The SIP security enhanced by using pairing-assisted Massey-Omura signcryption

Alexandre M. Deusajute

***Abstract*—Voice over IP (or VoIP) has been adopted progressively not only by a great number of companies but also by an expressive number of people, in Brazil and in other countries. However, this crescent adoption of VoIP in the world brings some concerns such as security risks and threats, mainly on the privacy and integrity of the communication. The risks and threats (which we can emphasize the man-in-the-middle attack, because his high danger degree) already exist in the signalling process to the call establishment. This signalling process is executed by specific types of protocols, like SIP (Session Initiation Protocol). After doing a bibliographical revision of the current SIP security mechanisms and analyzing some proposals to improve these mechanisms, we verified that the SIP vulnerability to the man-in-the-middle was not totally solved. Then we propose a new security mechanism for SIP in this paper, aiming both to be an alternative security mechanism and to solve the vulnerability to the man-in-the-middle attack. In our proposal we use a protocol for secure information exchange – the Massey-Omura protocol – which, when combined with Pairing-based Cryptography (PBC), provides a high security level for SIP in all its aspects.**

***Index Terms*—man-in-the-middle, Massey-Omura, pairing, SIP, VoIP**

I. Introduction

VOICE over IP (VoIP) is being adopted by an increasingly great number of enterprises to replace the traditional circuit switched infrastructure used for telephony services. Many service providers are seeking to enhance their messaging capabilities through the new IP telephony infrastructure instead of investing further in the traditional infrastructure. At the same time, the evolving IP Telephony infrastructure provides the opportunities of introducing new value added services, such as conferencing, web collaboration and online gaming. [1]

Nevertheless, as VoIP is based on normal IP networks, VoIP applications inherit the known and unknown security weaknesses that are associated to the IP protocol [2]. The signalling/control and the media data might be the major target of attacks. Even if we try to secure the VoIP traffic based on the IPsec security framework, two main factors would affect voice traffic when IPsec was used: the increase of the packet size and the prolonged time required to encrypt payload and headers. Besides this, the authentication as provided with IPsec is point-to-point (not end-to-end), that is, it protects machines only (logical address), whereas the users themselves are not identified as it should be desired in an end-to-end security [3]. The same occurs with SSL (Secure Sockets Layer) and TLS (Transport Layer Security).

VoIP calls are susceptible to DoS (Denial-of-Service) attacks, hacked gateways leading to unauthorized free calls, call eavesdropping, malicious call redirection, SPIT (Spam over Internet Telephony), and so forth. VoIP also presents certain specific security challenges. In order to avoid these kinds of attacks, both parties of a VoIP call – the call setup and the media stream itself – must be inspected. [4], [5]

The concern about the VoIP security increases if we consider the current scenario of expansion and adoption of the IP Telephony. It is estimated that in the year 2010 25% of all households in Western Europe will have abandoned traditional Public Switched Telephone Network (PSTN) services in favor of VoIP [6]. In Brazil, at the end of 2006, there were about 262.000 VoIP telephony subscribers. It is estimated that this number has increased to 600.000 subscribers till September 2007. Moreover, the VoIP providers have provoked a fall in the price of the minute in Embratel's international calls. [7]

In view of this whole crescent adoption of VoIP in the world (and, consequently, the increase of security risks and threats, including incidents and attacks), efforts to create security patterns for VoIP and for the media traffic were started some years ago. Several work groups of the IETF (Internet Engineering Task Force) have approved a series of RFCs (Request for Comments) aiming to establish security patterns for the protocols, which can be signalling (to make the call setup) or transport (to transfer the media from one place to the other) protocols.

The media transport protocol normally used is the RTP (Real-time Transport Protocol [8]). For this protocol it was established a specific security profile, the SRTP (Secure Real-time Transport Protocol [9]). This profile provides authentication and privacy to the media data transported. It was designed to add small overload on the packet size and to minimize the number of cryptographic keys that should be shared between two communication nodes. But the own profile does not define, in its specification, a scheme to exchange cryptographic keys and other security parameters between the nodes.

The solution for a key exchange scheme came from another work approved by IETF: the MIKEY (Multimedia Internet KEYing [10]). MIKEY offers mechanisms for a safe and reliable key management. Other advantages of MIKEY are the good use of the band and the low computational effort. The scheme offered by MIKEY was studied and has evolved, according to the RFC-4650 [11] and, more recently, according to a new improvement proposal that was submitted to the IETF [12]. It is interesting to note that the most recent

The author is master's degree student at LARC – Laboratório de Arquitetura e Redes de Computadores – at Escola Politécnica of the University of São Paulo (USP) – Av. Prof. Luciano Gualberto, tr. 3, 158 BR 05508-900 – São Paulo – SP – Brazil. Mail to: adeusajute@larc.usp.br

improvements in MIKEY have a common point: the concern about man-in-the-middle type attacks. Such improvements are making the MIKEY's cryptographic key exchange scheme stronger, by solving the little that remained from the SRTP vulnerabilities.

Another protocol, but of signalling type, that was benefited with SRTP and MIKEY was the H.323 one. With the establishment of the H.235 version 2 standard in November 2000, the ITU-T (International Telecommunications Union – Telecommunication Standardization sector) took a step towards interoperability by defining different security profiles to the H.323. This was necessary because the standard itself did not mandate particular features. The defined profiles provided different security levels and described a subset of possible security mechanisms. The H.235 version 3 already forecasts improvements as the introduction of AES (Advanced Encryption Standard). And, even although in discussion, the H.235v3 appendix G already forecasts the use of SRTP and of MIKEY in a combined way. [13]

However, for SIP (Session Initiation Protocol [14]), other promissory signalling protocol which is reaching acceptance by the market, the security is a subject that is not totally solved. Security problems with SIP refer to the RFC-2543 [15], which originated the SIP. In that RFC, the main mechanism to provide security was the PGP (Pretty Good Privacy). Although it was efficient which concern to the privacy (since it uses the foundations of the asymmetrical cryptography and RSA cryptographic keys), PGP is vulnerable to the man-in-the-middle attack. The RFC-3261 [14] makes obsolescent the RFC-2543. One of the improvements introduced by that new RFC was the change of the main security mechanism, passing from PGP to S/MIME (Secure Multi-purpose Internet Mail Extension). Although the change has brought gains in terms of security, the own RFC-3261 [14, p.247] admits that the vulnerability to the man-in-the-middle continues affirming that the security mechanisms foreseen by SIP are not completely unfailing against that attack type.

In this paper we propose an alternative security mechanism so that two parties communicating one with the other by VoIP, in a peer-to-peer (or, more precisely, endpoint-to-endpoint) mode, can, during the SIP signalling process to establish and setup the call, exchange a certain secret information in a safe way and not vulnerable to the man-in-the-middle attack. This secret information exchanged could be, for example, a cryptographic key to be used after in a RTP session to provide privacy to the conversation between two parties. Or could also be any information such as an encrypted SDP message. Thus, our collaboration is both to provide SIP a cryptographic key exchange scheme by using the own signalling process (that is, without needing an additional scheme, like MIKEY) and to offer an alternative to the current security mechanism (the S/MIME) which is used to give privacy to the signalling process. Our proposed scheme was based on another protocol to information exchange, the Massey-Omura protocol, whose sequence of message exchange is similar to the sequence of message exchange in a typical SIP signalling process. Although the Massey-Omura protocol already have certain security degree, this one is improved with the use of Pairing-based Cryptography (PBC).

Besides the Introduction, the rest of the paper is organized as follows: in section II we present some related works. In section III we show the fundamental concepts that will allow a better understanding of our proposal. In section V we describe our proposal in details, with comments about security and performance aspects. In section VI we summarize and suggest the future works that can be derived from this work.

II. Related works

The RFC-3329 [16] tries to solve the vulnerability to the man-in-the-middle attack in a SIP signalling scenario by using TLS and IKE (Internet Key Exchange) which is an IPsec's protocol and it is similar to MIKEY. However, IKE is more appropriate for SIP signalling scenarios using Proxies and not for peer-to-peer scenarios (like our proposal). Besides this, IKE is not a general end-to-end proposal, even for scenarios with Proxies. In order to provide end-to-end security for SIP signalling scenarios using Proxies, there are some good works proposed, as the one from Kim and Kim [5].

In another related work it is proposed the use of MIKEY messages both in the SDP and in the SIP message body [17]. We are going to explore this work a little bit because, even though our proposal was not based or inspired on that work, the motivations and the initial problems were similar.

The work presented by Bilien, Eliasson, Orrblad and Vatn [17] indicates that MIKEY messages need to be carried inside SIP messages as part of the signalling process for the call establishment by using SIP. The question is how to do that ? There are two project aspects related with this: how the MIKEY messages should be codified / encapsulated and which SIP messages should be used to carry those codified MIKEY messages.

