Implementation Issues in the Construction of an Application Framework for Secure SMS Messages on Android Smartphones

Alexandre Melo Braga[12], Romulo Zanco Neto[1], André Luiz Vannucci[1], and Ricardo Shiguemi Hiramatsu[1]
[1] Centro de Pesquisa e Desenvolvimento em Telecomunicações (Fundação CPqD)
Campinas, São Paulo, Brazil
[2] Universidade Estadual de Campinas (UNICAMP)
Campinas, São Paulo, Brazil
Email:{ambraga,romulozn,vannucci,ricardoh}@cpqd.com.br

***Abstract*—This paper details the construction of an application framework for SMS security that provides secrecy, integrity, authentication, and non-repudiation for SMS messages. The proposed framework integrates authenticated encryption and short digital signatures to management services for cryptographic keys and digital certificates. The framework hides from final users all details concerning certificate and key management. A flexible trade-off between security objectives and message length makes it possible to offer three levels of security: (i) secrecy only, (ii) secrecy and message authentication, and (iii) secrecy, origin authentication and non-repudiation. The main contribution is the use of short signatures for SMS origin authentication, which makes it possible to pack in a single, 140-byte SMS all information necessary to authenticate the origin of encrypted messages, while the user is still left with a useful length of text.**

Keywords - SMS; Cryptography; Android; Security.

I. INTRODUCTION

Nowadays, despite the growing popularity of new message services, such as instant messages, on mobile devices, the old-fashion Short Message Service (SMS) is still in plenty of use. Due to its higher reachability, relatively low cost, small amount of traffic, and existing infrastructure, varied flavors of SMS applications are being used in various fields, such as mobile commerce [12], home automation [24], and Machine-to-Machine (M2M) communication [30]. Even though SMS is not suitable for real-time remote control, as it suffers from transmission delay, message lost and lack of confidentiality [24], the ordinary SMS technology has evolved from a simple messaging service to an integral component of these security-critical applications. The SMS persistent popularity can also be attributed to modern smartphone platforms, such as Google Android [20], that provide to software developers an easy means to include SMS functionality into mobile applications.

This paper details the construction of an application framework for SMS security that provides secrecy, integrity, authentication, and non-repudiation for SMS messages. The proposed framework integrates authenticated encryption and short digital signatures to management services for cryptographic keys and Initialization Vectors (IVs). The main contribution of this paper is the use of short signatures for SMS origin authentication. Those signatures are only 33-bytes long, but provide the same security level of conventional signatures with at least twice its size. Such a small length saves space in message payload, so that an authenticated message can occupy only a single SMS.

The resulting application framework for SMS security is part of an integrated infrastructure for mobile security on mobile devices [5][6], that provides strong cryptography [3][4] to security-aware mobile applications [1][2].

The text is organized as follows. Section II offers background information about SMS internal workings. Section III provides related work on SMS security. Section IV details de design of the proposed solution. Section V contains a performance evaluation. Section VI discusses implementation issues of the prototype. Section VII concludes the text.

II. BACKGROUND

A. SMS workings

The Short Message Service (SMS) is a standardized facility defined as part of the Global System for Mobile Communications (GSM) series of standards [21] and the following description is based upon that documentation. SMS enables the transmission of up to 1120-bit (140 bytes or 160 7-bit characters) alphanumeric messages between mobile phones or external systems. The SMS service provides out-of-band delivery of messages, meaning that user can receive or transmit messages also when a voice call or data transfer is already in progress. The delivery and the integrity of an SMS are guaranteed by the network even in presence of temporary failures or unavailable stations.

Any message, sent via SMS, is not directly delivered to its destination, but it is stored into an SMS Center (SMSC) after passing through a Mobile Switching Center (MSC), which has the important role of message routing. If the destination device is unavailable or not connected to the GSM network, the messages are stored in the SMSC and delivered when the destination becomes available again, through another MSC. SMS has got a validity period, for which it will wait for the destination device to be available. After that time, the SMSC will delete the message. (The usual validity period is one day).

By enabling the delivery confirmation option, disabled by default, the sender of a SMS message can receive a

 ISBN: 978-1-61208-427-5

return message notifying whether it has been successfully delivered. Without the notification option, there is no guarantee about the correct reception or in the correct order of delivery of a previously sent SMS message.

Techniques for SMS concatenation and compression have been defined and incorporated in the GSM standard. However, these features may not be reliably implemented worldwide. Thus, only single uncompressed message delivery should work everywhere.

B. SMS on Android

Google's website on Android [20] is a well-documented source of information for software developers. According to that documentation [20], the basic building blocks of Android's internal messaging system are Intents. The Intent is a simple message object that holds additional information about an operation to be performed or of an event that has happened. Upon reception of an SMS message, intents are used to broadcast the contents of the received SMS message internally to registered receivers. To receive a broadcasted intent, an Android app needs to implement and register an appropriate broadcast receiver. The registered broadcast receiver's implementation determines the actions to be carried out, when a broadcast is received. Within an Android app, broadcast receivers can be registered either statically or dynamically.

