DISCRETE LOGARITHM (CONTINUED)

MATH 195

An example

Here is a baby example of a Diffie-Hellman exchange. Recall A picks an integer a (and keeps it secret), computes g^a, and sends g^a to B. Similarly, B picks an integer b (and keeps it secret), computes g^b, and sends g^b to A. A computes $h = (g^b)^a \in G$, B computes $h = (g^a)^b \in G$, and h is the common secret key. E given G, g, g^a, g^b hopefully cannot compute g^{ab}.

Take $G = \mathbb{F}_{32}^*$, $\mathbb{F}_{32} = \mathbb{F}_2[X]/(X^5 + X^2 + 1)$, $g = 00010 = X$ is a primitive root. Note that $\mathbb{F}_{32} \neq \mathbb{Z}/32\mathbb{Z}$!

A picks $a = 4$, $g^a = X^4 = 10000$; B picks $b = 5$, $g^b = X^5 = X^2 + 1 = 00101$. We have $h = (00101)^4 = ((00101)^2)^2 = (10001)^2$ by the freshperson's dream, and this is 100000001. We reduce this against $X^5 + X^2 + 1 = 100101$ and get the remainder $01100 = X^3 + X^2$:

$$\begin{array}{ccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & & & \\
 & & & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
 & & & & 0 & 1 & 1 & 0 & 0
\end{array}$$

In the same manner, we verify that $(10000)^5 = 01100$ as well.

ElGamal

Now we describe the cryptosystem of Taher ElGamal (1985) based on G, g.

We have $\mathcal{P} = G$, $\mathcal{C} = G \times G$, and let $\mathcal{R} = \{1, 2, \ldots, m-1\}$ be the space of random numbers. Prior to all communication, B picks an integer b (and keeps it secret) and makes g^b public. This is the public key.

Suppose A wants to send a message x to B, where we assume $x \in G$. A picks an integer a (and keeps it secret) and she sends g^a and $x \cdot (g^b)^a$. This is the encryption map:

$$\mathcal{K} \times \mathcal{P} \times \mathcal{R} \xrightarrow{E} \mathcal{C}$$
$$(k = g^b, x, a) \mapsto E_{k,a}(x) = (g^a, x(g^b)^a).$$

B recovers x by computing

$$x \cdot (g^b)^a((g^a)^b)^{-1} = x.$$

Decryption can be described as:

$$\mathcal{K} \times \mathcal{C} \xrightarrow{D} \mathcal{P}$$
$$(k = g^b, (f, y)) \mapsto D_k(f, y) = y \cdot (f^b)^{-1}$$

This is some of the material covered April 18–23, in Math 195: Cryptography, taught by Hendrik Lenstra, prepared by John Voight `jvoight@math.berkeley.edu`.

Example. Let $G = \mathbb{F}_{32}^*$ as above, $g = X = 00010$. Take $b = 5$, $k = g^b = 00101$. Let us send the message $x = 10101$, $a = 4$. A sends to B: $g^a = 100000$ together with

$$x(g^b)^a = (10101) \cdot (01101) = 00111.$$

B computes

$$(00111) \cdot (10000)^{-5}.$$

We cheat a little: $10000 = X^4$ so $(10000)^{-5} = X^{-20}$. We have $X^{31} = 1$ since $\#\mathbb{F}_{32}^* = 31$, so $X^{-20} = X^{11} = 100000000000$, which reduces to 00111, hence

$$(00111) \cdot (00111) = (00111)^2 = 10101$$

by the freshperson's dream, which is the correct message.

The problem faced by E: Knowing G, g, g^b, g^a, xg^{ab} but *not* b, a, she wants to compute x. We compare this to the previous problem faced by E (in Diffie-Hellman): knowing G, g, g^a, g^b, she wants to compute g^{ab}. It is clear that the ability to solve one of these problems is equivalent to the ability to solve the other (if you can compute x knowing xg^{ab}, you can divide to get g^{ab}, and if you can compute g^{ab} knowing xg^{ab}, you can divide to get x).

One possible improvement: send g^a once and for all.

Algorithms for Computing Discrete Logarithms

Recall that the discrete logarithm problem given a group G and an element $g \in G$ is the problem of finding $\log_g h$ upon input h. Recall $\log_g h = i$ if $h = g^i$ ($i \in \mathbb{Z}/m\mathbb{Z}$, $m = \operatorname{ord}(g)$), and $\log_g h$ is undefined if $h \notin \langle g \rangle$.

Method 1 (Complete enumeration). Compute $g^0 = 1$, $g^1 = g$, $g^2 = g \cdot g$, $g^3 = g \cdot g^2$, $\dots$, $g^{n+1} = g \cdot g^n$ until you encounter h. If you find $g^n = h$ then $n = \log_g h$. If before finding h you find $g^m = 1$, then $h \notin \langle g \rangle$. This algorithm is naive, and takes time $m/2 \sim m$ operations in G, which in this case are all multiplications.

This is fast for small m, and slow for large m.

Method 2 (Baby step-giant step). Pick a positive integer M with $M^2 \geq m = \operatorname{ord}(g)$, e.g. $\lceil \sqrt{m} \rceil$ (or the squareroot of any upper bound on the order of g or of the group will work).

