RSA Cryptosystem and its Limitations

Muhammad Furrukh Asif
Anish Kumthekar

December 11th 2019

Abstract

The RSA algorithm, developed in 1977 by Rivest, Shamir, and Adlemen, is an algorithm for public-key cryptography. The purpose of the RSA algorithm is to provide privacy and ensure reliability of data. Topics in elementary number theory such as Euler's phi function and Fermat's Little Theorem are the fundamentals of RSA encryption. In this paper, we first look at how the RSA algorithm functions and its significance. We then focus on the encrypting aspect of RSA and the weaknesses in its implementation. Particularly we explore the attacks that take advantage of a low public exponent and present the partial key exposure attack with the underlying mathematics in detail.

Introduction

While Number Theory can be thought of as a subject area in Mathematics with little practical usage, RSA proves that this is not the case. Before the RSA, if there needed to be an exchange of critical information between two parties, they would first have to meet and agree upon methods of encoding and decoding the information. What RSA accomplished is that now, instead of meeting the other party and discussing ways to decrypt the message beforehand, one could make their encryption procedure publicly available without sacrificing the method of decoding the information. This means that given that the receiver holds a private decryption key, they can use it to decrypt an encrypted message that is sent. Even if the message is intercepted, it will be impossible to decrypt the message without the decryption key.

This approach was first proposed by Whitfield Diffie and Martin Hellman in 1976 in their paper titled, "New Directions in Cryptography". The paper took a novel approach by suggesting that different keys could be used to perform encryption and decryption rather than the same key. This way the decryption could be kept secret, while allowing for the encryption to appear in public. This concept was known as 'public key cryptography'. While in 1970's there wasn't much need for cryptography, apart from governments using this approach, it was particularly significant for the growing internet in the following decades. The way in which Diffie and Hellman saw their method being implemented was when a site needed to send data to different computers. Reversing the order of decryption, they surmised that sites which possess the decryption key would decrypt the data, and then send it out to computers, who can all access the message once they receive it since they all can access the public encryption key.

The full RSA algorithm would come a year later as a solution to the integer factorization problem. The RSA algorithm was invented by Ron Rivest, Adi Shamir and Len Adleman. Today the cryptosystem has wide ranging applications from emails which use the digital signatures of RSA to securely transmit messages to credit card payment systems. While RSA is still seen to be secure, there have been multiple ways designed to take down the RSA. From timing attacks to those that take advantage of low values of the exponents, attacks today can crack the RSA however none of them are seen as devastating. They mostly warn against the improper use of RSA.

In this paper we will be exploring one of these attacks, the partial key exponent attack. The attack will be derived by using a main theorem and several supporting theorems, after which an example will be provided. Before proceeding onto the attack, however, we will explain the methodology of the RSA cryptosystem and provide an example of how it works.

How does the RSA cryptosystem work?

To encrypt a message using the RSA algorithm, one can use the encryption procedure outlined below: Firstly you have to choose a number n, which is a product of large primes p and q. From this choice, we can represent the textual or any other kind of message to be sent in numeric form. We assign some values to the message from 0 to $n-1$, calling it M, using some standard representation. The number M or the blocks M is broken into should be less than the value of n for the algorithm to work properly. For letters this representation is usually the letter's number in the alphabet from 1 to 26.
Next, we choose a public encryption exponent, e, which will be used to encrypt the message and transform it into ciphertext, C. Specifically the ciphertext is given as:

$$C \equiv E(M) \equiv M^e (\text{mod n})$$

Now, in order to decrypt this message, we raise the ciphertext to a different power d, which is the users private decryption exponent in this case. So the decryption is as follows:

$$D(C) \equiv C^d (\text{mod n})$$

This decrypted message should yield the original message after it has been decrypted, so $D(C) = M$ in order for the decryption to have worked.
Each user in the RSA algorithm keeps their public key, e, available for all to view, while their private key, d, is used to decrypt messages sent to the user. Now in order to choose the public and private keys, e and d, one has to first choose a value for d such that

$$\gcd(d, \phi(n)) = 1$$

where $\phi(n)$ is Euler's phi function (the phi function computes the number of positive integers less than n, that are relatively prime with n) applied to n. Next, e, can be found out from the following congruence:

$$e \cdot d \equiv 1 (\text{mod } \phi(\text{n}))$$

In order to prove why the above is true we can use a Theorem proposed by Euler:

