PEARL: a PErformance evaluAtor of cRyptographic aLgorithms for mobile devices

Bringel Filho[1], Windson Viana[1], Rossana Andrade[2] and André Jalles Monteiro[3]

[1] Mestrado em Ciência da Computação (MCC) – Universidade Federal do Ceará (UFC)
{Bringel, Windson}@lia.ufc.br
[2] Departamento de Computação – Universidade Federal do Ceará (UFC)
rossana@lia.ufc.br
http://great.lia.ufc.br
[3] Departamento de Estatística – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil
jalles@ufc.br

Abstract. Limited computational power imposes new challenges during the implementation of security and privacy solutions for mobile devices. The choice for the most appropriate cryptographic algorithm for each mobile device has become a critical factor. In this paper, we present an approach for performance evaluation of cryptographic algorithms for mobile devices. To validate the approach, a tool called PEARL (PErformance evaluAtor of cryptogRaphic aLgorithms for mobile devices) is introduced. This tool collects and analyzes information related to the executions of the cryptographic algorithms in the mobile devices. PEARL also allows evaluating the performance of symmetrical and asymmetrical cryptographic algorithms and hashing functions for the J2ME platform.

1 Introduction

With the continuous improvements of the mobile computing, the market has shown an enormous diversity of mobile devices, such as cellular phones, Palms and Pocket PCs. They offer Internet connectivity using wireless communication technologies (e.g., IEEE 802.11, GSM, GPRS) allowing the user to get access to information at any time and any place. Besides that, these devices provide not only applications available by the manufacturers (e.g., agenda, calculator), but also the possibility of creating new applications to their users. These applications are implemented in the development platform supported by the device, for example, J2ME (Java 2 Edition Micron), Superwaba, .Net and Brew.

At the same time that mobile users have Internet connection and application development capabilities, they need security in both data transmission and data storage, which can be guaranteed using cryptography.

However, certain cryptographic algorithms, which are simple and efficient when executed in high performance processors available in desktops, can be inefficient or unsuitable to be implemented in low performance processors available in the mobile devices mentioned previously. Thus, the limited computational power in these devices imposes new challenges during the implementation of cryptosystems. In this context, the choice of the most appropriate algorithm for a certain mobile device has become a critical factor and brought the need for evaluating the efficiency of cryptographic algorithms for these devices.

Different from the works that appear in the literature [6][7], this paper presents an approach and its respective tool that allow the performance evaluation of cryptographic algorithms in a large range of mobile device types (e.g., Palms [13], Pockets PC [9]). For this, the respective tool is implemented in J2ME. J2ME is a largely used platform for developer mobile applications and the great majority of the current mobile devices support J2ME.

The main contributions of this work can be summarized as follows: (1) the tool allows the performance evaluation of cryptographic algorithms in mobile devices; (2) the evaluation results brings the possibility of verifying the viability of the algorithm application in certain device; (3) the tool allows to identify the algorithm that has the best performance for a certain input size; (4) the tool allows to evaluate the algorithm performance in different virtual machines in the same device; and (5) the evaluation results allow the construction of optimized cryptosystems for the analyzed device.

The article is organized as follows. Section 2 presents related works. Section 3 introduces our approach for the performance evaluation of cryptographic algorithms for mobile devices. In Section 4, we show our tool, called PEARL, which implements the proposed approach and, in Section 5, a case study. Finally, Section 6 ends up with final considerations and future works.

2 Related Work

According to the research done in the literature, we consider [6] the closest work related to our approach and its respective tool. In [6], the authors had implemented and evaluated libraries of cryptographic systems for the Palm platform. These libraries include implementations in C language of stream ciphers (SSC2, ARC4 and SEAL 3.0), block ciphers (Rijndael, DES, DESX and TripleDES), hash functions (MD2, MD4, MD5 and SHA-1) and integer arithmetical operations of multiple precisions. The algorithms had been evaluated in Palm V (16 Mhz processor, 2Mb) and Palm IIIc (20 Mhz processor, 8Mb) devices. The evaluations had consisted of the algorithm executions in these devices by ciphering plaintext of various sizes (2 Kb, 50 Kb and 4Mb). In this data, the amount of ciphered bytes (or digested, in case of hash functions) was calculated in each second (bytes/s).

