Procedure for the Detection of Anomalies in Public Key Infrastructure (RSA Systems)

Antonio Castro Lechtaler [1, 2]; Marcelo Cipriano [1, 2, 3];
Eduardo Malvacio[1]; Sebastián Cañón[4]

[1] Escuela Superior Técnica - IESE, Buenos Aires, C1426AAA; [2] Facultad Regional Buenos Aires - Universidad Tecnológica Nacional, Buenos Aires, C1179AAQ; [3] Instituto Fátima, Buenos Aires, C1437BZH; [4] Universidad Tecnológica Nacional, Buenos Aires, C1041AAJ.
Antonio Castro Lechtaler, acastro@utn.edu.ar; Marcelo Cipriano, marcelocipriano@iese.edu.ar;
Eduardo Malvacio, edumalvacio@gmail.com; Sebastián Cañón, scanon@rec.utn.edu.ar

Abstract. Cryptographic techniques authenticate users and protect information confidentiality. These tasks are performed by subsystems called Oracles. The most popular is the RSA system based on two large primes granting secure services. In 2008, a programming error in Open-SSL of the Debian system was detected. Its number generator was biased and creating system vulnerabilities by turning certificates predictable. This paper analyses the generic performance of a RSA cryptographic Oracle and develops a methodology to detect irregularities and anomalies in the quality of the certificates. A solution is presented to identify possible colliding primes. In this way, network administrators and information security experts can audit the performance of cryptographic modules in use.

Keywords: Asymmetric Cryptography, SSL, RSA, Integers Primes, Predictable Primes.

1. Introduction

Confidentiality is one of the cornerstones of Information Security, according to the Norms BS 7799, ISO 17799 and others from the family ISO 27000. Cryptography provides cryptographic primitives for this service upon user and system requests.

Web server authentication, web browsing secure sessions, home-banking operations, virtual private networks (VPN), and e-commerce communications are some of the basic uses of confidentiality obtained through the application of cryptography.

Therefore, the absence of anomalies is essential in the sub-systems dealing with user authentication through public key cryptography. Otherwise, the system becomes vulnerable to attacks. Communication could fail if somebody knows the private key [1] [2].

Even when the RSA cryptographic primitive might be robust, its implementation with the SSL [3] protocol or others could be deficient.

The quality of the prime numbers might not be appropriate [4] or the generation of such numbers could be predicted, as it was identified in 2008. [5] [6].

Recently, Lenstra et al [7] have shown that 0.2 percent of the public keys in the web do not offer suitable security.

Our research focuses on the analysis of public key unpredictability offered by a specific PKI, used in a system or network for user authentication. [8] [9]

In this paper, a procedure is detailed to develop an auditing tool for SSL Oracles with public key frameworks and RSA encrypting schemes.

2. Public Key Infrastructure: Overview

The concept of Public Key Infrastructure (PKI) can be traced to the ideas of Whitfield Diffie and Martin Hellman published in 1976 in which users have two types of keys for cryptographic purposes: a public and a private key.

With the public key, messages can be encrypted and then decrypted with the use of only a secret private key. These systems are also called Asymmetric Cryptography to differentiate them of Symmetric Cryptography in which the same key is used for encrypting/decrypting messages.

Finally, the Diffie-Hellman theoretical propositions were implemented through **PKI technology,** helping develop for instance: user authentication, encrypted message transmission using others' public keys, own message encryption/decryption, and digital signature authentication or non-repudiation of transmitted information.

Authentication is the identity confirmation of a person, a computer, a server, or other. Through authentication, identity hacking can be avoided. Similarly, non-repudiation prevents a sender from refusing authorship of a message signed with its key.

PKI implementation may be public or private. They generate *certificates* which are delivered through a system based on the confidence of the certification authority and digital signatures.

This paper focuses ***on the analysis of the implementation of private PKT***

3. PKIs and the RSA System

Diffie and Hellman had the vision of an asymmetric encrypting structure. However, after a strenuous search, they could not deliver a practical rendering of their idea.

A year after publication, in 1977, Ron Rivest, Asi Shamir and Len Adleman from the Massachusetts Institute of Technology (MIT) presented a mathematical model following D&H ideas. The system is known as ***RSA system.***[1] The relevance of their proposal made it probably the most popular system in use.

The basic RSA ciphering system consists of a 3-tuple *(e, d, n)* where *e* is the ***public key***, *d* is the ***private key*** and *n* is the ***module,*** also public.

The source of a message *m* is encrypted by raising it to the power *e* of the receiver and its module *n*, following Gaussian congruencies. After this operation, the encrypted message *c* is obtained and transmitted. The receiver raises *c* to the *d* power module *n*, recovering the original message *m*.