As for how to code/encapsulate, there are two approaches. The first one is based on the RFC-4567 [18], which has instituted the use of specific extensions on SDP aiming the cryptographic key management. One of these extensions is the "key management attribute" (or "key-mgmt" for short) which allows MIKEY messages to be codified / encapsulated in SDP, as shown in the following example:

```
v=0
s=Secret discussion
t=0 0
c=IN IP4 lost.example.com
a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyO...
a=key-mgmt:keyp1 727gkdOshsuiSDF9sdhsdKnD/dhsoSJokdo7eWD...
a=key-mgmt:keyp2 DFsnuiSDSh9sdh Kksd/dhsoddo7eOok727gWsJD...
m=audio 39000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 42000 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

Note that the attribute "key-mgmt" can be used to offer, besides MIKEY, two more possibilities of protocols to exchange cryptographic keys, inside the SDP. Each attribute "key-mgmt" carries the data of the pertinent protocol, codified in base64.

The previous scheme works well when MIKEY is used as key exchange protocol on SRTP. However, when MIKEY is used with IPSec plus ESP (Encapsulation Security Payload), perhaps the SDP attribute is not the most correct location for a MIKEY message.

In order to use the MIKEY as an IPSec/ESP key management protocol, it was proposed a different approach, which is the second approach about how to code/encapsulate MIKEY messages. [19]

The second approach suggests that the MIKEY message be codified as a MIME message (Multipurpose Internet Mail Extensions) of multiple parts in the SIP message body. That is, instead of carrying a MIKEY message as a SDP attribute, it is suggested that the MIKEY message be carried in the MIME body of a SIP message. This approach is a more suitable solution for the established connections case using IPSec/ESP. And, in order to have MIKEY messages carried as a MIME payload, a correspondent MIME type has to be registered. The possibility of this approach was proved by Orrblad [19].

About which SIP messages to be used, Bilien, Eliasson, Orrblad and Vatn have proposed the use of an INVITE message to carry the initial MIKEY message [17]. As response to the initial MIKEY message (that is, the ending of the key exchange process) the following possibilities are offered, all of them associated with some SIP response: "200 Ok", "180 Ringing" ou "183 Session in Progress". We will explain these SIP basic concepts in subsection III-A.

III. Background

In this section, we will show the basic knowledge concerning the concepts which will be used in the paper, so that the reader can better understand our proposal. The concepts will be presented in a very objective way. If the reader wants to know more about the concepts, we recommend the additional reading of bibliographical references mentioned during the concepts explanation.

A. SIP

1) General features: signalling protocols are used to session establishment, modification and ending. One of these signalling protocols is SIP.[1] After the session was established, the media (audio, video, etc) can be transmitted by using some specific media transport protocol, like RTP.

Fig. 1 shows an example of signalling procedure using SIP. Note that, after having finished the signalling process, the media transport starts. And soon after the media transport ends, SIP is used again to end the session established previously. The scenario presented in the figure is of peer-to-peer type, which will be treated in this paper.

In a peer-to-peer scenario, each communicating party is called *user agent* (UA). An UA takes an instruction or information supplied by an user and acts as an agent on the behalf of that user to establish and to end media sessions with other *user agents*. An UA can assume a client role (user agent client – UAC) when emitting requests for another UA that, in this case, assumes a server role (user agent server – UAS) and it answers the requests made by the user agent client.

The interaction between user agents in a SIP session is made by *messages*. A SIP message can be a request or a response.

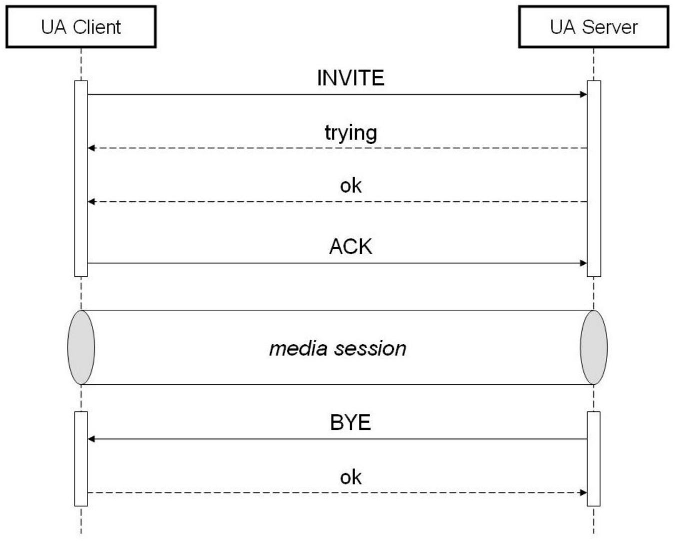

Fig. 1. Example of signalling process using SIP

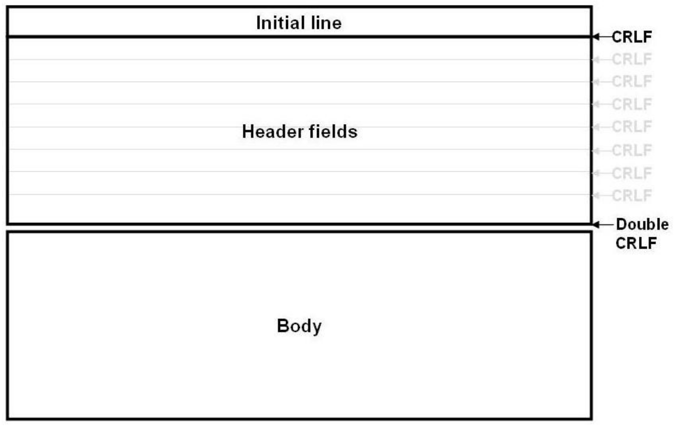

Fig. 2. SIP message – request or response – general structure (CLFR – Carriage Return/Line Feed – corresponds to a line change)

The *requests* are considered "verbs" in the protocol, because they request that a specific action be executed by another user agent. In the signalling process from Fig. 1 there are three types of SIP requests: INVITE (an "invitation" to establish the media session between the user agents), ACK (to confirm the reception of an INVITE's response) and BYE (to end a session previously established). But there are other requests such as CANCEL (to finish pending surveys or attempts for the session establishment)[2].

The *responses* are messages generated by an user agent server, to answer a request made by an user agent client. SIP admits several response types, grouped in six classes. The first five classes ("Provisional", "Successful", "Redirection", "Request Failure" and "Server Failure") were copied from the HTTP (HyperText Transfer Protocol [20]). The sixth class ("Global Failures") was created exclusively for SIP.

The general structure of a SIP message (showed in Fig. 2)

[1]Except for some eventual specific mention – or citation – the content of this subsection was based mainly on the RFC-2068 [20], on Johnston [21] and on the RFC-3261 [14].

[2]In the original SIP's RFC (the RFC-3261 one) there was only six requests. Before that, other RFCs were published and they have introduced more SIP requests.

is compounded by the following parts:

Initial line: its composing depends on the message type and it can be:

1) *Request-Line*: is a request name, followed by a Request-URI (Universal Resource Indicator) plus the protocol version. All information is separated by a simple space character (SP) and, at the end, there is a CLRF (Carriage Return/Line Feed). A Request-URI (or SIP-URI) indicates the user or the service to which the request is addressed. In other words, it is the request receiver.

2) *Status-Line*: is the protocol version followed by three digits numeric code (Status-Code) plus a text which explains the meaning of the numeric code. All of these elements also are separated by a simple space character (SP) and, at the end, there is a CRLF as well. This kind of Initial Line is normally presented in responses.

Header fields: SIP admits one or more header fields in only one message (request or response). That is one of the SIP features which make it very flexible. In the SIP specification there are a lot of header fields, grouped by their types. Thus, there are the following header field types: generic (since they can be used in any SIP message type), specific for requests, specific for responses and those ones for entities.

The mandatory header fields for an INVITE request are "Call-ID", "Contact", "CSeq", "From", "Supported", "To" and "Via". Except for "Supported", all the other header fields have a common characteristic: they are copied *ipsis litteris* to the response which is given for the INVITE request, no matter which response is. Besides this, it is possible to add other "tags" to the header field "To".

In an ACK request case, the mandatory header fields are "Call-ID", "CSeq", "From", "To" and "Via".

It is important to emphasize that INVITE and ACK requests also have some optional header fields but they need some values to avoid undesirable situations. These optional header fields are: "Authorization" (to carry authentication credentials of an user agent); "Content-Disposition" (to describe the SIP message body's function on the communication that was established); "Content-Encoding" (to specify the codification scheme applied to the SIP message body); "Content-Length" (to indicate the number of octets in the SIP message body); "Content-Type" (to indicate the SIP message body's media type).

Referring to the header field "Content-Length": when the number of octets is 0 (zero), it means that the SIP message does not have a body. This header field is optional because it is possible the generation of message bodies in a dynamic way, where the body size cannot be known *a priori*. However, if it is not supplied, an unnecessary consumption of bytes can occur in the message. For example, if the number of octets is not indicated in an UDP message, it is assumed that the SIP message body continues until the end of the datagram, even if the body payload is smaller than the datagram payload.

Body: is the part of a SIP message that can contain several types of information, including SDP (Session Description Protocol) information. This information can be about the media (not the media itself), or about QoS (Quality of Service), or even about security. It is important to emphasize that SDP is a protocol and, thus, it serves to describe the media streaming initialization parameters, which, in practice, is the content that we see or listen. SDP is the SIP message body default format and the more recent RFC which treat it is the RFC-4566 [22].

Requests must have an initial line, one or more header fields (some of them are exclusive for requests, we mean, they cannot be used in response messages) and a body. Responses must have an initial line, header fields and can or cannot have a body, depending on the response numeric code. For example, the response "200 – OK" have a body when the previous request is an INVITE request message.

2) Security threats: the information transmitted in the signalling protocols messages can be as sensitive and important as the own content of the session, that is, the media itself. Both the header fields and the body in a SIP message can contain secret information which must be protected.

Butcher, Li and Guo [3] have presented and described a series of threats against SIP, such as registration hijacking, message modification, CANCEL/BYE attacks, redirects, and others. Most of the time, the difficulty to defend is caused by the own SIP message structure. As SIP incorporates elements from HTTP to carry command data, it is very flexible and extensible to implement VoIP characteristics. On the other hand, it becomes very difficult for a SIP *parser* to test all the possible entries. Eavesdroppers can explore these vulnerabilities by creating and sending packets with malformed commands inside them to some networks nodes. These actions will certainly degrade the attacked nodes (perhaps causing "out of order" on the nodes) and a whole VoIP system can become unavailable .

Another attack type which SIP is susceptible to is the man-in-the-middle attack, as we can see in next subsection.

B. The man-in-the-middle attack

The *man-in-the-middle* is a kind of attack that occurs over a communication between two parties (a sender and a receiver) and it is performed by someone who wants to monitor the communication between two parties without tampering with the data and without exposing its own existence. It may modify the ciphertext stream in any manner whatsoever (deleting, delaying, substituting, or inserting ciphertexts) as long as it does not change the cleartexts received by the communicating parties. But, if it wants to monitor the communications for a long period of time, it would have to try to behave as transparently as possible, since any trace it leaves in the cleartexts is likely to arise suspicion [23]. In other words, the cleartext received by both communicating partiess does not suffer any modification.

The SIP is also susceptible to man-in-the-middle attacks. In those scenarios where there is a Proxy, an eavesdropper can impersonate a legitimate user agent, register itself with the Proxy and replace the legitimate registration with its own address. This way, those who access the Proxy to communicate with the legitimate user agent, will communicate with the malicious user agent. In peer-to-peer SIP scenarios, the eavesdropper can intercept the messages and modify some or all of the message attributes. Yet in this scenario attack, other actions can be performed by the eavesdropper, such as to redirect the messages to a third party. A serious problem coming from the impersonation is that the eavesdropper can send BYE messages at any moment, ending the communication and generating a kind of "scheduled DoS (Denial of Service)". [3]

