ELSEVIER

Information Processing Letters 79 (2001) 93–98

Information Processing Letters

www.elsevier.com/locate/ipl

Small generic hardcore subsets for the discrete logarithm: Short secret DL-keys

C.P. Schnorr *

Fachbereich Mathematik/Informatik, Universität Frankfurt, 60439 Frankfurt, Germany

Received 14 March 1999; received in revised form 12 May 2000

Communicated by S. Zaks

Abstract

Let G be a group of prime order q with generator g. We study hardcore subsets $H \subset G$ of the discrete logarithm (DL) $\log_g$ in the model of generic algorithms. In this model we count group operations such as multiplication and division, while computations with non-group data are for free. It is known from Nechaev [Math. Notes 55 (1994) 165] and Shoup [Lecture Notes in Comp. Sci., Vol. 1233, Springer, Berlin, 1997, p. 256] that generic DL-algorithms for the entire group G must perform $\sqrt{2q}$ generic steps. We show that DL-algorithms for small subsets $H \subset G$ require $\frac{1}{2}m + o(m)$ generic steps for almost all H of size $\#H = m$ with $m \leqslant \sqrt{q}$. Conversely, $\frac{1}{2}m + 1$ generic steps are sufficient for all $H \subset G$ of even size m. Our main result justifies to generate secret DL-keys from seeds that are only $\frac{1}{2}\log_2 q$ bits long.

Keywords: Computational complexity; Cryptography; Discrete logarithm (DL); Generic algorithms; Generic complexity; Hardcore subsets

1. Introduction

Many cryptographic schemes for digital signatures, encryption and key exchange rely on the hardness of the discrete logarithm (DL) problem [1–3,7,8]. The security of these schemes requires that the problem to compute the discrete logarithm of random group elements is hard. For security, private–public key pairs, ciphertexts and signatures must represent random instances of the DL-problem. As the computational costs of the DL-cryptosystems increase with the size of the group it raises the question whether the entire group must be used. We show that the DL-problem restricted to small random subsets H of the group G has nearly the same generic complexity as for the entire group. This suggests that DL-cryptosystems can be optimized by using small random subsets of the group. An example of such an optimization is to generate the secret key of a DL-cryptosystem from random seeds that are only $\frac{1}{2}\log_2 q$ bits long. The $\frac{1}{2}\log_2 q$ threshold is tight, its proof requires the generic model.

Let us mention some recent security results in the generic model which give reasonable evidence that various practical cryptosystems are secure. Shoup [9] proves security of the Schnorr identification scheme against active attacks. He also proves lower bounds for the Diffie–Hellman problem and the decisional Diffie–Hellman problem. The intractability of the latter problems is assumed in the security proofs of [1]. Schnorr [10] proves that almost all discrete log bits

* This work was initiated in 1998 during a stay at Bell Laboratories, Murray Hill, New Jersey. The support of Bell Laboratories is gratefully acknowledged.

E-mail address: schnorr@cs.uni-frankfurt.de (C. Schnorr).

0020-0190/01/$ – see front matter
PII: S0020-0190(00)00173-3

are simultaneously secure. Schnorr and Jakobsson [11] show that signed ElGamal encryption is non-malleable and plaintext aware provided that the hash function is random.

1.1. The generic DL-complexity

Let G be a group of prime order q with generator g and let $\mathbb{Z}_q$ denote the field of integers modulo q. The discrete logarithm $\log_g(h)$ of $h \in G$ is the integer $x \bmod q$ in $\mathbb{Z}_q$ that satisfies $g^x = h$. The discrete logarithm is defined modulo q as the order of g is q. Roughly speaking, an algorithm is generic if it does not use the binary encoding of the group elements. It can only use group elements for group operations such as multiplication/division (*generic steps*) and for equality tests. There are many groups for which the fastest known DL-algorithms are generic:

(1) general elliptic curves,
(2) general hyper-elliptic curves of genus 2,
(3) subgroups of prime order q of the multiplicative group $\mathbb{Z}_p^*$ of integers modulo a prime p for which p/q is so large that sieving methods are inefficient.

