

Taylor & Francis
Taylor & Francis Group

Cryptologia

ISSN: 0161-1194 (Print) 1558-1586 (Online) Journal homepage: www.tandfonline.com/journals/ucry20

RSA+: an RSA variant

Sören Kleine, Andreas Nickel, Torben Ritter & Krishnan Shankar

To cite this article: Sören Kleine, Andreas Nickel, Torben Ritter & Krishnan Shankar (04 Jun 2025): RSA+: an RSA variant, Cryptologia, DOI: 10.1080/01611194.2025.2496141

To link to this article: https://doi.org/10.1080/01611194.2025.2496141

© 2025 The Author(s). Published with license by Taylor & Francis Group, LLC

Published online: 04 Jun 2025.

Submit your article to this journal

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=ucry20

CRYPTOLOGIA
https://doi.org/10.1080/01611194.2025.2496141

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

RSA+: an RSA variant

Sören Kleine, Andreas Nickel, Torben Ritter, and Krishnan Shankar

Institut f€ur Anwendungssicherheit, Universit€at der Bundeswehr M€unchen, Werner-Heisenberg-Weg 39, Germany

ABSTRACT
We introduce a new probabilistic public-key cryptosystem which combines the main ingredients of the well-known RSA and Rabin cryptosystems. We investigate the security and performance of our new scheme in comparison to the other two.

KEYWORDS
public-key cryptography; encryption schemes; Rabin cryptosystem; RSA

2020 MATHEMATICS SUBJECT CLASSIFICATION
94A60

1. Introduction

The RSA and Rabin cryptosystems [see [Rabin 1979; Rivest, Shamir, and Adleman 1978]) certainly are among the best-known public-key cryptosystems. In both schemes, the computations are done in the unit group of the ring of integers modulo n, a composite number with two large prime factors p and q. The integer n forms (part of) the public key of each user, and the prime factors p and q are kept secret.

Whereas the RSA scheme is a widely used public-key cryptosystem, the scheme of Rabin is well-known mainly for theoretical reasons: in contrast to RSA, it is known that an attacker who is able to break the Rabin cryptosystem can factor the public modulus n. Therefore the safety of the Rabin cryptosystem relies on the very-well studied hard mathematical problem of factoring a composite number with two large prime factors (the same security level is conjectured to hold true also for RSA, but this has not yet been proven). On the other hand, the Rabin cryptosystem has certain disadvantages from the practical point of view: The decryption function in the Rabin cryptosystem returns four possible clear texts, which can be reduced to two by adding a parity bit, and therefore the user is faced with the additional task to decide which clear text is the correct message (see Section 4 below).

CONTACT Sören Kleine soeren.kleine@unibw.de Institut für Anwendungssicherheit, Universität der Bundeswehr München, Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany.

© 2025 The Author(s). Published with license by Taylor & Francis Group, LLC
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

The motivation for the present note was to somehow combine the RSA and Rabin schemes in order to create, by using elements from the RSA scheme, a new public-key cryptosystem which on the one hand is more practical than Rabin's cryptosystem and at the same time has a security level which is as close as possible to the security level of the Rabin cryptosystem. A related reason for pursuing RSA+ is that some attacks on RSA use knowledge of the public exponent e to derive d, the decryption exponent, and this protocol helps obfuscate the encryption exponent. We begin by describing our scheme and then investigate the pros and cons of our new approach. More precisely, we study the runtime of our scheme and compare it with plain RSA and Rabin implementations in Section 2. The third section is devoted to a security analysis of the cryptosystem, and in Section 4 we describe an advantage of our scheme over the classical Rabin cryptosystem: If one chooses the primes p and q carefully, it returns a unique clear text with high probability.

2. The cryptosystems

We begin by briefly recapitulating the (textbook) RSA and Rabin schemes. Afterwards we introduce our newly proposed cryptosystem.

2.1. (Textbook) RSA

Let p and q be two distinct prime numbers greater than 2^{λ}, where λ is a fixed security parameter, and let $n = p \cdot q$. Bob's public key is (n, e) where e is coprime with $\varphi(n) = (p-1)(q-1)$. His private key is (p, q, d), where d has been chosen such that $de \equiv 1 \pmod{\varphi(n)}$.

Note that d is efficiently computable via the extended Euclidean algorithm if p and q are known. In fact, the knowledge of the inverse d of e modulo $\varphi(n)$ is equivalent to the knowledge of the prime factors p and q of n (see for example [Rivest, Shamir, and Adleman 1978, Section IX]).

Encryption. In order to send a message $m \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ to Bob, Alice computes

$$c \equiv m^e \pmod{n}.$$

Decryption. Bob can decrypt the message *via* $m \equiv c^d \pmod{n}$.

Remark 1.1. It is recommended to choose p and q of the same size in RSA. One can find different concrete suggestions for the sizes of p and q in the literature (see also https://www.keylength.com). For this paper we consider primes with 1500 to 3000 bits, and such that $p < q < 2p$.

Remark 1.2. In practical implementations of the RSA scheme one often chooses *first* the encryption exponent $e = 2^{16} + 1 = 65537$ (this is a prime

number), and *then* one chooses prime numbers p and q such that e does not divide $p-1$ or $q-1$. This choice for e is made because encryption with this exponent is particularly efficient (see [Schneier 2015, p. 469]).

2.2. Rabin

The public key is just $n = p \cdot q$, where p and q are as above.

Encryption. A message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$ is encrypted by computing $c \equiv m^2 \pmod{n}$.

Decryption. First the square-roots of a given ciphertext c (lifted from $(\mathbb{Z}/n\mathbb{Z})^\times$) modulo p and q are computed. Then the Chinese Remainder Theorem (CRT) is used in order to obtain four different square-roots of c modulo n. It remains to decide which of them corresponds to the original message m (see also Section 4 below).

2.3. RSA+

As above $n = p \cdot q$ is the product of two large prime numbers. The public key of Bob is just n, as in the Rabin scheme.

