Cryptology Fall 2017

Chloe Martindale
TU/e

October 3, 2017

These notes are based on notes by Tanja Lange and Ruben Niederhagen. The last two weeks we have seen many theoretical constructions to both make and break cryptosystems. We will now go through an example of the 'best' way to set up a key exchange (at least given what we know so far), for example so that we can do AES.

We want to compute a shared secret between Alice and Bob using Diffie-Hellman, for which we first need to choose an construct a finite field. In practise of course this should be very large (we will see later in the course just how large), but for now we'll do a small example.

Example. Let's construct a finite field of 32 elements. $32{=}2^5$ is a prime power so such a field exists, and it will be a 5-dimensional vector space over the modular group $\mathbb{F}_2 := \mathbb{Z}/2\mathbb{Z}$. Recall that we can represent the finite field of 32 elements as

$$\mathbb{F}_{2^5} = \mathbb{F}_2[x]/f(x)\mathbb{F}_2[x],$$

where $f(x) \in \mathbb{F}_2[x]$ is a monic irreducible polynomial of degree 5. So, to construct our field, all we need to do is find a suitable polynomial $f(x)$. We claim that

$$f(x) = x^5 + x^4 + x^3 + x^2 + 1 \in \mathbb{F}_2[x]$$

is irreducible. $f(x)$ has no roots as $f(0) = f(1) = 1 \neq 0$, so $f(x)$ has no linear factors, but it could have factors of degree 2 or 3. So to check if $f(x)$ is irreducible we use Rabin's test. Now

$$\begin{aligned} x^{2^5} - x = x(x^5 + x^4 + x^3 + x^2 + 1)(x^{26} &+ x^{25} + x^{22} + x^{19} + x^{18} + x^{17} + x^{16} \\ &+ x^{15} + x^{13} + x^{12} + x^{11} + x^7 + x^5 \\ &+ x^3 + x^2 + 1), \end{aligned}$$

so $f(x)|(x^{2^5} - x)$ and hence (i) of the Rabin test in satisfied. As $n = 5$ is prime, for (ii) it suffices to show that $\gcd(f(x), x^2 - x) = 1$. As $x^2 - x = x(x-1)$ and neither 0 nor 1 are roots of $f(x)$ this holds. Hence $f(x)$ is irreducible in $\mathbb{F}_2[x]$.

Now that we have constructed a finite field $\mathbb{F}_{p^n}$, we should find an element $g \in \mathbb{F}_{p^n}^*$ such that the subgroup $\langle g \rangle$ of $\mathbb{F}_{p^n}^*$ generated by g is 'large' (i.e. $O(p^n)$).

Consider $\mathbb{F}_{32}$. The multiplicative group $\mathbb{F}_{32}^*$ has $32 - 1 = 31$ elements, so every element g of $\mathbb{F}_{32}^*$ has order dividing 31. Hence, any element except for 1 will generate the whole of $\mathbb{F}_{32}^*$, so take any element for g (e.g. $x \bmod f(x)$ or $x + 1 \bmod f(x)$ etc.).

Before we go ahead and use our newly constructed finite field for a key exchange, we should check that it's not susceptible to any of the attacks that we have seen so far. Recall from last week the 'Pohlig-Hellman' attack. This attack was devastating if the size of the multiplicative group was smooth - that is, if $\ell = |\langle g \rangle|$ had only small prime factors. In our example, $\ell = 31$, so the only prime factor is $O(p^n) = O(2^5)$, so we are safe against Pohlig-Hellman. (This example is too small for this to matter, but you get the idea.)

There is a further subletly in using Diffie-Hellman that we did not yet discuss. When studying the complexity of computing the shared secret key as an outsider, there is an important distinction between *computational* and *decisional* Diffie-Hellman:

Definition 1. Suppose that g, g^a, and g^b are known. The *computational Diffie-Hellman problem* or *CDH* is to compute g^{ab}.

Definition 2. Suppose that g, g^a, g^b, g^r, where r is a random integer, the *decisional Diffie-Hellman problem* or *DDH* is to decide whether $g^r = g^{ab}$.

At first glance the CDH and DDH look like the same problem, but if they are not, and for some particular group the DDH is much easier than the CDH, we would want to avoid this group. Let's try to do the DDH for a small example *without* computing discrete logs.

Example. Suppose that in the finite field $\mathbb{F}_{19}$, you're given that

$$(g, g^a, g^b, g^r) = (2, 7, 11, 13),$$

and you want to decide whether $13 = 2^{ab}$. What can we detect about a and b without computing the discrete log? Observe that the order of 2 is 18, so that for every $x \in \langle g \rangle$, we know that $x^9 = (x^{18})^{\frac{1}{2}} \pm 1$. In particular,

1. If a is even then there exists $a' \in \mathbb{Z}$ such that $a = 2a'$, so
$$(g^a)^9 = (g^{2a'})^9 = (g^{a'})^{18} = 1.$$
2. If a is odd, then $9a$ is odd (also mod 18), so there does not exist a' such that $g^{9a} = (g^b)^{18}$. Therefore $(g^a)^9 \neq 1$ and hence $(g^a)^9 = -1$.

So we can tell if a is even just by looking at the 9th power of g^a! We have that

$$(g^a)^9 = 7^9 \equiv 1 \bmod 19,$$

so a is even. In particular, this implies that ab is even, so let's check if r is even in the same way:

$$(g^r)^9 = 13^9 \equiv -1 \bmod 19,$$

so r is odd. Hence $r \neq ab$, and we have broken DDH!

