Anomaly Search in a Public Key Infrastructure OPENSSL v1.0 1e.

Antonio Castro Lechtaler; Marcelo Cipriano; Eduardo Malvacio

[1] Research Laboratory in Cryptographic Techniques and Information Security
Escuela Superior Técnica / Facultad de Ingeniería del Ejército - IESE, Buenos Aires, Argentina.

{acastro, marcelocipriano}@iese.edu.ar; edumalvacio@gmail.com

Abstract. Public key infrastructure is used in most networks requiring user authentication, digital signature, cryptographic confidentiality, and non repudiation of information. The main protocols and architectures used in a PKI are the public key cryptosystem and the RSA digital signature. Their robustness is based on the difficulty to obtain prime factors of a large composite number. Notwithstanding, if the primes generated by a PKI are predictable or represent a small set of possible primes, the security of the PKI certificate is compromised. This article is part of a line of research aiming to develop auditing tools of PKI performance and to detect anomalies in the generation and distribution of prime values delivered by OpenSSL, if such anomalies are present.

Keywords: Asymmetric Cryptography, PKI, OpenSSL, RSA, integer 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].

The way a Public Key Infrastructure (PKI) can be injured if the rate of certificated collisions emited is higher than the mathematical expected rate [4]. Also, a methotology for the detection of bad function is an Open-SSL system is presented by means for a formula [5].

This investigation tries to show our experience on searching collisions of prime numbers in the RSA module. Among more than 40.000.000 certificates from a PKI.

In this way, we compare the given values though our experience. With the predictions from mathmatical model. In case they do not, we will propose new formulaes or we should go on checking those formulaes.

Investigations regarding this specific issue up to now has shown no antecedent, since our work has been restricted to an only PKI.

Lenstra and others researches [10] have found prime collisions, but in many PKI, stating 0.2% collisions when the spectatives are 0%. In millions of PKI´s users this ratio is significative.

This paper is a "work in progress" that means it is a parcial investigation through a line that has been going on for the last two years in our laboratory [3]. As we show in our previous presentation in the simposiums and congress. In JAIIO 2012 we had detailed a procedure to develop an auditing tool for SSL Oracles with public key frameworks and RSA encrypting schemes [6].

2 Public Key Infrastructure: Overview

The concept of Public Key Infrastructure (PKI) can be traced back 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.

PKI technology constitutes a valuable resource to develop, for instance, user authentication, encrypted message transmission using other user public keys, own message encryption/decryption, and digital signature authentication or non-repudiation of transmitted information [9].

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

3 RSA System

In 1977, Ron Rivest, Asi Shamir, and Len Adleman from the Massachusetts Institute of Technology (MIT) presented a mathematical model following the Diffie-Hellman ideas. The system is known as ***RSA system.***[1] Its widespread applicability granted its popularity.

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 finding prime factors of large composite numbers, and on the discrete logarithm problem, making processing impossible in the short run, at least with current calculation tools.

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*.

[1] After their last names.

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

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.

N is the product of two prime numbers. PKI searches such primes through trial and error.[2] It generates a number of an appropriate size. It runs a primality test and if the number fails the test, PKI generates a new one. But if the number passes the rest, it generates a second one to obtain two "primes." [3]

The product of these primes, *p* and *q*, determine *N*; i.e.:

$$N=p*q \qquad \textbf{(3)}$$

In addition, *d* (the private key) is calculated as a function of *e* (the public key): [4]

$$e*d \equiv 1 \ mod\ (\ (p-1)*(q-1)\ ) \qquad \textbf{(4)}$$

Where

$$(p-1)*(q-1) = \varphi(N) \qquad \textbf{(5)}$$

This expression is known as Euler´s totient function for N.

Here lies the robustness of the system: finding *d* (the secret private key which opens the message) requires solving equation (4) which, in turn, involves solving formula (5). With *p* and *q* unknown, there is no expeditious method to find prime factors of a sufficiently large *N*. Thus, the system is robust [11].

---

[2] No algorithm is yet known to generate prime numbers; hence, the trial and error approach.

[3] The fastest primality tests used in a PKI are stochastic processes which consider a given number to be a *probable prime* if it passes the test a quantity *v* of times.; for example, the Rabin-Miller test. Because of the calculations involved, a deterministic primality test would prove impractical.

[4] By convention, $e = 65537 = 2^{16}+1$. $e = 3$ has also been used, but this low value can generate vulnerabilities.