In VoIP specific case, a man-in-the-middle attack can take place where the attacker is able to listen to the conversation between the two victims and also alter the communication. This includes playback of previously captured speech so that the receiver hears a different message from that the sender sent. Due to the unpredictable nature of human conversations, this attack may be difficult to be detected and it is much more efficient in a conversation as minor is the voice piece captured and reproduced later. Let's take for instance a situation where it is possible to change "no" to "yes" in response to a question of participation or "sell" to "buy" in a conversation with a financial advisor. The attack in those scenarios can be more disastrous if in the voice piece captured there is financial information, not coming from the victim but from the other communicating party [3]. The attacker could even introduce messages like "Sorry, the system cannot conclude the transaction." in the place of an authentic message indicating that the transaction was successfully concluded... but on the attacker behalf.

C. The Massey-Omura protocol

The Massey-Omura scheme [24], [25] is a three-stage encryption protocol that, like the Diffie-Hellman key agreement scheme [26], allows two parties which do not share any secret data to exchange confidential information over a non-secure channel. Despite having been published in the 80th decade, the Massey-Omura protocol had already been reported (not publicly) in the previous decade [27]. The most general form of the Massey-Omura protocol requires a *commutative encryption scheme*.

An *encryption scheme* is a tuple $(\mathcal{K}, \mathcal{M}, \mathcal{E})$ where $\mathcal{K}$ is the set of *keys*, $\mathcal{M}$ is the set of *messages*, and $\mathcal{E} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{M}$ is a mapping (the *cipher*) such that, for each $k \in \mathcal{K}$, $\mathcal{E}$ defines a permutation over M. Thus, it makes sense, for each $k \in \mathcal{K}$, to speak of the *inverse cipher* satisfying $\mathcal{E}^{-1}(k, c) = m \Leftrightarrow \mathcal{E}(k, m) = c$ for all m, $c \in \mathcal{M}$ (m is the *plaintext* and c is the *ciphertext*).

An encryption scheme is *commutative* if, for any two keys s_A, $s_B \in \mathcal{K}$ and any message $m \in \mathcal{M}$, the same ciphertext results from encrypting m first under s_A and then under s_B, or else encrypting m first with s_B and then with s_A, i.e. $\mathcal{E}(s_B, \mathcal{E}(s_A, m)) = \mathcal{E}(s_A, \mathcal{E}(s_B, m))$.

If Alice wants to send a message $\boldsymbol{M}$ to Bob by using the Massey-Omura protocol, she encrypts the message with her key (s_A) and sends the result to Bob; Bob encrypts what he has received with his key (s_B) and sends the new result back to Alice; she decrypts Bob's response (and gets the message encrypted by Bob's key only) and sends the result back to Bob, who finally decrypts and gets the original message. Note that Alice does not know (and does not need to know) Bob's key to communicate with him – and vice-versa.

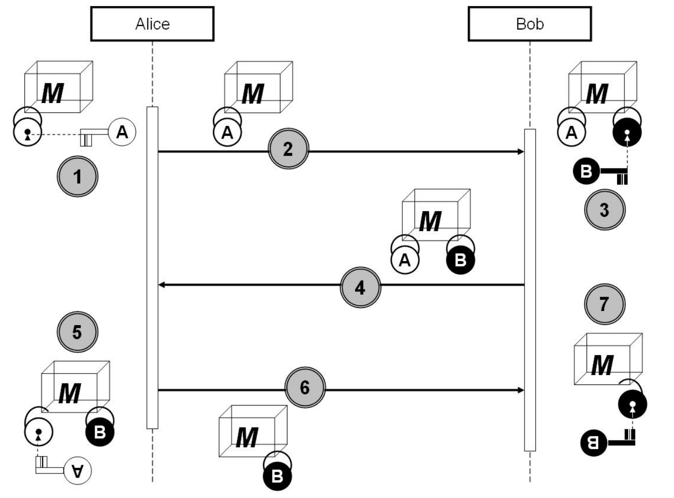

Fig. 3. Secret information exchange by using the Massey-Omura protocol

Fig. 3 illustrates the Massey-Omura protocol.

1) The Massey-Omura protocol against the man-in-the-middle attack: there is nothing in the Massey-Omura basic scheme that Bob can use to check if it was really Alice who sent him the message (in a similar manner, Alice cannot know whether the reply comes from Bob or from someone else). Bob cannot even check whether he gets the correct message without asking Alice. These restrictions prevent Bob and Alice to check each other and leave the protocol susceptible to some attack types, as the man-in-the-middle one.

Following, we will describe some attack scenarios where it is possible to understand better the Massey-Omura protocol vulnerabilities and its susceptibility to the man-in-the-middle attack. In these scenarios, let MITM be the abbreviation for man-in-the-middle. So, s_X is the MITM private key generated randomly. And M_λ is a generated spurious message.

Scenario 1: MITM sends a spurious message indistinguishable from an authentic message, taking advantage over a legitimate information.

1) MITM begins the communication.
   It computes $M_X = s_X M_\lambda$ and sends M_X to Alice, pretending to be somebody else (note that here Alice is the receiver, that is, she is "Bob")
2) Alice receives M_X.
   Following the protocol as the receiver, she computes $M_{AX} = s_A M_X$ and sends the result back.
3) MITM receives M_{AX}.
   It applies s_X^{-1} over M_{AX}: $s_X^{-1} M_{AX} = s_X^{-1} s_A M_X = s_X^{-1} s_A s_X M_\lambda = s_X^{-1} s_X s_A M_\lambda = s_A M_\lambda$.

   Now, if MITM wants, it can send a spurious message to

Bob, impersonating more "truly" Alice (because it can sign the spurious message $\boldsymbol{M}_\lambda$ with Alice private key). And, if MITM applies the same technique to attack Bob, it can impersonate more "truly" Bob as well, assuming a communication from Bob to Alice.

ANALYSIS: MITM does not recover s_A (neither does s_B). But it can create a spurious message and "sign it" indirectly. That is, supposing that Bob has capacity to authenticate Alice's message, depending on the authentication technique, Bob will conclude that the message is indeed from Alice. This is an interesting attack scenario because by simply applying any methods of authentication it does not solve the vulnerability.

Scenario 2: MITM gets the authentic message not directly.

1) Alice computes $M_A = s_A\boldsymbol{M}$ and sends the result to Bob.
2) MITM intercepts the communication and captures M_A, but does not interrupt the communication.
3) Bob receives M_A.
   He follows the protocol and computes $M_{BA} = s_B M_A$ and sends it back to Alice.
4) MITM intercepts the communication once more, captures M_{BA}, discarding it, and it interrupts the communication. So, it computes $M_X = s_X M_A$ (M_A was captured in step 2) and sends M_X to Alice, pretending to be Bob.
5) Alice receives M_X.
   She follows the protocol and computes:
   $M_B = s_A^{-1} M_X = s_A^{-1} s_X s_A \boldsymbol{M} = s_A^{-1} s_A s_X \boldsymbol{M} = s_X \boldsymbol{M}$.
   Alice sends M_B to MITM thinking that it is Bob.
6) MITM receives M_B.
   Thus, to get the original message, it computes:
   $s_X^{-1} M_B = s_X^{-1} s_X \boldsymbol{M} = \boldsymbol{M}$.

ANALYSIS: note that MITM could get the authentic message by interrupting the communication in step 2, not allowing Bob to receive anything. But MITM prefers not to do this because could there is an extra process or procedure (e.g., a confirmation phone call) which ensures Alice to check if Bob received something from her after the first communication. So, if Bob does not receive anything from Alice, she would notice that there is something suspicious and she would interrupt the communication in the beginning. However, even in cases like that (where MITM does not interrupt the first communication between Alice and Bob) and even supposing that there is an extra checking procedure or process, the MITM attack is done successful. Thus, the use of additional verifications, some of them out-of-system, cannot resolve totally the protocol susceptibility to the man-in-the-middle attack.

Those are some possible attack scenarios. Other ones were identified and detailed by Deusajute and Barreto [28].

D. Pairing Based Cryptography

Pairings have been attracting the interest of the international cryptography community because it enables the design of original cryptographic schemes and makes well-know cryptographic protocols more efficient. Due this, Pairing-Based Cryptography (PBC) has been considered an emerging field of Elliptic Curve Cryptography (ECC) that allows a wide range of applications. [29]

In a mathematical point of view, pairings are mappings over elliptic curves, because they map a pair of points from two elliptic curves (sometimes, from only one elliptic curve) over an element belonging to a multiplicative group in a finite field. On the other hand, it is a special sort of mapping because it has certain particular peculiarities which distinguish them solely.

Formally, a *pairing* (or *bilinear pairing*) can be defined as a map $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$, where $\mathbb{G}_0$ and $\mathbb{G}_1$ are two groups of order q for some large prime q, satisfying the following properties [30]:

1) Bilinearity: $e(aP, bQ) = e(P,Q)^{ab}$ for all $P \in \mathbb{G}_0$, $Q \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}$.
2) Non-degeneracy: for every $P \in \mathbb{G}_0$ there is $Q \in \mathbb{G}_1$ such that $e(P,Q) \neq 1$. Observe that, if $\mathbb{G}_0 = \langle P \rangle$ (i.e., G_0 is generated by P) and $\mathbb{G}_1 = \langle Q \rangle$, then $\mathbb{G}_T = \langle g \rangle$ with $g = e(P,Q)$.
3) Computability: there is an efficient algorithm to compute $e(P,Q)$ for all $P \in \mathbb{G}_0$, $Q \in \mathbb{G}_1$.

The "bilinear" designation comes from the fact that the mapping is linear in each of the two points included in the mapping, that is, $e(\alpha P) = e(P)^\alpha$ and $e(\alpha Q) = e(Q)^\alpha$ – in a multiplicative notation. In some literatures, to make a mapping be bilinear it is enough to consider the "Bilinearity" property. So, in those literatures, the properties "Computability" and "Non-degeneracy" are not considered basic properties from bilinear pairings, but only desirable properties so that the pairings can be computationally implemented. In practice, the fact is that "Bilinearity" property is the most important, because it is essential to protocols definitions, no matter what their type are (key agreement, encrypting, decrypting, signature verification, etc).

The property "Computability" is accepted by some authors when it refers to a bilinear mapping computationally implementable. When a bilinear pairing is not computationally implementable, it is to hard to be used that it is inappropriate to use in practice. In other words, although some intractable pairing can be useful in a theoretical analysis (e.g., to prove that there is a finite process to calculate something, even if in an exponential time), in an applied area as cryptography it is not so useful to consider such pairing type.

About the "Non-degeneracy" property, it must exist because there is no sense in using degenerate pairings for cryptographic applications due to the result $e(P,P) = 1$.

The typical form to implement bilinear pairings in practice is by using Weil or Tate pairing derivations. Among them, the Tate pairing is the most used.

An important corollary aroused from the definition previously presented is: if $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a bilinear pairing, then:

$$e(cP, Q) = e(P, cQ) \tag{1}$$

for all $P \in \mathbb{G}_0$, $Q \in \mathbb{G}_1$, and $c \in \mathbb{Z}$.

Proof. Specializing the bilinearity condition $e(aP, bQ) = e(P, Q)^{ab}$ to $a = c$ and $b = 1$ we get $e(cP, Q) = e(P, Q)^c$, and similarly taking $a = 1$ and $b = c$ we get: $e(P, cQ) = e(P, Q)^c$. Therefore, $e(cP, Q) = e(P, cQ)$ $\square$.

IV. Pairing-assisted Massey-Omura signcryption

In III-C1 we have shown some scenarios where the Massey-Omura protocol can be attacked by the man-in-the-middle. The main problem is that both Alice and Bob cannot verify each other during the stages transitions. And, even though they can do this, it is not any verification or any authentication that solves the vulnerability. Even extra systems procedures or processes would not avoid the man-in-the-middle attack as we have seen previously.

It is necessary a mechanism which can provide the protocol some authentication degree. This mechanism must be able to allow both the sender (Alice) and the receiver (Bob) to verify each other in the protocol's transitions. Pairing can be this mechanism. In particular, we use the result of (1), presented in previous section.