Following Nechaev [6] and Shoup [9] generic algorithms that compute $\log_g(h)$ for all $h \in G$ must perform $\Omega(\sqrt{q})$ multiplications/divisions. We slightly extend the generic model of Shoup by allowing for generic steps arbitrary multivariate exponentiations. Let the *generic DL-complexity* of a subset $H \subset G$ be the minimal number of generic steps to compute $\log_g(h)$ for *all* $h \in H$.

1.2. Our results

Let $m = \#H$ denote the size of H. We show that the generic DL-complexity is at least $\frac{1}{2}m + o(m)$ for almost all H of size $m \leqslant \sqrt{q}$.[1] On the other hand $\lceil \frac{1}{2}m \rceil + 1$ generic steps are always sufficient. Thus the generic DL-complexity is $\frac{1}{2}m + o(m)$ for almost all subsets $H \subset G$ of size $m \leqslant \sqrt{q}$. For $m = \sqrt{q}$ the generic DL-complexity is $\frac{1}{2}\sqrt{q} + o(\sqrt{q})$, i.e., about $\frac{1}{2\sqrt{2}}$ times the generic DL-complexity $\sqrt{2q}$ for the entire group G. Our main theorem shows a generic DL-complexity lower bound for subsets H of size $m = o(\sqrt{q})$. We subsequently extend this result to the case $m \leqslant \sqrt{q}$. Interestingly, our generic lower bounds hold for arbitrary multivariate exponentiations and not just for multiplications/division.

It is interesting to compare the optimal generic DL-algorithms with the *brute-force method*: given the set of logarithms $\log_g(H)$ test $g^x = h$ for all $x \in \log_g(H)$. This requires in the worst case m and on the average $\frac{1}{2}m$ generic steps. We show that the brute-force method is—up to a factor 2—optimal for almost all subsets H of size $m \leqslant \sqrt{q}$.

1.3. Short secret keys

Our main result justifies to generate secret keys of DL-cryptosystems from random seeds with $\frac{1}{2}\log_2 q$ bits. For this expand a random integer $x' \in_R [0, \sqrt{q}\,]$ of $\frac{1}{2}\log_2 q$ bits using a strong hash function SH into a pseudo-random integer $SH(x') \in_{PR} [0, q[$. The corresponding pair $x', g^{SH(x')}$ is a DL-key pair that is—for generic attacks—nearly as strong as pairs x, g^x for truly random $x \in_R [0, q[$. This is because the generic DL-complexity is for almost all subsets $H \subset G$ of size $\sqrt{q}$ about $\frac{1}{2\sqrt{2}}$ times the generic DL-complexity for G. Clearly, a strong hash function SH yields a set of pseudo-random public keys $SH[0, \sqrt{q}\,] \subset [0, q[$ of size $\Omega(\sqrt{q})$ since otherwise collisions $SH(x') = SH(x'')$ can be constructed using $o(\sqrt{q})$ function evaluations $[0, \sqrt{q}\,] \ni x \mapsto SH(x)$. Moreover, it is reasonable to assume that the set $SH[0, \sqrt{q}\,]$ does not fall into the exceptional class of subsets $H \subset G$ where the DL is easy in the generic model. Generating secret keys from short random seeds can be practical if a strong hash function SH is at hand anyway. Now, there is a theoretical justification that seeds of length $\frac{1}{2}\log_2 q$ are nearly of the highest security level while shorter seeds are less secure.

Moreover, as the generic DL-complexity is $\frac{1}{2}m + o(m)$ for almost all subsets $H \subset G$ of size m, it is sufficient to generate secret DL-keys from seeds x' ranging over a set of size m that is so large that $\frac{1}{2}m$ generic steps are infeasible—at present $m \geqslant 2^{80}$ is sufficient.

[1] The asymptotics as $o(m), o(1)$ is for $m \to \infty$. "For almost all H" means that the fraction of excepted H is *negligible*, i.e., less than $O(m^{-c})$ for all constant $c > 0$.

