Breaking RSA Encryption

Grant Haataja

MATH 488 - Mathematics Capstone - Fall 2019

December 10, 2019

Abstract

Developed in 1977, RSA **encryption** is a public-key cryptographic scheme that is heavily relied on for many of our modern security needs. The RSA abbreviation comes from the last names of Ronald Rivest, Adi Shamir, and Leonard Adleman, the three MIT researchers who came up with the necessary one-way function to make RSA encryption secure. A public-key scheme means the encryption key is not a secret, but individual users all have their own secret decryption key.

Although it is an extremely secure method of encrypting data, nothing is perfect and there are some flaws in the method. These flaws need to be analyzed in great detail in order to be understood and kept from being exploited. It is necessary to understand how RSA encryption can be broken, and how viable the methods of breaking it would be in both the present real world and the future.

Project Description

As the world of computers came into existence, dozens of new challenges presented themselves to early computer scientists and mathematicians. Among the most important of these challenges was the need to securely transmit data. Although RSA encryption began development in the early 1970s, was finalized in 1977, and patented in 1983 by MIT, it was not until the 1990s and the early days of the internet before RSA encryption started to be used.

Public-key encryption was an especially useful solution to the problem of internet security, since it allows for encryption to occur without any prior exchanging of encryption/decryption keys. In fact, the encryption key can be known to all and it does not affect the security of RSA encryption.

RSA encryption is used extensively to secure various facets of the internet, often used in tandem with other symmetric-key encryption algorithms for greater efficiency. This is due to the fact that RSA decryption is far more computationally heavy than other symmetric-key encryption, so often the symmetric algorithm key will be encrypted with RSA encryption, thus rendering the message unreadable to anyone who does not possess the required RSA decryption key [1]. RSA encryption is also used for authenticity verification of digital signatures, securing connections between VPN clients and servers, email security, and many other websites and online chats.

Running in the background of all these internet processes, RSA encryption uses extremely large prime numbers and several complex mathematical functions including Carmichael's totient function, generating primes, trapdoor functions, and computing the public and private keys for encryption/decryption.

Trapdoor functions are named for the characteristic that they are trivial to solve in one direction, but realistically impossible to solve in the other direction. Take the following example where x and y are prime numbers.

$$xy = 4695923$$

Is it possible to compute x and y? By hand it would be arduous at best. With the assistance of a computer program it could be done fairly quickly. But RSA encryption uses numbers that are often over 600 digits long. Even with a computer program, guessing the prime number divisors would take too long to be worth it.

Now look at a second example, given two prime numbers 1811 and 2593.

$$1811 * 2593 = z$$

Now is it possible to compute z? Of course! A basic calculator could do this instantly, and a grade-school math student could calculate this with pencil and paper in a matter of a few minutes. In fact, $1811 * 2593 = 4695923$, which illustrates the one-way nature of a trapdoor function. This one-way nature is necessary for RSA encryption.

The first critical step in RSA encryption is generating the prime numbers that will be used in the composition of the public and private keys. These numbers are selected with a **primality** test, which is an efficient algorithm for selecting prime numbers. [2]. In practice, the numbers must be very large and relatively far apart, but in order to understand the process, much smaller numbers will be used in examples.

Let prime numbers p and q be given by a primality test. For an example, take 1811 and 2593, as used above. The next step is to compute the modulus n.

$$n = 1811 * 2593 = 4695923$$

The Carmichael's totient function is then used.

$$\lambda(n) = \text{lcm}(p-1, q-1)$$

$\lambda(n)$ is called Carmichael's totient, and we can calculate that by finding the least common multiple of $p-1$ and $q-1$.

$$\lambda(4695923) = \text{lcm}(1811-1, 2593-1)$$

$$\lambda(4695923) = \text{lcm}(1810, 2592) = 2345760$$

After computing the Carmichael's totient of the modulus, the public key must be generated. The public key for RSA encryption is composed of a prime number e and the modulus n from above. Because of the one-way nature of RSA, it is not important for e to be kept secret, but it also cannot be too large or it will cause computations to be quite slow. Generally, 65537 is used, but a much smaller number will be used here for illustrative purposes. Take $e = 7$. Given a plaintext message m, a ciphertext c can be computed by the following formula.

