Comment on "Efficient and Secure Outsourcing Scheme for RSA Decryption in Internet of Things"

Damien Vergnaud

Comment on "Efficient and Secure Outsourcing Scheme for RSA Decryption in Internet of Things"

Damien Vergnaud, *Member, IEEE*

***Abstract*—Internet of Things (IoT) devices have grown in popularity over the past few years. The RSA public-key cryptographic primitive is time-consuming for resource-constrained IoT. Recently, Zhang, Yu, Tian, Tong, Lin, Ge and Wang proposed a two-party outsourcing protocol between a client and a server for RSA decryption in IoT. It relies on the Chinese Remainder Theorem as proposed by Quisquater and Couvreur in 1982 and is very efficient.**

We show that their protocol does not achieve the claimed security guarantees: (1) the (secret) decryption exponent, the plaintext and the factorization of the RSA modulus are revealed to a passive adversary, and (2) a malicious server can make the client accept an (invalid) value of its choice as the result of the delegated computation.

***Index Terms*—Cloud computing, Edge computing, Secure outsourcing, RSA, Internet of Things, Cryptanalysis**

I. Introduction

THE Internet of Things (IoT) is growing quickly and brings a new set of security concerns. It connects billions of physical devices (classical computing and communication devices, but all kinds of objects used in our everyday lives: cars, door locks, personal medical devices, ...) for collecting and sharing data, putting more sensitive information at risk.

Deploying cryptographic mechanisms on IoT devices is thus often desired (for securing communication, protecting firmware, and authentication). However, the computational resources of IoT devices can be very limited, and it seems very natural, as most of them are online to securely delegate the costly cryptographic operations to a device capable of carrying out them. Outsourcing cryptographic computations is a classical problem which was formalized in [5].

This problem is particularly challenging for public-key cryptography such as the RSA primitive [15]. In [18], Zhang, Yu, Tian, Tong, Lin, Ge and Wang designed an efficient outsourcing scheme for RSA decryption in IoT. RSA decryption is achieved via modular exponentiation and their protocol is based on the Chinese Remainder Theorem (CRT) as proposed in 1982 by Quisquater and Couvreur [12]. Zhang *et al.* claimed that their delegation protocol is highly efficient for both the client and the server and that the private key and the plaintext are concealed concurrently within the proposed scheme. They also claimed that it enables the client to detect any misbehavior of the server with a probability of 99.17%. They provided an efficiency analysis and they also mentionned that they provided rigorous proofs of security and verifiability in the formal security model from [5].

D. Vergnaud was with Sorbonne Universit, CNRS, LIP6, F-75005 Paris, France and Institut Universitaire de France, e-mail: damien.vergnaud@sorbonne-universite.fr.

In this note, we show that their protocol does not achieve the claimed security guarantees: (1) the (secret) decryption exponent, the plaintext and the factorization of the RSA modulus are revealed to a passive adversary, and (2) a malicious server can make the client accept an (invalid) value of its choice as the result of the delegated computation.

II. Description of Zhang *et al.*'s Protocol

We first provide a short description of the classical "textbook" RSA public-key encryption scheme [15] (using the notations from [18]):

Key Generation: On input a parameter $\lambda \in \mathbb{N}$, the algorithm picks uniformly at random two distinct prime numbers p and q of bit-length λ. It then computes $n = pq$ and $\varphi(n) = (p-1)(q-1)$ the Euler totient function of n. It picks uniformly at random an integer $e \in \{1, \ldots, \varphi(n)\}$ coprime with $\varphi(n)$ and computes the integer $d \in \{1, \ldots, \varphi(n)\}$ such that $ed \equiv 1 \bmod \varphi(n)$. It outputs (n, e) as the public-key and (n, d) as the private key.

Encryption: To encrypt a plaintext $M \in \mathbb{Z}_n$ for a public key (n, e), the algorithm outputs $C = M^e \bmod n$.

Decryption: To decrypt a ciphertext $C \in \mathbb{Z}_n$ with a private key (n, d), the algorithm outputs $M = C^d \bmod n$.

It has been proposed as soon as in 1982 by Quisquater and Couvreur [12] to use the Chinese Remainder Theorem (CRT) in order to improve the efficiency of the decryption algorithm. The "textbook" RSA-CRT public-key encryption scheme is modified as follows:

Key Generation: With the same notation as above, the algorithm additionally computes

$$d_p = d \bmod (p-1) \quad (1)$$

$$d_q = d \bmod (q-1). \quad (2)$$

It outputs (n, e) as the public-key and (n, d_p, p, d_q, q) as the private key.

