Towards a Formal Verification of a Multi-factor Authentication Protocol Using Automated Theorem Provers

Eduardo dos Santos, Jean Everson Martina and Ricardo Felipe Custódio
Laboratório de Segurança em Computação (LabSEC) – Programa de Pós-Graduação em Ciências da Computação
Departamento de Informática e de Estatística – Universidade Federal de Santa Catarina (UFSC)
Campus Universitário – Trindade – Florianópolis, SC, Brazil – 88040-900
Email: {esantos, everson, custodio}@inf.ufsc.br

***Abstract*—We present a proposal for a new multi-factor authentication scheme through the use of biometrics and smart-cards. Targeted to the Brazilian court system, which is being redesigned, the main goals of this protocol are to provide adequate division between authentication and authorisation services as well as differentiating the existent roles for improved security and management. In addition, we formalised the protocol into a first-order logic model and verified it with an automated theorem prover. Our main contributions are the design of the protocol and the usage of a methodology including design, verification, implementation and deployment of a protocol of nation-wide scale.**

***Keywords*-Multi-factor Authentication, Security Protocol, Formal Verification, Automated Theorem Proving.**

I. Introduction

Security protocol design and verification has been an established researched topic for a long time now. The security protocol design area is active in designing novel and complex protocols. These security protocols try to achieve goals imposed by the information revolution we live in. The protocol verification area has also been active. This is due to the fact that protocol designers often fail to enforce the goal they claim to achieve. For the sake of exemplifying failures, we can cite discoveries on widely deployed and verified protocols, such as the re-negotiation flaws discovered in SSL/TLS [1] or cite classical cases such as Lowe's attack [2] on Needham-Schroeder's protocol.

Our main goal in this paper is to propose a new protocol for use in the Brazilian Court System. Such protocol will be designed following a novel strategy of incremental design, verification, prototyping and deployment. Our secondary goals are to demonstrate that it is possible to design, from scratch, a protocol taking into account all phases of its design/deployment life-cycle and to establish an incremental assurance process based on the usage of formal methods and software techniques.

During the 1980's and mid-1990's, protocol verification was carried out informally. Informal verification was important because it taught us the importance of understanding the semantics behind the protocols' messages. Due to its informality, it is often easier to find and understand minor flaws by using such techniques. No complicated or extensive reasoning is usually involved. This is still a usual way of starting the evaluation process for security protocols, but it leaves space for flaws.

Since the mid-1990's, we have seen an interest on the usage of formal tools to help the verification of security protocol [3]. We can cite efforts such as: Burrows et al. Belief Logic [4], which first represented formally the beliefs of peers. The Bellare and Rogaway provable security study [5], where the security requirements are met provided the assumptions about the adversary's access to the system are satisfied. Abadi and Gordon's spi-calculus [6] which is a security tailored version of π-calculus [7], where processes communicate through different security channels. Ryan and Schneider's state enumeration [8], which applies the well known process calculus CSP [9] to protocol verification.

We developed a protocol for strengthening the authentication in the Brazilian Court System. Our protocols had to inherit most of the legacy authentication methods based on smart-cards and PKI. Our protocols were targeted to be an evolution for the court authentication system by adding the possibility of coupling biometry. Our protocol had to include an assisted installation process to assure the security of the system through a security ceremony. This set of requirements shaped the characteristics of the protocols we will show below. The authentication process happens in a reasonably controlled environment inside the courts, thus some of our privacy requirements were not strengthened.

The main aim of our protocol design, implementation, verification and deployment exercise was to develop the discipline of security protocols in a complete way. Our goals for the protocol were deliberately underestimated. We tried to achieve a one-way authentication from the client to the server and we did not pay attention to the privacy of biometric data. This happens due to the fact that the system is meant to be evolved to a match-on-card authentication scheme in the future. Our formal verification goals were to show that the protocols are able to stand a series of attacks represented by the conjectures we created. It is not shown in this paper, but the protocols were also implemented as a proof-of-concept using Java. Later it was re-engineered into a library for easy of deployment, concluding the process of design, implementation, verification and deployment we set to achieve.

In the next section we describe the protocol we proposed for the problems stated above (§II). We will cover the most important points in the protocol description. We will describe our initial premises and what we call our anchoring process, which composes a set of guarantees for a security ceremony behind the protocol. We will then describe the two major phases of the protocol which are User Registration and User Authentication. In the following section (§III) we will show our formalisation for the protocol. It was made using a formalisation technique based on first-order logic and an automated theorem prover. Our specification encompasses a Dolev-Yao attacker [10] and all conjectures in the verification are tested against it. We follow up by presenting related work in Section IV and we give our final remarks and summarise our contribution in Section V.