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

For the sake of simplicity, assume the plaintext message has been converted to a simple number, like $m = 111$. Then we can convert it to ciphertext using the above forumla.

$$c = 111^7 \bmod 4695923 = 3795638$$

It is not difficult to see that the number $c = 3795638$ is meaningless as a message without a decryption key. So, the next important step in the RSA process is generating the private key. Private keys are composed of a number d and the modulus n. To compute d, the following formula is used.

$$d = 1/e \bmod \lambda(n)$$

Of course, $1/e$ is not as simple as dividing 1 by e in modulo n, since only integers can be used in RSA encryption. The notation refers rather to the **modular inverse** of $e \bmod \lambda(n)$. Continuing with the above example, d can be computed with the formula above.

$$d = 1/7 \bmod 2345760 = 1675543$$

Now that the private key d has been generated, ciphertext messages can be decrypted with the following formula.

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

$$m = 3795638^{1675543} \bmod 4695923$$

Now it is easy to appreciate how RSA encryption/decryption is computationally heavy, especially when considering that typical values for p and q are over 600 digits long. This is why RSA is usually used in combination with a less extensive symmetric key scheme, and only the symmetric key is RSA encrypted, rather than the entire file or message. Using computer software, the above equation can be solved to find $m = 111$, as expected.

One particularly useful application of RSA encryption is for so-called signing messages, or verification of digital signatures. This is different from signing one's name in cursive and comparing it to a previously signed document from the same person, but it is a surprisingly similar concept. As an example, imagine a university professor uploads a syllabus to their personal website, then giving the link to their website to all their students. For those students who want to keep a copy of the syllabus on their computers for easy reference, they will go to the website and download a copy of the syllabus.

Now, although most people do not often consider this, depending on the security of the professor's website, a hacker might be able to embed malicious code within the file or change its nature. If the hacker did this correctly, it would not be possible to notice anything wrong with the file before downloading it.

That is where digital signing comes in. If the professor's website used digital signing, then when they first uploaded the syllabus to their website, in the background the website would have ran the entire content of the file through a hash function in order to compress the file, and then run RSA encryption on the hash digest.

Then, when a student later downloaded the syllabus from the website, before the file was approved for download the contents would be ran through the hash function again, and the student's computer would decrypt the RSA of the hash digest and compare the two values. If they matched, the digital signature is verified and the download can occur.

This is just one example out of a plethora of applications RSA encryption is used for in online security. It is imperative for these operations to remain secure from potential attackers. Thus, the potential flaws and vulnerabilities with RSA encryption will be analyzed and discussed at length. Everything will be studied from the point of failure in an attempt to get as greatly detailed report of the overall effectiveness of RSA encryption.

Brute Force Method

One can attempt to factor the modulus n to discover the private key d, thereby breaking that instance of the RSA algorithm, but if RSA is being implemented properly, keys will be generated frequently and just knowing one key does not help with a message that used a different key.

Furthermore, the odds of factoring the modulus are infinitesimal. Many articles and threads on various websites state that the amount of possible moduli for 1024-bit RSA is significantly larger than the number of atoms in the observable universe. Of course, this statement is highly debatable, but all sources agree that an algorithm which attempts to factor the modulus by using every possible pair of moduli factors is impossible in practice.

Quantum Method

One very interesting theoretical method for breaking RSA is by the Quantum Method. To understand this, one must understand quantum computing on a basic level. In theory, **quantum computing** works by utilizing qubits, which can be in states of 0 or 1 like regular bits, but also can exist in a state of **superposition** between 0 and 1. Because of the ability to exist in multiple states at once, quantum computing can be used to compute things that traditional computing does not have the power to do. But even though quantum computers do exist, they have not fully surpassed traditional computing methods in practice, due to the difficulty of controlling qubits. The most qubits used in a single experiment so far has been twelve. [6]

The challenge with qubits is their tendency to **decohere** after any disturbance, causing them to no longer follow the principles of quantum mechanics. Finding ways around this is one of the foremost challenges in quantum computing research today.

Now, how does one use quantum computing to crack RSA encryption? The answer to this provoking question lies in Shor's algorithm. This method has not been used in practice to break RSA, because it requires a powerful quantum computer, but the simplified method is as follows: "We can crack RSA if we have a fast way of finding the period of a known periodic function $f(x) = m^x \pmod{N}$" [7] The steps are outlined in detail:

1. Find the GCD of N and m, where N, is the number you are trying to factor, and m is a random positive integer less than N. Most likely, $\gcd(N, m) = 1$, and the algorithm continues. If the gcd does not equal one, then you have found a factor of N and the work is complete.

