Analysis of a Biometric Authentication Protocol for Signature Creation Application

A. Salaiwarakul and M.D.Ryan

School of Computer Science,
University of Birmingham, UK
{A.Salaiwarakul, M.D.Ryan}@cs.bham.ac.uk

Abstract. This paper presents an analysis of biometric authentication for signature creation application. We extend the established protocol in order to verify the two properties: secrecy and safety. We have analysed the protocol using applied pi calculus and ProVerif. The verification of the secrecy property shows that the protocol holds the biometric data securely while the verification of the safety property shows that an intruder could not deceive the application to allow her to sign any document using a legitimate user's signature.

1 Introduction

Biometric user authentication is a way to authenticate the user by using his biometric data: fingerprint, face recognition, or iris, for example. Biometric data cannot be considered a secret in the way that private keys or passwords can. In contrast with private keys, biometric data is given to possibly hostile hosts to which a user wishes to authenticate. In contrast with passwords, biometric data cannot be changed, and a user cannot conveniently choose different biometric data to present to different hosts in the way that one might use a different password for a webmail account as for a bank account. Moreover, in contrast with keys and passwords, biometric data such as user's facial characteristics and fingerprints are in the public domain, and can be captured without the user's consent or knowledge.

For this reason, protocols for biometric authentication should rely on proof of freshness of biometric data and cannot rely on its secrecy. Nevertheless, these protocols should protect its secrecy; we take the view that biometric data should be kept private as a matter of good practice. In this respect, it is rather like credit card numbers, which are not really private, since we voluntarily cite them on the phone and by unencrypted email, and allow restaurateurs and other retailers to handle the cards bearing them in our absence. Nevertheless, it seems sensible not to allow such data to be spread around without restriction. The same idea applies to biometric data. Even if user's biometric data (BD) could be captured by agents having access to smooth surfaces the user touches, or agents to whom the user authenticates, it should not be unnecessarily made easy for malicious agents to acquire it.

Processes involved in a biometric authentication could be classified as two steps: enrolment and verification. In the enrolment process, the user's registered biometric code (BC) is either stored in a system or on a smart card which is kept by the user. In

the verification process, user presents his biometric data (BD) to the system so that the biometric data will be compared with the stored biometric code. User verification can either be carried out within the smart card, a process called on-card matching, or in the system outside the card, known as off-card matching.

The on-card matching algorithm protects the user's stored biometric code. The biometric code is not necessarily transferred to the outside environment if using this type of matching. Even though the biometric data is not considered to be secret, the protocol should not reveal it without the user's agreement.

When the biometric data is used for biometric authentication, it should not only be protected from disclosure to an attacker, but also its origin should be guaranteed; this prevents an attacker presents the previously captured biometric data to the system in order to authenticate himself as the authorised user.

One of the applications that can use biometric authentication as part of the application is signature creation application. The application is used for signing a document, for example by using user's key which is stored in smart card, in order to proof the originator of the document. The application using biometric authentication protocol not only challenges with the classical problem, inappropriate disclosure of biometric data to an intruder but also the specific problem to the application, signature creation application, such as an intruder deceives the application to sign her document using an legitimate user's signature.

Our paper demonstrates a protocol that uses an on-card matching mechanism to protect the stored biometric code, and an encryption and cryptographic checksum mechanism to protect the presented biometric data. We also extend the protocol to complete the signature creation so that the safety property can be analysed. We analyze the protocol and verify the intended properties of the protocol.

2 Description of Protected Transmission of Biometric User Authentication Data for an On-card Matching Protocol

This protocol is presented by Waldmann, Scheuerman and Eckert [1]. The protocol prevents the user's biometric data from escaping from a biometric reader and protects the data packet using a cryptographic mechanism.

A signature creation application that stores the users biometric code on a smart card is used here to illustrate this protocol. This application enables the user to sign a document using his private key. The user's private key is stored on the smart card. It will be released if the user is successfully verified by using his biometric data.