If multiple broadcast receivers are registered for the same intent, they are called according to their given priorities. It is important to note that registered receivers have the possibility to abort a received broadcast intent, which prevents the intent from being forwarded to further registered receivers with a lower priority.

C. GSM encryption and SMS security

Global System for Mobile (GSM) communications has point-to-point encryption based upon the A5 family of ciphers [21]. A5/1 is the original cipher designed for use in the GSM protocol. After several weaknesses had been discovered, this cipher is not considered secure anymore [16]. A5/2 is a weakened version of A5/1 to allow for (historically kept) export restrictions to certain countries. It is therefore not considered secure [16]. A5/3 is also called Kasumi and is still in use today. Currently, known attacks do not inhibit its practical use in GSM. However, it is not intended to be used in further applications [16]. Though SMS encryption could be provided by network layer, on a point-to-point basis, it should be noted that end-to-end encryption of SMS still belongs to the application layer.

SMS technology is also inherently insecure, since the messages could be intercepted during the path that the messages follow during the transmission, including base stations, SMSC, and MSC. Particularly, SMS is subjected to replay attacks, message falsification, payload corruption, and sender impersonation.

Inside mobile devices, SMS sniffers and SMS catchers weaken the security of received and processed SMS messages, as these kinds of malware can simply use available APIs to intercept SMS messages [37]. SMS sniffers intercept SMS messages to spy on their content. After interception, the original message is forwarded to the default SMS-processing application (receiver). Differently, SMS catchers discard the message after its interception. This way, SMS catchers hide the original message from the user. This kind of malware intercepts incoming SMS messages either to spy on security-sensitive data transmitted via SMS or to receive SMS-based malware control commands [37]. The architecture of the Android platform facilitates the implementation of SMS catcher and sniffers even on non-rooted smartphones [37].

Starting with version 4.4, Android [20] enforces the forwarding of incoming SMS messages to a default SMS application. This renders realization of SMS catchers more difficult but by no means prevents SMS sniffers, as installed applications are still capable of reading incoming SMS messages [37].

III. RELATED WORK

The development of end-to-end encryption of SMS by mobile applications has a history of at least ten years and its evolution seems to coincide with the proliferation of mobile application platforms, as shown in next paragraphs.

In 2005, Hassinen presented SafeSMS [26] as an application meant for SMS end-to-end encryption. Encryption was based on a secret password shared between the sender and the recipient. The shared secret could also be generated and distributed by the system using a key exchange procedure, or by using a few text messages in a key exchange protocol, such as the Diffie-Hellman (DH).

In 2008, Lisonek and Drahansky [14] describe an application for securing of SMS which prevents tapping and substitution of SMS messages, by using RSA with OAEP padding. Also, Hossain et al [8] proposed a security scheme for improving the SMS security by encrypting SMS messages using GSM encryption technology, and digitally signing them with public key signature.

In 2009, Kuaté, Lo, and Bishop [36] proposed a supposed simple-to-use but cryptographically strong API for message encryption and authentication, called Linca, which was designed for limited devices, as well as a protocol called SMSSec for confidentiality and integrity.

In 2010, De Santis et al [7] presented Secure Extensible and Efficient SMS (SEESMS), a software framework written in Java, which allows two peers to exchange encrypted and digitally signed SMS messages. SEESMS supports the encryption of a communication channel through the Elliptic Curve Integrated Encryption Scheme (ECIES) and the Rivest-Shamir-Adleman (RSA) algorithms. The identity validation of the contacts involved in the communication is implemented through the RSA, Digital Signature Algorithm (DSA) and Elliptic Curve DSA (ECDSA) signature schemes. Also, Read and Martina presented SAMES [15], an Android application that allows

 ISBN: 978-1-61208-427-5

its users to send and receive text messages that are encrypted with the AES using hashing algorithms or Elliptic Curve DH (ECDH) for key creation. Certificates can also be created, signed, verified and sent to others using text messages. However, producing a method to convert certificates to a string less than 140 characters was not possible due to the signature itself exceeding this limit. In addition, Agoyi and Seral [25] compared RSA, ELGamal and Elliptic Curve Cryptography (ECC). They concluded, at the time (2010), that large key size algorithms were not suitable for SMS encryption due to small memory and low computational power of mobile phones. This has put ECC at an advantage over RSA and ELGamal in SMS encryption.

In 2011, Nanda and Awasthi analyzed Joint Channel Coding and Cryptography (JCCC) and Soft Input Decryption (SID) and proposed two algorithms to be used in SMS security: NTRUSign [9] and XTR – NR Signature [10]. Also, Qi, Pan, and Ding [31] used FPGA to implement the RSA algorithms and applied it in SMS encryption system. The implementation not only encrypted the SMS with hardware encryption, but also used a public-key server.

