Issues with the Implementation of the RSA Algorithm Key Generation Module (ROCA)

2.1 Introduction

In October 2017, the scheduled publication of papers at ACM CCS 2017*1 triggered major news reports related to cryptographic technology. One was the report of an attack called KRACKs, caused by flaws in the WPA/WPA2 specifications*2. Because this was an issue with the protocol itself, fears that it would become a major problem sparked an overreaction on social networks. However, it could be fixed with a relatively minor correction, so these fears were unjustified. This incident was ranked fourth in the JNSA Major Security News*3, and it created quite a stir regarding the circulation of unverified information and appropriate methods of reporting.

Another was an issue called the Return of Coppersmith's Attack (ROCA), which was caused by the defective implementation of the key generation module in the RSA cryptographic algorithm*4. Although ROCA was reported at around the same time as the KRACKs attack, it did not receive much coverage in Japan. On the other hand, because it was an attack that enabled factorization of the RSA public key more practically in terms of attack time and cost, countermeasures were shipped promptly. It was recognized that this attack degrades functionality while hindering the expected performance of cryptographic functions, so vendors developed patches and prompted users to replace RSA key pairs created using the vulnerable key generation module. Other incidents*5 where bugs or design flaws have caused implementations of cryptographic modules to be far easier to compromise than they should be, or were thought to be, have been disclosed in the past, and ROCA would be listed as one of them. In this volume of the report, we examine past failures and give an overview of the factors behind vulnerabilities in cryptographic implementations that result from reduced space for keys and various parameters similar to ROCA. I will also touch upon the future impact of the results from a series of research projects on attacks such as ROCA.

2.2 An Overview of ROCA

Cryptographic technology is employed when using security protocols such as SSL or TLS. Public-key cryptosystems are used for encryption (ensuring confidentiality) and digital signatures (ensuring data integrity) in applications such as browsers, where general users can tell that a connection is secure when a locked key mark is displayed. One example is the RSA cryptosystem, which bases its security on the difficulty of prime number factorization. Most server certificates contain an RSA public key, and in SSL and TLS, for example, they are widely adopted as a mechanism for guaranteeing the validity of servers. Recently, from the perspective of perfect forward secrecy*6, encryption methods where an ephemeral key (temporary key) is generated using the DH or ECDH algorithm each time are recommended instead of using the public key stored in the server certificate to ensure confidentiality. This makes it possible to use a dedicated signature algorithm when a browser or user confirms whether a server is valid. A typical example of this is the ECDSA signatures*7 based on elliptic curve cryptography. In fact, a growing proportion of server certificates are issued containing ECDSA keys instead of RSA public keys, and these certificates are supported by major browsers. On the other hand, RSA-based server certificates are still currently in wide use, and would be affected by ROCA.

In October 2017, a research team at Masaryk University in the Czech Republic reported a vulnerability in RSA key generation modules made by Infineon Technologies AG and discussed its impact. The RSA algorithm is a cryptosystem where two primes generated at the time of key generation are used as a private key, and a composite number obtained by multiplying these two primes is used as a public key. Security is guaranteed by the fact that a vast amount of computation is required for factorization of the composite number that serves as the public key, meaning decryption is not practically possible. Now it has been reported that a vulnerability was discovered in an implementation of the RSA key generation module, and the bias in generated keys could

*1 ACM Conference on Computer and Communications Security 2017 (https://ccs2017.sigsac.org). CCS 2017 - Accepted Papers (https://acmccs.github.io/papers/).
*2 Key Reinstallation Attacks (https://www.krackattacks.com).
*3 JNSA, "Major Security News" (http://www.jnsa.org/active/news10/) (in Japanese).
*4 Centre for Research on Cryptography and Security (CRoCS), Masaryk University, "ROCA: Vulnerable RSA generation" (CVE-2017-15361) (https://crocs.fi.muni.cz/public/papers/rsa_ccs17).
*5 Internet Infrastructure Review Vol.17 "1.4.1 The Issue of Many Public Keys Used with SSL/TLS and SSH Sharing Private Keys with Other Sites" (https://www.iij.ad.jp/en/dev/iir/017.html).
*6 Internet Infrastructure Review Vol.22 "1.4.2 Forward Secrecy" (https://www.iij.ad.jp/en/dev/iir/pdf/iir_vol22_infra_EN.pdf).
*7 NIST, "FIPS 186-4 Digital Signature Standard (DSS)" (https://csrc.nist.gov/publications/detail/fips/186/4/final). The use of ECDSA is stipulated in Chapter 6. This document also covers DSA signature schemes in Chapter 4, and RSA signature schemes in Chapter 5.

