A Secure Framework to Authenticate Remotely Digital Documents based on The TLS Protocol

Juan C. López Pimentel and Rodolfo E. Ibarra Orozco and Victor F. Ramos Fon Bon and Raúl Monroy

***Abstract*—Every day digital resources protection is an appealing research area because the amount of e-documents transmitted over non-secure channels such as the Internet has increased considerably. So, secure mechanisms must be implemented to identify illegal copies of crucial information.**

Many researchers have proposed solutions based on *controlled environments* applying asymmetric cryptography. Other researchers have applied some techniques such as steganography, cryptography and security protocols to deal with *uncontrolled environment*.

This paper presents a secure framework to sign and authenticate remotely digital documents focused on *uncontrolled environment*. The framework is compound by various security protocols used to ensure authentication and confidentiality in each of the services. Each of protocols has been formally verified using The AVISPA tool and we found that they provide strong authentication and privacy security properties. In addition, the framework has also been adapted to work with TLS and SSL protocols because they are the standard in e-commerce.

***Index Terms*—Security-protocols, steganography, cryptography, watermarking and e-commerce.**

I. Introduction

Sudden changes of network technology have led to a new era, The Digital World. Every day the amount of e-documents transmitted over non-secure networks such as the Internet has increased considerably. In fact, implementing secure mechanisms to protect digital documents has been a very attractive research area in the last years in order to prevent that a malicious user may produce illegal copies of crucial documents.

Protecting digital documents relates with two general scenarios:

- **Controlled-environment.** It consists in distributing digital documents within a computer network system. Hence, one can implement a public key infrastructure and assign to each participant a key pair. In this case, signing a document consists in only encrypting a document with the signer's private key. On the other hand, verifying the signature would consist in processing the cyphered message using the signer's public key, then the signer's data and message are revealed.
  However, if the document is exposed outside the limits of the controlled environment, then, how can we know who has signed such a document?.
- **Uncontrolled-environment.** It consists in distributing e-documents out of a controlled computer network system such as the Internet. One technique to protect authenticity in digital documents has been to apply redundant information in its content by means of steganography. Steganography provides important applications in the digital world, one of this is watermarking. Some of the main applications of watermarking is protection of copyright, [18].

Our research goes beyond of uncontrolled environment, we focus in providing a mechanism to sign and authenticate remotely digital documents. To do that, it is necessary to apply cryptographic protocols in order to authenticate and establish a secure channel between the participants, besides steganography and cryptography.

Many researchers have focused on document authentication using watermarking techniques and TLS protocols, [8], [23]; others have dealt with some problems like *The Customer's Right Problem and The Unbinding Problem* [19], [16], [7], [16], [6], [7], [5]. We have identified that neither of them has formally verified the protocols they have proposed, much less has stated what properties the protocol provides. Verification is a very important step in the software development process, especially because security protocols are considered critical applications and a lot of informatic crimes have arisen for not considering it. See [9] for a survey of faulty security protocols, where many of them were found to have potential flaws using formal verification.

This paper presents a secure framework where a user can sign and authenticate remotely digital documents focused on uncontrolled-environment. The framework contains four stages: i) establishing a secure channel; ii) secrets generation; iii) authentication using the secrets; and iv) providing the services of signing submitted e-documents or/and certifying the authenticity of any one signed previously. To ensure authentication and confidentiality we developed for each stage a security protocol. As a verification test, each of the protocols were formally verified using The AVISPA Tool Web interface and we found that they provide security and strong authentication properties. This framework also presents an improved version of the work *Symmetric Cryptography Protocol for Signing and Authenticating Digital Documents* presented in [10]; the framework has been adapted to work with TLS and SSL protocols because they are the standard in e-commerce.

The remaining of the paper is organized as follows: Section II summarises closer related works and compares to the work proposed in this research. Section III establishes the formalism used throughout the document and outlines our framework. Next sections describe the framework in detail: section IV explains how to establish a secure channel; section V states how to generate the corresponding secrets for

Manuscript received February 18, 2013; revised March 25, 2013. The research reported here was supported by PROMEP under Grant PROMEP/103.5/12/4721.

J.C. López Pimentel, R.E. Ibarra Orozco and V.F. Ramos Fon Bon are with the Department of Software Development Engineering, Universidad Politécnica de Chiapas. Calle Eduardo J. Selvas S/N y Avenida Manuel de J. Cancino. Col. Magisterial, Tuxtla Gutiérrez, Chiapas, México; emails: {jclopezpimentel,rodoibarra,ramosfonbon}@gmail.com

Raúl Monroy is with the Computer Science Department, Tecnológico de Monterrey, Campus Estado de México. Carretera al lago de Guadalupe, Km 3.5, Atizapán, 52926, Mexico; email: raulm@itesm.mx

providing authentication to the system; section VI describes the process of signing and consulting e-documents using the secure tunneling. Finally, our conclusions and future work are drawn in section VII.

