication and key agreement in low-power portable devices such as mobile phones. In [7], the author addresses that the ASK protocol is more secure and faster than the BCY[4] and Aziz-Diffie[3] protocols. In addition, no research has been carried out to analyze the vulnerability of the ASK protocol using model checking. Table 1 shows the basic notation of the ASK protocol. Fig.2 shows message sequences of the ASK protocol.

In the ASK protocol, *S* is the third-party key distribution server, *V* is the service provider to send content data to *U*, and *U* is the user that needs to be authenticated by *V*.

Message 1: *S* sends the certificate, *CertV* which includes *V* and K*v+*, to *V*, in order to distribute a public

Table 1. ASK protocol notation and meaning

Notation	Meaning
U	An identifier of the mobile user
V	An identifier of the service provider
S	An identifier of the certification authority
Kx+	Public key of X agent
Kx-	Private key of X agent
K	Diffie-Hellman exchange key(K1 = K2)
SK	Mutual session key(SK1 = SK2)
Rx	a random nonce generated by X agent
CertV	A certificate of V issued by S
CertU	A certificate of U issued by S

key *Kv+*(for Diffie-Hellman key agreement).

Message 2: *S* sends the certificate, *CertU* which includes *U* and *Ku+*, to *U*, in order to distribute a public key *Ku+*.

Message 3: *V* forwards his identity *U* and the public key *Kv+* to *U*. Then, the user *U* computes *K1* = {*Kv+*}*Ku-*, which is a Diffie-Hellman exchange key.

Message 4: *U* sends his own public key *Ku+* to *V*. The service provider *V* computes *K2* = {*Ku+*}*Kv-*, Thus, V and U have exchanged the same shared key *K(K1 = K2)*.

Message 5: *V* sends its certificate, *CertV*, and random nonce, *Rv*, encrypted with Diffie-Hellman key *K*. Thus, *V* and *U* compute the session key *SK* using *Rv*.

Message 6: Finally, the user *U* sends its certificate to the service provider *V*, also encrypted using *K*.

3.2 Formal Verification of the ASK Protocol

Based on the ASK protocol's notation, mentioned in previous section, we write the ASK protocol model in Casper script. The *#Free variables*, *#Protocol*, *#Specification*, *#Intruder information*, and *#Equivalences* header sections of the ASK protocol model using Casper script are shown. The other Casper header descriptions are not mentioned in this paper, because they are fairly trivial.

```
#Free variables
v, u : Agent
s : Server
pkv, pku : PublicKey
skv, sku : SecretKey
SPK :  Server -> ServerPublicKey
SSK :  Server -> ServerSecretKey
rv : Nonce
InverseKeys = (pkv, skv), (pku, sku),
              (SPK, SSK)
```

The *#Free variable* section declares the types of the free variables used in the *#Protocol description* section. The *v* and *u* are agent's identities; *s* is the server's identity; *pkv* and *pku* point out *Kv+* and *Ku+*, respectively; *skv* and *sku* refer to *Kv-* and *Ku-*, respectively; *SPK* and *SSK* are functions that return a server's public and secret keys; *rv* is a random nonce of *v* agent the "*InverseKeys*" line defines which keys are inverses of which others.

```
#Protocol description
0.    -> v : u
1a. s -> v : {v, pkv}{SSK(s)} % digV
1b. s -> u : {u, pku}{SSK(s)} % digU
2.  v -> u : pkv
3.  u -> v : pku
4.  v -> u : {{digV % {v, pkv}{SSK(s)},
              rv}{pku}}{skv}
5.  u -> v : {{digU % {u, pku}{SSK(s)},
              rv}{pkv}}{sku}
```

The *#Protocol* description section defines the protocol itself, as a sequence of messages. The notation {*m*}{*k*} represents that the message *m* is encrypted with key *k*. In addition, we write m%d, where m is a message and d is a variable, to denote that the recipient of the message should not attempt to decrypt the message m, but should instead store it in the variable d. Similarly, we write d%m to indicate that the sender should send the message stored in the variable d, but the recipient should expect a message form given by m. For example, the messages 1a and 4 mean that a certificate authority server sent the certificate message {*V, Kv+*}*Ks-* and a service provider has forwarded it to a user.

```
#Specification
Secret(v, rv, [u])
Secret(u, rv, [v])
Agreement(v, u, [rv, pku, skv])
Agreement(u, v, [rv, pkv, sku])
```