II. The Proposed Protocol

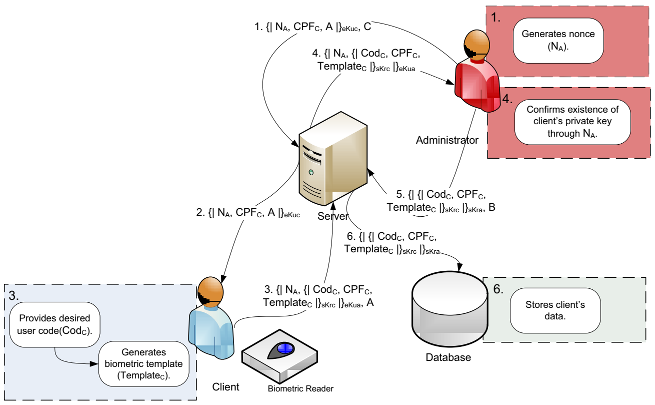
The design presented below takes into account a tailored protocol for usage in the Brazilian court system. The main aim of this protocol is to provide authentication using a public key system coupled with biometrics checked on the server-side.

Some unique characteristics of the execution environment must be taken into account when judging the protocol capability. We call special attention to its usage only inside a court of law in a non Internetworked environment. Another important requirement in the shape of how the protocols were designed has to do with the poor database security. This specific threat model is targeted, specially with the usage of double-signed messages.

A. *Initial Premisses*

An important part of our proposal is the inclusion of a security ceremony for the protocol. This security ceremony has the aim of reducing the trust into the trust of a PKI. To enable the security ceremony to go through, we add a set of characteristics for digital certificates into the source code of the system running the authentication process. Theses characteristics qualify who can act as Installer and Security Officers for the protocol. We do this as an effort to reduce the size of assumptions regarding the root of trust of our protocols as well as to aid in the DRM process of software.

1) Anchoring Process: The user registration process assumes that the proposed system relies on a X509v3 digital certificate issued by a trusted third party. The specification of this certificate [11] is done at source code level by setting the Distinguished Name (DN), or preferably, by referring to an exact certificate extension. Any certificate issued to the combination of DN and certificate extension by a reliable party to the system is acceptable. At the time of installation the private key related to a recognisable certificate will sign



1. {| N_A, CPF_C, A |}$_{eKuc}$, C
4. {| N_A, {| Cod_C, CPF_C, $Template_C$ |}$_{sKrc}$ |}$_{eKua}$
1.
Generates nonce (N_A).
4.
Confirms existence of client's private key through N_A.
Administrator
5. {| {| Cod_C, CPF_C, $Template_C$ |}$_{sKrc}$ |}$_{sKra}$, B
2. {| N_A, CPF_C, A |}$_{eKuc}$
Server
6. {| {| Cod_C, CPF_C, $Template_C$ |}$_{sKrc}$ |}$_{sKra}$
3. {| N_A, {| Cod_C, CPF_C, $Template_C$ |}$_{sKrc}$ |}$_{eKua}$, A
3.
Provides desired user code(Cod_C).
Generates biometric template ($Template_C$).
Client
Biometric Reader
Database
6.
Stores client's data.

Figure 2: Registration Dual-Factor

a ring of digital certificate characteristics, such as DNs or certificate extensions, that will enable the creation of system administrators. This intermediate ring of authorisation regards the delegation of rights and is part of the licensing control of the software system. Any element in this intermediate authorised ring is able to add Administrators to the administrator's ring, which will then be able to add users to the system. Every time the application is initialised, this certificate chain must be checked to validate the signatures and certificates for administrators in the administrator's ring. This process is pictured in Figure 1a.

The reason for such architecture is due to the necessity of reducing the premises we need to trust our protocol. The premises will be reduced to the trust in a Public-Key Infrastructure to assert identities and to ensure that they will not issue any incorrect extensions to certificates. All the trust delegation can be inferred only by referring to these premises.

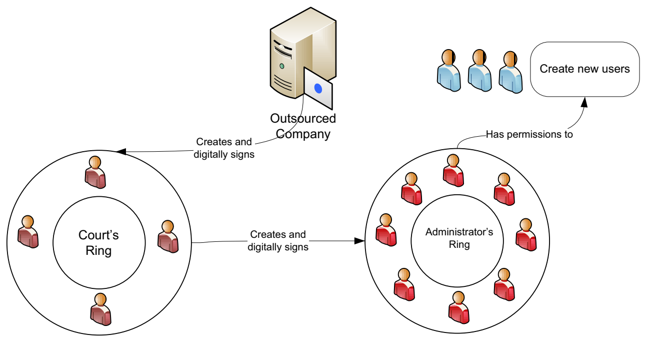
By not referring to any specific keys nor specific user characteristics we are able to make a flexible identity and authorisation system. We are also able to assist the digital rights management for the software company producing the software artefact. The trust and efficacy of such system may also be based on the usage of tamper-proof hardware, such as the one specified by the Trusted Computing Group [12].