In mathematical terms:

Encryption:

$$c \equiv m^e \ mod\ (n) \qquad \textbf{(1)}$$

Where *m*: message, *e:* public key, and *c:* encrypted message.

Decryption:

$$m \equiv c^d \ mod\ (n) \qquad \textbf{(2)}$$

Where *m*: message, *d*: private key, *c:* encrypted message.

RSA has also a digital signature system based on the same logic but reversing the steps of the process.

Security of these primitive cryptographies rely on the difficulty of finding prime factors of large compound numbers and the discrete logarithm problem, making processing impossible in the short term, at least for the time being.

4. ¿What is a Cryptographic Oracle?

In any system based on the ideas of public key encryption, user authentication, digital signature, and non-repudiation must operate with a sub-system giving cryptographic support, namely a server assigned to those purposes.

[1] after the initials of their last names.

In his 1938 doctorate speech, Türing introduced the paradigm of a super machine or ***Oracle Machine***.

The Oracle machine tackles the problems for which there are no algorithmic solutions. Although not fully applied yet, the concept is analogous to the sub-system dealing with certificate generation and other cryptographic tasks; thus, the term Oracle used in this paper.

No formula is available yet to generate prime numbers in a deterministic fashion. Hence, the Oracle is responsible for finding two sufficiently large prime numbers.

It generates such numbers randomly and runs a primality test. In case the test fails, it searches for new numbers.

If the prime numbers are accepted, the remaining operations are performed; and yield the 3-tuple *(e, d, n)*.

The protocols **Secure Sockets Layer – SSL** and later the **Transport Layer Security TLS** determine the steps required for secure communications, implementing the concept of an Oracle. Open-SSL packets are examples of Oracles.

5. Oracle Vulnerability

If an Oracle does not deliver its assigned tasks, connections between system and users become insecure. A bad performance and incorrect or malicious programming can add vulnerability to the communications intended to protect.

This vulnerability facilitates access to computer crime or intrusions to public or private systems, disturbing legitimate users and dangerously damaging security.

The attacker can clone certificates, hack user identities, access different systems, e-mail accounts, banking and credit card information and use the information to its advantage, among other violations.

Bad programming in the search of prime numbers can turn the list of such values predictable and diminish the quantity of *n* modules generated.

In this framework, the ***Oracle*** does not offer a safe working mode based on an unbiased probability distribution of modules with a large cardinality for the set of prime numbers predicted by Number Theory. Hence, it turns to an ***unsafe mode***, based on a biased value distribution of cryptographic modules.

6. The Debian Case ¿Bug or Viral Code?

The situation described in the previous section is genuine. It has been identified by Luciano Belo, an argentine researcher of the OpenSSL toolkit included in the Debian system.

Duly informed to developers, it was published as *DSA 1571-1* on May 13th, 2008, under the title ***New open ssl packages fix predictable random number generator***.

Its first vulnerable version 0.9.8c-1 was published on September 17th, 2006.

Due to a variable initialization error, the number generator became predictable. Hence, it was vulnerable to brute-force attack over a reduced set of 2^{15} values.

A bug is an ***innocent*** programming error; i.e. an authentic error. However, a viral code is the inclusion of malicious code such that it passes unnoticed during software review.

All systems relying on this Oracle for security had an open vulnerability for about twenty months.

¿Bug or viral code?

Clearly, administrators of Information Security Management Systems cannot allow these types of error, regardless of their origin.

7. Linus´s Law Concerning Software Bug Detection is no Longer Sufficient

Linus Torvalds, creator of the Linux kernel stated*:* ***given a sufficiently large number of eyes, all error becomes obvious***.

The statement addresses the open source software model which reveals programming codes; as opposed to closed source software or software package in which the user has no access to its source code and content.

Clearly, the Debian package was available for review. Nonetheless, it contained a considerable error.

Therefore, a ***sufficiently large number of eyes*** or the increasing complexity of the systems prevents error detection.

8. Anomaly Detection in the Behavior of RSA Oracles

Line by line review of a software code might take too long and require highly specialized personnel.

The ***Debian*** case clearly shows that this methodology fails to detect irregularities and errors.

Defective Oracles might pass successfully stages of development and testing in the production phase and software installation; they can therefore originate weaknesses and vulnerabilities with the damaging consequences mentioned previously.

In laboratory tests, we carried out an analysis of the problem with the objective of finding a mathematical algorithm to detect anomalies in the behavior of Oracles. With that purpose, a working framework was developed to focus the efforts of the research group.

The project is in the coding process, as well as formula and empirical testing, performing the required adjustments. The feasibility of carrying out the proposal is also under study.