In 2012, Saxena and Chaudhari performed research [33] in securing SMS with a variant of ECDSA. Also, Saxena, Chaudhari, and Prajapati [34] proposed an encryption approach that used a password-based key exchange protocol based on DH and generated a shared secret-key which could be used in message encryption as well as in MAC functions.

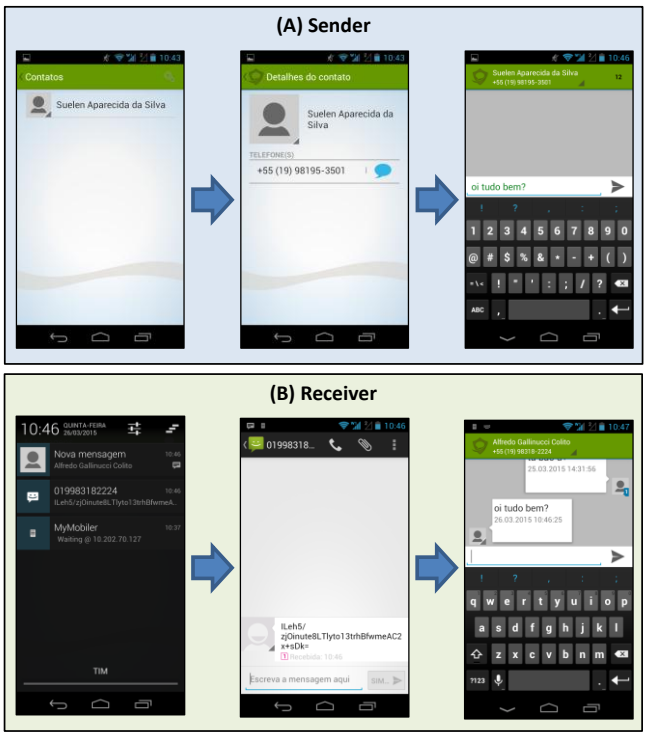

Figure 1. Screenshots showing the steps for sending and receiving encrypted SMS messages. In (A), Sender finds a contact, selects her, writes a message and sends it to her. In (B), Receiver is notified about an incoming message from a known contact, the message is shown encrypted by Android and can be decrypted only by CryptoSMS app.

Still in 2012, Chavan and Sabnees [39] proposed a technique that combines encryption and compression. The technique encrypts the SMS using ECC and after that, the encrypted SMS is compressed using a lossless algorithm. The supposed advantage is the decreasing of message lengths while maintaining the security. In addition, Pan, Ding, and Qi, [23] proposed a chaos-based encryption scheme combined with A5/1 algorithm for SMS security. The solution was tested on a phone-like system designed in Field-Programmable Gate Array (FPGA).

In 2013, Pereira et al [19] implemented SMSCrypto, a Java framework for securing SMS-based communications in mobile phones. The framework encloses a tailored selection of lightweight cryptographic algorithms and protocols, providing encryption, authentication, and signature services. Also, Khan, Bakhtiari, and Bakhtiari [28] proposed a framework that uses HTTPS for secure key exchange, as well as ECC and RSA as encryption algorithm to protect SMS messages against MITM attacks.

Still in 2013, Ariffi, Mahmod, Rahmat, and Idris [40] dealt with SMS encryption on Android smartphones. They proposed the use of a block cipher called 3D-AES for SMS encryption. Sagheer, Abdulhameed, and AbdulJabbar [11] proposed a solution for confidentiality and integrity for SMS by applying a hybrid cryptographic scheme which combines Advanced Encryption Standard (AES) for encryption and Rivest Cipher 4 (RC4) for key expansion and generation. It was implemented in Java and tested on a Nokia 5233. Pizzolante et al [38] investigates the feasibility of secure file transfer through SMS. Their solution compresses and eventually encrypts a variable-sized massage (or file) and sends it through standard SMS.

Again, in 2013, Saxena, Chaudhari, and Thomas [35] provided solutions to the repudiation attack on SMSs by using a variant of ECDSA. In 2014, Saxena and Chaudhari [32] proposed a protocol called EasySMS which provides end-to-end security during the transmission of SMS. This solution puts key management on the control of Mobile Network Operator (MNO).

In 2014, three unusual papers have appeared. First, Fahrianto, Masruroh, and Ando [17] argued that a combination of two “toy” ciphers (Caesar and Vinegére) was good enough to protect the secrecy of SMS, which is hardly believable. Second, Kashif [27] rediscovers RSA encryption for SMSs. However, the paper gives no clue about key management, randomization of RSA, and message splitting. Third, Al Bashar Abul Ulayee, Mesbah-Ul-Awal, and Newaj [22] used Caesar cipher in CBC mode to encrypt SMSs, and used a MAC for message authentication. The paper does implement a proprietary key management, arguing that the method is sufficient to surpass the weakness of Caesar Cipher, which is an unlikely assumption.

