Design Issues in the Construction of a Cryptographically Secure Instant Message Service for Android Smartphones

Alexandre Melo Braga, Daniela Castilho Schwab
Centro de Pesquisa e Desenvolvimento em Telecomunicações (Fundação CPqD)
Campinas, São Paulo, Brazil
{ambraga,dschwab}@cpqd.com.br

***Abstract*—This paper describes design and implementation issues concerning the construction of a cryptographically secure instant message service for Android devices along with its underlying cryptographic library. The paper starts by discussing security requirements for instant message applications, and proceeds to the architecture of cryptographic components and selection of cryptographic services. Concerning this last point, two sets of services were implemented: one based only on standardized algorithms and other based solely on non-standard cryptography.**

Keywords-Cryptography; Security; Android; Instant Message.

I. INTRODUCTION

Currently, the proliferation of smartphones and tablets and the advent of cloud computing are changing the way software is being developed and distributed. Contemporary to this context change, the use in software systems of security functions based on cryptographic techniques is increasing as well.

The scale of cryptography-based security in use today has increased not only in terms of volume of encrypted data, but also relating to the amount of applications with cryptographic services incorporated within their functionalities. In addition to the traditional use cases historically associated to cryptography (e.g., encryption/decryption and signing/verification), there are several new usages, such as privacy preserving controls, bringing diversity to the otherwise known universe of threats to cryptographic software.

This paper discusses the construction of a mobile application for secure instant messaging on Android devices and a cryptographic library intended to support it. The paper focuses on design decisions as well as on implementation issues. This work contributes to the state of the practice by discussing the technical aspects and challenges of cryptographic implementations on modern mobile devices. The contributions of this paper are the following:

- The design of cryptographically secure instant message service;
- The elicitation of strong security requirements for cryptographic key negotiation over instant messages;
- The selection of a minimum set of standard cryptographic services capable of fulfill the requirements;
- The selection of non-standard cryptography in order to replace the whole standard algorithm suite.

The remaining parts of the text are organized as follows. Section II presents related work. Section III details requirements and design decisions. Section IV describes implementation aspects. Section V outlines improvements under development. Section VI concludes this text.

II. RELATED WORK

Nowadays, secure phone communication does not mean only voice encryption, but encompasses a plethora of security services built over the ordinary smartphone capabilities. To name just a few of them, these are SMS encryption, Instant Message (IM) encryption, voice and video chat encryption, secure conferencing, secure file transfer, secure data storage, secure application containment, and remote security management on the device, including management of cryptographic keys. All these security applications have been treated by an integrated framework [3] as part of a research project [4].

This section focuses on security issues of IM protocols and applications, as well as cryptography issues on Android devices.

A. Security issues in IM protocols and applications

The work of Xuefu and Ming [7] shows the use of eXtensible Messaging and Presence Protocol (XMPP) for IM on web and smartphones. Massandy and Munir [12] have done experiments on security aspects of communication, but there are unsolved issues, such as strong authentication, secure storage, and implementation of good cryptography, as shown by Schrittwieser et al.[39].

It seems that the most popular protocol for secure IM in use today is the Off-the-Record (OTR) Messaging [32], as it is used by several secure IM apps. OTR Messaging handshake is based upon the SIGMA key exchange protocol [15], a variant of Authenticated Diffie-Hellman (ADH) [45], just like Station-to-Station (STS) [6][46], discussed in further detail at Section IV.

A good example of security issues found in current IM software is a recently discovered vulnerability in WhatsApp [36]. The vulnerability resulting from misuse of the Rivest Cipher 4 (RC4) stream cipher in a secure communication

