Solving the discrete logarithms in quasi-polynomial time

Ivo Kubjas

1 Introduction

1.1 Discrete logarithm

The discrete logarithm problem is analogous to the problem of finding logarithm in finite groups.

Definition 1 (Discrete logarithm)**.** Discrete logarithm of h to the base g is x such that

$$g^x = h.$$

We denote the discrete logarithm as $\log_g(h) = x$.

The discrete logarithm has similar algebraic properties as for logarithm in reals. Namely, $\log_g(hh') = \log_g(h) + \log_g(h')$.

The discrete logarithm problem is also randomly self-reducible. This means that there are no "difficult" problems. If there would be an element h in the group for which the logarithm is hard to solve, then it is possible to take $1 \leq x^\star \leq q$, where q is the order of the group. Now, instead of finding the logarithm of h, we find the logarithm of $hg^{x^\star}$ and subtract $x^\star$ from the result. We know that if $x^\star$ is chosen uniformly, then also $g^{x^\star}$ is uniformly distributed in the group and $hg^{x^\star}$ is uniformly distributed. Thus the difficult problem is reduced to solving the discrete logarithm for an average element in the group.

Random self-reducibility allows for "easy" problems in the same group.

1.2 The difficulty of computing discrete logarithm

Naturally, a question of why computing discrete logarithms is more difficult than computing logarithms arises. For example, there exists a simple algorithm to compute logarithms in reals.

Algorithm 1 (Finding logarithm)**.** Let the base be b. We find the logarithm to the base 2 and convert the logarithm to required base in the final step.

Input is a real $y = 2^x$ for which the logarithm x is being computed.

As the first step, the algorithm finds x' such that $2^{x'} < y < 2^{x'+1}$. Then y is divided by $2^{x'}$.

In the second step, an approximation using Taylor series is used. The Taylor series for logarithm is

$$\log(1+z) = \sum_{k=1} \frac{(-1)^{k+1} z^k}{k},$$

for $|z| < 1$. Because of the initial condition for x', we have that $y/2^{x'} - 1 < 1$. We use the Taylor series to approximately find the logarithm x''. The logarithm is then $x \approx x' + x''$.

As the final step, we convert the logarithm to required base using tables and logarithm properties.

In general, the methods for finding logarithms are constructed similarly as the first step consists of initial approximation using probing and then more specific approximation is used. As the constructions require approximate solving and convergence of power series, then this approach can not be used for finding exact discrete logarithms in cyclic groups.

2 Methods for solving the discrete logarithm problem

As we have just seen, inherently different methods should be used for finding discrete logarithms. There are conventional methods which reduce the discrete logarithm problem in a group to a same problem in subgroups and methods which use the birthday paradox to find collusions in different subsets of the group to reduce the computation effort.

2.1 Pohlig-Hellman algorithm

The Pohlig-Hellman algorithm allows us to reduce the solving of the discrete logarithm in a group of order $n = \prod p_i^{e_i}$ to solving the discrete logarithm in groups of order $p_i^{e_i}$. Thus, the difficulty of solving the discrete logarithm is as hard as solving the discrete logarithm in the subgroup of largest prime order.

Algorithm 2 (Pohlig-Hellman algorithm [PH78])**.** Let the order of the group G be $n = \prod_{i \in [P]} p_i^{e_i}$, where $p_i | n$ and p_i are primes for $i \in [P]$. Let the generator be g. Let $h = g^x$ be the element whose discrete logarithm x is being searched.

Let $[P]$ be the set $[P] := \{1, 2, \ldots, P\}$. For every $i \in [P]$, define

$$n_i = \frac{n}{p_i^{e_i}}, \quad g_i = g^{n_i}, \quad h_i = h^{n_i}.$$

As $g_i^{p_i^{e_i}} = g^n$ and the order of g is n, then the order of g_i is $p_i^{e_i}$. Also, $g_i^x = g^{n_i x} = h^{n_i} = h_i$.

The discrete logarithm is solved for h_i and g_i. Let the discrete logarithms be x_i. As the order of each g_i is $p_i^{e_i}$, then $x \equiv x_i \mod p_i^{e_i}$.

Using the Chinese remainder theorem, if all x_i are known, then it is possible to find x such that $x \equiv x_i \mod p_i^{e_i}$ for $i \in [P]$.