Encryption. If Alice wants to encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, she first finds a random number x coprime with $\varphi(n)$ (for example, one can choose $x > \sqrt{n}$ which has passed a strong pseudo-primality test). Then she computes

$$c \equiv m^x \pmod{n}$$

and

$$y \equiv x^2 \pmod{n}$$

and transmits the pair (c, y) to Bob.

Decryption. Using his knowledge of p and q Bob computes the four square roots $x_1, \ldots, x_4$ of y modulo n. For each such square root he tries to compute the inverse

$$u_i \equiv x_i^{-1} \pmod{\varphi(n)}$$

and in case this is possible he computes

$$c^{u_i} \pmod{n}.$$

The original message m is among the c^{u_i}.

We provide an implementation of the encryption and decryption algorithms in PARI and Python on GitHub.[1]

[1] https://github.com/soeren-kleine/RSA_plus-an-RSA-variant.

Remark 1.3. If x was chosen smaller than $\sqrt{n}$, then $y = x^2$ as integers and it would be easy to find the square-root x of y modulo n. Therefore it is important to choose $x > \sqrt{n}$. There is also another reason for this restriction: The parameter x chosen in the encryption step has to be coprime with $\varphi(n)$ in order to be invertible modulo $\varphi(n)$, i.e. otherwise decryption would not work. However, the person who encrypts a message m, using the public key of Bob, does not know $\varphi(n) = (p-1)(q-1)$. It is obvious that x has to be odd. However, depending on the choice of p and q, the integer $\varphi(n)$ is likely to be divisible also by other small prime factors. One could avoid this by restricting to *safe primes* p and q (that is by insisting that both $(p-1)/2$ and $(q-1)/2$ are also prime numbers, which are then so-called Sophie-Germain primes). We don't want to pursue this approach in this note because it considerably restricts the set of possible candidates for p and q.

Instead we use the fact that any *prime number* $x > \sqrt{n}$ is automatically coprime with $\varphi(n)$ (this holds if p and q have been chosen as above, since a prime dividing $\varphi(n)$ is less than or equal to $\frac{q-1}{2} < p \leq \sqrt{n}$). Since a primality proof of a large integer is very expensive we instead suggest to use a strong pseudo-prime x. If the pseudo-primality test used in this step of the algorithm is good enough then the probability of x being not coprime with $\varphi(n)$ is negligibly small. Note that we will suggest a much more efficient way for choosing an appropriate value for x in Section 2.1 below.

Remark 1.4. In the decryption algorithm of RSA+ the Chinese Remainder theorem gives four different square-roots of y modulo n. However, since n is odd, exactly two of them will be even (more precisely, if

$$x_i \in \{1, \ldots, n-1\}$$

is odd, then $n - x_i$ will be even). Therefore the corresponding x_i cannot be inverted modulo $\varphi(n)$. This shows a first difference with the Rabin cryptosystem: We obtain at most two possible decryptions for each ciphertext c, whereas Rabin's decryption algorithm yields 4 different plaintexts which have to be checked. Note that the latter can be reduced to two possible plaintexts by adding the parity of the message to the ciphertext.

In our new scheme, however, it is indeed possible that only one of the four square-roots $x_1, \ldots, x_4$ of y modulo n is coprime with $\varphi(n)$ (note that at least one of the four x_i will be the original choice of x, and therefore will be coprime with $\varphi(n)$ if x was chosen properly (see also Remark 1.3)). We will investigate the number of valid solutions of the decryption algorithm further in Section 4 below.

At the end of this section we remind the reader of two auxiliary results on the efficient computation of square roots modulo some prime number p. For a proof see [Cohen 1993, Section 1.5], for instance.

Lemma 1.5. *Let $p \equiv 3 \pmod 4$ be a prime, and let $y \in (\mathbb{Z}/p\mathbb{Z})^\times$ be a quadratic residue. Then the two square roots of y modulo p are*

$$x \equiv \pm y^{(p+1)/4} \pmod p.$$

Lemma 1.6. *Suppose that $p \equiv 5 \pmod 8$, and let $y \in (\mathbb{Z}/p\mathbb{Z})^\times$ be a quadratic residue.*

Let $\varepsilon = y^{(p-1)/4}$. Then ε is either congruent to 1 or -1 modulo p, and the two square roots of y modulo p are

$$x = \pm \begin{cases} y^{(p+3)/8} & \text{if } \varepsilon \equiv 1 \pmod p \\ 2^{(p-1)/2} \cdot y^{(p+3)/8} & \text{if } \varepsilon \equiv -1 \pmod p. \end{cases}$$

If $p \equiv 1 \pmod 8$ one can use the Tonelli-Shanks algorithm [Shanks 1973; Tonelli 1891] to compute a square root, but this is significantly more costly. In view of these facts, we will restrict to primes which are not congruent to 1 modulo 8 in all that follows.

3. Runtime analysis

In this section we first analyze the efficiency of our scheme from a theoretical point of view and compare it with the classical RSA and Rabin cryptosystems.

The public and secret keys in the RSA+ cryptosystem are exactly as in the Rabin cryptosystem (and therefore the key generation is less costly, as in the RSA scheme, if the encryption exponent e is chosen as in textbook RSA (see Remark 1.2)). An RSA+ ciphertext consists of two integers modulo n, i.e. it twice as long as an RSA or Rabin ciphertext.

For encryption of a plaintext via RSA+, we first have to find a suitable integer x which is coprime with $\varphi(n)$ (this step in theory is as expensive as the choice of the encryption exponent in the public key of the RSA scheme, but in practice it is more difficult because $\varphi(n)$ is not known at this point. Moreover, this step has to be re-done for each encryption). Then we have one RSA encryption step, and one Rabin encryption.

For the decryption of an RSA+ ciphertext, we first have a Rabin decryption step. Then we have to do one or two modular inversions modulo n, followed by one or two RSA decryptions.

