A Short Guide to RSA Encryption

Sam Kennerly

May 25, 2009

This document attempts to explain the mathematics behind the RSA encryption scheme. No knowledge of encryption other than basic modular arithmetic is assumed of the reader.

RSA is an acronym for the last names of Ron Rivest, Adi Shamir, and Leonard Adleman, who published a description in 1978. (A similar system was designed but not published by Clifford Cocks in 1973 while he was working for British intelligence.) The RSA scheme was patented in the U.S. but has been public domain since September 2000.

RSA is known to be insecure against an adversary capable of either factoring large *semi-prime* integers or solving *discrete logarithms* efficiently. The practical value of RSA depends on the fact that all known non-quantum computer algorithms for both problems are impractically time-consuming for semi-primes with more than ~ 1000 binary digits. A working (and reliable) quantum computer can, however, solve both problems in polynomial time using an algorithm written by Peter Shor in 1994. Shor's algorithm is thus a topic of much discussion in quantum computing research.

1 Introduction to Encryption

Thought problems in encryption conventionally involve two cryptographers Alice and Bob sending messages to each other while an unauthorized observer, Eavesdropping Eve, attempts to read the messages and/or send fake messages of her own. Each original message is called a **plaintext** and its encrypted version is called a **ciphertext**. An ideal encryption scheme allows Alice and Bob - but not Eve - to translate each other's ciphertexts and to verify the origin of each message by checking an encrypted **digital signature**.

The simplest encryption scheme is a **substitution cipher** in which each character in the plaintext is replaced by another character to create the ciphertext. Substitution ciphers have been used for thousands of years, and all digital text includes some kind of substitution

rules for representing characters as a string of binary digits. (Examples include ASCII and Unicode.) One method is to replace each letter in the alphabet with a number, like so:

$$A \to 1 \qquad B \to 2 \qquad C \to 3 \qquad \dots \qquad Y \to 25 \qquad Z \to 26$$

Obviously, we could have sent A to any other number as well. The important thing is that the substitution map be **invertible** - there must exist some map from the ciphertext back to the plaintext so the message is readable. (The hard part is making the inverse map easy for Alice and Bob but extremely difficult for Eve.)

Another example is the Caesar cipher, in which every letter in the alphabet is shifted forward three letters to produce the cipher. This is equivalent to the method above but with an extra step: to each number, we add 3 (mod 26), then invert the substitution to write the result in letters. The ciphering scheme is then:

$$A \to D \qquad B \to E \qquad C \to F \qquad \dots \qquad Y \to B \qquad Z \to C$$

The plaintext "HAIL CAESAR" leads to the ciphertext "KDLO FDHVDU." The related scheme **ROT13** is identical except that the alphabet is "rotated" 13 letters instead of 3. In modular arithmetic, 13 + 13 (mod 26) = 26 = 0, so ROT13 has the property of being **self-inverse**: the encryption and decryption algorithms are identical.

$$HAIL\,CAESAR \quad \to \quad UNVY\,PNRFNE \quad \to \quad HAIL\,CAESAR$$

Caesar ciphers are especially weak because they are easy to guess, preserve word length in an obvious way, and apply the same transformation to every character in the plaintext. (ROT13 is often used as a joke among cryptographers.) A much more effective variation is a centuries-old scheme sometimes called the **Vignère cipher**:[1] before sending any messages, Alice and Bob agree on a secret password. To encrypt an N-character message, the password is then repeated (and truncated as needed) to produce an N-character encryption key. The key is then converted into numbers and a Caesar-shift of that many characters is applied to each corresponding letter in the message. Using modular arithmetic, each character X_i is replaced by $X_i + K_i$ (mod 26) where K_i is the i-th character in the key.

Password: "FIDELIO"
Message: "MEET ME AT MIDNIGHT." (16 characters, not counting spaces)
Key: FIDE LI OF IDELIOFI = 6 9 4 5 12 9 15 6 9 4 5 12 9 15 6 9
$M \to (M + 6) = S \quad E \to (E + 9) = N \quad E \to (E + 4) = I \quad \dots \quad T \to T + 9 = C$
Ciphertext: SNIY YN PZ VMIZRVNC

[1] The oldest known example was written by Giovan Bellaso, not Vignère, in 1553. Charles Dodgson, known by his pen name Lewis Carroll, published it in a children's magazine as "The Alphabet Cipher."

The Vignère cipher is an example of a **polyalphabetic** cipher and variants of it are often (erroneously) believed to be unbreakable.[2] Substitution ciphers are typically vulnerable to **frequency analysis**, in which Eve measures how often each character appears and then makes educated guesses as to what that letter represents. (For example, "E" is much more common in English than "Q".)

Frequency analysis is especially effective when a cipher is used many times. A simple and provably secure[3] method is the **one-time pad**: for an N-character message, an N-digit key is chosen *at random* and then the corresponding Caesar shifts are applied as in the Vignère cipher. The key must be written down and sent to the intended recipient. (In some historical cases, this was literally done using a pad of paper.) Each one-time pad must be used *exactly once* or else it too becomes vulnerable to frequency analysis.

2 Public-Key Encryption

The keys in the Vignère cipher and one-time pad are examples of **private keys** because they share a common inconvenience: the key itself must itself be somehow secretly shared between Alice and Bob before any message is sent.

Public-key encryption was developed as an alternative to this scheme. As an analogy, imagine that Alice wants to send a message that must not be read by anyone but Bob. Bob buys a padlock and keeps the only key for himself, then sends the (unlocked) padlock by mail to Alice. Alice locks her message in an armored box and locks it with the padlock, then mails the box to Bob. Even if Eve intercepts the message, she cannot read it. In this case, the padlock is Bob's **public key** and the key to open the lock is his **private key**.