II. Related work

A. Security Protocols and Formal Support

Security protocols are also called cryptographic protocols because use cryptography to ensure the transmitted messages on the network. If a cryptographic protocol is not designed carefully, it may contain flaws, which can be the ideal starting point for yielding an attack. Such flaws can be subtle and hard to find.

A number of faulty protocols exists in the literature whose flaws were not found for a long time and that have received intensive analysis. Thus, experience has shown that the design of security protocols is particularly difficult and error-prone. Thereby, the verification of security protocols has attracted a lot of interest in the formal methods community yielding an abundance of tools and techniques in the last few years. Some methods are based on belief logics such as BAN [4] and GNY [3]; another methods are state exploration, like NRL [13], FDR [15], AVISPA [2]; another methods use the strand spaces model like [22], [20]; and finally, another methods are based on theorem proving, such as the Paulson's Inductive Method [17], Coral [21], etc. Of all, Coral, AVISPA and Athena seem to be the most powerful and suited that we are looking for: these tools are capable of determining whether or not a protocol is valid. In the case of unsatisfiability, a counterexample is output. The framework here presented has been precisely verified using The AVISPA Tool Web Interface. For a survey about formal support to security protocol development see [9].

The following works refer to researchs about copyright protection using remote access.

B. Image Authentication with Remote Access

Yusuke Lim, et.al. (2001), [8] propose a Web image authenticator, using invisible watermarks known as fragile. Any authorized user; i.e. one with valid access to the Web server, can generate an image with a watermark, or also can verify the authenticity of an image containing a watermark. They intend to use Secure Socket Layer protocol (SSL) to ensure access to the server, however, there is not evidence to have implemented it; so, this is left as future work. In addition, it is not clear if the watermarked image is ensured using any cryptographic system.

Yusuke Lim's work is based on the work of Yeung and Nintzer, [23], which aims to verify the authenticity of an image and to detect when an image has been altered including a small image as watermark. Neither of them, have described or showed any evidence about formal verification of their proposal or whether or not the watermarked image is applied a cryptographic process.

C. The Customer's Right Problem and The Unbinding Problem

Qiao and Nahrstedt, [19], introduce the so-called *The Customer's Right Problem*, which constists in implementing appropriated mechanisms for determining if a customer has a legal digital resource. Most works have been designed to protect the seller's ownership of digital contents rather than the buyer's right.

Qian and Nahrstedt, proposed a protocol based on embedding data of the customer in the digital resource, [19]. However, Memon and Wong, [16], identified that a problem persists, which they called *the unbinding problem*. It consists in implementing security mechanisms to avoid the seller can transplant a watermark and to blame the buyer for a piracy distribution. They proposed a buyer-seller watermarking protocol based on public key cryptography, even so the unbinding problem persists. In addition, details about authentication between the participants is not described.

Chin-Laung Lei et.al [7] also proposed a solution based on public key cryptography to solve the *customer's right problem* and *unbinding problem*. Such a solution fixes some problems of Memon and Wong's scheme [16] in the sense of taking care the customer's private data.

Franco Frattolillo et.al. presented a watermark protocol to be adopted in a web context, [6]. The main idea is to ensure the copyright protection of digital content distributed over the Internet. This work is based on [7]. The protocol includes four agents: client, seller, watermark certification authority, and a server. These agents communicate using Secure Socket Layer (SSL). Recently, Franco Frattolillo [5], has improved its watermark protocol in such a way a client can be authenticated with the server although the first one is not provided with digital certificates.

The above protocols try to solve the customer's right problem and the unbinding problem in multimedia content. Our research is very related with the previous ones but differ in the following aspects that ours provides: authentication step between the signer and the client; private connection established in the authentication step in order to apply signatures and to exchange digital documents; privacy and authentication to consult signatures of different documents; formal verification, guaranteeing security on the information flow, since, the main contribution of our research is to propose a framework compound by various security protocols, which are formally verified, this step is very important because a lot of informatic crimes have arisen for not considering it.

D. Signing and Consulting E-documents

López-Pimentel et.al. [10], [11] developed a cryptographic protocol to sign and authenticate digital documents with five stages: Initial Knowledge, Distributing Shared Key, Authentication, Signing and Secure Consulting process. There, the authentication part was formally verified; the proof was carried out on bmp documents and the least significative bit algorithm was used in the corresponding steganography part.

