# The RSA Algorithm

## Public-Key Cryptosystem

### History

The RSA algorithm was developed by three professors at MIT in 1977, Ron Rivest, Adi Shamir, and Leonard Adlemen, their initials give the algorithm its name. This was one of the first algorithms (mathematical processes) to implement the concept of public-key cryptography. The actual concept of public-key cryptography was discovered by Whitfield Diffie and Martin Hellman just one year earlier. Diffie and Hellman had devised a method that would allow the key to the cryptographic system to be known publicly but still have the system be a secure way to send messages. What they did not know was how to mathematically create one. Rivest, Shamir and Adlemen took the concept of Diffie and Hellman and devised a method that uses what is known as a one-way function. A one-way function is a mathematical process that is easy to do one way but is difficult to reverse. For example, it is easy to multiply two numbers together but it is much more difficult to factor a number. This is precisely why the RSA algorithm works. Your computer can multiply two 300 digit numbers together in an extremely small fraction of a second but if you ask your computer to factor a 600 digit number that is the product of two 300 digit primes you will be waiting a long time for the result. In fact, it would take the fastest supercomputer on Earth several billion years to factor the number.


Figure 1: Shamir, Rivest and Adleman[1]

As a historical note, Rivest, Shamir and Adlemen were not the first mathematicians to discover this technique. In 1973, Clifford Cocks, a British mathematician and cryptographer at the Government Communications Headquarters (GCHQ), had developed an equivalent system. The Government Communications Headquarters is a British intelligence agency responsible for providing signals intelligence and information assurance to the UK government and armed forces. GCHQ was originally established after the First World War as the Government Code and Cypher School (GC&CS or GCCS). During the Second World War it was located at Bletchley Park, which is where the British broke many of the German codes during World War II, including the Enigma machine. The GCHQ is the British equivalent of the NSA (National Security Agency) in the United States. Hence anything that was discovered at GCHQ had to remain classified and Clifford Cocks did not receive the recognition, or the money, from the discovery of the method. In fact, it was not until 1997 that GCHQ declassified his work.

[1]Photo taken in 1978, http://www.acm.org/fcrc/PlenaryTalks/rivest.pdf

### The Concept

In symmetric-key cryptography, the traditional method and the only one used before 1977, Alice and Bob would share an encryption and decryption key that only the two of them knew. When Alice wanted to send a message (plaintext) to Bob she would use the key to encrypt the message into cyphertext, she would then send the cyphertext to Bob where he would use the decryption key to decrypt the message back into plaintext and read what Alice had to say. In all transmissions we assume that a third person, Eve, could and does intercept the message. Now Eve does not have the key, she only has the cyphertext. It is her job to break the code either by finding the key and decrypting the message or by finding the meaning of the message without finding the entire key. One problem with this method was that Alice and Bob needed to exchange the key before they could send messages. If Alice and Bob were a long distance apart then they needed to either travel a great distance to meet or they needed to trust a third party to deliver the message. They could not send the key over the phone or e-mail since Eve could intercept the key and then decrypt all messages between Alice and Bob.

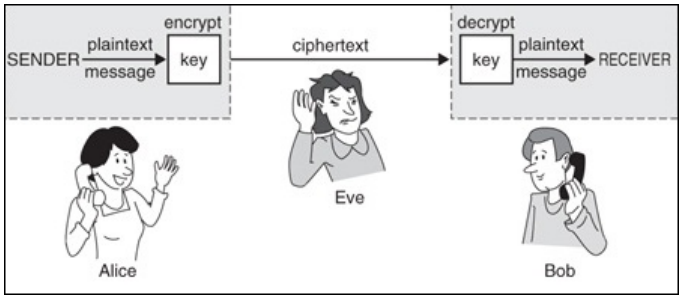


Figure 2: Symmetric-Key Cryptography

In public-key cryptography, for example in the RSA algorithm, the setup is slightly different. There is no need to exchange a key between Alice and Bob before any messages are sent. In this scheme, Bob creates an encryption key and a decryption key. He keeps the decryption key private, he is the only one who knows the decryption key. He then publishes the encryption key on the Internet so that everyone knows the encryption key, including Alice and Eve. Now when Alice wants to send Bob a message she uses the encryption key Bob published, encrypts the message and sends it to Bob. Bob then uses the decryption key to decrypt the message. Now Eve has both the cyphertext of the message and the encryption key. The decryption key can, of course, be calculated from the encryption key but since a one-way function was used in the creation of the keys, finding the decryption key from the encryption key is an extremely lengthy computation and may take years to solve. The beauty of the method is that no key exchange needs to take place and the transmission of the message is extremely secure.