Compute $h, h \cdot g, h \cdot g^2, \dots, h \cdot g^{M-1}$ (the baby steps) and $g^M, g^{2M}, \dots, g^{M^2}$ (the giant steps). Note that we step by 1 in g for the baby steps and by M in g for the giant steps. We check whether these two sequences have a member in common. If so, then $h \cdot g^i = g^{jM}$, so $h = g^{jM-i}$ and $\log_g h = jM - i$. If they do not, then $\log_g h$ doesn't exist.

Example. Let $G = \mathbb{F}_p^*$, $p = 29$, $g = 2$, $h = 3$. We take $M = \lceil \sqrt{29} \rceil = 6$. We compute the baby steps

$$3, 6, 12, 24 = -5, -10, -20 = 9$$

(notice that we double every time) and the giant steps

$$2^6 = 6, 6 \cdot 6 = 7, 13, \dots$$

but we see already that 6 is in common to both of these lists, so $2^6 = 3 \cdot 2$ in $\mathbb{F}_{29}$, so $3 = 2^5$ in $\mathbb{F}_{29}$.

Notice that we must compare two lists. If one does this naively (by comparing every two pairs of elements), then one requires time $O(M^2)$. However, by keeping the list sorted (using a quicksort algorithm or some such), then the time required is $O(M)$ (or perhaps slightly faster) with space $O(M)$.

Here is a proof that if $\log_g h$ exists, the algorithm will find it. Say $h = g^a$, $0 < a \leq m$. Then $a \leq M^2$, so the least multiple jM of M that is $\geq a$ is $\leq M^2$, so $0 < j \leq M$. Write $jM = a + i$. Then $i \geq 0$ and $i < M$ (otherwise $(j-1)M \geq a$, contradicting the choice of j). Hence

$$hg^i = g^a g^i = g^{a+i} = g^{jM}$$

so the sequences intersect, and the algorithm finds the logarithm.

If you want to find the *least* positive a with $h = g^a$, pick j *minimal* such that g^{jM} is in the first sequence and given j, pick i *maximal* such that $g^{jM} = hg^i$. Applying this method to $h = 1$, it will find the least *positive* a with $g^a = 1$, in other words, it will determine m.

If G is not required to be abelian, it may be wise to first test whether $hg = gh$. If $hg \neq gh$, then $\log_g h$ does not exist! (If it did, and $h = g^a$, then $hg = g^a g = g^{a+1} = gh$.) If $hg = gh$, then $\langle g, h\rangle$ is an *abelian* subgroup of G, so it is enough to have crytographic systems built upon abelian groups in some sense.

Method 3 (Pohlig-Hellman). The input: $G, g, m = \mathrm{ord}(g)$, $m = m_1 m_2 \dots m_t$, $m_i \in \mathbb{Z}_{>0}$ positive integers. It is important to know the order of g and a factorization of this order. The output is $\log_g h$, with time "dominated by" the quantity

$$\max\{\sqrt{m_1}, \sqrt{m_2}, \dots, \sqrt{m_t}\}.$$

(We are using the following fact from algebra: we have $\langle 1\rangle \subset \langle g^{m_t}\rangle \subset \langle g\rangle$, where now g^{m_t} has order $m' = m/m_t$, so we may compute the logarithm in a smaller group.)

Here is the algorithm, by steps:

(1) Compute $m' = m_1 m_2 \dots m_{t-1} = m/m_t$.
(2) Use the baby step-giant step method (or complete enumeration) to find $a = \log_{g^{m'}} h^{m'}$. If it doesn't exist, then $\log_g h$ doesn't exist either, and the algorithm stops. [Note: $h = g^a$, where a is taken modulo m, becomes $h^{m'} = (g^{m'})^a$, where now a is taken modulo m_t.]
(3) Compute hg^{-a}. If it is equal to 1, then we are done at this stage: $h = g^a$.
(4) Use the Pohlig-Hellman method with input

$$G,\ g^{m_t},\ m' = m_1 m_2 \dots m_{t-1},\ hg^{-a},$$

to compute $b = \log_{g^{m_t}}(hg^{-a})$ (in time essentially $\max\{\sqrt{m_1}, \dots, \sqrt{m_{t-1}}\}$). Output $\log_g h = m_t b + a$ if b exists, and if it does not, then the $\log_g h$ does not exist either.

The correctness of this method relies on the following claim:

Claim. We have as sets $\{x \in \langle g\rangle : x^{m'} = 1\} = \langle g^{m_t}\rangle$.

Proof. Suppose that $x = g^c$ with $x^{m'} = g^{cm'} = 1$. Then cm' is divisible by $m = m' m_t$, so

$$\frac{cm'}{m} = \frac{c}{m_t}$$

are both integers, and hence c is divisible by m_t. This proves one inclusion, and the other is clear: any element $g^{m_t d}$ is a power of g with $(g^{m_t d})^{m'} = g^{md} = 1$. □

Example. Given G, and $g \in G$, $m = \langle g \rangle = m_1 m_2 \dots m_t$, $t \geq 1$, and $h \in G$, we compute $\log_g h$ by reducing the problem to a computation involving $m' = m_1 \dots m_{t-1}$, $h^{m'}$, $g^{m'}$.

