Report on Attacks on RSA

By:
Nidhish Sawant, Siddharth Shah and Ankit Jaiswal

Indian Institute of Technology Goa

Supervised By
Dr. Arpita Korwar

20 August, 2020

Contents

1 Preliminaries

1.1 Groups Z_n and Z_n^*[Gal15]

The set $Z_n = \{0, 1, ..., n-1\}$ for $n \geq 1$ is a group under addition *modulo* n. For any $j > 0$ in Z_n, the inverse of j is $n - j$. This group is usually referred to as the group of integers *modulo* n.

Let G be a group. If n is a positive integer, then a^n is the element obtained by applying the group operation on a n times. If n is a negative integer then, $a^n = (a^{-n})^{-1}$. If n is zero then a^n is e, the group identity. The group G is called cyclic if there is an element a in G such that $G = \{a^n | n \in Z\}$. Such an element a is called a generator of G. Consider the group Z_n, element $1 \in Z_n$ is the generator of the group Z_n hence, Z_n is a cyclic group.

The set $Z_n^* = \{a \mid 1 \leq a < n,\ \gcd(a, n) = 1\}$, where $\gcd(a, n)$ is the Greatest Common Divisor of a and b for $n \geq 1$, is a group under multiplication *modulo* n. For any $j \geq 1$ in Z_n^*, the inverse of j is the coefficient of j in the equation $j \cdot s + n \cdot t = 1$ because this implies $j \cdot s \pmod n = 1$.

1.2 Euler Totient Function[CV13]

Euler Totient function of an integer n with prime factorisation as $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot p_3^{\alpha_3} ... \cdot p_r^{\alpha_r}$ is defined as

$$\phi(n) = (p_1 - 1)p_1^{\alpha_1 - 1} \cdot (p_2 - 1)p_2^{\alpha_2 - 1} \cdot (p_3 - 1)p_3^{\alpha_3 - 1} ... \cdot (p_r - 1)p_r^{\alpha_r - 1}$$

$\phi(n)$ is precisely the order of the group of Z_n^* because essentially it gives the number of elements $a < n$ such that $\gcd(a, n) = 1$.

1.2.1 Euler's Theorem[Gal15]

Euler's Theorem states that if a and n are relatively prime then:

$$a^{\phi(n)} \equiv 1 \pmod n$$

1.3 Carmichael Totient Function

The Carmichael Totient function associates every positive integer n to a positive integer $\lambda(n)$, defined as the smallest positive integer m such that

$$a^m \equiv 1 \pmod{n}$$

for every integer a between 1 and n that is co-prime to n.

1.3.1 Properties of $\lambda(n)$[EPS91]

- $$\lambda(n) = \begin{cases} \phi(n) & \text{if } n = p^r = \text{odd prime power or 2,4 }, \\ \frac{\phi(n)}{2} & \text{if } n = p^r = \text{even prime power except 2,4 }, \\ lcm(\lambda(p_1^{r_1}), \lambda(p_2^{r_2}), ...\lambda(p_k^{r_k})) & \text{if } n = p_1^{r_1} p_2^{r_2} ....p_k^{r_k} \text{ .} \end{cases} \tag{1}$$

- $\lambda(n)$ divides $\phi(n)$

- If n has maximum prime exponent r_{max} under prime factorization, then for all a (including those not coprime to n) and all $r \geq r_{max}$,

  $$a^r \equiv a^{r+\lambda(n)} \mod n.$$

  In particular, for square-free n ($r_{max} = 1$), for all a we have,

  $$a \equiv a^{1+\lambda(n)} \mod n. \tag{2}$$

2 RSA Cryptography

RSA [RSA77], which is the main focus of this report, is a very widely used computationally (but not theoretically) secure public-key cryptographic system. As it is a public-key cryptosystem, the pair of keys (the public key e and the private key d) involved in the encryption and decryption processes are different.