The computational complexity of the algorithm is bounded by the complexity of solving the discrete logarithm for the subgroup of G of the largest order. Finding x from x_i requires $O(P)$ operations if using the Chinese remainder theorem.

2.2 Baby step/giant step algorithm

The baby step/giant step algorithm solves the discrete logarithm problem in $O(\sqrt{n})$ steps using $O(\sqrt{n})$ of memory.

The idea of the algorithm is to compute two lists of values using different bases for exponentiation. Those lists are compared to find a collision and thus the solution to the discrete logarithm problem.

Algorithm 3 (Shank's baby step/giant step algorithm [Sha71])**.** Let G be a group of order n. Let $m = \lceil \sqrt{n} \rceil$. Let $h = g^x$ be the element whose discrete logarithm x is being searched.

The baby step is to create a set $A := \{(hg^{-r}, r) | 0 \leq r \leq m\}$. If the set A includes a pair $(1, r')$, then $hg^{-r'} = 1$ and thus $x = r'$.

Otherwise, the giant step is to create a set $B := \{((g^m)^q, q) | 0 \leq q \leq m\}$. Sets are compared to find a collision in the first element of the pairs.

If a collision $hg^{-r} = (g^m)^q$ is found then $h = g^{mq+r}$ and thus $x = mq + r$.

We see that the computation complexity is dominated by creating the two sets, both requiring $m = \lceil \sqrt{n} \rceil$ exponentiations. The memory cost is for storing A as each item in B can be tested if it is included in A and then discarded.

2.3 Pollard's ρ algorithm

The Pollard's ρ algorithm is an improvement over Shank's algorithm as it requires only constant amount of memory.

Algorithm 4 (Pollard's ρ algorithm)**.** The group G is divided into three subsets of (roughly) equal sizes A_1, A_2 and A_3.

The mapping f is defined as

$$f(b) = \begin{cases} gb & \text{if } b \in A_1 \\ b^2 & \text{if } b \in A_2 \\ hb & \text{if } b \in A_3 \end{cases}.$$

The first step is to take a random $x_0 \in [n]$ and take $b_0 = g^{x_0}$. Then, every b_i is calculated as $b_{i+1} = f(b_i)$.

The value b_i can also be written as $b_i = g^{x_i} h^{y_i}$ for

$$x_{i+1} = \begin{cases} x_i + 1 & \text{if } b_i \in A_1 \\ 2x_i & \text{if } b_i \in A_2 \\ x_i & \text{if } b_i \in A_3 \end{cases} \mod n, \quad y_{i+1} = \begin{cases} y_i & \text{if } b_i \in A_1 \\ 2y_i & \text{if } b_i \in A_2 \\ y_i + 1 & \text{if } b_i \in A_3 \end{cases} \mod n.$$

If there is a collision $b_i = b_{i+k}$, then $g^{x_i} h^{y_i} = g^{x_{i+k}} h^{y_{i+k}}$ and thus

$$\log(h) = \frac{x_i - x_{i+k}}{y_{i+k} - y_i}.$$

2.4 Index calculus

Combining the conventional methods, the complexity of finding the discrete logarithm was reduced to $O(\sqrt{p})$. The index calculus method allows even greater decrease in the complexity.

In general, the index calculus method consists of three phases.

1. The sieving phase

   During the sieving phase, relations between the elements of G are constructed. The elements are chosen such that the discrete logarithms are easier to find. If the elements are $\{g_i | i \in I\}$, then the relations can be described as $\prod_{i \in I} g_i^{m_i} = \prod_{i \in I} g_i^{n_i}$, or

   $$\sum_{i \in I} m_i \log g_i = \sum_{i \in I} n_i \log g_i. \tag{1}$$

2. The linear algebra phase

   During the linear algebra phase, the relations are solved for m_i and n_i thus finding the logarithms of given elements.

3. Individual logarithm phase

   To solve the discrete logarithm for arbitrary element y in G, one must find the relation between that element and some other elements. This step is iterated until y is represented as a linear combination of elements in $\{g_i | i \in I\}$

3 L notation

To define the computational complexity of the algorithms which solve the discrete logarithm problem, a L-notation is typically used. The value $L_q(\alpha, c)$ is calculated as

$$L_q(\alpha, c) := \exp((c + o(1))(\log q)^{\alpha}(\log \log q)^{1-\alpha}).$$

The L-notation is tightly related to the choice of the subset $\{g_i | i \in I\}$ in the sieving phase of index calculus. These elements are chosen according to their *smoothness*.