B. *User Registration*

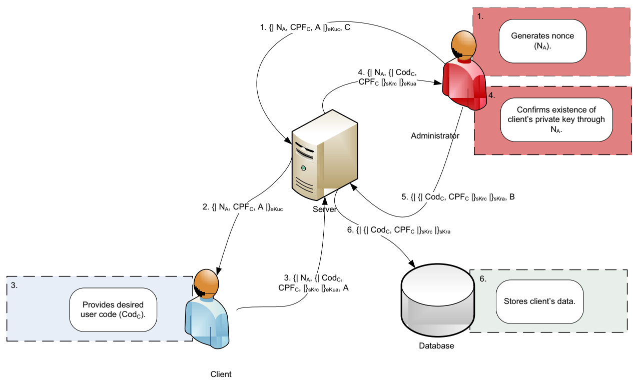
Our design for the protocols is composed of two different proposals. The first encompasses a protocol for the verification of possession of a private key, and the second, a dual-factor including the proof of possession and the usage of biometrics. The protocols for user registration are shown in the subsection below.

1) Proof of Possession proposal: The protocol used for user registration is represented in Figure 1b . The steps are described below:

1) The Administrator asks the server to register a user.



(a) Trust Anchoring


1. {| N_A, CPF_C, A |}$_{eKuc}$, C
1.
Generates nonce (N_A).
4. {| N_A, {| Cod_C, CPF_C |}$_{sKrc}$ |}$_{eKua}$
4.
Confirms existence of client's private key through N_A.
Administrator
Server
2. {| N_A, CPF_C, A |}$_{eKuc}$
5. {| {| Cod_C, CPF_C |}$_{sKrc}$ |}$_{sKra}$, B
6. {| {| Cod_C, CPF_C |}$_{sKrc}$ |}$_{sKra}$
3.
Provides desired user code (Cod_C).
3. {| N_A, {| Cod_C, CPF_C, |}$_{sKrc}$ |}$_{eKua}$, A
6.
Stores client's data.
Database
Client

(b) Registration Proof of Possession

Figure 1: Trust and Registration Processes

The Administrator generates a nonce (N_A) and concatenates the certificate extension identifying the user. This concatenation is encrypted with the user's public key and sent to the server for forwarding;

2) Server receives the message, discards the routing identifier C and forwards it to the corresponding user;
3) The user decrypts the message content with its private key and verifies that he is mentioned in the message. He then generates a token $\{|CPF_C, Cod_C|\}$, and signs it with his private key. This token is then concatenated with the nonce N_A. This concatenation is encrypted with the Administrator's public key and sent to the server for forwarding;
4) Server receives the message, discards the routing identifier A and forwards it to the Administrator;
5) The Administrator decrypts the message with his private key and compares the nonce received N_A with the one generated in step 1, confirming the possession of the Client's private key. The Administrator signs the token $\{|CPF_C, Cod_C\}_{sKr_C}$ with his private key, making it doubly signed. This double-signed token is sent to the server for routing
6) The server, upon receiving the double-signed token, discards the routing identifier B and forwards it to the database for storage.

2) Dual-Factor proposal: The addition of a biometric factor in the proposed user registration protocol requires only small changes. In short, the client should include his biometric templates in message 4. This data will be stored in the database and may subsequently be used for authentication. Figure 2 presents the proposed new registry modified for use with biometrics.

The protocol used for user registration with a biometric factor is described below:

1) The Administrator asks the server to register a user. The Administrator generates a nonce (N_A) and concatenates the certificate extension identifying the user. This concatenation is encrypted with the user's public key and sent to the server for forwarding;
2) Server receives the message, discards the routing identifier C and forwards it to the corresponding user;
3) The user decrypts the message content with his private key and verifies that the certificate extension hinting the user creation in the package. He then generates a token $\{|CPF_C, Cod_C, Template_C|\}$, and sign it with his private key. This token is then concatenated with the nonce N_A. This concatenation is encrypted with the Administrator's public key and sent to the server for forwarding;
4) Server receives the message, discards the routing identifier A and forwards it to the Administrator;
5) The Administrator decrypts the message with his private key and compares the nonce received N_A with that generated in step 1, confirming the possession of the Client's private key. The Administrator signs the token $\{|CPF_C, Cod_C, Template_C\}_{sKr_C}$ with its private key, making it doubly signed. This double-signed token is sent to the server for routing
6) The server, upon receiving the double-signed token, discards the routing identifier B and forwards it to the database for storage.

C. User Authentication

1) Proof of Possession proposal: The proposed authentication protocol is based on a challenge response strategy controlled by the server. The protocol is shown in Figure 3. The steps that compose our authentication protocol are described below:

1) The server starts the authentication process by generating a nonce that is sent to to the user;
2) client concatenates his CPF (source) and the identification of the server (destination) to the challenge

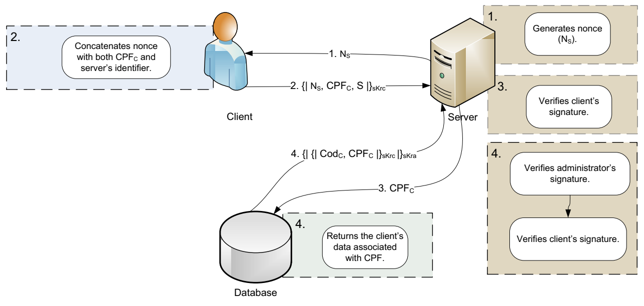

2.
Concatenates nonce with both CPF_C and server's identifier.
1. N_S
2. {| N_S, CPF_C, S |}$_{sKrc}$
Client
Server
1.
Generates nonce (N_S).
3.
Verifies client's signature.
4. {| {| Cod_C, CPF_C |}$_{sKrc}$ |}$_{sKra}$
3. CPF_C
4.
Verifies administrator's signature.
Verifies client's signature.
4.
Returns the client's data associated with CPF.
Database

