Security of Almost ALL Discrete Log Bits

C.P. Schnorr

Fachbereich Mathematik/Informatik
Universität Frankfurt, Germany
and
Bell Laboratories
Murray Hill, New Jersey
schnorr@cs.uni-frankfurt.de

June 8, 1998

Abstract

Let G be a finite cyclic group with generator α and with an encoding so that multiplication is computable in polynomial time. We study the security of bits of the discrete log x when given $\exp_\alpha(x)$, assuming that the exponentiation function $\exp_\alpha(x) = \alpha^x$ is one-way. We reduce he general problem to the case that G has odd order q. If G has odd order q the security of the least-significant bits of x and of the most significant bits of the rational number $\frac{x}{q} \in [0,1)$ follows from the work of Peralta [P85] and Long and Wigderson [LW88]. We generalize these bits and study the security of consecutive *shift bits* $\mathrm{lsb}(2^{-i}x \bmod q)$ for $i = k+1, ..., k+j$. When we restrict $\exp_\alpha$ to arguments x such that some sequence of j consecutive shift bits of x is constant (i.e., not depending on x) we call it a 2^{-j}-*fraction* of $\exp_\alpha$.

For groups of odd group order q we show that every two 2^{-j}-fractions of $\exp_\alpha$ are equally one-way by a polynomial time transformation: Either they are all one-way or none of them. Our *key theorem* shows that arbitrary j consecutive shift bits of x are simultaneously secure when given $\exp_\alpha(x)$ iff the 2^{-j}-fractions of $\exp_\alpha$ are one-way. In particular this applies to the j least-significant bits of x and to the j most-significant bits of $\frac{x}{q} \in [0,1)$. For one-way $\exp_\alpha$ the individual bits of x are secure when given $\exp_\alpha(x)$ by the method of Håstad, Näslund [HN98]. For groups of even order $2^s q$ we show that the j least-significant bits of $\lfloor x/2^s \rfloor$, as well as the j most-significant bits of $\frac{x}{q} \in [0,1)$, are simultaneously secure iff the 2^{-j}-fractions of $\exp_{\alpha'}$ are one-way for $\alpha' := \alpha^{2^s}$.

We use and extend the models of generic algorithms of Nechaev (1994) and Shoup (1997). We determine the generic complexity of inverting fractions of $\exp_\alpha$ for the case that α has prime order q. As a consequence, arbitrary segments of $(1-\varepsilon)\lg q$ consecutive shift bits of random x are for constant $\varepsilon > 0$ simultaneously secure against generic attacks. Every generic algorithm using t generic steps (group operations) for distinguishing bit strings of j consecutive shift bits of x from random bit strings has at most advantage $O((\lg q)\, j\sqrt{t}\,(2^j/q)^{\frac{1}{4}})$.

Keywords. Hard bit, secure bit, discrete logarithm, exponentiation, fractions of exponentiation, simultaneous security of bits, one-way function, generic network, generic one-wayness.

1 Introduction

An interesting problem for a one-way function $f(x)$ is to locate the hard/secure bits in the n-bit argument x which cannot be predicted from the function value $f(x)$ in polynomial time with success probability $\frac{1}{2} + 1/\text{poly}(n)$. BLUM and MICALI [BM84] introduced the notion of hard bits, respectively hard-core predicates. GOLDREICH and LEVIN [GL89] have shown that every one-way function f has logarithmically many one-bit predicates that are simultaneously secure for given $f(x)$.

Specifically, the exponentiation function $\exp_\alpha(x) = \alpha^x$ of a finite cyclic group G with generator α is a well known candidate one-way function that gives rise to various cryptographic applications. Let P be a prime such that $(P-1)/2^s$ is an odd integer and let $G = \mathbf{Z}_P^*$ be the multiplicative group of integers modulo P. Peralta [P85] shows that the $O(\lceil \lg\lg P \rceil)$ least-significant bits of $\lfloor x/2^s \rfloor$ are simultaneously secure when given $\exp_\alpha(x) \in \mathbf{Z}_P^*$ provided that $\exp_\alpha$ is one-way. LONG and WIGDERSON [LW88] show that the $O(\lceil \lg\lg P \rceil)$ most-significant bits of the rational number $\frac{x}{q} \in [0,1)$ are secure when given $\exp_\alpha(x) \in \mathbf{Z}_P^*$. HASTAD, SCHIFT, SHAMIR [HSS93] prove that $n/2$ bits of an n bit discrete log are simultaneously secure for $G = \mathbf{Z}_N^*$ with a random Blum modulus N provided that factoring Blum integers is hard. A Blum integer is a product of two primes that are both congruent 3 mod 4. Proving simultaneous security of more than logarithmically many discrete log bits is still an open problem for general groups.

In this paper we study the discrete logarithm for arbitrary cyclic groups G with an encoding so that multiplication is computable in polynomial time, polynomial time refers to the bit length n of the order of G. We generalize the least-significant bits of $x \bmod q$ and the most-significant bits of $\frac{x}{q}$ and we study the security of consecutive *shift bits* $\text{lsb}(2^{-i}x \bmod q)$ for $i = k+1, ..., k+j$ when given $\exp_\alpha(x)$. We reduce the general problem to the case that the group G has odd order q. When we restrict $\exp_\alpha$ to arguments x such that some sequence of j consecutive shift bits of x is constant in $\{0,1\}^j$ (i.e., does not depend on x) we call it a 2^{-j}-*fraction* of $\exp_\alpha$.

For groups of odd order q we prove in Section 3 that all 2^{-j}-fractions of $\exp_\alpha$ are equally one-way by polynomial time transformations: Either they are all one-way or none of them. We prove in Theorem 5 that arbitrary j consecutive shift bits of x are simultaneously secure when given $\exp_\alpha(x)$ iff the 2^{-j}-fractions of $\exp_\alpha$ are one-way. In particular this applies to the j least-significant bits of x as well as to the j most-significant bits of $\frac{x}{q}$. We note that if $\exp_\alpha$ is one-way then all individual bits of x are secure when given $\exp_\alpha(x)$. This follows from the proof method of HÅSTAD, NÄSLUND [HN98].

In Section 4 we consider groups G of even order $2^s q$. We show that given $\exp_\alpha(x)$ the j least-significant bits of $\lfloor x/2^s \rfloor$ as well as the j most-significant bits of $\frac{x}{q}$ are simultaneously secure iff the 2^{-j}-fractions of $\exp_{\alpha'}$ are one-way for $\alpha' := \alpha^{2^s}$. Note that $\exp_{\alpha'}$ is associated with the subgroup $G' \subset G$ of 2^s-powers which has generator α' and order q. We transform the given $y = \exp_\alpha(x) \in G$ in poly-time into some $y' \in G'$ such that the $(s+i)$-th bit of $\log_\alpha(y)$ coincides with the i-th bit of $\log_{\alpha'}(y')$.