1.4. Fast pseudo-random exponentiation

An intriguing challenge along this line is to replace SH in the short secret key representation by a pseudo-random function F that speeds up the exponentiation $x' \mapsto g^{F(x')}$.

2. The generic model

The *data* of a generic algorithm are partitioned into group elements in G and *non-group data* (arbitrary data except elements of G). We assume that the prime module q and the set $\log_g(H)$ are given, other non-group data are the collisions defined below. The *generic steps* of a generic algorithm are *multivariate exponentiations*: [2]

$$\text{mex}: \mathbb{Z}_q^d \times G^d \to G,$$

$$(a_1, \ldots, a_d, g_1, \ldots, g_d) \mapsto \prod_i g_i^{a_i} \quad \text{with } d \geqslant 0.$$

Multiplications/divisions are exponentiations with $d = 2, a_1 = 1, a_2 = \pm 1$. The operations mex with $d = 0$ are the *inputs* in G—e.g., g, h are inputs for the DL-computation.

Definition. A *generic algorithm* is a sequence of t generic steps

- $f_1, \ldots, f_{t'} \in G$ (inputs), $1 \leqslant t' < t$,
- $f_i = \prod_{j=1}^{i-1} f_j^{a_j}$ for $i = t' + 1, \ldots, t$, where $(a_1, \ldots, a_{i-1}) \in \mathbb{Z}_q^{i-1}$ depends arbitrarily on i, the non-group input and the set

  $$\mathcal{CO}_{i-1} := \{(j,k) \mid f_j = f_k,\ 1 \leqslant j < k \leqslant i-1\}$$

  of previous *collisions* of group elements.

[2] We count the same generic steps as in [9] however we allow arbitrary multivariate exponentiations while Shoup merely uses multiplication and division. On the surface the technical setup in [9] looks different as groups G are *additive* and associated with a random injective encoding $\sigma: G \to S$ of the group G into a set S of bit strings—the generic algorithm performs arbitrary computations on these bit strings. Addition/subtraction is done by an oracle that computes $\sigma(f_i \pm f_j)$ when given $\sigma(f_i), \sigma(f_j)$ and the specified sign bit. As the encoding σ is random it contains only the information about which group elements coincide—this is what we call the set of *collisions*. We dispense with the encoding σ and let the algorithm make arbitrary use of the set of collisions. We distinguish group and non-group data, a distinction that in the Shoup setup comes automatically with the oracle for the group operation.

The following operations are free of charge: testing equality of group elements, arbitrary computations using non-group data, the selection of the exponents $a_1, \ldots, a_{i-1}$ of a generic step and the selection of a non-group output. A generic algorithm for computing $h \mapsto \log_g(h)$ for $h \in H$ can use the set $\log_g(H)$ of all logarithms of elements in H for free. The probability associated with DL-algorithms refers to the random input $h \in_R H$. Generic algorithms are deterministic, internal coin tosses are useless as the algorithm can always select an optimal coin flip that maximizes its probability of success. The only possible way that the generic steps affect the computation of non-group data such as discrete log's is by collisions of group elements. [3] The example below shows how collisions reveal $\log_g h$.

3. A generic algorithm for computing $\log_g(h)$ for random $h \in H$

We give an example demonstrating the power of generic algorithms. The example algorithm is twice as fast as the brute-force method. It provides a generic DL-complexity upper bound that matches the lower bound of the main theorem. The generic steps of the example algorithm are determined by solving linear equations over $\mathbb{Z}_q$ related to $\log_g(H)$—that computation is free of charge. Let us emphasize that H is an arbitrary subset of G, not a subgroup. In particular, the neutral element of G needs not be in H. For convenience we assume that the generator g is in H.