Finally, still in 2014, Patil, Sahu, and Jain [29] studied SMS compression in order to minimize the overhead of

 ISBN: 978-1-61208-427-5

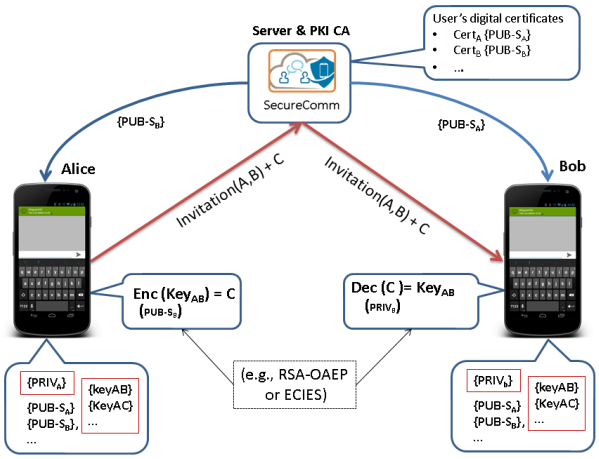

Figure 2. Shared-key transport is an invitation to a fellow to become a contact.

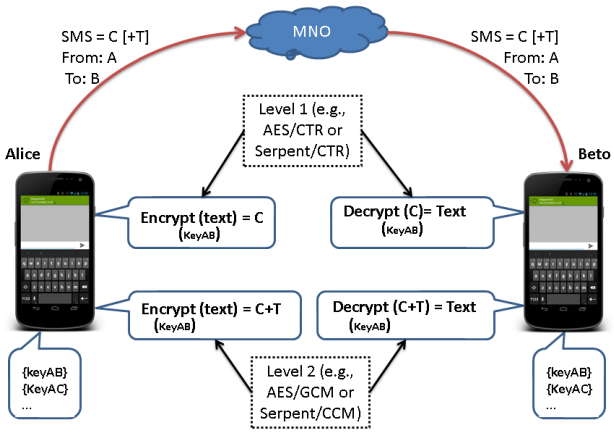

Figure 3. Levels 1 and 2 – encryption and authenticated encryption.

payload due to encryption, and proposed a method for compression of SMSs after encryption using ECC.

IV. PROPOSED SOLUTION

This section details the construction of the application framework for SMS security that provides encryption, integrity, authentication, and non-repudiation for SMS messages. From now, the framework is called CryptoSMS. CryptoSMS is not a stand-alone mobile application. In fact, it is supported by a server-side Java Enterprise Edition (JEE) application for management of users, apps and trust, which is integrated to a XMPP-based message service, a Public-Key Infrastructure (PKI) and Certification Authority (CA).

The general usage of CryptoSMS is quite simple, as seen in Figure 1, which shows screenshots for both sending and receiving of encrypted SMS messages. In Figure 1(A), Sender finds a contact, selects her, writes a message, and sends it to her. In Figure 1(B), the Receiver is notified about an incoming message from a known contact, the message is shown encrypted by Android (as was expected), and could only be decrypted by CryptoSMS.

Modern mobile platforms, such as Android, allow users to extend their device's behavior by installing new mobile apps. App installation is an import event, from service provider's point of view, and can be security sensitive. Not only app's binary code has to be accepted by the host device, but generally user accounts, along with other data, have to be created and registered as a profile for the new user.

CryptoSMS performs security sensitive actions during app installation, such as user account creation, key pair generation, public key upload, and (predefined) contacts synchronization, including the download of contact's digital certificates. Certification authority's root certificates are embedded in the app and distributed along with it.

It is a well-accepted behavior in social networks that users have contact lists (or friends) and that a pair of users becomes friends after an invitation is sent and accepted. In CryptoSMS, users' relationships behave like social networks contacts. This is an important usability aspect, as it resembles similar communication systems and utilizes the same software components, such as Android's notification service, to inform the user about new invitations.

The invitation process, illustrated in Figure 2, triggers several security related actions. For instance, when Alice sends to Bob an invitation to share secure SMSs, this request is supplemented by the generation of a secret key (K_{AB}), its encryption with Bob's public-key, and its secure transfer to CryptoSMS server. At this point, an asymmetric algorithm, such as RSA-OAEP or ECIES, can be used to securely transport K_{AB}. Bob's acceptance of Alice's invitation triggers the download of key K_{AB} to Bob's device.

The process for securing SMS messages depends on the desired security level and is illustrated in two parts by Figure 3 and Figure 4. The security level is not related to key sizes, as is usual in cryptography, since only 256-bit security is used for cryptographic algorithms. On the other hand, Security levels try to capture the security perception of the user and are related to the security objective of secrecy, integrity, authentication and non-repudiation, as follows:

- Level one grants only secrecy and was implemented with a symmetric block cipher (e.g., AES or Serpent) in in CTR mode, as shown in Figure 3;
- Level two grants secrecy, integrity, and message authentication, and was implemented with algorithms for symmetric authenticated encryption (e.g., AES/GCM or Serpent/CCM), as shown in Figure 3;
- Level three, illustrated in Figure 4, grants not only secrecy, integrity, and message authentication, but also grants user authentication and non-repudiation.