In Section 5 we prove one-wayness of 2^{-j}-fractions of $\exp_\alpha$ in the model of *generic algorithms*, i.e., for algorithms that do not depend on the encoding of the group. Models of generic algorithms have been introduced by Nechaev [Ne94] and Shoup [Sh97], we further extend these modelsby enlarging the class of group operations. The Nechaev, Shoup generic lower bounds for the discrete logarithm extend to small fractions of $\exp_\alpha$. For groups of prime order q we determine the generic complexity of inverting fractions of $\exp_\alpha$. As a consequence almost all discrete log bits are simultaneously secure against generic attacks. Generic one-wayness of fractions of $\exp_\alpha$ is the best result we can hope for, as the known complexity lower bounds for the discrete logarithm are bound to the generic model. We have the same evidence for the hardness of the discrete logarithm problem and for the simultaneous security of almost all discrete log bits. In the non-generic setting these problems are completely open. For generic algorithms these problems are equivalent by Theorems 11 and 13.

2 Preliminaries

Notation. We use for computation the model of probabilistic poly-time Turing machines (pptm for short) running in time poly(n) where n is the length of the input. We let lg denote the logarithm with base 2. If S is a set and $\mathcal{D}$ a distribution on S then by $x \in_{\mathcal{D}} S$ we mean an x chosen at random according to the distribution $\mathcal{D}$. If $\mathcal{D}$ is the uniform distribution we write $x \in_R S$.

A probability $\Pr_{\mathcal{D}}$ refering to a distribution $\mathcal{D}$ on S is called *negligible* if $\Pr_{\mathcal{D}} < n_S^{-c}$ for all constants $c > 0$ and for all sufficiently large n_S. Here the set S is variable refering to a family of sets. A *one-way function* is a poly-time computable function f such that for every pptm M the probability that $f(M(f(x))) = x$ is negligible. The probability is taken over the random x and M's random coin flips.

Let $\mathcal{D}, \mathcal{D}'$ be distributions on the same space S. We call $\mathcal{D}, \mathcal{D}'$ *poly-time indistinguishable* if for all pptm D $|\Pr_{s \in_{\mathcal{D}} S}[D(s) = 1] - \Pr_{s' \in_{\mathcal{D}'} S}[D(s') = 1]|$ is negligible.

Let G be a cyclic group with generator α and order $2^s q$ with an odd integer q and $s \geq 0$, $0 < 2^s q < 2^n$. If $y = \exp_\alpha(x) = \alpha^x$ then $x = \log_\alpha(y)$ is the *discrete logarithm* of y. Discrete log's range over the ring $\mathbf{Z}_{2^s q} = \mathbf{Z}_{/2^s q\mathbf{Z}}$ of integers modulo $2^s q$. We represent elements $x \in \mathbf{Z}_{2^s q}$ by their least non-negative residue $[x]_{2^s q}$ in the interval $[0, 2^s q)$. We use $[x]_{2^s q}$ for arithmetic expressions over $\mathbf{Z}$ while the arithmetic for $x \in \mathbf{Z}_{2^s q}$ is always modulo $2^s q$. Except for Section 4 we let the group order be odd, $|G| = q$. In Section 4 we reduce the general problem to the case of odd group order.

Least-significant, most-significant and shift bits. The binary representation $x = [x]_q = \sum_{i=1}^{n} \mathrm{ls}_i(x) 2^{i-1}$ uses $\mathrm{ls}_i(x) := \lfloor [x]_q / 2^{i-1} \rfloor$, also called the i-th *least-significant* bit of x. Let $L_j(x)$ denote the integer $\sum_{i=1}^{j} \mathrm{ls}_i(x) 2^{i-1}$ of the first j ls-bits of x. The bits $\mathrm{ms}_i(x)$ of the binary representation $\frac{1}{q}[x]_q = \sum_{i=1}^{\infty} \mathrm{ms}_i(x) 2^{-i} \in [0,1)$ are also called the *most significant* bits of x [LW88]. Identifying $true = 1$, $false = 0$ we have $\mathrm{ms}_1(x) =$ "$x > q/2$" and $\mathrm{ms}_i(x) =$ "$[2^{i-1}x]_q > q/2$" $=$ "$[2^i x]_{2q} > q$" for $i = 1, 2, \ldots$.

Definition. We call $\mathrm{lsb}(2^{-i}x) := \mathrm{ls}_1(\,[2^{-i}x]_q)$ for arbitrary integers i the i-th *shift bit* of x. Note that $[2^{-i}x]_q$ is the integer in $[0, q)$ that represents $2^{-i}x \bmod q$ where we divide modulo q by 2^i.

We have $\mathrm{lsb}(2^i x) = \mathrm{ms}_i(x)$ for $i = 1, 2, ...$ because $\frac{1}{q}[x]_q > q/2$ iff $\ell(2x) = 1$. Lemma 1 shows that the bits $\mathrm{lsb}(2^{-i}x)$ for $i = 0, ..., j-1$ are equivalent to the first j ls-bits of x. Thus the shift bits of x generalize at the same time both the ls-bits and the ms-bits of arbitrary shifts of $x \in \mathbf{Z}_q$.

Lemma 1. $[2^{-j}x]_q = 2^{-j}(\,[x]_q + \sum_{i=0}^{j-1} \mathrm{lsb}(2^{-i}x)\,2^i\,q\,)$ *for* $j = 1, 2, ...$

Proof by induction on j. For $j = 1$ we have $[\frac{1}{2}x]_q = \frac{1}{2}(\,[x]_q + \mathrm{lsb}(x)q\,)$, which describes *binary division* for $\mathbf{Z}_q$, see figure 1. This holds because we have $[\frac{1}{2}x]_q = \frac{1}{2}[x]_q$ for even $[x]_q$ and $[\frac{1}{2}x] = \frac{1}{2}([x]_q + q)$ for odd $[x]_q$. The claim for $j > 1$ follows by induction applying the case $j = 1$ with x replaced by $2^{-j+1}x$. □

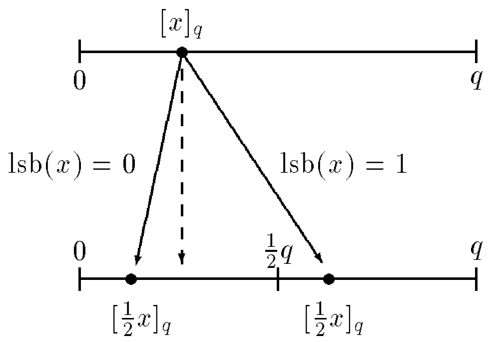

figure 1: binary division

By multiplying the equation of Lemma 1 with 2^j and taking it modulo 2^j we get

$$[x]_q \;=\; -\sum_{i=0}^{j-1} \mathrm{lsb}(2^{-i}x)2^i q \bmod 2^j. \tag{1}$$

Thus the bits $\mathrm{lsb}(2^{-i}x)$ for $i = 0, \ldots, j-1$ are equivalent to the first j ls-bits of x. In particular $L_j(x)$ and $\mathrm{lsb}(2^{-i}x)$ for $i = 0, \ldots, j-1$ are equivalent by poly-time transformations.

