Advanced Algorithms	**Lecture 5/6 : 22 and 28/3, 1995**
Lecturer: Johan Håstad	*Scribe: Christer Liedholm*

5 Discrete logarithms in GF[2^n]

The finite field (which is also a group under multiplication disregarding the 0) GF[2^n] is common in computer applications mainly due to the ease by which it can be handled by a computer.
We are going to describe a fast algorithm for the computing of discrete logarithms in GF[2^n] but first let us look at the structure of GF[2^n].

GF[2] is the simplest field of all, consisting only of the two elements 0 and 1. Addition is as usual except that $1 + 1 = 0$ and multiplication is the normal integer multiplication.
To construct the finite field GF[2^n] we begin with GF[2] and adjoin an element θ which is a root of an irreducible polynomial p(x) of degree n and coefficients in GF[2]. Every element of GF[2^n] can then be represented as a polynomial in θ with coefficients in GF[2] and degree less than n. Addition is as usual, that is coefficient-wise modulo 2, but multiplication is performed by first multiplying the two polynomials the standard way and then reducing modulo p(x).

Example 1 n = 3: *We choose* x^3+x+1 *as our irreducible polynomial so that* θ *is a root of the equation* $\theta^3+\theta+1 = 0$.
The elements in this field are now $0, 1, \theta, 1+\theta, \theta^2, 1+\theta^2, \theta+\theta^2, 1+\theta+\theta^2$. *Note that the polynomial* x^3+x^2+1 *would have done equally well in constructing the field* GF[2^3] , *but we could not have used* x^3+1 *as it can be factored as* $(x+1)^3$.

If we, in the example above, had chosen x^3+x^2+1 as our irreducible polynomial of degree 3 and then adjoined the element φ, satisfying $\varphi^3+\varphi^2+1 = 0$ one can ask oneself if the field we obtain this way is fundamentally difficult from the field obtained by adjoining θ, satisfying $\theta^3+\theta+1 = 0$.
The answer is no; the two fields are indeed isomorphic by the isomorphism $\varphi \rightarrow 1+\theta$, as can be readily checked, and we in fact have the following

Theorem 1 *Up to isomorphisms there is one and only one field with* 2^n *elements and it can be represented by choosing a suitable irreducible polynomial of degree* n.

The task of computing discrete logarithms in GF[2^n] can now be formulated as the following: Given an irreducible polynomial *f* of degree n, two elements *g* and *y* in GF[2^n] find an integer *x* so that $g^x = y$.

By simply trying all possible integers (up to 2^n) this can of course be obtained in 2^n steps. A better method is the Baby/Giant step algorithm giving a cost of $2^{n/2}$ steps to compute x.

Still better is the factor base method described in the previous lecture. Applied to GF[2^n] we proceed as the following: Choose an element r in GF[2^n] at random, compute g^r and hope that g^r factors as polynomials of "small" degrees. This will give linear relations between the discrete logs of the "small" polynomials. Having gathered enough such relations one can by solving a linear system get the discrete logs for all small polynomials. Finally one chooses r at random, hoping that $y*g^r$ factors in polynomials of small degree and we are through.
This will give a running time of

$$O\left(2^{c\sqrt{n\log n}}\right)$$

That is, exactly the same as in the case of GF[p], with $p \sim 2^n$.

We are now going to present the best known algorithm for computing discrete logs in GF[2^n] with a running time (heuristically) of

$$O\left(2^{cn^{1/3}(\log n)^{2/3}}\right)$$

Thus, considerably better than the previous ones.

The algorithm was presented by Coppersmith (1982) and starts by choosing a "good" representation of GF[2^n]. It will be favourable with a polynomial, defining the representation of GF[2^n], f(x) of the form $x^n + f_1(x)$, where deg $f_1(x)$ is as small as possible. It is enough to compute discrete logarithms with an irreducible polynomial of this kind as the discrete logs in GF[2^n] represented by another irreducible polynomial can be efficiently computed from the former logs (homework).

Heuristically, we can find an f_1 of degree $\sim \log n$ because

1: The probability that $x^n + f_1(x)$ is irreducible, with random f_1, is about 1/n as there are approximately $2^n/n$ irreducible polynomials of degree n.
2: There are n polynomials f_1 of degree log n.