The physical setup of the system is shown in Fig.1. The system consists of a PC and a terminal case. The PC contains a service application such as the signature creation application. Inside the terminal case are the security module card (SMC), tamper resistant terminal, and user card containing the user's credentials. In order to prevent fraud and interruption from an intruder, the fingerprint reader (including biometric feature extraction), secured viewer and smartcard interaction module (SIM) are embedded in the tamper resistant terminal.

Let us describe the biometric authentication process that takes place when the user wishes to sign a document using his signature. The user, Bob, uses his PC to open the

signature creation application and he is shown a document via the secured viewer. If he agrees to sign it, he will present his biometric data (in this example, his fingerprint) to the sensor. The security protocol is performed via SIM in order to validate the user (detailed description in the next section). If the user verification is successful, the user card will release Bobs signature to sign the document. The signing process is performed inside the tamper resistant terminal.

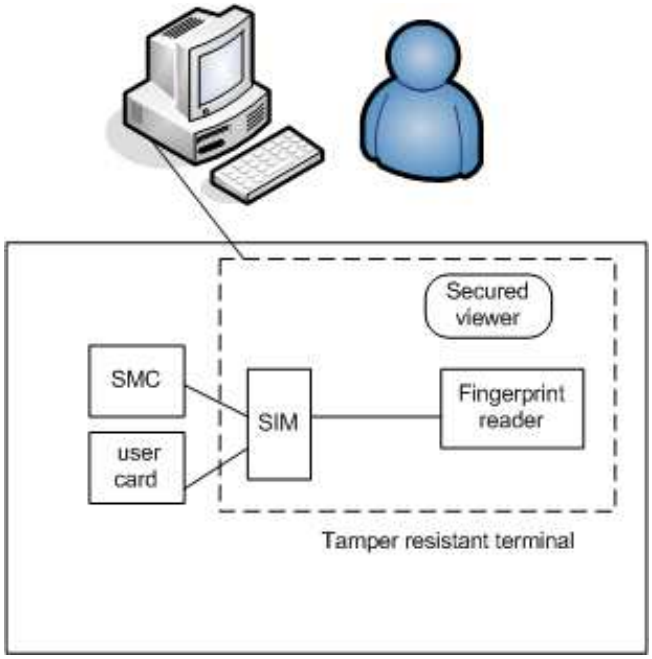

Fig. 1. The physical setup of how components are connected

Fig.2 illustrates the processes involved in the security protocol. The three components of the system that perform the security functions are the SMC, the SIM and the user card.

The SMC is responsible for generating the session keys for encryption and decryption of biometric data. It is a plug-in card to give flexibility to the manufacturer of the service system. For example, the certificate of the service system can be changed easily, if necessary. The user card holds the user's biometric code and other user credentials such as the user's signature if the service system is used for signature creation. The SMC and the user card cannot communicate directly and are outside the tamper resistant terminal so the SIM is responsible for the security protocol between the SMC and the user card.

Let us briefly describe how the protocol proceeds. The legitimate user, Bob, holds his user card, which stores his biometric code and private key. Before user authentication, the SMC and the user card perform mutual authentication, e.g. by using the

Needham Schroeder Lowe protocol; if this succeeds, they will calculate the session keys $SK.CG$ and $SK.CC$, and the initial value of the send sequence counter (SSC).

Apart from the new generated session keys, the SMC holds static keys, *SK.CC and *SK.CG, which are generated by the manufacturer. These keys are also installed in the SIM.

The CC which is included in the notation denotes the cryptographic checksum for ensuring data integrity while the CG represents the cryptogram which is used for data encryption. Consider the following example that represents the message M, which is encrypted using key *SK.CG and then hashed using *SK.CC as key.

$$\{\mathrm{M}\}_{*SK.CG} \,||\mathrm{H}_{*SK.CC}(\{\mathrm{M}\}_{*SK.CG})$$

The receiver of the above message could check the integrity of the received message by performing the hash function of the first argument and then comparing the result with the second argument. Moreover, the receiver could get the content of the message by performing message decryption using static key *SK.CG. The same idea applies to the message that uses the session key for encryption and hash function.

