Biometric Transaction Authentication Protocol: Formal Model Verification and "Four-Eyes" Principle Extension

Daniel Hartung and Christoph Busch

Norwegian Information Security laboratory
Faculty for Computer Science and Media Technology
Gjøvik University College
Teknologivn. 22, N-2802 Gjøvik, Norway
{daniel.hartung, christoph.busch}@hig.no

Abstract. The BTA protocol for biometric authentication of online banking transactions is extended to allow for multiple person authenticated transactions. In addition a formal specification is given, the protocol is modelled in the applied pi calculus and the security properties of data and person authentication as well as non-repudiation are verified using the tool ProVerif.

Keywords: Online Banking; Transaction Authentication; Payment Scheme; Non-Repudiation of Origin; ProVerif; Applied Pi Calculus; Biometric Systems; Template Protection

1 Introduction

The need for secure authentication methods is evident when looking at the assets transferred over the Internet, the level of interconnectedness and the posed threats: a recent example of malware affecting vital, well-protected infrastructures is the Stuxnet computer worm. And even more, badly protected client computers are exposed to threats: malware on clients endanger especially online banking transactions, whose manipulation promise rapid financial gain to attackers. This has to be prevented. However from a service providers view, not only the integrity of the data, but also its origin is to be guaranteed, which will be referred to as data and person authentication throughout the paper. Until now, no method for online banking transactions features non-repudiation of origin (natural person). One reasonable solution to this problem is the use of biometric systems, but not without raising threats to the users privacy.

In [5] a protocol was proposed that addresses the aforementioned problems, it uses a system for biometric person authentication using so called Privacy Enhancing Technologies (PETs) or Template Protection to authenticate online banking transactions without revealing the sensitive biometric data. At the same time the transaction data has to be authentic in order to get executed by the banking server side. These properties hold true even if the client is considered to be insecure and possibly controlled by an attacker.

The BTA protocol – Biometric Transaction Authentication Protocol – is summarized in the next section. It is modelled in section 3 using the applied pi calculus [6] and its security properties are verified using the tool ProVerif [3] in section 4. Before concluding the paper, an extension of the protocol, enabling multi-user, multi-modal as well as multi-factor authentication of single transactions, is given in section 6.

2 BTAP Wrap-Up

The goal of BTAP [5] is to enable data and person authentic online banking transactions on insecure client computer environments. To reach this goal a biometric subsystem has to be combined with classic cryptographic functionality. The critical transaction authentication is sourced out on a tamper-proof biometric transaction device (*BTD*) with limited functionality that can be certified using information technology security evaluations. The other different parties that communicate in the protocol are shown in figure 1: the customer using a potentially insecure client computer running a banking software (*BSW*) and a trusted online banking server (*OBS*).

Within the first phase of the protocol, the user is enrolled on the *BTD* using a biometric identifier and a pre-shared secret key (*SBV*). The user can afterwards conveniently initiate a transaction on the client as it is done nowadays using e.g. the online portal of the bank. The transaction information is then shared with the *OBS* and the *BTD*. On the *BTD* the information is displayed within the trusted environment, the user has to check and verify the data by presenting his or her biometric trait(s) to the sensor of the *BTD*. A seal *TOS*' is created within the *BTD* over the transaction data using the pre-shared key, that is released by the biometric sample. This seal is sent to the *OBS*, which can then check the authenticity of the transaction data as well as the authenticity of the transaction initiator – only in the case of a successful verification of the seal, the transaction is confirmed and executed.

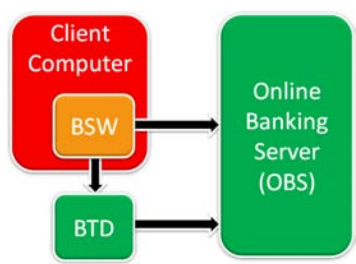

Fig. 1. Threat scenario: online banking SW (*BSW*) resides on possible malware controlled client environment and communicates with trusted online banking server (*OBS*) as well as with a secure biometric transaction device (*BTD*).

2.1 Information Flow Enrolment and Verification

The protocol involves more complex procedures inside the building blocks. Within the *BTD* the biometric subsystem is found, it covers the process of enrolment and verification that are inspired by the Helper-Data-Scheme [8] for privacy protection, which performs a fuzzy commitment. For the enrolment, a biometric sensor inside the *BTD* captures the biometric sample multiple times, extracts a fixed-length bit feature vector, which is then analyzed for reliable positions. The resulting reliable bit vector