 ISBN: 978-1-61208-376-6

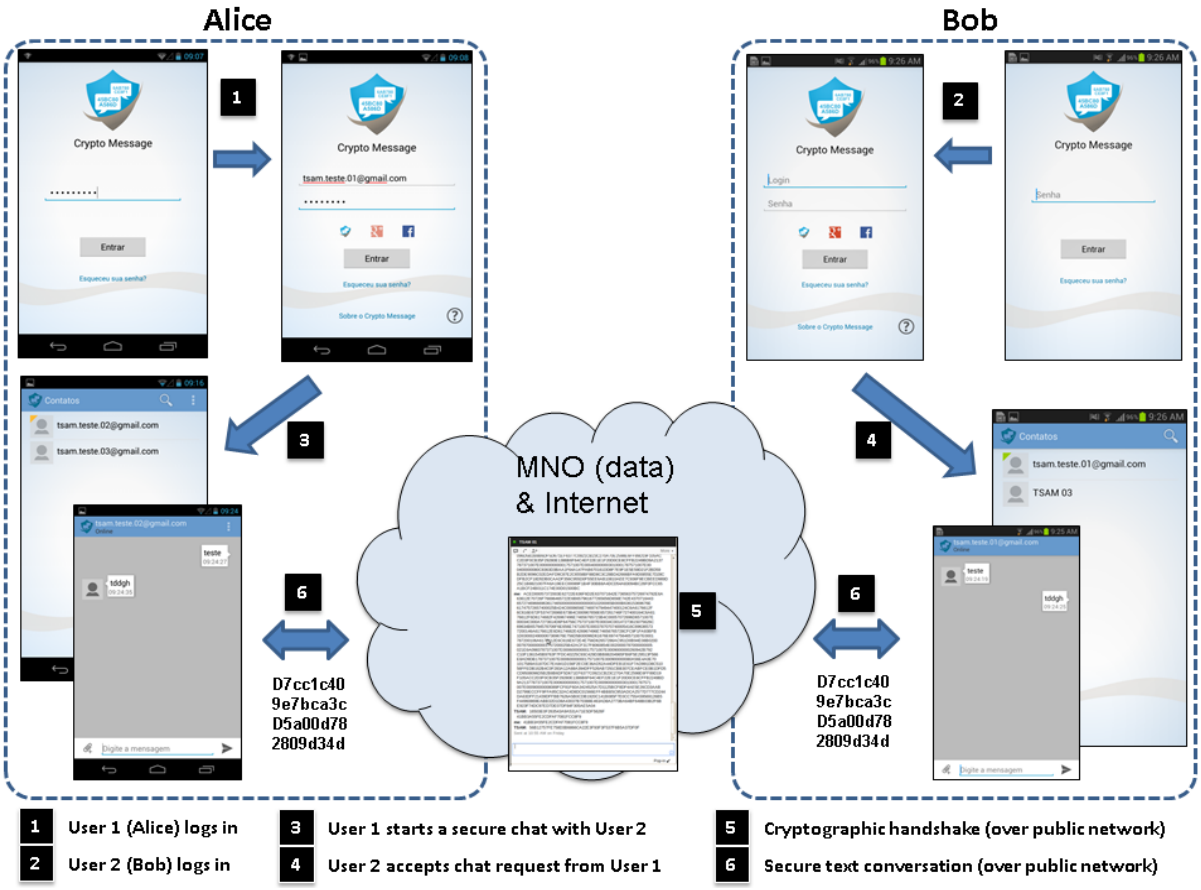

Figure 1. Basic flow of the secure exchange of instant messages.

protocol allowed the decryption, by a malicious third party able to observe conversations, of encrypted messages exchanged between two WhatsApp users. The issues related to this vulnerability are twofold. First, the incorrect use of RC4 stream cipher in place of a block cipher. Second, the reuse of cryptographic keys in both communication directions. The reuse of keys in a stream cipher and the existence of fixed parts, such as headers, at the communication protocol enabled the partial discovery of cryptographic keys.

B. Cryptography issues on Android devices

A recent study [2] showed that despite the observed diversity of cryptographic libraries in academic literature, this does not mean those implementations are publicly available or ready for integration with third party software. In spite of many claims on generality, almost all of them were constructed with a narrow scope in mind and prioritizes academic interest for non-standard cryptography. Furthermore, portability to modern mobile platforms, such as Android, is a commonly neglected concern on cryptographic libraries, as that evaluation has shown [2].

Moreover, there are several misuse commonly found on cryptographic software in use today. According to a recent study [24], the most common misuse of cryptography in mobile devices is the use of deterministic encryption, where a symmetric cipher in Electronic Code Book (ECB) mode appears mainly in two circumstances: Advanced Encryption Standard (AES) in ECB mode of operation (AES/ECB for short) and Triple Data Encryption Standard in ECB mode (TDES/ECB). There are cases of cryptographic libraries in that ECB mode is the default option, automatically selected when the operation mode is not explicitly specified by the programmer. A possibly worse variation of this misuse is the Rivest-Shamir-Adleman (RSA) cryptosystem in Cipher-Block Chaining (CBC) mode with Public-Key Cryptography Standards Five (PKCS#5) padding (without randomization), which is also available in modern cryptographic libraries, despite of been identified more than 10 year ago [34].

Another frequent misuse is hardcoded Initialization Vectors (IVs), even with fixed or constant values [34]. A related misuse is the use by the ordinary programmer of hardcoded seeds for PRNGs [24].