In order to sign a document using his electronic signature, Bob is shown the document via the secured viewer. The secured viewer is proposed in consideration of preventing an attacker that could interfere with the signal of the PC monitor. It is installed in the tamper resistant terminal so that an intruder could not interfere. If he agrees to sign it, he presents his fingerprint to the biometric reader that is situated in the tamper resistant terminal. To prevent replay of the presented biometric data, the SMC invents a fresh random nonce and sends it to the SIM to verify that the received message is fresh.

Before sending Bob's biometric data to the SMC, the SIM encrypts it with $^*SK.CG$ and also carries out the cryptographic checksum of encrypted user's biometric data using *SK.CC and the nonce.

After the SMC receives the message, it verifies its authenticity and validity. If this check is successful, it will send a reply "OK" back to the SIM. The SIM then sends a sign command to user card.

The SMC calculates the cryptogram of the biometric data, and the cryptographic checksum of the cryptogram along with the user command by using the session keys (*SK.CG* and *SK.CC*) and sends this packet to the SIM. On receipt, the SIM forwards this data package to the user card. The user card deciphers the package, checks the correctness, and verifies the received biometric data against the stored biometric code. Then the user card sends the result of the verification as well as the cryptographic checksum of the result back to the SMC via the SIM. The SMC verifies the correctness of the data it receives and the result of the user's verification. A positive result leads to the agreement of the signing process by the user card, the detail of which is beyond the scope of this protocol.

3 Completing the Protocol : Creating the Signature

It is necessary to extend the protocol, as described in the previous section, in order to completely verify the protocol and its properties. Here, we give some observations on the protocol and explain how the protocol should be completed in signature creation.

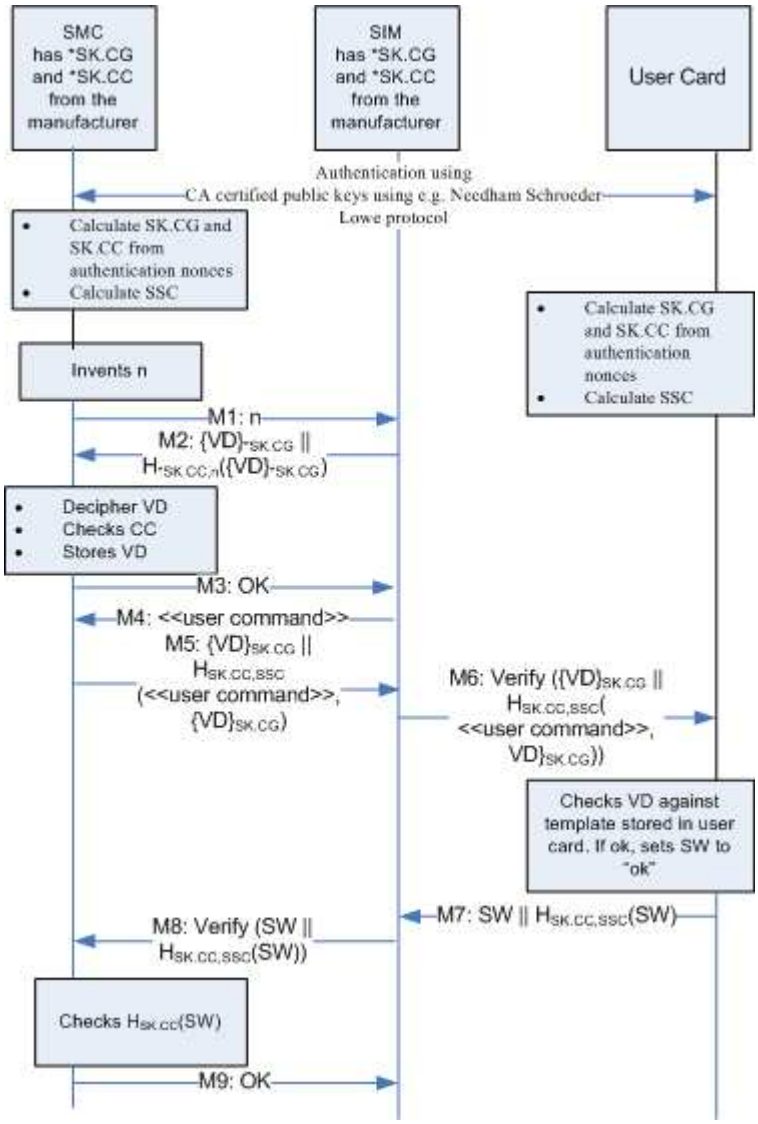

Fig. 2. The message sequence chart of [1]