In such a protocol it is considerated that the server must initially know all shared keys and all steganography algorithms previously chosen by the Client. Despite the above, the following two questions may arise if such a protocol is implemented.

1) Initial secrets distribution: To obtain authentication, the participants must have established a shared key K_{CS} with the server as the initial knowledge. The question here is: How was the key distributed? or how should it be distributed?. On

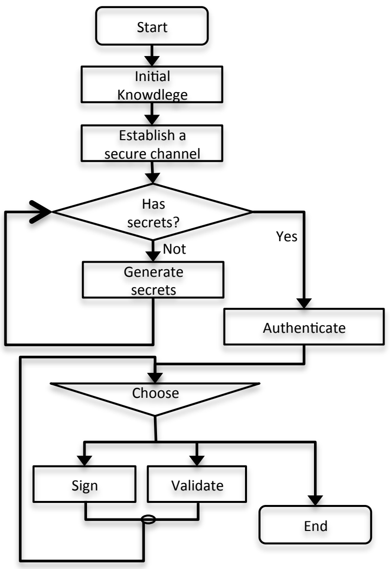

Fig. 1. Flowchart about the use of the framework from client view

the other hand, it also applies to the steganography algorithm, how was st distributed?.

2) Avoiding re-signature: What happens if a malicious client wants to sign an e-document, which has previously been signed (a type of the unbinding problem)?. What happens if a malicious client manipulates the e-document header and its content in order to apparent that it is a not a signed document?.

III. Outline and Notation

A. Outline of the framework

Figure 1 illustrates a flowchart denoting the general steps a client must follow to sign or validate a digital document.

We have developed a framework (see Table VIII), which has been splitted in the following stages:

- **Stage I** consists in establishing a secure channel. We have used the TLS protocol because it is the standard in the e-commerce and it has been subject to a lot of verification proofs (section IV).
- **Stage II** consists in the generation of some secrets (the symmetric key and the steganography algorithm of the client) and the establishment of the username and its password (section V-A). Note that, the generation of the secrets is carried out only the first time when a client is arriving to the system.
- **Stage III:** the agent uses the username and password to obtain authentication (V-B).
- **Stage IV:** once authenticated, the agent can sign any e-document (using cryptography and watermarking technique) that had not previously been signed (using the secrets indirectly created in Stage II). In addition, the agent also can certify the authenticity of any signed e-document; additionally, the client can also exit to the system (section VI).

As you can see, our signing an e-document process implies the use of cryptography, steganography and security protocol techniques. The following two sub-sections describe the notation used in the rest of the document.

B. Notation

1) Host level notation: Cryptography has two main mechanisms:

- Symmetric cryptography, the same key is used to encode and decode a message; by convention, symbol $\{\!|m|\!\}_{K_{AB}}$ will be used to denote that a message m has been cyphered under the key K_{AB}, which agents A and B know. An agent is a computer process that uses the client-server technology to establish a network communication. Sometimes we will use the terms agent or client indisctinctly.
- Asymmetric cryptography, two keys are used to encode and decode messages; an agent (e.g. A) has a key pair (public $K_{pub(A)}$ and private $K_{priv(A)}$). One key used to encode and the other one to decode (reciprocally); usually, key $K_{priv(A)}$ is kept in secret.

Table I summarizes previous notations:

TABLE I
Conventions in host level

Abbreviation	Description
A, B, C	Agents or clients able to establish communication with other agents or clients.
S	A trusted agent, best known as trusted Server
K	Symmetric key
K_{AB}	Symmetric key shared with agents A and B
$K_{pub(A)}$	Public key of Agent A
$K_{priv(A)}$	Private key of Agent A
$\{\!|m|\!\}_K$	Message m cyphered under key K
$m1; m2$	$m1$ and $m2$ are concatenated using symbol ;
N_a, N_b,N_c	Nonces, which are random unguessable numbers never been used before.

2) Notation at network level: Although the verification was carried out with the AVISPA tool, for easy representation issues we will use Alice and Bob notation, see table II.

TABLE II
Conventions in network level

Abbreviation	Description
Initial knowledge: $A : ik$	Initially agent A knows ik.
Message Sending and Receiving: $n.A \to B : m$	At step n agent A sends message m to agent B, which B receives.
Local process representation: $n.A \to B : m$ $m2 = f(m)$ $n2.A \to B : m2$	Between steps n and $n2$, message $m2$ was generated by B as a local process $f(m)$.

IV. Stage I: Establishing a secure channel

Transmitted messages on public networks are sensible to be seen for active or passive attackers who are able to manipulate them in order to take advantage of the situation. Therefore, it is necessary to use secure *cryptographic protocols*. There is a protocol so-called Transport Layer Security (TLS) and its predecessor, Secure Socket Layer (SSL), work over the transport layer of TCP/IP. The TLS protocol allows client/server agents to communicate across a hostile network; eventually, only the server is authenticated. Mutual authentication requires a public key infrastructure for clients, characteristic that is not available in an *uncontrolled environment*.

