Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Engineering a secure mobile messaging framework

Aniello Castiglione [a], Giuseppe Cattaneo [a], Maurizio Cembalo [a], Alfredo De Santis [a], Pompeo Faruolo [a], Fabio Petagna [a], Umberto Ferraro Petrillo [b,*]

[a] Dip. di Informatica "R. M. Capocelli", Università di Salerno, Via Ponte don Melillo, I-84084 Fisciano (SA), Italy
[b] Dip. di Scienze Statistiche, Università di Roma "Sapienza", P.le Aldo Moro 5, I-00185 Roma, Italy

ARTICLE INFO

Article history:
Received 22 July 2011
Received in revised form
28 May 2012
Accepted 14 June 2012

Keywords:
Mobile secure communications
SMS
Elliptic curve cryptography
Performance analysis
Encryption
RSA

ABSTRACT

It is quite usual in the world of scientific software development to use, as black boxes, algorithmic software libraries without any prior assessment of their efficiency. This approach relies on the assumption that the experimental performance of these libraries, although correct, will match the theoretical expectation of their algorithmic counterparts.

In this paper we discuss the case of SEESMS (Secure Extensible and Efficient SMS). It is a software framework that allows two peers to exchange encrypted and digitally signed SMS messages. The cryptographic part of SEESMS is implemented on top of the Java BC library (The Legion of Bouncy Castle, 2010), a widely used open-source library. The preliminary experimentations conducted on SEESMS, discussed in Castiglione et al. (2010), revealed some unexpected phenomena like the ECDSA-based cryptosystem being generally and significantly slower than the RSA-based equivalent. In this paper, we analyze these phenomena by profiling the code of SEESMS and expose the issues causing its bad performance. Then, we apply some algorithmic and programming optimizations techniques. The resulting code exhibits a significant performance boost with respect to the original implementation, and requires less memory in order to be run.

1. Introduction

SMS messages have become one of the most widespread form of communication. They have been originally conceived as a tool for personal communication, however they are being increasingly used also in other application fields, especially as part of economic transactions. Among the reasons of this success there are their easiness of use and their availability on almost all cellular phones. The original SMS specification did not take into account any security feature: the communication between two peers occurs without any preliminary verification of their identities, neither the text of the SMS being exchanged is encrypted or digitally signed. As a consequence of this, many services relying on the use of SMS may be subject to security weaknesses. This problem is well-know and has been addressed several times by the scientific community. An appealing solution is to modify the GSM standard specifications, introducing security features at the SMS protocol level: this solution would probably be the most effective, however it would be very expensive to promote and to adopt, and is unlikely to be followed in the near future. An alternative approach is to inject security features at the application level. This is typically achieved by running a special-purpose software on the mobile devices of the communicating peers, in order to secure the SMS messages they exchange. For example, *confidentiality* in an SMS based

* *Corresponding author*. Tel.: +39 0649910513.
E-mail addresses: castiglione@acm.org (A. Castiglione), cattaneo@dia.unisa.it (G. Cattaneo), maucem@dia.unisa.it (M. Cembalo), ads@dia.unisa.it (A. De Santis), pomfar@dia.unisa.it (P. Faruolo), fabpet@dia.unisa.it (F. Petagna), umberto.ferraro@uniroma1.it (U. Ferraro Petrillo).
0167-4048/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2012.06.004

communication can be achieved by setting-up a public-key infrastructure where the sender encrypts the message using the public key of the receiver, the resulting text is sent to the receiver which, in turns, decrypts it using his private key.