One of the purposes of the protocol is to enable the user to sign the document using the user's stored private key stored on the smart card. To verify the protocol and guarantee correctness, the protocol has to be extended. After the user biometric authentication succeeds (more precisely after the message M9 has finished), in order to sign a document using the user's key, the SIM sends a hash value for the document to the SMC. The hash value of the document is encrypted with one of the static keys, **SK.CG*. The SMC deciphers it and forwards the hash of the document, which is encrypted by the session key (shared by the SMC and the user card) *SK.CG* to the user card. The user card signs the hash value of the document and sends it back to the SIM. The document is signed only if the user is satisfied with the document he views from the terminal (via the secured viewer in the tamper resistant terminal). In accordance with signing a document, the rest of the protocol should be completed as shown in Fig.3.

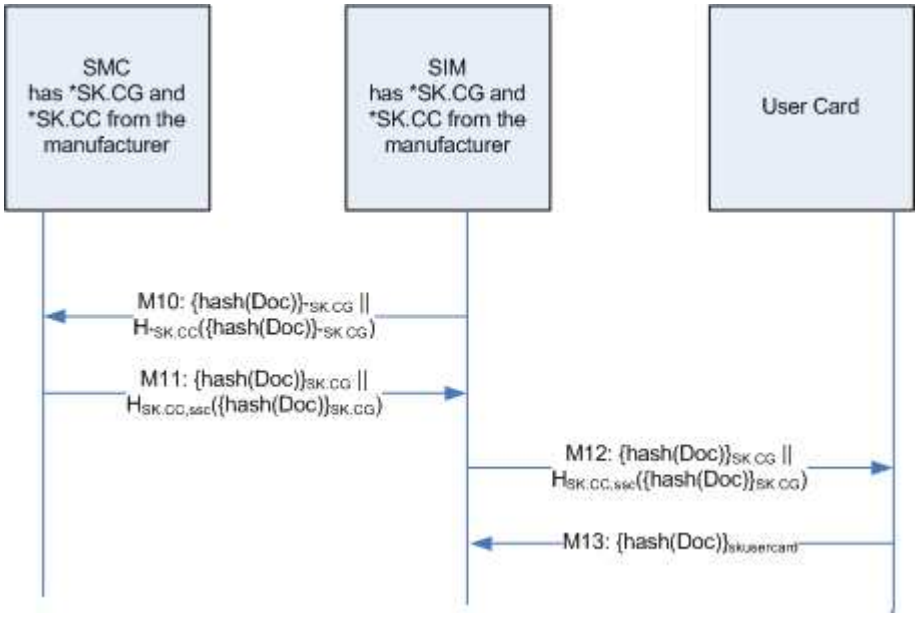

Fig. 3. The message sequence chart for signature creation

4 Capabilities of the Attacker

A Dolev-Yao style attacker can generate, mix, and replay messages transmitted in channels [2], even in cabling communication.

Biometric authentication uses a biometric reader in order to retrieve the user's biometric data. It is connected to the system via a USB cable. In addition, if a smart card is used to store the user's biometric code, a smart card reader is also connected to the system.

Although a smart card is a tamper resistant device in which the stored value cannot be modified without using the appropriate protocol, an attacker can still listen to the communication signal between smart card and reader. There is a prototype model that can be used as an example to describe this concept [4]. The smart card itself does not have a display; it needs another device then, i.e. a smart card reader, to show any value to the user. Communication between the user and the smart card must take place via the reader. If it is modified by a corrupted merchant, information flow between the smart card and the card reader can be intercepted. So if the smart card is used for storing the biometric code for user verification, the attacker can listen to the messages and capture this data easily.