If Eve wants to pretend she is Alice, she must intercept Bob's padlock and use it to lock away a fake message and mail it to Bob. To prevent this, Alice stamps all her messages with an elaborate wax seal that can be easily recognized by Bob but is extremely difficult to create without Alice's special stamping-tool. Here the wax seal is Alice's **public signature** and the stamping-tool is her **private (signature) key**. So long as both private keys are always kept private, Eve cannot interfere.

Digital public-key encryption schemes send, instead of padlocks and wax seals, difficult math problems that are much easier to solve (for sending) or create (for signing) by someone who knows a certain hint. That hint then becomes a private key and the math problems are used as public keys. In the case of RSA encryption, the difficult problems are:

[2]The legendary Enigma machine of Nazi Germany used a polyalphabetic cipher, but careless usage allowed Polish mathematicians to crack it and pass their methods on to Allied cryptographers.

[3]See Claude Shannon's proof published in 1949.

1. The **discrete logarithm problem**: Solve $\log_E[C] \pmod N$ where E is the public key (or "public exponent"), C is the ciphertext, and N is an *RSA number* sent along with the public key. C,E, and N are all large (> 1000 binary digits) integers. The "log" here means "take the logarithm base E with respect to modular multiplication (mod N)," or equivalently, "find M such that $M^E \pmod N = C$."

2. **Semi-prime factorization**: Given that N is the product of two prime numbers p and q, find p and q. (Such numbers are called **semi-prime**.) RSA private keys are generated using p and q, so if Eve can factor Alice's public RSA number N, she can create her own copy of Alice's private key.

3 RSA Encryption

Alice wants to send an integer M to Bob. Bob creates an RSA number, public exponent and private key in the following way. (Here Greek letters represent numbers that must be kept secret, while Roman letters except the message M are published for anyone to see.)

Before anything else, Bob uses a computer to generate two large prime numbers α and β. He multiplies them together to produce his RSA number N.

1. Find Euler's **totient** $\varphi(N)$, the number of *units*[4] in (mod N) arithmetic. Since N is semi-prime, it turns out that $\varphi = (\alpha - 1)(\beta - 1)$.

2. Choose a **public exponent** E in the range $1 << E < \varphi$. E must be **coprime** to φ, meaning that φ and E have no common factors except 1.

3. Find the multiplicative inverse (mod φ) of E. Call this number the **private key** δ. If φ is known, this can be accomplished using the *extended Euclidean algorithm*. (This is why it is important that Eve cannot factor N into α and β; if she could, then she could find φ and use the extended Euclidean algorithm to find δ.)

4. Send Alice his public key E and RSA number N.

Alice now creates an encrypted cipher C from M using Bob's key and RSA number:

$$C = M^E \pmod N$$

[4]In modular arithmetic, a **unit** is a number that has a multiplicative inverse. u is a unit if and only if $u^n \neq 0$ for any natural number n. Equivalently, u is a unit if and only if it is coprime to N. In (mod 4) arithmetic, for example, 1 and 3 are units while 0 and 2 are not. If p is prime, then every number except 0 is a unit (mod p), so $\varphi(p) = p - 1$. For a semi-prime $N = pq$, it can be shown that $\varphi(N) = (p - 1)(q - 1)$.

To decipher the message, Eve must solve the discrete logarithm problem $M = \log_E[C] \pmod{N}$. Bob, however, can use a shortcut involving δ:

$$C^\delta = (M^E)^\delta = M^{E\delta} \pmod{N}$$

Since $E\delta = 1 \pmod{\varphi}$, Bob knows that $\delta E = n\varphi + 1$ for some natural number n. According to the *Chinese remainder theorem*, the result simplifies to:

$$M^{E\delta} = M^{(n\varphi+1)} = M \pmod{N}$$

If $M < N$, then Bob has recovered the original message. If Alice wishes to send a longer message, she sends a string of integers $\{M_k\}$ such that each $M_k < N$.

In order to sign the message, Alice creates an **autograph** A that Bob will recognize. Using her own private key Δ (which is kept secret, even from Bob) and her own public key $\mathcal{E}$ and RSA number $\mathcal{N}$, she creates a **digital signature**:

$$D = A^\Delta \pmod{\mathcal{N}}$$

To verify the message, Bob exponentiates the signature:

$$D^\mathcal{E} = A^{\mathcal{E}\Delta} = A \pmod{\mathcal{N}}$$

If the decrypted signature A looks correct, Bob can be confident that whoever sent the message knows Alice's private key and therefore must be Alice.

In practice, it is important to note that actual digital signatures require *padding schemes* to be completely secure, and it is inadvisable to use the same keys for receiving and signing messages. Typically instead of an autograph, an encryption scheme will create a *hash* number based on the actual message data. This avoids sending the same signature twice.

There are still vulnerabilities in the RSA scheme - even against an Eve not armed with a quantum computer - if it is implemented carelessly. In particular, when Alice and Bob first communicate, they are vulnerable to a **man-in-the-middle attack**: if Eve has the ability to intercept both their messages, she can generate public and private keys and pretend to be Alice or Bob. The initial handling of keys is thus usually run by trusted authorities using *digital certificates*. As with any encryption scheme, Eve can still attempt **side-channel attacks** that attempt to read the message indirectly through some alternate means (e.g. watching Alice type the plaintext) without cracking the encryption scheme. These practical difficulties are, however, beyond the scope of this article.