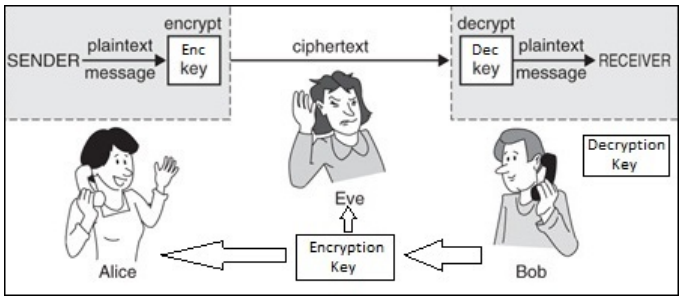


Figure 3: Public-Key Cryptography

**The Algorithm**

First, Bob creates the encryption and decryption keys as follows.

1. Bob chooses secret primes $p$ and $q$ and computes $n = pq$. In practice, the primes $p$ and $q$ are usually several hundred digits in length.
2. Bob chooses $e$ with $\gcd(e, (p-1)(q-1)) = 1$, that is $e$ and $(p-1)(q-1)$ have no common factors.
3. Bob computes $d$ with $de = 1 \mod (p-1)(q-1)$. Remember that mod just means clock arithmetic with a clock that has $(p-1)(q-1)$ numbers.
4. Bob makes $n$ and $e$ public and keeps $p$, $q$ and $d$ secret. So the public encryption key is $(n, e)$ and the private decryption key is $(p, q, d)$.

Now Alice is ready to send Bob a message and does the following.

1. Alice takes the message and converts it into a number $m$.
2. She then takes the $n$ and $e$ that Bob published and creates the cyphertext by computing

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

   Again, clock arithmetic with a clock that has $n$ numbers.
3. Alice then sends $c$ to Bob.

When Bob receives $c$ he decrypts the message by the following computation.

1. Bob takes the cyphertext $c$.
2. He computes

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

   The fact that $c^d \mod n$ turns out to be the number $m$ relies on a theorem from number theory known as Euler's generalization of Fermat's Little Theorem.
3. Bob converts the number $m$ back into the message.

So what about Eve? She has the cyphertext $c$ and the encryption key of $n$ and $e$. All she needs to decrypt the message is the number $d$. To get $d$ all she needs are $e$, $p$ and $q$ and since she has $e$ she really only needs $p$ and $q$. Here is where the strength of the method shows itself. To get $p$ and $q$, Eve must factor the number $n$. Factoring large integers is computationally infeasible, meaning that the computation can be done but it would take even the fastest computers on Earth many years to do.

**Example**

To get a feel for how the method works we will look at a small example.

1. Bob uses the primes $p = 7$ and $q = 11$. Then he computes $n = 7 \cdot 11 = 77$.
2. Next he chooses a number $e$ so that $e$ and the number $(p-1)(q-1) = 6 \cdot 10 = 60$ have no common factors. He chooses $e = 13$.
3. Now Bob finds the number $d$ such that $de = 1 \mod (p-1)(q-1)$. A little calculation and we get $d = 37$.
4. Bob publishes $n = 77$ and $e = 13$.

Now Alice wants to send him the very short message "H".

1. "H" is the eighth letter in the alphabet so she converts the message to $m = 8$.
2. Now Alice encrypts the message as

$$c = 8^{13} \mod 77 = 549755813888 \mod 77 = 50 \mod 77$$

3. Alice sends 50 to Bob.

Once Bob receives the encrypted message of 50 he does the decryption calculation.

1. Bob takes the cyphertext $c = 50$.
2. He computes

$$\begin{aligned} m &= 50^{37} \mod 77 \\ &= 727595761418342590332031250000000000000000000000000000000000 \mod 77 \\ &= 8 \mod 77 \end{aligned}$$

3. Bob converts the number $m = 8$ back into the message "H".

**Applications**

Public-key systems, like the RSA algorithm, are commonly used in transmission of secure data over e-mail and the Internet. When you log into a site or make a purchase your passwords and credit card numbers are encrypted with either the RSA algorithm or an equivalent method that utilizes another one-way function.

Methods like this are commonly used for small amounts of data. When transmitting large amounts of data, say gigabytes or terabytes of information, the use of RSA would be too slow for the average computer. So what is commonly done is the RSA algorithm is used to transfer a symmetric key for a faster encryption algorithm and the faster algorithm is used to encrypt and decrypt the large data set.