IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 23, 2020, accepted May 4, 2020, date of publication May 11, 2020, date of current version June 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993774

Securing Instant Messages With Hardware-Based Cryptography and Authentication in Browser Extension

GABRIEL ARQUELAU PIMENTA RODRIGUES[1], ROBSON DE OLIVEIRA ALBUQUERQUE[1], GABRIEL DE OLIVEIRA ALVES[1], FÁBIO LÚCIO LOPES DE MENDONÇA[1], WILLIAM FERREIRA GIOZZA[1], RAFAEL TIMÓTEO DE SOUSA, JR.[1], (Senior Member, IEEE), AND ANA LUCILA SANDOVAL OROZCO[1,2]

[1]Decision Technologies Laboratory-LATITUDE, Electrical Engineering Department (ENE), Cybersecurity INCT Unit 6, Technology College, University of Brasília (UnB), Brasilia 70910-900, Brazil
[2]Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Computer Science and Engineering, Universidad Complutense de Madrid (UCM), 28040 Madrid, Spain

Corresponding author: Ana Lucila Sandoval Orozco (asandoval@redes.unb.br)

This work was supported in part by CNPq–Brazilian National Research Council under Grant 312180/2019-5 PQ-2, Grant BRICS2017-591 LargEWiN, and Grant 465741/2014-2 INCT in Cybersecurity, in part by the CAPES–Brazilian Higher Education Personnel Improvement Coordination under Grant 23038.007604/2014-69 FORTE and Grant 88887.144009/2017-00 PROBRAL, in part by FAP-DF–Brazilian Federal District Research Support Foundation under Grant 0193.001366/2016 UIoT and Grant 0193.001365/2016 SSDDC, in part by the Brazilian Ministry of the Economy under Grant 005/2016 DIPLA and Grant 083/2016 ENAP, in part by the Institutional Security Office of the Presidency of Brazil under Grant ABIN 002/2017, in part by the Administrative Council for Economic Defense under Grant CADE 08700.000047/2019-14, in part by the General Attorney of the Union under Grant AGU 697.935/2019, and in part by DPI/DPG/UnB-Decanates of Research and Innovation and Postgraduate Studies of the University of Brasilia.

ABSTRACT Instant Messaging (IM) provides near-real-time communication between users, which has shown to be a valuable tool for internal communication in companies and for general-purpose interaction among people. IM systems and supporting protocols, however, must consider security aspects to guarantee the messages' authenticity, confidentiality, and integrity. In this paper, we present a solution for integrating hardware-based public key cryptography into Converse.js, an open-source IM client for browsers enabled with the Extensible Messaging and Presence Protocol (XMPP). The proposal is developed as a plugin for Converse.js, thus overriding the original functions of the client; and a browser extension that is triggered by the plugin and is responsible for calling the encryption and decryption services for each sent and received message. This integrated artifact allowed the experimental validation of the proposal providing authenticity of IM users with digital certificates and protection of IM messages with hardware-based cryptography. Results also shows the proposed systems is resistent to adversarial attacks against confidentiality and integrity and it is secure when considering cryptrographic tests like the Hamming distance and the NIST SP800-22.

INDEX TERMS Cryptography, authentication, instant messaging security, XMPP, browser extension.

I. INTRODUCTION

Instant Messaging (IM) services may provide advantageous features, such as near-real-time communication, group chats and the support for attaching files to messages. Due to these characteristics, and along with the fact that there are many free IM applications, the number of users of this Internet service has grown in recent years, with a forecast to keep growing up to 2022 [1].

As IM applications are not applicable only to leisure, the aforementioned users are, in fact, distributed in various activity areas, which reinforces the demand for researching such systems. One of these areas of use is the workplace, where the adoption of IM apps has intensified the integration and collaboration among employees, resulting in a raise of productiveness of the company [2]. Near-real-time communication can also be used in teaching and learning,

The associate editor coordinating the review of this manuscript and approving it for publication was Luis Javier Garcia Villalba.

operating as a pedagogical facilitator [3]; in automation, along with Internet of Things solutions such as Arduino [4]; or as a mean of information sharing to a big group of people [5].

However, sending messages in plaintext and not implementing other security measures in the IM system may result in scenarios that threaten properties of information security, for instance with the disclosure of data in transit or with the modification of the stored data [6] as discussed more thoroughly in Section II.

To increase the security levels of a communication system, one possible solution refers to public key cryptography and digital signature, as they promote a way to check the message's authenticity and integrity and enhance its confidentiality. These technologies will be reviewed more in depth in Section IV-B.

A more secure implementation of cryptography, known as end-to-end encryption (E2EE), is achieved if the cryptographic features reside only in the communicating ends, rather than in intermediary nodes, such as carriers, providers and gateways. In an end-to-end encrypted messaging system, neither the server that stores the messages, nor any other device through which the message may pass, is capable of reading the contents of the messages [7].

It is important to note, however, that cryptography and authentication mechanisms do not grant perfect safety, as, besides being breakable technologies, they do not cover all the attack surface area. To reduce this area, a good security measure is to install as few applications as required, decreasing the chances of a vulnerability. Nonetheless, it is possible to evaluate the efficacy of the cryptosystem with the use of statistical test suites for random data, such as NIST SP 800-22, TestU01, ENT, Diehard and Dieharder [8].

Complementary to MIM, it is interesting to consider the case of fixed desktop-based IM. Having an IM application in the computer is more appropriate in some scenarios, such as in the workplace, due to the ease to share documents saved in the computer and to the elimination of the need to use the phone, which could be a distraction. However, an IM application in the desktop may suffer from the same vulnerabilities and, therefore, also requires defense mechanisms and cryptographic solutions.

Thus, in this paper we propose an architecture for the utilization of an Instant Messaging application in desktop that complies with security standards, including the use of E2EE. In our proposal, the IM client runs in a web environment and uses a browser extension to communicate with the cryptographic server, which is hardware-based. The uncertainty of the cryptosystem used is evaluated with the NIST SP 800-22 test suite.

The main contributions of this work are: i) an architecture for integrating IM desktop applications and a hardware device; ii) a review of possible threats and vulnerabilities present in IM systems; and iii) a solution for providing confidentiality, authenticity and integrity to messages exchanged between users.

The rest of this paper is organized as follows. Section II discusses the problems and security threats faced when using an IM system. Section III provides related work, considering other projects concerning security on messaging systems, and how authors are addressing the solution. Section IV details the technologies used, providing a background for the basic concepts necessary to comprehend the architecture; while Section V depicts the proposed architecture, along with the tools used and a discussion on how they are integrated. Section VI shows experimental results regarding the security efficacy of the architecture, along with an adversary model, and tests concerning the randomness of the encrypted data and the performance of the system. Section VII concludes the paper and suggests future works.

II. PROBLEM STATEMENT

To achieve cybersecurity, it is fundamental to comply with its properties, which include confidentiality, integrity, authenticity and availability. Of these, cryptography and digital signature, and consequently the architecture proposed in this project, will not help achieve availability, and, therefore, the discussion on this section will not consider threats to this property.

Regarding the authenticity property, malicious users may impersonate others, and send messages posing as them; eavesdroppers may read the contents of the messages not destined to them, affecting the confidentiality property and the users' privacy; and concerning the integrity, others may tamper the original sent message, changing its content arbitrarily, according to their intentions.

A Man-in-the-Middle (MitM) attack may be used as an example of threat to the above mentioned security properties. In this attack, a malicious user is positioned between two parties communicating to each other, being capable of intercepting and even altering data transferred between them. This attack infringes not only the authenticity property, as the attacker poses as another communicating party, but the confidentiality, as the messages may be intercepted and read by undesired third parties, and the integrity property, since data may be tampered and presented as being original. Dudheria [9] showed that many IM applications are vulnerable to MitM attacks, including popular ones, such as Viber, Facebook Messenger, Snapchat and Skype.

There are various spyware and trojans that can either eavesdrop and modify users' messages, both on smartphones like Android [10], such as Trojan-Spy.SymbOS.Flexispy [11] and others; and on computers, like sniffers [12]. Such malware, running on promiscuous mode, commonly sniffs every packet received and sent by all Network Interface Cards (NICs) of the infected device, being able to read their payloads if they are transmitted in clear text. The proposed architecture is effective to prevent data exfiltration when the data is stolen from the network, but not when it is stolen directly from the users' device, for example with a key logger.

As stated, these applications may be vulnerable to data exfiltration, when a malware, for example, retrieves

unauthorized data from its victim. However, not only a malicious user may undesirably read the app's data: the government may, in some cases, demand access to messages exchanged and users' metadata to the application company, by means of its judiciary system. Such a court decision may be controversial, as in Brazil, whose Supreme Court has received two actions regarding the government rights over the personal data and user privacy, and the imposition of sanctions by public authorities if the access is denied by the application [13]. These actions are still in course.

Although many IM applications offer their own encryption services, they may not be reliable, with weak algorithms or vulnerabilities in the storage; or implement no encryption at all, which allows the inappropriate retrieval of messages in plaintext [14]. Additionally, even if using secure algorithms, the messages must pass through the provider's services, that may be able to read the contents, specially if the encryption uses a key to which providers have access.

Another issue is that for closed source applications, such as WhatsApp, it is not possible to audit the code and verify if E2EE is indeed used. Furthermore, even if it is truly implemented, the provider may still have access to metadata, such as the time, receiver and sender of a message, which may disclose potential private information, like the most frequent contacts a user chats with and the name of the groups he participates. This issue motivates the integration of strong encryption into these applications.

Despite permitting code audit and improvement, open source applications may also have malicious code injected into the original code and, therefore, its is important to not download an application from untrustworthy sources. There are also malicious applications that masquerade as legitimate ones, to fool users into downloading them. BRata (Brazilian Remote Access Tool Android), for instance, is a trojan that pretends to be an update for WhatsApp, but spies and steals data from the infected smartphone. It was available on the official Google Play Store and was downloaded over 10,000 times [15].