Let $\mathbb{G}_0 = \langle P \rangle$, $\mathbb{G}_1 = \langle Q \rangle$, and $\mathbb{G}_T = \langle g \rangle$ (with $\mathbb{G}_0$ and $\mathbb{G}_1$ not necessarily distinct) be groups of prime order n, and $e : \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ a bilinear pairing. Assume that there is a bijection $\pi : \mathcal{M} \rightarrow \mathbb{G}_0$.

Alice wishes to send a message $M = \pi(m)$ to Bob over a non-secure channel. Alice's key pair is $(s_A \in \mathbb{Z}_n^*, V_A = s_A Q)$, where s_A is her private key and $V_A = s_A Q$ is her public key. Similarly, Bob has the key pair $(s_B \in \mathbb{Z}_n^*, V_B = s_B Q)$, where s_B is his private key and $V_B = s_B Q$ is his public key.

The modified Massey-Omura protocol (enhanced) is:

- Step 1: Alice computes $M_A = s_A \boldsymbol{M}$
  Alice still computes a Security Parameter $M_A^\delta = s_A \; h(M_A)$ where h is a conventional "hash" function
  Then, Alice sends the computed results to Bob.

- Step 2: Bob receives M_A and the Security Parameter M_A^δ.
  Check 1: Bob checks whether $e(M_A^\delta, Q) = e(h(M_A), V_A)$. If the equality is not maintained then Bob interrupts the protocol.
  Otherwise he computes $M_{BA} = s_B M_A$ and sends the result back to Alice.

- Step 3: Alice receives M_{BA}.
  Check 2: Alice checks whether $e(M_{BA}, Q) = e(M_A, V_B)$. If the equality is not maintained then Alice interrupts the protocol.
  Otherwise she computes $M_B = s_A^{-1} M_{BA} = s_A^{-1} s_B s_A \boldsymbol{M} = s_A^{-1} s_A s_B \boldsymbol{M} = s_B \boldsymbol{M}$ and sends the result back to Bob.

- Epilogue: Bob receives M_B.
  Check 3: Bob checks whether $e(M_A, Q) = e(M, V_A)$. If the equality is not maintained then Bob interrupts the protocol, refusing the message.

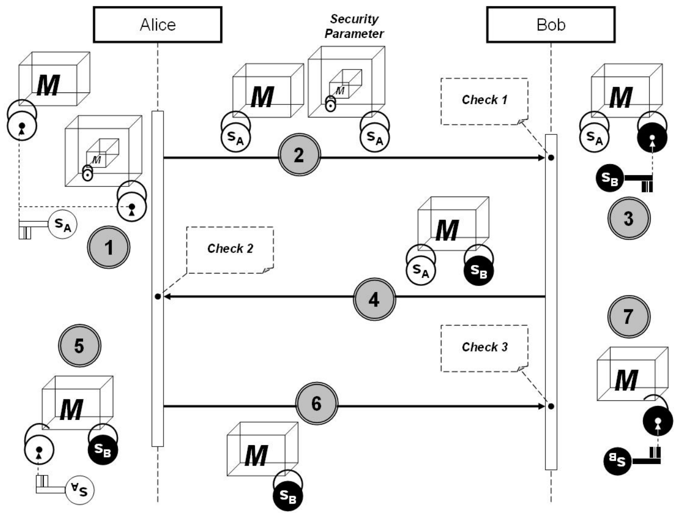

Fig. 4. The modified (and enhanced) Massey-Omura protocol: note that the Security Parameter only persists until the end of the first transition (that occurs because its function is just to make the Check 1 feasible)

Observe that the check points arise from bilinear pairing properties and also from (1):

(Check 1) $e(M_A^\delta, Q) = e(s_A \; h(M_A), Q) =^{(1)} e(h(M_A), s_A Q) = e(h(M_A), V_A)$.

(Check 2) $e(M_{BA}, Q) = e(s_B M_A, Q) =^{(1)} e(M_A, s_B Q) = e(M_A, V_B)$.

(Check 3) $e(M_A, Q) = e(s_A M, Q) =^{(1)} e(M, s_A Q) = e(M, V_A)$.

Fig. 4 shows schemetically the modified Massey-Omura protocol.

The attack scenarios presented in III-C1 are now exhibited under another optic.

Scenario 1 (*revised*): MITM sends a spurious message indistinguishable from an authentic message, taking advantage over a legitimate information.

1) MITM begins the communication.
   It computes $M_X = s_X \boldsymbol{M}_\lambda$ and $M_X^\delta = s_X h(M_X)$. So, it sends M_X and M_X^δ to Alice, pretending to be somebody else (notice that here Alice is the receiver, that is, shes "Bob").
2) Alice receives M_X and M_X^δ.
   Following the modified protocol as the receiver role, she has to perform the Check 1. To do this she has to use the public key from whom is trying to communicate with her. But, who is trying to communicate with her ?
   Let $T = \{V_K | V_K = s_K Q\}$ be the set of all public key in which Alice knows and trusts. $V_X \notin T$, where $V_X = s_X Q$ is the MITM public key. Then:
   $$e(M_X^\delta, Q) = e(s_X h(M_X), Q) = e(h(M_X), s_X Q) = e(h(M_X), V_X) \neq e(h(M_X), V_K).$$
   As the equality is not maintained for any $V_K \in T$, Alice

interrupts the protocol.

Scenario 2 (*revised*): MITM gets the authentic message not directly.

1) Alice computes $M_A = s_A\boldsymbol{M}$ and $M_A^\delta = s_A h(M_A)$. After doing this, she sends the results to Bob.
2) MITM intercepts the communication capturing M_A and M_A^δ, but does not interrupt the communication.
3) Bob receives M_A and M_A^δ.
   He follows the modified protocol, performing the Check 1 without troubles (because no adulteration happens caused by the MITM). So, Bob computes $M_{BA} = s_B M_A$ and sends it back to Alice.
4) MITM intercepts the communication again, captures M_{BA}, discarding it, and it interrupts the communication. So, it computes $M_X = s_X M_A$ (M_A was captured in step 2) and sends M_X to Alice, pretending to be Bob.
5) Alice receives M_X thinking of being M_{BA}.
   She also follows the modified protocol and checks whether $e(M_{BA}, Q) = e(M_A, V_B)$:

$$e(M_X, Q) = e(s_X M_A, Q) = e(s_X s_A\boldsymbol{M}, Q) = \\ e(s_X\boldsymbol{M}, s_A Q) = e(s_X\boldsymbol{M}, V_A) \neq e(M_A, V_B).$$

   and

$$e(M_X, Q) = e(s_X M_A, Q) = e(s_X s_A\boldsymbol{M}, Q) = \\ e(s_A\boldsymbol{M}, s_X Q) = e(M_A, V_X) \neq e(M_A, V_B).$$

   As the equality is not maintained anyway, Alice interrupts the protocol.

The security for the other scenarios was presented by Deusajute and Barreto [28]. There is also a detailed explanation in that work about the use of the additional computing $M_A^\delta = s_A H_A$, which we named Security Parameter.

V. Enhancing the SIP security by using the Massey-Omura protocol plus Pairing-based Cryptography

Consider a VoIP communicating scenario, peer-to-peer, with two user agents, a *caller* (or user agent client – UAC) and a *listener* (or user agent server – UAS). We will name the caller "user agent Alice" (UA Alice) and the listener "user agent Bob" (UA Bob). Assume that the channel through which the UA Alice communicate with the UA Bob is non-secure and can be attacked by a third party, the user agent man-in-the-middle.

Alice (UA Alice's user) wants to make a call to Bob (UA Bob's user), by VoIP. The signalling protocol used for the call establishment is SIP. During the signalling process, the UA Alice wants to send to UA Bob a secret content which can be, for instance, a symmetric cryptographic key. That key was generated to be used later on a RTP session to provide the privacy service to the conversation between Alice e Bob, by using of some symmetric cryptographic algorithm compatible with RTP (e.g., 3DES).

We will describe in following subsections our proposal to enable the secure exchange of the secret content between the UA Alice and the UA Bob, by taking advantage of SIP dynamic signalling process to establish a call.

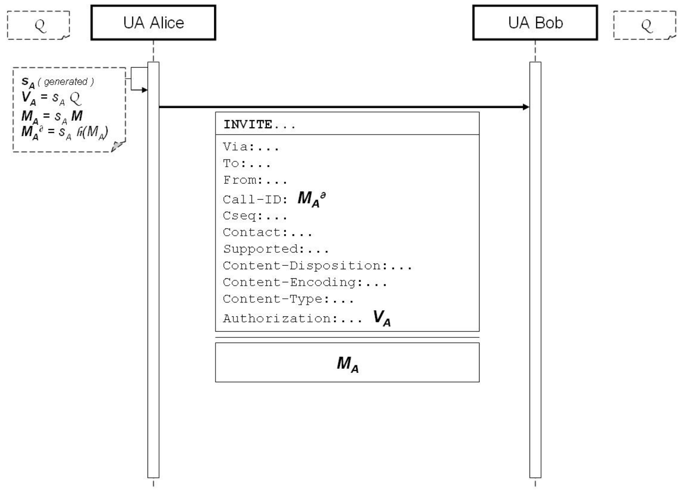

Fig. 5. Step 1: the UA Alice sends an INVITE to the UA Bob

A. Step 1: INVITE

1) Pre-existing information:

- Q: a public known value. In practice, it would be any point in an appropriate selected elliptic curve. This is very important because when using Elliptic Curve Cryptography nor all curves are appropriate for cryptography. There are curves which do not offer security with respect to computational aspects. That is, they are curves where it is possible to perform a decryption in a computationally feasible polynomial time. The elliptic curves which must not be used in cryptography are the degenerated, anomalous and supersingulars ones. [31], [32]
  Since Q is a point, there are two values included, one corresponding to the abscissa and another corresponding to the ordinate. Those two values could be previously recorded in suitable electronic circuits of the VoIP equipments which correspond to the user agents. Or even they could be built-in as binary "hard-code" in the executable program of softphones. That is possible, without generating security problems and vulnerabilities because, if a suitable elliptic curve was selected, then any point could be used without the need of periodic values changes.

2) Verification points: no verifications in this step.

3) Computation:

1) **UA Alice's private key generation – s_A:** this private key is generated randomly, but with high security level, so that it cannot be guessed easily or even "broken" by a brute-force attack, where the attacker tries all possible keys on a piece of ciphertext until an intelligible translation into plaintext is obtained [33]. There are several algorithms which can be used to generate this

private key – such as that one in the RFC-1750 [34] – and that can be easily implemented by software and by hardware.

2) **UA Alice's public key generation – V_A:** $V_A = s_A Q$. Once Q is a point and s_A is an integer number, even though reasonably large, V_A would be another point on the elliptic curve which is appropriately selected and from that one comes the Q point.

3) **Secret content encryption:** $M_A = s_A \boldsymbol{M}$
$\boldsymbol{M}$ represents any secret content. It is important to emphasize that because, despite all the preoccupation with the secure exchange of the cryptographic keys which can be used after in a RTP session, our proposal enable the exchange of any secret content, depending on the application requirements. For instance, the UA Alice could not only transmit a cryptographic key to be used in an eventual RTP session but could also transmit a SDP message concatenated to the cryptographic key. UA Bob would only have the additional work to find out where the concatenation occurs (e.g., a CRLF could be the "delimiter" of this concatenation) and, then, undo it.

4) **Security Paramenter computation – M_A^δ:**
$M_A^\delta = s_A h(M_A)$.
The h is a conventional hash function. That is, it is a hash function (e.g., SHA-1) which receives as its input the UA Alice's private key (s_A) and returns the value corresponding to the private key's cryptographic hash. The justification for the use of this Security Parameter is better explained by Deusajute and Barreto [28, p.08].

The UA Alice must be able to retain, in some manner, the encrypted secret content M_A and the other generated data, because part of the information will be used afterwards.

4) *SIP message preparation:*

The header fields "Via", "To", "From", "CSeq", "Contact" and "Supported" must be prepared according to the RFC-3261 [14]. The header fields "Content-Disposition", "Content-Encoding" and "Content-Type" must contain the following values respectively: "session", "compress" and "text/text". A brief explanation about the meaning of those header fields was given in subsection III-A. A detailed information can be obtained in the RFC-3261.

The other header fields must be prepared appropriately to carry some of the data which were previously calculated:

"Call-ID" must contain the Security Parameter M_A^δ, but not in the "...@hostname" format.

The use of "Call-ID" by this way could represent a violation of the specification made in the RFC-3261 for that header field. However, it is possible to proceed this way with "Call-ID" due to the following arguments:

- Typically, that header field contains random values followed by "@hostname". However, the use of "@hostname" is not mandatory. This non-obligatoriness is justified by the use of the keyword "MAY" within the specification made in the RFC-3261 [14, p.37]) for the header field "Call-ID" (in some examples of the own RFC-3261, the authors do not use "Call-ID" in the format "...@hostname").

- The RFC-3261 uses the keyword "RECOMMENDED" to indicate that the header field "Call-ID" should be a random and cryptographically generated value, as it is in RFC-1750 [34]. However, the interpretation for that keyword given in the RFC-2119 [35] allows us to use an alternative computation mechanism to generate the "Call-ID" value, because there is a relevant reason in this particular situation – the security of the our proposed scheme – to not obey (at least not in this point) the RFC-1750. Moreover, the way by which we are proposing the "Call-ID" value corresponds to a computation result that uses, both direct and indirectly, a value – s_A – which can be generated according to the own RFC-1750, as it was previously proposed. That is, "Call-ID" would be a random and cryptographically generated value anyway.

"Authorization", specifically the "auth-param" parameter, must contain the UA Alice's public key V_A. This header field contains authentication credentials from an UA. When an UAS (in this case, the UA Bob) receives a request from an UAC (in this case, the UA Alice), the UAS can authenticate the call originator before the request is processed by the UAS. [14, p.194]

In respect to the SIP message body, it must be prepared so that it is possible to carry the secret content *M*.

5) *SIP message instance for this step:*