Decryption: To decrypt a ciphertext $C \in \mathbb{Z}_n$ with a private key (n, d_p, p, d_q, q), the algorithm first computes $M_p = C^{d_p} \bmod p$ and $M_q = C^{d_q} \bmod q$ and outputs the unique $M \in \mathbb{Z}_n$ such that $M \equiv M_p \bmod p$ and

$M \equiv M_q \bmod q$ (thanks to the knowledge of $p^{-1} \bmod q$ and $q^{-1} \bmod p$):

$$M = p \cdot (p^{-1} \bmod q) \cdot M_q + q \cdot (q^{-1} \bmod p) \cdot M_p \bmod n$$

Zhang *et al.*'s delegation protocol for RSA-CRT public-key encryption scheme works as follows

1) Given a ciphertext $C \in \mathbb{Z}_n$, the client picks uniformly at random two integers r_1 and r_2 of Λ bits (for some integer parameter Λ) and computes:
$$d_{p_1} = d_p + r_1(p-1) \quad (3)$$
$$d_{q_1} = d_q + r_2(q-1) \quad (4)$$
2) For the purpose of verification, it also picks uniformly at random two integers r_3 and r_4 of Λ bits and three integers t_1, t_2, k in the range $\{2, 3, 4, \ldots, 11\}$. The client then computes:
$$d_{p_2} = d_p t_1 + k + r_3(p-1) \quad (5)$$
$$d_{q_2} = d_q t_2 + k + r_4(q-1) \quad (6)$$
3) The client queries the server (in a random order) the modular exponentiation of C to the power d_{p_1}, d_{p_2}, d_{q_1} and d_{q_2}.
4) The serveur computes the values $M_p = C^{d_{p_1}} \bmod n$, $M'_p = C^{d_{p_2}} \bmod n$, $M_q = C^{d_{q_1}} \bmod n$ and $M'_q = C^{d_{q_2}} \bmod n$ and sends them to the client.
5) The client checks whether the following equalities hold:
$$M_p^{t_1} C^k = M'_p \bmod p \quad (7)$$
$$M_q^{t_2} C^k = M'_q \bmod q. \quad (8)$$
If this is the case, the client outputs
$$M = p \cdot (p^{-1} \bmod q) \cdot M_q + q \cdot (q^{-1} \bmod p) \cdot M_p \bmod n$$
as the plaintext corresponding to C.

III. Cryptanalysis

A. Passive Attack on the Protocol Privacy

In this subsection, we describe an adversary which can recover the (secret) plaintext, the (secret) decryption exponent and the factorization of n from a passive eavesdropping of a single execution of the delegation protocol. Following Kerckhoff's principles [6], it is natural to assume that the adversary knows the public key (n, e) it attacks. Zhang *et al.* [18], indeed do not add the value of the public exponent e to the list of secret inputs in their "security proof". Actually, in practical applications, RSA users very often use $e = 2^{16} + 1 = 65537$ as the public exponent. In [8], Lenstra, Hughes, Augier, Bos, Kleinjung and Wachter performed a sanity check of public keys collected on the web and found in particular that more than 98.4% of RSA keys in X.509 certificates and more than 48.8% of RSA public keys in PGP (giving a 95.4 percentage over all the keys) used $e = 65537$ as the public exponent. For all considered RSA keys, less than 0.008% of all keys used a public exponent that does not belong to a very short list of 10 values.

Let us first assume that the adversary knows which query to the server corresponds to which exponent in the set $\{d_{p_1}, d_{p_2}, d_{q_1}, d_{q_2}\}$. From (1), there exists an integer γ such that

$$ed_p = 1 + \gamma(p-1).$$

Combined with (3), we get

$$\begin{aligned} ed_{p_1} = e(d_p + r_1(p-1)) &= ed_p + e \cdot r_1(p-1) \\ &= 1 + (\gamma + e \cdot r_1)(p-1) \end{aligned}$$

and thus $(p-1)$ is a divisor of $(ed_{p_1} - 1)$.

Similarly from (2) and (4), we obtain that $(q-1)$ is a divisor of $(ed_{q_1}-1)$. We thus get that $(ed_{p_1}-1)(ed_{q_1}-1)$ is a multiple of $(p-1)(q-1) = \varphi(n)$.

In [13], [14], Rabin provided a probabilistic polynomial-time algorithm which given an RSA modulus $n = pq$ and its Euler totient function $\varphi(n)$, outputs the factorization (p, q) in expected polynomial time. Rabin algorithm consists simply in computing some modular exponentiations modulo n of a random base with an exponent smaller than the known multiple of $\varphi(n)$. The expected number of these computations is constant and each of them has binary complexity

$$O(\log(n)^2 \cdot (\log(n \cdot e^2) + \Lambda)) = O(\log(n)^3 + \log(n)^2 \cdot \Lambda)$$

and is thus polynomial time.