Secure Sockets Layer – SSL protocols**,** and later *Transport Layer Security TLS***,** have modules to search for *p*, *q,* and *d*, to calculate *e*, and to issue the resulting certificate (*N, e, d*).

4 SSL and TLS Vulnerability: The Debian Case

The system keeps *p* and *q* hidden behind *N*. Vulnerabilities could arise from programming errors in the search modules for *p* and *q*. For example, the system could generate a reduced set of primes.

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

Attackers can clone certificates, hack user identities, access systems, e-mail accounts, and banking or credit card information; thus, they can use the information to their advantage, among other violations.

Bad programming can also turn the list of prime numbers predictable and therefore decrease the quantity of *N* modules generated.

In this framework, OpenSSL 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 distribution of cryptographic modules.

The situation described in the previous section is genuine. It was identified by Luciano Bello, an argentine researcher of the OpenSSL toolkit included in the Debian system. Its first vulnerable version 0.9.8c-1 was published on September 17th, 2006.

Due to a variable initialization error, the number generator turned predictable. Hence, it became vulnerable to brute-force attack over a reduced set of 215 values.

After notifying developers, a report was published as DSA 1571-1on May 13th, 2008, under the title *New OpenSSL packages fix predictable random number generator* [7, 12]

A bug is an unintentional programming error; i.e. an authentic error. However, a viral code is the inclusion of malicious code to pass 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 errors, regardless of their origin.

In this years has been detected more vulnerabilities in Debian´s Open-SSL and come more [8, 13, 14].

5 Anomaly Detection in the Performance of OpenSSL: Toy Model.

This methodology uses statistical tools to detect anomalies in the performance of a PKI. Considering that the size of real certificates is greater than 1024 bits, formulae

adjustment and corrections become difficult due to the high magnitudes and great quantity of numbers.[5]

Hence, the complexity of the problem is reduced by using the smallest certificates accepted by OpenSSL, i.e. 64 bits.

Methodologically, this reduction is knows as a Toy Model. The formula links the sample size c of requested certificates with the probability h of occurrence of at least one collision and with r indicating the quantity of primes of a given size that OpenSSL can generate.

Then,

$$c = \frac{r * (1 - \sqrt{1 - h})}{2} \qquad \textbf{(6)}$$

6 Experimental Procedure for Prime Collision Search in OpenSSL.

6.1 Analisys.

Tests were carried out to search for shared primes, also known as prime collision.

A PKI was installed and implemented for this purpose in an OpenSSL, version XX in a lab computer. The program was set to issue 64-bit certificates. C++ software was designed to request certificates and to store them in a vector of prime factors for each *N*. PKI was requested to deliver each certificate in a 5-tuple of the type (*N, e, d, p, q*) to reduce calculation load.

Whenever this is not possible, the scheme can be replaced with the factorization of *N*, given *e* and *d* – an algorithm developed previously in this line of research. [1, 2, 8].

Given the probabilistic nature of these primes, the tool searches for repeated values of primes in the issued certificates in samples of size *c*, and obtains the value of *h*, as given by formula (6) where *h* is the probability to find at least one shared prime in the sample.

With the results from all samples, the value of *h* is calculated; i.e., the empirical probability of finding at least one repeated prime or prime collision en *c* certificates.

6.2 Experimental Difficulties: Rise in Complexity.

The software stores a table with the values of primes *p* and *q* of each issued certificate. It checks whether a certificate has already been issued by the PKI. If that is the case, it notes a collision, following the procedure up to a total of *c* requests.

As the table size increases, the complexity of the problem also increases, in time as well as in space.

A significant increase in RAM memory use was observed. The computer can process up to 2000 samples of 1000 certificates each; i.e., a total of 2 million 64-bit certificates.

5 For instance, the number of primes in an *N* of 1024 bits is $5{,}7*2^{500}$ while for 64 bits is $1.4*2^{31}$.

Whenever a larger set of samples were processed, a memory overflow error occurred, abruptly stopping the process. The reason for this error has yet to be determined. Assessments are being carried out to identify the cause of the alleged anomaly.

However, it has been determined that when processing for each sample is complete, the variables reset and there should be no apparent reason for RAM memory to halt the process due to saturation.

6.3 Experimental Results.

Despite the previous discussion, the real time of running the experiment was significantly greater than expected. Notwithstanding, 40 million 64-bit certificates were analyzed grouped in samples of 1000 certificates each. In other words, 80 million primes were processed in search for collisions, in groups of 2000.

