Information & Security

I. Blagoev, T. Balabanov & I. Iliev
vol. 50, no. 2 (2021): 171-179
https://doi.org/10.11610/isij.5028

Published since 1998

ISSN 0861-5160 (print), ISSN 1314-2119 (online)

Research Article

RSA Weaknesses Caused by the Specifics of Random Number Generation

Ivan Blagoev (✉), *Todor Balabanov*, *Iliyan Iliev*

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. Georgi Bonchev Str., block 2, 1113 Sofia, Bulgaria, https://iict.bas.bg/en/index.html

ABSTRACT:

The rapid entry in digital transformation and Covid-19 moved many activities to the Internet. The application of cybersecurity tools gives a sense of good security condition of the used digital services. This is often how things look on the surface, but the problems sometimes is hard to notice. The current study presents weaknesses in the still widely used cryptographic algorithm RSA, which allows RSA cryptographic keys to be compromised. It demonstrates the connection with RNG as the root of all the resulting controversies around the issues under consideration.

ARTICLE INFO:

RECEIVED: 28 JULY 2021
REVISED: 07 SEP 2021
ONLINE: 18 SEP 2021

KEYWORDS:

cryptography, RSA algorithm, random numbers generation, vulnerabilities, cybersecurity

Introduction

The faster development of modern societies leads to greater digitalization. More activities and processes are much more productive and effectively managed by the incorporation of new technologies. All these processes even did accelerate and did prove their value when the world was affected by the global pandemic of Covid-19. A transformation that would have taken years had to happen within months. The societies were pushed to search for different living styles much more connected with the technologies. At first glance, it looks that the world is prepared for such a technology challenge, and in general, it is like

✉ Corresponding Author: E-mail: ivan.blagoev@iict.bas.bg

this. At the same time, the number of cybercrimes did rise. The encroachment on personal data, encroachment on personal money, and loss of information, extortion due to loss of information also escalated to unprecedented levels. All of this is a strong indicator that while technology and computing infrastructure have met the challenge, we are not ready for strong cybersecurity.[1,2,3]

Compliance with cybersecurity requirements is a prerequisite for the security and safety of IT infrastructures, digital resources, and personal data protection. In this respect, the topics of cryptography and the sufficiently reliable generation of random numbers that underlie any encryption system are of particular interest.[4]

For modern cryptography needs, two types of random number generators are used – a true random number generator (TRNG) and a pseudo-random number generator (PRNG).[5]

A True Random Number Generator (TRNG) is applied when the RNG needs to generate values at a given time that should be unique and should not be repeated in subsequent RNG calls.[6,7] The numbers obtained with this type of RNG are applied to operations that require unique / non-repeating numerical values generated over time.[8,9] An example of such a situation is the generation of a cryptographic key for encoding/decoding data, initialization vectors, initial numerical values (seed) for controlled RNGs, etc.[10,11]

Pseudo-Random Number Generator (PRNG): An initial random number from the micro or macro world (seed) is used as the basis for this generator, and a mathematical formula is used for subsequent numbers. From the initial value, by application of a particular algorithm, all random numbers generated subsequently originate. Subsequent values, in their order, are reproducible. The only unexpected and secret value that should be as unpredictable as possible is the initial number, which is the "root" at the base of this sequence and initiates the generation of the entire numeric sequence. From this technology are borrowed the authentication with One Time Password (OTP), the generation of cryptographic keys derived from the Master Root Key (applied in the compilation of portfolios in BlockChain – distributed ledger technology), authentication via HMAC, and others.

Traditional RNG security measures are mostly generalized statistics related to deviations from mathematical randomness.[12]

The hardware random number generator (HRNG)[13] or more True Random Number Generator (TRNG) is a device that generates random numbers from a physical process, not through an algorithm. This type of generator is radically different from those discussed so far because such devices are often based on micro-world phenomena that generate low-level, statistically random "noise" signals, such as thermal noise, photoelectric effect including beam splitter and other quantum phenomena. These stochastic processes are considered completely unpredictable in theory, unlike the paradigm for generating pseudo-random numbers, often applied in computer programs. In general, two main sources of practical quantum-mechanical physical probabilities are known: quantum mechanics at the atomic or subatomic level and thermal noise (some

of which are of a quantum-mechanical origin). Quantum mechanics states that some physical phenomena, such as the nuclear decay of atoms, are fundamentally random and generally unpredictable.