5 ProVerif Model

ProVerif is a protocol verifier developed by Bruno Blanchet [5], that is able to take as input a variant of the applied pi calculus [6]. This tool has been used to prove the security properties of various protocols [7–10]. It can be used to prove secrecy, authenticity and strong secrecy properties of cryptographic protocols. It can handle an unbounded number of sessions of the protocol and an unbounded message space. The grammar of processes accepted by ProVerif is described briefly below.

In order to verify properties of a protocol, query commands may be executed. The query 'attacker: m' is satisfied if an attacker may obtain the message m by observing the messages on public channels and by applying functions to them. The query $ev : f(x_1, \ldots, x_n) \Rightarrow ev : f'(y_1, \ldots, y_m)$ is satisfied if the event $f'(y_1, \ldots, y_m)$ must have been executed before any occurrence of the event $f(x_1, \ldots, x_n)$.

An advantage of using ProVerif as a verifier is that it models an attacker which is compliant with the Dolev-Yao model automatically. We do not need to explicitly model the attacker.

P, Q, R	processes
0	null process
P |Q	parallel composition
new n; P	name restriction
new x; P	variable restriction
if M = N then P else Q	conditional
event x; P	event launch
let x = M in P	replace the variable x with the term M in process P
in(M,x); P	message input
out(M,x); P	message output

5.1 Signature and Equational theory

In our model, ProVerif uses the following signatures and equations for calculating and solving messages. The function *getkey* retrieves the public key of the particular identity from the public key table which is stored in the server. In addition, *getkey* is coded as a private function to prevent components, other than those involving the system, using

this function. The symmetric encryption and decryption functions are utilising using *senc* and *sdec* respectively while the asymmetric encryption and decryption functions are done using *enc* and *dec*. In order to resolve an encrypted message, ProVerif uses the decryption equation to decrypt a message using a recognized key. The signed messages are extracted using the *checksign* equation. In our ProVerif model, some messages are hashed using such hash functions as h, g, or f. These hash functions implement two arguments; one is a key while the other one is a message content.

```
    (*Signature*)
private fun getkey/1. (*key retrieval*)
fun sk/1.             (*session key*)
fun senc/2.           (*symmetric encryption*)
fun sdec/2.           (*symmetric decryption*)
fun enc/2.            (*encryption*)
fun dec/2.            (*decryption*)
fun sign/2.           (*signature *)
fun checksign/2.      (*recovering signature*)
fun pk/1.             (*public key*)
fun host/1.           (*host function*)
fun h/2.              (*hash function*)
fun g/2.              (*hash function *)
fun f/2.              (*hash function*)
fun hashDoc/1.        (*hash function for a document*)
    (*Equation*)
equation getkey(host(x)) = x.
equation sdec(senc(x,K),K) = x.
equation dec(enc(x,pk(y)),y) = x.
equation checksign(sign(x,y),pk(y)) = x.
```

5.2 SMC Process

This process represents the operations and message transmission associated with the SMC. First, the SMC performs mutual authentication with the user card. It is not stated how this is done in [1]; we have used the Needham Schroeder Lowe protocol. If successful, it will calculate session keys (*SK.CG* and *SK.CC*) and SSC from the authentication nonces.

