Analysis of Security Protocols with Certificate over Open Networks: Electronic Payment System

Hyun-Seok Kim, Il-Gon Kim, Jin-Young Choi

Dept. of Computer Science and Engineering, Korea University, Seoul, KOREA

{hskim, igkim, choi}@formal.korea.ac.kr

Abstract

Electronic Commerce and Internet in wireless networks are profoundly changing the way of payment, but there is still little confidence among users concerning the security of their data. The application of formal techniques to the modelling and design of electronic commerce protocols should help to improve their reliability and so enhance the choices of these new technologies. In this paper, we show how the Casper, a special-purpose formal methods tool designed for the verification of the security protocols, was used in the analysis of the BCY(Beller, Chang, Yacobi), the Carlsen BCY and the Mu-Varadharajan BCY protocols. We describe the results of our analysis, which uncovered several vulnerabilities in the specification that would have made possible attacks such as man-in-the-middle attack and replay attack. Finally, we propose a new protocol resistant to these attacks and formally verify its correctness.

1 Introduction

The exponential growth of the Internet and electronic commerce brings not only prosperity, but also vulnerability. Numerous attacks pose a real challenge to different aspects of security mechanisms. Among these security mechanisms, security protocols play an essential role. They use cryptographic primitives as building blocks to achieve security goals such as authentication, confidentiality and integrity.

A generic set of security requirements for protocols, in open networks against which the protocols in this paper are evaluated, is as follows:

- 1) Mutual authentication of user and network
- 2) Exchange of certified public keys
- 3) Session key agreement
- 4) Joint control of session key

- 5) Mutual implicit key authentication
- 6) Mutual key confirmation
- 7) Mutual assurance of key freshness
- 8) Confidentiality
- 9) Initialisation of payment mechanism
- 10) Non-repudiation of origin

Several security protocols have been proposed recently for wireless communication. These include the authentication and key agreement protocol(AKA) used for the universal mobile telecommunication system[1]. For example, the ASPeCT protocol[2], the Boyd-Park protocol[3] and the BCY protocol[4] originally designed by Beller, Chang and Yacobi, subsequently revised by Carlsen[5] and Mu and Varadharajan[6], demonstrated the feasibility of public key cryptography in mobile communications.

In this paper, we focus on the main payment phase of the BCY protocols and its key exchange. We present the verifications of the BCY protocols and propose a new protocol using Casper and FDR. This paper is organized as follows; In Section 2, we describe related works to formal verification of electronic commerce protocols. In Section 3, we outline the use of CSP, Casper and FDR tool for automatically analysing security protocols. In Section 4, we describe the BCY protocols and its analysis result. In Section 5, a new BCY protocol, resistant to attacks, is proposed. In addition, formal verification of the new proposal is presented. In Section 6, we conclude the paper and discuss some issues that are raised by this work.

2 Related Works

There are several approaches to analyze the correctness of electronic payment system using formal methods. A well-known software tool is the NRL Protocol Analyzer,

which has been expanded to include some specification capabilities, thus being able to specify the security requirements of the SET protocol[7]. Another notable work includes analyzing the SET protocol using Lowes use of the Failures Divergences Refinement Checker in CSP [8] and Schneiders use of CSP.

iKP (i-Key-Protocol, $i = 1, 2, 3$)[9] is a family of payment protocols and has affected the design of SET. The verification has been done with CafeOBJ[10]. CafeOBJ is an algebraic specification language. The CafeOBJ system, an implementation of CafeOBJ, can be used as an interactive proof-checker or verifier on several levels.

NetBill[11] is an electronic commerce protocol designed to be used in commercial transactions of information through Internet. This work presents the verified results using SPIN. Its approach is the combination of the two notations : widely spread graphical tools taken from Object Modelling Technique to support the development of a model and a model checker(SPIN) to prove the necessary properties.

3 CSP, Casper and FDR

3.1. CSP

A more rigorous approach which is based on modelling all the agents taking part in the protocol including the communicating principals and the intruder as Communicating Sequential Processes(CSP)[8]. This method can be used to formalize messages, traces, intruders, and nonce challenges. A program named Casper, developed by Lowe, automatically produces the CSP description from a more abstract description, thus greatly simplifying the modelling and analysis process. It uses a lazy exploration strategy which examines the subset of intruder states reachable by the protocol rules effectively exploring the behaviour of the intruder in parallel with the protocol's evolution. A particular advantage of their methodology lays in its ability to reason about the absence of denial-of-service attacks. This technique requires the production of a CSP description of the protocol by hand. This has proved not only time-consuming but error-prone as well, even for experts in this area.