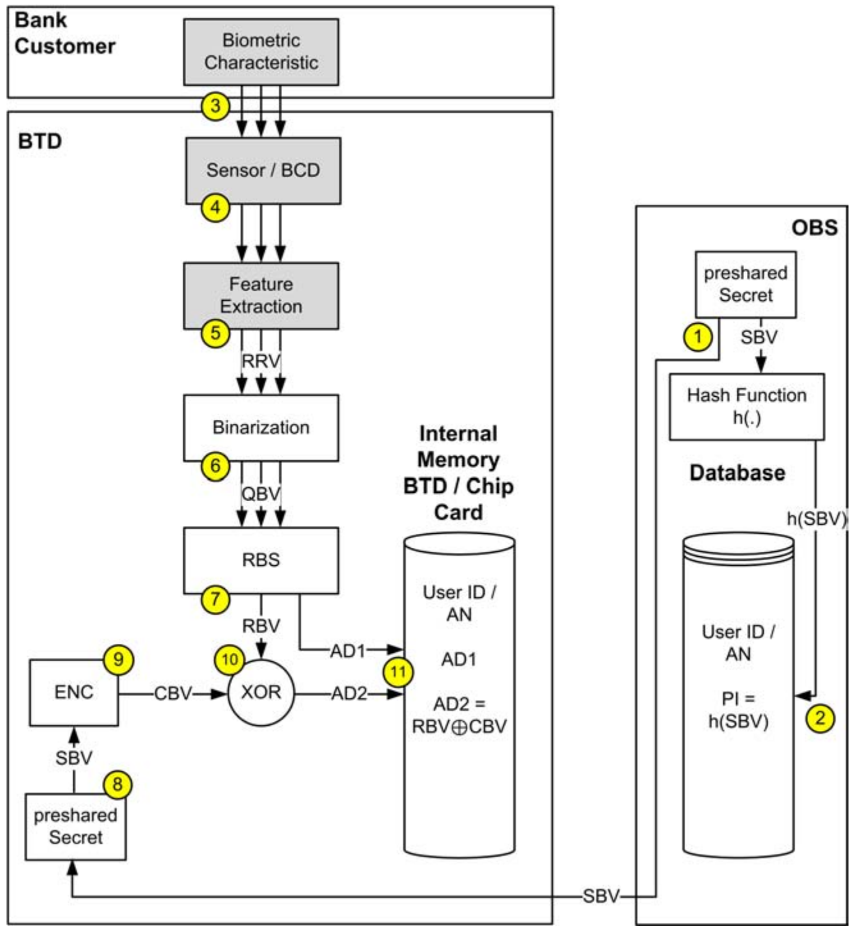

Fig. 2. Information flow of the enrolment protocol.

(RBV) is fused using the XOR-function ($\oplus$) with an error-encoded version of a pre-shared key (CBV = ECC(SBV)) that has the same length. Correcting errors using the decoding DEC of the ECC makes it possible to cope with the noise caused by the variability in the biometric information. The information stored on the BTD are not revealing any sensitive biometric information: pseudo identifier $PI = \text{hash}(SBV)$, auxiliary data $AD1$ = indexes of reliable positions in the feature vector, auxiliary data $AD2 = CBV \oplus RBV$. Figure 2 depicts the enrolment process of binding an identity to a pre-shared secret key, this process is modelled in a simplified way as described in section 3.6. Note that the pseudo identifier can be renewed or exchanged to enable revocation in a biometric system, which is not possible if the biometric information itself was used for the verification of identity. Furthermore no cross matching of different template protected biometric databases can succeed if the secret SBV is chosen independent from each other. Potentially sensitive biometric data is never stored or decrypted for comparison in its original form.