```
INVITE sip:alice@larc.usp.br SIP/2.0
Via: SIP/2.0/UDP larc.usp.br:5060
To: Bob the Builder <sip:bob@poli.usp.br>
From: Alice in Wonderland <sip:alice@larc.usp.br>
Call-ID: 082121f32b42a2187835d330a...
CSeq: 1 INVITE
Contact: sip:alice@larc.usp.br
Supported: 100rel
Content-Disposition: session
Content-Encoding: compress
Content-Type: text/text
Authorization: Digest usernament="UA Alice",
         realm="larc.usp.br",
         auth-param="84a4cc6f3082121f32b42a2187831a9e..."

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756...
```

B. *Step 2: 200 OK*

1) *Pre-existing information:*

- Q: a public known value (already explained in Step 1)

- h: the UA Bob has previous understanding that the UA Alice has used the hash function h to encrypt the secret content M_A. This "previous understanding" can

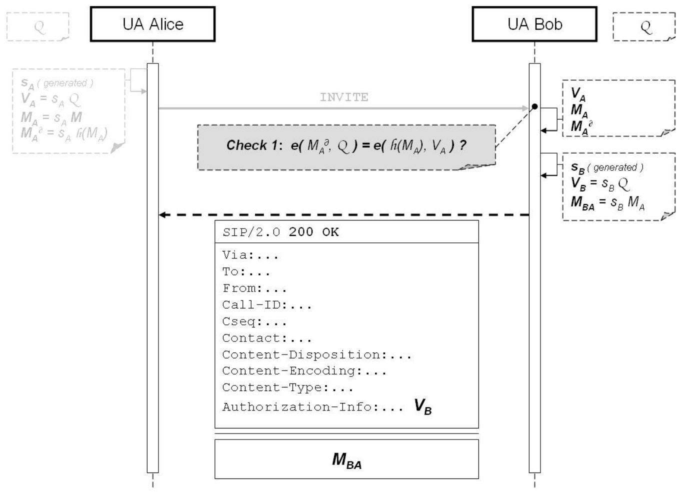

Fig. 6. Step 2: the UA Bob responds to the UA Alice

be due to an accepted criterion or due to a previously established agreement among the user agents.

2) *Verification points:* :

1) **Request type:** the UA Bob is waiting for an INVITE request. If it receives a request different from INVITE, the UA Bob prepares a SIP response message – specifically, the "603 Decline" one – and it sends to the UA Alice. Otherwise, it goes to the next verification point.

2) ***Check 1*:** the UA Bob checks whether $e(M_A^\delta, Q) = e(h(M_A), V_A)$.

   In order to perform this checking, the UA Bob must be able to parse the SIP message received from the UA Alice aiming to extract the Security Parameter M_A^δ (contained in the header field "Call-ID"), the UA Alice's public key V_A (contained in the header field 'Authorization", specifically in the "auth-param" parameter) and the encrypted secret content M_A (contained in the SIP message body). The UA Bob does not use M_A directly. The UA Bob must apply the hash function on the M_A value and use the result in the checking process.
   If $e(M_A^\delta, Q) \neq e(h(M_A), V_A)$ then the UA Bob prepares a SIP message response – specifically, the "401 Unauthorized" one – and it sends to the UA Alice. Otherwise, it goes to the next verification point.
   The UA Bob must be able to retain, in some manner, the parsed data V_A e M_A, because they will be used afterwards.

3) **UAS evaluation:** in this point, the UA Bob notifies its user (Bob in person) that he has a call. Depending on the elapsed time Bob has to answer the phone, the UA Bob may prepare and send to UA Alice a "Provisional" class response (where the message code have the format 1xx), to report that some action is being taken, but there is not a definitive answer yet. For instance, if the UA Bob sends to the UA Alice a "180 Ringing" response, it means that Bob's telephone is ringing. That is, the user Bob has already been notified – by the ring tone – that there is a call to him, but he has not answered the phone yet.

3) *Computation:*

1) **UA Bob's private key generation – s_B:** this private key is also generated randomly and the considerations are the same from UA Alice's private key generation.

2) **UA Bob's public key generation – V_B:** $\quad V_B = s_B Q$.

   Once Q is a point and s_B is an integer number, even though reasonably large, V_B would also be another point on the elliptic curve which is adequately selected and from that one comes the Q point.

3) **Secret content (already encrypted) encryption:** $\quad M_{BA} = s_B M_A$.
   It can be strange to have to encrypt something that has already been encrypted. However, the double encrypting is the great differential of the Massey-Omura protocol, enabling it to encrypt the exchanged information between two parties, without these parties to share their generated secret keys. In other words, each party generates a private key, keeping it with itself, without sharing it with the other party (therefore, one party does not know the private key value of the other party). And, even thus, it is possible to exchange information in a secret manner. Secret but not totally secure, due to the man-in-the-middle attack. Thus, it is necessary an additional "security layer", which is provided in our proposal by using Pairing-Based Cryptography.

The UA Bob must be able to retain, in some manner, the generated data, because some of them will be used afterwards.

4) *SIP message preparation:*

The values contained in the header fields "Via", "To", "From", "CSeq" and "Contact" of the INVITE request received by the UA Bob must be copied, without any changes, to the correspondent header fields in the SIP message response "200 OK".

The header field "Authentication-Info" must be prepared in an appropriate manner to carry the UA Bob's public key V_B. Specifically, the "nextnonce" parameter must be used specifically to carry the public key. An UAS (in this case, the UA Bob) may include this header field in a 2xx response to a request that was successfully authenticated using digest based on the "Authorization" header field. [14, p.164]

In respect to the SIP message body, it must be prepared so that it is possible to carry the secret content encrypted for the

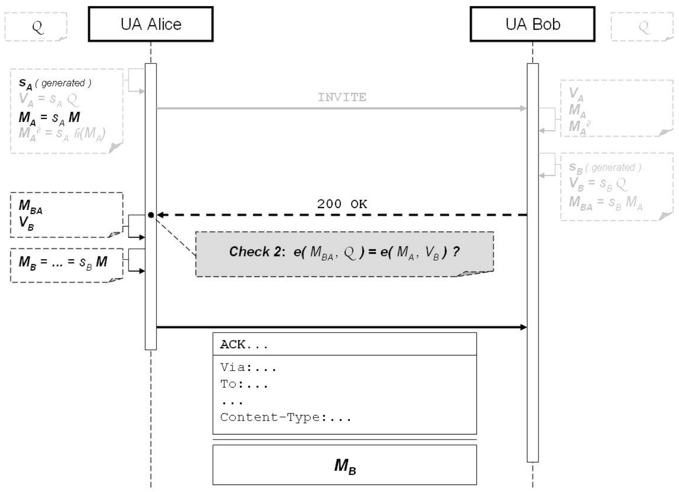

Fig. 7. Step 3: the UA Alice sends an ACK to the UA Bob

second time M_{BA}.

5) SIP message instance for this step:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP larc.usp.br:5060
To: Bob the Builder <sip:bob@poli.usp.br>
From: Alice in Wonderland <sip:alice@larc.usp.br>
Call-ID: 082121f32b42a2187835d330a...
CSeq: 1 INVITE
Contact: sip:alice@larc.usp.br
Content-Disposition: session
Content-Encoding: compress
Content-Type: text/text
Authentication-Info: nextnonce="08212f3a4cc6321783a9e..."

nj756tbB9HG4VQpfyF467GhIGfHfYT6ghyHhHUujhJhjH77n8HHGTrfvb
hyHhHUujhJh756tbB9HGT4VQpfyF467GhIGfHfYT6jH77n8HHGgrfvbnj
n8HHB9HG4VQbnj7567GhIGfHfYT6gGTrfvhJhjH776tbhyHhHUujpfyF4
T64VQbnj7GhIGfHfY756...
```