3.2 Casper and FDR

Casper[12] was developed in order to express the behavior of security protocols more easily and to allow their security properties to be verified. The Casper tool translates a high-level description of a security protocol into CSP, which is the process algebra of Communicating Sequential Processes developed by Hoare. A Casper script consists of two

parts: a general part that specifies a model of a system running the protocol and a specific part that defines a particular image of a given function by instantiating the parameters of the protocol.

To guarantee the safety of the security protocol in the design phase, we first make an abstract model of the protocol with Casper and then generate CSP code with the compilation function of Casper. Next, we can run the FDR model checking tool to verify whether the security protocol satisfies the security properties or not. We chose the Casper/FDR approach for the analysis of the security protocol because Casper provides not only a simple notation but also formal semantics which maps to the CSP language. In addition to this, the FDR model checking is very good at verifying concurrency models such as communication protocols.

4 Formal Verification of the BCY Protocols

4.1 The original BCY protocol and its verification

Several new protocols have been proposed for wireless communications. Among these, the BCY protocols [4] designed by Beller, Chang and Yacobi was the first to use cryptography for authentication and key agreement in low-power portable devices such as mobile phones. The protocol combines a Diffie-Hellman key exchange with the modular square root (MSR) encryption technique . The MSR technique reduces the computational burden placed on the mobile device. Certificates, denoted by $\{X_1, \dots, X_n\}K_{ca}$, contain the components X_1 to X_n as well as a hash of these components signed by a certification authority. As the Access Server uses two public keys, one for Diffie-Hellman key agreement(K_{md+}) and one for MSR encryption (K_{mm+}).

4.1.1 Modelling original BCY using Casper

The certificate is encrypted using the private key of the Certificate Authority who is a trusted third party. Since if it is encrypted with the private key of CA, any other user cannot spoof it. It provides confidence of certificate's information to participant. The certificate can be decrypted by public key of CA, which is known to Merchant and Client.

It is assumed that initial steps of the BCY protocol involve the Certificate Authority, the third party, passing the public key along with the certificate that proves the authenticity of the public key. The CA's public key is assumed to be with the users before the protocol is run.

During the first two steps of the BCY protocol, the principals exchange public keys. At this stage, each principal can compute the key encryption key, KK , using the Diffie-Hellman technique. Merchant(M) sends

Table 1. BCY protocol notation

C	An identifier of the Client
M	An identifier of the Access Server(Merchant)
CA	An identifier of the Certificate Authority
Kx+	Public key of X
Kx-	Private key of X
KK	A key encrypting key
SK	A session key
rX	A random nonce generated by X
TSx	An expiration time

Figure 1. The BCY protocol description

a message to the Client(C), containing his certificate($\{M, Kmd+, Kmm+\}Kca-$). The certificate contains the identity and public keys of principal M, all signed by the certification authority(CA). Upon receipt of M's certificate, Client can calculate the session key. M can also verify that the public key, received earlier, belongs to M by examining the certificate. Next C sends a message to M, containing his certificate($\{C, Kc+\}Kca-$), encrypted under rC. The certificate contains the identity and public key of principal C, all signed by the certification authority. Upon receipt of C's certificate, M can calculate the session key and also verify that the public key, received earlier, belongs to C by examining the certificate. The protocol concludes with a message from C to M containing dataM to provide authentication of C and vice versa.

#Free variable section of the Casper file declares the types of the free variables used in the protocol description. Given below are variables used in the different steps of the protocol. *Agent*, *Server*, *PublicKey*, *SecretKey* and *Nonce* are some of the basic data types available in Casper. We could define functions that take in an input parameter and return an output. It resembles a functional programming language in this aspect. Definition of a function called SPK

that takes in the name of an Server and returns a *Server-PublicKey* could be given as shared : *Server* $\rightarrow$ *Server-PublicKey*. Given below is the free variable section in the Casper file that declares variables used in the protocol.

#Free variables

M, C : Agent

CA : Server

pkmd, pkmm, pkc : PublicKey

skmd, skmm, skc : SecretKey

SPK : Server $\rightarrow$ ServerPublicKey

SSK : Server $\rightarrow$ ServerSecretKey

dataM, dataC : Nonce

rM, rC : Nonce

InverseKeys = (pkmd, skmd), (pkmm, skmm),
(pkc, skc), (SPK, SSK), (rM, rM), (rC, rC)

This next of the Casper file, #Processes, defines the processes that play roles of INITIATOR as Merchant, RESPONDER as Client, SERVER as Certificate Authority.