It is also critical that IM applications are regularly updated, as new discovered vulnerabilities are commonly fixed and patched in new versions. As an example, CVE-2019-3568 is a buffer overflow vulnerability in WhatsApp VOIP stack that allows remote code execution and is present in the application prior to v2.19.134 for Android [16].

In addition to message encryption, it is also important to require strong authentication mechanisms, as the messages are shown in plaintext in the client, and if anyone may have access to it, the confidentiality would still be violated, regardless of the use of cryptography to send the messages. To emphasize this concern, we mention the case in which Brazil's Justice Minister had his smartphone invaded, resulting in several Telegram messages leaked and exchanged in his behalf [17]. If an authentication mechanism were used in the IM application, the invader could still have access to other features in the smartphone, but not to the messages exchanged in the app. Anyway, there are diverse forensic approaches to the acquisition and analysis of digital evidence in smartphones, including from IM applications [18], [19].

This solution is not effective against spamming nor against social engineering attacks over IM apps, such as phishing. To prevent from these threats, users must be advised not to click on unknown links, download files from untrustworthy sources nor disclose personal information. As an example, this solution does not protect from the Bad Rabbit ransomware, a malware that spreads from drive-by downloads, disguising as an installation of an update for Adobe Flash Player [20].

III. RELATED WORK

Some researches have studied and compared the characteristics and security features of different MIM applications. As an example, Sutikno *et al.* [21] concluded that Telegram is, among the apps compared, the most secure, as its secret chat option provides all the security features used as parameters. The authors also demonstrated that the use of encryption in the application does not necessarily mean the messages exchanged are secure against eavesdroppers, as Viber's provider, despite using cryptography, have access to the key and, therefore, may read the contents of the conversation.

Moreover, WhatsApp messaging encryption is based on the Signal Protocol, and studies demonstrate that it is implemented adequately and the provider does not have access to the keys nor to the plaintext of the messages [22]. Nonetheless, a zero day vulnerability, related to a buffer overflow that allows the execution of arbitrary code, has been recently found on the application, which allows attackers to install spyware, turn on the microphone and read encrypted messages, with the attack vector being a phone call on the app [23].

However, it is not always the case that security breaches are due to technical problems, but may also be due to users' lack of security awareness. Schröder *et al.* [24] presented a study on the Signal IM application and its end-to-end encryption, and showed that the majority of the tested users were not able to identify a MitM attack taking place on the application, even with alerts popping up. The users selected for the research were computer science students and, even with the expected security background, had their privacy exposed. The authors suggest some improvements to Signal's usability in order to increase security levels.

Others have acknowledged the importance of cryptography in communication. According to Kim and Yoon [25], some cryptosystems used in IM are vulnerable to brute-force attacks, specially those with short keys, or that are based in passwords, as users' choices are commonly weak. To overcome this problem, the authors propose a Honey Encryption system for use in Instant Messaging, in which inputting the wrong key will give a plausible, yet incorrect, plaintext. This hinders the identification of bad keys, slowing down the brute force attack. However, although it could, this project does

TABLE 1. A comparison between related works and this project.

Related work	Description	Comparison to this work
[21], [22], [23], [24]	The authors examine and debate threats and vulnerabilities in IM systems, and also compare different applications	This work does not focus on a vulnerability, rather it proposes a solution for some of them
[25]	The research proposes the use of a honey encryption to countermeasure the brute-force vulnerability to guess the key	Instead of a honey encryption, this project implemented a hardware based cryptography to protect the messages
[26]	An implementation of a hardware-based cryptography for protecting IM messages	This work proposes a different architecture for integrating the IM software and the hardware, with a browser extension
[28]	The authors propose a Mozilla plugin for software encryption and steganography, not focused on the IM context	This research does not use steganography, as it would not be viable for IM messages. It also uses hardware cryptography

not use Honey Encryption for encrypting the IM messages exchanged between users.

Bao and Xu [26] have also studied and proposed a scheme in which IM messages are secured with hardware encryption and key generation. However, they do not use an architecture similar as the one proposed by this work. The authors also use, like this project, a combination of symmetric and public key cryptography for encrypting the chat content.

There are many security related browser extensions and documentation for object encryption available for XMPP (e.g. RFC 3923 [27]), but, to the best of this research knowledge, we could not find any suitable reference that combine the signing and encryption of XMPP messages using hardware-based solutions. As an example, Binu *et al.* [28] developed a Mozilla plugin that protects messages in two layers: it generates a file with an embedded message, through steganography, that is previously encrypted. For that reason, an eavesdropper must not only find the hidden message, but also decode the unintelligible ciphertext. However, this solution, unlike the solution proposed in this work, is not applicable in the IM scenario, as it would be unpractical to generate a file for every message desired to be transmitted.

This research differs from some of the cited in this section in that it does not study vulnerabilities present in IM applications. Instead, it proposes a solution that could mitigate some of the discussed breaches. Table 1 summarizes the different characteristics between this and the related works.

IV. TECHNOLOGIES REVIEW

The architecture used in this work comprehends different technologies, which are discussed in this Section.

A. INSTANT MESSAGING

For users to be able to exchange near-real-time messages, an IM solution is required, which comprehends a protocol, a client and a server. The technologies used for each of these are briefly reviewed.

1) THE PROTOCOL

As in the proposed architecture the IM client runs in a web browser, it is natural to conclude that the HTTP will be the application layer protocol used. However, due to its polling nature, it cannot be used in a messaging system, as a long polling interval would cause the message exchange not to be instant, whilst a short interval would result in many status 304 (Not Modified) responses. Also, the HTTP synchronicity would require the client to wait for the server response before making new requests, which is not desired in an application that must process many incoming and outgoing messages nearly simultaneously. For that reason, a better approach is to use an asynchronous, event-driven, publish-subscribe pattern, provided by the Extensible Messaging and Presence Protocol (XMPP).

XMPP, formerly called Jabber, is an open protocol for real-time messages exchange. It uses the Extensible Markup Language (XML) to format the data to be transmitted, along with its metadata, such as timestamp, source and destination entities. It supports, besides basic IM features like one-to-one messaging and multi-user chats (MUC), additional features to improve the usability, such as contacts list (or roster) and their availability (or presence), notification, blocking users and vCards.

From RFC 6120 [29], the basic unit of meaning in XMPP is a *stanza*, which are exchanged between two entities in the XML stream to configure the communication, request data from the other party and send the actual messages. Stanza is the first level element of the XML, and is named according to its purpose. `<presence/>` stanzas are used to listen to the availability status of an other entity, while `<iq/>` (*Info/Query*) stanzas are used to request general information to an other entity, in a request-response structure. Finally, `<messages/>` stanzas are used to transmit the actual messages.

Due to its extensible nature, inherited from XML, XMPP is not limited to IM. It is also used, for instance, in middleware, data syndication, web services, games and social networks [30]. This feature is achieved with the possibility of adding as many child XML elements to the message as necessary and with the XMPP Extension Protocol (XEP) series. In this work, the protocol is used for IM, but still some XEPs are used and some child elements are created, exploring the extensible characteristic of XMPP.

One XEP strictly related to this work is XEP-0124, which documents the use of Bidirectional-streams Over Synchronous HTTP (BOSH). BOSH aids the development of XMPP applications inside web browsers by

emulating the semantics of a long-lived, bidirectional TCP connection between two entities, such as a client and a server. It uses long-polling, with multiple synchronous HTTP request/response pairs, without requiring the use of frequent polling or chunked responses. The use of XMPP over BOSH is defined in XEP-0206.

However, according to the RFC 7395 [31], this long polling used by BOSH results in a higher transport overhead and other issues. RFC 6202 [32] also cites a maximal latency as a BOSH's long polling issue. As browsers do not support the direct use of XMPP and as BOSH suffers from overhead and latency, the WebSocket protocol, defined in RFC 7395, comes as a solution. It is a bidirectional protocol that emulates TCP and provides a simple message-based framing layer. In this project, XMPP over WebSocket is used.

2) THE CLIENT

Converse.js is an open source XMPP client designed to run in any browser, written in JavaScript and uses the XMPP library Strophe.js for its core functionality. It has been chosen for this work as it is a popular solution and, at the time of writing, has been translated into 28 languages and implemented several useful XEPs, such as multi-user chatrooms (XEP-0045), direct invitations to chat rooms (XEP-0249), service discovery (XEP-0030), file sharing / HTTP file upload (XEP-0363), message carbons (XEP-0280) and server-side archiving of messages (XEP-0313).

The client is displayed in the browser as overlay chat boxes, as a complement to the website. In this project, however, it is preferred to exhibit the chat boxes in full screen, which is achieved with the inverse.chat version.

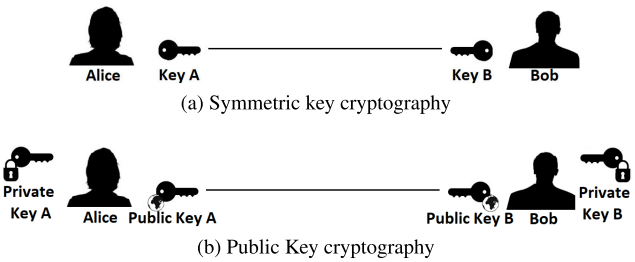
As an open source solution, Converse.js allows modifications in its source code, for improvement of functionality. The only recommended way for doing so is through plugins [33], which can be used to override and create new functions. In this research, different plugins are created for specific purposes towards the objective of securing conversation, each overriding and creating new functions and events related to entering a new chat box (the authenticity of the user should be verified prior to the possibility of interacting with the chat), sending messages (which should be encrypted prior to the transmission) and receiving messages (which should be decrypted prior to the presentation in the chat box).