Replacing in Lemma 1 x by $2^j x$ we get

$$0 \;\le\; \tfrac{1}{q}[x]_q - \sum_{i=1}^{j} \mathrm{lsb}(2^i x)2^{-i} \;<\; 2^{-j}, \qquad \sum_{i=1}^{j} \mathrm{lsb}(2^i x)2^{-i} \;=\; \lfloor \tfrac{2^j}{q}[x]_q \rfloor,$$

which again shows that $\mathrm{lsb}(2^i x) = \mathrm{ms}_i(x)$. We resume these equivalences:

Proposition 2. *The following entities are computationally equivalent for given* q :

- $L_j(x) = x \bmod 2^j$,
- $\mathrm{ls}_i(x)$ *for* $i = 1, ..., j$, *the first* j ls-*bits of* x,
- $\mathrm{lsb}(2^{-i}x) = \mathrm{ms}_{j-i}(2^{-j}x)$ *for* $i = 0, \ldots, j-1$ *the first* j ms-*bits of* $2^{-j}x$.

Corollary 3. *Let* G *have odd order* q. *Given* $\exp_\alpha(x)$ *every two shift bits* $\mathrm{lsb}(2^{-i}x)$ *and* $\mathrm{lsb}(2^{-j}x)$ *of random* $x \in \mathbf{Z}_q$ *are equally secure by poly-time transformations.*

Proof. Let $y = \exp_\alpha(x) \in_R G$. The i-th shift bit $\mathrm{lsb}(2^{-i}x)$ of x coincides with the j-th shift bit $\mathrm{lsb}(2^{-j}x')$ of $x' = 2^{j-i}x$. We can attack $\mathrm{lsb}(2^{-i}x)$ as the j-th shift bit of x' when given $\exp_\alpha(x')$. We get $y' = \exp_\alpha(x')$ as $y' := y^z$ with $z := 2^{j-i} \bmod q$. The transformation $y \mapsto y'$ permutes G in polynomial time. It is not assumed in the theorem that the discrete logarithm problem for G is hard. □

Writing discrete Log's with all bits equally secure. If we encode the discrete logarithm x into the bit sequence $\mathrm{lsb}(2^{-i}x)$ for $i = 1, ..., n$ then the individual bits of the encoding are equally secure when given $\exp_\alpha(x)$. From the encoding we easily get x via Equation 1.

3 Simultaneous security of discrete log bits, odd group order

Let G be a cyclic group with odd order q and generator α. We introduce the notion of 2^{-j}-fraction of the exponentiation function $\exp_\alpha$. Our *key* Theorem 5 shows that j consecutive shift bits of the discrete logarithm $x \in \mathbf{Z}_q$ are simultaneously secure when given $\exp_\alpha(x)$ iff the 2^{-j}-fractions of $\exp_\alpha$ are one-way. All 2^{-j}-fractions of $\exp_\alpha$ are equally one-way by poly-time transformations. Moreover the first j ls-bits and the first j ms-bits of x are equally secure when given $\exp_\alpha$.

We call the bits $\mathrm{lsb}(2^{-i}x)$ for $i = k+1, ..., k+j$ *simultaneously secure* if the bit string $(\mathrm{lsb}(2^{k+1}x), ..., \mathrm{lsb}(2^{j+k+1}x))$ is poly-time indistinguishable from random $z \in_R \{0,1\}^j$ when given $\exp_\alpha(x) \in_R G$. Formally, for every pptm D the difference

$$|\Pr[D(\exp_\alpha(x), (\mathrm{lsb}(2^{k+1}x), ..., \mathrm{lsb}(2^{j+k+1}x))) = 1] - \Pr[D(\exp_\alpha(x), z) = 1]|$$

must be negligible where the probability is over random x, z and D's coin flips.

2^{-j}-fractions of the exponentiation function. We call a part of $\exp_\alpha$ – where j consecutive shift bits of x are restricted to some constant 0,1-vector – a 2^{-j}-*fraction* of $\exp_\alpha$. A 2^{-j}-fraction of $\exp_\alpha$ – defined by a 0,1-vector $(c_1, ..., c_j)$ and a integer $k \in \mathbf{Z}$ – is the restriction of $\exp_\alpha$ to arguments x satisfying $\mathrm{lsb}(2^{k+i}x) = c_i$ for $i = 1, ..., j$.

Clearly, if $\exp_\alpha$ is one-way and $j = O(\lg n)$ then some 2^{-j}-fraction of $\exp_\alpha$ must be one-way. However, if 2^j is not polynomially bounded it is conceivable that no 2^{-j}-fraction is one-way.

We next normalize in various ways the problem whether a 2^{-j}-fraction of $\exp_\alpha$ is one-way. The various 2^{-j}-fractions of $\exp_\alpha$ are *all* equally one-way by polynomial time transformations: Either all 2^{-j}-fractions of $\exp_\alpha$ are one-way or none of them. In particular the one-wayness of a random 2^{-j}-fraction of $\exp_\alpha$ – where j consecutive shift bits of x are set to a random vector $(c_1, ..., c_j) \in_R \{0,1\}^j$ – and that of the particular 2^{-j}-fraction – where j consecutive shift bits of x are set to zero – are equivalent by polynomial time transformations. Propositions 2 and 4 will be used throughout the reminder of the paper.

Proposition 4. *The following problems are polynomial time equivalent* :

- *given* $\exp_\alpha(x)$ *and arbitrary* j *consecutive shift bits of random* x, *find* x.
- *given* $\exp_\alpha(x)$ *for random* x *with* $x = 0 \bmod 2^j$, *find* x.
- *given* $\exp_\alpha(x)$ *for random* x *with* $x < q2^{-j}$, *find* x.

Proof. The shift bits $\mathrm{lsb}(2^{k-i}x) \quad i = 0, \ldots, j-1$ coincide with the first j shift bits $\mathrm{lsb}(2^{-i}x') \quad i = 0, \ldots, j-1$ of $x' := 2^k x \bmod q$. We easily get $\exp_\alpha(x') := \exp_\alpha(x)^z$ with $z = 2^k \bmod q$ from $\exp_\alpha(x)$.