Figure 3: Authentication Proof of Possession

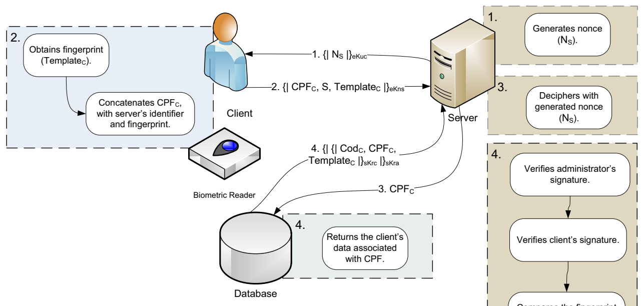

2.
Obtains fingerprint ($Template_C$).
Concatenates CPF_C, with server's identifier and fingerprint.
1. {| N_S |}$_{eKuc}$
2. {| CPF_C, S, $Template_C$ |}$_{eKns}$
Client
Server
Biometric Reader
1.
Generates nonce (N_S).
3.
Deciphers with generated nonce (N_S).
4. {| {| Cod_C, CPF_C, $Template_C$ |}$_{sKrc}$ |}$_{sKra}$
3. CPF_C
4.
Verifies administrator's signature.
Verifies client's signature.
Compares the fingerprint provided by client with the one stored in database.
4.
Returns the client's data associated with CPF.
Database

Figure 4: Authentication Dual-Factor

received. This token is signed with the private key of the client and sent to the server. The concatenation of the source and destination of the message is necessary to prevent oracle attacks;

3) Server verifies the signature of the client, confirming that the client is in possession of his private key. So, CPF_C is queried to the database, and
4) The database server returns the double-signed token to the Server . Upon receiving the message, the Server verifies that the data stored in the database provided by the Client and the database are equal.

2) Dual-Factor proposal: In the same way as the registration dual factor protocol, the authentication dual factor requires some changes to incorporate biometrics into its semantics. Figure 4 presents the dual factor version for authentication. In short, the changes are limited to inserting the biometric template when CPF_C and Cod_C are requested. A different final step is performed by the server, which also has to verify whether the user's provided biometric templates match with the one previously stored on the database.

III. Protocol Formalisation

The formalisation process adopted follows the model proposed by Weindenbach and it makes use of a first-order logic model [13] combined with a Dolev-Yao threat-model [10]. To proper evaluate the effectiveness of the proposed protocol, as well as its possible problems, we use an automatic theorem prover, in this case, SPASS version 3.5 [14].

This process consists of describing the set M of messages sent during the protocol execution. Each message is represented through the elements that make part of every first-order logic formula, such as quantifiers, predicates, functions, constants and connectives.

We will show the formalisation for the most complex case, which includes the biometrics registration and authentication. We also need to clarify that some formulae were modified from the actual input from the theorem prover. We deliberately are not showing technical connections between the formulae here for the sake of readability.

A. Main Logical Model

Both parts of our protocol, registration and authentication, are modelled together due to the strong relationship between them. In addition, taking this strategy allows us to better test the conjectures we intend, since we are dealing with a synchronous protocol. Being a synchronous protocol means that for a given step n to be executed it demands that all preceding steps have already been executed, successfully in their exact sequence. In short, the protocol must have their execution flow followed strictly.

The first thing to take into consideration on formally modelling any protocol is to properly define the entities (or agents) that make it up as well as their initial knowledge base. These actions are defined, respectively, by the predicates: $E(x)$ which returns $true$ if x is an agent in the protocol, and $Knows(x, y)$ which returns $true$ if x is known by agent y. Consider x and y as being mere placeholders for functions or constants that act as parameters to predicates or other functions.

Additionally, our logical model makes use of the following classes of functions, namely:

- Message composition functions: group many individual messages into a single one. For example: $pair()$ and $triple()$.
- Message exchange functions: indicate operations upon an individual message. For example: $sent()$.
- Key operation functions: indicate keys and their corresponding entities. For example: $kp()$ for a key pair, $krkey()$ for a private key and $kukey()$ for a public key.
- Security functions: indicate the use of cryptographic primitives on functions and constants. For example: $encr()$ for encryption and $sign()$ for digital signature.

Some constants (0-ary functions) are also defined as follows:

- Protocol entities: a (administrator), c (client), s (server) and d (database).

- Identity markers: used by the server on its message redirection feature: ida (to administrator), idc (to client), ids (to server) and idd (to database).
- Private keys and corresponding public-key certificates: kra and $cera$ (of administrator); krc and $cerc$ (of client).
- Nonces generated: na (by administrator) and ns (by server).
- Client's identification data: $cpfc$ (national identity number), $codc$ (username into system) and $templatec$ (personal biometric template). The national identity number is the Income Tax Number (CPF [15]) which is unique to every citizen.