It turns out that the most expensive step is the random choice of x in the RSA+ encryption function. We have made some runtime tests with 2000 bit primes p and q, and it turned out that the rather slow search for the random exponent x (including pseudo-primality tests for large integers)

makes RSA+ very much slower than textbook RSA (in our tests, it needed more than 40 times more time than RSA). In order to derive a competitive variant of RSA+, we propose the following improved version of our scheme.

3.1. A practical implementation

We describe a more efficient implementation of our RSA+ scheme. The main change will concern the choice of x, since it is very expensive to find a large number in the range $[\sqrt{n}, n]$ which is a strong pseudoprime. In fact, what we really want is that x is coprime with

$$\varphi(n) = (p-1)(q-1).$$

The main idea is as follows: Fix parameters $\alpha < \beta < \log_2(\sqrt{n})$ and choose a random pseudoprime $\ell \in [2^\alpha, 2^\beta]$ which is considerably smaller than n, but still large enough to ensure that it is very unlikely that ℓ divides the unknown integer $\varphi(n)$. Then we take

$$x = \ell^k$$

where k has been chosen such that x lies in the interval $[\sqrt{n}, n]$ (recall that we want x to be greater than $\sqrt{n}$ since otherwise it could be easily obtained from the congruence $y \equiv x^2 \pmod{n}$).

Algorithm 1: Efficient implementation of RSA+ encryption

Input : $n = pq$ is the product of two primes satisfying $p < q < 2p$; parameters $\alpha < \beta < \log_2(\sqrt{n})$; and a message $m \in (\mathbb{Z}/pq\mathbb{Z})^\times$
Output : $(c, y) \in (\mathbb{Z}/pq\mathbb{Z})^\times \times (\mathbb{Z}/pq\mathbb{Z})^\times$, an RSA+-encryption of m
$b = \texttt{BitLength}(p)$
$\ell = \texttt{RandomPrime}([2^\alpha, 2^\beta])$
$k = \texttt{Random}([\lfloor (b+1)/\log_2(\ell) \rfloor, \lfloor 3/2 * b/\log_2(\ell) \rfloor])$
$x = \ell^k$
$c = m^x \pmod{n}$
$y = x^2 \pmod{n}$
return (c, y)

We give a schematic overview of this improved encryption function in Algorithm 1. Note that the exponent k in this algorithm is chosen such that x lies between $\sqrt{n}$ and n.

How can we ensure that x as chosen above is coprime with $\varphi(n)$? The moderately small prime ℓ might be a divisor of $\varphi(n)$, since the person who encrypts a message does not know $\varphi(n)$. However, if ℓ has sufficiently

many bits, the probability of being a divisor of $\varphi(n)$ is negligibly small. In our tests we have chosen ℓ to be a random prime in the range $[2^{150}, 2^{190}]$ and never encounter this exceptional case. Moreover, there are enough primes in this interval to make an exhaustive search for x very expensive. On the other hand, for integers of this size a probabilistic test for pseudo-primality is still quite fast, so that the choice of ℓ does not slow down the encryption function of RSA+ too much. To determine optimal parameters α and β would need further investigation. Moreover, these parameters have to be checked regularly for their safety. For this study we have just done some ad-hoc tests on the potential runtime of a brute-force search for x when these parameters were used. If a potential attacker has much better resources, then the parameters α and β should be enlarged accordingly.

It turned out that this way of choosing x speeds up the RSA+ functions by more than a factor of ten (for 2000 bit primes p and q; the effect is even larger for larger bit sizes). We also implemented an improvement on the decryption function: There are at most two square roots x_1 and x_2 of y modulo n which are coprime with $\varphi(n)$. We do not consider further the remaining two square roots, i.e. we return only two possible messages instead of four (and in many cases one can indeed sort out one of the two remaining messages). Moreover, we use the Chinese Remainder Theorem and our knowledge of the prime factors p and q of n in order to efficiently compute the powers $c^{x_1^{-1}}$ and $c^{x_2^{-1}}$ modulo n. This speeds up the decryption moderately.

3.2. Runtime comparison

In our final runtime tests we generated 1000 keys and for each such key we chose 100 messages which we encrypted and afterwards decrypted using our RSA+ functions, and also via textbook RSA and textbook Rabin. For RSA we used the speed-up from Remark 1.2.

The computations were done on a customary laptop with the PARI programs which we published in our github repository[2]. The results are depicted in the following table. Here *bit length* means the approximate bit length of p and q. More precisely, if the bitlength is b, then p lies between 2^b and 2^{b+1}, and q lies between 2^b and 2^{b+2}. For example, if $b = 2000$,

Bit length/operation	RSA+	RSA	Rabin
1500 bit – encryption	5.34	0.05	0.004
1500 bit – decryption	8.04	5.97	3.110
2000 bit – encryption	15.86	0.09	0.005
2000 bit – decryption	23.38	14.02	6.085

[2]https://github.com/soeren-kleine/RSA_plus-an-RSA-variant.

then $n = p \cdot q$ will have between 4000 and 4003 bits. Moreover, we considered only prime numbers p and q which are not congruent to 1 modulo 8, since for such primes p and q particularly efficient routines for computing square roots modulo p and q exist (see Lemmas 1.5 and 1.6 above). In the following table we depict times in milliseconds on average (for one encryption, respectively, decryption).

Bit length/operation	RSA+	RSA	Rabin
3000 bit encryption	59.79	0.31	0.019
3000 bit decryption	84.11	64.31	30.432

The time for the key generation phase for RSA+ was 1487.27 milliseconds on average for 1500 bit primes and 5726.66 milliseconds on average for 2000 bit primes.

We also did a smaller test with 3000 bit primes (here we considered only 40 pairs of keys, each used for the encryption and decryption of 50 messages). Here are the results:

The time for the RSA+ key generation phase was 10687.15 milliseconds on average.