We have used the TLS protocol because, besides being the standard in e-commerce, the identity of the client is not relevant while consulting digital signatures and thus, it is suited for this process. However, for the signing process it has been extended because this process requieres mutual authentication, see next section.

Table III shows a detail description using Alice and Bob notation, its explanation is as follows:

- Initially, the trusted server knows its public and private key and both the client and the server know their own identities and the certification authority S_{CA}, who knows all issued public keys, here Key denotes a set of all public keys.
- Then, the following steps are carried out in order to establish the secure tunneling:
  - **First step:** a client connects to a TLS server requesting a secure connection (in plain-text) and presents a list of supported CipherSuites (ciphers and hash functions), $[Lc]$. Each session is identified by a session id N_C.
  - **Second step:** from list, the server picks the strongest cipher and hash function that it also supports $(f([Lc]))$ and notifies the client of the decision. The server sends back its identity in the form of a digital certificate and in plain-text. The certificate and the plain-text usually contains the server name S, the server's public encryption key and the trusted certificate authority S_{ca}.
  - **Third step:** the client may contact the server that issued the certificate and confirms that the certificate is valid before proceeding.
  - **Forth step:** the certification authority sends back to the client a confirmation about the credibility of the key.
  - **Fifth step:** in order to generate the session keys used for the secure connection, the client encrypts a random number $N_{C'}$ with the server's public key and sends the result to the server. Only the server should be able to decrypt it, using its private key.
- Finally, from the random number and the session ids, both parts generate a session key K'_{CS} for encryption and decryption. This new knowledge will be used in the following stage.

V. Stage II and Stage III: Generating Secrets to Obtain Authentication to the System

For signing or/and certifying an e-document it is necessary for a client to be authenticated with the server. The authentication procedure requires a username and a password, which are developed after the secure channel had been established between the client and the Server (using the TLS protocol). The first time that the client wants to be authenticated, he agrees a username and a hashed password; and the server generates a secret shared key and a secret steganography algorithm. These secrets are mapped with the username and the password. Obviously, it is the user responsability to ensure his password. Following two sub-sections explain in detail these processes.

TABLE III
SSL and TLS Protocol

Description
Initial Knowledge
$C : C$
$\mathsf{S} : K_{pub(S)}; K_{priv(S)}; \mathsf{S}$
$S_{ca} : \forall K_{pub} \in \mathsf{Key}$
Generating a session secret key
1.-$\mathsf{C} \rightarrow \mathsf{S} : C; N_C; [Lc]$
2.-$\mathsf{S} \rightarrow \mathsf{C} : f([Lc]); \mathsf{S}; N_S; N_C; K_{pub(S)}; S_{ca};$
$\{\!\vert Hash(\mathsf{S}; N_S; N_C; K_{pub(S)}; S_{ca}) \vert\!\}_{K_{priv(S)}}$
3.-$\mathsf{C} \rightarrow S_{ca}$:verification process with S_{ca}
4.-$S_{ca} \rightarrow \mathsf{C} : \mathsf{S}$ is authenticated
5.-$\mathsf{C} \rightarrow \mathsf{S} : N_C; \{\!\vert C; N_{C'} \vert\!\}_{K_{pub(S)}}$
New Knowledge Accumulated
$C : N_C; K_{pub(S)}; N_{C'};$
$K'_{CS} = Hash(N_{C'}, N_C, N_S)$
$\mathsf{S} : K_{pub(S)}; K_{priv(S)}; \mathsf{S}; N_C; N_{C'};$
$K'_{CS} = Hash(N_{C'}, N_C, N_S)$

A. Generating secrets

In particular, signing an e-document requires that a client should have indirectly a secret shared key and a secret steganography algorithm, which are generated in this stage. These secrets are mapped with the username and password. However, neither the shared key and the steganography algorithm must be known by the client. To do that, the user must establish a secure channel with the Server and obtain authentication. Table IV summarises the protocol in Alice&Bob notation, its description is as follows:

- Firstly, the user must have a shared key with the Server to encrypt all messages exchanged between them (it means to establish a secure channel). This is done using K'_{CS}, which is a session key obtained using the TLS protocol, see section IV for more details.
- Secondly, the user must generate his username and password through the following steps:
  - **First step:** the client sends his username u, and a list of personal data Pd including his email @.
  - **Second step:** the server sends back the username and a data obtained of the client's personal information, $f(Pd)$.
  - **Third step:** then, the client must generate the final password P, but only the hash is sent, $Hash(P)$. Sending $Hash(P)$, we can ensure that P will be only known by C.
- Finally, the server creates a steganography algorithm al, a shared key K_{CS}, which the system maps with the client username and hashed password. Note that, al and K_{CS} are unknown to C.