Since the result of quantum mechanical events cannot be predicted, they are considered as a "gold standard" for generating random numbers. In fact, one of the best random number generators for server systems is considered to be the photon-type quantum generator. Such generators are compact enough and can fit on a PCB, while at the same time, they have a very high performance rate. In many cases, such a hardware module has the capacity to power more than one public service server with quality random numbers.

Regardless of which of the random number generators is applied (uncontrolled or controlled), the overall success of the system depends on the statistical qualities of the produced numbers.[14] The rapidly growing demand for frequency bands, increasing volumes of stored data, and performing calculations, combined with the growing spectrum of cyber threats, ensure that our need for reliable and unpredictable random numbers will only grow in the future.[15]

The essence of RSA encryption is that it uses only information that is publicly available. With the public key, anyone can encrypt a message they want to send to the owner of the private key. This is possible because without knowing the values of p and q, no one but the owner of the private key can decode the message. Although everyone knows the public key $x = p * q$, this does not give them any effective way to find values for p or q. According to a group of researchers years ago, it was thought that even the discovery of a 232-digit number would take more than 1,500 years of computational time (distributed among hundreds of computers) to compromise such a private key.

On the surface, RSA encryption appears invulnerable. It could be said so far, with the exception of one small problem, almost everyone uses the same random number generators. An excellent source of entropy is needed to generate the high-quality prime numbers that make up the cryptographic keys in RSA. In conventional computer systems, sources of quality entropy are relatively scarce for such a task. For this reason, seeds derived from quality entropy have been widely used for years. The calculations for the new RSA keys are then performed through pseudo-random number generators.

Taking into account these facts, we can turn to a study of recent years, according to which a new idea is emerging, looking again at the well-known example: Let suppose that Bob and Alice publish public keys online. Because they both used the same program to generate random prime numbers, their public keys are more likely to have a common prime factor. Factoring Bob or Alice's public keys separately would be almost impossible but finding common factors between them is much easier. In fact, the time required to calculate the greatest common divisor between two numbers is close to proportional to the number of digits in the two numbers. Once the common divisor between Bob and Alice's keys has been identified, it can be invoiced to obtain the basic factorization of the two keys. From this point of view, it is possible to decode any messages sent to Bob or Alice.

Armed with this idea, researchers scanned the Internet and began collecting public keys from the algorithm. For this purpose, they collected 6.2 million real public keys. They then calculated the greatest common divisor between key pairs, compromising a key each time it shared a common factor with other keys. In this experiment, they managed to break 12,934 RSA keys. In other words, if the technology is used carelessly and the described weaknesses are not overcome, RSA encryption provides less than 99.8 % security.

At first glance, this seems like the whole story. Reading the research on the subject [16] more closely reveals something more disturbing. According to the authors, they were able to perform the entire calculation in a few hours on a single CPU machine. Viewed through the theoretical foundation of RSA, it should be assumed that it will take years to calculate the GCD (greatest common divisor) between 36 trillion key pairs, rather than hours, according to the study.

How did they do it? The authors hint in a footnote that their calculation is based on an asymptotically fast algorithm that allows them to reduce the time to perform the calculations to almost linear. The actual description of the algorithm is kept secret from the reader, perhaps to prevent malicious use. Just a few months after the article was published, subsequent reports have already discussed in detail various approaches presenting fast algorithms (such as the study on quasi-linear GCD computation and factoring RSA modules and even showing how to use GPUs to do the calculation with a rough force faster).[17]

It is possible to state here that it is not good to disclose information if it is to remain secret. On the other hand, if the weaknesses in cryptographic functions are not highlighted, we run the risk of being used by malicious individuals without the knowledge of others. In this case, in order to arrive at the results of the research, we must turn to the algorithms.

Given the characteristics of cryptography and the proposed approach, the algorithm will deal with integers having an asymptotically large number of digits. Therefore, addition and multiplication will not be considered as fixed and relative time operations.