These runtimes were compared directly in PARI. In fact, we have used many PARI functions in our python code, too, since typically the PARI routines performed much faster than similar algorithms from other libraries (for example, we compared runtimes with the routines from the sympy library). The results from the python runtime tests were in accordance with the above results.

The above tests show that a typical RSA+ decryption is almost as fast as one RSA decryption (more precisely, it is slower by a factor of about 30 to 50%). On the other hand, one RSA+ encryption is much slower than its counterpart in the RSA and Rabin schemes. In fact, the encryption is almost as costly as a RSA+ decryption, whereas encryption is much cheaper in the RSA and Rabin schemes. This is due to the choice of a secret exponent x which is part of the encryption phase of RSA+. Summing encryption and decryption times, we can conclude that the fast variant of our RSA+ scheme is slower than textbook RSA by a factor of only 2 to 3 and slower than Rabin by a factor of about 4–6.

Remark 2.1. In our tests we compared the textbook versions of the three cryptographic procedures. For a real-world application, each of the three schemes would have to be enhanced by, for example, some padding routine (one must not encrypt plaintexts directly in order to prevent dictionary attacks). Since these enhancements are more or less the same for all the three schemes, a comparison of the running times of the real-world versions of the three cryptosystems would yield even closer results.

4. On the security of the RSA+ cryptosystem

In the last section we have seen that RSA+ is slower than both the RSA and Rabin cryptosystems. In the next two sections we describe advantages of RSA+ over these two schemes. The current section focusses on security issues. We will prove that RSA+ is at least as secure as RSA (see Theorem 3.4 below). The ultimate goal would be to prove that the security of RSA+ depends only on the problem of factoring n. We argue that in some sense RSA+ seems to be closer to this goal than RSA (we will make this more precise below). First we introduce some terminology.

4.1. Security definitions

Any attacker of a public-key encryption scheme can use the public keys to encrypt arbitrary messages. In this subsection we recall, for the convenience of the reader, the notion of *chosen-plaintext attack* (CPA) *security* of a public-key encryption scheme (see [Katz and Lindell 2008, Section 12.2.1]). This security model is typically defined in terms of a game between a challenger (let's call him Bob) and an adversary (let's call him Oscar).

1. Bob runs the setup algorithm to generate the secret and public keys.
2. Oscar is given the public key; he chooses two messages m_0 and m_1 of equal length and gives them to Bob.
3. Bob randomly chooses a bit $i \in \{0, 1\}$ and encrypts the message m_i. He sends the corresponding ciphertext c back to Oscar.
4. Oscar has to decide whether c is the encryption of m_0 or m_1.

The cryptosystem is called CPA-secure if Oscar cannot efficiently guess the correct value of i in Step (4) with a probability which is significantly larger than $1/2$. Here efficiently usually means that Oscar has to make a decision within a time which depends on the size of the parameters of the cryptosystem in a polynomial way (the space needed for his computations usually is also limited). Note that for a public-key cryptosystem the property of being CPA-secure is equivalent to *eavesdropping indistinguishability*, see e.g. [Katz and Lindell 2008, Section 12.2].

Like textbook RSA (see [Hinek 2010, Section 1.3.5]), the RSA+ cryptosystem in its pure form is not CPA-secure. Though it is probabilistic in nature (unlike textbook RSA!), the Jacobi symbol $\left(\frac{m}{n}\right)$ of a plaintext m is revealed by the ciphertext (c, y). It is indeed straightforward to see that one has $\left(\frac{m}{n}\right) = \left(\frac{c}{n}\right)$, since the encryption exponents x used in RSA+ are all odd integers. As explained in [Ferguson and Schneier 2003, Section 12.5], a solution to avoid this problem is to use only squares as plaintexts.

4.2. On the security of RSA, Rabin and RSA+

In this subsection, we let $n = pq$ be as in Section 1. We first formulate a series of problems and then prove relations between these problems.

Definition 3.1. Let n be as above, and let $c, y \in (\mathbb{Z}/n\mathbb{Z})^\times$.

- (**Fact**(n)) Construct an oracle which can compute the two prime factors p and q of n.
- (**RSA**(n)) Construct an oracle that can decrypt *any* RSA encrypted message modulo n.
- (**RSA**(c, n)) Construct an oracle that can decrypt the given RSA encrypted message $c \in (\mathbb{Z}/n\mathbb{Z})^\times$.
- (**Rabin**(n)) Construct an oracle which can decrypt *any* Rabin encrypted message modulo n.
- (**Rabin**(c, n)) Construct an oracle which can decrypt the given Rabin encrypted message $c \in (\mathbb{Z}/n\mathbb{Z})^\times$.
- (**RSA** $+ (n)$) Construct an oracle that can decrypt *any* RSA+ encrypted message modulo n.
- (**RSA** $+ (c, y, n)$) Construct an oracle that can decrypt the given RSA+ encrypted message (c, y).

For example, by an oracle for problem (**RSA** $+ (n)$), we mean that there is some kind of unknown method or algorithm that takes as input any given tuple (c, y), where $c = m^x \pmod n$ and $y = x^2 \pmod n$ as above, and returns m and potentially a second unintelligible message. Note that it is unreasonable to expect that the oracle always returns only m, as this would mean that it can distinguish between intelligible and unintelligible messages. Similarly, an oracle for (**Rabin**(n)) will output, for any given $c \in (\mathbb{Z}/n\mathbb{Z})^\times$, the four square roots of c modulo n (but it cannot decide which of the square roots corresponds to the original message m).

First note that there are some obvious relations between the above problems. For example, if one can solve (**RSA**(n)) for n, then the corresponding oracle can of course solve (**RSA**(c, n)) for any $c \in (\mathbb{Z}/n\mathbb{Z})^\times$. In the following we will write such a relation as

$$(\textbf{RSA}(n)) \Rightarrow (\textbf{RSA}(c, n)) \ \forall c \in (\mathbb{Z}/n\mathbb{Z})^\times.$$

Similarly, (**Rabin**(n)) $\Rightarrow$ (**Rabin**(c, n)) and (**RSA** $+ (n)$) $\Rightarrow$ (**RSA** $+ (c, y, n)$) for all $c, y \in (\mathbb{Z}/n\mathbb{Z})^\times$.