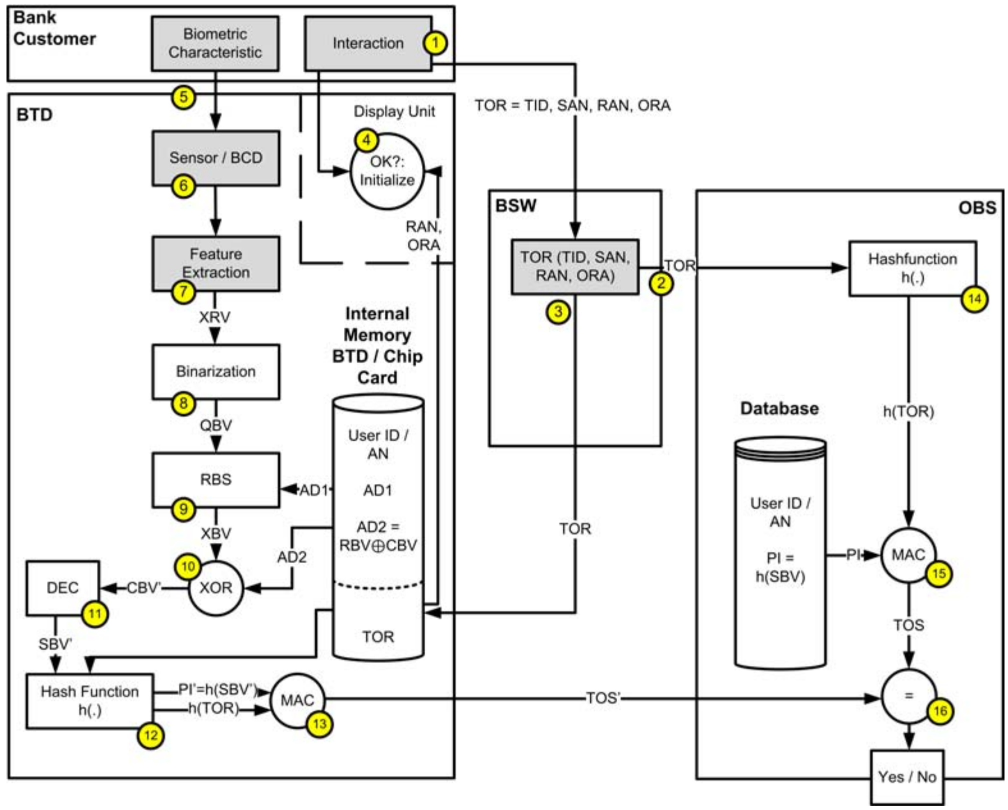

Fig. 3. Information flow of the transaction verification protocol in the core BTAP.

After this step, transactions can be authenticated as shown in Figure 3. Inverting the enrolment process is releasing the hash value of the pre-shared secret: the data subject presents the biometric trait, a biometric sample is generated, features are extracted. The helper data is loaded, so the system is able to extract the bits of the fixed-length feature vector at positions that should be reliable for the enrolled data subject. The resulting reliable bit vector XBV is releasing the key if the error correction capabilities ϵ (in bits) of the used code is higher than the amount of single bit errors $|(XBV \oplus RBV)|$ occurred during the feature extraction step:

$$\begin{gathered} AD2 \oplus XBV = (CBV \oplus RBV) \oplus XBV \\ = CBV \oplus (RBV \oplus XBV) = CBV' \\ \text{with } |(RBV \oplus XBV)| < \epsilon \\ \Rightarrow SBV = DEC(CBV) = DEC(CBV') = SBV' \end{gathered}$$

The hash value of the extracted secret bit vector SBV' is identical to the stored value PI=hash(SBV) if the enrolled biometric sample was presented and the noise could be compensated using the error correction decoder function DEC. The seal TOS'/TOS can be computed over the transaction data TOR (transaction identifier TID, sender account number SAN, receiver account number RAN, ordered amount ORA) using the keyed message authentication code function:

$$TOS' = mac(hash(TOR),\ hash(SBV'))$$

and accordingly on the server side

$$TOS = mac(hash(TOR),\ hash(SBV)).$$

2.2 Usage Scenario

The usage scenario of BTAP is seen in high value transactions like in the inter-banking sector, requiring a maximum level of security – the costs of enrolling the system in such an environment is negligible. Nonetheless since there is the need for secure authentication methods, BTAP could also be deployed in large scale, as in personal online banking transaction services, since the fixed cost for the BTD and the infrastructure would amortize considering the loss due to malware triggered false transactions over time.

3 Formal model

This section describes the formal method that was used to model BTAP and to analyse its security properties. The considered attacker model is sketched, the intended security properties are defined. Then the protocol is described using the exchanged messages as well as the applied pi calculus. The verification process based on the formal model is given in the end of this section.

3.1 Applied Pi Calculus and ProVerif

The applied pi calculus is a generalized version of the spi calculus [1], which itself is an extension of the pi calculus [6]. The pi calculus is a process calculus with the goal to formally describe concurrent systems, whose configuration may change during execution. Its variants are specifically designed to analyse and verify security properties of cryptographic protocols. The tool ProVerif was developed by Blanchet et al. [3] and it supports automated reasoning for applied pi calculus processes. It translates the protocol description into Horn clauses and acts upon them as a resolution prover. ProVerif fully automatically tries to prove security properties, its outcome can be either one of the following: robust safety can be proven, an attack as counter example is found, or it can neither prove or disprove robust safety according to the property. The protocol is modelled and verified using ProVerif, one advantage of using the tool: the Dolev-Yao attacker model, which is described in the next section, is specified and can be used directly.

3.2 Attacker Model

We assume the Dolev-Yao attacker model [4], which uses idealizations about the cryptographic primitives: an attacker can not learn from encrypted messages without the knowledge of the keys used for encryption. Changing an encrypted message without the knowledge of the key is detectable. Keys can not be guessed or learned from encrypted messages, also random numbers can not be guessed. Hash functions are collision free one-way functions. The attacker has full control over the communication channels, specifically he can: eavesdrop, inject and redirect messages. Furthermore he can generate keys and random numbers, as well as apply cryptographic primitives on what he learned.

3.3 Intended Security Properties

The intended properties of the BTA protocol are:

- *Authentication*: of the transaction data (integrity), the transaction initiator (proof of identity).
- *Non-Repudiation of Origin*: a valid transaction can not be repudiated by the initiator.
- *Secrecy*: the pre-shared secret and the sensitive biometric information stay secret.

Note: secrecy of the transaction data itself can not be assured if the client computer is compromised, and is therefore not covered in the core protocol. Additionally the internal BTD process is not modelled according to the applied pi calculus. Using the security assumptions, we model an idealized version of it.