The security properties that the protocol must provides are: P to be known only by C, K_{CS} to be known only by S and $Hash(P)$ known by C and S; and obtain strong authentication between the participants.

We have verified this protocol using The AVISPA Tool Web Interface, available via http://www.avispa-project.org/ finding the following:

- The protocol provides secrecy.
- We have proved that the protocol provides *authentication* from the client to the server on message $f(Pd)$ and *weak authentication* from the server to the client on message $Hash(P)$.

- When we proved *strong authentication*, it failed due to the lack of freshness property generating The Dolev and Yao attacker (of AVISPA tool) a replay attack. However, this attack is already not possible when we consider that K'_{CS} is fresh, which was obtained in an immediately previous step. Even though, we could include timestamps in all steps, however, that is not necessary.

TABLE IV
GENERATING SECRETS TO SING E-DOCUMENTS

Initial Knowledge
$C : K'_{CS}; [u :: @ :: Pd]$
$\mathsf{S} : K'_{CS}$
Generating secret key and steganography algorithm
1.-C $\rightarrow$ S : $\{\![u :: @ :: Pd]\!\}_{K'_{CS}}$
S generates $f(Pd)$ mapped with u
2.-S $\rightarrow$ C : $\{\!u; f(Pd)\!\}_{K'_{CS}}$
C generates a final password P
3.-C $\rightarrow$ S : $\{\!u; Hash(P)\!\}_{K'_{CS}}$
S generates as and K_{CS}, which is mapped with u and $Hash(P)$.
New Knowledge Accumulated
$C : N_C; K_{pub(S)}; N'_C; P; u; K'_{CS}$
$\mathsf{S} : K_{pub(S)}; K_{priv(S)}; \mathsf{S}; N_C; N_{C'}; K'_{CS}; al$ $Hash(P); [u :: @ :: Pd]; K_{CS}; al$

B. System authentication process

The authentication process of the system trusts in the users' username and password, so, it is a responsability of the user to ensure his password. However, any system cannot trust only in the username and password control access based in, because this information goes in plain-text. It is important to encrypt the information and to use security protocols in order to avoid possible attacks. So, the goal of this stage is to establish authentication with the system. Table V describes the authentication part and the explanation is as follows:

- Both the client and the server store all knowledge accumulated of previous steps and the client establishes a new session with the server using the TLS protocol, from that, they agree with session key K'_{CS} and then, the following messages are cyphered using this session key.
- However, we know that the TLS protocol only establishes a secure channel, but, the participants cannot be sure that the communication is really established between them or if some type of man in the middle attack is carried out. So, the following additional steps are proposed:
  - **First step:** the client sends a challenge to the server, a cyphered message using the $Hash(P)$ like a symmetric key containing a new nonce N_{C2}.
  - **Second step:** the server responds to the client with a new challenge, firstly includes the client's nonce N_{C2} and one of the personal data that the server knows about the client. This information has been cyphered too with the $Hash(P)$.
  - **Third step:** Once the client has accepted the previous step, it means that he has accepted the session key, so, the client sends back his nonce N_{C2} and his personal information that the server used like a challenge response.
- Finally, after the proofs the security properties that the protocol provides are:
  - The protocol provides secrecy of N_{C2}, since it is known only by C and S.
  - We have proved that the protocol provides *authentication* from the client to the server on message N_{C2}.
  - We have found that the protocol provides *strong authentication* on I. Since, the server is the only one who knows it, besides C.

TABLE V
AUTHENTICATION PROTOCOL

Process
Initial Knowledge
$C : N_C; K_{pub(S)}; N'_C; u; P; [Pd];$ K'_{CS}
$\mathsf{S} : K_{pub(S)}; K_{priv(S)}; \mathsf{S}; N_C; N_{C'}; K'_{CS}; Hash(P);$ $K_{CS}; [Pd]$
Authentication
1.-C $\rightarrow$ S : $C; N_C; \left\{\!\middle\| u; \{\!u; S; N_{C2}\!\}_{Hash(P)} \middle\|\!\right\}_{K'_{CS}}$
2.-S $\rightarrow$ C : $S; C; N_C; \left\{\!\middle\| \{\!u; N_{C2}; I\!\}_{Hash(P)} \middle\|\!\right\}_{K'_{CS}}$
3.-C $\rightarrow$ S : $\{\!u; N_{C2}; I\!\}_{K'_{CS}}$
New Knowledge Accumulated
$C : N_C; K_{pub(S)}; N'_C; u; P; [Pd];$ $K'_{CS}; N_{C2}$
$\mathsf{S} : K_{pub(S)}; K_{priv(S)}; \mathsf{S}; N_C; N_{C'}; K'_{CS}; Hash(P);$ $K_{CS}; [Pd]; N_{C2}$

