Experimental Detection of Anomalies in Public Key Infrastructure.

Antonio Castro Lechtaler[1,2], Marcelo Cipriano[1,2,3]
Eduardo Malvacio[1]

[1] Escuela Superior Técnica – IESE. Buenos Aires,
[2] Universidad Tecnológica Nacional Facultad Regional Buenos Aires. [3] Instituto Fátima.
[acastro@iese.edu.ar, marcelocipriano@iese.edu.ar, edumalvacio@gmail.com]

Abstract. Cryptographic techniques authenticate users and protect information confidentiality. These tasks are performed by subsystems called Oracles. The most popular Oracle 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 biased number generator created 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. Ten million certificates delivered by a private PKI were analyzed and found significant differences between theoretical predictions and experimental results.

Keywords: asymmetric cryptography, SSL, RSA, prime integers, predictable primes.

1. Introduction

Given the widespread use of network communications and services, public key cryptographic subsystems for user authentication should take measures to prevent system vulnerability anomalies.

The RSA system has shown attack resistance so far. However, weak applications might be present, for example in the SSL protocol [1] or others. The quality of primes might not be appropriate [2] or the generation of such values might be predictable, as it was demonstrated in the year 2008. [3] [4] [5]

Recently, researchers have shown that 0.2 percent of all public keys in the web are not secure. [6]

Our laboratory carries out a line of research [7] analizing the randomness of public keys for a particular PKI installed in a system or network for user authentication.

This article details the experimental results from running a software tool designed to audit SSL Oracles with RSA encryption. [8]

2. An Overview of Public Key Infrastructure

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. [9]

For instance, **PKI technology** develops user authentication, encrypted message transmission using others' public keys, own message encryption/decryption, and digital signature authentication or non-repudiation of transmitted information.

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

This paper focuses on the experimental results from the study of anomalies in a private implementation of a PKI system.

3. PKIs and the RSA System

In 1977, Ron Rivest, Asi Shamir, and Len Adleman from the Massachusetts Institute of Technology (MIT) presented a mathematical model following Diffie-Hellman ideas. The system is known as ***RSA system.***[1] The relevance of their proposal made it probably the most popular system in use. [10]

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.

Security of these primitive cryptographies relies 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.

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*.

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

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.

The Oracle machine tackles the problems for which there are no algorithmic solutions Türing introduced the paradigm of a super machine or ***Oracle Machine*** in 1938 [11]. 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.

---

[1] after their last names.

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

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.

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.

The situation described in the previous section is genuine. It has been identified by Luciano Bello, 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***. [12].

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.

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

Clearly, administrators of Information Security Management Systems cannot allow these types of error, because the system not offer a safe working mode, regardless their origin.

7. Mathematical Methodology for Anomaly Detection in Oracle Behavior

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.

For systems of *b* bits, let *P* be the set of all Oracle generated primer numbers of the required order or size:

$$P = \{p_1 ; p_2 ; p_3 ; \ldots ; p_r\} \text{ where } Card\ (P) = r \qquad \textbf{(1)}$$

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]

Then:

$$r = \Pi(2^b) - \Pi(2^{b-1}) \qquad \textbf{(2)}$$

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*.

$$N = \{n_t = p_i * p_j ; i \neq j ; i \leq r ; j < r\} \text{ where } \mathrm{Card}(N) = \frac{r*(r-1)}{2} \qquad \textbf{(3)}$$

The cardinality of *N* represents the number of certificates that the Oracle is able to generate for a key size of b bits.

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'*$\subset$ *P*.

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.

Given the size and quantity of such primes, it is not possible to recover the set *P* to detect anomalies or biases. Hence, a sample methodology is used in which each sample is formed of certificate requests to the Oracle.

Let *M* be a *sample* of *s* certificate requests to the Oracle:

---

[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.

[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 $\pi(2^{512}) \approx \frac{2^{512}}{\ln 2^{512}} - \frac{2^{511}}{\ln 2^{511}} \approx 5.7 * 2^{500}$

$$M = \{m_i / 1 \le i \le s\} \quad \textbf{(4)}$$

We shall call *collision of primes* when the presence of at least one prime factor is shared by two or more certificates in a sample M. That is:

$$m_i, m_j \in M; greatcommondivisor(m_i : m_j) \neq 1 \quad \textbf{(5)}$$

Let *m** be the quantity of samples *M* without collision of primes. Let *m* be the quantity of samples M. The probability of not finding collision of primes for *s requests* will be close to:

$$\frac{\mathrm{m}^*}{\mathrm{m}} \approx \mathrm{prob}(\mathrm{gcd}(\mathrm{M}) = 1) \quad \textbf{(6)}$$

Let *s* requests be an Oracle. Each request is formed by a certificate which has a module *n* formed by the product of two primes. The first request *s1* can yield from *k(k-1)/2* modules. In order not to collide with s1, the second request *s2* needs to be obtained from all the modules which do not share primes with *s1*. In [8] and [13] we show a formula indicanting the quantity of modules not sharing primes with each other.