3.1. Determining the step sequence of the algorithm $\mathcal{A}$

We construct $u_1, \ldots, u_t, v_1, \ldots, v_t \in \mathbb{Z}_q$ for the generic steps $f_i = g^{u_i} h^{v_i}$, $i = 1, \ldots, t$, as follows. Select distinct elements $x_1, \ldots, x_{2t-2} \in \log_g(H)$, with $x_1 = 1 = \log_g(g)$, and recursively determine $u_1, \ldots, u_t, v_1, \ldots, v_t \in \mathbb{Z}_q$ such that

$$(u_1, v_1) := (1, 0), \qquad (u_2, v_2) := (0, 1),$$

[3] The decision to terminate with a generic step may arbitrarily depend on the non-group input—such as q and $\log_g(H)$—and the previous collisions. Thus, t arbitrarily depends on the given non-group data.

and thus

$$\begin{aligned} &x_1(v_1 - v_2) = u_2 - u_1,\\ &x_{2i-4}(v_1 - v_i) = u_i - u_1,\\ &x_{2i-3}(v_2 - v_i) = u_i - u_2, \end{aligned} \quad \text{for } i = 3, \ldots, t.$$

This system of equations in the unknowns $u_3, \ldots, u_t$, $v_3, \ldots, v_t$ is always solvable. Given $u_1, \ldots, u_{i-1}$, $v_1, \ldots, v_{i-1}$ the two linear equations for u_i, v_i have determinant $x_{2i-4} - x_{2i-3}$ which is nonzero in $\mathbb{Z}_q$. Therefore u_i and v_i are uniquely determined. Note that we cannot have $v_1 = v_i$ or $v_2 = v_i$. If $v_1 = v_i$ we have $u_1 = u_i$ and this implies $u_i - u_2 = x_{2i-3}(v_2 - v_i) = u_1 - u_2 = x_1(v_2 - v_1)$, hence $x_{2i-3} = x_1$. This has been excluded as the x_i are distinct. As $v_1 \neq v_i$, $v_2 \neq v_i$ we have

$$x_{2i-4} = \frac{u_i - u_1}{v_1 - v_i}, \qquad x_{2i-3} = \frac{u_i - u_2}{v_2 - v_i}.$$

Moreover, $(u_i, v_i) \neq (u_j, v_j)$ holds for $3 \leqslant i, j, \leqslant t$ and $i \neq j$—since otherwise we must have $x_{2i-4} = x_{2j-4}$ which is excluded as $x_1, \ldots, x_{2t}$ are distinct. In summary, the pairs $(u_1, v_1), \ldots, (u_t, v_t)$ are pairwise distinct.

Let $\mathcal{A}$'s generic steps compute

$$f_k := g^{u_k} h^{v_k} \quad \text{for } k = 1, \ldots, t,$$

in particular for $k = 1, 2$ we get $f_1 = g$, $f_2 = h$. We have

$$\begin{aligned} &f_i = f_j\\ &\quad \text{iff } u_i + v_i \log_g(h) = u_j + v_j \log_g(h)\\ &\quad \text{iff } \log_g(h) = \frac{u_j - u_i}{v_i - v_j}. \end{aligned}$$

$\mathcal{A}$ gets from a collision $f_i = f_j$ the logarithm

$$\log_g(h) = \frac{u_j - u_i}{v_i - v_j}.$$

By the construction of $u_1, \ldots, u_t, v_1, \ldots, v_t$, $\mathcal{A}$ gets $\log_g(h)$ for $\log_g(h) \in \{x_1, \ldots, x_{2t-3}\}$. Otherwise $\mathcal{A}$ guesses that $\log_g(h) = x_{2t-2}$. $\mathcal{A}$ succeeds for random $h \in_R H$, $\#H = m$, with probability $(2t - 2)/m$. The case that $\log_g(h) = x_{2t-2}$ contributes $1/m$ to the success probability.

In order to succeed for all $h \in H$ of even size m we use the algorithm with $t = \frac{1}{2}(m + 2)$. Then $2t - 2 = m$ and $x_1, \ldots, x_{2t-2}$ exhaust H. The number of generic steps is $\frac{1}{2}m + 1$. This proves the following proposition where we let m—for simplicity—be even.

