State Recovery Attacks on Pseudorandom Generators

Andrey Sidorenko and Berry Schoenmakers

Eindhoven University of Technology
Eindhoven, The Netherlands
{a.sidorenko, l.a.m.schoenmakers}@tue.nl

Abstract: State recovery attacks comprise an important class of attacks on pseudorandom generators. In this paper we analyze resistance of pseudorandom generators against these attacks in terms of concrete security. We show that security of the Blum-Micali pseudorandom generator against state recovery attacks is tightly related to the security of the corresponding one-way function.

C. Wolf, S. Lucks, P.-W. Yau (Eds.): WEWoRC 2005, LNI P-74, pp. 53–63, 2005.

1 Introduction

One of the most fundamental issues in modern cryptology is randomness. Randomness is used for probabilistic encryption and digital signature algorithms. Nonces and session keys in cryptographic protocols must be random. However in practice true randomness is hard (if not impossible) to achieve. A possible solution to this problem is proposed by the theory of cryptographically secure pseudorandom generators that starts with a seminal paper of Yao [Ya82].

A pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length n, outputs a binary sequence of length $M > n$ that looks random for all efficient applications. The input to the generator is called the seed and the output is called the pseudorandom sequence. Security of a pseudorandom generator is a characteristic that shows how hard it is to tell the difference between the pseudorandom sequences and truly random sequences. A pseudorandom generator is said to be provably secure if distinguishing these two classes of sequences is as difficult as solving a well-known and supposedly hard (typically number-theoretic) problem [Ya82]. For instance, the Blum-Blum-Shub pseudorandom generator [BBS86] is secure under the assumption that factoring large Blum integers is a difficult problem. For the pseudorandom generator proposed by Kaliski [Ka88] the corresponding assumption is intractability of the elliptic curve discrete logarithm.

In order to prove security of a pseudorandom generator one has to show that an algorithm $\mathcal{D}$ that distinguishes the pseudorandom sequences from truly random sequences can be

reduced to an algorithm $\mathcal{S}$ that solves the corresponding difficult problem. The security reduction is said to be tight if running time and success probability of these two algorithms are about the same [BR96].

The pseudorandom generators mentioned above are based on the Blum-Micali (BM) construction [BM84]. The BM pseudorandom generator is proved to be asymptotically secure (see e.g. [Go01]). Namely, it is shown that there exists a polynomial-time reduction from the distinguisher $\mathcal{D}$ to the solver $\mathcal{S}$. In fact asymptotic security implies that no polynomial-time distinguishing attack exists. However this asymptotic statement says little about the security of the pseudorandom generator in practice for a particular seed length and against adversaries investing a specific amount of effort.

In practice it is important to know the concrete seed length that guarantees a certain level of security. We need for concrete analysis a reduction that is as efficient as possible. A tight reduction gives rise to a small modulus, which in turn ensures that the pseudorandom generator is efficient. To our knowledge, there exists no pseudorandom generator with tight security reduction. The reductions [Al88, BM84, FS00, Ka88, VV84] are polynomial-time but they are not tight. Moreover the reduction proposed by Fischlin et al. [FS00] for the Blum-Blum-Shub generator is shown to be optimal. It implies that no tight security reduction is possible for this pseudorandom generator. Inefficiency of security reductions explains the major disadvantage of provably secure pseudorandom generators, namely they are slow.

A way to improve tightness of the security reductions is to weaken Yao's definition of security for pseudorandom generators. The fact that there exists no efficient distinguishing attack on a pseudorandom generator means that every bit of the pseudorandom sequence is unpredictable. Is it necessary to demand such a strong property of pseudorandom generators? Our work raises an interesting open problem: is it possible to weaken the definition of security for pseudorandom generators in a reasonable way so that there exists a pseudorandom generator with tight security reduction?

In this paper we analyze security of pseudorandom generators against a subclass of distinguishing attacks, namely state recovery attacks. A state recovery attack $\mathcal{A}$ on a pseudorandom generator is an algorithm that, given a pseudorandom sequence, recovers the seed. We show that in case of the BM pseudorandom generators there exists a tight security reduction from the state recovery attack $\mathcal{A}$ to the solver $\mathcal{S}$.

The rest of the paper is organized as follows. In Section 2 we recall the definition of security for pseudorandom generators proposed by Yao [Ya82]. We consider state recovery attacks versus distinguishing attacks in Section 2.2. In Section 2.3 we give a simple example of an efficient state recovery attack. Section 3 contains the main result of the paper. In this section we prove robustness of BM generators against state recovery attacks. Two important examples, namely the Blum-Blum-Shub generator and the Kaliski generator, are discussed in Sections 3.1 and 3.2 respectively. In Section 4 we discuss the importance of tight security reductions. We consider the BBS generator as an example. Section 5 concludes the paper.