The main objective is the development of the final version of an ***auditing tool*** which might enable ***administrators of system security*** to test the performance of their O***racles***.

The detection of anomalies in the distribution of modules may be achieved with statistical analysis of samples of 3-tuples *(e, d, n)* originated by the Oracle.

Detecting a biased distribution can identify bad performance or errors in the Oracle programming. In turn, it might perform a search within the same error for its correction or the substitution of the Oracle with another.

9. Mathematical Methodology for the Anomaly Detection in the Behavior of Oracles

Given that the RSA cryptographic system works based on large prime numbers[2], the Oracle has to be able to obtain a large list of such numbers whose size may be selected by the user.

Let P be the set of all Oracle generated primer numbers of the required order or size:

[2] For example, for systems of 1024 bits, two primes are required, p and q, such that each one is at least 512 bits. That is the case of balanced primes, whose size is recommended for greater system security. However, certificates may even have 2048 and 4096 bits.

$$P = \{p_1 ; p_2 ; p_3 ; \ldots ; p_r\} \text{ where } Card\ (P) = r \quad \textbf{(3)}$$

Considering that the values p_i can be ordered, the value r is equal to the cardinality of P. The number is estimated with the function π(n) of Number Theory.[3]

Let N be the set of all different modules n_i such that they are the product of two prime values of the set P.

$$N = \{n_t = p_i * p_j ; i \neq j; i \leq r; j < r\} \text{ where } \text{Card}\ (N) = \frac{r*(r-1)}{2} \quad \textbf{(4)}$$

The cardinality of N represents the number of certificates that the Oracle is able to generate.

However, it is in this particular point where the weakness of anomalous Oracles lies: the instance in which the cardinality of N is drastically reduced by a bug or a viral code.

A bias – such as in the Debian case – might generate a set P' of prime values such that $P' \subseteq P$.

Let:

$$P' = \{p_1 ; p_2 ; p_3 ; \ldots ; p_{r'}\} \text{ where Card } (P') = r' \text{ such that } r' \leq r \quad \textbf{(5)}$$

P' makes a large number of certificates vulnerable with their security compromised, considering that one of the prime factors is revealed. Hence, the recovery of the private key is immediate.

Let:

$$N' = \{ n'_t = p'_i * p'_j : i \neq j,\ i \leq r';\ j < r' \} \text{ where } Card(N') = r'*(r'-1) \quad \textbf{(6)}$$

All compromised certificates belong to the set N'. Thus, we can define an expression to establish the security of an Oracle:

$$Sec(O) = 1 - \frac{Card(N')}{Card(N)} \quad \textbf{(7)}$$

For an unbiased Oracle, the large value of the cardinality of P, and therefore the cardinality of N, yields a quotient close to 0. On the other hand, the value becomes significant if the Oracle behaves anomalously.

[3] The function indicates that the number of prime numbers between 1 and a number n is $\lim_{n\to\infty} \frac{\pi(n)}{n/\ln} = 1$

For instance, the number of prime numbers of the order 2^{512} is $\pi(2^{512}) \approx \frac{2^{512}}{\ln 2^{512}} - \frac{2^{511}}{\ln 2^{511}} \approx 5.7 * 2^{500}$

We make the following assumptions:

a) The Oracle is programmed to deliver modules of the form $n = p^2$ where p is a prime number. Otherwise, the certificate originated could be broken down with a few calculations.

b) The Oracle keeps some kind of registry of delivered certificates. Hence, it cannot deliver two equal certificates. If it were not the case, the possibility exists of two users holding the same certificate.

10. Probabilistic Formula for Anomaly Detection

The Oracle delivers certificates based on a combination of two generated primes. Hence, certificates can be found that share a prime factor in its module. We shall call certificate collisions when this situation occurs.

If the Oracle works properly, the probability of finding two colliding certificates is significantly low.

Let r be the quantity of primes that an Oracle can find, equal to the value of the function π (x), being x the size of the prime numbers requested by the user. The value is calculated within the theoretical framework and with a normal probability distribution of the generated primes.

Hence, the probability of finding two colliding certificates, having requested one is:

$$\mathrm{Prob}(1) = \frac{2*(r-2)}{\frac{r*(r-1)}{2}} \approx \frac{4}{r} \quad \textbf{(8)}$$

Given that r is expected to be significantly large, the probability gets close to 0. Requesting an appropriate amount of certificates, the probability of colliding primes increases.