The above example shows that it can be easier to break the DDH than the CDH. It is difficult to give a general statement such as 'fields of this form are easy to break with DDH', so the best approach is to choose a finite field and a generator g of a large subgroup and then look at the *distribution* of

$$(g, g^a, g^b, g^{ab}) \text{ vs. } (g, g^a, g^b, g^{\text{random}})$$

and see how they compare. If they're very different, we say that g^{ab} is *distinguishable*.

1 Baby Step Giant Step

One of the best generic algorithms known for solving the discrete logarithm problem is baby step giant step. Again, we assume that we have a subgroup $\langle g \rangle \subseteq \mathbb{F}_q^*$ of prime order ℓ, and that given $h \in \langle g \rangle$, we want to find $a = \log_g(h)$. The baby step giant step algorithm is as follows:

1. For i from 0 to $\lfloor \sqrt{\ell} \rfloor$, compute $b_i = g^i$.
2. For j from 0 to $\lfloor \sqrt{\ell} \rfloor + 1$, compute $c_j = h \cdot g^{-\lfloor \sqrt{\ell} \rfloor \cdot j}$.
3. Find i and j such that $b_i = c_j$.
4. Return $a = i + \lfloor \sqrt{\ell} \rfloor \cdot j$.

Note that then $g^i = h \cdot g^{-\lfloor \sqrt{\ell} \rfloor \cdot j}$, so that in particular $g^{i+\lfloor \sqrt{\ell} \rfloor \cdot j} = h$. Also, every a can be written as $a = a_0 + a_1 \lfloor \sqrt{\ell} \rfloor$ with $0 \leq a_0 \leq \lfloor \sqrt{\ell} \rfloor$ and $0 \leq a_1 \leq \lfloor \sqrt{\ell} \rfloor + 1$, so this algorithm will always return an answer. Baby step giant step takes $2\sqrt{\ell}$ multiplications and one inversion, which is already much better than $O(\ell)$!

Example. Let's use the baby step giant step algorithm to compute $\log_9(37)$ in $\mathbb{F}_{101}^*$. We have that $|\langle 9 \rangle| = 50$, so $\ell = 50$, hence $\lfloor \sqrt{\ell} \rfloor = 7$. We first do the 'baby step', that is, we compute $b_i = 9^i$ for $0 \leq i \leq \lfloor \sqrt{\ell} \rfloor = 7$:

i	0	1	2	3	4	5	6	7
3^i	1	9	81	22	97	65	80	13

Now we do the 'giant step', this is, we compute $c_j = 37 \cdot 9^{-7j}$ for $j \geq 0$ until we find a value matching the table above:

j	0	1
c_j	37	65

We see from the tables that $b_5 = c_1$, that is, that $9^5 \equiv 37 \cdot 3^{-10} \bmod 101$, hence $a = 12$.

Baby step giant step gives us a large speed-up, but it uses $O(\sqrt{\ell})$ storage, which is huge! Another alternative is to use *Pollard's rho method*.

2 Pollard's rho method

Again, we assume that we have a subgroup $\langle g \rangle \subseteq \mathbb{F}_q^*$ of prime order ℓ, and that given $h \in \langle g \rangle$, we want to find $a = \log_g(h)$. In Pollard's rho method, we look for $b, c, b', c' \in \{1, \ldots, \ell\}$ such that

$$g^b h^c = g^{b'} h^{c'}.$$

This implies that in $\mathbb{F}_q^*$,

$$g^{b-b'} = g^{a(c'-c)},$$

hence

$$b - b' \equiv a(c' - c) \bmod \ell.$$

We find such b, c, b', c' by doing a pseudo-random walk on a graph with vertices G_i defined by: $G_0 = g$, $b_0 = 1$, $c_0 = 0$, and

$$(G_{i+1}, b_{i+1}, c_{i+1}) = \begin{cases} (G_i \cdot g, b_i + 1, c_i) & \text{if } G_i = 0 \bmod 3 \\ (G_i \cdot h, b_i, c_i + 1) & \text{if } G_i = 1 \bmod 3 \\ (G_i^2, 2b_i, 2c_i) & \text{if } G_i = 2 \bmod 3, \end{cases}$$

and aborting when we find a loop. Observe that finding a loop means that we have found some $i \neq j$ such that $G_i = G_j$, which gives that

$$g^{b_i} h^{c_i} \equiv G_i \equiv G_j \equiv g^{b_j} h^{c_j} \bmod \ell.$$

This algorithm loops after approximately $\sqrt{\frac{\pi}{2}\ell}$ steps and uses $O(1)$ storage as we can combine it with a cycle detection algorithm such as Floyd's cycle detection algorithm, so is the most common algorithm in practise. Note that we will see Floyd's cycle detection algorithm in the lecture on 05/10/2017 but it is non-examinable!

Example. Let's compute $a := \log_3(7)$ in $\mathbb{F}_{17}^*$ using Pollard's rho method. The algorithm outputs a list

$$(3, 1, 0), (9, 2, 0), (10, 3, 0), (2, 3, 1), (4, 6, 2), (11, 6, 3), (2, 12, 6).$$

The 4th and the 7th items in the list have the same first entry (i.e. $G_4 = G_7$), so we have a loop. Hence

$$3^3 \cdot 7^1 \equiv 3^{12} \cdot 7^6 \equiv 2 \bmod 17,$$

and $a \equiv (12 - 3)/(1 - 6) \bmod 16$, so

$$a = 11.$$

All the attacks presented so far do not make any use of any special properties of the group $\mathbb{F}_q^*$ or $\langle g \rangle$. Such algorithms are called 'generic algorithms' as they work for any group. For groups without special properties these algorithms are the best (known) to attack the discrete logarithm problem. Next time we will see a better attack on the discrete logarithm problem for which the group is a subgroup of $\mathbb{F}_q^*$.