The first step of the RSA algorithm is key generation and the generator of the keys is the one who owns them. For this step, first a large positive integer n is computed such that it is the product of two large primes p and q. Then $\lambda(n)$ is calculated. After that, the public key e is chosen to be any non-identity integer in the set $Z^*_{\lambda(n)}$. Then, the private key d is set as the inverse of e in the group $(Z^*_{\lambda(n)}, *)$. This ensures that any integer m in Z^*_n, when raised to the power $e \cdot d$, gives back m. And this is due to the the fact that $e \cdot d \equiv 1 \mod \lambda(n)$ and also the definition of $\lambda(n)$. This entire step of key generation is performed by the owner of the keys.

The owner of the keys then distributes the public key e and the modulus n among all those from whom he/she wants to receive messages securely.

After the key distribution step, anyone with the public key can encrypt his/her message m by doing the following simple computation:

$$c = m^e \mod n$$

where c is the cipher generated upon encryption. c is then sent to the owner of the keys.

Then, the owner of the keys can easily decrypt the cipher c by the following computation to get back m:

$$m = c^d \mod n$$

A key observation here is that as $n = p^1 * q^1$, $r_{max} = 1$. Due to this, the message m, which can be encrypted and later decrypted using this algorithm can be any integer in the set Z_n, and not just any integer in the set Z^*_n (By Equation 2).

Another important thing to note is that if we replace $\lambda(n)$ by $\phi(n)$ everywhere in the algorithm, then also it'll work just fine. All the steps would be the same, and all the properties will still be there. In fact, when the RSA algorithm was first published by its developers, they had used $\phi(n)$ instead of $\lambda(n)$.

3 Attacks on RSA Cryptosystem

3.1 Iterative Encryption

The Iterative Encryption algorithm is based on the fact that $Z^*_{\lambda(n)}$ is finite group and $e^{-1} = d$ is an element of $< e >$, the cyclic group generated by e

in $Z^*_{\lambda(n)}$. As referred by the name of the Algorithm, we repeatedly encrypt c until we get back c again.

3.1.1 Algorithm

The function takes the encrypted message, c and public key, e & n as inputs and returns the message, m. Lets see what happens if we repeatedly encrypt c. First take $C_1 = c = m^{e^1} \pmod n$ and encrypt it again to get $C_2 = m^{e^2} \pmod n$. In general we define $C_i = m^{e^i} \pmod n$. We keep track of C_j and C_{j-1} and check if C_j is same as $C_1 = c$. If it turns out that $C_j = C_1$ then in the following sub-section we will prove that, $j - 1 \mid q$ where q is order of e, and hence $C_{j-1} = m$, else we continue encrypting C_j.

3.1.2 Analysis of Algorithm

To know how the algorithm works we need to analyse the algorithm. For that, first observe that e and $d \in Z^*_{\lambda(n)}$ and $d = e^{-1} \pmod n$. Thus if $|e| = r$ then :-

$$\begin{aligned} e^r &= 1 \pmod \lambda)(n) \\ \implies ee^{r-1} &= 1 \pmod{\lambda(n)} = ed \pmod{\lambda(n)} \\ \implies d &= e^{r-1} \pmod{\lambda(n)} \\ \implies c^{e^{r-1}} &= C_r = m \pmod n. \end{aligned}$$

We also know that for all a, $a^k = a \pmod n$ if $k = 1 \pmod{\lambda(n)}$ and hence $c^{e^r} = c$.

