ElGamal

In 1985, Tahar Elgamal (b. 1955) published a public-key cryptosystem based upon another number theoretic problem: the discrete logarithm problem ("A public key cryptosystem and signature scheme based upon discrete logarithms," *IEEE Transactions on Information Theory*, 31(4), 469 – 473). (Elgamal worked for several years for Netscape where he helped develop SSL; he also developed the algorithm that was chosen by the NIST as the digital signature standard [DSS]. The ElGamal cryptosystem is based upon the Diffie-Hellman key exchange algorithm, which we will explore later.)

Recall that a logarithm is an exponent. Another way of expressing that 2 raised to the power 5 is 32 is that the logarithm of 32 to the base 2 is 5: $\log_2 32 = 5$..

It is relatively easy to raise numbers to powers; e.g., $3^{20} = 3486784401$, but it is harder to find a logarithm; e.g., what is the base-2 logarithm of 8589934592? . $2^{?} = 8589934592$.

For real numbers, we can solve $2^{?} = 8589934592$ by playing the "high-low" game. For example, $2^{10} = 1024$, which is too small. We need a larger exponent. $2^{20} = 1048576$; which is still too small. $2^{30} = 1073741824$ -- too small. $2^{40} = 1099511627776$ -- too large. $2^{35} = 34359735368$ -- still too large. Finally, we find that $2^{33} = 8589934592$. So, the base-2 logarithm of 8589934592 is 33. This scheme works because $y = \log_2 x$ increases monotonically with *x*.

The discrete logarithm problem refers to the problem of finding logarithms modulo some integer. Recall that when we mod out by an integer *n*, we are left with only finitely many integers – a discrete set – usually represented as 0, 1, 2, … *n*-1.

The discrete logarithm problem asks for a solution of something like this: $2^{?} = 9 \bmod 11$.

Raising an integer to powers modulo *n* scrambles the order of the results. For example, here are the powers of 2 modulo 11:

$$2^1 = 2 \bmod 11$$
$$2^2 = 4 \bmod 11$$
$$2^3 = 8 \bmod 11$$
$$2^4 = 5 \bmod 11$$
$$2^5 = 10 \bmod 11$$
$$2^6 = 9 \bmod 11$$
$$2^7 = 7 \bmod 11$$

$$2^8 = 3 \bmod 11$$

$$2^9 = 6 \bmod 11$$

$$2^{10} = 1 \bmod 11$$

So, the answer to our discrete logarithm problem is 6: $2^6 = 9 \bmod 11$.

Because the powers of 2 modulo 11 do not increase monotonically with the exponent, we cannot play the "high-low" game. There is no known efficient algorithm for computing discrete logarithms.

It is easy to discrete exponentiation, but it is hard to determine discrete logarithms. The ElGamal cryptosystem and the Diffie-Hellman key exchange system are based upon the discrete logarithm problem.

Notice that in the example above, the modulus 11 is a prime and the ten powers of the base 2 are representatives of the ten nonzero equivalence classes modulo 11: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. We describe this property by saying that 2 is a primitive root of (the prime) 11. A prime always has at least one primitive root. Because all ten of the nonzero integers modulo 11 occur as powers of 2, it could have taken 10 trials to discover that $2^6 = 9 \bmod 11$.

If a primitive root is not used for the base of the exponentiation, not all of the remainders modulo n will occur as powers of the base. For example, 3 is not a primitive root of 11. Let us try to solve $3^? = 5 \bmod 11$.

Computing the powers of 3 modulo 11 results in:

$$3^1 = 3 \bmod 11$$

$$3^2 = 9 \bmod 11$$

$$3^3 = 5 \bmod 11$$

$$3^4 = 4 \bmod 11$$

$$3^5 = 1 \bmod 11$$

$$3^6 = 3 \bmod 11$$

$$3^7 = 9 \bmod 11$$

$$3^8 = 5 \bmod 11$$

$$3^9 = 4 \bmod 11$$

$$3^{10} = 1 \bmod 11$$

Notice that only five of the representatives modulo 11 occur: 3, 9, 5, 4, 1. When we try to solve $3^{?} = 5 \bmod 11$, we are twice as likely to find a logarithm because there are only half as many possible powers. Either 3 or 8 can be used as the logarithm of 5 for base 3 and modulo 11. It is not absolutely essential that we pick the base of the exponentiation to be a primitive root of the modulus; however, if there is too much repetition, the number of trials needed to determine a logarithm would be reduced considerably.

There are tables of primitive roots for various moduli.

To keep calculation to a level that can be done on a calculator, we will do an example using a small modulus. Let the modulus be the prime $p = 2539$. 2 is a primitive root of 2539; we will take the base $a = 2$.

Nick wants to send the message `nicholas` to Alex. Setting `a` = `00`, `b` = `01`, …, `z` = `25`; `nicholas` becomes `1308020714110018`.

The message must be split into strings that are integers mod p 2539. Say,

`1308  0207  1411  0018`

To construct the remainder of his public key, Alex selects some positive integer $n < 2539 - 1$ and calculates $b = a^n \bmod p$. Say, $n = 51$; then $b = 2^{51} \bmod 2539 = 403$. Alex's public-key is $p = 2539$, $a = 2$, and $b = 403$. The private-key is $n = 51$. For large values of the prime p, the private key cannot be easily determined from knowing b.

Nick chooses a positive integer $k < 2539 - 1$ and calculates $y = a^k \bmod p$ Say, Nick selects $k = 15$. $y = 2^{15} \bmod 2539 = 2300$.

For each block of ciphertext `m`, Nick calculates $z = \mathtt{m}b^k \bmod p$.

$$z_1 = 1308 \times 1794 \bmod 2539 = 516$$
$$z_2 = 207 \times 1794 \bmod 2539 = 664$$
$$z_3 = 1411 \times 1794 \bmod 2539 = 2490$$
$$z_4 = 18 \times 1794 \bmod 2539 = 1824$$

The ciphertext message sent by Nick to Alex consists of the pairs y and z for each plaintext block.

`(2300, 516)  (2300, 664)  (2300, 2490)  (2300, 1824)`

How does Alex decrypt the message? Recall that Alex has some additional information; he knows n, the base-a logarithm of b. Anyone other than Alex "possesses" this information only

buried in the number $b = a^n \bmod p$. n cannot be determined from b without solving the discrete logarithm problem $b = a^? \bmod p$. Here is how Alex uses n to decrypt the message.

For each pair (y, z), using his knowledge of n, Alex calculates $zy^{-n} \bmod p$. The calculation

$$zy^{-n} \bmod p = mb^k (a^k)^{-n} \bmod p = m(ba^{-n})^k \bmod p = m(1)^k \bmod p = m$$

returns plaintext.

For our example,

$$y^{-n} = 2300^{-51} \bmod 2539 = (2300^{51})^{-1} \bmod 2539 = 1794^{-1} \bmod 2539 = 593 \bmod 2539$$

$$\mathtt{m}_1 = z_1 y^{-n} = 516 \times 593 \bmod 2539 = 1308$$
$$\mathtt{m}_2 = z_2 y^{-n} = 664 \times 593 \bmod 2539 = 207$$

$$m_3 = z_3 y^{-n} = 2490 \times 593 \bmod 2539 = 1411$$

$$\mathtt{m}_4 = z_4 y^{-n} = 1824 \times 593 \bmod 2539 = 18$$

`1308 0207 1411 0018` which was the plaintext message. Notice that the ElGamal cryptosytem essentially doubles the length of a plaintext message.