enable factorization in a much shorter time than expected. There have been several reports of incidents where vulnerabilities were recognized due to keys or parameters being derived from a narrower space than originally intended, but most of these were caused by flaws in pseudo-random number generation modules. Although the current issue could be categorized as similar if you only look at the research results, the true nature of the problem actually lies not in the pseudo-random number generator but in a flaw in the implementation of the part related to prime number generation.

2.3 Key Lifecycles and Previous Failures

Two types of keys are used in public key cryptosystems: a private key used for creating digital signatures and decryption, and a public key made available through a server certificate or key repository. Thus, users first generate key pairs prescribed by each cryptosystem, and when doing this process the key pairs are generated using a random number sequence that is derived from a pseudo-random number generation module and can be safely used as a source. This pseudo-random number generation module is deployed so that it can be used as necessary to obtain random number sequences both at the time of key generation and when keys are used (for example, at the time of signing). When using keys—such as when performing encryption with a public key, signing with a private key, or verifying a signature—an expiration date is often set for the public key. After a key expires, it is discarded and a new one is generated, so it has a lifecycle. This key management flow also includes a path for forcibly invalidating the key even before it expires if the corresponding private key leaks or may have leaked. SSL/TLS server certificates have an expiration date, and if you imagine a system where certificates are discarded before this date, you should be able to grasp this concept of lifecycles.

Next, we will look at a few incidents of failure to understand at which stage of the abovementioned key management lifecycle the problem occurred. One case similar to this one where problems occurred when generating RSA keys is a key generation issue in Debian OpenSSL that was disclosed in 2008*8. When using OpenSSL to generate keys in a specific version of Debian, there was a bug where private keys were derived from an extremely small key space. In this incident, a list of all public keys that could be generated from the vulnerable key generation module was released so that anyone could check their keys.

Another similar key generation problem was a flaw in Taiwan citizen cards*9 that was identified in 2013. These were IC cards that had passed the accreditation criteria of a cryptographic module called FIPS 140-2, but significant bias was seen in the prime number generator, and there have been reports of public keys that enabled efficient factorization. Unlike ROCA, this is categorized as a flaw in the pseudo-random number generation module. The vulnerable keys reported here include multiple examples of an issue reported in 2012 where private keys were shared unintentionally*10, enabling factorization with far less computational complexity. Cases where private keys were unintentionally shared were due to the fact that the key space was small, with an implementation for a certain IC card, for example, only able to generate a mere 36 different RSA public keys, and these reports had an extremely large impact*11*12.

There was also a similar case of failure at the time of keys being used rather than at the time of key generation. In this case, an Android application incorporating a Bitcoin wallet function reused the parameters employed for ECDSA signatures, enabling the private key to be identified from two different signatures belonging to the same entity*13. This implementation ignored the restriction that parameters should not be reused as a signature algorithm. Specifically, this was caused by the corresponding parameters overlapping by chance because the pseudo-random number generation module used by the Android application had low entropy outputs. As this demonstrates, issues similar to ROCA are occurring in various phases.

*8 VU#925211, “Debian and Ubuntu OpenSSL packages contain a predictable random number generator” (https://www.kb.cert.org/vuls/id/925211).
*9 Daniel J. Bernstein et.al, “Factoring RSA keys from certified smart cards: Coppersmith in the wild,” Cryptology ePrint Archive: Report 2013/599 (https://eprint.iacr.org/2013/599).
*10 PKI Day 2012, Y.Suga, “The Issue of Many Public Keys Unintentionally Sharing Private Keys with Other Sites” (http://www.jnsa.org/seminar/pki-day/2012/data/PM02_suga.pdf) (in Japanese).
*11 Arjen K. Lenstra et al., “Ron was wrong, Whit is right” (https://eprint.iacr.org/2012/064).
*12 Nadia Heninger et al., “Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices,” Proceedings of the 21st USENIX Security Symposium (https://factorable.net/paper.html).
*13 Bitcoin Project, “Android Security Vulnerability” (https://bitcoin.org/en/alert/2013-08-11-android).