All these constructions are specified in the formulae bellow:

1) $E(a)$
2) $Knows(kp(krkey(kra,a),kukey(cera,a)),a)$
3) $Knows(krkey(kra,a),a)$
4) $Knows(kukey(cera,a),a)$
5) $Knows(kukey(cerc,c),a)$
6) $Knows(cpfc,a)$
7) $Knows(na,a)$
8) $E(c)$
9) $Knows(kp(krkey(krc,c),kukey(cerc,c)),c)))$
10) $Knows(krkey(krc,c),c)$
11) $Knows(kukey(cerc,c),c)$
12) $Knows(kukey(cera,a),c)$
13) $Knows(triple(codc,cpfc,templatec),c)$
14) $Knows(codc,c)$
15) $Knows(cpfc,c)$
16) $Knows(templatec,c)$
17) $E(s)$
18) $Knows(kukey(cera,a),s)$
19) $Knows(kukey(cerc,c),s)$
20) $Knows(ns,s)$
21) $E(d)$

1) Formalisation of Registration: From this point on, we begin to formalise the exchange of messages between the protocol participants. Since the registration phase is usually the first to be run in any communication protocol, we model it first. This phase contains a total of 6 steps.

22) $Knows(kukey(cera,a),a) \wedge Knows(kp(krkey(kra,a),kukey(cera,a)),a) \wedge Knows(kukey(cerc,c),a) \wedge Knows(cpfc,a) \wedge Knows(na,a) \supset M(sent(a,s, pair(encr(triple(na,cpfc,ida),cerc),idc))) \wedge Stores(pair(na,cpfc),a)$

In the first step of the registration (formula 22), we need to make sure that the administrator is really in possession of his initial knowledge base (as defined by formulae 2 to 7). This is done through predicates $Knows()$. If so, the first message is sent to the server and the pair $(na, cpfc)$ is stored into the administrator's base.

23) $\forall xa[M(sent(xa,s, pair(encr(triple(na,cpfc,ida),cerc),idc))) \supset M(sent(s,c,encr(triple(na,cpfc,ida),cerc))))]$

Next, in formula 23, we notice that the unique difference to the previous formula is the post-conditional part (right-side)of the implication arrow. This happens because we are dealing with a synchronous protocol, which requires that the communication flow is strictly followed. In other words, it is not possible that formula n be evaluated to $true$ without all preceding formulae $n-1$ having also been evaluated to $true$. This situation will happen throughout the formalisation. After receiving the message from the administrator, the server forwards it to the corresponding client. The constant idc is removed since it is only used to indicate which entity the message should be forwarded to. The removal of such marking constants is always performed when forwarding messages.

24) $\forall xs[Knows(kp(krkey(krc,c),kukey(cerc,c)),c) \wedge Knows(kukey(cera,a),c) \wedge Knows(codc,c) \wedge Knows(cpfc,a) \wedge Knows(templatec,a) \wedge M(sent(xs,c,encr(triple(na,cpfc,ida),cerc))) \supset M(sent(c,s,pair(encr(pair(na, sign(triple(codc,cpfc,templatec),krc)),cera),ida)))]$

Continuing the message exchange flow, in formula 24, the client double-checks his knowledge base to make sure all information needed is actually available. He provides his personal code (*codc*), Income Tax Number ($cpfc$) and biometric template ($templatec$). This information is concatenated with nonce na, signed and encrypted. Then, all the information is sent to the server through predicate $sent()$.

25) $\forall xc[M(sent(xc,s,pair(encr(pair(na,sign(triple(codc,cpfc,templatec),krc)), cera),ida))) \supset M(sent(s,a,encr(pair(na, sign(triple(codc,cpfc,templatec),krc)),cera)))]$

Formula 25 only redirects the message received from server to administrator, in the same way as formula 23.

26) $\forall xs[Knows(kukey(cera,a),a) \wedge Knows(kp(krkey(kra,a),kukey(cera,a)),a) \wedge Knows(kukey(cerc,c),a) \wedge Knows(cpfc,a) \wedge Knows(na,a) \wedge M(sent(xs,a,encr(pair(na, sign(triple(codc,cpfc,templatec),krc)),cera)))) \supset M(sent(a,s,pair(sign(sign(triple(codc,cpfc,templatec),krc),kra),idd)))]$

Next, in formula 26 the administrator double-checks his entire knowledge base (same as Formula 22. He also verifies if the received message corresponds to the one sent by server (predicate $M(sent(xs,a,encr(pair(na, sign(triple(codc,cpfc,templatec),krc)),cera))))$. Then a new message containing the double-signed client data is created and sent to server for further redirection to database (predicate $M()$ in the postcondition side)).

27) $\forall xa[M(sent(xa,s,pair(sign(sign(triple(codc,cpfc,templatec),krc),kra),idd))) \supset Knows(cpfc,d) \wedge Knows(sign(sign(triple(codc,cpfc,$

$templatec), krc), kra), d), \wedge M(sent(s, d, sign(sign($
$triple(codc, cpfc, templatec), krc), kra))))]$

Finally, in the last step of registration (27), by receiving the previous message from administrator, the server only has to redirect it to database for storage. The message received (precondition side) and the message redirected only vary by the identity marker idd, which is removed prior to redirection.