Supposing we have such a representation of GF[2^n] the next step is to construct a database of the discrete logarithms of all "small" polynomials. By "small" we mean a polynomial of degree at most $n^{1/3}$ (this is in fact a simplification,we really need to calculate the logs of polynomials up to degree $n^{1/3}(\log n)^{2/3}$, but for simplicity of presentation we will disregard factors (even in the exponent) which are powers of log n).
Choose k so that 2^k is approximately equal to $n^{1/3}$, then put h equal to $\left\lceil \frac{n}{2^k} \right\rceil$, that is the smallest integer greater than $n/2^k \sim n^{2/3}$.

Put $w(\theta) = w_1(\theta)\theta^h + w_2(\theta)$, **(1)**

where deg w_1 and deg w_2 is at most $n^{1/3}$, and θ a root of f(x).
The polynomial (= element of GF[2^n]) $w(\theta)$ defined this way has degree ~ $n^{2/3}$ (or less) and this is also true of the polynomial

$$w(\theta)^{2^k} = w_1\left(\theta^{2^k}\right)\theta^{h*2^k} + w_2\left(\theta^{2^k}\right) = w_1\left(\theta^{2^k}\right)\theta^{h*2^k-n} f_1(\theta) + w_2\left(\theta^{2^k}\right)$$

If we are lucky and both polynomials above factors in irreducible polynomials of degree at most $n^{1/3}$ we get linear relations between discrete logs of elements of the type $P(\theta)$, where the degree of the polynomial P is at most $n^{1/3}$.

$$g^r = w(\theta) = \prod_{\deg(p)<n^{1/3}} p(\theta)^{a_p}$$

$$g^{r*2^k} = w(\theta)^{2^k} = \prod_{\deg(p)<n^{1/3}} p(\theta)^{b_p}$$

$$\sum_{\deg(p)<n^{1/3}} \left(b_p - 2^k a_p\right)\log p(\theta) = 0$$

We continue to choose polynomials $w(\theta)$ of the above type until we have a system of linear equations giving the discrete log of every polynomial p of degree at most $n^{1/3}$. The system is homogeneous so that we are in fact only able to compute these logarithms up to some multiplicative constant (equal for all logs of course).
To determine the value of the constant we could choose g^r at random and hope that this polynomial will factor in polynomials of degrees at most $n^{1/3}$. But this is indeed overoptimistic: The probability of this is on the order of

$$2^{-cn^{2/3}\log n}$$

so continuing to test g^r:s until we succeed would take much too long time.

The probability above can easily be deduced from the following

Lemma 1: *The probability that a polynomial of degree* n_1 *factors in polynomials of degree at most* n_2 *is on the order of* $\left(\frac{n_1}{n_2}\right)^{-\frac{n_1}{n_2}}$

The best we can hope for is to factor a random g^r in polynomials of degree at most $n^{2/3}$. The probability that this occurs is (by the lemma) on the order of

$$2^{-cn^{1/3}\log n}$$

.

Suppose we have found a factorisation: $g^r = \prod_i u_i$ where deg(u_i) is at most $n^{2/3}$.

This reduces the problem to computing the discrete logarithm of a polynomial $u_i = u(\theta)$, where deg(u) is at most $n^{2/3}$.
We do this by choosing a polynomial w(x) of the special type **(1)**

$$w(\theta) = w_1(\theta) * \theta^h + w_2(\theta), \quad h = \left\lceil \frac{n}{2^{k_1}} \right\rceil \text{ where } k_1 \text{ is to be specified later}$$

But this time we try to make w divide u, and then we hope for $\frac{w}{u}$ and $w^{2^{k_1}}$, for some k_1, to factor in "small" factors.

The condition that w divides u can be formulated as a linear system in the coefficients of the two polynomials so those (arrays of coefficients of) w(x) for which w divides u forms a linear subspace and thus is easily obtained.

Suppose that deg(w_1) = deg(w_2) = t.

Then $w^{2^{k_1}}(\theta) = w_1\left(\theta^{2^{k_1}}\right) * \theta^{h*2^{k_1}} + w_2\left(\theta^{2^{k_1}}\right) = w_1\left(\theta^{2^{k_1}}\right) * f_1(\theta) * \theta^{h*2^{k_1}-n} + w_2\left(\theta^{2^{k_1}}\right)$

So that deg(w) = t + h, and deg ($w^{2^{k_1}}$) = $t * 2^{k_1}$.
We now choose k_1 so that the two degrees above are (approximately) equal. This occurs when $2^{k_1} \approx \sqrt{\frac{n}{t}}$, which means that both deg(w) and deg ($w^{2^{k_1}}$) ~ $t * 2^{k_1} \sim \sqrt{nt}$

Our hope is now for w/u and $w^{2^{k_1}}$ to factor in small irreducibles. That is, irreducible polynomials with degree (significantly) less than $n^{2/3}$. This will give a linear expression of log U in the logs of polynomials of a smaller degree and thus we have simplified things once more.