For n-bit numbers, take $O(n)$ time. Using a multiplication operation, multiplication seems to take $O(n^2)$ time. However, it turns out that there is an algorithm (Schönhage – Strassen algorithm) that works in time $O(nlog2nloglogn)$.

Calculating the GCD using the Euclidean algorithm takes $O(n2lognloglogn)$ time. Once again, however, researchers have found a better algorithm that works in time $O(nlog2nloglogn)$. Fortunately, all of these algorithms have already been implemented in GMP (GNU MP Subquadratic), the C ++ library for working with large numbers. For the rest of the study, we will use the $\tilde{O}$ notation, a variant of the Big-O notation that ignores logarithmic factors. For example, while the calculation of GCD takes time $O(nlog2nloglogn)$, in the notation we write that it takes time $\tilde{O}(n)$.

Problem Transformation

Define the set of public RSA keys with $k_1, \ldots, k_n$, where each key is the product of two large prime numbers. Note that n is the total number of keys. Instead of calculating the GCD of each key pair, we can calculate for each key k_i GCD of it and the product of all other keys $\prod_{t=1} K_t$. If the key k_i shares one main factor with other keys, then this will give the main factor. However, if both main factors of k_i are shared with other keys, the calculation will not be able to actually extract the individual primary factors. This case may be rare enough and not worth paying much attention to.

Algorithm:

The algorithm has a slightly unusual recursive structure, as recursion occurs in the middle of the algorithm, not at the end.

At the beginning of the algorithm, all we have are the keys: $k_1, k_2, k_3, \ldots$

The first step of the algorithm is to connect the keys and calculate their results: $j_1 = k_1 K_2, j_2 = k_3 K_4, j_3 = k5_K 6, \ldots$

Then in recursion on the sequence of numbers $j_1, \ldots, j_n$ is calculated: $r_1 = GCD(j_1, \prod_{t<>1} j_t), r_2 = GCD(j_2, \prod_{t<>2} j_t), r_3 = GCD(j_3, \prod_{t<>3} j_t), \ldots$

The goal is to calculate $s_i = GCD(k_i, \prod_{t<>i} k_t)$ for each k_j key. The important thing here is that when i is odd, s_i can be expressed as $s_i = GCD(k_i, r_{(i+1)/2} k_{i+1})$ and that when i is even, s_i can be expressed as $s_i = GCD(k_i, r_{i/2} k_{i-1})$.

To understand why this is so, one can check whether the expression on the right side of GCD is guaranteed to be a multiple of $s_i = GCD(k_i, \prod_{t<>i} k_t)$, while also being a divisor of $\prod_{t<>i} k_t$. This, in turn, suggests that the GCD calculation will be exactly $GCD(k_i, \prod_{t<>i} k_t)$ as expected.

Execution time:

Let m denote the total number of bits needed to write $k_1, \ldots, k_n$. Each time the algorithm is repeated, it is ensured that the total number of bits in the recursion entry is not higher than the previous recursion level. This is because new entries are products of pairs of elements from old ones.

Therefore, each of the levels of $O(logn)$ of recursion acts on input with a total size of $O(m)$ bits. In addition, the arithmetic operations in each recursion level take the most time $\tilde{\mathcal{O}}(m)$. Thus, the total operating time of the algorithm is also time $\tilde{\mathcal{O}}(m)$ (since the recursion levels $O(logn)$ can be learned in the notation O-tilde).

If we expand the working time in standard Big-O notation, we get $O(mlog3mloglogm)$.

Is the approach practical?

At first glance, the triple logarithmic factor may seem to preclude the use of this algorithm. But in another study, it turns out that this presentation is quite reasonable. Cloostermans found that the algorithm takes approximately 7.65 seconds per thousand keys,[18] which means that it will take just over 13 hours to execute 6.2 million keys.

It also turns out that one of the LOG factors can be eliminated using another approach that avoids GCD calculations altogether, except at the first level of recursion, as evidenced, for example, by Heninger et al.[19] This improved algorithm takes about 4.5 seconds per thousand keys, resulting in a total run time of about 7.5 hours to work with 6.2 million keys. This, the calculation, which should take years, comes down to hours. All that is needed is the application of recursion, time series analysis to exploit the weakness in generating random numbers in systems.