The case that $\mathrm{lsb}(2^{-i}x)$ for $i = 0, \ldots, j-1$ are given is by Proposition 2 equivalent to the case that the first j ls-bits of x are given.

In order to transform a random x with given $L_j(x)$ into a random x' with $x' = 0 \bmod 2^j$ replace the unknown x by $x' = x - L_j(x)$, and replace $\exp_\alpha(x) = \alpha^x$ by $\alpha^{x - L_j(x)}$.

We transform an unknown x with $x = 0 \bmod 2^j$ into x' with $x' < q2^j$ in that we replace $\exp_\alpha(x)$ by $\exp_\alpha(x)^z$ with $z = 2^{-j} \bmod q$. □

Theorem 5. *Arbitrary segments of j consecutive shift bits of random x are simultaneously secure when given $\exp_\alpha(x)$ iff the 2^{-j}-fractions of $\exp_\alpha$ are one-way.*

Proof. Due to Proposition 4 the particular location of the j consecutive shift bits of x does not matter. Moreover we can choose a particular 2^{-j}-fraction of $\exp_\alpha$.

If the 2^{-j}-fraction of $\exp_\alpha$ is not one-way then j consecutive shift bits of x cannot be simultaneously secure for given random $\exp_\alpha(x)$. This is because we can distinguish j consecutive shift bits of x from truly random bits by inverting the corresponding 2^{-j}-fraction of $\exp_\alpha$ i.e., we reconstruct x from $\exp_\alpha(x)$.

Now suppose that for given random $\exp_\alpha(x)$ the first j ls-bits of x are not simultaneously secure, i.e. we can distinguish in probabilistic polynomial time and with non-negligible advantage δ the initial segment $L_j(x)$ of x from a truly random $z \in_R [0, 2^j)$. (The advantage δ is non-negligible in the bit length n of q, $\delta \geq 1/\mathrm{poly}(n)$. By Proposition 2 the first shift bits and to the first ls-bits of x are equivalent.) By Yao's argument, see [K97, section 3.5, Lemma P1], there exists an integer j' with $0 \leq j' < j$ and a probabilistic polynomial time oracle $O_{j'}$ which predicts $\mathrm{ls}_{j'+1}(x)$ when given $L_{j'}(x)$, $\exp_\alpha(x)$:

$$\Pr_{x,w}[O_{j'}(L_{j'}(x), \exp_\alpha(x)) = \mathrm{ls}_{j'+1}(x)] \geq \tfrac{1}{2} + \varepsilon,$$

where the advantage ε is at least δ/j and the probability is taken over $x \in_R \mathbf{Z}_q$ and $O_{j'}$'s random coin flips.

How to invert $\exp_\alpha$ *when given* $L_{j'}(x)$. We invert the $2^{-j'}$-fraction of $\exp_\alpha$ corresponding to the given $L_{j'}(x)$ in probabilistic ploynomial time. A main task is to determine $\mathrm{ls}_{j'+1}(x)$.

Determining $\mathrm{ls}_{j'+1}(x)$. Pick random $x_i \in_R \mathbf{Z}_q$ for $i = 1, \ldots, m := 2n\varepsilon^{-2}$. For every i the equation

$$O_{j'}\left(L_{j'}(x + x_i), \exp_\alpha(x + x_i) \right) = \mathrm{ls}_{j'+1}(x + x_i) \tag{2}$$

holds with probability at least $\frac{1}{2} + \varepsilon$. Here we easily get $\exp_\alpha(x + x_i) = \exp_\alpha(x) \exp_\alpha(x_i)$. Moreover we have $L_{j'}(x + x_i) = L_{j'}(x) + L_{j'}(x_i) - \sigma L_{j'}(q)$, where σ is 1 if $[x]_q + [x_i]_q \geq q$ and $\sigma = 0$ otherwise. We show below how to get σ with error probability at most $\frac{\varepsilon}{2}$. Given σ and $\mathrm{ls}_{j'+1}(x + x_i)$ we get $\mathrm{ls}_{j'+1}(x)$ from the equations

$$\begin{aligned} L_{j'+1}(x + x_i) &= L_{j'}(x + x_i) + 2^{j'} \mathrm{ls}_{j'+1}(x + x_i) \\ &= L_{j'}(x) + L_{j'}(x_i) + 2^{j'} (\mathrm{ls}_{j'+1}(x) + \mathrm{ls}_{j'+1}(x_i)) - \sigma q \bmod 2^{j'+1}. \end{aligned} \tag{3}$$

As we get $\mathrm{ls}_{j'+1}(x + x_i)$ with advantage ε and σ has error probability $\frac{\varepsilon}{2}$ we get $\mathrm{ls}_{j'+1}(x)$ with advantage $\frac{\varepsilon}{2}$. We guess $\mathrm{ls}_{j'+1}(x)$ for each of the $m = 2n\varepsilon^{-2}$ independent x_i and we determine $\mathrm{ls}_{j'+1}(x)$ by majority decision over the m guesses. Consider the error probability of that decision.

As the x_i for $i = 1, ..., m$ are independent so are the guesses for $\mathrm{ls}_{j'+1}(x)$. Using Chernoff's bound for the deviation of mutually independent identically distributed random variables the error probability of the majority decision of $\mathrm{ls}_{j'+1}(x)$ is at most $\exp(-2m(\frac{\varepsilon}{2})^2) = \exp(-n) < \frac{1}{2n}$ for $n \geq 2$. Here we use a particular form of the Chernoff bound which is due to HOEFFDING [H63], see exercise 4.7 of [MR95]. Let X_i be the 0,1-error variable of the i-th prediction of $\mathrm{ls}_{j'+1}(x)$ based on Equations 2,3. The X_i are independent with mean value $\mu \leq \frac{1}{2} - \frac{1}{2}\varepsilon$. Then we have

Hoeffdings bound. $\Pr[\frac{1}{m}\sum_{i=1}^{m} X_i \geq \mu + \frac{1}{2}\varepsilon] \;\leq\; \exp(-2m(\frac{1}{2}\varepsilon)^2)$.

Finding σ. In order to find $\sigma =$ "$[x]_q + [x_i]_q \geq q$" we guess initially the $1 + \lg \varepsilon^{-1}$ first ms-bits of x. We try all $2\varepsilon^{-1}$ possible bit strings running the inversion procedure $2\varepsilon^{-1}$ times. The ms-bits of x determine an interval $I \subset [0, 1)$ of length $\frac{\varepsilon}{2}$ that contains x. The interval I and x_i determine σ except that $q - [x_i]_q \in I$. As x_i is random the except case has probability $\frac{\varepsilon}{2}$. Thus we get σ with error probability $\frac{\varepsilon}{2}$.