3) THE SERVER

Openfire is a XMPP server written in Java which, according to its official page [34], is licensed under the Open Source Apache License. We found Openfire suitable for this project because it is a cross-platform, easy to setup and administer solution [35].

To store all data, including archived messages (XEP-0313), Openfire uses a back-end database, which can either be embedded or external. In this project, an external PostgreSQL databse is used.

Openfire also supports plugins to extend the functionality of the server. The official website provides a list of available plugins, but new ones can be developed according to the needs.



(a) Symmetric key cryptography

(b) Public Key cryptography

FIGURE 1. A comparison of symmetric and public key cryptography schemes.

B. CRYPTOGRAPHY SERVICE

For securing the communication, the use of a cryptography is necessary. In this project, a hardware-based cryptography is used.

1) CRYPTOGRAPHY

Data encryption is the main idea of this project, as it reduces the surface of vulnerability, discussed in Section II, and can be accomplished with two different approaches, namely symmetrical or public key.

In the symmetrical encryption (Figure 1a), both sender and receiver must share the same key, which must be known only to those allowed in the communication. As the same key can be used by more than two parties, it is not possible to determine the sender of the message based solely on this technology. It is also not possible to check the integrity of the received message and, as the number of users in the conversation grows, the more difficult it is to share and manage the key.

If a public key cryptography algorithm is used (Figure 1b), all parties in the communication have, each, one pair of keys, one of which is known to everyone, named public key; and the other, named private key, is private to its holder. As the keys are mathematically related, if one is used to encrypt the message, the other can be used to decrypt it. Thereby, to send a confidential message to an other party, one must encrypt the message with the receiver's public key, so that only his private key will decrypt it. However, if a message is encrypted with the private key, anyone else can use the related public key to decrypt it and verify the authenticity of the sender and the integrity of the message. The encryption using both keys can be combined to achieve confidentiality, authenticity and integrity.

Figure 1 illustrates the difference between both models of cryptography. Whereas in Figure 1a Key A and Key B are identical, in Figure 1b, Alice and Bob have each a different pair of keys. Thereby, if Alice, from Figure 1b, wants to send a confidential message to Bob, she will encrypt it with Bob's public key. After that, not even Alice, the author and sender of the message, will be able to decrypt the message back to its plaintext, but only Bob with his private key.

Although not necessarily public key encryption algorithms are more secure, they provide more properties and the keys are easier to manage, despite being slower. Because of that, it is the model adopted in this work.

Both symmetric and public key cryptography also provide confidentiality, as they make the contents of the messages unintelligible. Thereby, eavesdroppers are kept from reading the users' conversation.

It is important to state that no cryptography algorithm is completely secure [36], regardless of whether symmetric or public key is adopted, apart from One-Time Pad [37]. Although there will always be the chance of being broken, the cryptography system is said to be computationally secure if either the cost of breaking it exceeds the value of the encrypted information or the time required to break it exceeds the useful lifetime of the information [38].

2) HARDWARE-BASED CRYPTOGRAPHY

Software cryptography may suffice for many applications, as, if properly implemented, it provides a good level of security. However, some applications may require a greater level of secrecy and privacy, cases in which hardware based cryptography may be more adequate.

Many cryptography algorithms use random numbers for generating a key, a Initialization Vector (IV) or a nonce. The randomness of these numbers are critical for obtaining semantic security, which is said to be achieved if an adversary, after sending two plaintexts to a cryptosystem and receiving one ciphertext back, cannot guess with a probability higher than 0.5 which plaintext the ciphertext refers to [39].

However, as a software is essentially deterministic, it can only generate pseudo random numbers based, for example, on random walks [40]. A hardware device, however, is capable of generating true random numbers directly from a physical process, avoiding bias, and presenting bit independence, unpredictability and non-repeatability [41]. These factors increase the security of the parameters generated and used in the cryptography.

To assess such secrecy enhancements, the hardware used for encrypting the IM messages in this project is tested, in Section VI-D, in regard to the randomness of the ciphertext that it generates, and with the evaluation of the Hamming distances between the input and the output of the hardware.

It is expected that the hardware-based cryptography system used in this project achieves the properties cited by [41], that is, it is expected that the encrypted IM messages are not predictable to eavesdroppers, and are absent of bias, bit dependence and repeatability.

Another security improvement in hardware cryptography refers to the fact that any software may be vulnerable. As a hardware may be isolated from the software, it can be less vulnerable to malicious invasion, human error and malware contamination. In the context of this project, having a hardware cryptography also allows for keeping the cryptographic algorithm out of reach of other extensions or flaws in the browser.

To complement the hardware-based cryptography, a Two Factor Authentication (2FA) token may be used. Despite the availability of 2FA tokens in the market, such as Yubikey [42], a self-developed token, with support to PKCS #11, will be integrated into this IM architecture in future works.

3) DIGITAL SIGNATURE AND CERTIFICATES

The digital signature process, enabled only when using public key cryptography, consists of encrypting, with the sender's private key, the entire sent message or its hash code [38]. In both cases, not only authentication and non-repudiation is granted, but also integrity, as any change in the original message along the transmission channel will result in different encrypted values, causing the signatures not to match, thus evidencing the alteration, either intentional or due to noise in the channel. Although the two modes of digital signature present the same security attributes, encrypting the hash code makes the signature generation and verification processes more efficient, as it shortens long messages.

Thereby, Alice (Figure 1b), needing to send a signed message to Bob, will encrypt it using her private key and transmit the resulting ciphertext, along with the original message, to him. Upon receiving, Bob will decrypt the signature and compare it to the plaintext of the message. If they match, the message has really been sent by Alice and not been altered along the insecure channel; otherwise, it is either unauthentic or corrupted.

Digital signing allows the authentication of a message, as it is expected that only the owner of the private key knows it, which hinders impersonations and protect users from third parties. Furthermore, they also protect the users from one another, as it can be proofed, also based on the exclusive knowledge of the key, that the alleged sender in fact sent the message [38]. Hence one can not deny sending a signed message, which is known as non-repudiation, securing the other party against false denial of involvement in the communication.

To associate a public key to its holder, preventing mistakes and impersonators, a digital certificate, that proves the ownership of a public key, can be used. This electronic document is issued by a Certificate Authority (CA) and contains information fields such as its period of validity and the information of the subject's public key [38].

To improve security, each certificate is valid only through a period of time, after which the document is revoked. It is usually also a CA's responsibility to issue Certificate Revocation Lists (CRLs), that register the certificates either expired or compromised, and that should no longer be used.

C. INSTANT MESSAGING AND CRYPTOGRAPHY SERVICE INTERCOMMUNICATION

In order to encrypt and decrypt the massages exchanged between the IM clients, running in each user's browser, it is necessary to establish a channel of communication between Converse.js and the hardware cryptography service. This is

accomplished with the use of a web extension (or add-on or plug-in).

1) BROWSER EXTENSION

A browser extension is a software to change and add functionalities to a web browser, customizing it. Using web-based technologies (i.e. HTML, CSS and JavaScript), it uses the same web APIs as the web page in which it is running, but also having access to its own APIs, granting more functionalities to the developer [43].

However, as a web extension can manipulate, add and delete the page's contents and access data like the browser history, it can also be used maliciously, like the Stolen Pencil plug-in, that targeted academic institutions [44]. Because of that, users should always be security aware before installing an extension and verify its permissions, which should follow the principle of least privilege.

The permissions of a browser add-on can be found in its manifest file, that also defines, for example, the software's name, version, description and scripts, along with the URLs it is allowed to run in [45]. The scripts, used to add the core functionality of the extension, can be classified in either content scripts or background scripts.

The content scripts are the ones running in the context of the web pages. Using the standard Document Object Model (DOM), they can read and modify contents of the web page, while isolated from it, in order to avoid conflicts. However, they have a restricted access to the web extension API, necessary.

The background scripts are the event handlers for the extension and do not have access to the web pages' contents, but have a less restricted access to the web extension API. Running one persistent instance per extension, this script must remain inactive until triggered, when it performs its designed operation, and after which it must goes inactive again [46].

In this project, both types of scripts will be used, as content scripts may interact with the web page, reading the messages exchanged in the Converse.js client; and background scripts have access to the API necessary to communicate with programs outside of the context of the browser, i.e., Native Messaging Host (NMH). The messaging API permits the communication between content and background scripts.

2) NATIVE MESSAGING HOST

Although a web extension can have access to the messages sent and received in a IM client running in the browser, it can not directly communicate with the hardware cryptography server, as it is limited to the page's DOM and, therefore, another software is needed.

Native Messaging enables a web extension to communicate, through its background script, to a native application installed on the user's computer, not managed by the browser [47]. The information flows between extension and native application in the JSON format, UTF-8 encoded, using standard input (`stdin`) and standard output (`stdout`) [47].

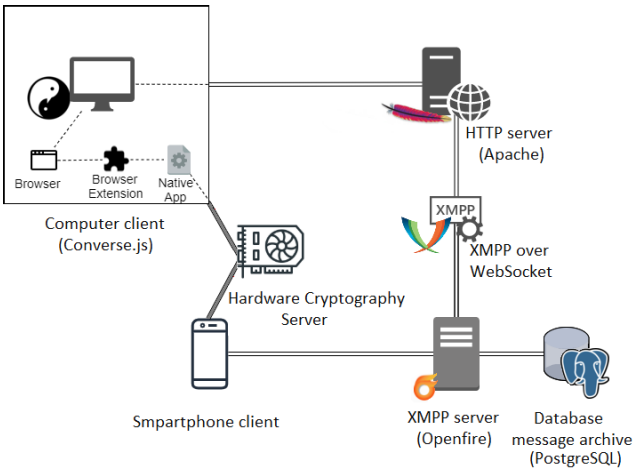


FIGURE 2. Instant messaging architecture used.