In level three, user authentication and non-repudiation of messages are accomplished by an unusual kind of digital signatures called short signatures, which are provided by asymmetric cryptographic algorithms and, as such, require management of key pairs and authenticated distribution of public keys. All cryptographic implementations are provided by a proprietary Cryptographic Service Provider (CSP) [3].

This security level three was implemented with a symmetric block cipher (e.g., AES or Serpent) in CTR mode and a short signature, such as Boneh-Lynn-Shacham (BLS) [13] or Zhang-Safavi-Susilo (ZSS) [18]. It is interesting to note that the short signatures used by CryptoSMS are only

 ISBN: 978-1-61208-427-5

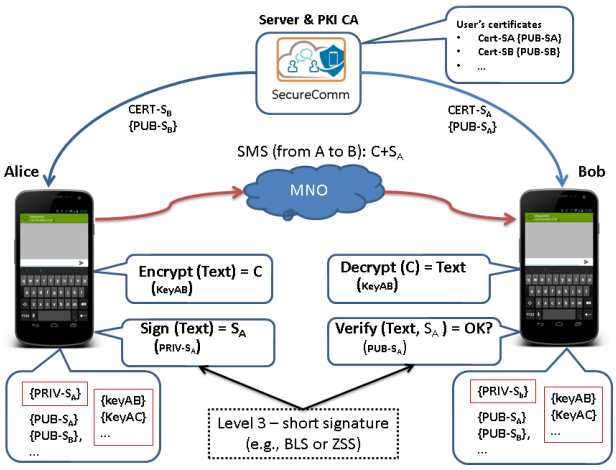

Figure 4. Security level 3 – encryption and short signatures.

33-byte long, but provide the same security of conventional, 66-byte ECDSA and are stronger then 256-byte RSA.

The resulting secure SMS carries with it an encrypted payload (in level one) and an authentication tag (in level two) or a short signature (in level three). Secret keys, public-key pairs, digital certificates, and other cryptographic parameters are all managed locally at the Android device by an app container fully integrated (synchronized) to CryptoSMS server and CA. It is an interesting usability aspect that users do not have to deal with cryptographic material, because it is all hidden from final user behind the concepts of invitations, contacts, data syncs, app installation, updates, and notifications.

V. PERFORMANCE EVALUATION

This section evaluates the performance in seconds taken to deliver an SMS message from one device to another. The performance of short signatures compared to conventional cryptography is also evaluated.

Figure 5(A) shows time measurements in seconds for SMS delivery from sender's device (dev1), through SMSC, to receiver's device (dev2). SMSC is the SMS routing center, a network entity that receives SMS messages from sender, temporarily stores them, and forwards them to receiver. Both sender and receiver were under the same SMSC, so network routing was minimal and latency was mostly due to network congestions.

A total of 20 SMS messages were sent in a loop of 50 iterations, resulting in 1,000 SMSs. There were no message losses or message disordering (rearrangement of message sequence during arrival). The overall time for SMS delivery stood around 20 seconds in average, and 90% of the values stood below that average. It is interesting to observe that the time taken from dev1 to SMSC is always balanced by the time taken from SMSC to dev2, preserving the average time.

Figure 5 also shows time measurements in milliseconds (ms) for two types of cryptographic services: symmetric encryption and digital signatures. The measurements were taken on a Samsung Galaxy S III (Quad-core 1.4 GHz Cortex-A9 processor, 1GB of RAM, and Android 4.1). Figure 5(B) shows time measurements of single-block encryption and decryption for Serpent and AES. Algorithms were setup with a 256-bit key. The bar chart shows the 9th centile of 10 thousand operations. Serpent is faster than AES.

Figure 5(C) shows generation of digital signatures for four algorithms: RSA (1024-bit key), ECDSA (with SHA-256), BLS, and ZSS, all of them for 256-bit security. For signature generation, RSA is the slowest one. Elliptic Curve Cryptography (ECC), as in ECDSA, is faster. Short signatures, such as BLS and ZSS, are not as fast as ECC. For signature verification, RSA is the fastest one, ECDSA is not that fast, and BLS/ZSS are the slowest ones.

Experiment has shown that network latency for SMS delivery (in seconds) is much larger than encryption and digital signature operations (in milliseconds). It is interesting to notice that the long latency for SMS delivery inhibits the use of key agreement protocols for negotiation of ephemeral keys through the SMS channel and it is an explanation for using pre-distributed secret keys.