2.4 The True Nature of the ROCA Issue

It could be said this issue that was found in a cryptographic library made by Infineon Technologies AG has the same root cause as other examples, such as the Debian OpenSSL bug, in that the generated key space is too small. However, it should be noted that this is due to problems with the key generation algorithm, rather than bias in the random data output from the pseudo-random number generation module.

Researchers claim that the vulnerability in the key generation module was not discovered through a source code review or reverse engineering. It came to light that the key space that can be established is small when bias is observed in key data after treating the key generation module as a black box and generating a ton of key pairs. RSA public keys are obtained from the product of two primes. The processing of a typical key generation module has a step for determining whether the odd numbers generated from random data as candidates are primes. Because these primality tests generally take time, they can become a bottleneck when generating keys in environments such as IC cards that have limited computation speed and memory space. We know that prime numbers generated by the vulnerable key generation modules discovered in this case are distinctive in that they are derived from a much smaller space than the full prime number space. As identified in the paper by the researchers, it is thought that the constraints of this distinctive prime number format were intended to accelerate prime generation. It seems the people who designed and implemented it did not realize that using this sort of acceleration would create a vulnerability. It is conceivable that a high bar was set for response time, and processing that should be carried out was omitted because performance fell far short of the processing goal.

Similar issues include the fact that each year there have been several dozen vulnerability reports indicating that certificate validation is omitted when conducting SSL/TLS communications in the background in Android applications. Typical browsers notify the user of the certificate validation results via the URL input field or security indicator when communicating with SSL/TLS servers. However, because this sort of information display or user action isn't always necessary for SSL/TLS communications carried out in the background on an Android app, we believe the decision was made to omit the certificate validation module to speed up processing.

The prime number p generated by the vulnerable module identified in ROCA has been found to have the following characteristics:

$$p = k*M + (65537^{a} \bmod M)$$

Here M is the product of n consecutive primes from 2, determined by the length of the prime you want to generate (i.e. n = 39 for 256 bits, and n = 71 for 512 bits). Because M is automatically fixed when n is fixed, prime number candidates are selected by moving parameters k and a as appropriate. For example, the generation of an RSA-512 public key is achieved by multiplying two 256-bit primes. According to the prime number theorem, which indicates the density of primes, it is known there are about $2^{248.5}$ candidates, so we can see it is possible to randomly select a prime from an extremely large prime number space. Meanwhile, in this vulnerable prime generation module with n = 39, M = 2*3*5*...*167 is about 219 bits long, so only 37 bits can be moved via parameter k. Similarly, only 62 bits can be selected via parameter a, resulting in just 99 bits worth of entropy. This means that primes are generated from a considerably smaller key space than usual.

RSA is an algorithm that bases its security on the difficulty of factorization, and the NIST has estimated the equivalent encryption strength in symmetric-key cryptography key bit length when using various RSA key lengths. For example, their tables show that 2048-bit RSA has 112 bits of security strength, and elliptic curve cryptography such as the aforementioned ECDSA has a security strength of exactly half its key length*14. With incidents such as the factorization of 768-bit RSA keys in 2010, the use of RSA keys of at least 2048-bit length is currently recommended. When using RSA signatures with PKI, organizations such as the CA/Browser Forum now prescribe a policy of migrating away from the use of 1024-bit RSA keys, as well as the SHA-1

*14 NIST, SP 800-57 Part 1 Rev. 4, "Recommendation for Key Management, Part 1: General" (https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final).

hash function*15 recognized as one of the compromised algorithms after collisions were discovered last year. Such policy is now followed when certificates are issued. A report published each year at CRYPTREC gives comparisons with the processing capability of supercomputers*16, and this supports the fact that 2048-bit RSA is sufficiently secure at this point in time.