A common misunderstanding concerning the correct use of IVs arises when (for whatever reason) programmers need to change operation modes of block ciphers. For instance, the Java Cryptographic API [20] allows operation modes to be easily changed, but without considering IV requirements.

 ISBN: 978-1-61208-376-6

According to a NIST standard [30], CBC and Cipher feedback (CFB) modes require unpredictable IVs. However, Output feedback (OFB) mode does not need unpredictable IVs, but it must be unique to each execution of the encryption operation. Considering these restrictions, IVs must be both unique and unpredictable, in order to work interchangeably with almost all common operation modes of block ciphers. The Counter (CTR) mode requires unique IVs and this constraint is inherited by authenticated encryption with Galois/Counter mode (GCM) [31].

The two remarkable differences between the prototype described in this text and the related work are the following. First, the prototype uses STS protocol and its variants to accomplish authenticated key agreement. This has the benefit of facilitating protocol extension to use alternative cryptographic primitives. Second, authenticated encryption is the preferred encryption mechanism to protect messages, so the burden of IV management is minimized.

III. REQUIREMENTS FOR SECURE IM APPLICATIONS

This section describes the primary usage scenario of a mobile application for secure IM, as well as the selection of cryptographic services required by that application. This scenario illustrates the requirements elicitation that guided the design of the library.

A. Primary usage scenario for IM applications

The prototype for cryptographically secure, end-to-end communication operates on a device-to-device basis, exchanging encrypted IM via standard transport protocols. In the following text, the prototype is called CryptoIM.

CryptoIM implements the basic architecture used by all IM applications, using the standard protocol XMPP [35] at the transport layer. The application then adds a security layer to XMPP, which is composed of a cryptographic protocol for session key agreement and cryptographic transaction to transport encrypted messages. Therefore, CryptoIM is able to transport encrypted information through public services offered by providers such as Google (for Gtalk or HangOut) and Whatsapp.

The usage scenario that inspired the implementation of CryptoIM was to secure end-to-end communication, as described before. The two sides of communication (Alice and Bob) want to use their mobile device to exchange confidential and authentic messages. In CryptoIM, when a user selects a contact she wants to talk to, the protocol for secure conversation is initiated behind the scenes. The following action flow can be observed in Figure 1:

1) User 1 enters the application;
2) User 2 enters the application;
3) User 1 opens a conversation with User 2;
4) User 2 accepts the conversation;
5) Security negotiation occurs;
6) Secure conversation proceeds as expected.

This basic flow represents the simplest behavior needed for secure conversation. A secure conversation can be canceled by either party by sending a cancellation message.

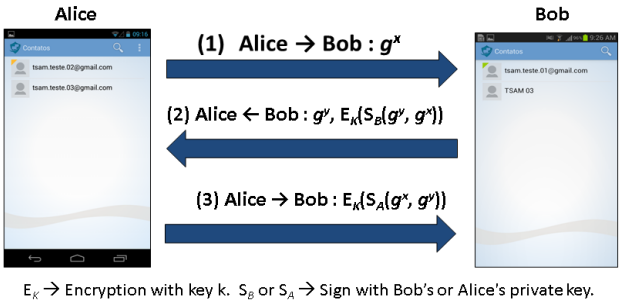

Figure 2. Station to Station (STS) protocol.

The security negotiation phase is indeed a protocol for key agreement, as illustrated by Figure 2.

B. Selection of cryptographic services

To accomplish the above mentioned scenario, Alice and Bob choose to use cryptographically secure communication with the following general requirements:

- An authentication mechanism of individual messages;
- An encryption algorithm and modes of operation;
- A key agreement protocol;
- A mechanism to protect cryptographic keys at rest.

In addition to a unique key for each conversation, that ensures security in the exchange of messages, a unique IV is generated for each exchanged message. To ensure that the protocol was followed in a transparent manner without user interference, automated messages were sent behind the scenes, so that the user does not see the exchange of messages for key negotiation. This prevents user from trying to interfere in the key agreement process.

To avoid known security issues in instant messaging applications [36][39], the key agreement protocol must provide the security properties described below [47]:

a) *Mutual authentication of entities*. For this property to be sustained in the protocol, signed messages must include the identities of both participants;
b) *Mutually authenticated key agreement*. The shared secret is a result of the underlying Key Agreement (KA) protocol. The freshness or novelty of the secret is the result of choosing random values for each conversation. The authenticity of secret sharing is guaranteed by digital signatures;
c) *Mutual confirmation of secret possession*. The decryption using a derived secret key confirms the possession of secret and evidences that the entity with knowledge of the secret is the same one signing the agreement messages. After a run of the protocol, the two participants observe each other performing encryption with shared secret key;
d) *Perfect Forward Secrecy (PFS)*. If a private key is compromised at some point in time, the security of session keys previously established is not affected. It is important for the maintenance of this property that the intermediate values are discarded and safely deleted at the end of a protocol run;

e) *Anonymity*. If the certificates are encrypted and the identities were omitted in the body of messages, a third party observing the communication network can not directly identify the interlocutors.