2 Security of Pseudorandom Generators

2.1 Distinguishing Attacks

We refer to a sequence of independent uniformly distributed bits as a truly random binary sequence. Informally speaking, a pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length n, outputs a binary sequence of length $M > n$ that looks random.

Let G be some pseudorandom generator that outputs binary sequences of length M. Let S_M be the set of the output sequences. Throughout this paper the notation "$\in_R$" indicates the process of selecting an element at random and uniformly over the corresponding set. The running time of a probabilistic algorithm is defined to be the maximum of the expected number of steps needed to produce an output, maximized over all inputs; the expected number is averaged over all coin flips made by the algorithm [Kn97].

The following definitions are due to [Ya82].

DEFINITION 2.1 *Let $\mathcal{D} : \{0,1\}^M \to \{0,1\}$ be a probabilistic algorithm that runs in time T. Let $\epsilon > 0$. $\mathcal{D}$ is called a (T, ϵ)-distinguishing attack on pseudorandom generator G if*

$$|\Pr[\mathcal{D}(s) = 1 \mid s \in_R S_M] - \Pr[\mathcal{D}(s) = 1 \mid s \in_R \{0,1\}^M]| \geq \epsilon,$$

where the probability is also taken over the internal coin flips of $\mathcal{D}$.

DEFINITION 2.2 *A pseudorandom generator is (T, ϵ)-secure if there exists no (T, ϵ)-distinguishing attack on this pseudorandom generator.*

Note that the original definition of cryptographically secure pseudorandom generator [Ya82] is given in the asymptotic sense. It states that a pseudorandom generator is secure if for any polynomial p no $(p(n), 1/p(n))$-distinguishing attack exists. However, for the concrete security analysis it is important to consider (T, ϵ)-distinguishing attacks for exact values of T and ϵ. The concrete version of the definition of cryptographically secure pseudorandom generator first appears in [Kn97].

In general the distinguishing attack $\mathcal{D}$ does not have to determine the seed in order to distinguish a pseudorandom sequence from a random sequence. For example, if the pseudorandom sequence is unbalanced $\mathcal{D}$ can just output the majority bit of the sequence.

2.2 State Recovery Attacks

Let G be a pseudorandom generator that outputs binary sequences of length M. Let S_M be the set of the output sequences of G.

DEFINITION 2.3 *Suppose there exists an algorithm $\mathcal{A} : \{0,1\}^M \to \{0,1\}^n$ that, given $s \in_R S_M$, outputs $x \in \{0,1\}^n$ such that $G(x) = s$ with probability ϵ and outputs $\emptyset$ with*

probability $1-\epsilon$*. Here the probability is taken over all choices of* s *and over internal coin flips of* $\mathcal{A}$*. Assume that* $\mathcal{A}(s) = \emptyset$ *for* $s \notin S_M$*. Let* T *be the running time of* $\mathcal{A}$*. Then* $\mathcal{A}$ *is called a* (T, ϵ)*-state recovery attack on pseudorandom generator* G*.*

Actually a state recovery attack interprets a pseudorandom generator as a one-way function $\{0,1\}^n \to \{0,1\}^M$. The more secure the one-way function the more time the state recovery attack takes. The following lemma implies that the class of state recovery attacks is a subclass of distinguishing attacks.

Lemma 2.4 *Any* (T, ϵ)*-state recovery attack is a* $(T, \epsilon - \epsilon 2^{n-M})$*-distinguishing attack.*

PROOF. The state recovery attack $\mathcal{A}$ can be transformed into a distinguishing attack $\mathcal{D}$ as follows. On input $s \in \{0,1\}^m$ set $\mathcal{D}(s) = 0$ if $\mathcal{A}(s) = \emptyset$, otherwise set $\mathcal{D}(s) = 1$. Then $\Pr[\mathcal{D}(s) = 1 \mid s \in_R S_M] = \epsilon$ and $\Pr[\mathcal{D}(s) = 1 \mid s \in_R \{0,1\}^M] = \epsilon 2^{n-M}$, so $\mathcal{D}$ is a $(T, \epsilon - \epsilon 2^{n-M})$-distinguishing attack. □

Pseudorandom generator is a central building block of an important cryptographic primitive, namely additive stream cipher. The pseudorandom generators used for the practical stream ciphers are fast but not provably secure. Therefore new attacks on the stream ciphers are published frequently. Ekdahl [Ek03] proposes an efficient state recovery attack against the Bluetooth standard and against the stream cipher A5/1, which is used in GSM standard for mobile telephones. A series of papers [Kn98, Ma05] analyze state recovery attacks on RC4 Keystream Generator.