2) Formalisation of Authentication: We now model the authentication phase of our protocol. This phase is slightly smaller, having 4 steps against the 6 steps present in the registration phase. The reason for modelling both protocols together is because they complement each other.

28) $Knows(kukey(cerc, c), s) \wedge Knows(ns, s) \supset Knows(ns, c) \wedge Stores(ns, s) \wedge M(sent(s, c, encr(ns, cerc)))$

In the formal model, the first step of authentication (28) is a message sent by the server to the connecting client. At first glance, this may sound strange since there is no explicit connection attempt being modelled between the client and server. We always assume that the client tries to connect first and, once the connection is established, the server immediately sends nonce ns freshly generated. For preventing replay attacks on the nonce, it is transmitted ciphered with the connecting client‘s corresponding public key. In addition, the server stores the nonce on its knowledge base for further comparison.

29) $\forall xs[$
$Knows(ns, c) \wedge Knows(cpfc, c) \wedge Knows(templatec, c) \wedge M(sent(xs, c, encr(ns, cerc))) \supset M(sent(c, s, encr(triple(cpfc, ids, templatec), ns)))]$

After receiving the nonce ns from server, the client requests the fingerprint and Income Tax Number from the person trying to log in. We notice again that the pre-conditional clause on the implication is almost exactly the same as the post-conditional clause in formula 28. The single difference in formula 29 is the $\forall xs$ meaning that the client can receive response from virtually any server. This mechanism allows us to cover all possible incoming message recipients a client receives, thus increasing the correctness of our logical model. To finalise this step, the client sends the Income Tax Number, the server identification and the biometric concatenated , all ciphered with the nonce ns. The whole piece is then sent to the server from which the client received the nonce ns.

30) $\forall xc[$
$Knows(cpfc, s) \wedge$
$M(sent(xc, s, encr(triple(cpfc, ids, templatec), ns))) \supset M(sent(s, d, cpfc))]$

In this step (30), the server extracts the Income Tax Number from the message it has just received. To do so, it uses the nonce ns associated with the current authenticating client. Again, we adopt the same strategy done throughout this formal model, to not explicitly specify the incoming recipient for a given message. Therefore, the server may receive messages from virtually any client. After extracting the Income Tax Number, the server sends it to the database to obtain the client's factors previously stored during registration.

31) $\forall xs[$
$Knows(cpfc, d) \wedge Knows(sign(sign(triple(codc, cpfc, templatec), krc), kra), d) \wedge M(sent(xs, d, cpfc)) \supset M(sent(d, s, sign(sign(triple(codc, cpfc, templatec), krc), kra))))]$

The last step of authentication (31) is straightforward. The database receives the Income Tax Number and performs a query on it, returning all the tuples associated with the given number. On our formal model, a 5-ary tuple containing the values stored at the end of formula 27 is retrieved. Then, the database simply forwards it to the server where digital signatures are verified and the templates, stored and provided, are matched against each other. If all verifications pass, the connection is accepted, otherwise rejected. Though those verification steps do not involve message exchange, being performed only by the server, they are not explicitly present on our logical model.

B. Attacker Model

The attacker model refers to the set of all formulae an attacker can make use of in order to break into the protocol successfully. The formulae describe the actions and knowledge of a given intruder i from a logical viewpoint. The basic intruder modelling follows the Dolev-Yao threat model [10] and it is as follows. This intruder model was also used in [16] when verifying the Brazilian Electronic Invoice protocols.

32) $Knows(kukey(cerc, c), i)$
33) $Knows(kukey(cera, a), i)$
34) $\forall X, Y, W[M(sent(X, Y, W)) \supset Im(W)]$
35) $\forall U, V[Im(pair(U, V)) \supset Im(U) \wedge Im(V)]$
36) $\forall U, V, W[Im(triple(U, V, W)) \supset Im(U) \wedge Im(V) \wedge Im(W)]$
37) $\forall U, V[Im(U) \wedge Im(V) \supset Im(pair(U, V))]$
38) $\forall U, V, W[Im(U) \wedge Im(V) \wedge Im(W) \supset Im(triple(U, V, W))]$
39) $\forall U, X, Y[Im(U) \wedge E(X) \wedge E(Y) \supset M(sent(X, Y, U))]$

Firstly, we must define the set of all publicly available information to the attacker (formulae 32 and 33). Next, the attacker's abilities upon the network need to be specified. For instance: record all messages (formula 34), divide a given message into small pieces (formulae 35 and 36), assemble new messages from existing parts (formulae 37 and 38), and send fake messages (formula 39). Every captured and/or modified message is also added to predicate Im, which represents the intruder's acquired knowledge.