With the evaluation result, the authors had identified that the SSC2 has better performance than ARC4 and SEAL 3.0 for small plaintext (1Kb). For big plaintext (4Mb), SEAL 3.0 performs two times faster than SSC2. From the block ciphers analysis, the authors had observed that the Rijndael is four times faster than the DES.

The performance evaluation has also demonstrated the use viability of integer arithmetical operations of multiple precision, which are operations used in asymmetrical cryptographic algorithms. It was also observed that implementations of embedded cryptographic algorithms in applications have better performance than implementations based on system libraries.

It is also worth to mention the work developed in [7], where the authors show a performance analysis of cryptographic protocols in mobile devices. The analysis is applied to SSL, S/MIME and IP/Sec protocols, which are widely used in network applications. The analysis results show that the time necessary to execute cryptographic functions is small, not causing significant impact in the performance of real time mobile transactions. The analysis was done only in the iPAC H3630 device (206 Mhz StrongARM processor, RAM 32 Mb with operational system Windows CE Pocket PC 2002). However, the analyzed device has a performance similar to a desktop and it is still necessary to investigate mobile devices with low computational power.

Our approach, on the other hand, allows evaluating the algorithm performance in any mobile device, if the device presents support for J2ME/MIDP 1.0 platform or higher version. The performance of each algorithm is evaluated according to the time necessary to execute cryptographic operations over an input text of pre-defined size. As a result of the evaluation, we present the amount of bytes that can be ciphered per second (bytes/s), in the case of symmetrical and asymmetrical cryptographic algorithms, and the amount of bytes that can be digested per second for hash functions.

3 An Approach for Performance Evaluation of Cryptographic Algorithms in Mobile Devices

This section describes an approach to evaluate the performance of cryptographic algorithms in mobile devices. The approach is divided in two phases, as follows. The first phase consists in collecting information related to the executions of the algorithms in the mobile devices (called samples). This information is used to evaluate the performance of the algorithms in the device. The second phase is responsible for evaluating this information to identify which are the algorithms with the best performance in the analyzed device.

The following sub-sections present the description of the collected sample format for each execution and more details about the approach phases.

3.1 Sample Format Description

The samples contain information related to the executions of the cryptographic algorithms in the analyzed device. The sample format allows identifying the analyzed devices, the virtual machine, the algorithm, the message size and the initialization and execution times. This information is enough to evaluate the cryptographic algorithm performance in each device. A sample is created and stored for further analysis in each execution of an algorithm with pre-defined input size. Each sample refers to a single execution and contains the following fields:

- *idDevice:* identifies the analyzed device. Each device receives an identification before initiating the information collection;
- *idVM:* identifies the virtual machine;
- *idAlg:* identifies the evaluated algorithm;
- *sizeText:* is the size of the input text.
- *timeInit:* is the algorithm inicialization time. Each algorithm has an specific inicialization time, that can change according to the input size;
- *timeExec:* is the algorithm execution time. For each input size and evaluated algorithm, we have a different execution time.

3.2 Sample Collection

As already mentioned in the previous sub-section, a sample contains information related to an execution in the analyzed device of an algorithm with an input text defined previously. The algorithms get input texts from different sizes. For each input size, it is collected the initialization algorithm time and the time necessary for ciphering or calculating the digest. The sample that contains this information is stored in a local repository in the mobile device called SLB – Samples Local Bank (see Fig. 1).

The size of the initial input text is 1Kb that is duplicated until reaching the maximum size multiple of two that is specified for the evaluation (e.g., 256Kb) or the maximum size that the device support (e.g. 64KB for palm devices with Sun J2ME VM). The algorithm performance evaluation becomes more precise with this variation in the input size. This variation allows the analysis of the variation on the execution time and the initialization time, according to the increase of the input text size.