C. Step 3: ACK

1) Pre-existing informations:

- Q: a public known value (already explained in Step 1)
- M_A: a value already computed by the own UA Alice in Step 1.
- s_A: a value generated and retained by the UA Alice since Step 1.

2) Verification points: :

1) **Request type:** the UA Alice is waiting for a SIP response message, or from the "Provisional" class (1xx) or even from the "Successful" class (2xx). If it receives a response from another class different of "Provisional" and "Successful", the UA Alice prepares and sends to the UA Bob a CANCEL type SIP message, aborting the call establishment process.
   If it receives a "Provisional" response (1xx), the UA Alice keep waiting for a new response from the UA Bob. When the UA Alice receives a "Successful" response (2xx), it goes to the next verification point.
2) ***Check 2*:** the UA Alice checks whether $e(M_{BA}, Q) = e(M_A, V_B)$.

   In order to perform this checking, the UA Alice must be able to parse the SIP message received from the UA Bob aiming to extract the UA Bob's public key V_B (contained in the header field 'Authentication-Info", specifically in the "nextnonce" parameter) and the re-encrypted secret content M_{BA} (contained in the SIP message body).
   If $e(M_{BA}, Q) \neq e(M_A, V_B)$ then the UA Alice prepares and sends to the UA Bob a CANCEL type SIP message, aborting the call establishment process. Otherwise, no more verifications have to be done.

3) Computation: the UA Alice computes a new encrypted secret content M_B.

$$M_B = s_A^{-1} M_{BA} = s_A^{-1} s_B s_A \boldsymbol{M} = s_A^{-1} s_A s_B \boldsymbol{M} = s_B \boldsymbol{M}$$

Notice that the result M_B in an unique one secret content, encrypted by the UA Bob's private key s_B. That is, the UA Alice only has removed the "security layer" that it own has applied over the initial secret content (***M***), in Step 1. However, the UA Alice "see" the result s_B ***M*** as an unique one piece, without distinguishing where s_B begins and where it ends, and also where ***M*** begins and where it ends.