More importantly, we have the following well-known

Theorem 3.2. (**Rabin**(n)) $\Longleftrightarrow$ (**Rabin**(c, n)) $\Longleftrightarrow$ (**Fact**(n)) $\Rightarrow$ (**RSA**(n)) *for each c, n as above. In other words, breaking Rabin is equivalent to factoring n, and factoring n is enough for breaking RSA.*

Proof. It is obvious that an oracle which can compute the prime factors p and q of n can solve both problems $(\mathbf{RSA}(n))$ and $(\mathbf{Rabin}(n))$. For the non-trivial implication

$$(\mathbf{Rabin}(c,n)) \Rightarrow (\mathbf{Fact}(n))$$

we refer the reader to [Rabin 1979, Section 4]. ■

Corollary 3.3. $(\mathbf{Rabin}(n)) \Rightarrow (\mathbf{RSA}(n))$, *i.e. the Rabin cryptosystem is at least as secure as RSA.*

It is *conjectured*, but not known, that in fact

$$\mathbf{RSA}(n) \Longleftrightarrow (\mathbf{Fact}(n)),$$

which would also imply that breaking $(\mathbf{RSA}(n))$ and $(\mathbf{Rabin}(n))$ are equivalent.

Now we turn to RSA+. In the following result we assume that there is an oracle that can decrypt RSA+ messages. We prove that the same oracle also solves the problem $(\mathbf{RSA}(n))$. This means that breaking RSA+ is at least as difficult as breaking the RSA cryptosystem, i.e. in some sense RSA+ is at least as secure as RSA.

Theorem 3.4. *For any n we have* $(\mathbf{RSA}+(n)) \Rightarrow (\mathbf{RSA}(n))$.

Proof. Suppose we have an oracle that is able to decrypt RSA+ messages as described above. Now, if (n,e) is an RSA public key and $c \equiv m^e \pmod{n}$ is a given RSA ciphertext, then we choose an integer $r > 0$ such that $\sqrt{n} < e^r < n$ and send $(c^{(e^{r-1})}, (e^r)^2)$ to the oracle. Note that $c^{(e^{r-1})} \equiv m^{(e^r)} \pmod{n}$. Then the oracle will successfully return m and potentially a second possible message m'.

Now even if m' were a meaningful message, we could single out m by checking whether $c \equiv m^e \pmod{n}$ or $c \equiv (m')^e \pmod{n}$.

Remark 3.5. The above represents the worst case scenario i.e. we are assuming that the oracle can decipher all RSA+ ciphertexts and then it follows that any RSA ciphertext can also be deciphered. We will show – under reasonable hypotheses – that the average complexity of RSA+ is also at least as good as that of RSA. See Remark 4.1 below.

Remark 3.6. Let (c,y) be an RSA+ ciphertext. It is quite common that the decryption procedure yields two possible plaintext messages (see also the next section for more details on how likely it is that this result will happen). If an attacker is given two tuples (m_1,x_1) and (m_2,x_2) such that $m_1^{x_1} \equiv c \equiv m_2^{x_2} \pmod{n}$ and such that both x_1 and x_2 are square roots of y modulo n, then he could easily factor n. This is because the two square-

roots of y must satisfy $x_1 \not\equiv -x_2 \pmod{n}$ since $-x_1$ will be even and thus will not be coprime with $\varphi(n)$.

It is not clear to us whether the knowledge of two potential plaintext messages m_1 and m_2 without knowing the corresponding encryption exponents x_1 and x_2 would suffice for factoring n. If this was true, then breaking RSA+ would be equivalent to factoring n, since receiving two possible plaintext messages can always (see the next section) be ensured via suitable choice of the parameters.

4.3. Attacks on RSA

If $\mathbf{RSA}(n)$ is equivalent to factoring n, then the same holds true for $\mathbf{RSA} + (n)$. Though we cannot prove the equivalence of $\mathbf{RSA} + (n)$ and $\mathbf{Fact}(n)$, Theorem 3.4 suggests that $\mathbf{RSA} + (n)$ could be 'closer' to $\mathbf{Fact}(n)$ than $\mathbf{RSA}(n)$ if these problems turned out to be not equivalent. Note that most (if not all) known attacks on RSA make use of the public exponent e. If one tries to apply such an attack to RSA+, however, one would have to compute the exponent x first, for which one would have to break Rabin's cryptosystem; but the latter is known to be equivalent to $\mathbf{Fact}(n)$.

The same is true if the attacker manages to find an inverse $\tilde{x}$ to x modulo $\varphi(n)$, by which he could easily compute the clear text m from a ciphertext c. But then $\tilde{x}^2$ was the inverse of $y = x^2$ modulo $\varphi(n)$, and it is well-known that the knowledge of both y and its inverse modulo $\varphi(n)$ allows for efficient factorization of n.

In the following we briefly discuss several attacks against RSA which do not carry over directly to our suggested implementation of the RSA+ scheme.

- Choosing a public exponent e which is too small makes RSA vulnerable against *small exponent attacks*. More precisely, in this case insufficiently padded messages m might be small enough such that $m^e < n$. Then the message can easily be read off the ciphertext c by any attacker. In the RSA+ scheme only exponents x with $x > \sqrt{n}$ are allowed. Therefore the small exponent attack described above is not practical against RSA+.
- If a message m is sent (without any kind of randomized padding) to many different people all of whom use the same rather small encryption exponent e (but different moduli n_i), then an attacker can use the Chinese Remainder Theorem in order to push the small exponent attack above a little further (*Håstad's attack*, see [Galbraith 2012, Example 24.4.1]). Again, this attack is not possible against the RSA+ scheme.
- If the *secret* decryption exponent d is too small (e.g. $d < \sqrt[4]{n}$), then RSA is vulnerable to *Wiener's attack*. The original attack by Wiener used the

continued fractions expansion of $\frac{e}{n}$ to approximate the private exponent d. The attack has been extended to larger exponents d using *lattice-based approaches* [Coppersmith 1996]. We are not aware of variants of the above attacks which would work if the public exponent e was not known.