A test case corresponds to an execution of an algorithm with a defined input text size. For each input size and algorithm, test cases are executed and they generate a sample for posterior analysis.

After the collection phase, the samples that are stored in the LBS are transmitted to a computer desktop, which has more processing power, to be analyzed. In the desktop, a database, called SRB - Samples Remote Bank (see Fig. 1), stores the received samples. The sample analysis is then executed on the SRB data instead of in the mobile device. The analysis technique is presented in the next sub-section.

3.3 Sample Analysis

In our approach, the test cases (i.e., executions of the algorithms with a specific input size) generate the samples to be analyzed. The sample analysis of an algorithm presents as a result the amount of ciphered or digested bytes per second, for each input size.

To calculate the speed (i.e., bytes/s) of the cryptographic operations we use the input text size (e.g., 1024, 2048, and 4096 Bytes) and the modal value of the execution time for that size. We use the modal value, which is the value repeated more often in a distribution, since we need to obtain a fast and close measurement that represents the most typical value of the distribution. So, at the end, we have as a result the speed that more frequently occurs for each text size and evaluated algorithm.

4 The PEARL Tool

In order to validate the approach presented in the previous section, a tool called PEARL (Performance EvaluAtor of cryptogRaphic aLgorithms for mobile devices) is developed. This section describes the architecture and the main functionalities of the tool.

4.1 Architecture

Fig. 1 illustrates the architecture of the PEARL tool, including its components and the interaction among them. The tool is developed in the Windows environment, using the Java language (J2ME/MIDP 1.0 platform and J2SE), Resin server and MySQL database. The tool also executes in the Linux environment by changing Java servlets (SR – Samples Receiver and SA – Samples Analyzer) to the Apache Tomcat server.

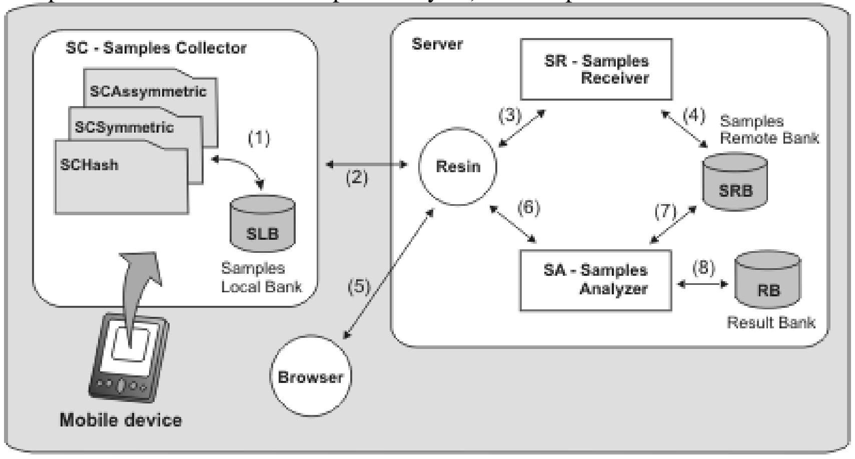

Fig. 1. PEARL Architecture: Components and their Relationships

The SC – Samples Collector module executes the test cases of the cryptographic algorithms in the analyzed device. The SCAsymmetric, SCSymmetric and SCHash submodules store the samples related to the algorithm test cases (1) in the SLB. After the collection phase, it is necessary the user interaction for executing the synchronization (i.e., the sending process) of the collected samples with the server (2, 3). The user invokes the SR servlet to receive and to store the samples sent in the SRB (4). After the sample reception, the user interacts with the system core using any Web browser (5) and invokes the SA servlet to execute the analysis of the received samples (6).