VI. STAGE IV: SIGNING AND VALIDATING E-DOCUMENTS

A. Signing an E-document

This stage begins onces the client has been authenticated. Table VI describes the process in Alice and Bob notation, the explanation is as follows:

- The initial knowledge consists in all messages generated in the authentication process, see V-B.
- The signing process consists in sending the e-document and receiving it, already signed:
  - **First step:** here, the client sends the e-document F to be signed and the information that will validate the document PD. The server signs the e-document using the secret key K_{CS} (which only the server knows), which maps with the client, the hashed password and the steganography algorithm al generated in the stage II, table IV. See more details below.
  - **Second step:** then, the server sends back the new signed e-document F' together with the hash of the old e-document Hash (F).
- Once carried out this process one can verify the document authenticity executing the consulting process, explained in the following subsection.

1) Signing: Tradicionally, signing a document (in a controlled environment) using only cryptography consists in encoding a message m using the agent's private key, e.g. $\{\!m\!\}_{K_{priv(A)}}$, as stated by [12]:

TABLE VI
SECURE SIGNING PROTOCOL

Process
Initial Knowledge $C : N_C; K_{pub(S)}; N'_C; u; P; [Pd]; N_{C2};$
Signing 1.-C $\rightarrow$ S : $\{\!\|u; F; \{PD\}\|\!\}_{K'_{CS}}$ it is considered that $false = signed(F)$ $F' = signing(F, \{PD\}, al, K_{CS})$ 2.-S $\rightarrow$ C : $\{\!\|u; F'; \text{Hash } (F)\|\!\}_{K'_{CS}}$

Any message encrypted under $K_{priv(A)}$ comes originally from A, unless A is compromised.

Note that, this proposition hold *iff* all agents in the network initially know whom each public key belongs to.

Let F be an e-document, which has been sent to B as follows: $\{\!|F|\!\}_{K_{priv(A)}}$. B can assume that F was signed by agent A. However, C cannot assume that F has previously been signed by A if B sends only F.

Anderson and Needham [1] proposed a principle: *Do not assume the secrecy of anybody else's "secrets" (except possibly those of a certification authority).*

So, the problem is that in an uncontrolled environment the case of C is very common, we can not trust that a message was previously signed using a private key, even though the key was generated by a trusted certification authority. Thereby, we must combine techniques as steganography and cryptography, as we do below:

$$F' = signing(F, \{PD\}, al, K_{CS})$$

F' denotes the new e-document and it provides the following properties:

$$\exists m.(m \in getSign(F', al)) \wedge (m \notin getSign(F, al)) \wedge$$
$$(F \overset{\odot}{\approx} F') \wedge (hash(F) \neq hash(F')) \wedge$$
$$size(F) = size(F') \wedge signed(F') \wedge !signed(F)$$
$$\rightarrow$$
$$m = \{PD\}$$

Here, operator $\overset{\odot}{\approx}$ denotes that F is similar to F' according to the human view in average. Function $getSign(F, al)$ returns the watermarked information applying al algorithm; function $size(F)$ denotes the size of a file in bytes and $signed(F)$ is a function returning true if F has already been signed before.

Function $signed(F)$ is introduced to avoid a malicious client signing a previously signed e-document. If the document header contains evidence of a signature, the signing operation must be rejected.

B. Validating the signature of an e-document

Although authentication was not necessary for this step, we consider forcing it because the information that the server and the client exchange must be guaranteed to be secure. Hence, the client must ensure that the answer comes from the server. Table VII describes the process in Alice and Bob notation, and the explanation is as follows:

- In a similar way of signing process, the initial knowledge here consists in all messages generated in the authentication process, see V-B.
- The certification process consists in sending the signed e-document and receiving a notification about the answer, as follows:
  - **First step:** the e-document to be certified F' is sent, together with the username (this must be a valid user).
  - **Second step:** then, the server checks its header and identifies who has signed it. The server compares it with the information extracted in its body. If the integrity persists, the server internally looks the corresponding client key and reveals the signature of the document, otherwise, a reject notification is displayed.
- Finally, with the result issued by the server, the client may certify the e-document.