2.5 The Impact of ROCA

As mentioned in the previous section, it is easy to see how the search space for factorization becomes smaller because the key space narrows significantly due to constraints on generated primes. Similarly, there is research that aims to make factorization easier through the use of the form of private key constraints, such as the well-known Coppersmith method*17 that appears in the title of ROCA. The Coppersmith method can efficiently restore p from product N of the two primes p and q (i.e. N = p*q) used in the RSA algorithm and the blobs of the lower half of the prime p enable successful factorization as a result. It is also one of the efficient methods that derived the private key of SSL/TLS servers in a competition designed to determine what level of threat the Heartbleed bug in OpenSSL posed*18.

By extending the Coppersmith method to ROCA, the cost of using the amount of cloud resources required for factorization is estimated to be as shown in Table 1. We can see that even the 2048-bit RSA keys in wide use today can be decrypted with practical low costs. Within just a week of the announcement of these results, it was suggested that factorization could be achieved with 5–25% more efficient costs. This means that factorization is even easier and less costly than estimated in the original paper*19.

Tools for checking whether RSA keys were generated by the affected cryptographic modules were released by the authors. Various verification methods were made available, including Python code*20 that lets you perform checks offline, an online version that lets you check by posting public keys via a browser*21, and a version that lets you send an S/MIME signature via email to obtain results. The system for checking for the ROCA vulnerability is very simple and concise*22. According to the authors there are no false positives, and the probability of accidentally generating keys vulnerable to ROCA is just 2^{-154}, which is negligible.

It has been announced that the certificates used with e-Residency ID cards*23 issued by the government of Estonia are affected by ROCA*24, and users are being asked to perform firmware updates and re-generate keys*25*26. According to the original paper, around 4,400 e-Residency ID cards were surveyed, and it was shown that all of them were affected. Also, the ROCA bug has been present since 2012, and although they would have expired by now, it is thought there are keys that continued to be used

Table 1: The Costs When Using the Cloud Resources Required for Factorization

RSA key length	CPU resources required for factorization	Factorization costs when using the cloud
512 bit	1.93 CPU hours	$0.06
1024 bit	97.1 CPU days	$40–$80
2048 bit	140.8 CPU years	$20,000–$40,000

*15 CRYPTREC Cryptographic Technology Guideline - SHA-1 (https://www.cryptrec.go.jp/topics/cryptrec_20180427_eval_gl_2001_2013r1.html) (in Japanese). Security Diary, SHAttered attack (SHA-1 collision discovery) (https://sect.iij.ad.jp/d/2017/02/271993.html) (in Japanese).
*16 CRYPTREC Report (http://www.cryptrec.go.jp/english/report.html).
*17 Don Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known", EUROCRYPT96, 178–189.
*18 IIJ-SECT Security Diary, "Regarding the feasibility of private keys leaking due to the Heartbleed bug" (https://sect.iij.ad.jp/d/2014/04/159520.html) (in Japanese).
*19 The cr.yp.to blog, "2017.11.05: Reconstructing ROCA" (https://blog.cr.yp.to/20171105-infineon.html).
*20 ROCA: Infineon RSA key vulnerability (https://github.com/crocs-muni/roca/).
*21 ROCA Vulnerability Test Suite (https://keychest.net/roca), Test your RSA Keys (https://keytester.cryptosense.com).
*22 Key fingerprinting (https://github.com/crocs-muni/roca/blob/master/roca/detect.py).
*23 Republic of Estonia, "e-Residency" (https://e-resident.gov.ee/become-an-e-resident/).
*24 Politsei.ja Piirivalveamet, "Possible Security Vulnerability Detected in the Estonian ID-card Chip" (https://www2.politsei.ee/en/uudised/uudis.dot?id=785151).
*25 Politsei.ja Piirivalveamet, "For the user of ID-card and mobile ID" (https://www2.politsei.ee/en/nouanded/isikut-toendavad-dokumendid/id-kaardi-ja-mobiil-id-kasutajale.dot).
*26 Politsei.ja Piirivalveamet, "Renewal of document certificates-frequently asked questions" (https://www2.politsei.ee/en/teenused/isikuttoendavad-dokumendid/sertifikaatide-uuendamine/).