- An adversary who knows a fraction of the secret exponent bits (e.g. via a side-channel attack) can launch a so-called *partial key exposure attack* [Boneh, Durfee, and Frankel 1998]. With such side channel attacks the adversary gets either the most significant or the least significant bits of d. Using the algorithms from [Boneh, Durfee, and Frankel 1998] the entire secret exponent d can be recovered in polynomial time. The attacks from [Boneh, Durfee, and Frankel 1998] only work for small public exponents $e \leq \sqrt{n}$, and the algorithms make heavy usage of the known exponent e. Although there exist improvements on the original work for larger public exponents (see for example [Blömer and May 2003]) we are not aware of any variant of this attack which would work without the knowledge of e.

We conclude with two final remarks.

Remark 3.7.

1. In modern RSA protocols, RSA-OAEP (Optimal Asymmetric Encryption Padding) is used in order to prevent padding attacks like Bleichenbacher's attack [Bleichenbacher, 1998]. Using OAEP one can obtain a cryptosystem that is secure against chosen ciphertext attacks (CCA), see [Bellare and Rogaway 1995] and [Boneh 2001]. The same technique can be combined with the RSA+ scheme in order to make it CCA-secure.
2. Finally, we note that RSA+, like RSA, cannot be used for post quantum cryptography since an attacker having access to a quantum computer can just factor the modulus n via Shor's algorithm [Shor 1997].

5. On the number of possible clear texts output by the decryption function

A major drawback of Rabin's cryptosystem is the fact that decryption leads to four possible messages. If the original message was a text, it is usually easy to make the right choice. However, if the message was a number, this is much more difficult. So it is an advantage of RSA+ over Rabin that it only leads to at most two possible messages. Of course, this can also be achieved for Rabin's cryptosystem by simply adding the parity of the message to the ciphertext. This then always leads to two possible messages.

If both primes dividing n are congruent to 3 (mod 4), then a work-around for Rabin has been suggested by Williams [Williams 1980]; see also [Galbraith 2012, Chapter 24.2]. This enhancement works under the assumption that $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$, but one has to restrict to messages $m < n/8 - 1$. The Rabin–Williams decryption function has a 25% probability of failure if the ciphertext has not been prepared via the Williams enhancement [Galbraith 2012].

On the one hand, breaking the Rabin–Williams scheme is still equivalent to factoring n (see [Galbraith 2012, Theorem 24.2.15]); on the other hand the redundancy scheme proposed by Williams significantly slows down both the encryption and decryption functions since one has to compute certain Jacobi symbols. Moreover, the Rabin–Williams algorithm only works under the above congruence conditions on p and q. Therefore the key generation phase takes about three times as long as the corresponding step in the RSA, Rabin and RSA+ schemes.

A similar method works for RSA+ without shrinking the plaintext space: If Bob chooses x with Jacobi symbol $\left(\frac{x}{n}\right) = +1$, then the second odd square root x' of $y = x^2 \pmod n$ has $\left(\frac{x'}{n}\right) = -1$. This follows from the fact that -1 is not a quadratic residue mod p nor mod q. The RSA+ decryption procedure can be modified to ignore the second odd square root of $y \pmod n$ and proceed with $u \equiv x^{-1} \pmod n$. The resulting RSA+ scheme is slower than Rabin–Williams by a factor of roughly 3 to 4, but can handle slightly longer messages. However, we do not recommend to use this version of RSA+, as it is more efficient to choose primes p and q such that there is very high probability that decryption only produces one message. See below for a heuristic.

Moreover, there is the following well-known chosen ciphertext attack against Rabin–Williams: Any message will be encrypted as x^2, where x is an even integer with Jacobi symbol $\left(\frac{x}{n}\right) = +1$. If an adversary asks to decipher $c = x^2$ for some x with $\left(\frac{x}{n}\right) = -1$, he will receive a message which allows him to find an even square root x' of c such that $\left(\frac{x'}{n}\right) = +1$. This can be used to factor n and thus reveals the private key. This attack does not carry over to RSA+: The adversary can choose a message m and an exponent x. If he asks to decipher the pair (m^x, x^2), he will get m and m' such that $(m')^{x'} = m^x$, where $(x')^2 = x^2$. Here x' is the second root that we would need to factor n. To find x', however, he had to solve the discrete logarithm problem, which is hard. So he only learned how to decipher a single ciphertext.

In contrast to Rabin–Williams, even if one of the primes dividing n is congruent to 1 (mod 4) or if x is not chosen to be a square mod n, the RSA+ decryption scheme still sometimes only produces one possible

message. This happens whenever the second odd square root of $y \pmod n$ is not coprime with $\varphi(n)$. The probability of a randomly chosen integer being divisible by a prime ℓ is $1/\ell$. Let us denote the set of odd primes dividing $\varphi(n)$ by $\mathcal{L}_n$. Then heuristically, the possibility that decryption only produces one possible message is given by the formula

$$1 - \prod_{\ell \in \mathcal{L}_n} \left(1 - \frac{1}{\ell}\right). \tag{4.1}$$

For instance, if $\varphi(n)$ is divisible by 3, 5 and 7, then this probability is at least

$$1 - \frac{2}{3} \cdot \frac{4}{5} \cdot \frac{6}{7} = \frac{19}{35} > \frac{1}{2}.$$

So we expect that there is only one possible message at least in every second case. In a series of tests we observed an accuracy of formula (4.1) of roughly 99%. We used pairs (p, q) of primes of bit length 1500 and for each of a few hundred pairs of such primes we RSA+-encrypted and decrypted 10^4 messages. Then we checked whether decryption yields one or two possible results.