4) SIP message preparation:

The header fields "Via", "To", "From", "CSeq", "Contact" and "Supported" must be prepared according to the RFC-3261 [14]. The header fields "Content-Disposition", "Content-Encoding" and "Content-Type" must contain the following values respectively: "session", "compress" and "text/text". As mentioned, a brief explanation about the meaning of those header fields was given in subsection III-A. Detailed information can be obtained in the RFC-3261.

In respect to the SIP message body, it must be prepared so that it is possible to carry the new encrypted secret content M_B.

5) SIP message instance for this step:

```
ACK sip:alice@larc.usp.br SIP/2.0
Via: SIP/2.0/UDP larc.usp.br:5060
To: Bob the Builder <sip:bob@poli.usp.br>
From: Alice in Wonderland <sip:alice@larc.usp.br>
Call-ID: 082121f32b42a2187835d330a...
CSeq: 1 INVITE
Content-Disposition: session
Content-Encoding: compress
Content-Type: text/text

HfYT6ghyHhHUujhJhjH77nnj756tbB9HG48HHGTrfvbVQpfyF467GhIGf
T4VQpfyF467GhIGHhHUujhfHfYTGgrfvbn6jH77n8HHhyJh756tbB9HGj
hyHhHUujpfyF4n8HHB9HG4VQbnj7567GhIGfHfYT6gGTrfvhJhjH776tb
HfY756T64VQbnj7GhIGf...
```

D. Epilogue: secret content *M* retrieving

1) Pre-existing information:

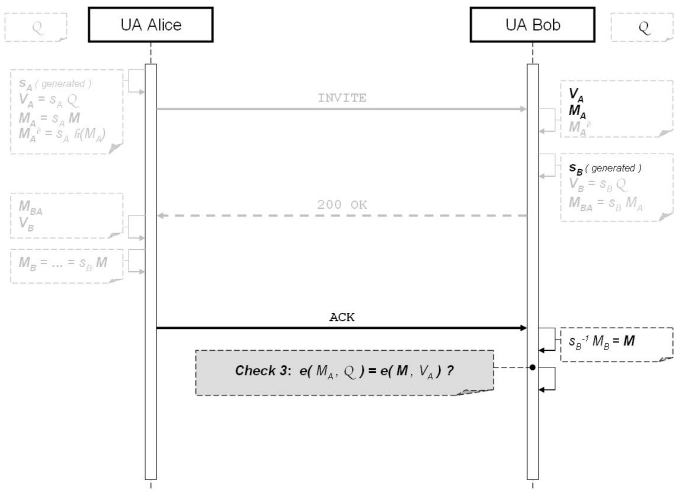

Fig. 8. Epilogue: the UA Bob retrieves the secret content sent by the UA Alice

- Q: a public known value (already explained in Step 1)
- M_A: a value retained by the UA Bob since Step 2.
- V_A: a value retained by the UA Bob since Step 2.
- s_B: a value generated and retained by the UA Bob since Step 2.

2) Verification points: :

1) **Request type:** the UA Bob is waiting for an ACK request. If it receives a request different from ACK, the UA Bob prepares a BYE type SIP message and sends it to the UA Alice. Notice that it is possible to the UA Bob already to send a BYE because it may be assumed that a session was established. The UA Bob has received an INVITE and it has answered these INVITE with a "200 OK". [21]

   Before it goes to the next (and final) verification, the UA Bob must retrieve the secret content which, in this moment, is only encrypted by the UA Bob's private key. That is, to retrieve the secret content ***M***, it is enough that the UA Bob uses its secret key (s_B) to perform a decryption job:

$$s_B^{-1} M_B = s_B^{-1} s_B \boldsymbol{M} = \boldsymbol{M}$$

   To be sure that the UA Bob has retrieved, indeed, the secret content sent by the UA Alice, it is enough to do the following verification.

2) ***Check 3*:** the UA Bob checks whether $e(M_A, Q) = e(M, V_A)$.

   To perform this checking, the UA Bob uses the information that was retaining since Step 2 (M_A e V_A) plus the secret content ***M*** decrypted newly and the well known public information Q.

   If $e(M_A, Q) \neq e(M, V_A)$ then the UA Bob prepares and sends to the UA Alice a BYE type SIP message, ending the established session. Otherwise, the secret content ***M*** can finally be used for its purposes.

Fig. 9 shows all the new proposed signalling process.

E. Security and performance aspects

In the previous subsections, our proposed scheme took advantage of the big similarity of the Massey-Omura protocol with the typical SIP signalling process, to enable a secret content exchange between two user agents (and without sharing the secret keys which were generated by the own user agents).

The Massey-Omura protocol by itself only ensures the privacy of the exchanged information. Thus, even if the secret content is captured, it is not possible to discover it by using brute-force attack or cryptanalysis. However, a man-in-the-middle attack could capture not only the secret content but also other data in traffic between the user agents, so that they can be compounded to discover the secret content. The same kind of attack could also be used to try spoofing some of the user agents, making them think they are communicating one with the other when, indeed, they are communicating with the man-in-the-middle (which could take advantage of this situation to change or replace the data in traffic). To solve this vulnerability, it was necessary to use Pairing-Based Cryptography. Thus the security is complete.

All of what was presented in subsection IV. More details about the security verifications for Massey-Omura enhanced by using pairing can be obtained in Deusajute and Barreto's work [28].

As for about the performance, it is necessary to pay attention principally to the check points, which use pairing computations. The other computation jobs can be implemented in a well optimized manner by several ways. It is important to notice that the check points may be the "neck" of the proposal if the pairing computations are not well implemented. In some practical experiences, the use of pairing has been an interesting alternative, when well implemented. Just to illustrate, we emphasize the work of Zhang and Kim [36] where pairing was used in a practical situation and it was compared with RSA [36, p.08]. Those authors used the Weil Pairing which, normally, has a performance smaller than the Tate pairing (in other words, the results obtained by Zhang and Kim [36, p.08] could still be better).

Exactly due to its best computational performance, the Tate pairing is the most used in practical situations. It can be computed by the Miller's algorithm in sub-exponential time. And, as an improvement for the Miller's algorithm, it was created a faster algorithm, the BKLS [37].

There are other alternatives for pairings, as the Eta pairing [38] which is based on Tate pairing, but computed by a more effective manner. That is, under certain conditions it is computationally faster than the Tate pairing. The final

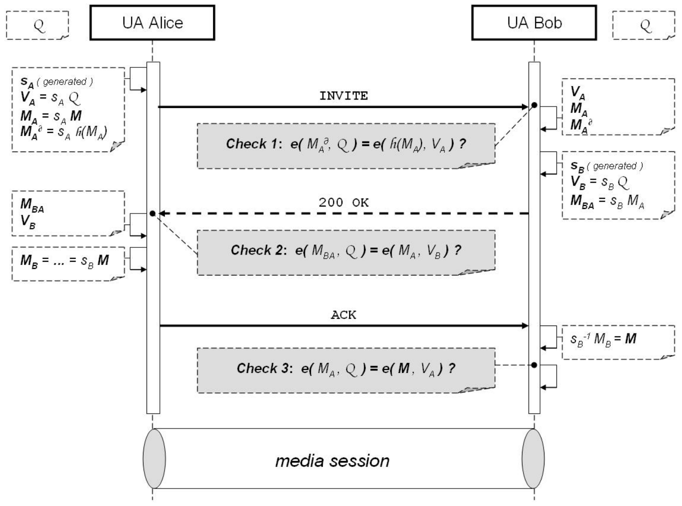

Fig. 9. After performing the final verification without troubles, it is possible to establish the session for secure media traffic

result is enhanced to a different exponent, but can easily be modified to produce exactly the same value. And an advance was presented recently: TinyPBC [29], an open source code which is an implementation of the Eta pairing.

Another proposed pairing was the Ate pairing [39] which is the Eta pairing with the parameters permuted and that also permits other kinds of optimization.

VI. Summary and future works

In this paper we have presented an alternative for the SIP's security mechanisms. Our proposal can provide to SIP real trustworthy security mechanism in all aspects, including the vulnerability to one of the scariest and harmful kinds of attack currently practiced, the man-in-the-middle attack.

One of the benefits of our proposal is that it allows to embed, already in the signalling process, the cryptographic key exchange so that they can be used to ensure the privacy on the media session. Thus, it would not be necessary to use an additional protocol (like MIKEY) to perform the cryptographic key exchange. So it is possible to save a stage (it means less computing and more nimbleness on the user's point of view) and to ensure the security. It is important to note that this saving of one stage is also possible by using the SIP's native security mechanisms, according to the RFC-3261. However, it is not possible to ensure wholly the security. Even if is used the S/MIME foreseen in the SIP's RFC to encrypt the media session key on the own S/MIME envelope, the S/MIME is vulnerable to the man-in-the-middle, as it was pointed in this paper.

Moreover, there are other direct and indirect benefits:

- Supposing that there are risks or security problems which were identified during the typical process for the call establishment, our proposal allows to anticipate and avoid the establishment of the session for the media traffic through a channel which was assumed to be safe. If the risks or security problems appear after the conclusion of the call establishment process, the media session will already be with the privacy insured, once it already happened the safe exchange of the cryptographic keys which will be used in such session.