The cryptographic library supporting CryptoIM was designed to meet each one of these general requirements, resulting in an extensive implementation.

IV. DESCRIPTION OF THE IMPLEMENTATION

As a general goal, the CryptoIM cryptographic library is intended to be used in the protection of cryptographically secure communication via mobile devices. In order to be useful, the cryptographic library had to accomplish a minimum set of functional requirements. Each functional requirement generated a set of non-functional or supplementary requirements, mostly related to correctness of algorithms, compliance to industry standards, security, and performance of the implementation.

In order to facilitate the portability of the cryptographic library for mobile devices, in particular for the Android platform, the implementation was performed according to standard cryptographic Application Programming Interface (API) for Java, the Java Cryptographic Architecture (JCA), its name conventions, and design principles [16][20]-[23].

Once JCA was defined as the architectural framework, the next design decision was to choose the algorithms minimally necessary to implement a scenario of secure communication via mobile devices. The choice of a minimum set was an important design decision in order to provide a fully functional Cryptographic Service Provider (CSP) in a relatively short period of time. This minimalist construction had to provide the follow set of cryptographic functions:

a) A symmetric algorithm to be used as block cipher, along with the corresponding key generation function, and modes of operation and padding;
b) An asymmetric algorithm for digital signatures, along with the key-pair generation function. This requirement brings with it the need for some sort of digital certification of public keys;
c) A one-way secure hash function. This is a support function to be used in MACs, signatures and PRNGs;
d) A Message Authentication Code (MAC), based on a secure hash or on a block cipher;
e) A key agreement mechanism or protocol to be used by communicating parties that have never met before, but need to share an authentic secret key;
f) A simple way to keep keys safe at rest and that does not depend on hardware features;
g) A Pseudo-Random Number Generator (PRNG) to be used by all the key generation functions.

The current version of this implementation is illustrated by Figure 3 and presents the cryptographic algorithms and protocols described in the following paragraphs. The figure shows that frameworks, components, services and applications are all on top of JCA API. CryptoIM's Cryptographic Service Provider (CSP) is in the middle, along with BouncyCastle and Oracle providers. Arithmetic libraries are at the bottom.

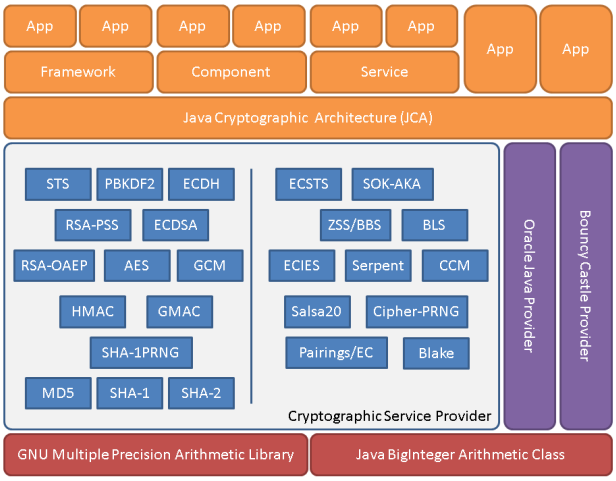

Figure 3. Cryptographic Service Provider Architecture.

Figure 3 shows CryptoIM CSP divided in two distinct cryptographic libraries. The left side shows only standardized algorithms and comprises a conventional cryptographic library. The right side features only non-standard cryptography and is an alternative library. The following subsections describe these two libraries.

A. *Standard Cryptography*

This subsection details the implementation choices for the standard cryptographic library. The motivations behind this implementation were all characteristics of standardized algorithms: interoperability, documentation, and testability. The programming language chosen for implementation of this cryptographic library was Java. The standard cryptography is a pure-Java library according to JCA.

The block cipher is the AES algorithm, which was implemented along with thee of operation: ECB, and CBC [30], as well as the GCM mode for authenticated encryption [31]. PKCS#5 [5] is the simplest padding mechanism and was chosen for compatibility with other CSPs. As GCM mode uses only AES encryption, the optimization of encryption received more attention than decryption. Implementation aspects of AES and other algorithms can be found on the literature [17][28][43]. This AES implementation was inspired by [33].