3.4 Security Assumptions

The security assumptions for the verification of BTAP are listed below:

- BTD (in the model B) is tamper proof: no malware infection or manipulation of the processes and the storage of the BTD are possible (Note the advantages of using the privacy enhancing technology: revocation is enabled, the templates are protected additionally, only nonsensitive data is stored, storage capacity is negligible, efficient processing of the bitstrings, no hill climbing attacks possible). BTD supports secure I/O.
- Biometric subsystem: the biometric sensor can only be spoofed with unreasonable effort (suitable for unsupervised authentication). Biometric traits are unique and can not be replicated. The feature extraction system is able to extract a feature vector close to the enrolled sample, in a way that the shared key is released correctly (see section 2.1).
- Enrolment phase is completed by the authentic person, the process is not tampered.
- Helper-Data-Scheme (HDS) is not leaking private information about the extracted biometric feature vector nor the pre-shared secret. The biometric entropy is high enough to enable reasonable long pre-shared secrets to avoid brute force attacks.
- Online banking server *OBS*, or short S in the model: trusted and secure environment. Its public key *pkEncS* for encryption and pkSignS for signatures are publicly available.
- Client computer is considered untrusted and can be manipulated by malware.
- Secret keys are secret: pre-shared key SBV is shared[1] between server *OBS* and BTD, extracted biometric feature vector is also secret.

[1] In a real-life scenario the key could be shared using a secure independent channel. Personalized confidential (physical) mails or credentials could serve as a direct input to the BTD.

– Computational limitations are: none for the client and server, no public-key crypto for *BTD*.
– Communication channel between server *S* and client *C* (running the banking software *BSW*), unidirectional channels from *C* to the *BTD* and from *BTD* to the server.

3.5 BTAP: Message Sequence

Informally a protocol can be described by the messages that are exchanged, the core message sequence for BTAP [5] is given below, where {} indicate an encryption with a symmetric key `Kxy`, a public key `pkEncX` from X for encryption, or a signature using the private key `prSignX` from X. `X->Y` stands for a message from X to Y. The four parties are client `C`, server `S`, biometric transaction device `B` and user `U`:

```
Message 1: C->S: {(Nonce1, AN, ORA, RAN)}pkEncS
Message 2: S->C: {(Nonce1, Nonce2, AN, ORA, RAN)}prSignS
Message 3: C->B: (Nonce2, AN, ORA, RAN)
Message 4: U->B: (Ok)
Message 5: B->S: (mac(hash(Nonce2, AN, ORA, RAN), hash(SBV')))
Message 6: S->C: {hash(true, Nonce2, AN, ORA, RAN)}prSignS
```

The transaction information consists of the sender account number *AN*, ordered amount of money to be transferred *ORA*, and the receiver account number *RAN*. Nonces are random numbers that are used only once for proof of freshness. *Nonce1* in message 1 and 2 serve as server authentication, only the owner of the private signature key *prSignS* (server *S*) can decrypt message 1 and reply the correct *Nonce1* (*Nonce1* should include a simple time stamp besides the random part, that has to be checked for freshness on the server side before sending message 2). Message 1 is encrypted with the public encryption key of the server. *Nonce2* is included for the freshness of the transaction data, to avoid replay attacks and to limit the validity using a timestamp part. The transaction data received by the server as well as *Nonce1* and *Nonce2* are signed and send back to the client as message 2. The client forwards the information in message 3 to the *BTD*. The user has to check and verify the transaction data displayed on the *BTD* with his or her biometric trait(s), which is modelled simplified as message 4. The pre-shared key *SBV* is released and used to create a seal *TOS'=mac(hash(Nonce2, AN, ORA, RAN), hash(SBV'))* using a message authentication code (MAC) mechanism in message 5 with *hash(Nonce2, AN, ORA, RAN)* as the message and *hash(SBV')* as the secret key. The server confirms the transaction in message 6 only if the seal from message 5 is identical to the seal *TOS* that can be created on the server side with the information from message 1, *Nonce2*, and the pre-shared key *SBV'*.

3.6 BTAP: Model in the Applied Pi Calculus

The internal processes of the biometric key release inside the *BTD* are not modelled here, since we are assuming a secure and tamper-proof environment and an idealized biometric subsystem. An attacker has no access per definition on the internal variables and processes. In order to model the process of checking and verification of the authentic transaction data by the user, we use the following approximation: the authentic transaction data is modelled as data signed with the secret key (the reliable biometric

information XBV or equivalently RBV (see section 3.4)) of a "public-key biometric" system only known to the user and verifiable by, among others, the BTD.
The attacker can create an arbitrary number of transaction information, which is modelled as *evilRAN* and *evilORA*. As we will see in section 4, this is interesting for proving if such transaction information can be falsely authenticated.