The user's biometric data package is received and deciphered. Next, it encrypts and calculates the cryptographic checksum of the biometric data, and sends it to the SIM. If the user's authentication is successful, it will receive the verification result back from the user card, send the reply back to the SIM, and wait for the hash of the document to be sent back. After receiving the hash of the document, it will verify the validity of the document, encrypt, and calculate the cryptographic checksum of the hash of the document with the session keys *SK.CG* and *SK.CC* respectively.

```
let SMC =
(* Authentication using Needham Schroeder
```

```
   Lowe Protocol *)
  in(c,hostX);
  let hostSMC = host(pkSMC) in
  out(c,(hostSMC,hostX));
  in(c,ms);
  let(pkX,=hostX) = checksign(ms,pkS) in
  new Na;
  out(c,enc((Na,hostSMC),pkX));
  in(c,m);
  let(=Na,NX2,=hostX) = dec(m,skSMC) in
  out(c,enc(NX2,pkX));
  let SKCG = h(Na,NX2) in
  let SKCC = g(Na,NX2) in
  let SSC = f(Na,NX2) in
(*  After the authentication succeeds*)
  new n;
  out(c,n);
  in(c,(m1,m2));
  if h((sSKCC,n),m1) = m2 then
  (
    let BDreceived = sdec(m1,sSKCG) in
    out(c,OK);
    in(c,m13);
    let BDsenc = senc(BDreceived,SKCG) in
    out(c,(BDsenc,h((SKCC,SSC),(m13,BDsenc))));
    in(c,(m8,m9));
    if h((SKCC,SSC),m8) = m9 then
      if m8 = success then
        out(c,OK);
    in(c,(m16,m17));
    if h(sSKCC,m16) = m17 then
    (
       let M1 = sdec(m16,sSKCG) in
       let M1senc = senc(M1,SKCG) in
       out(c,(M1senc,h((SKCC,SSC),M1senc)))
     )

   ).
```

5.3 SIM Process

In the real-life model, the user is presented with the document that he will sign using his key on the secured viewer. If he agrees to sign it, then he places his biometric data on the biometric reader which is installed in the SIM. Therefore, for ease of understanding, in the ProVerif model, the document that the user wants to sign is created within the SIM. The SIM receives a fresh random nonce and then sends the user's biometric data

encrypted with *SK.CG*, along with the cryptographic checksum created using *SK.CC*, and the nonce, to the SMC. When the SIM receives the signal from the SMC that the user's biometric data is correct, it sends the user's command authorizing the signature as a reply.

The SIM carries out the security protocol between the SMC and the user card by receiving and forwarding messages between those two components. After the user's authentication succeeds, the SIM generates the hash value of the document, encrypts it, and calculates its cryptographic checksum. It then sends these data to the SMC. The SMC is waiting to receive the document which is to be signed by the user card.

```
let SIM =
(* communiation messages start *)
   in(c,nx);
   in(userChBD,BD);
   in(userChText,userText);
   let BDsenc = senc(BD,sSKCG) in
   out(c,(BDsenc,h((sSKCC,nx),BDsenc)));
   in(c,m20);
   if m20 = OK then
   (
     out(c,userCommand);
     in(c,(m4,m5));
     out(c,(m4,m5));
     in(c,(m6,m7));
     out(c,(m6,m7));
     in(c,okm);
     if okm = OK then
     (
     let digest = senc(hashDoc(userText),sSKCG) in
     out(c,(digest,h(sSKCC,digest)));
     in(c,(m13,m14));
     out(c,(m13,m14));
     in(c,m15)
     )
   ).
```

5.4 UserCard Process

First, the user card executes the mutual authentication with the SMC. Then, it calculates the session keys *SK.CG* and *SK.CC*, and *SSC*. Next, the user card awaits a package of the user's biometric data. It verifies the validity and authenticity of the received message. It decrypts the package and verifies the received biometric data against the stored biometric code. If they match, the verification result is set to be successful. In our ProVerif model, they are always set to match so that we can verify the protocol until the end (the signing process of the document) without blocking through failure in biometric verification. The verification result is sent out along with the checksum of the result which is

computed using *SK.CC*. Then, it acquires the hash of the document and signs it using the user's signature, which is stored in the user card.

```
let UserCard =
(* Authentication using Needham Schroeder
   Lowe Protocol *)
  in(c,m);
  let(NY,hostY) = dec(m,skUserCard) in
  let hostUserCard = host(pk(skUserCard)) in
  out(c,(hostUserCard,hostY));
  in(c,ms);
  let(pkY,=hostY) = checksign(ms,pkS) in
  new Nb;
  out(c,enc((NY,Nb,hostUserCard),pkY));
  in(c,m3);
  if Nb = dec(m3,skUserCard) then
    let skcg = h(NY,Nb) in
    let skcc = g(NY,Nb) in
    let ssc = f(NY,Nb) in
(* The authentication succeeds,
   message communication starts *)
  in(c,(m10,m11));
  if  h((skcc,ssc),(userCommand,m10)) = m11 then
  (
      let BDsdec = sdec(m10,skcg) in
      if BDsdec = BD then
        let SW = success in
        let m12 = h((skcc,ssc),SW) in
        out(c,(SW,m12));
        in(c,(m18,m19));
        if h((skcc,ssc),m18) = m19  then
        (
          let M2 = sdec(m18,skcg) in
          out(c,sign(M2,skUserCard))
         )
   ).
```

