RSA

The most commonly used public-key cryptosystem is RSA, which is named after its three developers Ron Rivest (b. 1947), Adi Shamir, and Leonard Adleman (b. 1945). At the time of the algorithm's development (1977), the three were researchers at the MIT Laboratory for Computer Science. Their algorithm was first announced in Martin Gardner's "Mathematical Games" column in the August, 1977, *Scientific American*. Their formal paper "A method for obtaining digital signatures and public-key cryptosystems" was published in 1978 in the *Communications of the Association for Computing Machinery*. RSA was patented, but the patent expired in 2000. A British mathematician Clifford Cocks who was working for CESG developed a similar algorithm in 1973; however, there is no public record that it was implemented.

The Merkle-Hellman cryptosystem was an attempt to convert a one-way function based upon the combinatorial knapsack problem to a trapdoor function. RSA and ElGamal, the two public-key cryptosystems that we will examine, are both based upon number-theoretic problems. RSA is based upon the fact that it is easy to multiply together two large primes but it is hard to factor a large integer. ElGamal is based upon the discrete logarithm problem – the fact that it is easy to raise *a* to the power *n* modulo *p* but it is hard to determine modulo *p* the logarithm (i.e., the exponent) base *a* of *b*. If some mathematician wakes up tomorrow and has a solution of the factoring problem, RSA-encrypted messages could be decrypted. Similarly, if some mathematician wakes up tomorrow and has a solution to the discrete logarithm problem, ElGamal-encrypted messages could be decrypted. Research in number theory can have significant impact on these public-key cryptosystems. To avoid "putting all of one's eggs in one basket," cryptosystems based upon other than number theoretic problems have been developed, but they are not as popular as RSA and ElGamal.

RSA is based upon a number theoretic result due to the Swiss mathematician Leonard Euler (1707 –1783) (pronounced "oiler"). In the form in which it is used in the RSA cryptosystem, this corollary states that if *p* and *q* are prime numbers and *a* is an integer that is not divisible by either *p* or *q*, then $a^{(p-1)(q-1)} = 1 \bmod pq$.

Two large prime numbers *p* and *q* are needed to construct the keys. The modulus $n = pq$. While, in practice, the modulus is typically 1024 bits or longer, for our example, we will use very small primes, say $p = 53$ and $q = 61$. Then $n = pq = 3233$. *p* and *q* are not public, but *n* is public. The security of the RSA algorithm depends on the attacker not being able to factor *n* into its two prime factors *p* and *q*.

The encryption exponent *e* can be any integer that has a multiplicative inverse modulo $(p - 1)(q - 1)$. For our example, we will choose $e = 37$, which has a multiplicative inverse modulo

$$(p-1)(q-1) = (53-1)(61-1) = 52 \times 60 = 3120 .$$

Using the Euclidean algorithm (see the appendix Finding multiplicative inverses), the multiplicative inverse of *e* modulo $(p-1)(q-1)$ is calculated; that number is the decryption exponent *d*. For our example, $d = 253$. *e* is public, and *d* is private.

For our example, Alex will send a message to Nick. Alex looks up Nick's public key: *n* and *e*. These two numbers are available to anyone who might want to send an encrypted message to Nick. Nick keeps the decryption exponent *d* secret; it is his private key. The private key is used by Nick to decrypt any messages sent to him that have been encrypted with his public key.

Alex wants to send the message `math` to Nick. First, the message must be converted to a string of numbers. In practice, ASCII (American Standard Code of Information Interchange) numbers are usually used; we will use our usual `a` = `00`, `b` = `01`, …, `z` = `25`. So, `math` would be converted to the string `12001907`. The string is broken into blocks each of which is small than the modulus -- 3233 in our case. Four-digit blocks will work: 1200 1907.

The ciphertext message `C` is obtained by raising each plaintext message block `m` to the exponent *e* modulo *n*.

$$\mathtt{m}^e \bmod n = \mathtt{C}$$

$$1200^{37} \bmod 3233 = 2223$$

$$1907^{37} \bmod 3233 = 3126$$

The message that Alex transmits to Nick is 2223 3126.

How does Nick decrypt the message from Alex?

Decryption depends on Euler's result that $a^{(p-1)(q-1)} = 1 \bmod pq$. Because $n = pq$, this says that $a^{(p-1)(q-1)} = 1 \bmod n$. Now *e* and *d* are inverses modulo $(p-1)(q-1)$; i.e., $ed = 1 \bmod (p-1)(q-1)$. Another way of saying this is that *ed* is 1 plus a multiple of $(p-1)(q-1)$; i.e., $ed = 1 + k(p-1)(q-1)$, for some integer *k*.

Recall that ciphertext is $\mathtt{C} = \mathtt{m}^e \bmod n$. When Nick receives the ciphertext blocks, he raises each of them to the power *d*, his decryption exponent.

$$\mathtt{C}^d = (\mathtt{m}^e)^d = \mathtt{p}^{ed}$$

$$= \mathtt{m}^{1+k(p-1)(q-1)} = \mathtt{m}^1 \mathtt{m}^{k(p-1)(q-1)}$$

$$= \mathtt{m}\left(\mathtt{m}^{(p-1)(q-1)}\right)^k = \mathtt{m}(1)^k = \mathtt{m} \bmod n$$

which is the plaintext message.

Here is the decryption for our example.

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

$$2223^{253} \bmod 3233 = 1200$$

$$3126^{253} \bmod 3233 = 1907$$

So, the message decrypts to `1200 1907` which converts to `math`.

For Beth to cryptanalyze Alex's message to Nick, she would have to be able to construct Nick's private key *d*, which is the inverse of *e* modulo $(p - 1)(q - 1)$. By using the extended Euclidean algorithm, she can construct the inverse of *e* easily if she knows *p* and *q*, but to get *p* and *q*, Beth would need to be able to factor *n*, which is part of Nick's public key. That is the problem for her because there is no efficient way to factor large integers.

However, in the future, factoring of large integers in polynomial time might be possible; then RSA-encrypted messages could be broken. In 1994, Peter Shor (b. 1959), who was then working at Bell Labs, discovered a quantum computer algorithm that would factor large integers in polynomial time. Fortunately for cryptographers the construction of a quantum computer that could take advantage of the algorithm and break an RSA-encrypted message is not in sight; the best effort so far: in 2001, a 7-qubit quantum computer was able to use Shor's algorithm to factor 15.

RSA is used because it is a survivor. RSA has been attacked since its development in 1977. Mathematicians have improved factoring algorithms, but cryptographers have countered by increasing key sizes. The quantum computer is a threat, but not an immediate one. The algorithm has been so thoroughly tested that it is felt that its weaknesses are known – there are, for example, good and bad choices for *p* and *q* and *e* – but the known problems can be dealt with in implementation. In particular, because cryptology is now done in the open in universities and not just in secret in black chambers, algorithms can be publicly tested by experts. Cryptosystems need to earn trust, and RSA has earned it. But, an unexpected breakthrough in factoring algorithms could immediately doom it.

New algorithms should be viewed with suspicion. Remember, Merkle-Hellman looked good for a while. A cryptosystem that has known weaknesses can be a better choice than a cryptosystem that has not been thoroughly tested. Over many years, DES became a trusted cryptosystem. Its replacement AES was selected in a very open way that gave it a lot of initial trust, but now that it has been selected it will even more carefully probed for weaknesses.