In this work, the native application is a program that sends and receives, through the network, the plaintext and the ciphertext from the cryptography server.

V. PROPOSED ARCHITECTURE

This section depicts the architecture used in the experiments, first in the regards of the tools and technologies it comprehends, and how they are arranged; and then in respect to their intercommunication, and how the messages flow between them.

A. GENERAL ARCHITECTURE

The technologies described in Section IV are integrated as shown in Figure 2. A smartphone client is illustrated in the picture just to indicate it could be integrated and used in this architecture, however, as it is not the focus of this work, its functioning is not detailed.

An Apache web server is used to host the Converse.js webpage, to allow the users access to the client from the browser. To support the communication from a web based XMPP client to Openfire, XMPP over WebSocket is used. The clients communicate to Openfire, a XMPP server, that handles the messages and stores them in a PostgreSQL database.

A thorougher explanation of the architecture inside the computer client and how its elements intercommunicate with the hardware cryptography server is provided in Section V-B.

The first step to use the XMPP client is to authenticate as a user. As a password based authentication is vulnerable to brute force and dictionary attacks, a more secure approach is to use Simple Authentication and Security Layer (SASL), specified in RFC 4422 [48]. To achieve this, Converse.js must be configured to use an external authentication mechanism.

SASL is a framework that provides, besides authentication, data integrity and confidentiality. It supports different authentication mechanisms, and the shared secret mechanism is the one implemented in this project. In this mechanism, the server sends a challenge to the party trying to authenticate, which should send back a response, proving that it knows the shared secret. This is much more secure than sending the secret over

FIGURE 3. Converse.js main screen for the user Alice, showing the roster, unread messages notification and contacts presence.

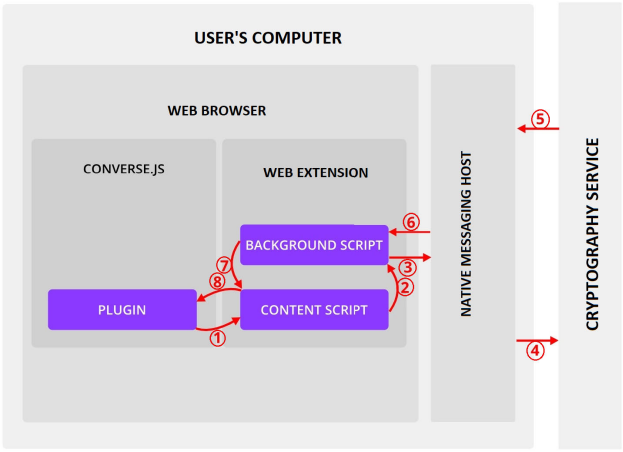


FIGURE 4. Message flow for sending requests and receiving responses from the hardware cryptography service.

the wire, as it could be intercepted. The use of SASL in XMPP is defined in XEP-0034 [49].

After authenticated, the user has access to the Converse.js main page, which, in its full screen version, displays to the left a list of contacts and group chats of which the user is a participant. If any of these contacts or groups is clicked, a chat box opens in the panel to the right of the list. The described screen is shown in Figure 3.

For a user to be able to read (i.e. decrypt) and to send (i.e. encrypt) messages, however, it is first necessary that the extension knows the user's certificates. Therefore, each time a user clicks on a contact name, triggering the event `chatBoxOpened`, before opening the correspondent chat box, the client should check if both the sender's and the receiver's certificates are already known. If they are not, they must be downloaded and stored, and also checked if they are still valid, consulting a CRL. The plugin overrides `chatBoxOpened` event to add this behaviour, and if, for any reason, a valid certificate can not be downloaded, the chat box is not displayed and an error message explains what happened to the user.

The behaviour is analogous to respond to a user click on a group name. The `chatRoomOpened` event is overrode with the plugin, in order to check and download the certificates of every participant of the correspondent group, before actually opening the chat box.

Additionally, as an user may receive a message from another user whose chat box he has never opened before, the plugin also overrides the `onMessage` event, which is triggered every time a new message is received, to check the existence and validity of the certificates.

After all necessary and valid certificates are downloaded and present on the client, the user may send encrypted messages and read decrypted received messages. To accomplish this secure messages exchange, some original Converse.js functions must be overridden.

B. ARCHITECTURE IN THE CLIENT

To encrypt and decrypt any message, a client needs to communicate with hardware cryptography server. This section explains how this communication occur, along with the Converse.js functions that needed to be overridden. Figure 4 depicts a diagram for the message flows between the computer client and the the hardware cryptography server.

The `createMessageStanza` function builds the XML stanza and its attributes, such as the type of stanza and the Jabber ID of the destination and the sender. The plugin overrides this function, in order to change the body attribute, which contains the actual message, to the ciphertext provided by the cryptography service. To get the ciphertext, the plugin communicates asynchronously and directly to the content script of the web extension, with a `promise` that triggers an event listened by the content script, process represented as ① in Figure 4. The creation of the stanza, in the overrode function, only finishes when the promise is resolved ⑧, with the ciphertext set as the body of the message and with the addition of new child elements, such as the used cryptography version and information of the public keys and certificates used. After that, Converse.js follows its normal operation for sending messages.

The payload sent by the plugin to the content script contains a message identifier, the Jabber ID of the destination parties, the plaintext and the cryptography version. This information is used in processes from ① to ⑧, along with the operation type (which can be, in this context, either encrypt or decrypt, but it generally include add and check user certificates and others) and the request identifier, which are added by the content script in the message sent to the background script through messaging API ②. A callback is configured to obtain the response from the background script when there is one ⑦, which can then be passed to the plugin. When writing the code for this messaging between scripts, security aspects were considered, regarding the use of dangerous APIs that could allow cross-site scripting (XSS) [50].

The background script, upon receive of the contents script's request, connects and sends a message to the native application, through the NMH API ③, which consumes the

cryptography API to request the operation to the cryptography server ④. When the operation is done, the server returns the result to the native application ⑤, which will forward the message to the background script ⑥ and follow the flow until it reaches the Converse.js plugin, after which the client continuous as originally designed.

For message decryption, the plugin overrides the `onMessage` functions of both of `ChatBoxes`, responsible for handling messages received from a single user chat, and `ChatRoom` objects, responsible for handling messages received from a group chat. These functions are overrode to decrypt the message before displaying it in the chat box, following the same flow depicted in Figure 4 and detailed in previous paragraphs.

That way, an E2EE is achieved, as only the communication parties possess their respective private key, and a server or database administrator, for example, would not be able to decipher it.

Another consequence is that messages are transmitted and stored in cyphertext, and third parties will not be able to eavesdrop them in the wire nor modify them in the database.

As it is a prototype, the authors' first concern was to implement the encryption and decryption for text messages, as they are the fundamental feature for an IM application. However, other features, such as attaching files, enrich the users' experience, are viable in the proposed architecture, and will be included in future versions of this work.

Also, as the focus of this work is the implementation of the higher level architecture, the authors assumed, as a premise, that the hardware encryption server was capable of handling multiple requisitions simultaneously, and did not consider any aspect regarding the lower level implementation of the hardware device.

VI. EXPERIMENTAL RESULTS

In this Section, we inspect the data transmitted by the XMPP client and the server, before and after the implementation of the cryptographic architecture.

First, we demonstrate how an eavesdropper may compromise the confidentiality of the communicating parties, and, then, the better level of privacy achieved with data encryption. The end of the section provides a brief evaluation of the performance of the proposed architecture, estimating the additional latency caused by the encryption and decryption of sent and received messages.

The experiments are conducted between two parties, namely Alice and Bob, in the XMPP domain `unb.br`.

A. BEFORE ENCRYPTION ENFORCEMENT

Prior to the implementation of the proposed system, the stanzas are transmitted in the wire in plaintext, as shown in Listing 1. This is not a secure communication, as an eavesdropper can easily identify, from the `body` tag of the stanza, that Alice is trying to arrange a meeting with Bob at a given location and time.

```
<?xml version="1.0"?>
<body xmlns="http://jabber.org/protocol/
    httpbind"
      rid="3944335484"
      sid="2511aabc3446a0de9bb1a098">
  <message xmlns="jabber:client"
           from="alice@unb.br"
           id="e4cad784-bca0-4600-b87b-
               b7656a73925f"
           to="bob@unb.br"
           type="chat">
    <body>Meet me at 15&#xB0;45'47.5"S 47&#
        xB0;52'18.6"W at 5:15 am tomorrow</
        body>
    <active xmlns="http://jabber.org/
        protocol/chatstates"/>
    <request xmlns="urn:xmpp:receipts"/>
    <origin_id xmlns="urn:xmpp:sid:0"
               id="e4cad784-bca0-4600-b87b-
                   b7656a73925f"/>
  </message>
</body>
```

Listing 1. XML of the plaintext stanza transmitted.

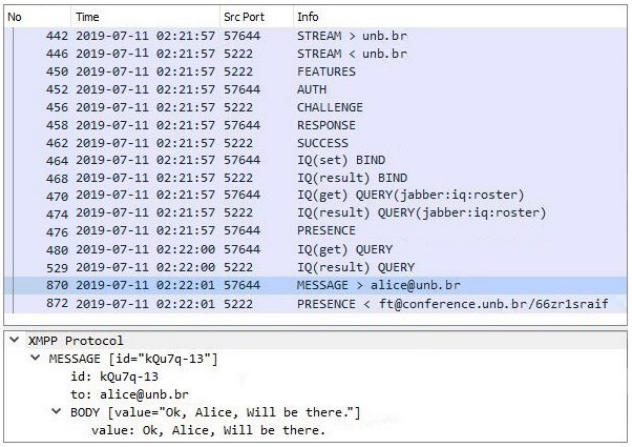


FIGURE 5. Wireshark used to eavesdrop Bob's message to Alice.