5.5 U process

To demonstrate user interaction in the protocol, we model the process U. The user receives a document and checks it. If he is satisfied with the contents, he will place his finger on the reader.

```
let U =
  in(TextCh,t);
  if t = Text  then
```

```
   out(userChBD,BD);
   out(userChText,t).
```

5.6 S process

In order to authenticate identities using the Needham Schroeder Lowe protocol, the server is modelled using process S, which is used for providing a public key to the identity. The server process receives the request from the identity a that it wants to communicate with identity b. The server process retrieves the public key of the identity b from the server's public key table. It then signs the package of the public key and the identity b using its private key and outputs to the channel. The receiver of this package ensures that the public key it received comes from the genuine server by checking the signature. The public key will be used later in the receiver process in order to perform the Needham Schroeder Lowe authentication which needs the public keys for decryption.

```
let S =
  in(c,m);
  let(a,b) = m in
  let sb = getkey(b) in
  out(c,sign((sb,b),skS)).
```

5.7 Main Process

In the main process, the static keys **SK.CG* and **SK.CC*, and the private keys of the SMC, the SIM and the user card are created. Private channels for the user's document and biometric data are set up. The public keys of each of the components are distributed on the public channels. There are many *SMC*, *SIM*, *user card*, and *U* processes in the system. The U processes represent Alice and Bob who input the documents and the biometric data.

```
process
new userChBD;
new userChText;
new AliceTextCh;
new BobTextCh;
new BobBD;
new AliceBD;
new sSKCG;
new sSKCC;
new skSMC;
let pkSMC = pk(skSMC) in
out(c,pkSMC);
new skBobUserCard;
new skAliceUserCard;
let pkBobUserCard = pk(skBobUserCard) in
```

```
let pkAliceUserCard = pk(skAliceUserCard) in
out(c,pkBobUserCard);
out(c,pkAliceUserCard);
new skS; let pkS = pk(skS) in
out(c,pkS);
!out(AliceTextCh,AliceText);
!out(BobTextCh,BobText);

((!S) | (!SMC) | !SIM |

(let TextCh = AliceTextCh in
 let Text = AliceText in
 let BD = AliceBD in !U) |

(let TextCh = BobTextCh in
 let Text = BobText in
 let BD = BobBD in !U) |

(let skUserCard = skAliceUserCard in
 let BD = AliceBD in !UserCard) |

(let skUserCard = skBobUserCard in
 let BD = BobBD in !UserCard))
```

6 Analysis of the Protocol

A signature application protocol is used as an example of using biometric authentication in order to verify the user who uses the smart card to sign a document that he is the correct user.

An intruder could interfere between the smart card and smart card reader to try to listen to the communication and capture user's biometric data [4]. Moreover, an intruder could play with messages to lead a legitimate user to sign her messages.

6.1 Secrecy of the Biometric Data

This property is used to verify that the protocol does not reveal the user's biometric data without permission. Even though we consider the biometric data to be public, it is good practice to keep it private so that no one else except the sender and the receiver knows the content of messages. The protocol should not allow the data presented by user to be announced to others. Analysis of this property verifies whether an attacker can intercept the biometric data when it is sent from one component to another. In our model, the biometric data is represented as BobBD, the biometric data of legitimate user, Bob. The ProVerif implementation is:

```
query attacker : BobBD
```

ProVerif responds to the *query* command by using a Dolev-Yao style attacker to attempt to compose or decompose messages and establish whether an attacker can reach the biometric data (BobBD).