This servlet, in turn, recovers the samples stored in the SRB (7) to do the analysis. As a result, we get the amount of ciphered or digested bytes per second (i.e., the algorithm throughput) for each text size and evaluated algorithm. The analysis results are stored in the RB (8).

4.2 Samples Collector

As mentioned before, the Samples Collector (SC) module is responsible for the test cases of the cryptographic algorithms in the mobile devices. This module is divided in 3 (three) submodules: SCSymmetric, SCHash and SCAsymetric.

Each submodule allows to choose the minimum and maximum text input sizes, the algorithm to be evaluated, and the amount of test cases that are necessary for the analysis. It is also possible to execute all algorithms with the same minimum and maximum text input sizes and amount of test cases.

All submodules are developed in J2ME platform. We use J2ME Wireless Toolkit 1.0.4_01 [7] and SonyEricsson J2ME SDK [14] tools. The evaluated cryptographic algorithms are implemented and distributed for the Legion of the Bouncy Castle, which is a research group that develops open source Java APIs in the security area.

To calculate timeExec and timeInit, we apply the java.util.Date class, specifically the getTime() method, which returns in milliseconds the time since 00:00 h of January 1st, 1970. We will detail each submodule as follows:

1) *SCSymmetric:* is responsible for the test cases of the symmetrical cryptography algorithms. They are divided in 2 (two) categories of algorithms: block ciphers and stream ciphers. This submodule allows to evaluate any one of the following 18 (eighteen) cryptographic algorithms: AES, AES Fast, AES Light, Blowfish, CAST5, CAST6, 3DES, DES, IDEA, RC2, RC5 32 bits, RC5 64 bits, RC6, Rijndael, Serpent, Skipjack, and Towfish block ciphers as well as RC4 stream cipher. Some of these algorithms are variations of existing ciphers, such as, the AES Fast and the AES Light (they are variations of the AES) and others are different implementations for the same algorithm like Rijndael and AES. For each block cipher algorithm and input text size, it is evaluated four ciphering ways: Electronic Codebook (ECB), Cipher Block Cleaning (CBC), Cipher Feedback (CFB) e Output Feedback (OFB). Then, four test cases are generated; each of them applies one of the blocks ciphering way mentioned before. In the test cases that use CFB or OFB ciphering ways, the block size used is 64 bits and the ciphering key size used is 128 bits.

2) *SCHash:* is responsible for the test cases of the hash functions. In total, we apply 10 (ten) algorithms (functions and variations): MD2, MD4, MD5, RIPEMD128, RIPEMD160, SHA 256, SHA 384, SHA 512, SHA-1 and Tiger. As mentioned in section II, SCHash does not collect the timeInit information, just the timeExec.

3) *SCAsymmetric:* is responsible for the test cases of the asymetrical cryptography algorithms. It is possible to evaluate 2 (two) public key algorithms: RSA and ElGamal. TimeInit is the amount of time for the key generation. The created key is 1024 bits, with public exponent e = 65537.

4.3 Samples Analyzer and Result Bank

The Samples Analyzer (SA) module is responsible for calculate the throughput (i.e. byte/s) of the cryptographic operations using the approach mentioned at the section 3.3.

The results of SA are stored at the Result Bank (RB). So, at the end of the PEARL tool executing, RB has a result speed for each algorithm at each device. This allows a mobile application developer knows the time that a specific device spends to execute an algorithm of cryptographic.

5 Case Study

As a case study, PEARL tool was executed at Palms M130 and M515 and Sony Ericsson P800 smartphone. Table 1 presents more technical details about them.

For each device, symmetrical and asymmetrical algorithms, as well as hash functions are executed. The initial size of the evaluated text is 1Kb (see section 4), being duplicated until reaching 256Kb. This maximum size is chosen for being the maximum multiple of two that a vector of bytes can be allocated to the memory of the Palm devices listed above using the IBM virtual machine. On the other hand, P800 Smartphone does not have such limit, allowing evaluations with a vector of bytes greater than that, (e.g., 2048 Kb).