Developing the following array for each request, it yields:

$s_1 = p_1 p_2$				
$p_1 p_3$	$p_2 p_3$			
$p_1 p_4$	$p_2 p_4$	$s_2 = p_3 p_4$		
...	...	...	...	
$p_1 p_r$	$p_2 p_r$	$p_3 p_r$	...	$P_{r-1} p_r$

For *s=1* the probability of not finding collision of primes in a sample is 1 given that it is the only request.

For *s=2*, the probability of not finding collisions is such that the new certificate does not belong to the set of modules that share primes with the previous certificate.

Then,

$$prob\ [\mathrm{gcd}(\ M\ ) = 1] = \frac{\frac{k*(k-1)}{2} * \left[\frac{k*(k-1)}{2} - [2*(k-1)+1]\right]}{\frac{k*(k-1)}{2}} \quad \textbf{(7)}$$

For h requests in each sample, the following expression is obtained:

$$prob\ [\mathrm{gcd}(\ M\ ) = 1] = \prod_{i=0}^{h-1} \left[1 - \frac{i*[2*(k-i)+1]}{\frac{k*(k-1)}{2}}\right] \quad \textbf{(8)}$$

The value of *h* -- i.e. the quantity of requests per sample -- must be less than the value of the least integer of *k/2*. If h is equal or greater than such value, the probability of not finding collisions is zero.

8. Experimental Design and Evidence for Anomaly Detection in the Behavior of Oracles

8.1 Hypothesis and Assumptions

Hypothesis: It is possible to detect the anomalous behavior of an Oracle by means of probability analysis by not finding collisions between primes in the certificates of delivery.

The following assumptions are made:

Assumption 1: Given a key size of b bits, an Oracle is capable of delivering certificates using primes with size b-1 bits.

Assumption 2: Search and selection of prime values is a random process (or at least pseudorandom) in such a way that we assume equiprobability for each value in the size of b-1 bits.

Assumption 3: The Oracle does not "remember" the certificates previously issued. Thus, it may repeat primes in the generation of new certificates, but not in the same certificate. This assumption also contemplates the possibility of repeating a certificate already issued.

Assumption 4: Throughout sample selection, the probability of not finding collisions fits from the experimental to the theoretical model.

Assumption 5: Given the previous assumptions, if an Oracle does not fit all the criteria, we will consider that research should be necessary to develop new assumptions not contemplated or that an Oracle is defective and presents an anomalous behavior.

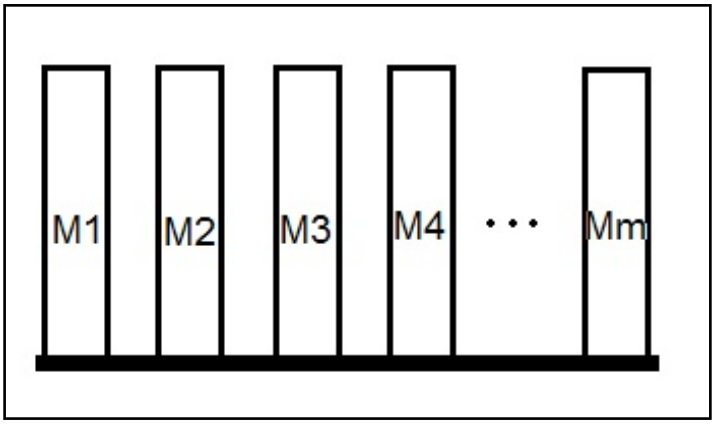
8.2 Experimental Design

We set up a PC with the latest version of OpenSSL for Windows, downloaded from the internet [14].

We define the set M consisting of s certificates requested to OpenSSL through software developed in our laboratory. This tool tests for collision of primes in *M*. The output is the quantity of found collisions, in case there are any.

Repeating the process m times and accumulating in *m** the quantity of sets *M* in which collisions were found, the quotient in formula (6) is calculated which we shall call the experimental probability of not finding collision of primes.

Such that:



We compare the experimental probability with the theoretical probability of not finding collision of primes in m samples of s requests each.

9. Experimental Evidence

With the experimental design and the epistemological background, the hypothesis and assumptions, the software tool developed, and the sample selection, the experimental tests were carried out.