BLS and ZSS are not as fast as ECC, since their constructions are based on complex mathematical structures

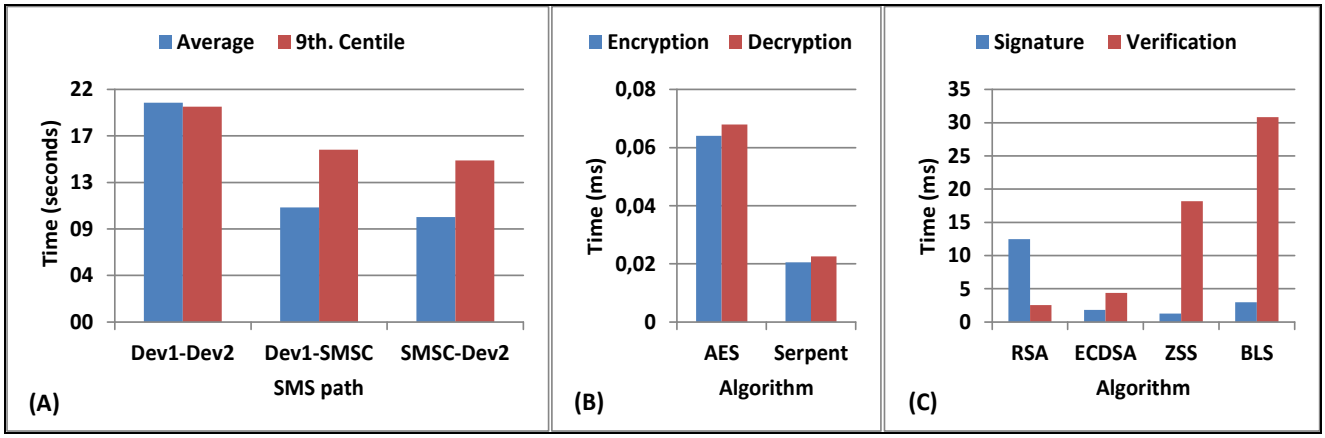

Figure 5. Three bar charts showing time measurements. In (A), time elapsed to transmit SMS messages from sender's device (dev1), through SMSC, to receiver's device (dev2). In (B), time for encryption and decryption of messages using AES and Serpent. In (C), time for digitally sign and verify signatures of messages with 64-byte length using RSA-PSS, ECDSA with SHA-256, ZSS, and BLS.

 ISBN: 978-1-61208-427-5

called bilinear pairings that require more computations. Here, there is a tradeoff, because the short signature can be roughly half the size of a regular ECDSA signature, but the verification is less efficient.

VI. DISCUSSION

This section discusses important implementation issues of CryptoSMS framework. One of them, the incorrect use of hardcoded Initialization Vectors (IVs), even with fixed or constant values, is a frequent issue on mobile devices. According to a NIST standard [41], the Counter (CTR) mode requires unique IVs and this constraint is inherited by authenticated encryption with Galois/Counter mode (GCM).

CryptoSMS implements IV management services in order to fulfill the uniqueness requirement for CTR, GCM and CCM modes of operation, as illustrated by Figure 6. During the invitation process, an IV initialization package is generated by Alice and saved at server until Bob retrieves it. The IV is split in two: a base and a count. The IV package consists of a base IV for Alice, a base IV for Bob, and a nonce to be used by both Alice and Bob as an initial counter.

Another aspect is the order in which encryption and authentication is performed over plaintext. Authenticated encryption (e.g., GCM and CCM) does not suffer from such an issue because encryption and message authentication are embedded in a single service. However, short signatures have to be programmatically composed with encryption by the application programmer. The correctly way to make this composition is to encrypt the message, then sign the encrypted message. This method provides not only integrity of cipher text and plaintext, but also does not provide any side information on the plaintext.

Base64 encoding is used as a sanitization measure in order to avoid misinterpretation of binary SMS messages by other receiver apps running at the same device, besides CryptoSMS. This measure reduces the total length of a single message from 140 to 105 bytes. After excluding the length of a 20-byte tag, the user is left with 85 bytes of text length. When a 33-byte signature is used instead, the user has 72 bytes of text length.

CryptoSMS makes it possible to pack in a single, 140-byte SMS message an encrypted payload along with its authentication tag or short signature. The payload could be at most 105-bytes long, which is enough for a large number of applications. Split the text in a sequence of SMSs is a way to circumvent this limitation. However, transmission delay, message lost, and reception out of order of the message sequence, may cause incorrect decryption.

Finally, a last concern is that CA software had to be customized to support digital certificates for non-standard algorithms. This means that certificates has to be generated and verified for public-keys of short-signature algorithms.

VII. CONCLUDING REMARKS

This paper discussed the construction of an application framework for SMS security on Android smartphones. The framework provides secrecy, integrity, authentication, and non-repudiation for SMS messages. The use of authenticated encryption and short digital signatures makes it possible to pack in a single SMS all necessary information to authenticate encrypted SMSs, while preserving a useful length of messages. The usability of security features is addressed by offering easy-to-use security levels and a seamless integration of cryptographic management into common app management functions.