Iteration. Once we have found $\mathrm{ls}_{j'+1}(x)$ we replace the unknown $[x]_q = x$ by $x_{new} := \frac{1}{2}(\,[x]_q - \mathrm{lsb}(x)\,)$. For this we replace the corresponding $\exp_\alpha(x) = \alpha^x$ by $(\,\alpha^{x-\mathrm{lsb}(x)}\,)^z$ with $z := 2^{-1} \bmod q$ – note that we know $\mathrm{lsb}(x) = \mathrm{ls}_1(x)$ from $L_{j'}(x)$. We iterate the procedure to find $\mathrm{ls}_{j'}(x), ..., \mathrm{ls}_1(x)$. For each iteration we get $L_{j'}(x_{new})$ from $L_{j'}(x_{old})$ and $\mathrm{ls}_{j'+1}(x_{old})$, and we update the first $1 + \lg \varepsilon^{-1}$ ms-bits of $x_{new} = \frac{1}{2}(\,[x]_q - \mathrm{lsb}(x)\,)$; this is easy as we are given $\mathrm{lsb}(x) = \mathrm{ls}_1(x)$. Each iteration decreases the bit length of x. We are done after n iterations.

Time bounds. The time for the computation of x is $O(n\,m\,T + \varepsilon^{-1}) = O(n^2\varepsilon^{-2}T) = O(n^2 j^2 \delta^{-2} T)$, where T is the time of oracle $O_{j'}$. (Guessing the $1 + \lg \varepsilon^{-1}$ first ms-bits of x requires $O(\varepsilon^{-1})$ steps. As the calls of oracle $O_{j'}$ do not depend on these ms-bits the $O(\varepsilon^{-1})$ workload only adds to the overall workload.) The probability of success of the computation is at least $\frac{1}{2}$ as each iteration fails with probability at most $\frac{1}{2n}$. □

Security of individual ls-bits. Are individual bits $\mathrm{ls}_j(x)$ secure when given $\exp_\alpha(x)$? By Proposition 4 $\mathrm{ls}_j(x)$ is at least as secure as an arbitrary sequence of j consecutive shift bits. By Theorem 5 $\mathrm{ls}_j(x)$ is secure if the 2^{-j}-fractions of $\exp_\alpha$ are one-way. This one-wayness is problematic for large j.

HÅSTAD, NÄSLUND [HN98] give a direct method to prove security for individual bits $\mathrm{ls}_j(x)$. They present the method for the RSA-function $E_N(x)$ but the method works as well for the exponentiation function $\exp_\alpha$. The HN-method requires that we can in poly-time transform $\exp_\alpha(x)$ into its square root $\sqrt{\exp_\alpha(x)}$, and into powers $\exp_\alpha(ax)$ for arbitrary integers a. If the group order q is odd these transformations are in fact poly-time. This is because $\sqrt{\exp_\alpha(x)} = \exp_\alpha(x)^{\frac{1}{2} \bmod q}$ and 2 is invertible modulo q. Therefore [HN98] implies that all individual bits $\mathrm{ls}_j(x)$ are secure [1] when given $\exp_\alpha(x)$ iff $\exp_\alpha$ is one-way.

[1] We disregard "trivial" advantage in distinguishing a bit due to bias.

4 Simultaneous security of discrete log bits, even group order

Let G be a cyclic group of order $2^s q$ with an odd integer q and an $s \geq 1$. Let α be a generator of G. It is well known that the first s ls-bits of x can easily be obtained from $\exp_\alpha(x)$. We show that the next j bits $\mathrm{ls}_{s+1}(x), ..., \mathrm{ls}_{s+j}(x)$ are secure when given $\exp_\alpha(x)$ iff the 2^{-j}-fractions of $\exp_{\alpha'}$ with $\alpha' := \alpha^{2^s}$ are one-way. Note that α' generates the subgroup $G' \subset G$ of 2^s-powers of G. The claim for G follows from that for G' proven in Theorem 5. Moreover the bit strings $\mathrm{ls}_{s+1}(x), ..., \mathrm{ls}_{s+j}(x)$ and $\mathrm{ms}_1(x), ..., \mathrm{ms}_j(x)$ are equally secure when given $\exp_\alpha(x)$.

Computing the first s of the ls-bits of x. Given the group order, the generator α and $\alpha^x = \exp_\alpha(x)$ we easily get the first s ls-bits of x, i.e. we get $L_s(x) = [x]_{2^s} = x \bmod 2^s$. We have $\mathrm{ls}_1(x) = 0$ iff α^x is a square in G, i.e. iff $\alpha^{x\, 2^{s-1} q} = 1_G$. Continuing recursively we see for $i \leq s$ that $\mathrm{ls}_i(x) = 0$ iff $(\alpha^{x - L_{i-1}(x)})^{2^{s-i} q} = 1_G$.

Reduction to odd group order. Let $G' \subset G$ be the subgroup of all 2^s-powers of G. This subgroup has odd order q and generator $\alpha' := \alpha^{2^s}$. Given $y = \exp_\alpha(x) \in G$ we get $L_s(x)$ and $y' := y / \exp_\alpha(L_s(x)) \in G'$ in poly-time. We have

$$\lfloor \log_\alpha(y)/2^s \rfloor = \log_{\alpha'}(y'), \quad \mathrm{ls}_{s+i}(\log_\alpha(y) = \mathrm{ls}_i(\log_{\alpha'}(y')) \text{ for all } i \geq 1. \tag{4}$$

By Theorem 5 this reduction yields:

Theorem 6. *The bits* $\mathrm{ls}_{s+1}(x), ..., \mathrm{ls}_{s+j}(x)$ *are simultaneously secure when given* $\exp_\alpha(x)$ *iff the* 2^{-j}*-fractions of* $\exp_{\alpha'}$ *are one-way.*

Square roots and principal square roots. Let $\exp_\alpha(x) \in G$ be a square, i.e., $\mathrm{ls}_1(x) = 0$. As G has even order $2^s q$ there are two square roots $\pm \exp_\alpha([x]_{2^s q}/2)$. Here we let $1 \in G$ denote the neutral element and let $-1 \in G$ be the square root of 1 other than 1. It is well known that given a generator α square roots can be computed in polynomial time.

