Diffie-Hellman, Implementation

Based on *Cryptography Engineering* by Schneier, Ferguson, Kohno, Chapter 11

Lecture notes of Alexander Wood
awood@jjay.cuny.edu

John Jay College of Criminal Justice

Diffie-Hellman Key Exchange

Public information: Large prime p, a generator $g \in_R \mathbb{Z}_p^*$.

Alice		**Bob**
$x \in_R \mathbb{Z}_p^*$		$y \in_R \mathbb{Z}_p^*$
	$\xrightarrow{g^x \pmod p}$	
	$\xleftarrow{g^y \pmod p}$	
$\kappa = g^{xy} \pmod p$		$\kappa = g^{xy} \pmod p$

Pitfall: g Is Not Primitive

What if g is not a primitive element in $\mathbb{Z}_p^*$? Recall this means that the subgroup generated by g is *not* the whole group!

Pitfall: g Is Not Primitive

Say g is a non-primitive element of order $1,000,000$. This means that the set $\{1, g, g^2, \ldots, g^{q-1}\}$ contains exactly $1,000,000$ elements.

This is effectively small enough for Eve to carry out a brute-force search for the generator.

Pitfall: g Is Not Primitive

To protect against this we see it is important that Alice and Bob are able to verify that p and g were both chosen properly.

Namely, they need to check that p is a “large” prime and that g is primitive modulo p.

Pitfalls: Eve Intercepts!

What if Eve intercepts the values g^x and g^y sent by Bob and Alice and replaces them with 1?

Alice		**Eve**		**Bob**
$x \in_R \mathbb{Z}_p^*$				$y \in_R \mathbb{Z}_p^*$
	$\xrightarrow{g^x}$		$\xrightarrow{1}$	
	$\xleftarrow{g^y}$		$\xleftarrow{g^y}$	
$\kappa = g^{xy}$				$\kappa = 1$

The protocol *looks* like it completed correctly to Alice And Bob – except now Eve knows the key Bob is using!

Pitfalls: Eve Intercepts!

This scenario is easy enough to protect against – just have Bob check that his key is not 1.

There are more sophisticated methods Eve can use, however, which are less easily detectable by Alice or Bob.

Small Order Element Replacement Attack

Instead of replacing g^x with 1, now Eve will replace it with $h \in \mathbb{Z}_p^*$ where h has small order.

Small Order Element Replacement Attack

Recall that the **order** of an element h modulo p is the smallest positive value q which satisfies

$$h^q = 1 \pmod{p}.$$

This means that h **generates** a q-element subgroup of $\mathbb{Z}_p^*$,

$$\{1, h, h^2, \ldots, h^{q-1}\}$$

Small Order Element Replacement Attack

So when we say that Eve replaces g^x with h such that h has small order, we deduce that the subgroup generated by h is small!

How can Eve use this to her advantage?

Small Order Element Replacement Attack

Let's look at this attack in more detail.

Alice		**Eve**		**Bob**
$x \in_R \mathbb{Z}_p^*$				$y \in_R \mathbb{Z}_p^*$
	$\xrightarrow{g^x}$		$\xrightarrow{h}$	
	$\xleftarrow{g^y}$		$\xleftarrow{g^y}$	
$\kappa = g^{xy}$				$\kappa = h^y$

Eve knows h, but not y. How can Eve determine the value of h^y?

Small Order Element Replacement Attack

Eve knows h, and knows that Bob's key will be h^y for some y. All Eve has to do to decrypt a message Bob sends is try every possible value which h^y can take – which is not very many, since the subgroup generated by h is small.

How can Alice and Bob protect themselves against *this* attack?

Small Order Element Replacement Attack

To protect against this attack, Alice and Bob must verify that they numbers they receive from one another do not generate small subgroups.

Subgroups

We should notice a common theme – Alice and Bob need to be able to deduce information about subgroups modulo $\mathbb{Z}_p^*$.

Subgroups

All multiplicative subgroups modulo a prime p can be generated from a single element. This means that each subgroup can be written as

$$\{1, h, h^2, h^2, \ldots, h^{q-1}\}$$

for some $h \in \mathbb{Z}_p^*$, where q is the order of h.

Subgroups

The order q of any subgroup of $\mathbb{Z}_p^*$ must be a divisor of $p-1$.

Conversely, for every divisor q of $p-1$, there is a subgroup of $\mathbb{Z}_p^*$ of order q.

How can we use this information to avoid small subgroups modulo p?

Subgroups

How can we use this information to avoid small subgroups modulo p?

We must avoid having small divisors of $p - 1$. Why is this a problem?

Subgroups

We want $p - 1$ to have no small divisors. However every large prime is odd, meaning $p - 1$ is always even. Hence 2 is always a small divisor of $p - 1$.