TABLE VII
SECURE CONSULTING SIGNTURES PROTOCOL

Process
Initial Knowledge $C : N_C; K_{pub(S)}; N'_C; u; P; [Pd]; N_{C2};$ $K'_{CS}; [Pd]$ S : $K_{pub(S)}; K_{priv(S)};$ S$; N_C; N_{C'}; K'_{CS};$ $Hash(P); N_{C2}; K_{CS}; al; [Pd]$
Secure Consulting process 1.-C $\rightarrow$ S : $\{\!\|u; F'\|\!\}_{K'_{CS}}$ it is considered that $true = signed(F')$ $\{PD\} = revealing(F', al', K_{CS})$ 2.-S $\rightarrow$ C : $\{\!\|\{PD\}; \text{Hash } (F')\|\!\}_{K'_{CS}}$

The following formula is used to know who has signed an e-document:

$$\{PD\} = revealing(F', al', K)$$

where al' is the inverse algorithm of al.

Table VIII provides the general framework. In general, we have used java as programming language and it has permitted to extend our application using JSP and JavaScript for the Internet environment. For TLS protocol we have used KeyTool; for encryption issues we have developed xoring algorithm together with AES; to hide messages we implemented the least significant bit steganography algorithm. Finally, we can sign and obtain signatures in formats like BMP (bitmap image file), and PNG (portable network graphics).

VII. CONCLUSIONS AND FURTHER WORK

Throughout this document we described a secure framework to sign and validate remotely e-documents. This framework is compound by various cryptographic protocols based on a previous work, which we have formally verified using The AVISPA Tool Web Interface. We have proved this framework in various individual protocols and they provide *agreement*, the fourth (last) level of authentication according to the hierarchy of Lowe, [14].

The framework provides authentication between the client requesting a service and the server as a responder and service generator. In addition, the framework also provides secrecy properties, in this case ensure the transmitted messages and guarantees a secure signature carried out by the server.

TABLE VIII
GENERAL FRAMEWORK FOR SIGNING AND CERTIFYING E-DOCUMENTS

	Initial Knowledge $C : C$ $\mathsf{S} : K_{pub(S)}; K_{priv(S)}; \mathsf{S}$ $S_{ca} : \forall K_{pub} \in \mathsf{Key}$	
I	**Establishing a secure channel** 1.-C $\rightarrow$ S : $C; N_C; [Lc]$ 2.-S $\rightarrow$ C : $f([Lc]); \mathsf{S}; N_S; N_C; K_{pub(S)}; S_{ca};$ $\left\{\!\left\lvert Hash(\mathsf{S}; N_S; N_C; K_{pub(S)}; S_{ca}) \right\rvert\!\right\}_{K_{priv(S)}}$ 3.-C $\rightarrow$ S_{ca} :verification process with S_{ca} 4.-S_{ca} $\rightarrow$ C : S is authenticated 5.-C $\rightarrow$ S : $N_C; \{\!\lvert C; N_{C'} \rvert\!\}_{K_{pub(S)}}$	
II	**Generating Secrets** 1.-C $\rightarrow$ S : $\{\!\lvert [u :: @ :: Pd] \rvert\!\}_{K'_{CS}}$ S generates $f(Pd)$ mapped with u 2.-S $\rightarrow$ C : $\{\!\lvert u; f(Pd) \rvert\!\}_{K'_{CS}}$ C generates a final password P 3.-C $\rightarrow$ S : $\{\!\lvert u; Hash(P) \rvert\!\}_{K'_{CS}}$ S generates as and K_{CS}, which is mapped with u and $Hash(P)$.	
III	**Authentication** 1.-C $\rightarrow$ S : $C; N_C; \left\{\!\left\lvert u; \{\!\lvert u; S; N_{C2} \rvert\!\}_{Hash(P)} \right\rvert\!\right\}_{K'_{CS}}$ 2.-S $\rightarrow$ C : $S; C; N_C; \left\{\!\left\lvert \{\!\lvert u; N_{C2}; I \rvert\!\}_{Hash(P)} \right\rvert\!\right\}_{K'_{CS}}$ 3.-C $\rightarrow$ S : $\{\!\lvert u; N_{C2}; I \rvert\!\}_{K'_{CS}}$	
IV	**Signing** 1.-C $\rightarrow$ S : $\{\!\lvert u; F; \{PD\} \rvert\!\}_{K'_{CS}}$ $F' = signing(F, \{PD\}, al, K_{CS})$ 2.-S $\rightarrow$ C : $\left\{\!\left\lvert u; F'; \mathsf{Hash}\ (F) \right\rvert\!\right\}_{K'_{CS}}$	**Validating** 1.-C $\rightarrow$ S : $\left\{\!\left\lvert u; F' \right\rvert\!\right\}_{K'_{CS}}$ $\{PD\} = revealing(F', al', K_{CS})$ 2.-S $\rightarrow$ C : $\left\{\!\left\lvert \{PD\}; \mathsf{Hash}\ (F') \right\rvert\!\right\}_{K'_{CS}}$

Our research is ongoing, we have planned further verification, but now in an real environment, problems or vulnerabilities than can arise while moving from mathematical expressions to real implementation.

