Public-key cryptography

Rosalie Iemhoff

Public-key cryptography is a way to encode messages via a public (encoding) key, such that the receiver is the only one who can decode it.

There are two algorithms: an *encoding* algorithm $E(e, y)$ that given the number e encodes a message x into a number $E(e, x)$, and a *decoding* algorithm $D(d, y)$ that given the number d decodes a message y in such a way that if $y = E(e, x)$, then $D(d, y) = x$, that is, $D(d, E(e, x)) = x$.

The intriguing thing is that the receiver, Bob, can construct the numbers d and e in such a way that key e could be made public, known to everybody, while d is kept private, only known to Bob. The point is that it is computationally infeasable to obtain d from e, that is, it cannot be done in polynomial time, given that $\mathsf{NP} \neq \mathsf{P}$. Thus knowing e will not help to decode the encoded message $E(e, x)$.

How does Bob construct e and d? The method used for this is called RSA after the inventors Ron Rivest, Adi Shamir, and Len Adleman, who proposed this construction in 1978. It works as follows. Recall that $(y = x \bmod z)$ means that x is the rest of y divided by z.

Bob picks two large prime numbers p en q and computes a number e that is relative prime to n (e and n have no common divisors besides 1), where $n = \phi(pq) = pq - p - q + 1$. pq is p times q, and $\phi(n)$ is the *Euler function* on n, which value is the number of numbers smaller that n that are relative prime to it. We omit the technical details here, it is enough to know that this function has one specific property described below.

Bob announces both e and pq. This is the public encryption key known to all. Bob also constructs a number d such that $ed = 1 \bmod n$ that he keeps private. Thus $ed = 1 + mn$ for some number m.

Suppose Alice would like to send an encoded message x to Bob. Using the public key she computes the number $(x^e \bmod pq)$, let us call it y. Thus the encoding $E(e, x)$ equals $(x^e \bmod pq)$. Alice sends y to Bob.

When Bob receives y he simply raises it to the dth power: y^d. Thus the decoding $D(d, y)$ equals y^d. Now

$$y^d = x^{ed} = x^{1+mn} \bmod pq = x \bmod pq.$$

We are not going to prove these identities, but for those who would like to know the details: it is based on Fermat's Little Theorem which implies that $x^{mn} \bmod pq = 1 \bmod pq$. The theorem can easily be found on the web. Thus Bob divides y^d by pq and knows that the rest he finds is the message from Alice.

Why is this algorithm safe? It is a mathematical fact, which we will not prove, that d can be computed from p, q, and e using Euclid's algorithm. Euclid's alogrithm runs in polynomial time, but finding the prime factorization of a number is not known to be in P, although it clearly is in NP. Thus knowing pq will not help us find p and q efficiently, and thereby keeps this system, that is used every second over the whole world, safe. However, when it would turn out that $\mathsf{P} = \mathsf{NP}$, an effcient algorithm might be found, and who knows what happens then ...