In conclusion, it can be said that the weaknesses do not stem from an error in the arithmetic of RSA. They come from the technological weakness with which RSA is implemented. Computer systems, if they are of a newer generation, have hardware and software improvements that allow them to generate quality random numbers. However, the danger of this vulnerability remains. Therefore, RSA needs really big random numbers. The current criteria for a reliable RSA key is a minimum of 2048 bits, and the recommended length is even 4096 bits. Other studies have also found that between 4096, 8192, and 16384 bits of an RSA key, the greater security of larger keys is minimal. The reason also comes from the limitations of random number generators. Larger RSA switches require extremely large real random numbers, which are extremely difficult to obtain in a computer system. Even if the silicon module for HwRng is used for this purpose, the entropy buffer is 4096 bits, and it accumulates slowly, with the limitations coming from the technology. When using RSA cryptography in systems with significantly less hardware such as IoT, the generation of RSA keys will be even weaker. Again, the reasons are the same, and this type of device often does not have specialized hardware to enrich the entropy. For this reason, many such devices often become easy victims in cyber attacks.

Thus, at the heart of any encryption system is an algorithm and a generator for random numbers. Therefore, no matter how complex encryption algorithms are applied, they are considered to be as vulnerable as the random number generator that underlies this system.

The efficiency of RNG is measured by the degree of entropy to generate random numbers.

The complexity of analyzing a given random number generator is a function of the quality of its entropy, seasonality, and tendency to collide. These are the moments when the random number generator will generate a value that is a cyclic or value field, which leads to the repetition or generation of a new but expected value. Through time series mathematics, it is possible to determine the entropy over time, and it is likely to calculate (predict) the possible future reappearance of the data. The detection of seasonality in the obtained values, deviations, or collisions may also indicate weaknesses of the random number

generator. If the generator is of good quality, then it will follow the analysis of a very large number of statistical values from a numerical array generated with a high degree of entropy and unpredictability, which will be very resource-intensive and complex. This will also make it very attack-resistant throughout the cryptography associated with this generator.

Conclusions

The study of the presented weaknesses in the asymmetric RSA algorithm is of high importance for a more secure and fast transition to the modern digital transformation. Hardware solutions that could significantly support the quality of RNG in computer systems have been listed. Such solutions would significantly affect the security of modular cryptography. The study also reveals an RSA's cryptographic security problems when using it. It would lead to searching for replacement of modular cryptography with other solutions that are more secure when using the widespread in computer systems RNG solutions.

Acknowledgements

This research is partially supported by the Bulgarian Ministry of Education and Science (contract D01–205/23.11.2018) under the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)," approved by DCM # 577/ 17.08.2018.

References

[1] Julian Jang-Jaccard and SuryaNepal, "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences* 80, no. 5 (2014): 973-993.

[2] Georgi Kostadinov and Tatiana Atanasova, "Security Policies for Wireless and Network Infrastructure," *Problems of Engineering Cybernetics and Robotics* 71 (2019): 14-19.

[3] Kristina Dineva and Tatiana Atanasova, "Regression Analysis on Data Received from Modular IoT System," *33rd annual European and Modelling Conference ESM'2019*, Palma de Mallorca, Spain, December 2019.

[4] Velizar Shalamanov, Vladimir Monov, Ivaylo Blagoev, Silvia Matern, Gergana Vassileva, and Ivan Blagoev, "A Model of ICT Competence Development for Digital Transformation," *Information & Security: An International Journal* 46 (2020): 269-284, https://doi.org/10.11610/isij.4619.

[5] David F. DiCarlo, "Random Number Generation: Types and Techniques," Theses (Liberty University, Center for Computer and Information Technology, 2012).

[6] James Carr, "Simple Random Number Generation," *Computers & Geosciences* 29, no. 10 (2003): 1269-1275, https://doi.org/10.1016/j.cageo.2003.07.002.

[7] Pierre L'Ecuyer, "Random Number Generation," in *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, edited by James E. Gentle, Wolfgang Karl Härdle, and Yuichi Mori (Berlin, Heidelberg: Springer, 2007), https://doi.org/10.1007/978-3-642-21551-3_3.