Lemma 1. *Let j be the smallest non-negative integer such that $C_j = c$. Then, $C_{j-1} = m$.*

Proof. To do so we first observe that the only way this is not the case is when $j < r+1$, as if $j \geq r+1$, then the first case will be when $j = r+1$ and it is true for $j = r+1$ as shown above. Now assuming $j < r+1$ and $C_j = c$ we can see that $C_{j+1} = C_j^e = c^e \pmod n = C_2$ and similarly $C_{j+i} = C_{1+i}$ for all i. Hence we have that the elements in the sequence $\{C_i \mid i = 1, 2, ...j\}$

keep on repeating sequentially and are all then elements of form C_i. We also know that $C_{r+1} = c$ which implies C_j must be same as C_{r+1} and hence $C_{j-1} = C_r = m$. □

Through this analysis we can also observe that this algorithm is of order poly $log(n) * \lambda(\lambda(n))$, as r is of order $\lambda(\lambda(n))$. This also brings some questions to the table, what are the values of n for which $\lambda(\lambda(n))$ is small? And is it computationally easy to calculate n, such that $\lambda(\lambda(n))$ is big enough? These are the questions we could have worked further upon.

3.2 Common Modulus Attack

This type of attack needs some special requirements to be met by the cryptosystem. We will present these requirements in the form of a special scenario.

Suppose, the overall cryptography infrastructure of an organization for the secure communication of its employees among themselves and with the outside world is managed by the Information Security Department (ISD) of the organization. And to make the public key infrastructure maintenance simple for the organization, it is decided that a single common modulus n will be used throughout the organization, while the individual employees will be handed their respective pair of keys (a public encryption exponent e and the corresponding private decryption exponent d). The ISD also decides to choose the keys e and d for each of the organization's employees from the set $Z^*_{\phi(n)}$, instead of $Z^*_{\lambda(n)}$, which, as we have mentioned earlier, is an alternate form of RSA. Moreover, the ISD decides that the encryption exponents will be chosen so that the gcd of any pair is 1; for example, to speed up selecting values of encryption exponent e, the ISD selects from a database of prime numbers less than $\phi(n)$.

An eavesdropper can, under certain circumstances, decrypt messages sent in this system without needing access to any private keys.

Suppose that the CEO dispatches the same message M to two different employees whose encryption exponents, respectively, are e1 and e2. The encrypted messages are:

$$E = M^{e_1} \mod n$$

$$\text{and } F = M^{e_2} \mod n$$

An eavesdropper has access to the public keys of both the employees, (i.e. n, e_1 and e_2) and the encrypted messages E and F. With these things in hand, he/she can, with the help of the extended Euclidean algorithm, easily find x and y, such that $e_1 * x + e_2 * y = 1$.

Then all that needs to be done is the following computation:

$$E^x * F^y \mod n$$

$$= M^{e_1 * x} * M^{e_2 * y} \mod n$$

$$= M^{e_1 * x + e_2 * y} \mod n$$

$$= M \mod n.$$

Hence, the eavesdropper successfully gets the original message M. It is important to note here that if the gcd of e_1 and e_2 were not 1, then the eavesdropper would not have got hold of the original message.

Hence, one defense against this attack is be to ensure that the set of all encryption exponents has a number larger than 1 as a common factor. Though this defends sufficiently against an outside eavesdropper, we shall see next that it is no defense against an insider attempting to steal proprietary information from the organization.

3.3 Attack when the public encryption exponent is "small"

For this attack, we will continue using the running scenario, with a small change, that the gcd of any two encryption exponents need not be 1 (i.e. it can be any integer ≥ 1). But no two encryption exponents should be equal, due to obvious privacy reasons.

Now, as the name of the attack suggests, we also have the condition that for some employee, the encryption exponent is "small". With all these conditions, that employee has the following method for finding out $\phi(n)$ for the common modulus n.

Notice that $\phi(n)$ is not a priori known to the employees, but it is known that, if e and d are respectively the encryption and decryption keys of the malicious employee, for some natural number $k < \min(e, d)$ we have $e * d = 1 + k * \phi(n)$. This is because $\phi(n)$ is greater than both e and d, and hence, k must be less than both of e and d for the equation to hold true.

Thus, from the above equation, we have $\phi(n) = (e * d - 1)/k$. So, the malicious employee needs to check less than $\min(e, d)$ number of values of k, to search for $\phi(n)$. And as e is quite "small", this method becomes very efficient.