All other protocol steps are modelled straightforward according to the message sequence shown in Sec. 3.5. The ProVerif code for the definition of functions, reductions and free names is given below. The number behind a function name is its cardinality. As primitives we need the hash-, mac-function as well as public-key crypto in this model, the destructors describe the behaviour of the abstract functions:

```
(* Constants *)
data true/0.

(* Functions *)
fun hash/1.
fun mac/2. (* with destructor checkmac/2.*)

(* Asymmetric Encryption *)
fun pencrypt/2. (* with destructor pdecrypt/2 *)
fun prv/1. (* private part of a key pair *)
fun pub/1. (* public part of a key pair *)

(* Reductions *)
reduc  pdecrypt(pencrypt(x, prv(y)), pub(y)) = x;
  pdecrypt(pencrypt(x, pub(y)), prv(y)) = x.
reduc  checkmac(mac(y, x), x) = y.

(* Security Assumptions *)
(* Public Channels / Free Names *)
free c, cs, sb, cb, ub, uc, ORA, RAN, m, m2, m3.
```

The core of the protocol model are the processes, which define the behaviour of the communicating parties using the applied pi calculus. The processes are behaving like the user (processU), the client (processC), the server (processS), the BTD (processB) as well as the attacker (processAttacker). If a message is not as expected, the 0.-process is executed (process stops).

ProcessC receives a message m on the open channel uc. m is expected to have the form of a 2-tuple, the two elements are defined as ORA and RAN in the rest of the process. A nonce (*Nonce1*) is created and send on the open channel cs (to the server) with the transaction data received in m as well as the fixed account number, all encrypted with public encryption key of S. A reply is expected on cs in the form of a 5-tuple. The values received should be signed with the private signature key of the server, and they are expected to be equal to *Nonce1*, AN, ORA and RAN. On the second position a new nonce is received, which is defined *Nonce2*. The new nonce (used as a transaction identifier) as well as the transaction data is send on the open channel cb (also to the BTD B). The last line indicates the process to be waiting for the decision of the server (without function in the model, for the notification if a transaction was successful):

```
let processC =
  in(uc, m); (* user interaction: transaction data generated *)
  let (ORA, RAN) = m in
```

```
  (new Nonce1;
  out(cs, pencrypt((Nonce1, AN, ORA, RAN), pub(secretEncS))); (* Message 1 *)
  in(cs, reply); (* Message 2 *)
  let (= Nonce1, Nonce2, = AN, = ORA, = RAN) = pdecrypt(reply, pub(secretSignS)) in
    (out(cb, (Nonce2, AN, ORA, RAN)); (* Message 3 *)
    in(cs, decision))).
```

ProcessS describes the server behaviour. It receives a message on channel *cs*, which is encrypted with the public encryption key of *S*. Its decrypted form is expected to be a 4-tuple (*Nonce1*, *SAN*, *ORA*, *RAN*). If *Nonce1* is fresh (was not received before) and its timestamp is valid, a fresh and random number is generated (*Nonce2*) and send on *cs* with *Nonce1* as proof of authenticity as well as *SAN*, *ORA* and *RAN*, all signed with the private signature key from *S*. Note: in the model the freshness check of *Nonce1* is not performed due to limitations in the abstraction of memory in the applied pi calculus. The next expected message is the seal sent on channel *sb* (from the *BTD B*). If the MAC was created using the secret pre-shared key *hash(SBV)* and using the transaction data received earlier in *m*, then the server accepts the transaction and creates a signed authentication reply over the transaction data including the nonce.

```
let processS =
 in(cs, m);
 let (Nonce1, SAN, ORA, RAN) = pdecrypt(m, prv(secretEncS)) in (* Message 1 *)
   (new Nonce2;
   out(cs, pencrypt((Nonce1, Nonce2, SAN, ORA, RAN), prv(secretSignS))); (* Message 2 *)
   in(sb, m2); (* Message 5 *)
   if checkmac(m2, hash(SBV)) = hash((Nonce2, SAN, ORA, RAN)) then
     (* Message 6 *)
     out(cs, pencrypt(hash((true, Nonce2, SAN, ORA, RAN)), prv(secretSignS)))).
```

ProcessB describes the biometric transaction device (*BTD*, here short: *B*). It receives message *m3* on channel *ub* (from the user). The message is expected to be a signed hash value of the authentic transaction data, only the party in possession of the private signature key can sign. This is a simplified model of the biometric subsystem. Message 3 is received from the (possibly malware infected) client *C*. Only if the hash of this transaction data is equal to the received signed hash, the seal (keyed MAC) is created over the message *m*:

```
let processB =
 (* reliable and authentic RAN, ORA from the user *)
 in(ub, m3);
 let hashvalue = pdecrypt(m3, pub(XBV)) in
   (* possibly UNreliable and UNauthentic RAN, ORA from the client *)
   (in(cb, m); (* Message 3 *)
   (let (Nonce, = AN, ORAin, RANin) = m in
     (if hashvalue = hash((ORAin, RANin)) then
       out(sb, mac(hash(m), hash(SBV)))))). (* Message 5 *)
```

ProcessU models the user, which is creating new authentic ordered amount and receiver account numbers (a new transaction). It signs these values with the secret private key (check and verify with biometric trait) and sends it on channel *ub*. The transaction data is not considered to be private (guessable + insecure client) and needs to be submitted to the client *C*, so it is made available on channel *uc*:

```
let processU =
```

(user creates new transaction *)*
new $authORA$;
new $authRAN$;
(user checks and verifies authentic transaction data *)*
out(ub, pencrypt(hash(($authORA$, $authRAN$)), prv(XBV)));
out(uc, ($authORA$, $authRAN$)).