6.2 Safety

This property is used to verify that the document is signed only if the user's authentication is successful and that only the legitimate user signs the agreed document. We analyze this by checking whether an attacker can sign someone else's documents using the signature of the legitimate user. From our assumption, we check whether an intruder, Alice, can intercept messages to make the legitimate user, Bob, sign her document. The ProVerif implementation is:

```
query attacker : sign(AliceText,skBobUserCard)
```

ProVerif analyzes this *query* command by checking whether an attacker can sign AliceText (which is not the document that is shown to the legitimate user, Bob) using Bob's signature. We assume that the user's signature is the same as the private key of the user card that the user holds.

7 Conclusion

We have analyzed two properties of the protocol: *secrecy* and *safety*

Although we consider the biometric data to be public, we still need to verify that the protocol which uses this resource does not reveal it without the user's consent. The data should not be revealed to anyone who is neither the sender nor the intended receiver. The positive result of the verification illustrates that the presented biometric data remains private within the protocol and an attacker cannot acquire it.

The positive result of the safety property shows that the protocol guarantees that even if the presented biometric data is captured from the previous submitted data packet, it cannot lead the user card to sign a document that the user is not willing to sign.

Since the biometric data can be captured (as explain in section 1), the hardware is required to be capable of ensuring that the biometric data has come from the user's live presentation, not (for example) a fake rubber finger.

The properties of the biometric authentication protocol should be proposed and stated clearly when creating a biometric authentication protocol. In future work, we will consider which properties are desirable of a biometric authentication protocol.

References

1. Waldmann, U., Scheuermann, D., Eckert, C.: Protected Transmission of Biometric User Authentication Data for Oncard-Matching. ACM Symposium on Applied Computing (2004) 425–430
2. Dolev, D. and Yao, A.C.: On the Security of Public Key Protocols. In Proceedings of 22nd IEEE Symposium on Foundations of Computer Science (1981) 350–357

3. Gobioff, H., Smith, S., Tygar, J. D., Yee, B.: Smart Cards in Hostile Environments. 2nd USENIX Workshop on Electronic Commerce (1996)
4. Bond, M.: Chip and Pin (EMV) Point-of-Sale Terminal Interceptor. availability = internet(http://www.cl.cam.ac.uk/ mkb23/interceptor/). (2007)
5. Blanchet, B.: ProVerif : Automatic Cryptographic Protocol Verifier User Manual (2005)
6. Abadi, M., Fournet, C.: Mobile Values, New Names, and Secure Communication. POPL 2001.
7. Abadi, M., Blanchet, B., and Fournet, C.: Just Fast Keying in the Pi Calculus. ACM Transactions on Information and System Security (TISSEC), 10(3):1-59, July 2007.
8. Abadi, M., and Blanchet, B.: Computer-Assisted Verification of a Protocol for Certified Email. Science of Computer Programming, 58(1-2):3-27, October 2005. Special issue SAS'03.
9. Kremer, S., Ryan, M.: Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In Proceedings of the European Symposium on Programming. Lecture Notes in Computer Science 3444. Springer Verlag (2005) 186-200
10. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and Receipt-freeness in Electronic Voting. In 19th Computer Security Foundations Workshop. IEEE Computer Society Press (2006)
11. Prabhakar,S., Paankanti,S., Jain, A.K.: Biometric Recognition: Security and Privacy Concerns. IEEE Security & Privacy (2003) 33-42
12. Chen, L., Person, S., Proundler, G., Chen, D., and Blanceheff, B.: How can you trust a computing platform? Proceedings of Information Security Solutions Europe Conference (ISSE 2000)
13. Davida,G.I., Frankel, Y. and Matt, B.J.: On Enabling Secure Application Through Off-line Biometric Identification. Security and Privacy. IEEE Symposium (1008) 148-157
14. Matsumoto, T., Matsumoto, H., Yamada, K., Hoshino, S.: Impact of Artificial Gummy Fingers on Fingerprint Systems Proceedings of SPIE Vol.4677. Optical Security and Counterfeit Deterrence Techniques IV (2002)