#Processes

INITIATOR (M, pkmd, pkmm, pkc, skmm,
skmd, dataM, rM) knows SPK

RESPONDER (C, pkmd, pkc, skc, dataC, rC)
knows SPK

SERVER (s, pkmd, pkmm, pkc, M, C)
knows SSK, SPK

#Specification specifies the requirements of the protocol. The protocol is checked against these specifications to determine if the properties of the protocol hold during subsequent runs of the protocol. The first statement below signifies that after a complete run of the protocol M believes that only C know about dataM. The third specifications states that after a successful run of the protocol, M and C agree upon the value of rM and rC.

#Specification

Secret(M, dataM, [C])

Secret(C, dataC, [M])

Agreement(C, M, [rM, rC])

The processes are instantiated with the values in the actual variables section(#System).

#System

INITIATOR (Merchant, PKmd, PKmm, PKc,
SKmd, SKmm, SKc, NM, RM)

RESPONDER (Client, PKmd, PKc, SKc, Nc, Rc)

SERVER (Sam, PKmd, PKmm, PKc, Merchant,
Client)

The following shows the intruder definition (*#Intruder Information*).

```
#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Merchant, Client,
  Sam, Mallory, Nm, PKmd, PKmm, PKc,
  PKm, SKm, SPK(Sam), Rm}
```

In addition, at this next of the Casper file we use the #Equivalences, we define algebraic equivalence of session key SK in Diffie-Hellman key exchange. The key KK and SK between M and C have the following equivalence relationship:

```
KK = {Kc+}Kmd- (M's KK)
    = {Kmd+}Kc- (C's KK)

SK = {{{Rc}Kc+}Kmd-} (M's SK)
    = {{{Rc}Kmd+}Kc-} (C's SK)
```

```
#Equivalences
forall nc, pkmd, pkc, pkmd, skc, rc.
  {{{nc}{rc}}{pkc}}{skmd} =
    {{{nc}{rc}}{pkmd}}{skc}

forall nm, pkmd, pkc, skmd, skc, rc.
  {{{nm}{rc}}{pkmd}}{skc} =
    {{{nm}{rc}}{pkc}}{skmd}
```

4.1.2 Protocol Goals

Secrecy - also called concealment or confidentiality - has a number of different possible meanings; the designer of an application must decide which one is appropriate. The strongest interpretation would be: an intruder is not able to learn anything about any communication between two participants of a system by observing or even tampering the communication lines. That is, he cannot deduce the contents of message, sender and receiver, the message length, the time they were sent, and not even the fact that a message was sent in the first place.

The following predicate is implemented in CSP language.

```
SECRET_SPEC_0(s_) =
  signal.Claim_Secret?A_!s_?Bs_ ->
  (if member(Mallory,Bs_) then
    SECRET_SPEC_0(s_)
  else SECRET_SPEC_1(s_)) []leak.s_->
  SECRET_SPEC_0(s_)
```

Formally speaking, If A has completed a protocol run apparently with B, and B is honest and uncompromised, then the key accepted during that run by A is not known to anyone other than B. Similarly, if B has completed a run with honest and uncompromised A, then the key accepted by B is not known to anyone other than A. That is, at the end of a successful protocol run each principal(A, B) should possess

the other's public key and ensure that this key belongs to the other principal. In addition, the participants of the protocol should be satisfied that the session key is a suitable key for communication and that it is fresh. Each principal should also be able to confirm that the other protocol participant possesses the session key(SK) and ensure it to be a shared secret.

Authentication : A system provides strong authentication if the following property is satisfied: if a recipient R receives a message claiming to be from a specific sender S then S has sent exactly this message to R. For most applications this formulation must be weakened, since in most cases communication channels are subject to both technical errors and tampering by attackers. A system provides weak authentication if the following property is satisfied: if a recipient R receives a message claiming to be from a specific sender S then either S has sent exactly this message to R or R unconditionally notices this is not the case.

```
AuthenticateINITIATORToRESPONDER
  Agreement_0(A) =
  signal.Running1.INITIATOR_A.B ->
  signal.Commit1.RESPONDER_B.A -> STOP
```

Formally speaking, events of the form *Running1.INITIATOR _ A.B* in A's run of the protocol are introduced to mark the point that should have been reached by the time B performs the *Commit1.RESPONDER _ B.A event*. Occurrence of *Running1.INITIATOR _ A.B* run means simply that Agent A is following a protocol run apparently with B. If a *Running1.INITIATOR _ A.B* event must always have occurred by the time the *Commit1.RESPONDER _ B.A event* is performed, then authentication is achieved. That is, the protocol is checked against these specifications to determine if the properties of the protocol hold during subsequent runs of the protocol.