The main result of this paper is that the security of the Blum-Micali pseudorandom generators [BM84] against state recovery attacks is *tightly* related to the security of the corresponding one-way function. Thus security against state recovery attacks is guaranteed for much smaller seed lengths. The new reduction is discussed in Section 3. Before presenting the main result, we discuss a simple example of an efficient state recovery attack for linear congruential generators.

2.3 Example of an Efficient State Recovery Attack

It is well-known that for any LFSR-based pseudorandom generator the initial state can be found using the Berlekamp-Massey algorithm [Ma69]. Similar problems – but maybe less well-known – concern the popular linear congruential generators. Even when all the details of the generator are kept secret, a state recovery attack is quite easy to implement.

The linear congruential generator works as follows. Let A be a positive integer of bit length a for some $a > 0$. Let $j, k, s_1 \in \mathbb{Z}_A$. The seed of the linear congruential generator (LCG) is a quadruple (A, j, k, s_1). The length of the seed $n = 4a$. The pseudorandom sequence $s = \{s_1, s_2, \ldots, s_B\}$ is generated as follows:

$$s_i = js_{i-1} + k \bmod A,\ i = 2, 3, \ldots, B.$$

The pseudorandom sequence is a sequence of integers $s_i \in \mathbb{Z}_A,\ i = 1, 2, \ldots, B$. s_1 is both a component of the seed and the first element of the pseudorandom sequence. The length of the pseudorandom sequence is $M = aB$.

Although the LCG works well for a number of Monte Carlo problems it is not suitable for cryptographic purposes. The reason is that it does not resist a state recovery attack.

Marsaglia [Ma68] shows that the LCG has a defect that cannot be removed by adjusting the seed. Namely, if l-tuples $(s_1, s_2, \ldots, s_l), (s_2, s_3, \ldots, s_{l+1}), \ldots, l \geq 3$, produced by the LCG are viewed as points in the unit cube of n dimensions, then all the points will be found to lie in a relatively small number of parallel hyperplanes. This is the intuition behind the following lemma proposed by Marsaglia [Ma68].

Lemma 2.5 *For all i, $i = 1, 2, \ldots, B-3$, Δ_i is a multiple of A, where*

$$\Delta_i = \det \begin{bmatrix} s_i & s_{i+1} & 1 \\ s_{i+1} & s_{i+2} & 1 \\ s_{i+2} & s_{i+3} & 1 \end{bmatrix}.$$

Suppose an adversary knows a few first s_i's. Greatest common divisor of several Δ_i's gives the value of A. Then it can compute j and k by solving the following system of linear equations

$$\begin{cases} js_1 + k = s_2 \bmod A \\ js_2 + k = s_3 \bmod A \end{cases}$$

As a result the seed is revealed.

The above argument shows that there exists an efficient state recovery attack on the LCG. To run this attack, the adversary does not have to know many output values s_i. In practice 5 values $s_1, \ldots, s_5$ are enough.

3 The Blum-Micali Construction

A family of cryptographically strong pseudorandom generators is proposed by Blum and Micali [BM84]. Let f be a one-way permutation defined over some domain D. Let b be a hard-core predicate for f. The seed x_1 of the pseudorandom generator is a uniformly distributed element of D. The pseudorandom sequence (the BM sequence) $s \in \{0,1\}^M$ is generated as follows:

$$s_i = b(x_i),\ x_{i+1} = f(x_i)$$

for $i = 1, \ldots, M$.

If an adversary does not know the seed it cannot distinguish the BM sequence from a random sequence in polynomial-time with non-negligible advantage [BM84]. Therefore due to Lemma 2.4 no polynomial-time state recovery attack on the BM generator is feasible. It means that, as the seed length increases, no polynomial-time adversary can retrieve the seed of the pseudorandom generator. However this asymptotic statement says little about

the security of the pseudorandom generator in practice for a particular seed length and against adversaries investing a specific amount of effort.

The security of the BM generator is proved by reduction (see e.g. [Go01]). It is shown that the problem of distinguishing BM sequences from random sequences can be reduced to inverting f. The reduction proposed is polynomial time but it is not tight.

In this paper we prove the impossibility of state recovery attacks on BM generators in a different way. We show that there exists a *tight reduction* between state recovery attacks on a BM generator and inversion algorithms for the corresponding one-way function.

Theorem 3.1 *Suppose the Blum-Micali pseudorandom generator is vulnerable against a $(T_{\mathcal{A}}, \epsilon)$-state recovery attack $\mathcal{A}$. Then there exists a probabilistic algorithm $\mathcal{S}$ that, given $y \in_R D$, calculates $x = f^{-1}(y)$ in time $T_{\mathcal{S}} = 2T_{\mathcal{A}} + cM$ with probability ϵ, where c is the computational complexity of f. The probability is taken over all choices of y and over internal coin flips of $\mathcal{S}$.*

PROOF. On input $y \in D$, algorithm $\mathcal{S}$ first constructs M bits of the BM sequence with seed x and then uses $\mathcal{A}$ to retrieve x. In total there are three steps.