An expression has been found relating the size of the sample c with the probability of occurrence h of at least one colliding prime, where r indicates the quantity of primes that the Oracle can generate:

Then:

$$c = \frac{r*(1-\sqrt{1-h})}{2} \quad \textbf{(9)}$$

11. Procedure for Anomaly Detection in Oracles

For the detection of anomalous behavior in Oracles without reviewing the programming code it is necessary to:

1) Analyze the generated prime values through the certificate requests. The Oracle may be authorized to behave in two different ways when facing the auditing tool (or even a possible attack):

a) It can deliver the 5-tuple *(e, d, n, p_1, p_2)* in which, besides the known data, the prime factors are included. Access to prime factors of the module is granted given that the Oracle can show them when delivering the certificate.

b) It can deliver the 3-tuple *(e, d, n)*. In this case the tool may decompose *n* to find the constituting prime factors. Several procedures are used for this purpose. Our laboratory has developed a factorization procedure using the additional information provided by the public and private keys. In fact, we have demonstrated that it is more efficient in computational terms than other algorithms performing similar tasks.

2) With the prime values obtained from 1) the tool constructs the set P'.[4]

3) A probability of finding a certificate collision in a sample is determined and the *c* value is calculated.

4) The occurrence or not of a certificate collision in a given sample does not add information regarding the performance of the Oracle. The test needs to be repeated several times such that it may corroborate that it deviates from expected performance.[5]

12. Conclusions

The paper presents a methodology open for codification to detect anomalies in Oracle performance applied to RSA public key cryptographic systems.

The search for anomalies in the performance of Oracles without code review is a necessary task. The 2008 Debian case demonstrated the Linus' Law may not apply when

[4] In case no algorithms are used for the decomposition of modules, the Greatest Common Divisor can be applied between 2 certificates. If they do not have primes in common, it yields 1. Otherwise, it will indicate the shared prime.

[5] Assuming a prob. 0.5 of getting a colliding prime number, the experience is equivalent to tossing a coin and detecting whether the coin is loaded or not. Possible outcomes are heads (collision) or tails (non-collision). The toss needs to be repeated a significant number of times and then calculate how it deviates from the expected performance.

software codes are complex and few eyes are on the watch. Through certificate analysis, colliding primes are searched, giving an indication of anomalous performance.

13. Further Work

This work will continue with the application of the methodology and later Oracle testing. Upon empirical evidence, adjustments are expected to be made such that the end result shall provide the community with an auditing tool for Oracle performance evaluation.

Acknowledgements

The financial support provided by *Agencia Nacional para la Promoción Científica y Tecnológica* and *CITEDEF* is gratefully acknowledged.

References.

1. Castro Lechtaler, A; Cipriano, M; Benaben, A. y Quiroga, P. Study on the effectiveness and efficiency of an algorithm to factorize N given E and D. Proceedings of 9th Iberoamerican Seminar on Information Technologies Security - 13th International Convention and Fair. ISBN 978 - 959 - 286 - 010 - 0. Pg. 7.604 to 7.609. La Habana. February 2009.
2. Castro Lechtaler, A; Cipriano, M; Benaben, A; Arroyo Arzubi, A y Foti, A. Development, Testing and Performance Evaluation of Factoring. Proceedings of Chilean Computing Week 2009. XXI Encuentro Chileno de Computación (ECC). Pg. 194 to 198. Santiago de Chile, Chile. November 2009.
3. Holz , R; Braun, N, and others. The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements. In Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, IMC 2011, pages 427-444. ACM, 2011.
4. Loebengerger, D; Nüsken, M. Analyzing Standards for RSA Integers. In A. NITAJ and D. Pointcheval, editors. Africacrypt 2011. Volume 6737 of Lecture Notes in Computer Science, pgs. 260-277. ISBN 978-3-642-21968-9, Springer, 2011.
5. http://www.debian.org/security/2008/dsa-1571/, 2008.
6. Moore, H. (http://digitaloffense.net/tools/debian-openssl/). Debian OpenSSL Predictable PRNG Toys. 2008.
7. Lenstra, A; Hughes, J; Augier, M y otros. Ron was wrong, Whit is right. e-print International Association for Cryptologic Research. 15 Feb 2012. http://eprint.iacr.org/2012/064.
8. Castro Lechtaler, A; Cipriano, M. Anomalías en Oráculos Criptográficos tipo RSA por medio de análisis probabilísticas y estadísticos. XIV Workshop de

Investigadores en Ciencias de la Computación. ISBN 978-950-766-082-5. Pg. 40-44. Posadas, Argentina. 2012.

9. Castro Lechtaler, A; Cipriano, M. Detección de anomalías en Oráculos tipo OpenSSL por medio de análisis de probabilidades. XVII Congreso Argentino de Ciencias de la Computación CACIC 2011, WARSO. Isbn 978-950-34-0756 - 1. Pg. 1096-1104. La Plata, Argentina. 2011.