The Secret to Sending Secret Messages

†Richard Muniu

†Department of Mathematics, Swarthmore College

Don't Look, Stranger!

Often times, we need to send information that requires privacy against other parties, called *adversaries*. Such information includes messages that we wouldn't want other people to read, private records, legal documents, forms, and passwords. This is common in web browsers, chat applications, email, VPNs, and any other types of communication platforms that require securely sending data to servers or other people. How then, can we achieve this? The solution is simple, I promise: Cryptography.

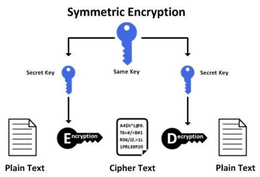
Introduction

Cryptography is the practice and study of techniques used to communicate information between parties such that only the intended recipient can read and process that information. Encryption is the process through which such information is encoded in a way that prevents unauthorized access by parties to whom the communication is not intended. Encryption doesn't prevent unauthorized access to the encrypted content, through interception for instance, but prevents adversaries from understanding what the original content of the information is.

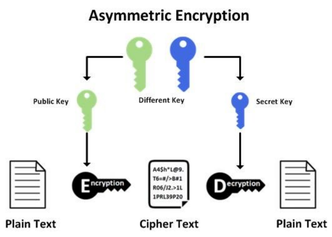
Public Key Cryptography

There are two different types of encryption schemes: symmetric cryptography and public key cryptography (sometimes called asymmetric encryption).

Symmetric cryptography requires both the sender and receiver to have a single, common key that is used to encrypt and decrypt the message, which results in the need for a large number of unique key pairs. You can think of this as putting a message in a box and locking it with a padlock whose key you have, and then sending it to someone who also has an identical key and can thus open the box and retrieve your message.



Public key cryptography, on the other hand, uses a pair of keys; one public one which can be shared with everyone, and one private one that the receiver uses to decrypt messages encoded using the public key. This is considerably better in terms of number of keys and security, which we shall discuss here.



THE RSA METHOD

The basic idea of public key cryptography is pretty intuitive; encryption and decryption are inverses. More rigorously, the cryptographic scheme needs to satisfy the properties below:

(a) Both E and D should be easy to compute, as they are used frequently.

(b) Decrypting the ciphertext gives back the original message M, formally written as,

$$D(C) = D(E(M)) = M.$$

(c) If a message M is first decrypted then encrypted, the final output should be the original message. Formally,

$$E(D(M)) = M.$$

(d) Publicly revealing E does not provide an easy way to compute D. This ensures security, and only the user can decrypt messages encoded using E or compute D efficiently.

RSA achieves this by using modular exponentiation using prime numbers and and *semiprime numbers*, which are the product of two primes. The RSA method uses three positive integers n, e, and d, which are carefully selected to satisfy the procedure requirements above. The encryption key is the pair (n, e), which is made publicly available, while the decryption key is the pair (n, d) which the receiver keeps private. Here, n is called the *modulus*, e is called the *encryption exponent*, and d is called the *decryption exponent*.

To encrypt a message, we first convert it to some numerical form M using an encoding scheme, then encode it into it's ciphertext equivalent C using the formula

$$C = M^e (\text{mod } n).$$

The sender then transmits C to the receiver by some means, after which the receiver can decrypt the ciphertext using the formula,

$$M = C^d (\text{mod } n),$$

which they can then decode to it's plain text equivalent using the same original scheme.

Euler's Totient Function Φ

Φ of a number n, written $\Phi(n)$, is a function that measures the "divisibility" of a number n, that is, the number of positive integers less than n that do not share a common factor (excluding 1, which is a factor of everything) with n. As a simple example, consider $n = 8$. For all the positive integers less than 9 (that is; 1,2,3,4,5,6,7,8), only 1,2,4,5,7, and 8 are coprime to 9. Therefore, $\Phi(9) = 6$.
Calculating Φ of a prime number p is rather easy; since p has no other factors apart from 1 and itself, then every positive integer less than p is relatively prime to p. Thus,

$$\Phi(p) = (p - 1).$$

Euler's totient function is a multiplicative function, meaning that for any two primes p and q, $\Phi(p * q) = \Phi(p) * \Phi(q) = (p - 1)(q - 1)$.