Definition 2. An integer is y-smooth if all its prime factors are lower than y.

Definition 3. A polynomial over a finite field is m-smooth if all its irreducible factors have degree lower than m.

It is possible to estimate the probability of choosing y-smooth element from the group.

Estimate 1 ([CEP83, PGF98])**.** *The probability for an arbitrary integer lower than x to be y-smooth (and arbitrary polynomial with degree less than n to be m-smooth) is*

$$u^{-u+o(1)},$$

with $u = \frac{\log x}{\log y}$ (for polynomials $u = \frac{n}{m}$).

The complexity of the algorithms finding the discrete logarithms can be related to the probability of the smoothness of the elements.

If $\alpha \to 1$, then the equation becomes $L_q(\alpha, c) = \exp((c + o(1)) \log q)$ and the complexity is exponential in $\log q$. On the other hand, if $\alpha \to 0$, then we have $L_q(\alpha, c) = \exp((c + o(1)) \log \log q) \approx (\log q)^{c+o(1)}$. Thus by decreasing the α, the complexity becomes polynomial in $\log q$ and if α increases, the complexity becomes exponential in $\log q$. As $\log q$ denotes the number of bits required for representing elements of a given group, then $L_q(\alpha, c)$ is a natural measure for the difficulty of solving a discrete logarithm in a given group.

Thus, the argument α denotes the asymptotic complexity of the algorithms. If the exact value is not important in the context, then also the notation $L(\alpha)$, where c is a constant, is used. The additional argument c provides the exact value and thus allows comparing the complexity of different algorithms.

4 Quasi-polynomial algorithm for solving discrete logarithms

In [JOP], Barbulescu et al proposed a new method for solving the discrete logarithm using index calculus. Their method relies on the properties of restricted finite fields.

Definition 4. A finite field K admits a sparse medium subfield representation if

- it has a subfield of q^2 elements for a prime power q, i.e. K is isomorphic to $\mathbb{F}_{q^{2k}}$ with $k \geq 1$;
- there exists two polynomials h_o and h_1 over $\mathbb{F}_{q^{2k}}$ of small degree, such that $h_1 X^q - h_0$ has a degree k irreducible factor.

It is now possible to propose a tool for solving the discrete logarithm problem. As the result depends on several heuristics, then the proof for the proposition is not exact.

Proposition 1 (Proposition 2, [BGJT14])**.** *Let $K = \mathbb{F}_{q^{2k}}$ be a finite field that admits a sparse medium subfield representation. Under the heuristics explained below, there exists an algorithm whose complexity is polynomial in q and k and which can be used for the following two tasks.*

1. *Given an element of K represented by a polynomial $P \in \mathbb{F}_{q^2}[X]$ with $2 \leq \deg P \leq k - 1$, the algorithm returns an expression of $\log P(X)$ as a linear combination of at most $O(kq^2)$ logarithms $\log P_i(X)$ with $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$ and of $\log h_1(X)$.*
2. *The algorithm returns the logarithm of $h_1(X)$ and the logarithms of all the elements of K of the form $X + a$, for a in $\mathbb{F}_{q^2}$.*

The proposition is used for finding discrete logarithm of $P(X) \in K$ in a straightforward way. As K is isomorphic to $\mathbb{F}_{q^2 k}$ and we take the coefficients of $P(X)$ in $\mathbb{F}_{q^2}$, then $\deg P(X) \leq \mathbb{F}_{q^k}$. It is possible to apply the first result of Proposition 1, obtaining

$$\log P(X) = e_0^1 \log h_1(X) + \sum_i e_i^1 P_i^1(X),$$

with $\deg P_i^1(X) \leq \lceil (k-1)/2 \rceil$.

For every P_i^1, we repeatedly apply the Proposition 1 until we obtain a linear combination

$$\log P(X) = e_0^{\kappa(\log k)} \log h_1(X) + \sum_i e_i^{\kappa(\log k)} P_i^{\kappa(\log k)}(X),$$

with $\deg P_i^{\kappa(\log k)}(X) = 1$. Now, it is possible to apply the second part of the proposition to find the discrete logarithms of $h_1(X)$ and $P_i^{\kappa(\log k)}(X)$. Using the linear combination, it is now possible to deduce $\log P(X)$.