Proposition 1. *The above-mentioned algorithm $\mathcal{A}$ computes $\log_g(h)$ for random $h \in H$ and even $m = \#H$ with probability $(2t - 2)/m$ using t generic steps. $\mathcal{A}$ always succeeds for $t = \frac{1}{2}m + 1$.*

Main Theorem 2. *Every generic algorithm $\mathcal{A}$ with t generic steps satisfies for almost all subsets $H \subset G$ of size m with $m = o(\sqrt{q})$:*

$$\Pr_{h \in_R H}\left[\mathcal{A}(h) = \log_g(h)\right] \leqslant \frac{2t}{m} + o(1).$$

4. The generic DL-complexity for small subsets

The upper bound $2t/m + o(1)$ of $\mathcal{A}$'s probability of success in Theorem 2 is tight as the example algorithm succeeds with probability $(2t - 2)/m$. Hence, the generic complexity of $\log_g$ is at least $\frac{1}{2}m + o(1)$ for almost all subsets H of size $m = o(\sqrt{q})$. Below we extend the latter result to the case $m \leqslant \sqrt{q}$.

Proof of Theorem 2. Let $H = \{g^{x_1}, \ldots, g^{x_m}\} \subset G$ be a random multiset, where the random elements $x_i \in_R \mathbb{Z}_q$ for $i = 1, \ldots, m$ are chosen independently at random with repetition. H has size m counted with multiplicities. As $m = o(\sqrt{q})$, repetitions $x_i = x_j$, $i < j$, have probability $o(1)$ and are disregarded in the following. Importantly, the elements in H are mutually independent. Let $\mathcal{A}$'s generic steps compute

$$f_k := g^{u_k} h^{v_k} \quad \text{for } k = 1, \ldots, t,$$

where the pairs $(u_k, v_k) \in \mathbb{Z}_q^2$ are pairwise distinct and (u_k, v_k) depends arbitrarily on the set of logarithms $\log_g(H) \subset \mathbb{Z}_q$ and on previous collisions $f_i = f_j$ with $i < j < k$. The distinctness of the (u_t, v_t) is not a restriction as repetitions can easily be removed. For simplicity we do not require that $g, h \in \{f_1, \ldots, f_t\}$.

We first consider *constant* step sequences $\boldsymbol{u} = (u_1, \ldots, u_t)$, $\boldsymbol{v} = (v_1, \ldots, v_t) \in \mathbb{Z}_q^t$ for which u_k, v_k do not depend on previous collisions but depend arbitrarily on $\log_g(H)$. In case of a collision $f_i = f_j$ we have

$$\log_g(h) = \frac{u_j - u_i}{v_i - v_j}.$$

(We have $v_i \neq v_j$, as $v_i = v_j$ implies $u_i = u_j$ and the case $(u_i, v_i) = (u_j, v_j)$ has been excluded.) We denote

$$x_{i,j} := \frac{u_j - u_i}{v_i - v_j}$$

and

$$H_{\boldsymbol{u},\boldsymbol{v}} := \{x_{i,j} \in \log_g(H) \mid 1 \leqslant i < j \leqslant t\}.$$

Thus $\mathcal{A}$ succeeds if $\log_g(h) \in H_{\boldsymbol{u},\boldsymbol{v}}$. Hence $p := \#H_{\boldsymbol{u},\boldsymbol{v}}/m$ is, for random $h \in_R H$, the probability that there is a collision.

If $\log_g(h) \notin H_{\boldsymbol{u},\boldsymbol{v}}$ then all $\mathcal{A}$ gets to know is that $\log_g(h) \in \log_g(H) \setminus H_{\boldsymbol{u},\boldsymbol{v}}$. Then $\mathcal{A}$ can at best guess for $\log_g(h)$ one of the $m - \#H_{\boldsymbol{u},\boldsymbol{v}}$ elements in $\log_g(H) \setminus H_{\boldsymbol{u},\boldsymbol{v}}$. Thus $\mathcal{A}$'s probability of success is for given H and random $h \in H$ at most

$$p + (1-p)\frac{1}{m - \#H_{\boldsymbol{u},\boldsymbol{v}}} = p + \frac{1}{m} = \frac{\#H_{\boldsymbol{u},\boldsymbol{v}}}{m} + \frac{1}{m}.$$