We work around this by asserting that $p - 1$ have no small factors besides 2.

Safe Primes

We wish to define a safe prime for p which we may use in the protocol. A **safe prime** p is a “large enough” prime of the form

$$p = 2q + 1$$

where q is also prime.

Safe Primes

For a safe prime p we can determine all subgroups of $\mathbb{Z}_p^*$. In fact, the subgroups are given by

(1) The trivial subgroup, $\{1\}$

(2) The subgroup of order 2, given by $\{1, p-1\}$

(3) The subgroup of size q.

(4) The full group of size $2q$.

Safe Primes

It is easy to avoid subgroups (a) and (b). We will wish to use subgroups of form (c), which are of order q.

We need a way to distinguish between subgroups of order q and subgroups of order $2q$.

Squares Modulo p

Consider the set of all numbers x which are **square** modulo p. This means that there exists a $y \in \mathbb{Z}_p^*$ such that

$$x = y^2 \pmod{p}$$

In other words, x is the square of some number y.

Squares Modulo p

Exactly half of the numbers modulo p are squares, half are not squares.

A generator of the entire group is a non-square. Why?

Squares Modulo p

A square element modulo p can never generate a non-square. Observer, if $x = y^2 \pmod{p}$, then

$$x^j = (y^j)^2 \pmod{p}$$

for all integers j.

Squares Modulo p

Therefore, **any generator of the entire group is a non-square.**

Furthermore, **any generator of the subgroup of order q is a square.**

Legendre Symbol

A mathematical function called the **Legendre symbol** is able to calculate whether a number is square modulo p efficiently.

Alice and Bob should use an efficient Legendre symbol algorithm to verify that g is a non-square.

Legendre Symbol

If x is odd, then g^x is a non-square. If x is even then g^x is square. Therefore, Eve is able to deduce whether Alice's private value x is even or odd.

To avoid this problem, use only squares modulo p. This is the subgroup of order q. Since q is prime, there are no further subgroups for us to worry about!

Finding Safe Primes

The following is an algorithm for using a safe prime.

(1) Choose (p, q) such that $p = 2q + 1$, and p and q are prime.
(2) Choose a random α in the range $2, \ldots, p - 2$.
(3) Set $g = \alpha^2 \pmod{p}$.
(4) Check that $g \neq 1$ and $g \neq p - 1$. If g equals one of these values, repeat from step (2).
(5) Output parameter (p, q, g), suitable for Diffie-Hellman.

Using Smaller Subgroups

While the safe prime approach works, it is inefficient. For an n-bit prime p, the value q is $n - 1$ bits long and hence all exponents are $n - 1$ bits long. Exponentiation takes, on average, $3n/2$ multiplications modulo p.

Using Smaller Subgroups

We can work around this by using smaller subgroups. However, we must be careful when we do this to avoid the security pitfalls mentioned earlier.

Using Smaller Subgroups

Following is an algorithm for generating the security parameter using a smaller subgroup.

(1) Choose a `256`-bit prime q.
(2) Find a prime $p = Nq + 1$ for a large arbitrary (even) value N. (p should be thousands of bits long.)
(3) Find an element of order q as follows:
    - $\alpha \in_R \mathbb{Z}_p^*$, set $g \leftarrow \alpha^N$
    - Verify that $g \neq 1$ and $g^q = 1$. Repeat until this is satisfied.
(4) Output parameter (p, g, q).

Using Smaller Subgroups

The values Bob and Alice exchange are in the subgroup generated by g, which is now of order q. To verify that Eve is not substituting a different value of g Alice and Bob should now check that the number they received, say r, is in fact in the subgroup generated by g. This reduces to verifying the following:

- $r \neq 1$, and
- $r^q = 1 \pmod{p}$, and
- $1 < r < p$

Using Smaller Subgroups

Why is this faster than the safe primes case? Well, since $r^q = 1$ for all r in the subgroup generated by g, we have that

$$r^e = r^{(e \mod q)} \pmod p$$

which is much faster to compute!

Efficiency of Using Smaller Subgroups

The prime p is at least `2000` bits long! With safe primes it takes 3000 multiplications to compute g^x; in the smaller subgroups case, it takes only 384 multiplications. Wow!

Size of p

In symmetric-key protocols we looked at previously, key sizes were able to be much smaller. Public-key sizes are much larger than symmetric-key sizes, and public-key protocols tend to run more slowly. Key should be on the order of thousands of bits; `2048` is an absolute minimum, but it is recommended to go as close to `4096` or higher that your system can handle.

References

- *Cryptography Engineering* by Schneier, Ferguson, Kohno, Chapter 11