- Although we focused our proposal in the signalling process for the VoIP's call establishment, the use of pairing-based authentication can be expanded for other SIP transitions. For instance, suppose that a man-in-the-middle attack happens to end an established media session abruptly. The MITM can do this by sending a spurious BYE request at any time. So, the BYE request does not come from any user agents included in the peer-to-peer communication. To prevent this situation an additional check point could be implemented – based on pairing – to check if the BYE request comes from one of the trusty user agents or not. That is possible because both user agents will already have, in this moment of the communication, enough information one from other to enable a very well-aimed verification.

- If no changes happen in the signalling protocol, the only customizations to be made are the verification points and the computations stipulated by the proposal (such as the secret keys generation and the check points based on pairing). Those customizations can be implemented on the software level by a simple, but optimized, manner. Due to this, our proposal is direct and easily applicable in softphones where the customizations and distribution of the application occur easier and faster, if compared to a change in a VoIP telephone project. For instance, consider the ISPs (Internet Service Providers) which offer VoIP services. Those ISPs would not need to create a new and specific softphone based on our proposal. They would only generate a new version for the application (to include the necessary customizations) and they would make this new version of the softphone available on the ISP site. The softphone users could quickly obtain and easily install the new version by the habitual processes of download.

In respect to *future works*, the two last items give, to each one, interesting possibilities. Particularly in the case of the previous item, a first future work could be the implementation of a softphone based on our proposal. And from this implementation, other works could appear as well, such as performance measurements and comparison with the other SIP security mechanisms (existent or even the proposed ones).

Besides this, attack scenarios could be elaborated, simulated and tested, with formal basis, by using appropriate tools, as the Casper/FDR2 ("http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Casper/"). Casper is a compiler which accepts a syntax very similar to the syntaxes that are used to specify protocols.

For the reader who wants to know more about future works involving tests, we recommend a reading of Lee *et al.* [40] to the establishment of a test environment and to perform simulations by using SIP.

References

[1] F. Cao and S. Malik, "Security analysis and solutions for deploying ip telephony in the critical infrastructure," *Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks*, pp. 171–180, 2005.

[2] P. C. K. Hung and M. V. Martin, "Security issues in voip applications," *Canadian Conference on Electrical and Computer Engineering*, pp. 2361–2364, 2006.

[3] D. Butcher, X. Li, and J. Guo, "Security challenge and defense in voip infrastructures," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1152 – 1162, 2007.

[4] V. M. Quinten, R. V. Meent, and A. Pras, "Analysis of techniques for protection against spam over internet telephony," in *Lecture Notes in Computer Science – Dependable and Adaptable Networks and Services*. Springer Berlin / Heidelberg, 2007, vol. 4606, pp. 70–77.

[5] I. Kim and K. Kim, "Secure session management mechanism in voip service," in *Lecture Notes in Computer Science – Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*. Springer Berlin / Heidelberg, 2007, vol. 4743, pp. 96–104.

[6] *Mobile and VoIP to inherit the earth*, EletricNews.Net, 2005, available: http://www.theregister.co.uk/2005/06/27/rising_mobile_voip_revenues/.

[7] *Estatísticas do mercado de VoIP*, Teleco – Informação em Telecomunicações, 2007, available: http://www.teleco.com.br/voip_estatis.asp.

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RFC 3550 – RTP: a transport protocol for real-time applications*, Jul. 2003, available: http://www.faqs.org/rfcs/rfc3550.html.

[9] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, *RFC 3711 – the secure real-time transport protocol (SRTP)*, Mar. 2004, available: http://www.faqs.org/rfcs/rfc3711.html.

[10] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman, *RFC 3830 – MIKEY: multimedia Internet keying*, Aug. 2004, available: http://www.faqs.org/rfcs/rfc3830.html.

[11] M. Euchner, *RFC 4650 – HMAC-authenticated Diffie-Hellman for multimedia Internet keying (MIKEY)*, Sep. 2006, available: http://www.faqs.org/rfcs/rfc4650.html.

[12] A. Barreto and A. Faleiros, *MIKEY DHHMAC-SAS: the new MIKEY transportation mode*, Jun. 2007, (IETF draft 'draft-barreto-ietf-dhhmac-sas-00.txt'. Work in progress.) Available: ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-barreto-ietf-dhhmac-sas-00.txt.

[13] D. R. Kuhn, T. J. Walsh, and S. Fries, *Security considerations for voice over IP systems – recommendations of the National Institute of Standards and Technology (NIST)*, Jan. 2005, available: http://csrc.nist.gov/publications/nistpubs/800-58/SP800-58-final.pdf.

[14] J. Rosemberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *RFC 3261 – SIP: session initiation protocol*, Jun. 2002, (RFC-2543's upgrade). Available: http://www.faqs.org/rfcs/rfc3261.html.

[15] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, *RFC 2543 – SIP: session initiation protocol*, Mar. 1999, available: http://www.faqs.org/rfcs/rfc2543.html.

[16] J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, and T. Haukka, *RFC 3329 – security mechanism agreement for the session initiation protocol (SIP)*, Jan. 2003, available: http://www.faqs.org/rfcs/rfc3329.html.

[17] J. Bilien, E. Eliasson, J. Orrblad, and J.-O. Vatn, "Secure voip: call establishment and media protection," *2nd Workshop on Securing Voice over IP*, 2005.

[18] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman, *Key management extensions for session description protocol (SDP) and real time streaming protocol (RTSP)*, Jul. 2006, available: http://www.faqs.org/rfcs/rfc4567.html".

[19] J. Orrblad, "Alternatives to mikey/srtp to secure voip," Master's thesis, Royal Institute of Technology (KTH), Sweden, Mar. 2005, available: http://www.minisip.org/publications/Thesis_Orrblad_050330.pdf.

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, *RFC 2068 – Hypertext Transfer Protocol – HTTP/1.1*, Jan. 1997, available: http://www.faqs.org/rfcs/rfc2068.html.

[21] A. B. Johnston, *SIP: understanding the session initiation protocol*. Artech House Publishers, 2001.

[22] M. Handley, V. Jacobson, and C. Perkins, *RFC 4566 - SDP: session description protocol*, Jul. 2006, available: http://www.faqs.org/rfcs/rfc4566.html.

[23] R. Rivest and A. Shamir, "How to expose an eavesdropper," *Communications of the ACM*, vol. 27, no. 4, pp. 393–394, 1984.

[24] J. L. Massey and J. K. Omura, "A new multiplicative algorithm over finite fields and its applicability in public key cryptography," in *Advances in Cryptology – Eurocrypt'83*, Udine, Italy, 1983.

[25] ——, *United States patent 4567600: method and apparatus for maintaining the privacy of digital messages conveyed by public transmission*, Omnet Associates, Jan. 1986, available: http://www.freepatentsonline.com/4567600.html.

[26] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. vol. 22, no. num. 06, pp. pages 644–654, 1976.

[27] M. J. Williamson, "Non-secret encryption using a finite field," CESG, UK, Tech. Rep., Jan. 1974, available: http://www.cesg.gov.uk/site/publications/media/secenc.pdf.

[28] A. M. Deusajute and P. S. L. M. Barreto, "Authenticated massey-omura encryption with minimal effort from bilinear pairings," 2007, unpublished.

[29] L. B. Oliveira, M. Scott, J. López, and R. Dahab, "Tinypbc: pairing for authenticated identity-based non-interactive key distribution in sensor networks," Cryptology ePrint Archive, Report 2007/482, Dec. 2007, available: http://eprint.iacr.org/2007/482.

[30] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Criptology*, London, UK: Springer-Verlag, 2001, available: http://eprint.iacr.org/2001/090/.

[31] I. F. Blake, G. Seroussi, , and N. P. Smart, *Elliptic curves in cryptography*. Cambridge University Press, 1999.

[32] I. F. Blake, G. Seroussi, and N. P. Smart, *Advances in elliptic curve cryptography*, 1st ed. Cambridge University Press, 2005, london Mathematical Society Lecture Note Series Collection.

[33] W. Stallings, *Cryptography and network security*, 4th ed. Pearson Prentice-Hall, 2006.

[34] r. D. Eastlake, S. Crocker, and J. Schiller, *RFC 1750 – randomness recommendations for security*, Dec. 1994, available: http://www.faqs.org/rfcs/rfc1750.html.

[35] S. Bradner, *RFC 2119 – key words for use in RFCs to indicate requirement levels*, Mar. 1997, available: http://www.faqs.org/rfcs/rfc2119.html.

[36] F. Zhang and K. Kim, "Signature-masked authentication using the bilinear pairings," Cryptology and Information Security Laboratory (CAIS), Information and Communications University, Tech. Rep., 2002, available: http://caislab.icu.ac.kr/Achievement/technical%20report/data/author-pairing.pdf.

[37] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," ser. Lecture Notes in Computer Science, vol. 2442. Springer-Verlag, 2002, pp. 354–368, advances in Cryptology – CRYPTO 2002.

[38] P. S. L. M. Barreto, S. Galbraith, C. O'hEigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," Cryptology ePrint Archive, Report 2004/375, 2004, available: http://eprint.iacr.org/2004/375.

[39] F. Hess, N. P. Smart, and F. Vercauteren, "The eta pairing revisited," Cryptology ePrint Archive, Report 2006/110, 2006, available: http://eprint.iacr.org/2006/110.

[40] G. Lee, S. T. Kim, I. K. Han, C. Y. Lee, S. H. Park, D. W. Yi, and J. M. Oh, "Security and test environment for sip," in *Lecture Notes in Computer Science – Computational Science and Its Applications – ICCSA 2007*. Springer Berlin / Heidelberg, 2007, vol. 4706, pp. 157–165.