We see from Lemma 3 that $\#H_{\boldsymbol{u},\boldsymbol{v}} \leqslant 2t + o(m)$ for almost all $H \subset G$ of size m. Here we use that $t^2/q = o(1)$ holds for $t = o(\sqrt{q})$, and also that $\exp(-2m_t) \leqslant \exp(-2\sqrt{m})$ is negligible for $t \leqslant \frac{1}{2}m - \sqrt{m}$ while $\#H_{\boldsymbol{u},\boldsymbol{v}} \leqslant 2t + o(m)$ is trivial for $t > \frac{1}{2}m - \sqrt{m}$. Therefore Lemma 3 proves Theorem 2 for constant $\boldsymbol{u}, \boldsymbol{v}$.

Lemma 3. *For random H of size m and $m_t := m - 2t + 2$ we have*

$$\Pr_H\Big[\max_{\boldsymbol{u},\boldsymbol{v} \in \mathbb{Z}_q^t} \#H_{\boldsymbol{u},\boldsymbol{v}} \geqslant 2t - 2 + m_t t^2/q\Big] \leqslant \exp(-2m_t).$$

Proof. Let $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{Z}_q^{2t}$ be a constant step sequence such that $\#H_{\boldsymbol{u},\boldsymbol{v}}$ is maximum for some H. Consider the corresponding equations

$$x_{i,j}(v_i - v_j) = u_j - u_i \quad \text{for } x_{i,j} \in H_{\boldsymbol{u},\boldsymbol{v}}. \tag{1}$$

Select a maximum subset of the linear equations in (1) that are linearly independent—when the constants $u_1, \ldots, u_t, v_1, \ldots, v_t$ are replaced by variables over $\mathbb{Z}_q$. That linear independence is a property of the set of triples $(x_{i,j}, i, j)$ with $x_{i,j} \in H_{\boldsymbol{u},\boldsymbol{v}}$. Let I denote the set of pairs (i, j) corresponding to these linearly independent equations and let $H_I := \{x_{i,j} \mid (i,j) \in I\}$. We next show that $\#I = 2t - 2$. The solutions of Eqs. (1) for $(i, j) \in I$ form a linear space of dimension $\geqslant 2$: if $(\boldsymbol{u}, \boldsymbol{v})$ is a solution then so is $(\alpha\boldsymbol{u}, \beta\boldsymbol{v})$ for $\alpha, \beta \in \mathbb{Z}_q$, and thus $\#I \leqslant 2t - 2$. Moreover for $t \geqslant 4$, $\#I = 2t - 2$ and random $x_{i,j} \in_R \mathbb{Z}_q$ Eqs. (1) for $(i, j) \in I$ are linearly independent except for an event of probability $O(1/q)$.

Next we prove that $2t - 2$ linearly independent equations for $(i, j) \in I$ determine the step sequence $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{Z}_q^{2t}$ up to constant factors $\alpha, \beta \in \mathbb{Z}_q$. Suppose there exist two such step sequences $(\boldsymbol{u}, \boldsymbol{v}), (\boldsymbol{u}', \boldsymbol{v}')$ satisfying $(\boldsymbol{u}, \boldsymbol{v}) \neq (\alpha\boldsymbol{u}', \beta\boldsymbol{v}')$ for all $\alpha, \beta \in \mathbb{Z}_q$. If two such step sequences satisfy the linear equations (1) for all $(i, j) \in I$ for the same I then there exist $\lambda, \lambda' \in \mathbb{Z}_q$ and $(i, j) \notin I$ such that

$$\frac{\lambda(u_j - u_i) + \lambda'(u'_j - u'_i)}{\lambda(v_i - v_j) + \lambda'(v'_i - v'_j)} \in \log_g(H) \setminus H_{\boldsymbol{u},\boldsymbol{v}}$$

holds for some $1 \leqslant i < j \leqslant t$. Then $(\boldsymbol{u}^*, \boldsymbol{v}^*) := \lambda(\boldsymbol{u}, \boldsymbol{v}) + \lambda'(\boldsymbol{u}', \boldsymbol{v}')$ is a step sequence for which $H_{\boldsymbol{u},\boldsymbol{v}}$ is properly contained in $H_{\boldsymbol{u}^*,\boldsymbol{v}^*}$—contradicting to the assumption that $\#H_{\boldsymbol{u},\boldsymbol{v}}$ is maximum. This proves the claim that the step sequence $(\boldsymbol{u}, \boldsymbol{v})$ is determined—up to constant factors—by the $x_{i,j} \in H_I$ via Eqs. (1).