We call $\exp_\alpha([x]_{2^s q}/2)$ the *principal* square root of $\exp_\alpha(x)$. By definition the discrete log of the principal square root of $y = \exp_\alpha(x)$ is half the discrete log of y. The two square roots of y differ by the factor $-1 = \alpha^{2^{s-1} q}$, if $\pm y'$ are the two square roots of y then $|\log_\alpha(y') - \log_\alpha(-y')| = 2^{s-1} q$. As q is odd $\log_\alpha(\pm y')$ differ in the ls_s-bit. The equality of that bit with $\mathrm{ls}_s(\log_\alpha(y))$ characterizes the principal square root y' of y, we have :

Lemma 7. 1. *Let $y \in G$ be a square with square roots $\pm y'$ then y' is the principal square root of y iff* $\mathrm{ls}_s(\log_\alpha y') = \mathrm{ls}_{s+1}(\log_\alpha y)$.
2. *Let y' be a random square root of a random $y \in_R G$. Deciding with advantage ε whether y' is principal for the given y is equivalent to predicting* $\mathrm{ls}_{s+1}(\log_\alpha y)$ *with advantage ε.*

The complexity of deciding the principal square root. Theorem 6 and Lemma 7 characterize the complexity of deciding whether a random square root of a random square in G is a principal. Deciding principality with a non-negligible advantage is as hard as inverting $\exp_\alpha$ in prob. poly-time, a result which is due to Blum, Micali [BM84]. By Theorem 6 with $j = 1$ the bit $\mathrm{ls}_{s+1}(x)$ is secure for given $\exp_\alpha(x)$ or else the 2^{-s-1}-fractions of $\exp_\alpha$ can be inverted in probabilistic poly-time. As we easily get $L_s(x)$ from $\exp_\alpha(x)$ this means that $\mathrm{ls}_{s+1}(x)$ is secure provided that $\exp_\alpha$ is one-way. By Lemma 7 the problem to decide principality of a square root of y is equivalent to predicting $\mathrm{ls}_{s+1}(\log_\alpha y)$, so we get the [BM84] result.

We next extend the BM-result proving simultaneous security of the first j ms-bits of x. Clearly there is a proof similar to that of Theorem 5, the difference is that in the iteration we multiply x by 2 instead of using division by 2.

Lemma 8. *The bit strings* $\mathrm{ls}_{s+1}(2^j x), ..., \mathrm{ls}_{s+j}(2^j x)$ *and* $\mathrm{ms}_1(x), ..., \mathrm{ms}_j(x)$ *are equivalent by polynomial time transformations when given* $\exp_\alpha(x)$.

Proof. The equation $[2x]_{2^s q} = 2[x]_{2^s q} - \mathrm{ms}_1(x) 2^s q$ yields by induction on j

$$[2^j x]_{2^s q} = 2^j [x]_{2^s q} - (\textstyle\sum_{i=1}^{j} \mathrm{ms}_i(x) 2^{j-i})\, 2^s q, \text{ and thus}$$

$$L_{s+j}(2^j x) = 2^j L_s(x) - (\textstyle\sum_{i=1}^{j} \mathrm{ms}_i(x) 2^{j-i})\, 2^s q \bmod 2^{s+j}. \tag{5}$$

We get $\exp_\alpha(2^j x)$, $L_s(x)$ and $L_s(2^j x)$ in poly-time from $\exp_\alpha(x)$. Given $L_s(x)$ and $L_s(2^j x)$, $\sum_{i=1}^{j} \mathrm{ls}_{s+i}(2^j x) 2^{s+i}$ and $(\sum_{i=1}^{j} \mathrm{ms}_i(x) 2^{j-i})\, 2^s q \bmod 2^{s+j}$ are equivalent by Equation 5. Here we use that q is odd, so we can invert q modulo 2^{s+j}. □

Theorem 9. *The bits* $ms_1(x), ..., \mathrm{ms}_j(x)$ *are simultaneously secure when given* $\exp_\alpha(x)$ *iff the* 2^{-j}*-fractions of* $\exp_{\alpha'}$ *are one-way.*

Proof. Suppose that the bit string $\mathrm{ms}_1(x), ..., \mathrm{ms}_j(x)$ is poly-time distinguishable from random $z \in_R \{0,1\}^j$ when given $\exp_\alpha(x) \in_R G$. We show how to distinguish via Lemma 8 the bit string $\mathrm{ls}_{s+1}(2^j x), ..., \mathrm{ls}_{s+j}(2^j x)$ from random z.

Given $\exp_\alpha(2^j x) \mathrm{ls}_{s+1}(2^j x), ..., \mathrm{ls}_{s+j}(2^j x)$ we get a random 2^j-root $\exp_\alpha(x)$, and via Equation 5 we get the corresponding bit string $\mathrm{ms}_1(x), ..., \mathrm{ms}_j(x)$ in poly-time. Thus using the distinguishing algorithm for the ms-bits we can also distinguish the ls-bits from random z. Therefore by Theorem 6 the 2^{-j}-fractions of $\exp_{\alpha'}$ cannot be one-way. This proves one direction of the claim, the converse is obvious. □

Equal security of the ms- and the ls-bits. In particular Lemma 8 shows that the security results of [P85] and those of [LW88] are equivalent. This equivalence of [P85] and [LW88] is not apparent from these papers.

Security of individual ls-bits. The [HN98]-method of proving security of individual bits $\mathrm{ls}_j(x)$ when given $y = \exp_\alpha(x)$ cannot be directly applied to even order groups G because there is no poly-time algorithm for computing square roots. However Equation 4 shows that the bits $\mathrm{ls}_{s+j}(\log_\alpha(y))$ and $\mathrm{ls}_j(\log_{\alpha'}(y'))$ coincide. Also $y' := y / \exp_\alpha(L_s(x))$ is random in G' if y is random in G. By [HN98] the individual bits $\mathrm{ls}_j(\log_{\alpha'}(y'))$ are secure when given $y' \in_R G'$. We conclude that the bits $\mathrm{ls}_{s+j}(\log_\alpha(y))$ are individually secure when given $y' \in_R G'$. Now y and y' only differ by $\alpha^{L_s(x)}$ which is clearly independent of $\mathrm{ls}_{s+j}(\log_\alpha(y))$. As $\exp_\alpha$ and $\exp_{\alpha'}$ are equally one-way this proves

Theorem 10. *The individual bits* $\mathrm{ls}_{s+j}(x), \mathrm{ms}_j(x)$ *for* $j \geq 1$ *are secure when given* $\exp_\alpha(x)$ *iff* $\exp_\alpha$ *is one-way.*

5 Generic networks, one-wayness of fractions of $\exp_\alpha$

The important question is whether the exponentiation function and its fractions are one-way. No complexity lower bound is known for the discrete logarithm for Turing machines or Boolean networks. But for generic algorithms NECHAEV [Ne94] and SHOUP [Sh97] have shown an exponential lower bound. The significance of this lower bound is that important classes of discrete log algorithms are generic. The known algorithms for the discrete logarithm in general groups, specifically elliptic curves, are all generic. The number field sieve and the quadratic sieve are non-generic but they only apply to particular groups. For this section let G be a group of prime order q.

We establish a complexity lower bound for generic networks computing the discrete logarithm of a 2^{-j}-fraction of the exponentiation function. Every generic network that computes $\log_\alpha$ for a 2^{-j}-fraction of $\exp_\alpha$ has to perform at least $\sqrt{q\,2^{-j+1}} - O(1)$ group operations. We conclude that almost all bits of the discrete logarithm are simultaneously secure against generic attacks.