40) $\forall U, V[Im(U) \wedge E(V) \supset Knows(krkey(U, V), i)]$
41) $\forall U, V[Im(U) \wedge E(V) \supset Knows(kukey(U, V), i)]$
42) $\forall U, V, X[Im(U) \wedge Knows(kukey(V, X), i) \wedge E(X) \supset Im(encr(U, V))]$

43) $\forall U, V, X[Im(U) \wedge Knows(krkey(V, X), i) \wedge E(X) \supset Im(sign(U, V))]$

Secondly, in the formulae above, the cryptographic operations that can be performed by the attacker in an attempt to break the protocol up are covered. By recording every message exchanged between entities, the attacker can exhaustively test all possible combinations until finding a key (formulae 40 and 41). The attacker also has the ability of composing all possible signed and ciphered messages with the keys in his knowledge (formulae 42 and 43).

44) $\forall U, V, W[Im(encr(U, V)) \wedge Knows(krkey(V, W), i) \wedge E(W) \supset Im(U))]$

Furthermore, we also add a formula that allows the intruder to learn the message contents for which he knows the decryption key. A good reason for using formula 44 is to verify whether the whole protocol's security intrinsically depends on the security of its private keys.

C. *Initial Results of the Formal Analysis*

Although results are far from what would be expected for a fully verified protocol, the results are promising. Due to space constraints, the formal proofs are not present in this work. We will now enumerate the most important facts seen so far during the protocol execution on SPASS.

Fact 1: Model Saturation

The first fact worth mentioning is the saturation of our model. Saturation means that the intrinsic characteristics of our protocol remained stable. In addition, SPASS was able to derive X clauses and kept Y clauses.

Fact 2: $\exists U[M(sent(d, s, sign(sign(triple(codc, U, templatec), krc), kra)))]$

Now, we create a test for our modelling. Fact 2 reproduces the last implication's postcondition side to make sure all preceding formulae run accordingly. Here, we should not forget the synchronicity property.

We tested a series of conjectures regarding the gathering of knowledge by the Intruder. Most of the conjectures were related to the possibility of the attacker faking authentication or stealing the nonce in the non-biometric version of the protocol. None of the conjecture we were able to elaborate enabled us to achieve a successful attack.

We have not created an attack which our model has not resisted. Its is not a claim of security, but rather a claim that we tested it thoroughly. We leave this task of creating new attacks to our readers who can use the main logical model as well as the intruder model to tests and evaluate their claims.

IV. Related Work

In the literature, we found several examples of security protocols which use more than one factor to offer stronger authentication to the participants involved [17], [18], [19], [20], [21].

Additionaly, we discovered that some schemes aim at using the biometric template as input for generating stronger cryptographic systems [22], [23], [24], [25], [26] This approach seems convenient for the scenarios of key exchange and session (temporary) key generation and to avoid the acquisition of costly third party digital certificates.

Although security protocols that use biometrics in some way are not new, we did not find any example of formal verification in such class of protocols. Indeed, there were examples of formal verifications claimed by the protocol developers, but, those in reality, neither made use of logics, nor sound mathematical analysis so far. None of the main tools known to assure a security protocol‘s correctness were used on biometric protocols. Consequently, those proof statements may rely on unrealistic premises or do not cover attacker movements that might have been easy to detect with the proper technique.

The closest work to our verification goals was performed by Jan Jurjens in [27], [28]. In his works, the author used a biometric protocol as case study to evaluate his approach to check C code and UML diagrams for security properties' correctness. However, in spite of the sound results achieved, his methods only evaluated the security properties at low-level, giving no way to determine whether the biometric protocol analysed was also secure at the message exchange level.

V. Conclusions and Future Work

In this work, we proposed a new authentication protocol that relies on different factors to ensure security. As its main features, we can highlight: the presence of different entities to share responsibilities as well as preventing attacks based on a single weak point and the use of strong premises bound on all execution levels (software signing, trust anchoring and proof of possession).

Additionally, to help us verify whether the protocol contains any vulnerabilities on its high-level description, we created a first-order logic model to test attacks with an automated theorem prover. Although no flaws have been found so far, our proposal still needs deeper analysis to make sure it is indeed secure. As far as we are aware, our work is the first to employ verification techniques based on logics to evaluate a biometric protocol's objectives.

Although not contained in this paper, we also implemented a prototype of the protocol to determine its feasibility for execution. After such implementation we developed the protocol into a library for easier integration within the actual authentication system for courts in Brazil.

As future work, the authors will continue working on extending the attack conjectures to cover the protocol better. Furthermore, a formalisation onto High-Order Logic (HOL) with the theorem prover Isabelle [29] is planned in order to acquire an even broader understanding of the protocol's capabilities and limitations. Another interesting research topic would be to investigate how user's privacy is affected in our scheme.

Acknowledgments

The authors would like to thank CAPES Brazil on grant 41001010025P2 and CNPq/FINEP Brazil on grant #01.11.0008.00 { 1504/10 } for funding and supporting. We would also like to thank to Nicole Contini for proof reading.

References

[1] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the https protocol," *IEEE Security and Privacy*, vol. 7, pp. 78–81, 2009.