As during each descent every discrete logarithm of an element is represented as a linear combination of $O(kq^2)$ discrete logarithms of elements of smaller degree, then the discrete logarithms at the end of the loop is $O(kq^2)^{\kappa(\log k)}$, it is necessary to solve the same number of discrete logarithm instances.

Alternatively, we can state the discussion as a result.

Theorem 1 (Theorem 3, [BGJT14])**.** *Let $K = \mathbb{F}_{q^{2k}}$ be a finite field that admits a sparse medium subfield representation. Assuming the same heuristics as in Proposition 1, any discrete logarithm in K can be computed in a time bounded by* $\max(q,k)^{O(\log k)}$.

Before we go to the proof of Proposition 1, we remind some definitions used in the proof.

Definition 5. Given a polynomial $P(X)$ over a field $\mathbb{F}$, it is said to be an irreducible polynomial if it can not be written as a product of two smaller degree polynomials $P_1(X)$ and $P_2(X)$ such that $P(X) = P_1(X)P_2(X)$.

The irreducible polynomials are somewhat similar to prime numbers as they can not be factored.

Definition 6. A coset of a subgroup H of a group G in respect to g is a set

$$gH = \{gh : h \in H\}.$$

The set of all cosets of all subgroups H of G form a quotient group G/H.

If we consider the set of matrices M over $\mathbb{F}_q$, then we consider cosets where two matrices m_1 and m_2 belong to the same coset if there exists $a \in \mathbb{F}_q$ such that $m_1 = am_2$.

Definition 7. A general linear group $GL_n(\mathbb{F})$ is a group of all $n \times n$ invertible matrices over $\mathbb{F}$.

Definition 8. A n-dimensional scalar matrix is a $n \times n$ diagonal matrix. The set of n-dimensional scalar matrices over $\mathbb{F}$ is denoted as $Z_n(\mathbb{F})$.

Definition 9. A projective linear group $PGL_n(\mathbb{F})$ is a quotient group

$$PGL_n(\mathbb{F}) = GL_n(\mathbb{F})/Z_n(\mathbb{F}).$$

Proof of Proposition 1. Let the degree of $P(X)$ be D such that $1 \leq D \leq k-1$.

The proof of the proposition follows the steps of index calculus.

The sieving phase

The systematic equation is defined as:

$$X^q - X = \prod_{a \in \mathbb{F}_q} (X - a). \tag{2}$$

The projective line $\mathbb{P}^1(\mathbb{F}_q)$ is defined as a set of equivalence classes in $\mathbb{F}_q^2$ with the two elements $x, y \in \mathbb{F}_q^2$ being equivalent if $y = ax$ for some $a \in \mathbb{F}$. The points in $\mathbb{P}^1(\mathbb{F}_q)$ are denoted as $(\alpha : \beta)$ for $\alpha \neq 0$ and $\beta \neq 0$. The projective line also includes the point at infinity: $(1 : 0)$. Thus there are $q + 1$ points in the projective line $\mathbb{P}^1(\mathbb{F}_q)$.

As the points are equivalence classes, it is possible to choose a representatives $\mathcal{S} = \{(\alpha, \beta)\}$ such that

$$X^q Y - XY^q = \prod_{(\alpha,\beta) \in S} (\beta X - \alpha Y). \tag{3}$$

A homographic mapping is a relation

$$m \cdot P = \frac{aP + b}{cP + d},$$

where $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $ad - bc \neq 0$.

We are interested in all 2×2-dimensional invertible matrices over $\mathbb{F}_{q^2}$. We also consider all different matrices which are multiples of other matrix as a single matrix. These matrices form a set $\mathcal{P}_q$.

Alternatively, given projective linear groups $PGL(\mathbb{F}_{q^2})$ and $PGL(\mathbb{F}_q)$, it can be shown that $\mathcal{P}_q = PGL(\mathbb{F}_{q^2})/PGL(\mathbb{F}_q)$.