Theorem. If m is a positive integer and a is an integer with $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 (\text{mod } m)$
To see how RSA makes use of the theorem above, we can apply encryption and decryption to some message M, which is assumed to be one large number. The message sent to a receiver by some user in this case can be encrypted by raising it to power e (mod n) which is $M^e(\text{mod n})$ where n is a product of large primes p and q. Now in order to decipher the message, the receiver you raise the encrypted message to the power d using your private decryption key d. This, then leads to $(M^e)^d(\text{mod n}) = M^{ed}(\text{mod n})$. Now assuming that $e \cdot d \equiv 1(\text{mod } \phi(\text{n}))$, we can write M^{ed} as:

$$M^{ed} = M^{1 \text{ mod } \phi(\text{n})} = M^{0 \text{ mod } \phi(\text{n})} \cdot M = M^{\phi(n)} \cdot M$$

And now

$$M^{ed}(\text{mod n}) = M^{\phi(n)} \cdot M(\text{mod n})$$

Now utilizing Euler's which states $a^{\phi(m)} \equiv 1(\text{mod } m)$ we get $M^{\phi(n)} \equiv 1(\text{mod } n)$ so the above equation reduces to

$$M^{\phi(n)} \cdot M(\text{mod n}) = M(\text{mod n})$$

and since $M < n$ we have the following result:

$$M^{ed}(\text{mod n}) = M$$

This above fact is important, since decrypting and encrypted message should yield the same message that was sent. And in order to arrive at this fact, and generate the exponents e and d it was essential to assume $e \cdot d \equiv 1(\text{mod } \phi(\text{n}))$.

RSA cryptosystem example

A simplified example that shows RSA's working can be demonstrated here: Given values for p, q, n and d we can look to encipher and decipher a message. Certain values can be assigned to the variables: p = 53, q = 61, n = 53·61 = 3233 and d = 2753. These numbers are arbitrarily picked, while it is made sure that $gcd(\phi(3233), d) = 1$, where $\phi(3233) = 52 * 60 = 3120$. We now choose the d value to be sufficiently large, so that it can't be easily guessed and to make our calculations with a smaller e possible. From the chosen d value which satisfies the gcd, we can then obtain a value for e from the congruence $ed \equiv 1$ $(\text{mod } \phi(n))$, plugging in for the values we get $e \cdot 2753 \equiv 1 \pmod{3120}$. From testing out different multiples of 2753, we obtain the value of e to be 17 since $2753 \cdot 17 \equiv 1 \pmod{3120}$.
Let's pick a word to be our message. In this case the word is 'Math'. Every letter in this word can be represented as two blocks of numbers from 01 to 26 going from a to z. We can also break down the message to be two letters in each block, so the message reads ma th. In this way 'Math' can be represented

as 13012008.
Now in order to encode this message we can encipher the message block by block and convert it into ciphertext. In order to do this, as seen above we have to raise the message to the power of e (mod n). The first block now becomes 1301^17 (mod 3233) = 2230 and the second block similarly becomes 2008^17 (mod 3233) = 2038. So combining the blocks we get the ciphertext to be 22302038.
Now we can decipher the message by using the private decryption key that receiver has. To do this, we raise the the ciphertext to the power of d (mod n). Breaking the calculation down by block we get 2230^2753 (mod 3323) = 1301 and 2038^2753 (mod 3323) = 2008. As can be seen we have the original message that was to be sent decrypted.

Partial Key Exposure Attack

Supporting Theorems

The proof of the partial key exposure attack, depends on the following Coppersmith's Theorem.

Coppersmith's Theorem. Let $f(x, y)$ be a polynomial in two variables over the integers, Z, of maximum degree δ in each variable separately, and assume the coefficients of f are relatively prime as a set. Let X, Y be bounds on the desired solutions x_0, y_0. Define $f^*(x, y) := (Xx, Yy)$ and let D be the absolute value of the largest coefficient of f^*. If $XY < D^{\frac{2}{3\delta}}$, then in polynomial time in $(logD, 2^\delta)$, we can find all integer pairs (x_0, y_0) with $p(x_0, y_0) = 0, |x_0| < X, |y_0| < Y$.

We do not prove the above theorem by Coppersmith, however, we present a Corollary that is helpful in the proof of the main theorem.

Corollary. Let $N = pq$ be an n-bit RSA modulus. Let $r \geq 2^{\frac{n}{4}}$ be given and suppose $p_0 :\equiv p \pmod{r}$ is known. Then it is possible to factor N in time polynomial in n.