The last process, the attacker, is simply creating evil (non-authentic) transaction information and makes it available on channel c. The idea behind this is to check later, if non-signed transaction data can be authenticated:

let *processAttacker* =
new *evilORA*;
new *evilRAN*;
out(c, (*evilORA*, *evilRAN*)).

The following steps are modelled in the main process that is executed initially: create a new secret biometric feature vector XBV and make its public part available for verification. This is for the simulation of the checked and verified transaction data. A new secret pre-shared key SBV is created, as well as a sender account number AN, which is made public. *secretEncS* and *secretSignS* are the secrets for generating the servers key-pairs, again, the public keys are made available to all parties on channel c. The last part describes the processes that can run after this initialization in parallel. Note: an unlimited number of processes is indicated by !*process*. A parallel execution of two processes X and Y is defined by (*processX*) | (*processY*). That means any number of process instances of the user (*processU*), the client (*processC*), the server (*processS*), the *BTD* (*processB*) and the attacker (*processAttacker*) can run in parallel. The client and server are running by purpose with an unbound number of instances in this model, this may be counter intuitive but can be understood when looking at the specific processes:

process
new XBV;
out(c, pub(XBV));
new SBV;
new AN;
out(c, AN);
new *secretEncS*;
new *secretSignS*;
out(c, pub(*secretEncS*));
out(c, pub(*secretSignS*));

((!*processAttacker*) | (!*processU*) | (!*processC*) | (!*processS*) | (!*processB*))

4 Verification of security properties

In order to verify security properties, queries have to be formalized that are checked by ProVerif. A query of the form **query** *attacker*:x., checks if the attacker gets to know x during the execution of the processes. The attacker model is set to active.

(Queries *)*
(Query 1: reliable bit vector extracted from biometric trait(s) *)*

query *attacker*:XBV.
(Query 2: modelled as public-key system *)*
query *attacker*:prv(XBV).
(Query 3: pre-shared secret key *)*
query *attacker*:SBV.
(Query 4: *)*
query *attacker*:hash(SBV).
(Query 5: encryption secret for public-key server construction *)*
query *attacker*:$secretEncS$.
(Query 6: private encryption server key *)*
query *attacker*:prv($secretEncS$).
(Query 7: signature secret for public-key server construction *)*
query *attacker*:$secretSignS$.
(Query 8: private signature server key *)*
query *attacker*:prv($secretSignS$).
(Query 9: seal over authentic transaction data *)*
query *attacker*:mac(hash(($Nonce2, AN, authORA, authRAN$)), hash($SBV$)).
(Query 10: seal over arbitrary transaction data *)*
query *attacker*:mac(hash(($Nonce2, AN, evilORA, evilRAN$)), hash($SBV$)).
(Query 11: server reply over authentic transaction data *)*
query *attacker*:pencrypt(hash((true, $Nonce2, AN, authORA, authRAN$)), prv($secretSignS$)).
(Query 12: server reply over arbitrary transaction data *)*
query *attacker*:pencrypt(hash((true, $Nonce2, AN, evilORA, evilRAN$)), prv($secretSignS$)).

Execution of the queries in ProVerif shows: query 9 (authentic seal) and query 11 (authentic reply from the server) are true. That means, the attacker gets to know information that is available on the channels after a successful run of the transaction authentication protocol using authentic transaction data on the server as well as in the *BTD*. The fresh nonce with limited time validity inside the seal and the server reply avoid replay and delayed-play attacks, therefore the information can not be used to authenticate another transaction.

To wrap up the ProVerif simulation we could show, that the attacker does not get knowledge about the secret keys and the biometric feature vector. Non-authentic transaction data does not get sealed because of the process of checking and verifying inside the secure environment. If the integrity of a verified transaction is compromised, the two generated seals, the one inside the *BTD* and the one inside the server will differ, in this case the transaction is dropped. Non-repudiation of origin is ensured using the biometric subsystem, which only releases the key that is used to generate the seal, if the enrolled person is verifying the transaction. The private server keys stay secret, therefore the authenticity of the server towards C is guaranteed in the protocol, since only the owner of the private encryption key can respond with the correct nonce from message 1 (S only responds to message 1 if *Nonce1* is fresh). Attacks on availability are possible in our model if the attacker drops messages from the channels.

The security properties from section 3.3 hold if the security assumptions from section 3.4 hold true. Especially the assumption, that the authentic user is completing the enrolment phase correctly, is necessary for the non-repudiation of origin property. In a real-life scenario the enrolment of a user could be performed under controlled conditions to satisfy the assumption.

A drawback of the core protocol is that the user is incapable of deciding if the transaction was successfully executed, since a malware infected client can compromise

/ drop the result from the server, this issue and new features are addressed in the next section.