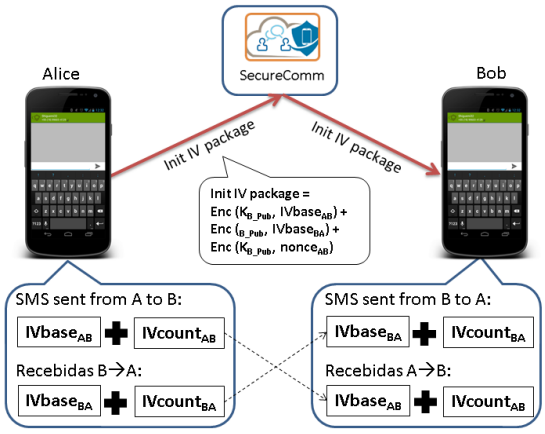

Figure 6. IV management for SMS security.

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.

REFERENCES

[1] A. Braga and A. Colito, "Adding Secure Deletion to an Encrypted File System on Android Smartphones," in proc. of The 8th Int'l Conf. on Emerging Security Inform., Systems and Technologies (SECURWARE), 2014, pp. 106–110.

[2] A. Braga and D. Schwab, "Design Issues in the Construction of a Cryptographically Secure Instant Message Service for Android Smartphones," in proc. of The 8th Int'l Conf. on Emerging Security Information, Systems and Technologies (SECURWARE), 2014, pp. 7–13.

[3] A. Braga and E. Morais, "Implementation Issues in the Construction of Standard and Non-Standard Cryptography on Android Devices," in proc. of The 8th Int'l Conf. on Emerging Security Information, Systems and Technologies (SECURWARE), 2014, pp. 144–150.

[4] A. Braga and E. Nascimento, "Portability evaluation of cryptographic libraries on android smartphones," in proc. of Cyberspace Safety and Security (CSS), 2012, pp. 459–469.

[5] A. Braga, "Integrated Technologies for Communication Security on Mobile Devices," in proc. of the 3rd Int'l Conf. on Mobile Services, Resources, and Users (Mobility), 2013, pp. 47–51.

[6] A. Braga, E. Nascimento, and L. Palma, "Presenting the Brazilian Project TSAM–Security Technologies for Mobile Environments," in proc. of Security and Privacy in Mobile Information and Comm. Systems, no. i, 2012, pp. 53–54.

[7] A. De Santis, A. Castiglione, G. Cattaneo, M. Cembalo, F. Petagna, and U. F. Petrillo, "An Extensible Framework for Efficient Secure SMS," in proc. of The Int'l Conf. on

 ISBN: 978-1-61208-427-5

Complex, Intelligent and Software Intensive Systems (CISIS), 2010, pp. 843–850.

[8] A. Hossain, S. Jahan, M. M. Hussain, M. R. Amin, and S. H. S. Newaz, "A proposal for enhancing the security system of short message service in GSM," in proc. of The 2nd Int'l Conf. on Anti-counterfeiting, Security and Identification (ASID), 2008, pp. 235–240.

[9] A. K. Nanda and L. K. Awasthi, "Encryption based channel coding algorithm for secure SMS," in proc. of The World Congress on Information and Communication Technologies (WICT), 2011, pp. 1282–1287.

[10] A. K. Nanda and L. K. Awasthi, "Joint Channel Coding and Cryptography for SMS," in proc. of The Int'l Siberian Conf. on Control and Communications (SIBCON), 2011, pp. 51–55.

[11] A. M. Sagheer, A. A. Abdulhameed, and M. A. AbdulJabbar, "SMS Security for Smartphone," in proc. of The 6th Int'l Conf. on Developments in eSystems Engineering (DeSE), 2013, pp. 281–285.

[12] A. Pourali, "The presentation of an ideal safe SMS based model in mobile electronic commerce using encryption hybrid algorithms AES and ECC," in proc. of The 8th Int'l Conf. on e-Commerce in Developing Countries: With Focus on e-Trust (ECDC), 2014, pp. 1–10.

[13] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," J.Cryptol., vol. 17, n. 4, 2004, pp.297–319.

[14] D. Lisonek and M. Drahansky, "SMS Encryption for Mobile Communication," in proc. of The Int'l Conf. on Security Technology (SECTECH'08), 2008, pp. 198–201.

[15] D. Read and J. Martina, "SAMES-Short Anonymous Message Encryption Scheme," in proc. of X Simpósio Brasileiro em Segurança da Informação e de Sistemas computacionais (SBSeg), Fortaleza, Ceará, Brasil, 2010.

[16] ENISA, "Algorithms, key size and parameters report", nov. 2014. Retrived [July 2015] from www.enisa.europa.eu/ activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014.

[17] F. Fahrianto, S. Masruroh, and N. Ando, "Encrypted SMS application on Android with combination of caesar cipher and vigenere algorithm," in proc. of The Int'l Conf. on Cyber and IT Service Management (CITSM), 2014, pp. 31–33.

[18] 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.

[19] G. C. C. F. Pereira et al., "SMSCrypto: A lightweight cryptographic framework for secure SMS transmission," in Jour. of Sys. and Software, vol. 86, no. 3, 2013, pp. 698–706.

[20] Google, Inc., "The Android Project," Retrived [July 2015] from http://www.android.com.

[21] GSM Doc 28/85 "Services and Facilities to be provided in the GSM System" rev2, June 1985.