It turns out that a good choice for t (= the degrees of w_1 and w_2 above) is t ~ $n^{2/3} \Rightarrow 2^{k_1}$ ~ $n^{1/6}$, giving a degree of about $n^{5/6}$ for the polynomials w and $w^{2^{k_1}}$. According to Lemma 1 both w/u and $w^{2^{k_1}}$ factors in polynomials of degree at most $\sqrt{n}$ with an affordable probability, i.e. roughly $2^{-n^{1/3}(\log n)^c}$. Observe that the simultaneous probability that both w/u and $w^{2^{k_1}}$ factors in polynomials of the right size is on the order of the square root of the probability of just $w^{2^{k_1}}$ to factor this way and disregarding constant factors in the exponent this probability is of the appropriate size.

Supposing that both w/u and $w^{2^{k_1}}$ factors in polynomials of degree at most $\sqrt{n}$ we have in fact expressed the discrete logarithm of u linearly in the logs of polynomials of degree about $\sqrt{n}$. This is obvious if we take logarithms and solve for u. Summing up, we have reduced the problem of finding the discrete logarithm of a polynomial of degree $n^{2/3}$ to the easier problem of finding the logarithm of a polynomial of degree $\sqrt{n}$. More general; had

we started with the task of computing a discrete log of a polynomial of degree x, the method above transforms the problem into finding the log of a polynomial of degree $n^{-1/3}\sqrt{nx}$, which is less than x, supposing that x is greater than $n^{1/3}$.
Iterating this step we come up with easier and easier sub problems until we (after a limited number of steps) reach the degree of $n^{1/3}$ for which we have pre-calculated the logarithms and we are through.
To compute the number of steps necessary to reach a degree of $n^{1/3}$ we put $x = n^{1/3}x_1$, so that the variable x_1 tells us how far from the favourable degree of $n^{1/3}$ we are. After one iteration the degree has lowered to $n^{1/3}\sqrt{x_1}$. After one more we have a degree of $n^{1/3}x_1^{1/4}$ and so on. Starting with a degree of $n^{2/3}$, that is $x_1 = n^{1/3}$, we have after k iterations $x_1^{(k)} = \left(n^{1/3}\right)^{2^{-k}} = 2^{c2^{-k}\log n} = 2^c$ if we choose $k = \log\log n$, meaning that we have now reduced the problem to computing the discrete log of polynomials of degree $c_1 n^{1/3}$.
Thus we have expressed the log of g^r linearly in the logs of polynomials of degree $c_1 n^{1/3}$ whose logs we have already computed (we actually only computed the logs of polynomials of degree $n^{1/3}$, but the constant is immaterial) up to an additive constant so this new relation is enough to determine the value of the constant.

To determine the amount of work needed for such a calculation we observe that we are dealing with a tree of depth log log n. Each node branches into at most n nodes on the level below so the total number of nodes is bounded by $n^{\log\log n} = 2^{(\log n)\log\log n} < 2^{\varepsilon n^{1/3}}$ and the amount of work in each node is bounded by $2^{cn^{1/3}(\log n)^{2/3}}$ (which is a bound of the work in the node at the highest - most work intensive - level, remember that the value $n^{1/3}$ above really has to be increased to $n^{1/3}(\log n)^{2/3}$). This means that the total cost for the calculation of the additive constant above is bounded by $2^{c_1 n^{1/3}(\log n)^{2/3}}$ which is what we aimed for.

The cost of the calculation of the logs of all small polynomials is bounded by the same limit, this is easily established by invoking lemma 1 and the fact that we need to generate about $2^{cn^{1/3}(\log n)^{2/3}}$ linear relations between the logs of the polynomials of small degree (less than $2^{n^{1/3}(\log n)^{2/3}}$) to be able to solve the system.

Summing up we can divide the algorithm into three steps:

1. Build a database consisting of the discrete logs (up to an additive constant) of all small polynomials.

2. Express the discrete log of g^r (i.e. r) in the logs of small degree polynomials and use this to determine the constant above.

3. Pick g^r at random until hg^r (where h(θ) is the element of GF[2^n] whose logarithm we have to compute) factors nicely and proceed as above