In each device we execute 50 test cases for each input size and algorithm, as follows: 3400 executions of block cipher symmetrical algorithms, 50 executions of the RC4 stream cipher algorithm, 500 executions of hash functions and 100 executions of asymmetrical cryptographic algorithms. The results of the executions of the asymmetrical algorithms are not able to finish before the end of this paper writing.

Beyond the throughput for each device-algorithm execution, the analysis of RB generated by AS (see Fig. 1) shows for all the three devices that RC4 stream cipher presents the best performance among the symmetrical cryptographic algorithms. On the other hand, the Rijndael algorithm presents the worst performance, it does not matter the operation way (i.e., ECB, CBC, CFB and OFB) and evaluated text size. Table II presents a summary of the evaluation results of the symmetrical cryptographic algorithms, considering the ECB operation way in each analyzed device.

The minor throughput column shows the worst algorithm throughput (e.g., 61,02 Bytes/s is the Rijndael algorithm throughput in the Palm M130 device). Similarly, the major throughput column presents the best performance algorithm throughput in the analyzed device. The growth order of the algorithm throughput is established considering the plaintext size equal to the final size of the evaluated text (i.e., 256 kb).

We observe with the evaluation of hash functions that the MD4 algorithm presents best performance. On the contrary, the MD2 presents the worst performance. Table 3 demonstrates a summary of these evaluation results in each analyzed device. The growth order of the hash function throughput is established considering the plaintext size equal to 256 kb.

The analysis results of the Palm devices demonstrates that, despite the Palm M515 has twice memory than Palm M130, the throughput of cryptographic algorithms in these devices is very close (see Table 2 and Table 3). We can also observe that the growth order of the algorithm throughput in these devices is equal, differing from the growth order presented in the SonyEricsson P800

Table 1. Mobile Devices Investigated

Device	RAM	Processor	SO	VM
Palm M130	8Mb	Motorola Dragonball VZ 33 Mhz	Palm OS 4.1	IBM
Palm M515	16Mb	Motorola DragonBall VZ 33 Mhz	Palm OS 4.1	IBM
Sony Ericsson P800	12Mb	32-bit RISC ARM9 @ 156 MHz	Symbian OS 7.0	Sony Ericsson

Table 2. Summary of the algorithm evaluations of symmetrical cryptography

Device	Minor throughput (Bytes/s)	Major through-put (Bytes/s)	Throughput increasing order
Palm M130	61,02	1587,6	Rijndael < 3DES < Serpent < Skipjack < RC2 < AES Light < DES < IDEA < CAST6 < RC6 < AES < CAST5 < Twofish < AES Fast < RC5 32 Bits < BlowFish < RC5 64 Bits < RC4
Palm M515	60,97	1575,4	Rijndael < 3DES < Serpent < Skipjack < RC2 < AES Light < DES < IDEA < CAST6 <RC6 < AES < CAST5 < Twofish < AES Fast < RC5 32 Bits <BlowFish < RC5 64 Bits < RC4
Sony Ericsson P800	5953,49	2048000	Rijndael < 3DES < Serpent < CAST6 < AES < AES Light < Skipjack < RC2 < IDEA < CAST5 < RC6 < DES < Twofish < RC5 32 Bits < BlowFish < AES Fast < RC5 64 Bits < RC4

Table 3. Summary of algorithm evaluations of hash functions

Device	Minor throughput (Bytes/s)	Major throughput (Bytes/s)	Throughput increasing order
Palm M130	88,12	2546,07	MD2 < SHA256 < SHA512 <= SHA384 < SHA1 < RIPEMD160 < RIPEMD128 < Tiger < MD5 < MD4

Palm M515	87,9	2546,57	MD2 < SHA256 < SHA512 <= SHA384 < SHA1 < RIPEMD160 < RIPEMD128 < Tiger < MD5 < MD4
Sony Ericsson P800	9309,1	2048000	MD2 < RIPEMD160 < SHA256 < SHA512 <= SHA384 < SHA1 < RIPEMD128 < MD5 < Tiger < MD4

These results show that PEARL tool can be used by mobile application developer to verifying the viability of an algorithm application in J2ME at an specific device, identify the algorithm that has the best performance for a certain input size and knows the throughput of specific algorithm.