4.1.3 Result of Verification

When we run this description of protocol in Casper with FDR tool, we found several known flaws in the protocol. The original BCY protocol doesn't satisfy the security properties below;

- At the step 2, the Nonce(rC) just can not ensure freshness of M's Certificate. The Intruder can reuse M's Certificate. It makes possible to replay attack.
- The Session Key(SK) can not be ensured as secure key due to Nonce's non-freshness . The Intruder can capture messages and delete them. It may result in fail in key agreement between two agents.
- It is possible to fake Certificate's identity easily. It can be sniffed by intruder.

Finally, the problem is concerned with replay attacks that can be mounted against the protocol. The below describes attack scenario of original BCY.

Figure 2. Attack scenario of original BCY

Consider an attacker who is malicious towards a mobile merchant M. The attacker may record communications, including the moves in this protocol, between the client and the mobile merchant. Some time after M and C complete their communication session, the attacker can initiate a communication with the client C by replaying messages previously sent to C by old session key. Clearly the messages will pass all the tests by the client, which results in the attacker being successful in impersonating the mobile merchant M.

4.2 Analysis of Carlsen BCY Protocol

4.2.1 Modelling Carlsen using Casper

The Carlsen, revised-BCY, suggested that an expiration time, TSM, be included in M's certificate and a random nonce, rM, be included in the first and second protocol messages in order to protect session key.

4.2.2 Protocol Goals

The protocol goals remain unchanged from the original BCY version.

4.2.3 Result of Verification

Carlsen omits M's identity from the certificate. Thus, V's certificate was changed from $\{M, Kmd+, Kmm+\}Kca-$ in the original BCY protocol to $\{Kmd+, Kmm+, TSM\}Kca-$. The Carlsen BCY protocol also doesn't satisfy the security properties below:

Figure 3. The Carlsen BCY protocol description

- At the step1, this protocol wants to prevent M's Certificate from being reused including M's timestamp at the Certificate in the original BCY. Its result is success. But At the step2, the Intruder can replay attack due to not having timestamp of C's Certificate
- This protocol want to confirm authentication by sending M's nonce (rM) at the step 1, 2 in the original BCY. But it results in fail in key agreement. Because C can not believe the freshness of M's session key
- It is possible to be sniffed M's Certificate due to not having M's identity at the step1.

4.3 Analysis of Mu-Varadharajan BCY Protocol

4.3.1 Modelling Mu-Varadharajan using Casper

Mu and Varadharajan [6] suggested further modifications to overcome some problems associated with Carlsen's BCY protocol. Firstly, they reintroduced M's identity into the certificate. Secondly, an expiration time TSC is introduced into C's certificate to avoid attacks where an intruder replays a compromised certificate. Thus, the certificates were changed to $\{M, Kmd+, Kmm+, TSM\}Kca-$ and $\{C, Kc+, TSC\}Kca-$.

4.3.2 Protocol Goals

The protocol goals remain unchanged from the original BCY version.

Figure 4. The Mu-Varadharajan BCY protocol description

4.3.3 Result of Verification

- At the step2, this protocol wants to prevent C's Certificate from being reused including C's timestamp at the Certificate in the Carlsen BCY. Its result is success.
- M's timestamp is introduced in the Carlsen BCY. It is possible that C prevents Intruder from replay attack. But in reverse, it can be replay attack from C without C's timestamp. In this protocol, C's timestamp is introduced into C's Certificate to prevent replay attack. Its result is success.
- M's nonce (rM) is still not encrypted in this protocol. It ensured confirmation of authentication by sending M's nonce at the step1,2 in the Carlsen BCY. But it is possible for the Intruder to reuse old session key due to not encrypting M's nonce. It effects on fail in key agreement.

5 Proposed modification to the BCY Protocols

5.1 Modelling revised Mu-Varadharajan using Casper

In light of above these failures of three protocols, these protocols generally rely on shared long-term keys between mobile devices and network servers in order to allow the generation of session keys. This means that once a shared session key is established between two mobiles across a network, symmetric encryption algorithms can protect digitally encoded speech and control information.

But This has possibility of known key attack. This attack applies if, in the event that an old session key is compromised, future session keys can also be compromised. Known key attacks are related to those discussed by Denning and Sacco with respect to the use of timestamps to prevent message replay. Such attacks are of particular concern in any environment where the probability of compromise of session keys is significantly greater than that of long-term keys.

So we propose that a number of amendments are made to the Mu-Varadharajan BCY protocol. The new protocol, which is outlined in Fig.5 and referred to as revised Mu-Varadharajan BCY protocol.