In other words, by choosing n appropriately one can enlarge the possibility that decryption returns only one message. In particular small prime factors of $\varphi(n)$ have relatively large impact. Of course, if all prime factors of either $p-1$ or $q-1$ are small, this makes the cryptosystem vulnerable to factoring methods such as Pollard's $p-1$ method [Pollard 1974]. But choosing the primes such that $p-1$ is divisible by, say, 3 and 7, whereas $q-1$ is divisible by 5, is not a big restriction.

On the other hand, there do always exist choices for x which result in two possible plaintext messages returned by the decryption procedure. Indeed, it suffices to exclude two possible residue classes modulo each prime divisor $\ell \in \mathcal{L}_n$ (more precisely, we want x and $n-x$ to be not divisible by ℓ, i.e. we exclude the residue classes of 0 and n modulo ℓ). This means that by choosing random values for x we have a good chance to find exponents which are not contained in any of the 'forbidden' residue classes (recall that the person who encrypts a message does not know the prime factors p and q). Note that two possible plaintext decryptions of the same ciphertext might be used for factorization of n; see Remark 3.6.

Remark 4.1. We discuss the average complexity of RSA+ compared to that of RSA. Fix a modulus n. Let $\mathcal{C}$ be the set of all pairs (c, e) where e is a valid RSA exponent and $c = m^e$ is a corresponding RSA ciphertext and let $\mathcal{C}' \subseteq \mathcal{C}$ be the subset of all $(c, e) \in \mathcal{C}$ such that $e \in [\sqrt{n}, n]$. Similarly, we let $\mathcal{C}^+$ be the set of all valid RSA+ ciphertexts (c, y), that is $c = m^e$ as before and $y = e^2$ where $e \in [\sqrt{n}, n]$. Then we have a surjective map

$$\begin{aligned} \text{plus} : \mathcal{C}' &\to \mathcal{C}^+ \\ (c, e) &\mapsto (c, e^2). \end{aligned}$$

As explained above, for a given $(c, y) \in \mathcal{C}^+$ there are one or two preimages under this map. If $a \in [0, 1]$ denotes the probability that there are two preimages, then one has $|\mathcal{C}'| = (1 + a)|\mathcal{C}^+|$. Now suppose that there is an oracle that can decipher b% of all RSA+ ciphertexts. Let $U^+ \subseteq \mathcal{C}^+$ be the set of all ciphertexts that the oracle can decipher, and let $U' = \text{plus}^{-1}(U^+)$ be its preimage. It follows as in the proof of Theorem 3.4 that the oracle can decipher an RSA ciphertext c whenever $(c, e) \in U'$.

Let $(c, y) \in \mathcal{C}^+$. If we assume that the events '$(c, y) \in U^+$' and '(c, y) has two preimages under plus' are stochastically independent, then the probability of having two preimages for a random $(c, y) \in U^+$ is still a. Then

$$\frac{|U'|}{|\mathcal{C}'|} = \frac{(1 + a)|U^+|}{(1 + a)|\mathcal{C}^+|} = \frac{|U^+|}{|\mathcal{C}^+|} = \frac{b}{100},$$

i.e. the probability that the oracle can decipher a random $(c, e) \in \mathcal{C}'$ is b%.

Of course, there are two extreme scenarios: (i) it happens that each $(c, y) \in \mathcal{U}^+$ has exactly one preimage under plus, i.e. the oracle can only decipher $(c, y) \in \mathcal{C}^+$ that possess only one preimage, and (ii) each $(c, y) \in \mathcal{U}^+$ possesses two preimages. In case (i) the oracle can only decipher $(1 + a)^{-1}b\%$ of all possible RSA messages in $\mathcal{C}'$, whereas in case (ii) it can decipher $2(1 + a)^{-1}b\%$. Note that the latter is strictly greater than b% unless $a = 1$ (in which case both p and q must be Fermat primes).

If we do not make any assumption on the oracle, then some of the pairs $(c, y) \in \mathcal{U}^+$ will have one preimage under plus, whereas the remaining part of the decipherable ciphertexts has two preimages under plus. So assume that the oracle can decipher $b'\%$ of all pairs $(c, e) \in \mathcal{C}'$, where

$$(1 + a)^{-1}b \leq b' \leq 2(1 + a)^{-1}b$$

by the above considerations. Let $(c, e) \in \mathcal{C} \setminus \mathcal{C}'$, that is $e < \sqrt{n}$. We randomly choose an odd integer t such that $et \in [\sqrt{n}, n]$. If t happens to be coprime with $\varphi(n)$ then the pair $(c^t, et) \in \mathcal{C}'$ consists of a valid RSA exponent et and the corresponding encryption $c^t = m^{et}$ of the original message m. As the probability for t of being coprime with $\varphi(n)$ is relatively large, we find such a t after not too many attempts. The problem now is that the resulting pair in $\mathcal{C}'$ is not completely random (but this approach grants much more freedom than choosing t to be a power of e as in the proof of Theorem 3.4). In this way, we may obtain any pair in the subset $\{(c', e') \in \mathcal{C}' \mid e' \text{ is composite}\}$ of $\mathcal{C}'$. To see this, first note that the map $c \mapsto c^t$ is bijective on $(\mathbb{Z}/n\mathbb{Z})^\times$ as t is coprime with $\varphi(n)$. Moreover, for every composite $e' \in [\sqrt{n}, n]$ there is a prime $\ell < \sqrt{n}$ dividing e' so that we can write $e' = et$ with $e = \ell$ and $t = e'/\ell$.

However, we will never get a pair (c', e'), where e' is prime. Moreover, we get a pair (c', e') the more likely the more divisors e of e' with $e < \sqrt{n}$ exist.