for over five years without the vulnerability coming to light, despite the fact that factorization was possible from the beginning. Each company, including Japanese vendors, has provided detailed information on the products enabling TPM (Trusted Platform Module) chips*27*28. The recommended actions were to update the firmware and discard any RSA key pairs generated by the vulnerable TPM chips, then create new ones. TPM is a tamper-resistant chip that protects against physical attacks on private keys, and it had been regarded as more secure than storing key data in memory or other storage devices. However, with the discovery of ROCA, new potential disadvantages of making a module that uses keys as a black box were revealed. While outsourcing parts that are difficult to manage lets you take a more hands-off approach, new threats may arise in areas no longer under your control.

2.6 How ROCA Was Discovered

As we saw in section 2.4, the ROCA vulnerability was discovered by focusing on the prime generation modules that output special form keys. The process that discovered a bias by observing a large volume of primes without reverse engineering or accessing the source code is based on prior research by the same team. This was a study of the RSA keys generated by 38 cryptographic software products and IC cards presented at USENIX Security 2016, which was held in August 2016*29. The study revealed that each cryptographic library had distinctive features using heat maps that plotted primes p and q on the two axes to show the frequency distribution of seven bits from 2nd to 8th most significant. It also organized the 38 products into 13 categories by investigating trends through extraction of just the most distinctive nine bits, covering a prime's 2nd to 7th most significant bits, the 2nd least significant bit, prime mod 3 (as a value), and public key N mod 2 (as a value). The Infineon module targeted by ROCA was categorized as Class 12, and at this point it was identified that it had more distinctive characteristics than the cryptographic libraries belonging to other categories.

Related research results were also released by the same research team at ACSAC 2017, which was held after ACM CCS*30. This was an attempt to identify the cryptographic library that generated a public key from just the public key information using their prior knowledge of cryptographic libraries. The same approach was used at USENIX Security 2016 to identify the cryptographic library by calculating the remainder of mod 3 and mod 4 for public key N and detecting whether there was bias. The researchers claim there are one percent or fewer false positives. They also pointed out the impact on privacy issues. For example, observing the public key information used in Tor reveals that different nodes are identified as the same user or specified region.

2.7 The Potential Impact of ROCA on Other Cryptographic Algorithms

Because ROCA itself is a vulnerability in the key generation module for the RSA cryptosystem, or in other words an issue with prime generation logic, it is unlikely to affect cryptographic algorithms other than RSA that generate and use primes but do not need to keep them secret. In the RSA cryptosystem, for example, two 1024-bit primes are required as private key candidates when generating a 2048-bit RSA public key. Meanwhile, the private keys for the abovementioned ECDSA signatures used in Bitcoin can be generated simply by deriving an integer (256-bit length), so no complex logic is required. In other words, only the validity of the pseudo-random number generation module affects the security of the key generation module, so it is unlikely that a vulnerability like ROCA will come into play.

In the cryptographic key lifecycle, we consider the phase in which the key will be used and not key generation. It is normally necessary to generate and sign different parameters each time, as seen with the aforementioned problems of pseudo-random number generation modules in Android, but the special logic seen in the prime generation process is not required here either. However, in the generating process of private keys or various parameters, considering the cases where random data is derived by

*27 Infineon Technologies AG, "Information on TPM firmware update for Microsoft Windows systems as announced on Microsoft's patchday on October 10th 2017" (https://www.infineon.com/cms/en/product/promopages/tpm-update/?redirId=59160).

*28 CERT, "Vulnerability Note VU#307015, Infineon RSA library does not properly generate RSA key pairs" (https://www.kb.cert.org/vuls/id/307015).

*29 Centre for Research on Cryptography and Security (CRoCS), Masaryk University, "The Million-Key Question - Investigating the Origins of RSA Public Keys" (https://crocs.fi.muni.cz/public/papers/usenix2016).