5 BTAP Extension: Secret Message Exchange

Even though an attacker can not gain information from the seal, it is desirable to encrypt all exchanged messages to ensure privacy of the banking information. Note that the seal in message 5 does not need to be encrypted, since an attacker can not get any information about the key, nor the message from the MAC value. The best known forgery attacks for an MAC based on iterated keyed hash functions are birthday attacks, that are also used to find collisions in hash functions [2, 7]. Note also that the property of secrecy of the messages can not hold when the client is compromised, since for convenience reasons the client is still used to generate the transactions and to communicate with the server.

```
Message 1: C->S: {(Nonce1, Ksc, AN, ORA, RAN)}pkEncS
Message 2: S->C: {{(Nonce1,Nonce2,Nonce3,AN,ORA,RAN, ...
                 {((Nonce2, AN, ORA, RAN)}Kbs))}prSignS}Ksc
Message 3: C->B: (Nonce3,AN,ORA,RAN,{(Nonce2, AN, ORA, RAN)}Kbs)
Message 4: U->B: (Ok)
Message 5: B->S: {(mac(hash(Nonce2, AN, ORA, RAN), hash(SBV)))}Kbs
Message 6: S->C: {{hash(true, Nonce2, AN, ORA, RAN), ...
                 ({hash(true, Nonce2, AN, ORA, RAN)}Kbs)}prSignS}Ksc
Message 7: C->B: {hash(true, Nonce2, AN, ORA, RAN)}Kbs
```

Message 1 carries a symmetric session key Ksc, encrypted with the server's public encryption key pkEncS for an encrypted communication between S and C (PKI key verification required). Another symmetric session key, derived from the pre-shared secret SBV, is securing the communication for message 5, 6 and 7:

$$Kbs = onewayfunction((hash(Nonce3), hash(SBV))).$$

Since hash(SBV) is known to S and B, Kbs can only be computed within the two parties (on B after the enrolled user presents his or her biometric trait to release SBV).

After releasing Kbs on the *BTD*, it is ensured to the device, that S has received the information (Nonce2, AN, ORA, RAN), since it is forwarded encrypted with Kbs in message 3 from the client. The *BTD* can check if the same transaction information was also send from the client and displayed to the user. Only if the two sets are identical, the transaction seal is created, otherwise a warning is shown on the secure display. When receiving message 7, it is proven to the *BTD*, that the server executed the transaction encoded in the authentic transaction data. On the secure display of the *BTD* the decision can be shown to the user.

The extended protocol does not send any transaction data in an unencrypted form over the channels, without the need for public-key crypto on the *BTD*. This extension ensures that the transaction data stays private and that the execution of the authentic transaction can be verified.

6 BTAP Extension: Online Banking Transactions Using the "Four-Eyes" Principle

Authentication of transaction data through multiple persons might be part of a policy if the ordered amount succeeds the liability of a single person or role. This procedure might help to prevent financial frauds. BTAP is extendible without much effort to comply with this requirement. Three different scenarios of a multiple-person authentication are identified, the pros and cons are discussed thereafter: 1.) one local *BTD*, one shared secret, 2.) one local *BTD*, multiple shared secrets, 3.) multiple remote *BTDs*, multiple shared secrets.

6.1 One local BTD, one shared secret

The enrolment process of the Helper-Data-Scheme subsystem (Fig. 2) has to be adapted, the shared secret has to be binded to n different data subjects. Therefore n different auxiliary-data-1 (*AD1*) sets have to be generated that define the reliable positions in the fixed length biometric feature vectors of each biometric trait. The pseudo identifier is created as in the original enrolment: $PI = hash(SBV)$. Only one auxiliary-data-2 (*AD2*) is generated during the process using the following formula for the error correction encoded pre-shared secret $CBV = ECC(SBV)$ and the data subjects reliable boolean biometric feature vectors RBV_i for $i = 2...n$ and $n \geq 2$:

$$AD2 = CBV \oplus (\bigoplus_{i=1...n} RBV_i)$$

The result of this adapted enrolment: the shared secret can only be released and therefore the transaction seal can only be generated over the transaction data, if all enrolled biometric feature vectors RBV_i can be extracted during the authentication phase. This means, every enrolled person must verify the transaction data locally with his or her biometric trait. Advantage: the order of presenting the biometric traits is negligible since the XOR-operation is commutative (still *AD1* is person specific and therefore an ID claim like a token is needed); a data subject k could be revoked, by just presenting the biometric trait (where RBV_k can be extracted from), *AD2* could be updated accordingly:

$$\begin{aligned} AD2' &= AD2 \oplus RBV_k \\ &= CBV \oplus (\textstyle\bigoplus_{i=1...n} RBV_i) \oplus RBV_k \\ &= CBV \oplus (\textstyle\bigoplus_{i=1...(k-1),(k+1)...n} RBV_i) \oplus (RBV_k \oplus RBV_k) \\ &= CBV \oplus (\textstyle\bigoplus_{i=1...(k-1),(k+1)...n} RBV_i) \end{aligned}$$

The drawback in this operation mode is that the amount of bit errors that can be corrected stays limited – only CBV carries the error-correction code. Evenly distributed bit errors in the feature vectors RBV_i would affect all positions of the codeword.