The asymmetric algorithm is the RSA Probabilistic Signature Scheme (RSA-PSS) built over the RSA signature algorithm. PSS is supposed to be more secure them ordinary RSA [27][43]. Asymmetric encryption is provided by the RSA Optimal Asymmetric Encryption Padding (RSA-OAEP) [27][43].

Two cryptographically secure hashes were implemented, Standard Hash Algorithm 1 (SHA-1) [26] and Message Digest (MD5). It is well known by now that MD5 is considered broken and is not to be used in serious applications, it is present for ease of implementation. In current version, there is no intended use for these two hashes. Their primary use will be as the underling hash function in

 ISBN: 978-1-61208-376-6

MACs, digital signatures and PGNGs. The MAC chosen were the Hash MAC (HMAC) [29] with SHA-1 as the underling hash function, and the Galois MAC (GMAC) [31], which can be directly derived from GCM mode. Standard Hash Algorithm 2 (SHA-2) family of secure hashes supplies the need for direct use of single hashes.

The need for a Key Agreement (KA) was fulfilled by the implementation of Station-to-Station (STS) protocol (Figure 2), which is based on Authenticated Diffie-Hellman (ADH) [45], and provides mutual key authentication and key confirmation [6][46].

Finally, the mechanism for Password-based Encryption (PBE) is based on the Password-Based Key Derivation Function 2 (PBKDF2) [5], and provides a simple and secure way to store keys in encrypted form. In PBE, a key-encryption-key is derived from a password.

B. Non-standard Cryptography

This subsection details the implementation choices for the alternative cryptographic library. The motivation behind the special attention given to the selection of alternative cryptographic algorithms was the recently revealed weaknesses intentionally included by foreign intelligence agencies in international encryption standards [19]. This fact alone raises doubt on the confidence of all standardized algorithms, which are internationally adopted.

In this context, a need arose to treat what has been called "alternative cryptography" in opposition to standardized cryptographic schemes. The final intent was strengthening the implementation of advanced cryptography and fostering their use. The non-standard cryptography is packaged as dynamic library written in C and accessible to Java programs through a Java Native Interface (JNI) connector, which acts as a bridge to a JCA adapter.

By the time of writing, this alternative library was under the final steps of its construction. It provides advanced mathematical concepts, such as bilinear pairings and elliptic curves, which are not fully standardized by foreign organizations and suffer constant improvements. The most advanced cryptographic protocols currently implemented are based on a reference implementation [8] and listed below.

a) Elliptic Curve Diffie–Hellman (ECDH) [11]. The key agreement protocol ECDH is a variation of the Diffie-Hellman (DH) protocol using elliptic curves as the underlying algebraic structure.
b) Elliptic Curve Digital Signature Algorithm (ECDSA) [25]. This is a DSA-based digital signature using elliptic curves. ECSS [11] is a variant of ECDSA.
c) Sakai-Ohgishi-Kasahara (SOK) [37]. This protocol is a key agreement for Identity-Based Encryption (IBE). It is also called SOKAKA (SOK Authenticated Key Agreement).
d) Boneh-Lynn-Shacham (BLS) [9]. A short digital signature scheme in which given a message m, it is computed S = H (m), where S is a point on an elliptic curve and H() is a secure hash function.
e) Zhang-Safavi-Susilo (ZSS) [14]. Similar to the previous case, it is a more efficient short signature, because it utilizes fixed-point multiplication on an elliptic curve rather arbitrary point.
f) Blake [41]. Cryptographic hash function submitted to the worldwide contest for selecting the new SHA-3 standard. It was ranked among the five finalists of this competition.
g) Elliptic Curve Augmented Encryption Scheme (ECIES) [11]. It is an asymmetric encryption algorithm over elliptic curves. This algorithm is non-deterministic and can be used as a substitute for RSA-OAEP, with the benefit of shorter cryptographic keys.
h) Elliptic Curve Station-to-Station (ECSTS) [11]. Variation of STS protocol using elliptic curves and ECDH as a replacement for ADH.
i) Salsa20 [18]. This is a family of 256-bit stream ciphers submitted to the ECRYPT Project (eSTREAM).
j) Serpent [40]. A 128-bit block cipher designed to be a candidate to the contest that chose the AES. Serpent did not win, but it was the second finalist and enjoys good reputation in the cryptographic community.