We call the $x_j \in \log_g(H) \setminus H_I$ *free*. There are $m - \#I = m_t$ free $x_j \in \log_g(H)$. The free x_j are statistically independent of $(\boldsymbol{u}, \boldsymbol{v})$ as $(\boldsymbol{u}, \boldsymbol{v})$ is determined by the $x_{i,j} \in H_I$. The free x_j are uniformly distributed over $\log_g(H)$. Hence,

$$\Pr_H[x_j \in H_{\boldsymbol{u},\boldsymbol{v}} \setminus H_I] = \frac{\binom{t}{2} - \#I}{q}.$$

Therefore, the expected number of free $x_j \in H_{\boldsymbol{u},\boldsymbol{v}} \setminus H_I$ is

$$m_t \frac{\binom{t}{2} - \#I}{q} \leqslant m_t \frac{\binom{t}{2}}{q}.$$

Next we bound the deviation from the expected value.

The events $[x_j \in H_{\boldsymbol{u},\boldsymbol{v}} \setminus H_I]$, for the free x_j, are m_t Poisson trials that are mutually independent. By Chernoff's bound we have for $\varepsilon > 0$:

$$\Pr_H\left[\#\{\text{free } x_j \in H_{\boldsymbol{u},\boldsymbol{v}} \setminus H_I\} \geqslant m_t\binom{t}{2}\frac{1+\varepsilon}{q}\right] \leqslant \exp(-2\varepsilon m_t). \tag{2}$$

(More precisely, we use Hoeffding's bound [4] as in Exercise 4.7 of [5].) Inequality (2) with $\varepsilon = 1$ proves Lemma 3 as H_I consists of $2t - 2$ non-free $x_{i,j}$. □

To complete the proof of Theorem 2, consider the case that u_k, v_k are recursively defined depending on previous collisions $f_i = f_j$ with $i < j < k$. Consider

the first collision for which j is minimal. The first collision occurs for a constant step sequence $(\boldsymbol{u}', \boldsymbol{v}') \in \mathbb{Z}_q^{2t'}$. This is because all non-group data are constant—i.e., not depending on h—unless there is a collision. A first collision occurs if $\log_g(h) \in H_{\boldsymbol{u}',\boldsymbol{v}'}$ for constant $\boldsymbol{u}', \boldsymbol{v}'$, which happens with probability $\#H_{\boldsymbol{u}',\boldsymbol{v}'}/m$. This shows that $\mathcal{A}$'s probability of success is at most the maximum of $\#H_{\boldsymbol{u}',\boldsymbol{v}'}/m + 1/m$ over all constant $\boldsymbol{u}', \boldsymbol{v}' \in \mathbb{Z}_q^{t'}$ for $t' \leqslant t$. By Lemma 3 this maximum is at most $2t/m + o(1)$ for almost all H of size m. □

The case $m \leqslant \sqrt{q}$. By the previous argument, lower bound proofs need only to cover generic algorithms with constant step sequences $\boldsymbol{u}, \boldsymbol{v}$. If $m \leqslant \sqrt{q}$ we can in the proof of Theorem 2 still disregard repetitions $x_i = x_j$, $i < j$, of the random $x_i \in \log_g(H)$ as the expected number of repetitions is at most $\binom{m}{2}/q \leqslant \frac{1}{2}$. Therefore, inequality (2) holds for $m \leqslant \sqrt{q}$. Setting

$$m := \sqrt{q}, \qquad t := \tfrac{1}{2}\sqrt{q}(1-\varepsilon),$$

$$m_t := m - 2t + 2$$

we have

$$m_t = \varepsilon\sqrt{q} + 2 \quad \text{and}$$

$$m_t \frac{\binom{t}{2}}{q} = \varepsilon(1-\varepsilon)^2(1+\varepsilon) \cdot \tfrac{1}{8}m.$$