Once the employee finds out $\phi(n)$, with the help of any other employee's public exponent, he/she can easily find out their private exponent using the extended Euclidean algorithm, thus breaking RSA.

Conjecture: In the process of finding the possible values of $\phi(n)$, one (here, the malicious employee) finds that there are multiple such possible values. So, this algorithm does not always give the exact value of $\phi(n)$, but rather narrows it down to a few values. Another observation is that, given the public key (e_1) of some other employee, when the malicious employee tries to find the corresponding private key (d_1), he/she finds that for each of the possible values of $\phi(n)$, there exists a possible value of d_1 (which is not necessarily distinct from the other possible values, found by using the other possible values of $\phi(n)$).

3.4 Wiener's Attack

The Wiener's Attack is based on choosing small private decryption key. The reason why one would choose small decryption key is to speed up the process of decryption. But, small decryption key always leads to large encryption key, e, and thus slowing down the encryption process.

Let us look at the Euclid's Algorithm to find the Greatest Common Divisor of two numbers a and b such that $a < b$. We get a list of equations using long division.

$$\begin{aligned}
b &= q_1 * a + r_1 \quad 0 \leq r_1 < a \\
a &= q_2 * r_1 + r_2 \quad 0 \leq r_2 < r_1 \\
r_1 &= q_3 * r_2 + r_3 \quad 0 \leq r_3 < r_2 \\
&\vdots \\
r_n &= q_{n+2} * r_{n+1} + r_{n+2} \text{ and } r_{n+2} = 0,\ r_{n+1} > 0
\end{aligned}$$

The last non-zero remainder r_{n+1} is the $\gcd(a, b)$.
Consider the rational number b/a with $\gcd(a, b) = 1$. Then $r_{n+1} = 1$. Using

the same set of equations we get:

$$\begin{aligned} b/a &= q_1 + r_1/a \\ &= q_1 + 1/(a/r_1) \\ &= q_1 + 1/(q_2 + r_2/r_1) \\ &= q_1 + 1/(q_2 + 1/(q_3 + r_3/r_2) \\ &\vdots \\ &= q_1 * r_1 + 1/(q_2 + 1/(q_3 + 1/(q_3 + 1/(q_4 + (\ldots + 1/(q_{n+2})\ldots))))) \end{aligned}$$

The last expression is known as the continued fraction expansion of b/a, and for typographical convenience it is written as $b/a = [q_1, q_2, q_3, q_4, \ldots, q_{n+2}]$. The number $C_j = [q_1, q_2, q_3, q_4, \ldots, q_{j+1}]$ is known as $\boldsymbol{j^{th}}$ **convergent of** $\boldsymbol{b/a}$. These convergent can be calculated easily usind Euclid's Algorithm, and they satisfy a very important property. The following is the one which we will use.

Theorem 2. *[Kop14] Assume that* $\gcd(a, b) = 1$*. If* r, s *are any natural numbers such that* $gcd(r, s) = 1$*, and* $|a/b - r/s| < 1/(2s^2)$*, then* r/s *is one of the convergents of* a/b*.*

Theorem 3. *Let* n *be an RSA modulus, say* $N = p * q$ *where* p *and* q *are primes, and let* e *be the public encryption key and* d *be the private decryption key. If* $d < \sqrt[4]{N}/3$ *(i.e* d *is small),* $q < p < 2q$ *and* $ed = 1 + k\phi(N)$ *with* $k < min\{e, d\}$ *then* $\left|\frac{e}{N} - \frac{k}{q}\right| < \frac{1}{2d^2}$*.*

Proof.

$$\begin{aligned} ed &= 1 + k\phi(N) \\ \text{Now, } \phi(N) = (p-1)(q-1) &= N - p - q + 1. \\ \text{Since } p < \sqrt{N}, q &< \sqrt{N}, \\ \therefore |p + q - 1| &< 3\sqrt{N}, \\ \therefore |N - \phi(N)| &< 3\sqrt{N}. \end{aligned}$$