The knowledge[1] of $(ed_{p_1}-1)(ed_{q_1}-1)$ therefore allows the adversary to recover the factorization (p, q) of n. It can then compute $\varphi(n) = (p-1)(q-1)$ and from this knowledge, it obtains $d = e^{-1} \bmod \varphi(n)$ and recover the plaintext corresponding to C as $M = C^d \bmod n$.

In the general case where the adversary does not know which delegated exponentiation in (C, α_1), (C, α_2), (C, α_3) and (C, α_4) corresponds to which exponent in $\{d_{p_1}, d_{q_2}, d_{p_2}, d_{q_2}\}$, it can simply apply the previous attack for the six pairs $\{x, y\} \subset \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ and apply Rabin's algorithm with the 6 values $(ex-1)(ey-1)$ as a possible multiple of $\varphi(n)$. This increases the running time only by a constant factor. Zhang *et al.*'s protocol does not provide privacy since this probabilistic polynomial time recovers the plaintext, the (secret) decryption exponent and the factorization of n from a passive eavesdropping of a single execution of the protocol (and *Theorem 1* from [18] is therefore flawed).

Remark 1. *It is worth mentioning that even if we assume that the public exponent e is kept secret, an adversary can run the same attack by using the information obtained in two independent executions of the protocol. Indeed, given the pair (d_{p_1}, d_{q_1}) from the first execution and (d'_{p_1}, d'_{q_1}) from the second execution, it can computes $d_{p_1} - d'_{p_1}$ which is a multiple of $(p-1)$ and $d_{q_1} - d'_{q_1}$ which is a multiple of $(q-1)$. From those two multiples, the adversary can again compute a multiple of $\varphi(n)$ and run the previous attack.*

[1]The knowledge of $(ed_{p_1} - 1)$ is most likely sufficient since for $x \in \mathbb{Z}_n^*$, we have $x^{(ed_{p_1}-1)} \equiv 1 \bmod p$ and in most cases $x^{(ed_{p_1}-1)} \not\equiv 1 \bmod q$ and thus $\gcd(x^{(ed_{p_1}-1)} - 1, n) = p$ reveals the factorization (p, q) of n.

B. Active Attack on the Protocol Verifiability

In this subsection, we show that a malicious server can make the client accept (with overwhelming probability) an arbitrary message $\widetilde{M} \in \mathbb{Z}_n$ with $\widetilde{M} \neq M$ as the output of a delegation protocol.

First of all, when it receives the four exponentiation queries (C, α_1), (C, α_2), (C, α_3) and (C, α_4), the server runs the previous attack to obtain the decryption exponent d and the factorization (p, q) of n.

The malicious server then computes $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$. Among the four received exponents $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, it identifies d_{p_1} as the α_i such that $(d_p - \alpha_i)$ is a multiple of $(p-1)$ for $i \in \{1, 2, 3, 4\}$ (and d_{q_1} as the α_j such that $(d_q - \alpha_j)$ is a multiple of $(q-1)$, for $j \in \{1, 2, 3, 4\}$). With overwhelming probability (over the randomness used in the key generation algorithm), these exponents d_{p_1} and d_{q_1} are identified uniquely.

To identify d_{p_2}, the server checks which of the two remaining exponents modulo $(p-1)$ is equal to $[d_p t + k \bmod (p-1)]$ for t and k in the range $\{2, 3, 4, \ldots, 11\}$. Again, with overwhelming probability (over the randomness used in the key generation algorithm), the exponent d_{p_2} is identified uniquely and when this is done the four exponents sent in a random order are identified. Eventually, using an exhaustive search over the range $\{2, 3, 4, \ldots, 11\}$, the malicious server can also compute the integers t_1, t_2 and k which satisfy (5) and (6).

It can then set $\widetilde{M}_p = \widetilde{M} \bmod p$ and $\widetilde{M}_q = \widetilde{M} \bmod q$ for an arbitrary $\widetilde{M} \neq M$ in $\mathbb{Z}_n$. It computes $\widetilde{M}'_p = \widetilde{M}_p^{\,t_1} C^k \bmod p$ and $\widetilde{M}'_q = \widetilde{M}_q^{\,t_2} C^k \bmod q$. It reply to the client with the 4-tuple $(\widetilde{M}_p, \widetilde{M}'_p, \widetilde{M}_q, \widetilde{M}'_q)$ in the order corresponding to the identified exponents $(d_{p_1}, d_{p_2}, d_{q_1}, d_{q_2})$. These values satisfy (7) and (8) and the user outputs $\widetilde{M} \neq M$ as the plaintext corresponding to C.