The `&#xB0;` value in the message of Listing 1 is the hexadecimal Unicode representation for the char '°', meaning the latitude and longitude of the location proposed by Alice for the meeting.

Bob's response may also be read, in plaintext, with the use of a packet sniffer, posing as an eavesdropper, as demonstrated in Figure 5, in which Wireshark is used to disclose that Bob has accepted Alice's invitation. Thereby, third unwanted parties are capable of infringing the conversation privacy, specially if the sniffer is set on promiscuous mode, which enables the reading of packets addressed to other machines.

Filtering the capture, in Figure 5, to display XMPP packets only, the malicious traffic interception shows all steps taken for establishing the connection between client and server, in which the packets originated in port 57644 are sent by the client, while the source port 5222 (standard port for XMPP client connections) refers to packets transmitted by the server.

First, the stream opening (packets number 442 and 446) is exchanged between client and server, then the server's announcement of the features it supports (packet 450), followed by the client authentication (from packet 452 up to 462). These steps are more deeply explained by Saint-Andre, Smith and Tronçon [30].

After authenticated, Bob may be able to send IQs and get their responses (packets 464, 468, 470, 474, 480 and 529) and send his presence and get other's (packets 476 and 872). Bob can also send messages to other users, such as in packet number 870, in which he sends a response to Alice. The body of this response is shown in the bottom part of the Figure 5.

More than reading the contents of exchanged messages, an eavesdropper may also infer, from the Figure 5, that Bob is a member of a MUC named `ft`, as evidenced by the `conference` subdomain of packet 872. However, the proposed architecture will not prevent such discovery from eavesdroppers, as only the message data will be encrypted.

B. AFTER ENCRYPTION ENFORCEMENT

With the purpose of securing the XMPP messages, the architecture described in Section V is implemented. Thereafter, the `body` tag of message stanzas are encrypted prior to the transmission over the network, and decrypted before being displayed on the screen to the receiver user.

Listing 2 shows the XML stanza for the same message transmitted in Listing 1, that is, `Meet me at 15°45′ 47.5″S 47°52′ 18.6″W at 5:15 am tomorrow`, but now protected from eavesdroppers with encryption, after the implementation of the proposed architecture. New XML child elements and attributes are also added.

The `body` tag is now unintelligible, and only someone possessing the recipient's private key will be able to decipher it. Other tags are also added to the stanza, such as `pubKey`, which gives the information regarding the public key of the sender, present in its certificate; and also the `signature`, which is the message encrypted with the sender private key, assuring its authenticity and integrity, verifiable with the use of the correspondent public key.

The legitimate user's client, upon message receive, will decrypt the text using the private key and the clear text will appear in the chatbox, allowing the user to read it. Any message sent by a user is also shown in clear text on its own chatbox. The encrypted message, transmitted as in Listing 2, is decrypted and shown to the users in the conversation, as depicted in Figure 6.

C. ADVERSARY MODEL

This adversary model considers that an unauthorized third party, named Eve, intends to violate the communication between Alice and Bob in different manners. In this adversary model, it is assumed that: a) Eve has access only to the public keys of Alice and Bob; b) Eve has or can intercept the transmitted ciphertext; c) Eve has a large computational power; d) Eve does not have physical access to the computers of Alice and Bob, to capture the deciphered data in memory.

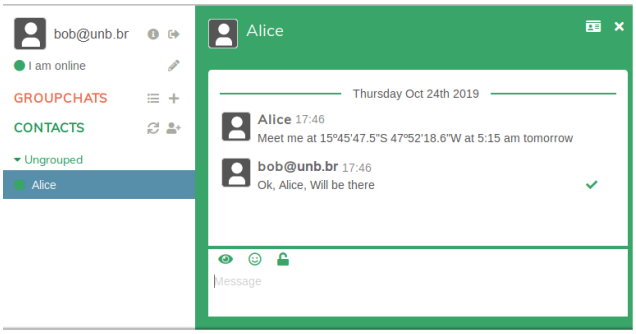


FIGURE 6. Chatbox displaying clear text messages exchanged between Alice and Bob.

```
<?xml version="1.0"?>
<message from="alice@unb.br/converse.js
    -106733886" to="bob@unb.br" type="chat"
    id="48bf8e76-3a9b-4c35-a6fd-9bd4a2b9fd29"
    >
  <body>
      fASPD2wpA4aahVKf1Ia7Yei51Heh82jEEmEmwB
  HFnzLAPqkX0Xq3YGS5mjCPP4IOlhpkaALISuwoKnHn
  j+BvMD4SpaG8GniMbPDduDHd8=</body>
  <active/>
  <request/>
  <delay stamp="2019-10-24T17:46:57-03:00"/>
  <keys ver="1000">
    <pubKey>
        MHkwEwYHKoZIzj0CAQYIK8MAAgIEAQEDYgAE
    JqugjJEvHYxw9JE5waFuMWx1yLpGUPg3817292YP
    BLcV9nPuO0nlRDNCXQreMXXDB5XTOSvqZrtuxJHb
    u4c4z9RVzPYkh6WFq5h6suY5+6g1djg2S9hdV8SS
    2lhC</pubKey>
    <key certid="817e846d63a30883">
        clxSXhu9utSY8oC+RVqrRdP8d4m4AhMy0ecs
    7bq78PU=</key>
    <key certid="e838122de2919d6c">7
        pixyDv7Agbzybp2OSFun0nXhY8nSfCWHVDal
    3Ymbu8=</key>
  </keys>
  <signature>
      MGYCMQDUuS7zjkwqErhQvZR7lZoz2Trl1hJFbV
  Pex4V3MGRUs89pyrkikzxLEprXncGi1rQCMQDLjoX1
  8mOMG5PbhkunK2IofM7SVw04ZozizLOZCPW82FQ7oF
  r1WEZrX4T4U1BL48=</signature>
</message>
```

Listing 2. XML of the encrypted stanza transmitted.

Nevertheless, in possession of a valid private key or with physical access to the hardware cryptosystem, Eve would be able to impersonate, modify and eavesdrop messages, violating the confidentiality, integrity and authenticity properties. With a private key, Eve would be able to conduct, for example, a MitM attack.

Considering a scenario with the cited conditions a) to d), Eve, in an attempt to eavesdrop the communication and affect its confidentiality, would have to derive the receiver private key from its public key to be able to decipher the message. It should be remarked that the process of deriving a private key is not trivial [51] and, considering that the parameters

of the cryptosystem were adequately chosen, the task could require more time to be completed than the time of validity of the protected message, or it would be more expensive to break the system than the value of the encrypted information. In that scenario, it would not be viable for Eve to accomplish a successful attack to break the cryptographic system that protects the message.

It is relevant to mention that, to the best of the authors' knowledge, the largest factored integer for the RSA algorithm is RSA-250, using a 829 bits key [52]. Nowadays, the minimum length recommended for a RSA key is 4096 bits, and such assumption is used in the proposed cryptographic system in this work.

It is important to remember that the use of quantum computing significantly reduces the time necessary to break a cryptosystem [53]. However, this technology is not yet available to ordinary use and its adoption is not considered for Eve in this case.

Furthermore, it is interesting to consider a scenarion in which Eve knows that Bob could only respond to Alice's invitation in two manners: either accepting it, sending the message `Accept` to Alice; or declining it, with the message `Dismiss`, and no other message is allowed. Assuming the cryptosystem is semantically secure, that is, no information from the plaintext is viable to be extracted from the ciphertext, Eve would not be capable of identifying to which of the two possible plaintexts the ciphertext would decipher.

In this scenario, even if Eve had access to the cryptosystem as a black box, with the ability to encrypt any message of her desire, which is known to be the chosen-plaintext attack, she would not be able to deduce Bob's response by encrypting either `Accept` nor `Dismiss` and comparing the generated ciphertexts. The reason for this is that the used cryptosystem is non-deterministic, that is, different ciphertexts are generated for different encryption processes of the same plaintext.

If Eve, instead of eavesdropping, intended to modify a message or impersonate a communicating party, the integrity property would be infringed. But to succeed in this attack, she would still have to have access to a party's private key to be able to sign the message as the legitimate user, incurring in the same key deriving problem discussed previously.

Another attack possibility for Eve is the capacity to intercept and block the message to reach Alice or Bob. For instance, blocking Alice's invite for Bob, who would not receive the message. In this event, neither the confidentiality nor the integrity would be affected, but the availability property. In this regard, the proposed system presented herein would not be able to prevent the attack because the system does not control all the possible communications channels available for Alice and Bob to communicate.

D. ARCHITECTURE EVALUATIONS

In this section, the proposed architecture is evaluated, first with a comparison between the ciphertext and its correspondent plaintext. Then the avalanche effect for the cryptosystem is assessed and, in the sequence, a battery of tests is conducted regarding the predictability and the uncertainty of the generated ciphertext. The final evaluation regards the performance of the architecture.

```
def hamming_distance(ciphertext, plaintext):
    assert len(ciphertext) == len(plaintext)
    return sum(ch1 != ch2 for ch1, ch2 in
        zip(ciphertext, plaintext))
```

Listing 3. Python function used to calculate the Hamming distance.

1) CIPHERTEXT AND PLAINTEXT COMPARISON

To compare a ciphertext to its correspondent plaintext and evaluate their similarity, the Hamming distance between them is measured. In general terms, the Hamming distance indicates the number of characters (in a string) or of bits (in a bitstream) at which two strings or bitstreams differ at the same position. For calculating the distance, it is fundamental that both strings or bitstreams have the same length.

The Python function shown in Listing 3 is used to calculate the Hamming distance between `ciphertext` and `plaintext`.