The *#Specification* section is used to specify security properties of the protocol. The lines starting with 'Secret' represent that secrecy property is certain secret information between only specific hosts. Secrecy property states that intruders cannot obtain this secret information during a run of the protocol whenever its secrecy is claimed. The second line is interpreted as "The service provider *v* thinks that *rv* is a secret that should be known only to *v* and *u*." The lines starting with 'Agreement' define the authentication property. The authentication property represents the

establishment guarantees when it has completed, concerning the party it has apparently been running with. The fifth line means that "the user *u* is authenticated to the service provider *v* with *rv*, *pkv*, and *sku*."

```
#Intruder information
Intruder = Mallory
IntruderKnowledge = {Vendor, User, Sam,
                     Mallory, Rm, PKv,
                     PKu, PKm, SKm,
                     SPK(Sam)}
```

The *#Intruder information* section contains the basic intruder knowledge needed to attack communicating agents. In this paper, we assume that a intruder *Mallory* knows the identities of all the agents and a server, his own single nonce *Rm*, all public keys, and his own secret key *SKm*.

```
#Equivalences
forall rv, pkv, pku, skv, sku .
{{rv}{pku}}{skv}={{rv}{pkv}}{sku}
```

The *#Equivalences* section defines algebraic equivalences between messages. The second line expresses the property that the encryption property used is commutative. For example, the session key $\{\{Rv\}\{Ku+\}\}Kv-$ in the ASK protocol has the same meaning with $\{\{Rv\}\{Kv-\}\}Ku+$.

After running the FDR model checking tool, we found that the ASK protocol does not satisfy the above authentication property('Agreement(v, u, [rv, pku, skv]') described in *#Specification* section. A general attack scenario, which could be found in the ASK protocol is summarized below:

1. S → I(V) : {V, Kv+}Ks-
2. I(S) → V : {V, Kv+}Ks-
3. V → I(U) : Kv+
4. I(U) → V : Ku+
5. V → I(U) : {{V, Kv+}{Ks-}}, Rv}{Ku+}}Kv-
6. S → I(U) : {U, Ku+}Ks-
7. I(S) → U : {U, Ku+}Ks-
8. I(V) → U : Kv+
9. U → I(V) : Ku+
10. I(V) → U : {{{V, Kv+}{Ks-}, Rv}{Ku+}}Kv-
11. U → I(V) : {{{U, Ku+}{Ks-}, Rv}{Kv+}}Ku-

The notation *I(V)* represents the intruder *I* imitating *V* to fake or intercept a message. Through the man-in-the-middle attack of the ASK protocol, in the 10th message, an intruder masquerading as *V* could forward the message $\{\{V, Kv+\}\{Ks-\}\}, Rv\}\{Ku+\}\}Kv-$ to *U*, which is intercepted in the 7th message. The above attack scenario shows that the service provider *V* is not authenticated to the user *U*, because the identity, public keys and session keys of *V* and *U* in the ASK protocol could be intercepted, faked and replayed by a malicious intruder *I*. The verification result using FDR model checking tool confirms that the ASK protocol doesn't satisfy some requirements which mobile security protocols must abide by, due to the following weakness identified in [7]:

- The identity confidentiality of a user may be compromised, since the user's public key is sent in clear in the 4th message(see Fig.1). If the public key of a user is known, then user confidentiality is completely lost.
- The service provider is not authenticated to the user, because the 5th message(see the attack scenarios in the ASK protocol) may be intercepted and replayed by an intruder.
- The mutual key agreement of a session key between the user and the service provider may be interrupted, because there is no freshness checking of a session key. If a session key should ever be compromised, then known key attacks become possible. It means that the user cannot establish that the session key is fresh.