As there are $2t - 2 \leqslant m(1-\varepsilon)$ non-free $x_{i,j} \in H_{\boldsymbol{u},\boldsymbol{v}}$, inequality (2) shows that the event

$$\frac{\#H_{\boldsymbol{u},\boldsymbol{v}}}{m} \geqslant (1-\varepsilon) + \tfrac{1}{8}\varepsilon(1-\varepsilon)^2(1+\varepsilon)$$

has probability at most

$$\exp(-2\varepsilon m_t) \approx \exp(-2\varepsilon^2\sqrt{q})$$

for random H of size m. Moreover, $(1-\varepsilon) + \frac{1}{8}\varepsilon(1-\varepsilon)^2(1+\varepsilon) \geqslant 1-\varepsilon$, and $\exp(-2\varepsilon^2\sqrt{q})$ is negligible for $\varepsilon = q^{-1/5}$. So let $\varepsilon := q^{-1/5}$.

We conclude that generic algorithms with $t := \frac{1}{2}\sqrt{q} - q^{3/10}$ generic steps succeed, for almost all H of size $m = \sqrt{q}$, at most with probability $1 - q^{-1/5}$. This shows that the generic DL-complexity is for $m = \sqrt{q}$ at least $\frac{1}{2}\sqrt{q} - q^{3/10} = \frac{1}{2}m + o(m)$ for almost all subset H of size $m = \sqrt{q}$. Moreover, the cryptographic relevant q, $q \approx 2^{160}$, satisfy $\frac{1}{2}\sqrt{q} - q^{3/10} \approx \frac{1}{2}\sqrt{q} \approx 2^{79}$. Therefore, the generic DL-complexity for subsets of size 2^{80} is close to 2^{79}.

The case $m = q$. For $H = G$ and $t < \sqrt{2q}$ we have that $\#H_{\boldsymbol{u},\boldsymbol{v}}/q \leqslant \binom{t}{2}/q < 1$. Therefore, the generic DL-complexity is at least $\sqrt{2q}$ for the entire group G.

Acknowledgements

I wish to thank Carl Pomerance for some useful communications on this subject and Marc Fischlin for proof reading the manuscript.

References

[1] R. Cramer, V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, in: Proc. Crypto'98, Lecture Notes in Comput. Sci., Vol. 1462, Springer, Berlin, 1998, pp. 13–25.

[2] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory 22 (6) (1976) 644–654.

[3] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inform. Theory 31 (1985) 469–472.

[4] W. Hoeffding, Probability in equalities for sums of bounded random variables, J. Amer. Stat. Assoc. 58 (1963) 13–30.

[5] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, UK, 1995.

[6] V.I. Nechaev, Complexity of a determinate algorithm for the discrete logarithm, Math. Notes 55 (1994) 165–172.

[7] T. Okamoto, Provably secure identification schemes and corresponding signature schemes, in: Proc. Crypto'92, Lecture Notes in Comput. Sci., Vol. 740, Springer, Berlin, 1992, pp. 31–53.

[8] C.P. Schnorr, Efficient signature generation for smart cards, J. Cryptology 4 (1994) 161–174.

[9] V. Shoup, Lower bounds for discrete logarithms and related problems, in: Proc. Eurocrypt'97, Lecture Notes in Comput. Sci., Vol. 1233, Springer, Berlin, 1997, pp. 256–266.

[10] C.P. Schnorr, Security of almost all discrete log bits, in: Electronic Colloquium on Computational Complexity, Report TR 98-033. Available at http://www.eccc.uni-trier.de/eccc/.

[11] C.P. Schnorr, M. Jakobsson, Security of signed ElGamal encryption, in: T. Okamoto (Ed.), Advances in Cryptology —Asiacrypt'00, Lecture Notes in Comput. Sci., Vol. 1976, Springer, Berlin, 2000, pp. 73–89.