To calculate the Hamming distance between the plaintext and the cyphertext we assume:

Plaintext: `Meet me at 15°45′ 47.5″S 47°52′ 18.6″W at 5:15 am tomorrow.`

And the corresponding ciphertext: `fASPD2wpA4aahVK f1Ia7Yei51Heh82jEEmEmwBHFnzLAPqkX0Xq3YGS 5mjCP P4IOlhpkaALISuwoKnHnj+BvMD4SpaG8 GniMbPDduDHd 8 =`.

For the Hamming distance calculations, it is necessary to ensure that they are equal in length. For comparison purposes only, as the ciphertext is longer than the plaintext, a white space padding is done to the ending of the plaintext. After this process, both strings are 106 characters long.

After the calculation, the Hamming distance between these two strings is equal to 106 which means that all characters at the same position in both strings are different.

To calculate the Hamming distance at the bit level, both strings are converted to their binary format. Zero padding is done to the plaintext to conform to the required conditions to hamming distance calculations. After the padding, both bitstreams are 725 bits long.

The ideal value for the Hamming distance between the ciphertext and its plaintext, in binary, would be half of the total bitstream length, that is 50% of the bits are changed. The reason for this is that a 0% change (Hamming distance of 0) means that the ciphertext and the plaintext are identical, and, therefore, the ciphertext is readable to third parties. In contrast, a 100% change (Hamming distance equal to the message length) implies that the plaintext is the exact opposite of the plaintext, indicating that the encryption process is simply a XOR operation with a bitstream of only ones, and could be easily broken by an attacker.

TABLE 2. Hamming distance between the plaintext and the ciphertext.

Plaintext	Ciphertext	Hamming distance
Meet me at 15°45'47.5"S 47°52'18.6"W at 5:15 am tomorrow	fASPD2wpA4aahV Kf1Ia7Yei51Heh 82jEEmEmwBHFnz LAPqkX0Xq3YGS5 mjCPP4IOlhpkaA LISuwoKnHnj+Bv MD4SpaG8GniMbP DduDHd8=	106 (text level) 351 (bit level)

The calculated Hamming distance between the bitstreams is 351, out of a total of 725 bits, which means that 48.41% of the bits are different, a close value to the ideal 50%. Table 2 summarizes the findings of this comparison.

2) AVALANCHE EFFECT

In the following test, Alice sends her invitation to Bob again, but this time proposing the meeting to start one minute earlier, at 5:14. Thus, the plaintext `Meet me at 15°45′ 47.5″S 47°52′ 18.6″W at 5:14 am tomorrow` is to be encrypted.

The only modification in Alice's message is the time for the meeting, changing from 5:15 to 5:14. Therefore, the only difference is between the characters 5 and 4.

Translating these characters to their binary ASCII codes results:

Character 5: `0011 0101`.

Character 4: `0011 0100`.

Hence, the Hamming distance between these two characters is 1, as only the least significant bit changes and, as the remainder of Alice's messages are identical, it may be concluded that the Hamming distance between the two messages is 1.

The encryption process for the new invitation at 5:14 am, using the same architecture, results in the ciphertext: `j8q7xKbiasDfeYEBHdjZtS2SBHaUmguIPnx09YFH RdXdpgSWUPECTw2toPxZpnLwSiMySePeK2EJjFi+ dCyDX YPEqEkF6tk7avA0DE0lGIk=`.

Using the code depicted in Listing 3 the Hamming distance between the bitstreams of the two ciphertexts transmitted by Alice is measured as 346.

Therefore, a Hamming distance of 1 in the plaintext results in a large change in the ciphertext, of Hamming distance of 346. This is known as the avalanche effect, and it is demonstrated in Table 3.

The avalanche effect is a desired property in a cryptographic system to conceal the changes in the plaintext from an attacker who has access only to the ciphertext.

As demonstrated in this test, the hardware-based cryptography used on this architecture produces the avalanche effect, resulting in a more secure communication between the parties in the IM application.

TABLE 3. The avalanche effect evidenced by the Hamming distance.

Message 1	Message 2	Hamming distance
Meet me at 15°45'47.5"S 47°52'18.6"W at 5:15 am tomorrow	Meet me at 15°45'47.5"S 47°52'18.6"W at 5:14 am tomorrow	1 (bit level)
fASPD2wpA4aahV Kf1Ia7Yei51Heh 82jEEmEmwBHFnz LAPqkX0Xq3YGS5 mjCPP4IOlhpkaA LISuwoKnHnj+Bv MD4SpaG8GniMbP DduDHd8=	j8q7xKbiasDfeY EBHdjZtS2SBHaU mguIPnx09YFHRd XdpgSWUPECTw2t oPxZpnLwSiMySe PeK2EJjFi+dCyD XYPEqEkF6tk7av A0DE0lGIk=	346 (bit level)

3) RANDOMNESS EVALUATION

All tests in this section are conducted using the ciphertext from the `body` tag of Listing 2, in bit format. The results for these tests, along with a brief explanation for each evaluation, are shown in Table 4.

The National Institute of Standards and Technology (NIST) SP 800-22 [54] test suite is used to evaluate the randomness of the ciphertext generated by the proposed architecture.

It is important to state that some of the tests from NIST SP 800-22 could not be applied to the ciphertext bitstream due to the short length of the evaluated ciphered message. But, it does not mean that the proposed system would not pass the NIST SP 800-22 tests for larger cipher messages. In the present case, a part of the test suite is used to demonstrated that the proposed solution is robust enough in terms of cryptographic assumptions.

Having the above considerations said, a ciphertext must be as random as possible to reduce its predictability. In other words, the more random it is, the more it improves its secrecy and reduces the chances of it being maliciously revealed.

The number of bits in the bitstream could not be increased by extending the message length as, to achieve semantic security, the length of the ciphertext does not vary significantly with the length of the plaintext. Thus, the attacker cannot infer the message length from the ciphertext. For this reason, the tests that require a minimum bitstream length greater than 725 bit, which is the size of the ciphertext presented, are not used for the evaluation of the proposed architecture.

Comparing the results displayed in Table 4 with results from the literature [55]–[58], it can be inferred that the cryptosystem used in this work passes the NIST 800-22 battery test.

4) PERFORMANCE EVALUATION

To evaluate the performance of the proposed architecture, the `performance` interface is used [59]. The `now` method of this interface, which returns the time elapsed since the time origin (that is, since the beginning of the current document's lifetime), is called immediately before process ①

TABLE 4. Results for the NIST SP 800-22 test suite [54].

Test	Description	Parameter used	P-value	Compared reference
Frequency	Calculates the proportion of ones and zeros, and assesses its closeness to 0.5	N/A	0.480411	[55]
Block frequency	Similar to the frequency test, but the proportion is calculated for each block of the bitstream	Length of block = 128 b	0.651928	[56]
Cumulative sums	Tests the maximum distance from zero of a random walk defined by the cumulative sum of the sequence, both in forward and in reverse directions	N/A	0.560465 (forward) 0.593158 (reverse)	[55]
Runs	Defines substring of consecutive ones and consecutive zeroes and considers if the oscillation among these homogeneous substrings is too fast or too slow	N/A	0.615917	[57]
DFT (Spectral)	Detects periodic patterns in the bitstream, indicating a non-random feature in the sequence	N/A	0.250046	[55]
Serial	Evaluates whether the number of occurrences of the 2^m m-bit overlapping patterns is close to the value expected for a random sequence, in which all possible patterns have equal probabilities	Length of block (m) = 16 b	0.017445 0.158956	[58]
Linear complexity	Focuses on the the length of a linear feedback shift register (LFSR): random sequences have longer LFSRs	Length of block = 25 b	0.528184	[56]

TABLE 5. The performance of the architecture for different operations and message lengths.

Message length (chars)	Operation	Time measured (ms)
2	Encryption	540
	Decryption	538
128	Encryption	532
	Decryption	544
256	Encryption	543
	Decryption	534
512	Encryption	555
	Decryption	549
1024	Encryption	537
	Decryption	544
2048	Encryption	542
	Decryption	545

starts and also immediately after process ⑧ is finished (refer to Figure 4). Subtracting these values returns the time taken to complete all processes from ① to ⑧. Table 5 shows the obtained results.

The results showed that it takes about 540 ms for the message to go through the architecture, from ① to ⑧. It did not vary significantly with the operation (encrypting or decrypting text messages), nor with the length of the messages (considering that text messages exchanged in this type of application is not usually very long).

The low latency does not considerably decrease the speed of the message delivery, and, therefore, the architecture improves the security on the data transmission without affecting its performance, and the communication can still be considered instantaneous.

To prevent attacks such as Meltdown [60] and Spectre [61], the `now` method rounds the returned value (since Firefox 60, it rounds to 1 millisecond [59]), but the precision is high enough to assess the latency added by the proposed architecture, as the rounding (1 ms) is much lower than the calculated time (around 540 ms).

VII. CONCLUSIONS AND FUTURE WORKS

Considering the importance of encrypting messages in an IM application, this work demonstrated the use of a hardware based encryption service to provide authenticity, confidentiality and integrity to the messages exchanged.

Using a web extension and its API, it was possible to establish a communication between an IM client running in a user's browser and a hardware based cryptography system, enabling the verification of validity of digital certificates and the encryption of the body of instant messages.

Tests conducted on the cryptography system and on the overall architecture indicate that it is able to guarantee confidentiality, authenticity and integrity to the IM application in a manner not predictable by unauthorized third parties. Furthermore, the additional latency caused by the architecture is of a low order, and does not jeopardize the quality nor the immediacy of the communication.

A. FUTURE WORKS

One important feature of IM is the ability to send and receive files, which should also follow security principles. Converse.js already implements the transmission of files, but this project was limited to ciphering and deciphering textual messages only. Thus, as a future work, the architecture will be extended to include the secure transmission of files of any extension. This expansion does not require any change in the proposed architecture, but only the improvement of the Converse.js plugin, overriding the functions related to sending and receiving files.