*30 Centre for Research on Cryptography and Security (CRoCS), Masaryk University, "Measuring Popularity of Cryptographic Libraries in Internet-Wide Scans [ACSAC 2017]" (https://crocs.fi.muni.cz/public/papers/acsac2017).

simply filling the most significant bits with zeros or repetition of a short bit string, we cannot ignore the potential for vulnerabilities caused by a small data space even in the ECDSA signature algorithm. At this time it is possible to check whether a public key in the Bitcoin network has derived the same key pair by referring to the blockchain history. In other words, it is possible to determine whether you have shared private keys with other third parties. As such, it is conceivable that there exist implementations in which a backdoor that makes the key space smaller has been incorporated even though it appears on the surface that keys are generated according to proper procedures.

If it is unintentionally revealed that a key generation module is vulnerable, as with ROCA, this could lead to attacks that share key pairs with another party with a higher probability than expected when repeatedly generating keys. This has no effect on cold wallets that ensure security by not connecting signing keys themselves and/or signature modules to the network. This demonstrates how crucial the key generation phase is for all virtual currencies, not just Bitcoin. As seen in the abovementioned example of the Taiwan citizen cards, even cryptographic libraries or HSMs (Hardware Security Modules) certified FIPS 140-2 require key management that takes into account the fact that vulnerable products may exist. In the Bitcoin network, Bitcoin addresses are used as owner IDs. Bitcoin addresses are created by a generated key pair using the ECDSA signature method on the elliptic curve known as secp256k1. Then, after calculating the digest from the public key data with version information and checksum data, using two different hash functions, this is finally converted into a human-readable string of 26 to 35 characters using Base58 encoding. There are known methods for deriving a "desired Bitcoin address" so that characters specified by the user appear in the Bitcoin address during this procedure. The hash functions used here are SHA-2 and RIPEMD-160, and because it is difficult to derive the intended output from them, different key pairs are derived through trial and error repeatedly. This method generates a significant number of key pairs, so secure random data is required here as well. Many generation tools such as these are now available, but there is no way to trust them, and as they are distributed as binary rather than source code in some cases, they must be used with caution.

Here we discussed vulnerabilities in cryptographic modules in which the RSA algorithm is implemented, along with privacy issues that could occur in the future. It was reacknowledged that implementations of cryptographic algorithms are susceptible to bugs because its prime number generation and primality tests require slightly abstruse logic, but the RSA algorithm itself is mostly unscathed, and it can be used securely as long as knowledge regarding its implementation is shared. With the expectation that it will no longer be usable in the future when factorization becomes trivial due to the advent of quantum computers, next-generation algorithms known as post-quantum cryptography are also being developed*31. NIST is currently conducting a standardization competition, and the anticipated schedule involves cryptanalysis and evaluation over the next three to five years, followed by the sharing of a standardized draft within another two years or so*32. When implementing next-generation cryptographic technology, there may be mechanisms that designers and implementors find more complicated and situations that require a vast quantity of knowledge to understand. Comprehending the true nature of issues like ROCA that occur on modern cryptosystems such as RSA should serve as a lesson for the next generation.

Author:
Yuji Suga
Senior Engineer, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ
Dr. Suga has been in his current position since July 2008. He is engaged in investigation and research activities related to cryptography and information security as a whole. He is a member of the CRYPTREC Cryptographic Technology Promotion Committee.
Dr. Suga also serves as secretariat of the Cryptographic protocol Evaluation toward Long-Lived Outstanding Security Consortium (CELLOS). He is assistant secretary of the ISEC Technical Committee of the Institute of Electronics, Information and Communication Engineers (IEICE). He is an organizing committee member for IWSEC2018.
He is an organizing committee member for ECC2018. Virtual Currency Governance Task Force (VCGTF) Security WG member.

*31 Internet Infrastructure Review Vol.31 "1.4.3 Trends in Post-Quantum Cryptography" (https://www.iij.ad.jp/en/dev/iir/pdf/iir_vol31_EN.pdf).
*32 Dustin Moody, "THE SHIP HAS SAILED: The NIST Post-Quantum Crypto 'Competition'" (https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/asiacrypt-2017-moody-pqc.pdf).