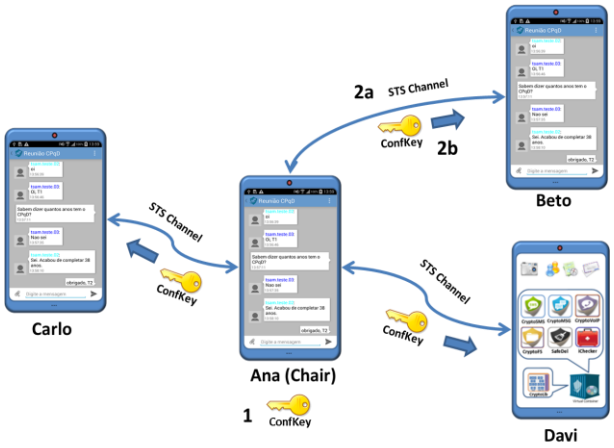

Figure 4. Key agreement for secure conference.

C. Evaluation of standard and non-standard cryptography

A previous work [2] identified lack of alternative cryptography in public libraries, such as non-standard elliptic curves and bilinear pairings. This prototype attempts to fulfill this gap by offering alternatives to possibly compromised standards. Its construction has been discussed in a recent paper [1]. Only key points are recalled here.

Considering security, protection against side-channel attacks was an important issue in the choice of alternative cryptography. Schemes with known issues were avoided, while primitives that were constructed to resist against such attacks were regarded. Also, the library offers alternatives for 256-bit security for both symmetric and asymmetric encryption. For instance, in symmetric encryption, Serpent-256 replaces AES-256. In asymmetric encryption, the same security level is achieved by elliptic curves over 521-bit finite fields, and replaces standard RSA with 15360-bit keys.

Considering performance measurements, experiments [1] have shown that standard cryptography can be competitive to other implementations. Also, in higher security levels, the

 ISBN: 978-1-61208-376-6

performance of non-standard elliptic-curve cryptography is significantly better than standard alternative. In contrast, non-standard pairings-based cryptography has shown relatively low performance. Figure 6 illustrates this behavior for signature operations on a Samsung Galaxy S III (1.4 GHz quad-core Cortex-A9, 1 GB RAM, and 16GB storage). Complete results can be found in [1].

The observed responsiveness shown by the prototype is quite competitive and usage has shown that delay caused by key negotiation is negligible, considering a local wireless network (Wi-Fi) and a household deployment of a XMPP server with few users. However, additional effort needs to be taken in order to optimize the mobile app as well as improve both performance and scalability on server-side application.

V. IMPROVEMENTS UNDER DEVELOPMENT

By the time of writing, two improvements were under construction. The first is a mobile PKI responsible for digital certification, which is fully integrated to the mobile security framework. PKI's Server-side is based upon the EJBCA PKI [13]. Client-side follows recent recommendations for handling certificates on mobile devices [38].

The second is a secure text conference (or group chat) via instant messages. As depicted in Figure 4, the Organizer or Chair of the conference requests the conference creation to the Server, as this is an ordinary XMPP feature. The key agreement for the requested conference proceeds as follows, where $Enc_k(x)$ means encryption of x with key k:

1. Chair (C) creates the key for that conference (c_k);
2. For each guest (g[i]), Chair (C) does:

   a) Opens a STS channel with key k: C ←→ g[i], key k;

   b) Sends c_k on time t to g[i]: C → g[i]: $Enc_k(c_k)$.

The steps above constitute a point-to-point key transport using symmetric encryption, which is provided by the STS protocol. After that, all guests share the same conference key and the conference proceeds as a multicast of all encrypted messages. Figure 5 shows a screenshot for a secure conference, in which users are differentiated by colors. Both the conversation and the interface are in Portuguese.

VI. CONCLUDING REMARKS

This paper discussed design and implementation issues on the construction of a cryptographically secure Instant Message application for Android and the underlying cryptographic library that supports it. This text has shown how cryptographic services can be crafted to adequately fit to a secure IM service in a way that is transparent to the final user, without sacrificing security. A well defined architecture allowed the selection and use of non-standard cryptography.

Future work includes other cryptographically secure services, such as SMS, group chat, and mobile PKI, as well as protections against side-channels and vulnerabilities of insecure programming. Also, performance over 3G networks is being measured and analyzed, for future improvements.

ACKNOWLEDGMENT

The authors acknowledge the financial support given to this work, under the project "Security Technologies for Mobile Environments – TSAM", granted by the Fund for Technological Development of Telecommunications – FUNTTEL – of the Brazilian Ministry of Communications, through Agreement Nr. 01.11. 0028.00 with the Financier of Studies and Projects - FINEP / MCTI.