For every matrix $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathcal{P}_q$, we replace X with $(aP+b)$ and Y with $(cP+D)$ in Equation (3), obtaining

$$\begin{aligned}
(aP + b)^q (cP + d) - (aP + b)(cP + d)^q &= \prod_{(\alpha,\beta) \in \mathcal{S}} \beta(aP + b) - \alpha(cP + d) && (4)\\
&= \prod_{(\alpha,\beta) \in \mathcal{S}} (-c\alpha + a\beta)P - (d\alpha - b\beta) && (5)\\
&= \lambda \prod_{(\alpha,\beta) \in \mathcal{S}} P - x(m^{-1} \cdot (\alpha : \beta)). && (6)
\end{aligned}$$

Here $P - x(m^{-1} \cdot (\alpha : \beta))$ denotes $P - u$ when $m^{-1} \cdot (\alpha : \beta) = (u : 1)$, thus $u = \frac{d\alpha - b\beta}{-c\alpha + a\beta}$, or 1 if $m^{-1} \cdot (\alpha : \beta) = \infty$.

If 1 is a multiplier in the product, then there are q multiples and $q + 1$ otherwise.

Now, we replace the left-hand side of the equation. Because of Definition 4, we have $X^q \equiv \frac{h_0(X)}{h_1(X)}$. We also denote a^q as $\tilde{a}$ if $a \in \mathbb{F}_{q^2}$. We denote $\tilde{P}(X)$ if we raise every coefficient in $P(X)$ to the q-th power. Then the left-hand side is

$$\begin{aligned}
\mathcal{L}_m :=& (\tilde{a}\tilde{P}(X^q) + \tilde{b})(cP(X) + d) - (aP(X) + b)(\tilde{c}\tilde{P}(X^q) + \tilde{d}) = && (7)\\
&\left(\tilde{a}\tilde{P}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{b}\right)(cP(X) + d) - (aP(X) + b)\left(\tilde{c}\tilde{P}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{d}\right) && (8)
\end{aligned}$$

We see that the power of the denominator is the same as the power of h_1. The numerator has a degree at most $(1 + \max(\deg h_0, h_1))D$. If the numerator of $\mathcal{L}_m$ is $\lceil D/2 \rceil$-smooth, then $m \in \mathcal{P}_q$ yields a relation.

We now construct a row vector $v(m)$ for m. Each coordinate μ of $v(m)$ is 1 if $P - x(\mu)$ appears in the right hand side of the Equation (4) and 0 otherwise. The length of $v(m)$ is $q^2 + 1$.

The matrix $H(P)$ is constructed from rows $v(m)$ where m yields a relation. The m is taken from every coset of $\mathcal{P}_q$. We need the following heuristic, which has been studied in [BGJT14].

Heuristic 1. *For any $P(X)$, the set of rows $v(m)$ for cosets $m \in \mathcal{P}_q$ that yield a relation form a matrix which has full rank $q^2 + 1$.*

In the supporting argument of this heuristic, it is shown that this matrix has $\Theta(q^3)$ rows. Thus it is required that q is large enough that the number of rows is at least than $q^2 + 1$.

The linear algebra phase

Using linear algebra, it is possible to write $P(X)$, as a vector $(\ldots, 0, 1, 0, \ldots)$ where 1 corresponds to $P(X)$ as a linear combination of rows of $H(P)$. Thus we also write $\log P$ as a linear combination of $\log P_i$, where P_i are the values on the left-hand sides of the Equation (4). As there are $q^2 + 1$ columns, then the combination requires up to $q^2 + 1$ rows. As each row involves at most $\deg \mathcal{L}_m \leq (1 + \max(h_0, h_1))D$ polynomials, then the polynomial P is expressed as a linear combination of $O(Dq^2) = O(kq^2)$ polynomials of degree less than $\lceil D/2 \rceil$ and h_1.

Individual logarithm phase

This phase is done similarly to the rest of the proof except that instead of polynomials P, a degree one polynomial X is used. In this case, both sides of Equation (4) involve only linear polynomials. A linear system is obtained with unknowns $\log(X + a)$ with $a \in \mathbb{F}_{q^2}$. The solution of the system relies on the following heuristic.

Heuristic 2. *The linear system constructed from all the Equations (4) for $P(X) = X$ has full rank.*

Time complexity

It is known that the order of $PGL(\mathbb{F}_{q^i})$ is $q^{3i} - q^i$, so the set of cosets $\mathcal{P}_q$ has $q^3 + q$ elements. It takes polynomial time in q and $\deg P = O(k)$ to check wheter $m \in \mathcal{P}_q$ yields a relation from Equation (4). The linear algebra step can be done in $O(q^{2\omega})$ time using fast matrix multiplication algorithms. Thus, the algorithm runs in polynomial time in q and K. □

The heuristical claims in the proof have supporting arguments in [BGJT14].