We processed 10.000 (10^4) samples (sets *M*) with 1.000 (10^3) certificate requests each. In other words, we searched for collisions in 10.000.000 (10^7). For practical processing purposes, we requested OpenSSL 64 bit certificates; i.e. the size of the prime factors of each module is b=32 bits.

The theoretical model predicts, following formula 8 with these parameters:

$$prob\ \left[\gcd(\ M\ ) = 1\right] = \prod_{i=0}^{999}\left[1 - \frac{i * \left[2 * (k - i) + 1\right]}{\frac{k * (k-1)}{2}}\right] \qquad \textbf{(9)}$$

where k is:

$$k = \frac{2^{32}}{\ln(2^{32})} - \frac{2^{31}}{\ln(2^{31})} \qquad \textbf{(10)}$$

Thus, in (9) the theoretical probability is 0.979

However, the experimental results are shown below. Each lot contains 1.000 sets M; each set holds 1.000 certificates. The column *Collisions* details the quantity of sets M for which collision does not occur; i.e. m*.

Lot	Collision
1	349
2	326
3	356
4	344
5	334
6	326
7	316
8	325
9	332
10	341

The probability from experimental results yields 0,3349±0,01008 and not 0.979, the theoretical probability.

10. Conclusions

The theoretical model differs significantly form the experimental model. Analyzing the theoretical framework of the model, we shall conclude:

a) *the Oracle is biased and shows anomalous behavior.* This conclusion may indicate that our hypothesis was correct. However, before accepting it, the other possible conclusion needs to be refuted, at least at this stage of the research.

b) *at least one hidden undetected cause is present adding a variable to the theoretical model, originating the difference between the predicted and the experimental results.*

12. Further Work

Before developing software to control the normality or anomaly of the behavior of Oracles of the type SSL, the strength of the primes in the certificates needs to be explored. It could be the case that the Oracle is delivering strong primes;[4] i.e. primes meeting additional conditions other than primality, and hence, the set of certificates which may be delivered is reduced and yields the observed difference. [15]

We do not have the precise quantity of strong primes in a particular interval. The research could lead to work on a heuristic formula for such purpose.

Thus, further work shall focus on this area and on repeating the tests detailed here. In this way, formula refinements can be developed to offer definite criteria to determine normal or anomalous behavior of an Oracle. Once the hypothesis is verified, research efforts shall concentrate in the development of software for *Oracle auditing.*

[4] p is large; p-1=aq+1 where q is a large prime; q-1=bt+1 t is also a large prime; p+1=cw-1 where w is a large prime.

13. Acknowledgements.

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

14. References.

1. Holz, R; Braun, N, and others. The SSL landscape: a thorough análisis 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.
2. Loebenberger, 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. Springer, 2011.
3. http://www.debian.org/security/2008/dsa-1571/, 2008.
4. Moore, H. Debian OpenSSL Predictible PRNG Toys. See http://digitaloffense.net/tools/debian-openssl/, 2008.
5. http://www.citedef.gob.ar/si6/descargas/openssl-debian-defcon16.pdf (accesed 18 july 2012).
6. Lenstra, A; Hughes, J; Augier, M y otros. Ron was wrong, Whit is right. e-print International Association for Cryptologic Research, http://eprint.iacr.org/2012/064, 15 Feb 2012.
7. Castro Lechtaler, A; Cipriano, M. Detección de Anomalías en Oráculos Criptográficos tipo RSA por medio de análisis probabilísticas y estadísticos. XIV Workshop de Investigadores en Ciencias de la Computación. Pg. 40-44. Posadas, Argentina. 2012.
8. 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, Pg.1096-1104. La Plata, Argentina. 2011.
9. Rivest, R.; A. Shamir; L. Adleman (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM21.
10. Bellare, Mihir; Rogaway, Phillip; Introduction to Modern Cryptography, Lecture Notes, University of California San Diego, 2006.
11. Turing, Allan; "Systems of logic defined by ordinals", Proc. Lond. Math. Soc., Ser. 2, 45: 161–228; This was Turing's Ph.D. thesis, Princeton University (1938).
12. http://lists.debian.org/debian-security-announce/2008/msg00152.html (accesed 18 july 2012).
13. Castro Lechtaler, A; Cipriano, M; Malvacio, E; Cañón, S; Procedure for the Detection of Anomalies in Public Key Infrastructure (RSA Systems). 41° Jornadas Argentinas de Informática (JAIIO), (Congress proceedings in the press) La Plata, Argentina.
14. http://www.openssl.org/ (accesed 18 july 2012).
15. Rivest, R; Silverman, R; Are 'Strong' Primes Needed for RSA?, Cryptology ePrint Archive: Report 2001/007. http://eprint.iacr.org/2001/007