[8] Andrew Jin, David Ling, and Alwyn Goh, "Biohashing: Two factor authentication featuring fingerprint data and tokenised random number," *Pattern Recognition* 37 (2004): 2245- 2255.

[9] Carmen Camara, Honorio Martín, Pedro Peris-Lopez, and Muawya Aldalaien, "Design and Analysis of a True Random Number Generator Based on GSR Signals for Body Sensor Networks," *Sensors* 19, no. 9 (2019), 2033; https://doi.org/10.3390/s19092033.

[10] Salih Ergün, "Security analysis of a chaos-based random number generator for applications in cryptography," *15th International Symposium on Communications and Information Technologies (ISCIT)*, Nara, Japan, 2015, 319-322, https://doi.org/10.1109/ISCIT.2015.7458371.

[11] Boris Ryabko, JaakkoAstola, and Mikhail Malyutov, *Compression-Based Methods of Statistical Analysis and Prediction of Time Series* (Cham, Switzerland: Springer International Publishing, 2016).

[12] Wade Trappe and Lawrence Washington, *Introduction to cryptography with coding theory,* 2nd ed. (Upper Saddle River, NJ: Pearson, 2006).

[13] Markus Dichtl, "How to Predict the Output of a Hardware Random Number Generator," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems CHES 2003* (Berlin, Heidelberg: Springer, 2003), 181-188, https://doi.org/10.1007/978-3-540-45238-6_15.

[14] Ahmad Lavasani and Taraneh Eghlidos, "Practical next bit test for evaluating pseudorandom sequences," *Scientia Iranica* 16, no. 1 (2009): 19-33.

[15] Joseph Hart, Rajarshi Roy, and Thomas Murphy, "Optical random number generation – harvesting entropy from noise and chaos," *51st Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, USA, March 2017, https://doi.org/10.1109/CISS.2017.7926165.

[16] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter, "Ron was wrong, Whit is right," 2012, https://eprint.iacr.org/2012/064.pdf.

[17] Kerry Scharfglass, Darrin Weng, Joseph White, and Christopher Lupo, "Breaking weak 1024-bit RSA keys with CUDA," *13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Beijing, China, 14-16 December 2012, https://doi.org/10.1109/PDCAT.2012.58.

[18] Bouke Cloostermans, "Quasi-linear GCD computation and factoring RSA moduli," Thesis (Eindhoven University of Technology, Department of Mathematics and Computer Science, 2012).

[19] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices," *21 st Security Symposium Security'12, Bellevue, WA*, Aug. 8-10, 2012, pp. 205-220.

About the Authors

Ivan **Blagoev** is a postdoc at the Institute of Information and Communication Technologies, Bulgarian Academy of Sciences. He holdss a PhD from the IICT-BAS, an MSc from the New Bulgarian University, and a BSc in Informatics from the University of Veliko Tarnovo. He is the CEO of Omega Systems Ltd. Ivan has many scientific publications in the fields of cryptography, cyber security, and financial forecasting. https://orcid.org/0000-0002-6413-8469

Todor **Balabanov** is an assistant professor at the Institute of Information and Communication Technologies, Bulgarian Academy of Sciences. He holds a PhD from the IICT-BAS, an MSc from the New Bulgarian University, and an engineering BSc from the Technical University of Sofia. Todor is a chairman of the Association of Evaluators, Proficients and Experts, and a vice-chairman of the Reasonable Gaming Association. He is the CEO of Velbazhd Software LLC. Todor has many scientific publications in the fields of artificial intelligence, decision making, and financial forecasting. He has contributed to four books in the field of computer science. E-mail: todor.balabanov@iict.bas.bg. https://orcid.org/0000-0003-3139-069X

Iliyan **Iliev** is a PhD student at the Institute of Information and Communication Technologies, Bulgarian Academy of Sciences. He has an MSc from the University of Library Studies and Information Technologies and a BSc from the Sofia University "St. Kliment Ohridski." Iliyan has strong programming skills and wide system administration experience. The scientific interests of Iliyan are in the fields of computer networks, operating systems, and the Internet of Things. E-mail: iliyan.iliev@iict.bas.bg. https://orcid.org/0000-0003-3274-9828