[2] G. Lowe, "An attack on the needham-schroeder public key authentication protocol," *Information Processing Letters*, vol. 56, no. 3, pp. 131–136, Nov. 1995.

[3] C. Meadows, "Formal verification of cryptographic protocols: A survey," in *Proceedings of Asiacrypt 96*, 1996.

[4] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Trans. Comp. Syst.*, vol. 8, pp. 18–36, 1990.

[5] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in Cryptology*, ser. Lecture Notes in Computer Science, D. R. Stinson, Ed., vol. 773. Springer-Verlag, Berlin Germany, 1994, pp. 232–249.

[6] M. Abadi and A. D. Gordon, "A calculus for cryptographic protocols: The spi calculus," *Information and Computation*, vol. 148, no. 1, pp. 1–70, 10 Jan. 1999.

[7] R. Milner, *A Calculus of Communicating Systems*, ser. Lecture Notes in Computer Science. Springer-Verlag, 1980, vol. 92.

[8] P. Y. A. Ryan and S. A. Schneider, "An attack on a recursive authentication protocol — A cautionary tale," *Information Processing Letters*, vol. 65, no. 1, pp. 7–10, Jan. 1998.

[9] C. A. R. Hoare, *Communicating sequential processes*. Englewood Cliffs, N.J: Prentice-Hall International, 1985.

[10] D. Dolev and A. Yao, "On the security of public key protocols," *Information Theory, IEEE Transactions on*, vol. 29, no. 2, pp. 198 – 208, mar 1983.

[11] R. Housley and T. Polk, *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[12] Trusted Computing Group, "TPM main specification," TCG, Main Specification Version 1.2 rev. 103, Jul. 2007.

[13] C. Weidenbach, "Towards an automatic analysis of security protocols in first-order logic," in *16th International Conference on Automated Deduction*. Springer, 1999, pp. 314–328.

[14] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and P. Wischnewski, "Spass version 3.5," in *CADE-22*, ser. LNCS. Springer, 2009, vol. 5663, pp. 140–145.

[15] Brazil Federal Tax Department, "Income Tax Number (CPF) - Questions and Answers," http://www.receita.fazenda.gov.br/PessoaFisica/CPF/PerguntasRespostas/PerguntasRespostas.htm.

[16] J. E. Martina and L. A. C. Boal, "A formal analysis of the brazilian electronic bill of sale protocols," in *Proceedings of the Brazilian Symposium on Information and Computer System Security*, Gramado, Brazil, Semptember 2008.

[17] D. Yang and B. Yang, "A biometric password-based multi-server authentication scheme with smart card," in *Computer Design and Applications (ICCDA), 2010 International Conference on*, vol. 5, june 2010, pp. V5–554 –V5–559.

[18] ——, "A provable security biometric password multi-server authentication scheme with smart card," in *Data, Privacy and E-Commerce (ISDPE), 2010 Second International Symposium on*, sept. 2010, pp. 33 –38.

[19] H. Mathew, S. Raj, P. Gundapu, and S. Angeline, "An improved three-factor authentication scheme using smart card with biometric privacy protection," in *Electronics Computer Technology (ICECT 2011)*, vol. 3, april 2011, pp. 220 –223.

[20] P. Bodnar, "A solution to remote biometric identification," in *Information Technology, 2008. IT 2008. 1st International Conference on*, may 2008, pp. 1 –4.

[21] S. Sanyal, A. Tiwari, and S. Sanyal, "A multifactor secure authentication system for wireless payment," in *Emergent Web Intelligence: Advanced Information Retrieval*. Springer, 2010, pp. 341–369.

[22] F. Hao, R. Anderson, and J. Daugman, "Combining cryptography with biometrics effectively," University of Cambridge, ComputerLab, Tech. Rep. UCAM-CL-TR-640, Jul. 2005.

[23] S. Kanade, D. Petrovska-Delacretaz, and B. Dorizzi, "Generating and sharing biometrics based session keys for secure cryptographic applications," in *Biometrics: Theory Applications and Systems (BTAS 2010)*, 2010, pp. 1 –7.

[24] D. Pointcheval and S. Zimmer, "Multi-factor authenticated key exchange," in *Applied Cryptography and Network Security*, ser. LNCS. Springer, 2008, vol. 5037, pp. 277–295.

[25] F. Hao and D. Clarke, "Security Analysis of a Multi-Factor Authenticated Key Exchange Protocol," Newcastle University, Computing Science, Tech. Rep. CS-TR-1312, Feb. 2012.

[26] S. Kanade, D. Petrovska-Delacrétaz, and B. Dorizzi, "Multi-biometrics based crypto-biometric session key generation and sharing protocol," in *13th ACM workshop on Multimedia and security*. New York, NY, USA: ACM, 2011, pp. 109–114.

[27] J. Jurjens, "Sound methods and effective tools for model-based security engineering with uml," in *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, may 2005, pp. 322 – 331.

[28] ——, "Code security analysis of a biometric authentication system using automated theorem provers," in *21st Computer Security Applications Conference*, dec. 2005, pp. 10pp. –149.

[29] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002, lNCS Tutorial 2283.