ElGamal Cryptosystems

The ElGamal Cryptosystem is implicitly based on the difficultly of finding a solution to the discrete logarithm in $\mathbb{F}_p^*$: given a primitive element a of $\mathbb{F}_p^*$ and another element b, the discrete logarithm problem (DLP) is the computational problem of finding $x = \log_a(b)$ such that $b = a^x$.

Efficient algorithms for the discrete logarithm problem would render the ElGamal Cryptosystem insecure, the possibly weaker Diffie-Hellman problem (DHP) is the precise problem on which the cryptosystem is based: given $b = a^x$ and $c = a^y$ in $\mathbb{F}_p^*$, compute a^{xy}.

Note that a^{xy} can not be formed as any obvious algebraic combination of a^x and a^y like $a^x a^y = a^{x+y}$. In fact, other cryptosystems rely on the difficult of the Decision Diffie-Hellman problem (DDHP) being hard: given a^x, a^y and c, decide whether or not $c = a^{xy}$. Both the DHP and the DDHP are easy of the DLP is easy.

Definition. Recall that an element a of $\mathbb{F}_p^*$ is said to be *primitive* if and only if

$$1, a, a^2, \ldots, a^{p-2}$$

are all distinct. Primitive elements always exist in any finite field.

ElGamal Protocol

Public key: (a, a^x, p) where p is a prime, a is a primitive element of $\mathbb{F}_p^*$, and x is an integer $1 \leq x < p - 1$.
Private key: The integer x.
Initial setup:
1. Alice obtains Bob's public key (a, a^x, p).

For each message m Alice $\rightarrow$ Bob:
1. Alice chooses a private element y randomly in $1 \leq y < p - 1$.
1. Alice $r = a^y$ and $s = ma^{xy}$.
2. Alice sends the ciphertext message $c = (r, s)$ to Bob.
3. Bob deciphers the ciphertext message as $m = r^{-x}s \bmod p$.

The correctness of the deciphering is verified as follows:

$$r^{-x}s = (a^y)^{-x}ma^{xy} = ma^{-yx}a^{xy} = ma^{yx-xy} = m.$$

Disrete Logarithms

The main known attack on an ElGamal cryptosystem is to solve the discrete logarithm problem: given both a and a^x (in the finite field $\mathbb{F}_p$), find the value for x. In order for the discrete logarithm problem (DLP) to be hard, it is not enough to choose any prime p. One needs to select a prime p such that $p - 1$ has a large prime factor. Suppose, on the contrary, that $p - 1$ is divisibly only by primes less than some positive integer B. Such a number is said to be B-smooth. The DLP can be reduced to solving a small number of discrete logarithm problems of "size" B rather than of size $p - 1$.

As an example, let r be a prime divisor of $p-1$, and let $m = (p-1)/r$. Suppose that we want to solve for x such that $b = a^x$. The exponent is defined up to multiples of $p-1$. If we raise both sides to the power m, then for the problem $b^m = a^{mx}$ a solution x is well-defined up to multiples of r:

$$a^{m(x+r)} = a^{mx+mr} = a^{mx}a^{p-1} = a^{mx},$$

since $a^{p-1} = 1$.

If we now find that $p-1 = r_1 r_2 \cdots r_t$ for pairwise distinct primes r_i, the by the Chinese remainder theorem the value of $x \bmod p-1$ can be determined from its modular values $x \bmod r_i$, for all $1 \le i \le t$. So the hardness of the DLP determined by the size of the largest prime divisor of $p-1$.

Exercise. Suppose that a prime power r^k divides $p-1$. How would you solve the DLP for $x \bmod r^k$?

Algorithmic Considerations

A naïve algorithm for solving the discrete logarithm problem for $\log_a(b)$ is to compute $1, a, a^2, \ldots$ until a match is found with b. As we have just seen, it is possible to replace a with $a_1 = a^m$ and b with $b_1 = b^m$ in order to solve $\log_{a_1}(b_1)$ modulo r such that $rm = p-1$. In this way we have to build the list $1, a_1, a_1^2, \ldots, a_1^x$ of length at most r before finding b_1.

An alternative approach is called the baby-step, giant-step method. We set $s = [\sqrt{r}]+1$ and to form a first list $1, a_1, a_1^2, \ldots, a_1^{s-1}$ of length s, called the baby steps, then form the second list $b_1, a_1^s b_1, a_1^{2s} b_1, \ldots, a_1^{s^2} b_1$ of giant steps, to find a match.

If a match is found, say $a_1^i = b_1 a_1^{js}$, then we have found $b_1 = a_1^{i-js}$, so $x = i - js \bmod r$. On the other hand, if x is a solution to the DLP $\bmod r$, then we can write $x = i - js$ for some $0 \le i, -j \le s$, so the above algorithm finds a match.

Diffie–Hellman Key Exchange

Diffie and Hellman proposed the following scheme for establishing a common key. The scheme is widely used because of the simplicity of its implementation, however an naive implementation without identity authentication leaves the protocol subject to a man-in-the-middle attack.

1. A and B decide on a large prime number p and a primitive element a of $\mathbb{Z}/p\mathbb{Z}$, both of which can be made public.

2. A chooses a secret random x with $\mathrm{GCD}(x, p-1) = 1$ and B chooses a secret random y with $\mathrm{GCD}(y, p-1) = 1$.

3. A sends Bob $a^x \bmod p$ and Bob sends Alice $a^y \bmod p$.

4. Each is able to compute a session key $K = a^{xy} = (a^x)^y = (a^y)^x$.

An eavesdropper only has knowledge of p, a, a^x and a^y, and would need to break the Diffie-Hellman problem to be able to come up with the session key.

Man in the Middle Attack

The man-in-the-middle attack is a protocol for an eavesdropper E to intercept a message exchange between A and B. The attack is premised on a Diffie-Hellman key exchange, but the principle applied to any public key cryptosystem for which the keys used for public key exchange is not certified with a cerification authority.

We assume that A and B have agreed on a prime p and a primitive element a of $\mathbb{Z}/p\mathbb{Z}$, and that E is positioned between A and B. Having observed this Diffie-Hellman initialization E prepares for the man-in-the-middle attack.

1. A chooses a secret key x, creates a public key a^x, and sends it to B, which is intercepted by E.

2. E chooses a private integer z at random, and creates the alternative public key a^z which she sends to B, pretending to be A. At the same time she sends same key a^z to A, now posing as B.

3. Now E has established a common session key a^{xz} with A and common session key a^{yz} with B. Message exchanges between A and B pass through E and can be deciphered, read, modified, re-enciphered, and resent in transit.

The breakdown of the key exchange protocol is due to lack of identity authentication of the communicating parties. If, for instance the public key (a, a^x, p) of A could be confirmed with an independent certification authority, then B would not have confused E with A.