Zhang *et al.*'s protocol does not achieve verifiability since this probabilistic polynomial time (and *Theorem 2* from [18] is therefore flawed).

IV. Conclusion

There is a long history of protocols for outsourcing group exponentiations in different settings (*e.g.* public/secret, fixed/variable bases and public/secret exponents) in groups of *known prime* order and in the RSA setting of groups of *secret unknown prime* order (see [10], [7], [1], [9], [2], [3], [17], [4], [16]). Chevalier *et al.* [4] provided simple constructions (essentially optimal in terms of operations in the underlying group) in groups of *known prime* order. For RSA-based cryptography, most proposed protocols are variants of two protocols (named RSA-S1 and RSA-S2) that were proposed by Matsumoto, Kato and Imai in 1988 [10] and analyzed by Mefenza and Vergnaud [11]. For a variable base (which is the case of interest for RSA decryption/signature), all known secure delegation protocols only improve the client efficiency by a constant factor and are thus probably not suitable for limited devices in IoT. Chevalier *et al.* proved lower bounds on the efficiency for generic modular outsourcing protocols (in prime order groups) [4]. These bounds suggest that improving the protocols from [11] in unknown order groups is probably difficult.

Acknowledgment

The author is supported in part by the French ANR ALAMBIC Project (ANR-16-CE39-0006). Olivier Blazy is gratefully acknowledged for providing a copy of the paper [18]. Finally, the author thanks the anonymous referees for their reviews and comments and the Editor Kim-Kwang Raymond Choo for handling the manuscript.

References

[1] Philippe Béguin and Jean-Jacques Quisquater. Fast server-aided RSA signatures secure against active attacks. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 57–69, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany.

[2] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Improving secure server performance by re-balancing SSL/TLS handshakes. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *ASIACCS 06: 1st ACM Symposium on Information, Computer and Communications Security*, pages 26–34, Taipei, Taiwan, March 21–24, 2006. ACM Press.

[3] Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang, and Wenjing Lou. New algorithms for secure outsourcing of modular exponentiations. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012: 17th European Symposium on Research in Computer Security*, volume 7459 of *Lecture Notes in Computer Science*, pages 541–556, Pisa, Italy, September 10–12, 2012. Springer, Heidelberg, Germany.

[4] Céline Chevalier, Fabien Laguillaumie, and Damien Vergnaud. Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016: 21st European Symposium on Research in Computer Security, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 261–278, Heraklion, Greece, September 26–30, 2016. Springer, Heidelberg, Germany.

[5] Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 264–282, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.

[6] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, page 583, 1883.

[7] Chi-Sung Laih, Sung-Ming Yen, and Lein Harn. Two efficient server-aided secret computation protocols based on the addition sequence. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'91*, volume 739 of *Lecture Notes in Computer Science*, pages 450–459, Fujiyoshida, Japan, November 11–14, 1993. Springer, Heidelberg, Germany.

[8] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 626–642, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[9] Chae Hoon Lim and Pil Joong Lee. Security and performance of server-aided RSA computation protocols. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 70–83, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany.

[10] Tsutomu Matsumoto, Koki Kato, and Hideki Imai. Speeding up secret computations with insecure auxiliary devices. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany.

[11] Thierry Mefenza and Damien Vergnaud. Cryptanalysis of server-aided RSA protocols with private-key splitting. *Comput. J.*, 62(8):1194–1213, 2019.

[12] Jean-Jacques Quisquater and Christophe Couvreur. Fast decipherment algorithm for RSA public key cryptosystem. *Electronic Letters*, 18(21):905–907, 1982.

[13] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, USA, 1979.

[14] Michael O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12:128–138, 1980.

[15] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[16] Damien Vergnaud. Secure outsourcing in discrete-logarithm-based and pairing-based cryptography (Invited talk). In *Information Security Theory and Practice - 12th IFIP WG 11.2 International Conference, WISTP 2018, Brussels, Belgium, December 10-11, 2018, Revised Selected Papers*, pages 7–11, 2018.

[17] Yujue Wang, Qianhong Wu, Duncan S. Wong, Bo Qin, Sherman S. M. Chow, Zhen Liu, and Xiao Tan. Securely outsourcing exponentiations with single untrusted program for cloud storage. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014: 19th European Symposium on Research in Computer Security, Part I*, volume 8712 of *Lecture Notes in Computer Science*, pages 326–343, Wroclaw, Poland, September 7–11, 2014. Springer, Heidelberg, Germany.

[18] H. Zhang, J. Yu, C. Tian, L. Tong, J. Lin, L. Ge, and H. Wang. Efficient and secure outsourcing scheme for RSA decryption in Internet of Things. *IEEE Internet of Things Journal*, to appear:14 pages, 2020.