Alternatively the XOR-operation is applied to the concatenation of all RBV_i vectors and *CBV*. The entropy of the concatenated feature vector will be increased compared to a single feature vector, a longer key *SBV* and a longer resulting *CBV* could be used for high security demands:

$$AD2 = CBV \oplus \ (RBV_1,\ ...\ RBV_k,\ ...,\ RBV_n)$$

Advantage: Higher level of security against brute force attacks on the secret *SBV*. Disadvantage: the system is inflexible, a re-enrolment is needed if data subject k is not allowed to authenticate online banking transactions anymore.

6.2 One local BTD, multiple shared secrets

When using multiple shared secrets, again an ID claim like a token is needed to distinguish between the enrolled data subjects. A binding of a pre-shared secret key and each extracted reliable boolean biometric feature vector (RBV_i) has to be conducted. This relates to n different enrolments on the same biometric transaction device (BTD) as described in the core BTAP. In this scenario, it is possible to create n different transaction order seals (TOS_i) over the same transaction order record *TOR = (TID, SAN, RAN, ORA)* using a keyed MAC-function:

$$TOS_i = mac(hash(TOR), hash(SBV_i))$$

The seals are send independently from each other to the server, which knows all the enrolled subject for a specific banking account. Advantage: Flexible solution for the user enrolment; Fine-grained policies on the server side enable different levels of security and flexible requirements (number of seals, seals from specific persons) for a transaction based on the ordered amount or the receiver account number, or other metadata. And the non-repudiation property is hold in this scenario, since a unique pre-shared key is bind to a natural person.

6.3 Multiple remote BTDs, multiple shared secrets

As seen in the previous case, a flexible system could be constructed using multiple shared secrets and one local *BTDs*. The same description applies to this case, with the difference that different *BTDs* could be used independent from each other, no ID claim is needed if every data subject is enrolled on a different *BTD* using a different pre-shared secret. This case enables time-shifted transaction authentication but it requires the distribution of pending transactions to the client, which could be done by using the online banking portal, simple e-mail transfer or a dedicated software.

6.4 Additional Authentication Factors and Multiple Biometric Modalities

BTAP can be extended to a multiple factor authentication system, adding possession as well as knowledge authentication factors that are given as input to the *BTD*. Including this information, which is shared with the server side, the transaction seal *TOS* would be computed as:

$$TOS = mac(hash(TOR), (hash(SBV), hash(Password), hash(TokenSecret)))$$

with the keyed mac-function. Adding additional authentication factors would strengthen the BTAP even more.

Extracted reliable biometric feature vectors RBV_i originating from multiple biometric modalities M_i with $i = 2...n$ and $n \geq 2$ of the same person, like e.g. fingerprint and fingervein data, can be used to generate a concatenated biometric feature vector $RBV' = (RBV_1, ..., RBV_n)$ that is used to release the pre-shared key in the BTA protocol.

7 Conclusions

The proposed security properties could be proven using a formal model of the core BTA protocol message exchanges and the protocol verification tool ProVerif. The protocol enables non-repudiable person and data authentic online banking transaction. The extensions enable privacy of the transaction data and in addition new security features: transactions can be sealed by multiple individuals to comply with restrictive policies. BTAP supports multiple biometric modalities and can be extended for multi-factor authentication as well. In the near future the pi-calculus must be extended in order to be able to deal with noisy biometric data as part of security protocols – then also the internal processes of the biometric transaction device could be modelled and verified.

References

1. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: the spi calculus. In: CCS '97: Proceedings of the 4th ACM conference on Computer and communications security. pp. 36–47. ACM, New York, NY, USA (1997)
2. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: the cascade construction and its concrete security. Foundations of Computer Science, Annual IEEE Symposium on 0, 514 (1996)
3. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW-14). pp. 82–96. IEEE Computer Society, Cape Breton, Nova Scotia, Canada (Jun 2001)
4. Dolev, D., Yao, A.C.: On the security of public key protocols. In: SFCS '81: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science. pp. 350–357. IEEE Computer Society, Washington, DC, USA (1981)
5. Hartung, D., Busch, C.: Biometric transaction authentication protocol. The International Conference on Emerging Security Information, Systems and Technologies 4 (2010)
6. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, i. Information and Computation 100(1), 1 – 40 (1992)
7. Prenel, B., Oorschot, P.C.v.: Mdx-mac and building fast macs from hash functions. In: CRYPTO '95: Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology. pp. 1–14. Springer-Verlag, London, UK (1995)
8. Tuyls, P., Goseling, J.: Capacity and examples of template-protecting biometric authentication systems. In: Biometric Authentication. vol. 3087, pp. 158–170. Springer Berlin / Heidelberg (2004)