5 Complexity of the algorithm in different fields

We consider some specific choices of the parameters and see, how the result of Theorem 1 applies to these choices.

Corollary 1. *For finite fields of cardinality $Q = q^{2k}$ with $q + O(1) \geq k$ and $q = (\log Q)^{O(1)}$, there exists a heuristic algorithm for computing discrete logarithm in quasi-polynomial time*

$$2^{O((\log\log Q)^2)}.$$

Proof. We use the Proposition 1. From Theorem 1, the complexity of the traversal is $q^{\log q}$. Since $Q \approx q^{2q}$, we have $q = (\log Q)^{O(1)}$. Thus we have $2^{O\left((\log\log Q)^2\right)}$. □

One case, where this corollary applies, is when $Q = p^n$ with $p \geq (\log Q)^{O(1)}$ and $n \geq (p + \delta)$.

Another case is when n is composite and includes a factor so that $\mathbb{F}_Q$ includes a subfield $\mathbb{F}_{p^m}$.

Corollary 2. *For finite fields of cardinality Q and characteristic bounded by $(\log Q)^{O(1)}$, there exists a heuristic algorithm for computing discrete logarithms in quasi-polynomial time*

$$2^{O((\log\log Q)^2)}.$$

Proof. The idea is to embed $\mathbb{F}_Q$ in some other larger field which corresponds to the case in previous corollary.

Let $n = \log Q / \log p$, thus $Q = p^n$. If n is odd, then $k = n$ and $k = n/2$ otherwise. We take $q = p^{\lceil \log_p k \rceil}$ and apply the proposition to the field $\mathbb{F}_{q^{2k}}$. □

Using this corollary, it is possible to compute discrete logarithms in $\mathbb{F}_{2^n}$ in time $2^{O((\log n)^2)}$.

Corollary 3. *For finite fields of the form $\mathbb{F}_Q = \mathbb{F}_{q^{2k}}$ where q is bounded by $L_Q(\alpha)$, there exists a heuristic algorithm for computing discrete logarithms in subexponential time*

$$L_q(\alpha)^{\log\log Q}.$$

Proof. If $q = L_Q(\alpha)$, then $\log q = O((\log Q)^\alpha (\log\log Q)^{1-\alpha})$ and thus $k = \log Q / \log q = O((\log Q / \log\log Q)^{1-\alpha})$. So, $k \leq q + \delta$. Using Theorem 1, we have that the complexity is $q^{O(\log k)}$. By substitution, we get the desired result. □

6 Conclusion

In this summary, we reviewed conventional methods for solving the discrete logarithm problem. As these methods have a natural bound on the time complexity, then more efficient methods need to be used for decreasing the time required for solving a discrete logarithm problem. The well-known index calculus method has allowed for reducing the time complexity down to quasi-polynomial. It is not known, if it would be possible to have a better asymptotic difficulty as the current descent rate could only be overcome by a fixed descent rate. This would however imply that the discrete logarithm problem is not that difficult as previously thought. Considering that the problem has been very well studied, then the possibility of new breakthroughs is diminishing.

The discrete logarithm problem is fundamental in cryptography and having efficient algorithms for solving the instances would greatly impact the security of cryptosystems.

Fortunately, the index calculus method does not apply well to the elliptic curve groups as there are no efficient means for finding the smoothness basis of these groups. However, fields over integers are still used for DSA signature schemes. The new results imply that the parameters need to be considered carefully to not damage the security properties of the signatures.

References

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Advances in Cryptology–EUROCRYPT 2014*, pages 1–16. Springer, 2014.

[CEP83] E Rodney Canfield, Paul Erdös, and Carl Pomerance. On a problem of Oppenheim concerning "factorisatio numerorum". *Journal of Number Theory*, 17(1):1–28, 1983.

[JOP] Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. The past, evolving present and future of discrete logarithm.

[PGF98] Daniel Panario, Xavier Gourdon, and Philippe Flajolet. An analytic approach to smooth polynomials over finite fields. In *Algorithmic number theory*, pages 226–236. Springer, 1998.

[PH78] Stephen C Pohlig and Martin E Hellman. An improved algorithm for computing logarithms over and its cryptographic significance (corresp.). *Information Theory, IEEE Transactions on*, 24(1):106–110, 1978.

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Pure Math*, volume 20, pages 415–440, 1971.