Generic algorithms/networks. The idea of a generic algorithms for the computation of discrete logarithms of cyclic groups goes back to NECHAEV [Ne94]. A full model of generic algorithms has been presented by SHOUP [Sh97]. We extend these models. Generic networks perform a straight-line computation with unbounded fan-in and fan-out, whereas the NECHAEV merely uses trees. The non-uniform model is more powerful, similarly as Boolean networks are more powerful than Turing machines. Our *generic steps* are general multivariate exponentiations, while [Ne94], [Sh97] merely use multiplication/division in groups. Unlike [Sh97] we distinguish between the generic group operations and the non-generic steps without using a random encoding for the group elements. Our probabilities do not depend on such a random encoding.

Definition of generic networks. A generic network has two types of inputs, *group inputs* and *auxiliary inputs*. Possible group inputs are public parameters as the generator α, the unit element $1_G \in G$ and particular group elements. The actual inputs are random group elements, e.g. random $y = \exp_\alpha(x) \in G$. Possible auxiliary inputs are the bit length q of the group order, the group order $|G|$, the prime factor decomposition of the group order and so on. Via the auxiliary inputs we get algorithms/networks that are generic for classes of groups that are defined by additional knowledge on G, the order of G and so on. This largely extends the [Sh97] model, where the group G is fixed.

The computation consists of *generic steps* that perform arbitrary *multivariate exponentiations* $\quad \mathrm{mex}_{\mathbf{a}} : G^d \to G, \quad (g_1, ..., g_d) \mapsto g_1^{a_1} \cdot ... \cdot g_d^{a_d}$ with given $\mathbf{a} = (a_1, ..., a_d) \in \mathbf{Z}^d$ and arbitrary, unbounded $d \in \mathbf{N}$.

The *ν-th generic step* of the network either computes an input group element or it performs a group operation $\mathrm{mex}_{\mathbf{a}}$, $\mathbf{a} = (a_1, ..., a_{\nu-1})$, using the previously computed group elements $F_1, ..., F_{\nu-1}$. The result of the ν-th step is $F_\nu := F_1^{a_1} \cdot ... \cdot F_{\nu-1}^{a_{\nu-1}}$. The exponents $a_1, ..., a_{\nu-1} \in \mathbf{Z}$ are determined by – and depend arbitrarily on

- the round number ν
- the set $\mathcal{CO}_{\nu-1} =_{def} \{(i,j) \mid F_i = F_j, 1 \le i < j \le \nu - 1\}$ of previous *collisions*

- the auxiliary inputs.

The *output* of the network is an arbitrary bit string or integer that is determined by – and depends arbitrarily on – the set of all collisions $\mathcal{CO}_t$ and the auxiliary inputs.

Generic steps $\text{mex}_{\mathbf{a}}$ are counted at unit costs, the other operations, arbitrary functions of the auxiliary inputs and equality tests for group elements, are for free. The *length* t of a generic algorithm is the number of input group elements plus the number of the generic steps $\text{mex}_{\mathbf{a}}$.

All probabilities refer to the random input (but not to the random encodings of the group elements as in [Sh97]). Generic networks do not need internal coin flips as we can fix an optimal coin flip due to non-uniformity. There are no oracles for the group operations as in [Sh97]. Instead, only a generic step accesses the group elements for group operations and equality tests. The next theorem extends NECHAEV's lower bound to fractions of the exponentiation function and to generic networks.

Theorem 11. *Every generic network $\mathcal{A}$ of length t which inverts $\exp_\alpha$ for a 2^{-j}-fraction of $\exp_\alpha$ succeeds for at most a $(\binom{t}{2}+1)2^j/q$-fraction of the arguments.*

Proof. Let F_1 denote the input $y = \exp_\alpha(x)$ and $F_2, F_3, ..., F_t$ the results of the group operations of $\mathcal{A}$. The ν-th step of $\mathcal{A}$, its group operation $\text{mex}_{\mathbf{a}}$ with exponents $a_1, ..., a_{\nu-1}$, only depends on the set $\mathcal{CO}_{\nu-1} = \{(i,j) \mid F_i = F_j,\ i < j \le \nu - 1\}$ of previous collisions (and the auxiliary inputs). $\mathcal{A}$'s output $x_{\text{out}} \in \mathbf{Z}_q$ depends only on $\mathcal{CO}_t$. The probability calculation below shows that, except with probability $\binom{t}{2}2^j/q$, $\mathcal{CO}_t$ is constant, i.e. independent of the input y. If x_{out} does not depend on y then it is correct with probability at most $2^j/q$ as y ranges uniformly over a set of size $q2^{-j}$. Hence $\mathcal{A}$'s probability of success is at most $(\binom{t}{2}+1)2^j/q$.

Probability calculation. We assume w.l.o.g. that there are no collisions $F_i = F_j$, $i \neq j$, that do not depend on the input $y$as such collisions are useless and are easy to eliminate from $\mathcal{A}$. By the assumption we have $\mathcal{CO}_\nu = \emptyset$ or else $\mathcal{CO}_\nu$ depends on y. Next we show that

$$\Pr_y[\, F_i = F_\nu,\ \mathcal{CO}_{\nu-1} = \emptyset] \ \le\ 2^j/q \quad \text{for } i = 1, \dots, \nu - 1.$$

If $F_i = F_\nu$ then by the assumption the group element $F_i\, F_\nu^{-1}$ depends on the input y. As $F_i\, F_\nu^{-1}$ results from a multivariate exponentiation depending on y it *permutes* G when y ranges over G. (A multivariate exponentiation acts as a permutation on G if all except one input are fixed. Here we use that G has prime order q.) It is assumed that y ranges randomly over a subset of G of cardinality $q\,2^{-j}$, e.g. over $\{\exp_\alpha(x) \mid \text{with } x = 0 \bmod 2^j\}$. Hence $\Pr_y[\, F_i\, F_\nu^{-1} = 1_G \mid \mathcal{CO}_{\nu-1} = \emptyset\,] \le 2^j/q$. We finally get

$$\Pr_y[\,\mathcal{CO}_t \neq \emptyset\,] \ \le\ \sum_{\nu=1}^{t}\sum_{i=1}^{\nu-1} \Pr_y[\, F_i = F_\nu, \mathcal{CO}_{\nu-1} = \emptyset\,] \ \le\ \binom{t}{2}\, 2^j/q. \qquad \square$$

Conclusions. By Theorem 11 a generic algorithm for $\log_\alpha$ that succeeds for a 2^{-j}-fraction of $\exp_\alpha$ must have length $t \ge \sqrt{q2^{-j+1}-2}$. This lower bound is tight up to a factor 2. By the SHANKS baby step giant step method we can compute discrete logarithms of 2^{-j}-fractions of $\exp_\alpha$ using $O(\lg(q\,2^{-j})\sqrt{q\,2^{-j}}\,)$ Turing steps. This algorithm is, essentially, generic. It yields a generic algorithm of length $2\lfloor\sqrt{q\,2^{-j}}\rfloor$ for the discrete logarithm of a

2^{-j}-fraction of $\exp_\alpha$. (The complexity decreases from counting Turing steps to counting generic steps. We get the intersection of two sets of group elements at zero generic costs as equality tests are for free.)