Consider the following equation:

$$\begin{aligned}
|e/N - k/d| &= \left|\frac{ed - kN}{Nd}\right| \\
&= \left|\frac{ed - k\phi(N) - kN + k\phi(N)}{Nd}\right| \\
&= \left|\frac{1 - k(N - \phi(N))}{Nd}\right| \leq \left|\frac{3k\sqrt{N}}{\sqrt{N}\sqrt{N}d}\right| = \frac{3k}{\sqrt{N}d}
\end{aligned}$$

$$\begin{aligned}
\text{Now } k\phi(n) = ed - 1 < ed & \\
& \therefore k\phi(n) < ed \; and \; e < \phi(n) \\
& \therefore k\phi(n) < d\phi(n) \\
& k < d
\end{aligned}$$

$$\begin{aligned}
\text{Since } k < d \ \&\ d < \sqrt[4]{N}/3 & \\
\text{Hence, } \left|\frac{e}{N} - \frac{k}{d}\right| \leq \frac{3k}{\sqrt{N}d} < \frac{3d}{\sqrt{N}d} < \frac{1}{d\sqrt[4]{N}} & \\
\text{Now } d < \sqrt[4]{N}/3,\ 2d < 3d, & \\
& \therefore \left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{d\sqrt[4]{N}} < \frac{1}{d * 2d} = \frac{1}{2d^2} \\
& \therefore \left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{2d^2} \\
\text{Hence Proved} & \qquad (3)
\end{aligned}$$

□

Thus by **Theorem 2** k/d is a unique convergent of e/N. There can be an efficient algorithm to find d using this method (i.e examining the convergents). Once the d is found k can be easily found, because it is nothing but the numerator of that particular convergent. Hence, $\phi(N)$ can be found and N can be factored.

3.4.1 Algorithm

Since $N = p * q$, and p and q are primes therefore $\phi(N)$ must be an even number.

- Hence, d cannot be even so, first test is to check whether the guessed d is even or not.
- Second, $\phi(N) = \frac{ed-1}{k}$ must be a whole number. Hence, check whether the guessed $\phi(N)$ is whole or not.
- The equation

  $$\begin{aligned}(x-p)(x-q) &= x^2 - x(p+q) + pq \\ &= x^2 - x(N - \phi(N) + 1) + N\end{aligned}$$

  Has roots p and q. Using the quadratic formula we can find p and q. Using these p and q check whether $p * q = N$ or not.

If all these three test are passed then the value of guessed d is the original private encryption key.

Some conjectures:

- There can be certain other tests such as to check whether the roots $p \; and \; q$ are real or not.
- The answer to why there are not more tests after we find $N = p * q$ to be true is not given in this report.

4 Conclusion

We conclude that although a perfectly robust (i.e. computationally secure) form of the RSA cryptosystem can exist, not having proper knowledge on the inner working or the mathematics behind the RSA algorithm can lead to one of the many special cases, in which there is scope for an attacker (which could be an outsider or someone from within the organization) to take advantage of a computationally efficient method to break the RSA implementation.

References

[CV13] Shashank Chorge and Juan Vargas. Rose Hullman Undergraduate Mathematics Journal. https://scholar.rose-hulman.edu/cgi/viewcontent.cgi?article=1081context=rhumj, 2013. [Online; accessed 11-August-2020].

[EPS91] Paul Erdös, Carl Pomerance, and Eric Schmutz. Carmichael's lambda function. *Acta Arithmetica*, 1991.

[Gal15] Joseph A. Gallian. *The Complexity of the Annihilating Polynomial.* 2015.

[Kop14] Swastik Kopparty. Continued fractions and rational approximation. https://sites.math.rutgers.edu/ sk1233/courses/ANT-F14/lec2.pdf, 2014. [Online; accessed 11-August-2020].

[RSA77] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. https://people.csail.mit.edu/rivest/Rsapaper.pdf, 1977. [Online; accessed 11-August-2020].