Figure 5. The revised Mu-Varadharajan protocol description

We suggest further modifications to overcome some problems associated with Mu-Varadharajan protocol. M's Nonce is encrypted using the public key of User ($\{rM\}Kc+$). This process establishes a secure channel between M and C. Because it prevents replay attack by introducing encrypted public key with two agents and ensures mutual authentication by including timestamp in Certificate respectively.

One of the basic properties of the proposed protocol is the establishment of a secret session key, which can be used to encrypt subsequent communications between two entities. The proposed protocol is also responsible for the secure transfer of payment and charging information between the Merchant and the Client.

5.2 Result of Verification

We can get the result from our experiment that would prevent an intruder from spoofing any other user.

Table 2. Requirements achieved by each of the protocols

Lists	Original	Carlsen	Mu-V.	Proposed BCY
1	*	*	*	S
2	S	S	S	S
3	*	*	S	S
4	*	S	S	S
5	*	*	*	S
6	*	*	*	S
7	*	*	*	S
8	S	S	S	S
9	S	S	S	S
10	S	S	S	S

S : Satisfy the Requirement, * : Fail the Requirement

From table 2 it can be seen that the proposed protocol meets all of the security requirements listed in section 1. This protocols were primarily designed to provide link security to protect against passive attacks on the air interface.

First of all, we encrypted Merchant's nonce to ensure confidence of session key by Client's public key. It results that modified protocol meets the requirement #1, #5, #6 among vulnerability found in Mu-V. protocol. Therefore ensuring freshness of nonce makes our protocol meet the requirement #7.

6 Discussion and Conclusions

In electronic payment systems the mobile's merchant domain server needs to be on-line at the time of the call setup in order to help establish a session key between mobiles. This means that the payment system's server needs to be ultra reliable. A solution to this problem is to use public key algorithms between mobiles. Public key protocols can be more readily used in mobile systems due to their improved computational abilities.

The main aim of this paper was to propose a secure protocol for electronic payment system. Original BCY and Carlsen BCY protocol have same vulnerabilities, i.e. key agreement, replay attack and entity authentication. The third Mu-Varadharajan BCY protocol has result from key freshness problem. Old session key is reused to enable the impersonation of a legitimate user.

However revised Mu-Varadharajan BCY protocol we proposed is giving confidence in the correctness. Especially, revised Mu-Varadharajan BCY protocol is aimed at providing mutual authentication and key agreement in electronic commerce system.

References

- [1] D.S. Wong, A.H. Chan, "Efficient and mutually authenticated key exchange for low power computing devices", Proceedings of ASIACRYPT, 2001.
- [2] G. Horn, B. Preneel, "Authentication and payment in future mobile systems", Journal of Computer Security 8, 2000, pp. 183-207.
- [3] C. Boyd, D.-G. Park, "Public key protocols for wireless communications", Proceedings of 1998 International Conference on Information Security and Cryptology, 1998.
- [4] M.J. Beller, L.-F. Chang, Y. Yacobi, "Privacy and authentication on a portable communications system", IEEE Journal on Selected Areas in Communications, 1993, pp. 821-829.
- [5] U. Carlsen, "Optimal privacy and authentication on a portable communications system", ACM Operating Systems Review 28, 1994, pp. 16-23.
- [6] Y. Mu, V. Varadharajan, "On the design of security protocols for mobile communications: Information Security and Privacy", Lecture Notes in Computer Science, Vol. 1172, Springer, Berlin, 1996, pp. 134-145.
- [7] G. Bella, F. Massacci, and L. C. Paulson., "Verifying the SET registration protocols", IEEE Journal of Selected Areas in Communications, 21(1):77.87, 2003.
- [8] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, NJ, 1985.
- [9] Bellare, M., Garay, J.A., Hauser, R., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G., Herreweghen, E.V., Waidner, M., "Design, implementation and deployment of the iKP secure electronic payment system", IEEE Journal of Selected Areas in Communications 18, 2000, pp. 611-627.
- [10] CafeOBJ: CafeOBJ web page. <http://www.ldl.jaist.ac.jp/cafeobj>, 2001.
- [11] Cox, B.; Tygar, J.D. and Sirbu, M., "NetBill Security and Transaction Protocol", In Proceedings of the First USENIX Workshop in Electronic Commerce", July, 1995, pp. 77-88.
- [12] G. Lowe, "Casper: A compiler for the analysis of security protocols", Proceedings of the 1997 IEEE Computer Security Foundations Workshop X, IEEE Computer Society, Silver Spring, MD, 1997, pp. 18-30.