Proof. We know that p_0 is $p \pmod{r}$, then we have $q_0 :\equiv q \equiv \frac{N}{p_0 \pmod{r}}$ By the definition of congruence we have $p = p_0 + xr$ and $q = q_0 + yr$ for some $x, y \in \mathbb{Z}$ Since, $N = pq$ we can can create the following polynomial,

$$f(x, y) = (p_0 + xr)(q_0 + yr) - N$$

To apply Coppersmith's theorem to this bivariate polynomial, the solution, (x_0, y_0) has to be bounded. These bounds are given as $0 < x_0 < X$ and

$0 < y_0 < Y$ where $X = Y = \frac{2^{\frac{n}{2}+1}}{r}$. As N is an n bit number, by definition, $N < 2^n$. If $x_0 = X$, $y_0 = Y$ and let $r = 2^{\frac{n}{4}}$, then

$$(rx)(ry) = (r \cdot \frac{2^{\frac{n}{2}+1}}{r})^2$$

$$(rx)(ry) = 2^{n+2}$$

As $N < 2^{n+2}$, x_0 and y_0 must be strictly less than X and Y, respectively and so the upper bounds hold. Now we have a bivariate polynomial where the solutions are bounded and so we can apply Coppersmith's Theorem. $\square$

Main Theorem

Theorem. Let $N = pq$ be an n-bit RSA modulus. Let $1 \leq e$, $d \leq \phi(N)$ satisfy $ed \equiv 1 \pmod{\phi(N)}$. There is an algorithm that given the $\frac{n}{4}$ least significant bits of d computes all of d in polynomial time in n and e.

Proof. Let $\frac{n}{4}$ least significant bits of d be d_0, then we have $d \equiv d_0 \pmod{2^{\frac{n}{4}}}$. By the definition of the encryption exponent, e

$$ed \equiv 1 \pmod{\phi(N)}$$

By the definition of congruence and divides,

$$ed - 1 = k\phi(N)$$

for some $k \in \mathbb{R}$. Since, $d \leq \phi(N)$, $0 \leq k < e$. Also we know $N = pq$ where p and q are primes, so we have that $\phi(N) = (p-1)(q-1)$

$$ed - 1 = k(p-1)(q-1)$$

$$ed = 1 + k(N - p - q + 1)$$

Let $s = p + q$,

$$ed = 1 + k(N - s + 1)$$

Reducing this equation we have,

$$ed_0 \equiv 1 + k(N - s + 1) \pmod{2^{\frac{n}{4}}}$$

Since $k < e$, the number of candidate values of k is finite and plugging them in the above equation and solve for $s \pmod{2^{\frac{n}{4}}}$. For each of these values we solve

$$p^2 - sp + N \equiv 0 \pmod{2^{\frac{n}{4}}}$$

to obtain a value for $p \pmod{2^{\frac{n}{4}}}$. The above equation is true as

$$p^2 - sp + N \equiv 0 \pmod{2^{\frac{n}{4}}}$$

$$p^2 - (p+q)p + pq \equiv 0 \pmod{2^{\frac{n}{4}}}$$

$$p^2 - p^2 - pq + pq \equiv 0 \pmod{2^{\frac{n}{4}}}$$

$$0 \equiv 0 \pmod{2^{\frac{n}{4}}}$$

Using this value and the above mentioned corollary we can recover all of d and then factor out N efficiently. $\square$

Partial Key Attack Example:

The example can be set up as follows: we are cracking a message for RSA modulus, where $n = 1633$ and $e = 23$. Now we also assume that by some methods like timing attacks we are able to attain the last three least significant bits of d which are 011 in this case, or 3 in decimals. Now looking at this equation we derived in the proof above:

$$ed_0 \equiv 1 + k(N - p - q + 1) \pmod{2^{\frac{n}{4}}}$$

in this problem we have

$$23 \cdot 3 \equiv 1 + k \cdot (1633 - s + 1) \pmod{2^3}$$

Simplifying we get:

$$69 \equiv 1 + k(1634 - s) \pmod 8$$

$$5 \equiv 1 + k(1634 - s) \pmod 8$$

Now we can begin to test possible values for k, which will satisfy the above congruence, and can then be solved for s which can be used in the p quadratic expression given above. k in this case can take on values from 1 to 23, Starting with $k = 1$, we are actually able to hit the solution straightaway with the first value of k. So now if $k = 1$, we have:

$$4 = 1(1634 - s) \pmod 8$$

and therefore,

$$s \equiv 6 \pmod 8$$

now using this in equation $p^2 - sp + N \equiv 0 \pmod{2^{\frac{n}{4}}}$ we have:

$$p^2 - 6p + 1633 \equiv 0 \pmod 8$$

Simplifying,

$$p^2 - 6p \equiv 7 \pmod 8$$

$$p(p - 6) \equiv 7 \pmod 8$$

Solving for p, we get $p \equiv 7 \pmod 8$. Now with this value of $p_0 \equiv p \pmod 8$, a possible value for $q_0 \equiv q \pmod 8$ can also be determined. In order to determine q_0, we have to realize $p_0 \cdot q_0 \equiv N \pmod 8$, thus:

$$7 \cdot q_0 \equiv 1633 \pmod 8$$

$$7 \cdot q_0 \equiv 1 \pmod 8$$

Using a multiple of 7 such as 49 which is congruent to 1 (mod 8) we can get $7 \cdot q_0 \equiv 49 \pmod 8$, dividing both sides by 7 we get $q_0 \equiv 7 \pmod 8$. Thus we have arrived at values for both p_0 and q_0. Since the algorithm has worked so far, we can now be assured that k picked is correct. With the values we obtained for p_0 and q_0 we can now create polynomial obtained from Coppersmith that will allow us to get the factorization for n from its roots:

$$f(x, y) = (rx + p_0) \cdot (ry + q_0) - N$$

which now becomes

$$f(x, y) = (8x + 7) \cdot (8y + 7) - 1633$$

The roots of this polynomial will provide the correct factorization of n. In this case it is easy to see $x = 2$ and $y = 8$, otherwise this could be usually done in polynomial time. This could be done through polynomial time, by testing values for x and y with bounds being set on these values through $X = \frac{2^{\frac{n}{2}+1}}{r}$ and $Y = \frac{2^{\frac{n}{2}+1}}{r}$. $X = Y = 2^3 + 1 = 9$ are the upper bounds. Thus the values for x_0 and y_0 can be tested from 0 to 9. However, in this case we have found the factorization of n, with $p = 8 * 2 + 7 = 23$ while $q = 8 * 8 + 7 = 71$. Now $\phi(n)$ can also be computed as $(p - 1) \cdot (q - 1) = 22 * 70 = 1540$.
Finally we can recover all of d by solving the equation below:

$$ed - k \cdot \phi(n) = 1$$

as $k = 1$ we have

$$23 \cdot d - 1540 = 1$$

and so $d = 67$. In this way, given n, e which are publicly available, and obtaining the last 3 significant digits of d, any message sent using the given key can be decoded.

Future of the RSA Algorithm

As computing power increases, allowing factoring algorithms to become faster the attacks on RSA will become stronger. For example, five years ago a 512-bit value of n was considered safe and today, it can be broken in just a few days. According to the MIT Technology Review, quantum technology will be able to catch up with the encryption methods used today sooner than expected. A study has shown that a quantum computer could break 2048-but RSA in 8 hours. There are several ways of improving RSA's security, the most fundamental being to choose longer keys which would make factorization difficult and help against attacks. RSA and encryption systems in general must strengthen to prevent attacks due to increased computing speed and capacity, and mathematical breakthrough in factoring large numbers.

Conclusion

In this paper, we explore the RSA algorithm developed in 1977 by Rivest, Shamir and Adleman for public-key cryptography. It is one of the most widely used cryptography systems as it is used for encrypting emails, is the heart of credit card transactions and many other tasks involving internet security. After exploring RSA's mechanism, this paper focused on the Partial Key Exposure Attack. Including this attack, there have been many attempts to break RSA but none of these attacks are devastating. So as of now with proper implementation, RSA can be trusted with the security of the digital world. However, with increasing computing power, public-key cryptography systems must be improved accordingly to maintain security.

References

Boneh, Dan, et al. *An Attack on RSA Given a Small Fraction of the Private Key Bits.* An Attack on RSA Given a Small Fraction of the Private Key Bits.

Everstine , Eric W. *Partial Key Exposure Attack On Low-Exponent RSA.* Partial Key Exposure Attack On Low-Exponent RSA.

Milanov, Evgeny. *The RSA Algorithm.* The RSA Algorithm.

Kaliski , Burt. *The Mathematics of the RSA Public-Key Cryptosystem.* The Mathematics of the RSA Public-Key Cryptosystem.

Boneh, Dan. *Twenty Years of Attacks on the RSA Cryptosystem.* Twenty Years of Attacks on the RSA Cryptosystem.

Kelly, Maria D. *The RSA Algorithm: A Mathematical History of the Ubiquitous Cryptological Algorithm.* The RSA Algorithm: A Mathematical History of the Ubiquitous Cryptological Algorithm.