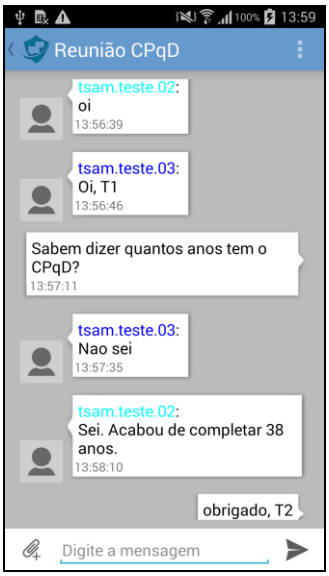

Figure 5. Screenshot of a secure text conference (group chat).

REFERENCES

[1] A. M. Braga and E. M. Morais, "Implementation Issues in the Construction of Standard and Non-Standard Cryptography on Android Devices," The Eighth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2014), in press.

[2] A. Braga and E. Nascimento, Portability evaluation of cryptographic libraries on android smartphones. In Proceedings of the 4th international conference on Cyberspace Safety and Security (CSS'12), Yang Xiang, Javier Lopez, C.-C. Jay Kuo, and Wanlei Zhou (Eds.). Springer-Verlag, Berlin, Heidelberg, 2012, pp. 459-469.

[3] A. M. Braga, "Integrated Technologies for Communication Security on Mobile Devices", The Third International Conference on Mobile Services, Resources, and Users (Mobility) , 2013, pp. 47–51.

[4] A. M. Braga, E. N. Nascimento, and L. R. Palma, "Presenting the Brazilian Project TSAM – Security Technologies for Mobile Environments", Proceeding of the 4th International Conference in Security and Privacy in Mobile Information and Communication Systems (MobiSec 2012). LNICST, vol. 107, 2012, pp. 53-54.

[5] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification", Version 2.0, RFC 2898. Retrieved [July 2014] from tools.ietf.org/html/rfc2898.

[6] B. O'Higgins, W. Diffie, L. Strawczynski, and R do Hoog, "Encryption and ISDN - A Natural Fit", International Switching Symposium (ISS87), 1987.

[7] B. Xuefu and Y. Ming, "Design and Implementation of Web Instant Message System Based on XMPP", Proc. 3rd International Conference on Software Engineering and Service Science (ICSESS), Jun. 2012, pp. 83-88.

[8] D. Aranha and C. Gouvêa, "RELIC Toolkit. Retrieved [July 2014] from code.google.com/p/relic-toolkit.

[9] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing", J. Cryptology, 17(4), Sept. 2004, pp. 297–319.

[10] D. Bornstain, Dalvik VM Internals. Retrieved [July 2014] from sites.google.com/ site/io/dalvik-vm-internals.

[11] D. Hankerson, A. J. Menezes, and S. Vanstone. Guide to Elliptic Curve Cryptography, Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2003.

[12] D. T. Massandy and I. R. Munir, "Secured Video Streaming Development on Smartphones with Android Platform", Proc. 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Oct. 2012, pp. 339-344.

[13] EJBCA PKI CA. Retrieved [July 2014] from http://www.ejbca.org.

[14] F. Zhang, R. Safavi-Naini, and W. Susilo, "An Efficient Signature Scheme from Bilinear Pairings and Its Applications", in F. Bao, R. H. Deng and J. Zhou, ed., 'Public Key Cryptography', 2004, pp. 277-290.

[15] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols." Advances in Cryptology-CRYPTO 2003, Springer Berlin Heidelberg, 2003, pp. 400-425.

[16] How to Implement a Provider in the Java Cryptography Architecture. Retrieved [July 2014] from docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/HowToImplAProvider.html.

[17] J. Bos, D. Osvik, and D. Stefan, "Fast Implementations of AES on Various Platforms", 2009. Retrieved [July 2014] from eprint.iacr.org/2009/501.pdf.

[18] J. D. Bernstein, The Salsa20 family of stream ciphers. Retrieved [July 2014] from cr.yp.to/papers.html#salsafamily.

[19] J. Menn, Experts report potential software "back doors" in U.S. standards. Retrived [July 2014] from http://www.reuters.com/article/2014/07/15/usa-nsa-software-idUSL2N0PP2BM20140715?irpc=932.

[20] Java Cryptography Architecture (JCA) Reference Guide. Retrieved [July 2014] from docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html.

[21] Java Cryptography Architecture Oracle Providers Documentation for Java Platform Standard Edition 7. Retrieved [July 2014] from docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html.

[22] Java Cryptography Architecture Standard Algorithm Name Documentation for Java Platform Standard Edition 7. Retrieved [July 2014] from docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html.