Another common feature in IM applications is real-time streaming, either in audio or in video. Although Converse.js does not originally implement it even in clear data, and although it is a complex task to fully synchronize it with the hardware in asynchronous messages, it is possible to develop a plugin to extend its functionalities and, then, update the browser extension to include the encryption of this type of data.

To improve the security of the system, the support for a self-developed 2FA token may be implemented in the architecture, assuring the authentication and non-repudiation properties.

The authors also indicate as future work a study regarding the lower levels of implementation of the hardware cryptography device, with the implementation of requests serialization, Multiple-input and Multiple-Output (MIMO) and other measures necessary to guarantee scalability and performance.

REFERENCES

[1] *Instant Messaging Statistics Report, 2018–2022*, The Radicati Group, Inc., Palo Alto, CA, USA, 2018.
[2] A. Bolton, M. Meg, and J. Fluker, "Transforming the workplace: Unified communications & collaboration usage patterns in a large automotive manufacturer," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, doi: 10.24251/HICSS.2017.661.
[3] Y. Tang and K. F. Hew, "Is mobile instant messaging (MIM) useful in education? Examining its technological, pedagogical, and social affordances," *Educ. Res. Rev.*, vol. 21, pp. 85–104, Jun. 2017, doi: 10.1016/j.edurev.2017.05.001.
[4] J. C. de Oliveira, D. H. Santos, and M. P. Neto, "Chatting with Arduino platform through telegram Bot," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Sep. 2016, pp. 131–132, doi: 10.1109/ISCE.2016.7797406.
[5] H. Setiaji and I. V. Paputungan, "Design of telegram Bots for campus information sharing," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 325, Mar. 2018, Art. no. 012005, doi: 10.1088/1757-899X/325/1/012005.
[6] N. Hindocha and E. Chien, "Malicious threats and vulnerabilities in instant messaging," Symantec, Cupertino, CA, USA, White Paper, 2003. [Online]. Available: https://vxug.fakedoma.in/papers/malicious-threats-vulnerabilities-instant-messaging-03-en.pdf
[7] J. van den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proc. 25th Symp. Operating Syst. Princ. (SOSP)*, 2015, pp. 244–254, doi: 10.1145/2815400.2815417.
[8] Y. Ma, T. Chen, J. Lin, J. Yang, and J. Jing, "Entropy estimation for ADC sampling-based true random number generators," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 11, pp. 2887–2900, Nov. 2019, doi: 10.1109/TIFS.2019.2908798.
[9] R. Dudheria, "Assessing vulnerability of mobile messaging apps to man-in-the-middle (MitM) attack," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 7, pp. 23–35, Jul. 2018, doi: 10.5815/ijcnis.2018.07.03.
[10] H. Abualola, H. Alhawai, M. Kadadha, H. Otrok, and A. Mourad, "An Android-based Trojan Spyware to study the notificationlistener service vulnerability," in *Proc. ANT/SEIT*, 2018, pp. 465–471, doi: 10.1016/j.procs.2016.04.210.
[11] Kaspersky. (2007). *Kaspersky Security Bulletin 2006: Mobile Malware*. Accessed: Mar. 23, 2020. [Online]. Available: https://securelist.com/kaspersky-security-bulletin-2006-mobile-malware/36129/
[12] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic Malware analysis in the modern era—A state of the art survey," *ACM Comput. Surveys*, vol. 52, no. 5, pp. 1–48, Sep. 2019, doi: 10.1145/3329786.
[13] A. N. L. E. Lemos and L. P. Kurtz, "Sovereignty over personal data in Brazil: State jurisdiction facing denial of access to users' data by online platform Whatsapp," in *Proc. Global Internet Governance Academic Netw., Annu. Symp. (GigaNet)*, 2017, pp. 1–23, doi: 10.2139/ssrn.3107293.
[14] K. Rathi, U. Karabiyik, T. Aderibigbe, and H. Chi, "Forensic analysis of encrypted instant messaging applications on Android," in *Proc. 6th Int. Symp. Digit. Forensic Secur. (ISDFS)*, Mar. 2018, pp. 1–6, doi: 10.1109/ISDFS.2018.8355344.
[15] SecureList. *Fully equipped Spying Android RAT from Brazil: BRATA. 2019*. Accessed: Sep. 2, 2019. [Online]. Available: https://securelist.com/spying-android-rat-from-brazil-brata/92775/
[16] CVE-2019-3568 Detail. (2019). *National Institute of Standards and Technology*. Accessed: Sep. 2, 2019. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-3568
[17] The Rio Times. (2019). *Brazil's Justice Minister's Smartphone Hacked for Six Hours*. Accessed: Jun. 26, 2019. [Online]. Available: https://riotimesonline.com/brazil-news/brazil/brazils-justice-ministers-smartphone-hacked-for-six-hours/
[18] A. Simão, F. Sícoli, L. Melo, F. Deus, and R. Sousa, Jr., "Acquisition and analysis of digital evidence in Android smartphones," *Int. J. Forensic Comput. Sci.*, vol. 6, no. 1, pp. 28–43, Dec. 2011, doi: 10.5769/J201101002.
[19] A. M. M. Soares and R. T. de Sousa, Jr., "A technique for extraction and analysis of application heap objects within Android runtime (ART)," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, 2017, pp. 147–156, doi: 10.5220/0006204101470156.
[20] A. S. Petrenko, S. A. Petrenko, K. A. Makoveichuk, and P. V. Chetyrbok, "Protection model of PCS of subway from attacks type ≪wanna cry≫, ≪petya≫ and ≪bad rabbit≫ IoT," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (EIConRus)*, Jan. 2018, pp. 945–949, doi: 10.1109/EIConRus.2018.8317245.
[21] T. Sutikno, L. Handayani, D. Stiawan, M. A. Riyadi, and I. M. I. Subroto, "WhatsApp, Viber and telegram which is best for instant messaging?" *Int. J. Electr. Comput. Eng.*, vol. 6, no. 3, p. 909, Jun. 2016, doi: 10.11591/ijece.v6i3.pp909-914.
[22] I. Antunes and L. A. Kowada, "Explorando o Sistema de Criptografia signal no WhatsApp," in *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Porto Alegre, Brazil: Sociedade Brasileira de Computação, 2018, pp. 181–195.
[23] Kaspersky. (2019). *One Call on WhatsApp is Enough to Establish Surveillance*. Accessed: May 23, 2019. [Online]. Available: https://www.kaspersky.com/blog/whatsapp-call-zeroday/26941/
[24] S. Schröder, M. Huber, D. Wind, and C. Rottermanner, "When SIGNAL hits the fan: On the usability and security of state-of-the-art secure mobile messaging," in *Proc. IEEE Eur. Workshop Usable Secur.*, 2016, pp. 1–7, doi: 10.14722/eurousec.2016.23012.
[25] J.-I. Kim and J. W. Yoon, "Honey chatting: A novel instant messaging system robust to eavesdropping over communication," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 2184–2188, doi: 10.1109/ICASSP.2016.7472064.
[26] H.-B. Qin and X. Xu, "Solution to secure instant messaging based on hardware encryption," in *Proc. IEEE 16th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2015, pp. 844–847, doi: 10.1109/ICCT.2015.7399959.
[27] P. S.-Andre, *End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)*, document RFC 3923, RFC Editor, 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3923.txt
[28] P. K. Binu, H. L. Sreekutty, and V. S. Sreekutty, "Security plugin for mozilla which integrates cryptography and steganography features," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Dec. 2016, pp. 1–6, doi: 10.1109/ICCIC.2016.7919538.
[29] P. S.-Andre, *Extensible Messaging and Presence Protocol (XMPP): Core*, document RFC 6120, RFC Editor, 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6120.txt
[30] P. Saint-Andre, K. Smith, and R. Tronçon, *XMPP The Definitive Guide: Building Real-Time Applications With Jabber Technologies*, 1st ed. Newton, MA, USA: O'Reilly Media, 2009.
[31] L. Stout, J. Moffitt and E. Cestari, *An Extensible Messaging and Presence Protocol (XMPP) Subprotocol for WebSocket*, document RFC 7395, RFC Editor, 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7395.txt