[22] H. Al Bashar Abul Ulayee, M. Mesbah-Ul-Awal, and S. Newaj, "Simplified Approach Towards Securing Privacy and Confidentiality of Mobile Short Messages," in proc. of The 4th Int'l Conf. on Advanced Computing Communication Technologies (ACCT), 2014, pp. 403–408.

[23] J. Pan, Q. Ding, and N. Qi, "The Research of Chaos-based SMS Encryption in Mobile Phone," in proceedings of The 2nd Int'l Conf. on Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012, pp. 501–504.

[24] L. Pu, "An Improved Short Message Security Protocol for Home Network," in proc. of The Int'l Conf. on Future Computer and Communication, (FCC'09), 2009, pp. 62–65.

[25] M. Agoyi and D. Seral, "SMS Security: An Asymmetric Encryption Approach," in proceedings of The 6th Int'l Conf. on Wireless and Mobile Communications (ICWMC), 2010, pp. 448–452.

[26] M. Hassinen, "SafeSMS - end-to-end encryption for SMS," in proc. of the 8th Int'l Conf. on Telecommunications, (ConTEL), vol. 2, 2005, pp. 359–365.

[27] M. Kashif, "Secure SMS Communication Using Encryption Gateway and Digital Signature," in proc. of The 17th IEEE Int'l Conf. on Computational Science and Engineering (CSE), 2014, pp. 1430–1434.

[28] M. M. Khan, M. Bakhtiari, and S. Bakhtiari, "An HTTPS approach to resist man in the middle attack in secure SMS using ECC and RSA," in proc. of The 13th Int'l Conf. on Intelligent Sys. Design and Appl. (ISDA), 2013, pp. 115–120.

[29] M. Patil, V. Sahu, and A. Jain, "SMS text Compression and Encryption on Android O.S," in proc. of The Int'l Conf. on Computer Comm. and Informatics (ICCCI), 2014, pp. 1–6.

[30] N. Gligoric, T. Dimcic, D. Drajic, S. Krco, and N. Chu, "Application-layer security mechanism for M2M communication over SMS," in proc. of The 20th. Telecommunications Forum (TELFOR), 2012, pp. 5–8.

[31] N. Qi, J. Pan, and Q. Ding, "The Implementation of FPGA-based RSA Public-key Algorithm and its Application in Mobile-phone SMS Encryption System," in proc of The 1st Int'l Conf. on Instrumentation, Measurement, Computer, Communication and Control, 2011, pp. 700–703.

[32] N. Saxena and N. S. Chaudhari, "EasySMS: A Protocol for End-to-End Secure Transmission of SMS," in proc. of The IEEE Transactions on Information Forensics and Security, vol. 9, no. 7, 2014, pp. 1157–1168.

[33] N. Saxena and N. S. Chaudhari, "Secure encryption with digital signature approach for Short Message Service," in proc. of The World Congress on Information and Communication Technologies (WICT), 2012, pp. 803–806.

[34] N. Saxena, N. S. Chaudhari, and G. L. Prajapati, "An extended approach for SMS security using authentication functions," in proc. of The 7th IEEE Conf. on Industrial Electronics and Applications (ICIEA), 2012, pp. 663–668.

[35] N. Saxena, N. S. Chaudhari, and J. Thomas, "Solution to an attack on digital signature in SMS security," in proc. of The 5th Int'l Conf. on Modeling, Simulation and Applied Optimization (ICMSAO), 2013, pp. 1–6.

[36] P. H. Kuaté, J. L.-C. Lo, and J. Bishop, "Secure asynchronous communication for mobile devices," in proc. of the Warm Up Workshop for ACM/IEEE ICSE 2010, 2009, pp. 5–8.

[37] P. Teufl, T. Zefferer, C. Woergoetter, A. Oprisnik, and D. Hein, "Android - On-device detection of SMS catchers and sniffers," in proc. of The Int'l Conf. on Privacy and Security in Mobile Systems (PRISMS), 2014, pp. 1–8.

[38] R. Pizzolante, B. Carpentieri, A. Castiglione, A. Castiglione, and F. Palmieri, "Text Compression and Encryption through Smart Devices for Mobile Communication," in proc. of The 7th Int'l Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013, pp. 672–677.

[39] R. R. Chavan and M. Sabnees, "Secured mobile messaging," in proc. of The Int'l Conf. on Computing, Electronics and Electrical Technologies (ICCEET), 2012, pp. 1036–1043.

[40] S. Ariffi, R. Mahmod, R. Rahmat, and N. A. Idris, "SMS Encryption Using 3D-AES Block Cipher on Android Message Application," in proc. of The Int'l Conf. on Advanced Comp. Science App. and Tech. (ACSAT), 2013, pp. 310–314.

[41] NIST SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. 2007. Retrived [July 2015] from http://csrc.nist.gov/publications/ nistpubs/800-38D/SP-800-38D.pdf.

 ISBN: 978-1-61208-427-5