Trapdoor Functions

A *trapdoor function* (also called a *one-way function*) is a function that is easy to compute in one direction but verry difficult to compute in the other without special information, which is called the "trapdoor". Calculating the Euler's totient ($\Phi(n)$) of a semiprime number n is one such function - without knowing the unique prime factorization, it is intractable to compute $\Phi(n)$. RSA encryption and decryption relies heavily on this.

Fermat's Little Theorem

The theorem states that for two integers a and p, if p is prime and p is coprime to a, then $a^{p-1} \equiv 1 \pmod p$.This is a special case of Euler's Totient Theorem, which states that if two integers b and m are coprime, then $b^{\Phi(m)} \equiv 1 \text{ mod}(m)$.

Picking Public and Private Keys

The first step in creating an RSA key involves picking two large primes of similar size, say p and q. We then compute the semiprime number $n = p * q$. In practice, for security, both p and q are primes with 100 or more digits. Next, we calculate $\Phi(n) = (p - 1)(q - 1)$. Lastly, we pick e and d as inverses of each other modulo $(p-1)(q-1)$, such that $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$, or $ed - 1 = k(p-1)(q-1)$ for some integer k.

RSA Encryption Works!

To prove this, we show that $(M^e)^d \equiv M \pmod n$, which is the same as showing that n is a divisor of $(M^e)^d - M$. Factoring and substituting in $ed - 1$ from above gives us

$$(M^e)^d - M = M(M^{ed-1} - 1) = M(M^{k(p-1)(q-1)} - 1).$$

We can show that $(M^e)^d \equiv M \pmod n$, that is n is a divisor of $(M^e)^d - M$, by showing that both p and q divide $M(M^{k(p-1)(q-1)} - 1)$. If p divides M, then it also divides $M(M^{k(p-1)(q-1)} - 1)$. Suppose p does not divide M. Then by Fermat's Little Theorem,

$$M^{p-1} \equiv 1 \pmod p, \text{ and so}$$
$$(M^{p-1})^{k(q-1)} = M^{k(p-1)(q-1)} \equiv 1 \pmod p.$$

Therefor, p is a divisor of $M^{k(p-1)(q-1)} - 1$, so p is also a divisor of $M(M^{k(p-1)(q-1)} - 1)$. We can repeat the same reasoning for q, so q divides $M(M^{k(p-1)(q-1)} - 1)$. Since p and q are distinct primes, then their product $p \cdot q = n$ also divides $M(M^{k(p-1)(q-1)} - 1)$, therefore

$$(M^e)^d \equiv M \pmod n \text{ as desired.}$$

In other words, RSA decryption is the inverse of RSA encryption.

Acknowledgements

Special thanks to Janet Talvacchia for her constant encouragement and robust advice, and to Lijia Liu, Tom Wang, Graham Doskoch, and Malini Kohli for peer reviews, insight, and encouragement.

References

[1] Galbraith, Steven D. *Mathematics of Public Key Cryptography.* Cambridge University Press, 2012.

[2] Sinkov, Abraham, and Todd Feil. *Elementary Cryptanalysis: a Mathematical Approach.* Mathematical Association of America, 2009.

[3] Hoffstein Jeffrey et. al *An Introduction to Mathematical Cryptography.* Springer New York, 2008.

[4] M. Düll, B. Haase, G. Hinterwäder, M. Hutter, C. Paar, A. H. Sánchez, P Schwabe, "High-speed Curve25519 on 8-bit 16-bit and 32-bit microcontrollers", Des. Codes Cryptogr, vol. 77, 2015.

[5] Y. F. Chen et al.,"Verifying Curve25519 Software", Proc. ACM SIGSAC Conf Computer and Communications Security (CCS 14), 2014.

[6] Wikipedia: Quantum Computing, `https://en.wikipedia.org/wiki/Quantum_computing`