It should be noted that even if the risks related to the security vulnerabilities of mobile communication are constantly increasing and are gaining media attention, users commonly still pay relatively small attention to these issues. Moreover, users are generally not inclined to install on their mobile devices software that would impact significantly on their user experience, by messing up the user interface or by excessively slowing-down common operations. This problem has been faced by SEESMS (Castiglione et al., 2010), a framework that allows users to exchange secure SMS messages. In this framework, the end-user can customize the degree of security to use when exchanging messages so to achieve an optimal trade-off between the overall security of the communication and the performance overhead experienced on his device. The original description of SEESMS was accompanied by the results of several experiments aimed at measuring this overhead in several communication scenarios, using different combinations of security parameters. Most of the proposed results were in line with the theoretical expectations, however there were some noteworthy exceptions. This was the case of the experiments concerning the generation of signatures using the Elliptic Curve based (Johnson et al., 2001) and the RSA based (Rivest et al., 1978) digital signature cryptosystems, denoted respectively as ECDSA and RSA. Despite the common expectations, ECDSA performed very poorly and revealed to be often slower than RSA, even when using long keys. Such a behavior is likely to be due to some issues in the design and/or the implementation of the ECDSA cryptosystem available with the Bouncy Castle (BC) library (The Legion of Bouncy Castle, 2010), a popular Java based cryptographic library used by SEESMS.

The objective of this paper is two-fold. First, it tries to further analyze and provide a stronger explanation to the results presented in Castiglione et al. (2010). The second objective is more general and concerns with the way algorithmic software libraries are often used without a preliminary performance analysis, and with the (possibly wrong) assumption that their experimental behavior will be always in line with the theoretical expectations. In our case, we will discuss some of the performance issues of the ECDSA implementation coming with the BC library and we will show how some of these issues could be effectively overcome by using both theoretical and programming optimizations. In our experiments, the cryptosystems using the optimized library exhibit a significant performance boost and a lower memory footprint than the original ones.

2. SEESMS

SEESMS (Secure Extensible and Efficient SMS) is a Java based framework for exchanging secure SMS, presented in Castiglione et al. (2010), that aims to be efficient by supporting several cryptosystems through a modular architecture. It can be seen as a tool that uses an SMS based communication channel as bearer service to exchange encrypted, non-repudiable and tamper-proof messages. The current version of SEESMS supports some of the most used digital signature schemes (i.e., RSA, DSA, ECDSA (Johnson et al., 2001)) and public-key based cryptosystems (i.e., RSA, ECIES (Certicom Research, 2010a)).

The usage of SEESMS requires an initial registration phase toward a trusted third-party server, called Secure SMS Management Center (SSMC), which delivers to the user a customized copy of the client application and which initiates the key-exchange protocol needed by the user to generate a pair of cryptographic keys, where the public key is sent to SSMS and made public.

The SEESMS framework adopts a hybrid architecture. If a user is interested in sending/receiving a secure message through SEESMS and has never used it before, then he has to contact a trusted third-party server, called Secure SMS Management Center (SSMC), to request a copy of the SEESMS client application and initiate the user registration phase. The SSMC is also in charge of storing and providing the public keys of legitimate registered users. Otherwise, the communication occurs in a peer-to-peer fashion.

One of the main advantage of SEESMS over similar systems is the possibility, for the user, to choose which combination of cryptosystem/security parameters to use during his communication. This possibility, which requires a certain degree of awareness from the final user, allows to achieve a good trade-off between the desired security level for the communication and the overall efficiency of the system. Moreover, one of the two peers of a communication (e.g., a service provider) could set a minimum security level to be fulfilled during the communication, giving the other peer the possibility to increase (but not decrease) it.

From an architectural point of view, the flexibility of SEESMS has been made possible by the adoption of a modular architecture (see Fig. 1) where the cryptographic functions of the framework are not built into SEESMS but are delegated to some external pluggable modules. The cryptosystems available in SEESMS have been implemented with the help of the BC library. This is a very popular cryptography API, available both for Java and C#, supporting several cryptosystems and compliant with the Java Cryptography Architecture.

We refer the interested reader to Castiglione et al. (2010) for further information about the architecture and the inner workings of SEESMS.

3. Experimental setup

Several tests have been conducted in order to evaluate the efficiency of the cryptographic algorithms available with SEESMS and to determine which security configuration would better suit the needs of a user. The framework is designed to handle both encryption/decryption operations and generation/verification of digital signatures operations. Nevertheless, in the tests have been evaluated only the signature related operations because, otherwise, it would have implied a longer exposition. Moreover, this choice is justified by the observation that signing operations have a computational complexity similar to the encryption ones.