The experimental value of h was obtained, the probability of finding at least one repeated prime grouped in 1000 certificates of 2 different primes each.

The obtained value of h is 0.66128571. Hence, at least 661 certificates contain repeated primes for each 1000 requests of the 64-bit size.

These results are significantly higher than expected, according to formula (6). Considering c equals 1000 and k = 1.4*231 primes of 32-bit size, the probability of collision h is 0.0000012910868 – a much lower value tan the results obtained.

7 Conclusions and Future Work

Analysis suggests that collisions may have originated from requesting a large quantity of certificates within a small lapse between one request and the next.

Further considerations aim to determine whether the distribution of prime collisions suggests some pattern evading the random process for selection of primes.

In the case that the distribution of collisions does not present anomalies, this factor shall be disregarded.

Another possibility is that some mechanism in the generation of primes within OpenSSL did not react with sufficient margin; thus, it did not give the generator “seed” enough time to change, yielding the same prime [5,6].

Considering that OpenSSL is an open source tool, access to codes is granted. Search within the code for the prime generation processes is an arduous endeavor. In this manner, Luciano Bello detected the bug in the Debian case, as mentioned in section 4.

However, this type of search is what the present proposal is aiming to avoid by developing an auditing tool of the performance of such processes: to avoid code reviews in account of the complexity of their analysis.

In addition, the tool could audit the performance of SSL and TSL protocol applications, even when they belong to commercial or boxed software, whose owners do not reveal the programming codes.

The entire experimental design is undergoing full review, before concluding that the OpenSSL could be malfunctioning [9].

Thus, the following elements are currently under evaluation: Formulae used in the calculations of the model, Software development and Open SSL installation.

Furthermore, experimental trials are being conducted again, increasing the number of repetitions.

8 Acknowledgements

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

9 References

1. Castro Lechtaler, A; Cipriano, M; Benaben, A. and 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. (2009): ISBN 978 - 959 - 286 - 010 – 0. 7604 – 7609.
2. Castro Lechtaler, A; Cipriano, M; Benaben, A; Arroyo Arzubi, A and Foti, A.: Development, Testing and Performance Evaluation of Factoring. Proceedings of Chilean Computing Week 2009. XXI Encuentro Chileno de Computación. (2009). 194 - 198.
3. Castro Lechtaler, A; Cipriano, M.: Detección de anomalías en oráculos criptográficos tipo RSA por medio de análisis probabilísticos y estadísticos. XIV Workshop de Investigadores en Ciencias de la Computación. (2012). ISBN: 978-950-766-082-5. 40 - 44.
4. 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. (2011). ISBN: 978-950-34-0756-1. 1096 – 1104.
5. Castro Lechtaler, A; Cipriano, M; Malvacio, E.: Experimental Detection of Anomalies in Public Key Infrastructure. XVIII Congreso Argentino de Ciencias de la Computación. (2012). ISBN: 978-987-1648-34-4. 1646 – 1655.
6. Castro Lechtaler, A; Cipriano, M; Malvacio, E; Cañón, S.: Procedure for the Detection of Anomalies in Public Key Infrastructure (RSA Systems). 13th Argentine Symposium of Technology - AST 2012, 41 JAIIO. (2012). ISSN: 1850 - 2806. 288 - 298.
7. Debian, The Universal Operating System. Información sobre seguridad. (http://www.debian.org/security/2008/dsa-1571/).
8. Debian, The Universal Operating System. Información sobre seguridad. (http://www.debian.org/security/#DSAS).
9. Holz, R; Braun, N, et al.: 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. 427 - 444. ACM, (2011).
10 Lenstra, A; Hughes, J; Augier, M et al.: Ron was wrong, Whit is right. e-print International Association for Cryptologic Research, (http://eprint.iacr.org/2012/064)
11. Loebenberger, D; Nüsken, M.: Analyzing Standards for RSA Integers. In A. NITAJ and D. Pointcheval, eds. Africacrypt . V.6737 of Lecture Notes in Computer Science, 260 - 277. Springer (2011).

12. Moore, H. Debian OpenSSL Predictable PRNG Toys. (http://digitaloffense.net/tools/debian-openssl/) .
13. National Vulnerability Database. (http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-2110).
14. The OpenSSL: cryptograpy and SSL / TLS toolkit: (http://www.openssl.org/news/secadv_20120419.txt)