If we assume in addition that these events are stochastically independent of the event that the oracle can decipher a pair in $\mathcal{C}'$, then we can conclude that the oracle can indeed decipher $b'\%$ of all RSA messages for the given modulus n.

6. Conclusion

In this article we proposed a novel public-key cryptosystem which combines the encryption methods of the well-known RSA and Rabin cryptosystems. We have seen that the algorithm, if implemented efficiently, performs about 3 times as slow as the RSA cryptosystem (the factor is close to 2.5 on average) and it takes up to about four times the runtime of the Rabin cryptosystem. Since a public-key cryptosystem is typically used only for exchanging keys for a much more efficient symmetric scheme, this slower runtime does not appear to be a crucial drawback.

On the other hand, breaking RSA+ is at least as difficult as breaking the RSA scheme, and it seems reasonable to expect that breaking RSA+ is closer to factoring the large modulus n, which is widely believed to be a very hard mathematical problem. When compared to the original Rabin scheme, our algorithm has the benefit of producing at most two (compared to four) possible decrypted messages from a fixed ciphertext – in many situations we obtain in fact a unique decrypted plaintext. We also compared the RSA+ scheme to the Rabin–Williams enhancement of the original Rabin cryptosystem.

Acknowledgments

The fourth named author would like to thank Kimball Martin, Marco Streng and Larry Washington for useful comments on an early draft of the manuscript. Moreover, we thank the anonymous referees for several useful suggestions to significantly improve the article.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

No funding was received for conducting this study.

About the authors

Sören Kleine received a Ph.D. in mathematics from University of Göttingen. He is currently a Post-Doc at University of the Bundeswehr Munich. His research interests include algebraic number theory and cryptography.

Andreas Nickel holds a PH.D. in mathematics from University of Augsburg. He is full professor for mathematics at University of the Bundeswehr Munich. His research focus is on number theory and related topics.

Torben Ritter received a B.Sc. in computer science from University of the Bundeswehr Munich. He is currently a student of Cyber Security at University of the Bundeswehr Munich.

Krishnan Shankar holds a Ph.D. in mathematics from the University of Maryland. He is full professor at James Madison University working, among others, on Riemannian Geometry, Lie groups, and Cryptography.

Data availability statement

The code for the used PARI and Python programs is available at GitHub.

References

Bellare, M., and P. Rogaway. 1995. Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT'94*, ed. Alfredo De Santis, 92–111. Berlin, Heidelberg: Springer Berlin Heidelberg.

Bleichenbacher, D. 1998. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1. In *Advances in Cryptology—CRYPTO '98*, ed. Hugo Krawczyk, 1–12. Berlin, Heidelberg: Springer Berlin Heidelberg.

Blömer, J., and A. May. 2003. New partial key exposure attacks on RSA. In *Advances in Cryptology – CRYPTO 2003*, ed. Dan Boneh, 27–43. Berlin, Heidelberg: Springer Berlin Heidelberg.

Boneh, D. 2001. Simplified OAEP for the RSA and Rabin functions. In *Advances in Cryptology—CRYPTO 2001*, ed. Joe Kilian, 275–91. Berlin, Heidelberg: Springer Berlin Heidelberg.

Boneh, D., G. Durfee, and Y. Frankel. 1998. An attack on RSA given a small fraction of the private key bits. In *Advances in Cryptology—ASIACRYPT'98*, ed. Kazuo Ohta and Dingyi Pei, 25–34. Berlin, Heidelberg: Springer Berlin Heidelberg.

Cohen, H. 1993. *A course in computational algebraic number theory*. Vol. 138 of *Graduate texts in mathematics*. Berlin: Springer-Verlag.

Coppersmith, D. 1996. Finding a small root of a univariate modular equation. In *Advances in cryptology—EUROCRYPT '96 (Saragossa, 1996)*. Vol. 1070 of *lecture notes in computer science*, 155–65. Berlin: Springer.

Ferguson, N., and B. Schneier. 2003. *Practical cryptography*. Hoboken, NJ: John Wiley & Sons.

Galbraith, S. D. 2012. *Mathematics of public key cryptography*. Cambridge: Cambridge University Press.

Hinek, M. J. 2010. *Cryptanalysis of RSA and its variants*. Boca Raton, FL: Chapman & Hall/CRC Cryptography and Network Security. CRC Press.

Katz, J., and Y. Lindell. 2008. *Introduction to modern cryptography*. Boca Raton, FL: Chapman & Hall/CRC Cryptography and Network Security. Chapman & Hall/CRC.

Pollard, J. M. 1974. Theorems on factorization and primality testing. *Proceedings of the Cambridge Philosophical Society*. 76:521–8.

Rabin, M. O. Digitalized signatures and public-key functions as intractable as factorization. MIT Laboratory for Computer Science, MIT-LCS-TR 212, 1979.

Rivest, R. L., A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21 (2):120–6. doi: 10.1145/359340.359342.

Schneier, B. 2015. *Applied cryptography. Protocols, algorithms and source code in C. 20th anniversary edition. With a foreword by Whitfield Diffie*. 2nd ed. Hoboken, NJ: John Wiley & Sons, reprint of the 1996 edition.

Shanks, D. 1973. Five number-theoretic algorithms. In *Proceedings of the Second Manitoba Conference on Numerical Mathematics (Univ. Manitoba, Winnipeg, Man., 1972)*. Vol. VII of Congress. Numer, 51–70. Winnipeg, MB: Utilitas Math.

Shor, P. W. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* 26 (5):1484–509. doi: 10.1137/S0097539795293172.

Tonelli, A. 1891. Bemerkung über die Auflösung quadratischer Congruenzen. *Nachrichten von der Königl. Gesellschaft der Wissenschaften und der Georg-Augusts-Universität zu Göttingen* 1891:344–6.

Williams, H. C. 1980. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*. 26:726–9.