Take $G = \mathbb{F}_{101}^*$, with $g = 2 \in G$, $m = 100 = 10 \cdot 10$, $h = 3$. We have $m' = 10$, and we compute

$$h' = h^{m'} = 3^{10} = ((3^2)^2 \cdot 3)^2 = (-60)^2 = 3600 = -36.$$

We also compute $g' = g^{m'} = 2^{10} = 1024 = 14 = g^{m'}$. Note that since g has order 100 (it is a primitive root), $g^{m'}$ has order 10.

Now we compute $\log_{g^{m'}} h^{m'}$ using the baby step-giant method. We need $M^2 \geq m_t$, so $M = 4$. We compute $h', h'g', \dots, (h')(g')^{M-1}$, which is the sequence

$$-36, -36 \cdot 14 = 1, 1 \cdot 14 = 14, 14 \cdot 14 = -6.$$

Now we compare it to the sequence $(g')^M, (g')^{2M}, \dots, (g')^{M^2}$, and get

$$14^4 = 36, 36^2 = -17, -17 \cdot 36 = -6$$

so bingo (!): we see that $14^{12} = -36 \cdot 14^3$, so $-36 = 14^9$. In other words,

$$a = \log_{g'} h' = \log_{14}(-36) = 3M - 3 = 9.$$

Of course, since $-36 \cdot 14 = 1$, we know already $-36 = 14^{-1} = 14^9$, since 14 has order 10. In fact, it is easier to work with $a = -1$, since we only care about a modulo 10.

Now we compute $hg^{-a} = 3 \cdot 2^{-(-1)} = 6 \neq 1$. We compute $g^{m_t} = 2^{10} = 14$, and $m' = 10$, and $hg^{-a} = 6$. We compute $b = \log_{g^{m_t}}(hg^{-a}) = \log_{14} 6$. We can do this again using baby step-giant step: if we start computing, we get $6, 6 \cdot 14 = -17$, which already occurs in our second list (which is unchanged) as $36^2 = 14^8$, so $14^7 = 6$, or $b = 7$.

We then output $\log_g h = \log_2 3 = m_t b + a = 70 - 1 = 69$. Whew! We check our work:

$$2^4 = 16, 2^8 = 256 = -47, \dots, 2^6 4 = 1089 = -22$$

and $2(16)(-22) = 2(-352) = 2(-49) = -98 = 3$.

In fact, there are (much) better discrete logarithm algorithms that apply to $\mathbb{F}_p^*$ (p prime) (and other similar multiplicative groups). However, on groups coming from (general) elliptic curves nothing essentially better than baby step-giant step or Pollig-Hellman is known.

Conclusion: for a pair G, g to be secure for use in a discrete logarithm-based cryptosystem, it is desirable that the number $m = \text{ord}(g)$ has a large prime factor. There are three methods for construction G, g, m.

(1) The Mersenne-prime method: Pick p prime such that $2^p - 1$ is prime, pick $f \in \mathbb{F}_2[X]$ irreducible of degree p, and use $G = \mathbb{F}_{2^p}^*$, $\mathbb{F}_{2^p} = \mathbb{F}_2[X]/(f)$, $g = X$, $m = 2^p - 1$. (Can also use p with $2^p - 1$ prime up to a few small factors.) We have the following amazing fact:

Fact. If ℓ is a prime number, $2^\ell - 1$ also prime, and $X^\ell + X + 1 \in \mathbb{F}_2[X]$ irreducible, then $X^{2^\ell} + X + 1$ is irreducible in $\mathbb{F}_2[X]$.

Therefore with $\ell = 2$, $2^2 - 1 = 3$ is prime so $X^2 + X + 1$ is irreducible; now with $\ell = 3$, $2^3 - 1 = 7$ is prime, so $X^3 + X + 1$ is irreducible, continuing on with $2^7 = 127$ and $2^{127} - 1 = 170141183460469231731687303715884105727$ prime, we find that $X^{17...27} + X + 1$ is irreducible!

(2) The $kr + 1$ method: Pick a large prime r, pick a small k such that $kr + 1$ has no small prime factors (so, e.g. we insist $k \equiv 0 \pmod 2$, $k \not\equiv -r^{-1} \pmod 3$, and so on). Test whether $2^k \not\equiv 1 \pmod{kr+1}$, $2^{kr} \equiv 1 \pmod{kr+1}$. If not, try another k, and if yes, then take $p = kr + 1$, which is prime if $k \leq r$, and $G = \mathbb{F}_p^*$, $g = 2^k$, and $m = r$.

   Fact. If r is a prime number and $k \in \mathbb{Z}$, $0 < k \leq r$. Put $p = kr + 1$. Then p is prime if and only if there exists $a \in \mathbb{Z}$ such that $a^k \not\equiv 1 \pmod p$, and $a^{kr} \equiv 1 \pmod p$.

(3) Elliptic curves. To be discussed in the next lecture.