Another research we are working is about remote document authenticity once it has already been printed and later on scanned. In addition, we are considering to deal with *the unbinding problem*, Memo and Wong [16], in the sense that, not other distributors can apply a re-signature.

REFERENCES

[1] R. Anderson and R. Needham. Robustness principles for public key protocols. In Don Coppersmith, editor, *Proceedings of the 15th Annual International Cryptology Conference, CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 1995.

[2] AVISPA Team. *AVISPA v1.0 User Manual*, v1.0 edition, June 2005.

[3] Stephen H. Brackin. A HOL extension of GNY for automatically analyzing cryptographic protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop, CSFW'96*, pages 62–77. IEEE Computer Society, 1996.

[4] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426(1):233–71, 1989.

[5] Franco Frattolillo. Watermarking protocol for web context. *IEEE Transactions on Information Forensics and Security*, 2(3-1):350–363, 2007.

[6] Franco Frattolillo and Salvatore D'Onofrio. A web oriented watermarking protocol. In *IEC (Prague)*, pages 91–96, 2005.

[7] Chin-Laung Lei, Pei-Ling Yu, Pan-Lung Tsai, and Ming-Hwa Chan. An efficient and anonymous buyer-seller watermarking protocol. *Image Processing, IEEE Transactions*, 13:1618–1626, 2004.

[8] Yusuk Lim, Changsheng Xu, and David Dagan Feng. Web based image authentication using invisible fragile watermark. In *Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing - Volume 11*, VIP '01, pages 31–34, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.

[9] J. C. López-Pimentel and R. Monroy. Formal support to security protocol development: a survey. *Journal Computacin y Sistemas, special volume celebrating 50 years of Computing in Mexico*, 12:89–108, 2008.

[10] J. C. López-Pimentel, R. Monroy, and Vctor F. Ramos F. Symmetric cryptography protocol for signing and authenticating digital documents. In *Proceedings in the Communications in Computer and Information Science (CCIS)*, Series of Springer LNCS, pages 9–23. Springer, 2011.

[11] J. C. López-Pimentel, Vctor F. Ramos F, and R. Monroy. A web service for signing and authenticating digital documents based on symmetric cryptography protocol. *International Journal of Digital Information and Wireless Communications (IJDIWC2011)*, 1:434–444, 2012.

[12] Juan Carlos López-Pimentel. *On the Automated Correction of Faulty Security Protocols*. PhD thesis, Tecnológico de Monterrey, Campus Estado de México, 2008.

[13] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using FDR. In *Proceedings of the 2nd International Workshop on Tools and Algorithms for Construction and Analysis of Systems, TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[14] Gavin Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop, CSFW'97*, pages 31–44. IEEE Computer Society, 1997.

[15] Catherine Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[16] N. Memon and P. W. Wong. A buyer-seller watermarking protocol. *IEEE Trans. Image Processing*, 10:643–649, 2001.

[17] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal in Computer Security*, 6(1-2):85–128, 1998.

[18] V.M. Potdar, S. Han, and E. Chang. A survey of digital image watermarking techniques. *INDIN '05. 3rd IEEE International Conference on Industrial Informatics*, 1(1):709–716, 2005.

[19] Lintian Qiao and Klara Nahrstedt. Watermarking schemes and protocols for protecting rightful ownership and customer's rights. *Journal of Visual Communication and Image Representation*, 9(3):194–210, 1998.

[20] X. D. Song, S. Berezin, and A. Perrig. Athena: A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1-2):47–74, 2001.

[21] G. Steel, A. Bundy, and M. Maidl. Attacking the asokan-ginzboorg protocol for key distribution in an ad-hoc bluetooth network using coral. In H. König, M. Heiner, and A. Wolisz, editors, *IFIP TC6 /WG 6.1: Proceedings of 23rd IFIP International Conference on Formal Techniques for Networked and Distributed Systems*, volume 2767, pages 1–10, Berlin, Germany, 2003. FORTE 2003 (work in progress papers).

[22] F. J. Thayer Fabrega, J.C. Herzog, and J.D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings Symposium on Security and Privacy.*, pages 160–171. IEEE computer Society, 1998.

[23] Minerva M. Yeung and Fred Mintzer. An invisible watermarking technique for image verification. *Image Processing, International Conference on*, 2:680, 1997.