6 Conclusion and Future Work

The performance evaluation of cryptographic algorithms is a requirement for the safe and efficient development of cryptosystem in devices of low computational power.

The approach presented in this paper and its respective tool allows evaluating the efficiency of the algorithms in a large variety of mobile devices (e.g., Palms, cellular, Pocket PC). The approach phases allows managing both qualitative and quantitative information. The PEARL tool implements the approach, which identifies the most efficient algorithms in J2ME environments at a certain analyzed device as presented in the case study.

The case study validates the approach and its tool, presenting the evaluation results of the cryptographic algorithms in the Palms M130, M515 and smartphone SonyEricsson P800. The results can interfere directly in the developer´s choice of the algorithm that will be composing the cryptosystem developed to provide security during the data transmission and storage. Thus, the analysis results allow the development of optimized cryptosystems.

It is worth mentioning that just the performance factor does not guarantee the development of safe cryptosystems. For each algorithm is necessary to study the security offered for the algorithm and its vulnerabilities to attacks mentioned in the literature.

We consider as future work the development of a framework that will allow the development of safe applications for mobile devices in J2ME platform. From the evaluation results, the developer will be able to choose the algorithms to be added to the framework to guarantee the security in the data transmission and storage of data in the mobile device. Aiming at organizing a repository of the analysis results, we also propose new evaluations in other devices.

Finally, we also would like to investigate about the plaintext type (e.g., ASCII text, binary text) and its effects on the performance of cryptographic algorithms. During the evaluations of this work we have already identified variations on the execution time according to input type changes.

References

1. Winkle W. V.; Palm Business Applications. The Ultimate guide to mobile computing: 2002 mobile Technology. p. 82-89, 2002.
2. Stallings, William. Network security essentials: applications and standards / William Stallings. ISBN: 0-13-016093-8.
3. The Legion of Bouncy Castle. Criptography API para Java. Disponível em <www.bouncycastle.org> Acesso em: fev 2003.
4. Menezes, A. J.; Oorschot, P. C. V.; Vanstone S. A.; Handbook of Applied Cryptography. Out, 1996.
5. Susilo, W. Securing Handheld devices. 10th IEEE International Conference, Wollongong, p. 349-354, 2002.
6. Wong, D. S.; Fuentes, H. H.; Chan, A. H.; The Performance Measurement of Cryptographic Primitives on Palm Devices. Boston, p. 1-10, 2002.
7. Java 2 Platform, Micro Edition (J2ME). Disponível em <http://java.sun.com/j2me>. Acesso em: 10 mar. 2003.
8. Argyroudis, P. G.;Verma, R.;Tewari, H.;O'Mayony, D. Performance Analysis of Cryptographic Protocols on Handheld Devices, 2003.
9. Windows CE, Disponível em <http://www.microsoft.com/windowsce/>. Acesso em: 10 jan. 2004.
10. Superwaba: The Java VM for PDAs. Disponível em < http://www.superwaba.com.br>. Acesso em: 10 mar. 2003.
11. Microsoft. Mobile ASP.NET Web Applications. Disponível em < http://www.asp.net/default.aspx?tabIndex=6&tabId=44>. Acesso em: 10 mar. 2003.
12. Qualcomm. Qualcomm Brew. Disponível em <http://www.qualcomm.com/brew/>. Acesso em: 15 mar. 2003.
13. PalmOne. Disponível em <http://www.palm.com/>. Acesso em: 15 mar. 2003.
14. SonyEricsson. Disponível em: <http://developer.sonyericsson.com/site/global/home/p_home.jsp>. Acesso em: 10 mar. 2003.