Breaking RSA Encryption - Project Proposal

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 for 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 3 and $\lambda(n)$. 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.

Work Plan

The vast majority of the work to be done on this project is researching all the possible flaws of RSA encryption. Most of the time required will be devoted to such research. Many sources will be referenced to obtain the broadest spectrum of information, including the sources cited below and quite likely many more.

The project will take 10 weeks in total, with the finished final draft ready for submission and the final visual presentation ready to go at the end of the 10th week. Drafts of the paper will be updated as the research occurs. The bulk of the research can be split into more specific

topics as follows:

- Explaining how RSA encryption works
  - Trapdoor functions
  - Carmichael's totient function
  - Generating primes
  - Computing public/private keys
  - Specific examples for used of RSA
- Brute force method
  - Explain algorithms for brute force
  - Analyze run time and impossibility
- Bypass method
  - Not really breaking RSA
  - Break symmetric scheme without using the symmetric key
- Quantum method
  - Briefly explain quantum computing
  - Shor's algorithm
  - Analyze possibility
  - Predict when quantum computing may destroy cryptology as we know it
- Other methods
  - Research any other possibilities

We can schedule the research over a period of five weeks, and devote the three remaining weeks to finalizing the paper. The following is an estimated timeline of the required work for this project over the semester.

TABLE 1 Timeline

Week 01: Sep. 30	Finalize research for explaining RSA encryption
Week 02: Oct. 07	Research brute force method
Week 03: Oct. 14	Off-week for traveling
Week 04: Oct. 21	Research bypass method
Week 05: Oct. 28	Research quantum method
Week 06: Nov. 04	Finalize research
Week 07: Nov. 11	Finish writing bulk of paper
Week 08: Nov. 18	Finalize formatting and start creating visual presentation
Week 09: Nov. 25	Off-week for thanksgiving
Week 10: Dec. 02	Finalize paper and visual presentation

This schedule is just an estimate and may change as research and draft updates progress.

Acronyms

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

VPN Virtual Private Network. 2

Glossary

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

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