[32] S. Loreto, P. S. Andre, S. Salsano, G. Wilkins, *Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP*, document RFC 6202, RFC Editor, 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6202.txt
[33] Writing a Plugin. (2019). *ConverseJS*. Accessed: May 25, 2019. [Online]. Available: https://m.conversejs.org/docs/html/plugin_development.html
[34] Openfire. (2019). *Ignite Realtime*. Accessed: May 25, 2019. [Online]. Available: https://www.igniterealtime.org/projects/openfire/
[35] M. Sharma, *Openfire Administration*, 1st ed. Birmingham, U.K.: Packt Publishing, 2008.
[36] Y. Kumar, R. Munjal, and H. Sharma, "Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures," *Int. J. Comput. Sci. Manage. Stud.*, vol. 11, no. 3, pp. 60–63, 2011.
[37] S. M. Bellovin, "Frank Miller: Inventor of the one-time pad," *Cryptologia*, vol. 35, no. 3, pp. 203–222, Jul. 2011, doi: 10.1080/01611194.2011.583711.
[38] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2016.
[39] K. Sako, "Semantic security," in *Encyclopedia of Cryptography and Security*, 2nd ed. Berlin, Germany: Springer, 2011, pp. 1176–1177.
[40] Y.-G. Yang and Q.-Q. Zhao, "Novel pseudo-random number generator based on quantum random walks," *Sci. Rep.*, vol. 6, no. 1, pp. 1–11, 2016, doi: 10.1038/srep20362.
[41] T. E. Tkacik, "A hardware random number generator," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2002, pp. 450–453, doi: 10.1007/3-540-36400-5_32.
[42] J. Reynolds, T. Smith, K. Reese, L. Dickinson, S. Ruoti, and K. Seamons, "A tale of two studies: The best and worst of YubiKey usability," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 872–888, doi: 10.1109/SP.2018.00067.
[43] Mozilla. (2019). *What Are Extensions?* Accessed: May 27, 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/What_are_WebExtensions
[44] Mitre. (2018). *Stolen Pencil*. Accessed: May 27, 2019. [Online]. Available: https://attack.mitre.org/groups/G0086/
[45] Mozilla. (2019). *Anatomy of An Extension*. Accessed: May 27, 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Anatomy_of_a_WebExtension#Background_pages
[46] Mozilla. (2019). *Manage Events with Background Scripts*. Accessed: May 27, 2019. [Online]. Available: https://developer.chrome.com/extensions/background_pages
[47] Mozilla. (2019). *Native Messaging*. Accessed: May 27, 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Native_messaging
[48] A. Melnikov, K. Zeilenga, *Simple Authentication and Security Layer (SASL)*, document RFC 4422, RFC Editor, 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4422.txt
[49] R. Norris, J. Miller, and P. S. Andre. SASL Integration. XEP-0034, XMPP Extensions, 2003. [Online]. Available: https://xmpp.org/extensions/xep-0034.html
[50] Google. (2019). *Messaging Passing Security Considerations*. Accessed: May 30, 2019. [Online]. Available: https://developer.chrome.com/extensions/messaging#security-considerations
[51] L. N. Childs, "RSA cryptography and prime numbers," in *Proc. Cryptol. Error Correction*, 2019, pp. 135–151, doi: 10.1007/978-3-030-15453-0_9.
[52] Phys.org. (2020). *New Record Set for Cryptographic Challenge*. Accessed: Apr. 16, 2019. [Online]. Available: https://phys.org/news/2020-03-cryptographic.html
[53] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," 2019, *arXiv:1905.09749*. [Online]. Available: http://arxiv.org/abs/1905.09749
[54] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, Al. Heckert, J. Dray, S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards & Technol., Gaithersburg, MD, USA, Tech. Rep. 800-22 Rev 1a, 2010.
[55] D. Xu and D. E. Tamir, "Pseudo-random number generators based on the collatz conjecture," *Int. J. Inf. Technol.*, vol. 11, no. 3, pp. 453–459, Sep. 2019, doi: 10.1007/s41870-019-00307-9.
[56] M. Y. M. Parvees, J. A. Samath, and B. P. Bose, "Cryptographically secure diffusion sequences—An attempt to prove sequences are random," in *Advances in Big Data and Cloud Computing*. Berlin, Germany: Springer, 2019, pp. 433–442, doi: 10.1007/978-981-13-1882-5_37.
[57] T. Kawashima and S. Mito, "Random number generation using magnetic domain images of magneto-optical materials," *Jpn. J. Appl. Phys.*, vol. 59, no. SE, Apr. 2020, Art. no. SEEA07, doi: 10.35848/1347-4065/ab6cb0.
[58] T. Kaya, "Memristor and trivium-based true random number generator," *Phys. A, Stat. Mech. Appl.*, vol. 542, Mar. 2020, Art. no. 124071, doi: 10.1016/j.physa.2019.124071.
[59] Mozilla. (2018). *Performance Interface Documentation*. Accessed: Dec. 4, 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Performance
[60] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 973–990.
[61] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 1–19, doi: 10.1109/SP.2019.00002.

GABRIEL ARQUELAU PIMENTA RODRIGUES was born in Brasilia, Brazil. He received the B.Sc. degree in communication networks engineering from the University of Brasília (UnB), in 2017, with an Exchange Program at the University of Limerick, Ireland, from 2014 to 2015. Since 2016, he has been a Researcher with LATITUDE-UnB, where he is currently a Scholarship Holder of a research project in partnership between the University of Brasilia and the Institutional Security Office of the Presidency of the Republic of Brazil. He is also working with the Security Operations Center (SOC), Dataprev, a state-owned company, in Brasilia. His research interests include network security and forensics.

ROBSON DE OLIVEIRA ALBUQUERQUE received the degree in computer science from the Catholic University of Brasília, in 1999, the M.B.A. degree in computer networks from the Educational Union of Brasília, in 2001, the master's degree in electrical engineering from the University of Brasília (UnB), Brazil, in 2003, the D.E.A. degree from the Complutense University of Madrid, Spain, in 2007, the doctor degree in electrical engineering from UnB, in 2008, and the Ph.D. degree in information systems from the Complutense University of Madrid, in 2016. He is currently pursuing the Ph.D. degree with the Post-Graduate Program on Electrical Engineering–Cybersecurity (PPEE), UnB. In 2000, he has participated and conducted projects in information technology, security and computer networks and published research results in the fields of computer networks, cybersecurity, information security, and network security. He also researches cybersecurity solutions applied to distributed systems and computer networks using machine learning techniques. He has expertise in computer and network forensics, applied cryptography, security in cloud computing and the Internet of Things. He is currently a Researcher with UnB and a member of GASS Research Group, Complutense University of Madrid. His fields of interest and research include cybersecurity, network security, information security, distributed systems, and computer networks.

GABRIEL DE OLIVEIRA ALVES was born in Uberaba, Brazil. He received the degree in information systems technology from the Centro Universitário Planalto do Distrito Federal, in 2005, and specialized in instructional design for EaD from the IBDIN-Instituto Brasileiro de Desenho Instrucional, in 2013. Since 2015, he has been a Researcher with LATITUDE-UnB. He is currently an Assistant Professor III with the Centro Universitário do Distrito Federal (UDF), where he is the Online Discipline Coordinator (DOLs). His research interests include distance education, project management, IT governance, user experience (UX), interface (UI), EaD, agile, and active methodologies.

FÁBIO LÚCIO LOPES DE MENDONÇA received the degree in data processing from the Catholic University of Brasilia (UCB), in 2004, the master's degree in electrical engineering from the University of Brasilia (UnB), in 2008, and the Ph.D. degree from UnB, in 2019. He is currently a Consultant in computer networks, software engineering, and project management. He has been conducting research activities as a Project Manager with the Decision-Making Technology Laboratory (LATITUDE), UnB, since 2005. He is currently a Substitute Professor of network engineering course with the UnB Electrical Engineering Department and an Adjunct Professor with the Projeção College (UniProjeção). His research interests include computer networks, software systems, project management, and the Internet of Things.

WILLIAM FERREIRA GIOZZA was born in Jaguarão, Brazil, in 1953. He received the degree in electronics engineering from the Aeronautical Technological Institute (ITA), São José dos Campos, Brazil, in 1976, the M.Sc. degree in electrical engineering from the Federal University of Paraíba (UFPb), Campina Grande, Brazil, in 1979, and the D.Eng. degree in computer science from the University of Paris 6, Paris, France, in 1982. From 1982 to 1998, he was an Associate Professor with UFPb. From 1998 to 2009, he was a Full Professor with Salvador University, Salvador, Brazil. Since 2009, he has been with the Decision Technologies Laboratory-LATITUDE, Electrical Engineering Department, University of Brasilia (UnB), Brasilia, Brazil, where he has been the In Charge of optical communication, high-speed networking, and cybersecurity education and research.

RAFAEL TIMÓTEO DE SOUSA, JR. (Senior Member, IEEE) received the bachelor's degree in electrical engineering from the Federal University of Paraíba (UFPB), Campina Grande, Brazil, in 1984, the master's degree in computing and information systems from the Ecole Supérieure d'Electricité–Supélec, Rennes, France, in 1985, and the Ph.D. degree in telecommunications and signal processing from the University of Rennes 1, Rennes, in 1988. He was a Visiting Researcher with the Group for Security of Information Systems and Networks (SSIR), Ecole Supérieure d'Electricité–Supélec, from 2006 to 2007. He has worked in the private sector from 1988 to 1996. Since 1996, he has been a Network Engineering Associate Professor with the Electrical Engineering Department, University of Brasília (UnB), Brazil, where he is currently the Coordinator of the Professional Post-Graduate Program on Electrical Engineering–Cybersecurity (PPEE) and supervises the Decision Technologies Laboratory (LATITUDE). He is Chair of the IEEE VTS Centro-Norte Brasil Chapter (IEEE VTS Chapter of the Year 2019) and of the IEEE Centro-Norte Brasil Blockchain Group. He is currently a Researcher with the Productivity Fellowship Level 2 (PQ-2) granted by the Brazilian National Council for Scientific and Technological Development (CNPq). His professional experience includes research projects with Dell Computers, HP, IBM, Cisco, and Siemens. He has coordinated research, development, and technology transfer projects with the Brazilian Ministries of Planning, Economy, and Justice, as well as with the Institutional Security Office of the Presidency of Brazil, the Administrative Council for Economic Defense, the General Attorney of the Union and the Brazilian Union Public Defender. He has received research grants from the Brazilian research and innovation agencies CNPq, CAPES, FINEP, RNP, and FAPDF. He has developed research in cyber, information and network security, distributed data services and machine learning for intrusion and fraud detection, as well as signal processing, energy harvesting and security at the physical layer.

ANA LUCILA SANDOVAL OROZCO was born in Chivolo, Magdalena, Colombia, in 1976. She received the degree in computer science engineering from the Universidad Autónoma del Caribe, Colombia, in 2001, and the M.Sc. degree in research in computer science and the Ph.D. degree in computer science both from the Universidad Complutense de Madrid, Spain, in 2009 and 2014, respectively. She holds a Specialization Course in computer networks from the Universidad del Norte, Colombia, in 2006. She is currently a Postdoctoral Researcher with the Universidad Complutense de Madrid. Her main research interests are coding theory, information security and its applications.