Corollary 12. *The minimal length t of generic networks that invert a 2^{-j}-fraction of* $\exp_\alpha$ *is* $\Theta(\sqrt{q 2^{-j}})$.

Theorem 13. *Every generic network $\mathcal{A}$ of length t with input $y = \exp_\alpha(x) \in G$ distinguishes $L_j(x)$ and random $z \in_R [0, 2^j)$ at most with advantage*

$$\delta := |\ \Pr_y[\ \mathcal{A}(L_j(x), \exp_\alpha(x)\,) = 1] - \Pr_{y,z}[\ \mathcal{A}(z, \exp_\alpha(x)\,) = 1]\ | \le O(\, n\, j \sqrt{t}\, (2^j/q)^{\frac{1}{4}}\,).$$

Proof. The given generic network $\mathcal{A}$ of length t and advantage δ yields by Yao's argument [K97, section 3.5, Lemma P1] for some $j' < j$ a generic prediction algorithm $O_{j'}$ of length t which, for given $L_{j'}(x)$ and $\exp_\alpha(x)$, predicts $\mathrm{ls}_{j'+1}(x)$ with advantage $\varepsilon \ge \delta/j$. By Proposition 2 $L_{j'}(x)$ is equivalent to the first j' shift bits of x. Theorem 5 yields a generic algorithm for the inversion of the 2^{-j}-fraction of $\exp_\alpha$ corresponding to the known $L_j(x)$ which uses oracle $O_{j'}$ as subroutine with t generic steps. Each iteration of the inversion algorithm of Theorem 5 performs an additional generic step to transform $\exp_\alpha(x)$ into $\exp_\alpha(x_{new})$. Each oracle call $O_{j'}\left(L_{j'}(x + x_i), \exp_\alpha(x + x_i)\right)$ requires one further generic step to compute $E_N(x + x_i)$. So we get a generic inversion algorithm of length $O(n^2 \delta^{-2} j^2 t)$. By Corollary 12 we must have $O(n^2\delta^{-2}j^2t) = \Omega(\sqrt{q\, 2^{-j}})$ hence $\delta = O(n\, j \sqrt{t}\, (2^j/q)^{\frac{1}{4}})$. $\square$

Conclusions. Given random $\exp_\alpha(x)$, $L_j(x)$ is generically indistinguishable from random $z \in_R [0, 2^j)$ provided that $j < (1 - \beta) \lg q$ for fixed $\beta > 0$. This is because such j satisfies $2^j/q < q^\beta$, and thus the advantage of Theorem 13 becomes negligible for $t \le \mathrm{poly}(n)$. Hence, all except an arbitrarily small β-fraction of the bits of x are simultaneously secure against generic attacks. Note that β can converge to 0 as q increases, it is sufficient that β is large enough so that $\lim_{q\to\infty} \frac{\beta \lg q}{\lg\lg q} = \infty$. This result is nearly optimal since no fraction of $1 - O(\frac{\lg\lg q}{\lg q})$ bits of x can be simultaneously secure, because the remaining bits can be guessed in polynomial time $2^{O(\lg\lg q)} = (\lg q)^{O(1)}$.

Corollary 14. *For groups G of prime order q, almost all bits of the discrete log of random $y \in_R G$ are simultaneously secure against generic attacks.*

References

[BM84] M. Blum and S. Micali: How to Generate Cryptographically Strong Sequences of Pseudo-random Bits. Siam J. Comp. 13, (1984), pp. 850-864.

[GL89] O. Goldreich and L.A. Levin: Hard Core Bit for any One Way Function. Proc. of ACM Symp. on Theory of Computing (1989) pp. 25-32.

[HN98] J. Håstad and M. Näslund: The Security of Individual RSA Bits. Proc. of IEEE Symp. on Foundations of Computer Science (1998).

[HSS93] J. Håstad, A.W. Schrift and A. Shamir: The Discrete Logarithm Modulo a Composite Hides $O(n)$ bits. J. of Computer and Systems Sciences 47 (1993), pp. 376-404.

[H63] W. Hoeffding: Probability in Equalities for Sums of Bounded Random Variables. J. Amer. Stat. Ass. 58 (1963), pp. 13-30.

[K97] D.E. Knuth: Seminumerical Algorithms, 3rd edition, Addison-Wesley, Reading, MA (1997).

[LW88] D.L. Long and A. Wigderson: The Discrete Logarithm Hides $O(\log n)$ bits. Siam J. Computing 7 (1988), pp. 363-372.

[Ne94] V.I. Nechaev: Complexity of a Determinate Algorithm for the Discrete Logarithm. Mathematical Notes 55 (1994), pp. 165-172.

[MR95] R. Motwani and P. Raghavan: Randomized Algorithms. Cambridge University Press Cambridge UK, 1995.

[N94] *NIST:* "Digital Signature Standard (DSS), Federal Information Processing Standard" PuB 186, 1994 May 19.

[P85] R. Peralta: Simultaneous Security of Bits in the Discrete Log. Proceedings Eurocrypt'85, Springer LNCS 219 (1986), pp. 62-72.

[R79] M.O. Rabin: Digital Signatures and Public Key Functions as Intractable as Factorization. TM-212, Laboratory of Computer Science, MIT, 1979.

[RSA78] R.L. Rivest, A. Shamir and L. Adleman: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Comm. ACM, 21 (1978), pp. 120-126.

[S91] C.P. Schnorr: Efficient Signature Generation for Smart Cards. Journal of Cryptology 4 (1991), pp. 161-174.

[Sh97] V. Shoup: Lower Bounds for Discrete Logarithms and Related Problems. Proc. Eurocrypt'97, LNCS 1233 (1997), Springer Berlin, pp. 256-266.

[VV84] U.V. Vazirani and V.V. Vazirani: Efficient and Secure Pseudo-Random Number Generation. In Proc. 25th Symp. on Foundations of Computing Science (1984) IEEE, pp. 458-463.

[Y82] A.C. Yao: Theory and Application of Trapdoor Functions. Proc. of IEEE Symp. on Foundations of Computer Science (1982), pp. 80-91.