3.3 Modified ASK Protocol

In light of these weakness, we propose a modified ASK protocol that a number of amendments are made to the ASK protocol. Firstly, we remove messages 3 and 4, because they are overlapped with messages 5 and 6, and the confidentiality of public key of the user may be compromised by an intruder. Secondly, we add two expiration times, *TSv* and *TSu*, which are included in *V*'s certificate, *CertV*, and *U*'s certificate, *CertU*. Thus, the service provider and the user confirm that public keys of *V* and *U* are current, not replayed by an intruder. Thirdly, we include the user's nonce, *Ru*, in order to generate a new session key. The session key becomes *h(ru, rv)*; *h* represents an one-way hash function such as MD4 and MD5.

The user and the service provider confirm freshness of the session key, because it is exchanged securely using Diffie-Hellman key, $\{Ku+\}Kv-$. Fig.3 shows a modified ASK protocol that we propose in this paper.

We verify security properties, i.e., secrecy and authentication properties, of the modified ASK protocol using Casper/FDR. The *#Specification* section shows the secrecy and authentication properties.

Message 1. S → V : CertV

Message 2. S → U : CertU

Message 3. V → U : {CertV, Rv}Kv-

[U computes : Ru]

Message 4. U → V : {CertU, {Rv, Ru}Ku-}Kv+

[V computes : SK = h(Rv, Ru)]

Message 5. V → U : {{Rv, Ru}Ku+}Kv-

[V computes : SK = h(Rv, Ru)]

Figure 3. Modified ASK protocol

```
#Specification
Secret(v, ru, [u])
Secret(u, ru, [v])
Agreement(v, u, [rv, ru, pku, skv])
Agreement(u, v, [rv, ru, pkv, sku])
```

After running FDR tool, we confirm that the modified ASK protocol satisfies all of these properties, shown in above. This result means that the user's nonce *Ru* is not intercepted by an intruder, and the service provider is authenticated to the user with the session key and Diffie-Hellman key. Thus, we show the correctness of the modified ASK protocol using Casper/FDR approach in the design phase.

4 Conclusion

With the rise of mobile communication networks, numerous security protocols have been proposed. However, they are designed and developed based on an informal technique, so it is acknowledged that they have security weakness that a designer didn't expect. Therefore, it is very important to guarantee the secrecy and authentication properties of mobile security protocols against a malicious intruder. In this paper, we have verified the safety of the ASK protocol and showed that it has violated some security requirements. Then, we proposed a modified protocol and showed that it is secure against replay attack. We believe that a modified version is more robust and faster than the ASK protocol, because it has used reduced message communication steps to bring about computation overhead and battery computation.

References

[1] M. Abadi, M. Burrows, and R. Needham. "A Logic of Authentication", *Proceeding of the Royal Society*, Series A, 426, 1871, pp. 233-271, December 1989.

[2] M. Aydos, B. Sunar, and C.K. Koc, "An elliptic curve cryptography based authentication and key agreement protocol for wireless communication", presented at the 2nd Int. Workshop Discrete Algorithms and Methods for Mobility, Dallas, TX, Oct. 1998.

[3] A. Aziz and W. Diffie, "Privacy and authentication for wireless local area networks", IEEE Personal Commun., First Quarter 25 -31, 1994.

[4] M. J. Beller, L. -F. Chang and Y. Yacobi, "Privacy and authentication on a portable communications system", *Proceedings of the International Computer Symposium*, Vol.1, pp.821-829, 1994.

[5] Formal Systems(Europe) Ltd. Failure Divergence Refinement-FDR2 User Manual, 1999.

[6] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[7] G. Horn, K. M. Martin, and C. J. Mitchell, "Authentication Protocols for Mobile Network Environment Value-Added Services", IEEE Transactions on Vehicular Technology 51, pp.383-392, 2002.

[8] I. G. Kim and J. Y. Choi, "Formal verification of PAP and EAP-MD5 Protocols in wireless networks : FDR Model Checking", in the 18th International Conference on Advanced Information Networking and Applications(AINA 2004), pp.264-269, 2004.

[9] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using FDR", *Proceedings of TACAS*, number 1055 in LNCS. Springer, 1996.

[10] G. Lowe, "Casper: A compiler for the analysis of security protocols", 10th IEEE Computer Security Foundations Workshop, 1997.

[11] P. Y. A. Ryan and S. A. Schneider, modelling and analysis of security protocols: the CSP Approach, Addison-Wesley, 2001.