[23] Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files 7 Download. Retrieved [July 2014] from www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html.

[24] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in android applications," Proceedings of the 2013 ACM SIGSAC conference on Computer and Communications Security (CCS '13), 2013, pp. 73–84.

[25] NIST FIPS PUB 186-2. Digital Signature Standard (DSS). Retrieved [July 2014] from csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf.

[26] NIST FIPS-PUB-180-4. Secure Hash Standard (SHS). March 2012. Retrieved [July 2014] from csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.

[27] NIST FIPS-PUB-186. Digital Signature Standard (DSS). Retrieved [July 2014] from csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf.

[28] NIST FIPS-PUB-197. Announcing the ADVANCED ENCRYPTION STANDARD (AES). Federal Information Processing Standards Publication 197 November 26, 2001.

[29] NIST FIPS-PUB-198. The Keyed-Hash Message Authentication Code (HMAC). Retrieved [July 2014] from csrc.nist.gov/publications/fips/fips198/fips-198a.pdf.

[30] NIST SP 800-38A. Recommendation for Block Cipher Modes of Operation. 2001. Retrieved [July 2014] from csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

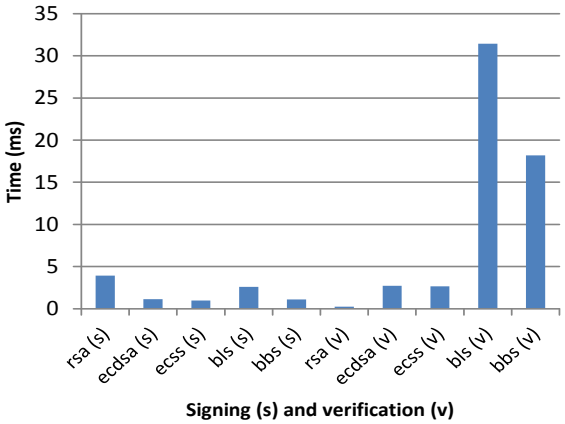

Figure 6. Time measurements for signature algorithms.

[31] NIST SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. 2007. csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.

[32] Off-the-Record Messaging webpage. Retrieved [July 2014] from otr.cypherpunks.ca.

[33] P. Barreto, AES Public Domain Implementation in Java. Retrieved [July 2014] from www.larc.usp.br/~pbarreto/JAES.zip.

[34] P. Gutmann, "Lessons Learned in Implementing and Deploying Crypto Software," Usenix Security Symposium, 2002.

[35] P. Saint-Andre, K. Smith, and R. Tronçon, "XMPP: The Definitive Guide - Building Real-Time Applications with Jabber Technologies", O'reilly, 2009.

[36] Piercing Through WhatsApp's Encryption. Retrieved [July 2014] from blog.thijsalkema.de/blog/2013/10/08/piercing-through-whatsapp-s-encryption.

[37] R. Sakai, K. Ohgishi, and M. Kasahara. "Cryptosystems based on pairing". The 2000 Symposium on Cryptography and Information Security (SCIS 2000), Okinawa, Japan, January 2000, pp. 26–28.

[38] S. Fahl, M. Harbach, and H. Perl, "Rethinking SSL development in an appified world," Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13 (2013), 2013, pp. 49–60.

[39] S. Schrittwieser et al., "Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications". Proc. 19th Network & Distributed System Security Symposium, Feb. 2012.

[40] SERPENT webpage, "SERPENT A Candidate Block Cipher for the Advanced Encryption Standard". Retrieved [July 2014] from www.cl.cam.ac.uk/~rja14/serpent.html.

[41] SHA-3 proposal BLAKE webpage. Retrieved [July 2014] from https://131002.net/blake.

[42] SpongyCastle webpage, Spongy Castle: Repackage of Bouncy Castle for Android, Bouncy Castle Project (2012), Retrieved [July 2014] from rtyley.github.com/spongycastle/

[43] T. St. Denis. "Cryptography for Developers", Syngress, 2007.

[44] The Legion of the Bouncy Castle webpage. Legion of the Bouncy Castle Java cryptography APIs. Retrieved [July 2014] from www.bouncycastle.org/java.html.

[45] W. Diffie and M. Hellman, "New Directions in Cryptography", IEEE Transact. on Inform. Theory, vol. 22, no. 6, Nov. 1976, pp. 644-654.

[46] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges", Designs, Codes and Cryptography (Kluwer Academic Publishers) 2 (2), 1992, pp. 107–125.

[47] W. Mao, "Modern cryptography: theory and practice", Prentice Hall PTR, 2004.