2. Find the period of $m \pmod{N}$, $m^2 \pmod{N}$, and $m^3 \pmod{N}$. This is the only step that requires a quantum computer superior to traditional computers.

3. If the period P is even, continue with the algorithm. If it is odd, go back to step 1 and choose another random integer m.

4. Confirm that $m^{P/2} + 1 \neq 0 \pmod{N}$. If this is true, continue on to step 5. Otherwise, go back to step 1 and choose another random integer m.

5. Compute $\gcd(m^{P/2} - 1, N)$. The result will be a non-trivial prime factor of N, and will give you the key to break anything encrypted using RSA with the key N. [7]

Clearly, the key to this method working is finding the period of exponential functions. This can be achieved by transforming them into hyperbolic sine and cosine functions to find a periodicity. Implementing this period-finding step requires superposition. With superposition, one can turn the function into a quantum transform, and perform the quantum Fourier transform.

By doing this, due to the functionality of quantum superposition allowing qubits to be in multiple states at the same time, the period of the function is approximated with high accuracy.

Right now, this is largely theoretical due to the fact that no quantum computer exists now that can complete this process. When or if such a computer will be developed is a matter of much debate within the quantum computing field of science.

Conclusion

If the trends of technological development are to be trusted, quantum computing will eventually surpass traditional computing, and when that happens RSA will indeed be vulnerable to quantum attacks. While this may seem like cause for alarm, it is simply the natural progression of technology. Even now, as there are researchers working hard on bolstering quantum computers to solve these complex problems like factoring large numbers, there are other researchers developing algorithms that will allow for secure cryptographic methods against quantum attacks. [6] So, if and when the technology emerges, cryptologists will be ready.

Acronyms

GCD Greatest Common Divisor. 6

RSA Public-key encryption technology named after Rivest, Shamir, and Adleman. 1

VPN Virtual Private Network. 2

Glossary

decohere The action of a qubit falling out of a quantum state and becoming a traditional bit. 6

encryption The process of converting information or data into a code, especially to prevent unauthorized access [5]. 1

modular inverse The modular inverse of an integer a is an integer x such that $ax = 1$ with the respective modulus m. 4

primality The property of being a prime number. 3

quantum computing A theoretical area of computing that could transform computer memory from a binary system to one that uses quantum superposition to exist in multiple states of memory at once. 6

superposition The ability of a quantum system to be in multiple states at the same time [6]. 6

References

[1] Lake, J. (2018, December 10). What is RSA encryption and how does it work? Retrieved from https://www.comparitech.com/blog/information-security/rsa-encryption/

[2] Hoffoss, D. (2012, January 23). Nice Discussion of Fermat pseudoprimes, Carmichael numbers, and their distribution, courtesy of Jeffrey Leon's webpage. Retrieved from http://home.sandiego.edu/~dhoffoss/teaching/cryptography/10-Rabin-Miller.pdf

[3] Jamgekar, R. and Joshi, G. (2013, February). File Encryption and Decryption Using Secure RSA. Retrieved from https://pdfs.semanticscholar.org/b64b/1bccd310b7b4913d9a2ab1e77eba01c4aef0.pdf

[4] Robert Edward Lewand. *Cryptological Mathematics*. The Mathematical Association of America, 2000.

[5] Oxford. (2019). Definition of encryption in English. Retrieved from https://www.lexico.com/en/definition/encryption

[6] University of Waterloo, Institute for Quantum Computing. (2019). Quantum computing 101. Retrieved from https://uwaterloo.ca/institute-for-quantum-computing/quantum-computing-101

[7] Anastasia Marchenkova. (August 13, 2015). Break RSA encryption with this one weird trick. Retrieved from https://medium.com/quantum-bits/break-rsa-encryption-with-this-one-weird-trick-d955e3394870

[8] Mark Rethana. (August 28th, 2018). Basic Guide to Quantum Computing and Superposition. Retrieved from https://medium.com/@mark.rethana/a-beginners-guide-to-the-quantum-computing-and-superposition-536e4fc040a2

[9] Hüseyin Bodur, Resul Kara. (June, 2015). Secure SMS Encryption Using RSA Encryption Algorithm on Android Message Application. Retrieved from https://www.researchgate.net/publication/298298027_Secure_SMS_Encryption_Using_RSA_Encryption_Algorithm_on_Android_Message_Application

[10] Short Tech Stories. (June 23rd, 2017). How does RSA work? Retrieved from `https://hackernoon.com/how-does-rsa-work-f44918df914b`