1. Generate $s_2, \ldots, s_M : s_2 = b(y), s_3 = b(f(y)), \ldots, s_M = b(f^{M-2}(y))$.
2. Calculate $z_i = \mathcal{A}(i, s_2, s_3, \ldots, s_M)$ for $i = 0, 1$.
3. If $f(z_i) = y$ for one of the values $i \in \{0, 1\}$ set $x = z_i$.

Since the complexity of Step 1 is cM the running time of algorithm $\mathcal{S}$ is $T_{\mathcal{S}} = 2T_{\mathcal{A}} + cM$. The success probability of $\mathcal{S}$ is ϵ. □

In practice $cM \ll T_{\mathcal{A}}$. Hence $T_{\mathcal{S}}$ and $T_{\mathcal{A}}$ differ only by a factor of 2. Therefore the above reduction is tight. This reduction is called linear-preserving in terms of [Ha99].

We analyze two members of the Blum-Micali family: the Blum-Blum-Shub (BBS) generator [BBS86], and the generator based on the elliptic curve discrete logarithm [Ka88].

3.1 The BBS Pseudorandom Generator

Let N be a Blum integer, that is $N = pq$, where p, q are prime, $p \equiv q \equiv 3 \bmod 4$. Denote by n the bit length of N. Let $E_N(x) = x^2 \bmod N$ for $x \in \mathbb{Z}_N$. E_N is referred to as the Rabin function. Let QR_N be the set of quadratic residues modulo N. It is known that E_N permutes QR_N.

The seed x_1 of the BBS generator is a random quadratic residue modulo N. The pseudorandom sequence $s \in \{0, 1\}^M$ is generated as follows:

$$s_i = x_i \bmod 2,\ x_{i+1} = E_N(x_i)$$

for $i = 1, \ldots, M$.

The BBS generator is a BM generator with $D = QR_N, f(x) = E_N(x), b(x) = x \bmod 2$.

The following statement shows that a state recovery attack on the BBS generator can be efficiently reduced to a factoring algorithm.

Corollary 3.2 *Suppose the BBS pseudorandom generator is vulnerable against a $(T_{\mathcal{A}}, \epsilon)$-state recovery attack $\mathcal{A}$. Then there exists a probabilistic algorithm $\mathcal{F}$ that factors the modulus N in expected time $T_{\mathcal{F}} = 2\epsilon^{-1}T_{\mathcal{A}} + O(\epsilon^{-1}n^2M)$.*

PROOF. The proof uses a well-known fact that calculating square roots modulo N is equivalent to factoring N.

Let $x \in_R \mathbb{Z}_N \backslash QR_N$ and $y = E_N(x)$. According to Theorem 3.1 there exists an algorithm $\mathcal{S}$ that, given y, calculates $z \in QR_N$ such that $y = E_N(z)$ in time $T_{\mathcal{S}} = 2T_{\mathcal{A}} + O(n^2M)$ (the complexity of squaring modulo N is $O(n^2)$, where n is the length of N) with probability ϵ. The probability is taken over all choices of x and over internal coin flips of $\mathcal{S}$.

The following two possibilities each have probability $1/2$:

1. $z = +x \bmod p$ and $z = -x \bmod q$,
2. $z = -x \bmod p$ and $z = +x \bmod q$,

In Case 1 $\gcd(z - x, N) = p$ and in Case 2 $\gcd(z - x, N) = q$. Thus calculation of $\gcd(z - x, N)$ yields the factorization of N. The complexity of the gcd evaluation is negligible in comparison with $O(n^2M)$.

Algorithm $\mathcal{F}$ now works as follows. Select $x \in_R \mathbb{Z}_N$ and use $\mathcal{S}$ to determine z. If $\mathcal{S}$ succeeds calculate $\gcd(z - x, N)$. If $\mathcal{S}$ fails select $x \in_R \mathbb{Z}_N \backslash QR_N$ once again etc. On average $\mathcal{F}$ repeats the above procedure ϵ^{-1} times. Therefore $T_{\mathcal{F}} = \epsilon^{-1}T_{\mathcal{S}} = 2\epsilon^{-1}T_{\mathcal{A}} + O(\epsilon^{-1}n^2M)$. □

Alexi et al. [Al88], U. Vazirani and V. Vazirani [VV84] show that the BBS pseudorandom generator remains asymptotically secure if one extracts the first few least significant bits of x_i per iteration. Although the BBS generator with several output bits per iteration is beyond the Blum-Micali construction security of this generator against state recovery attacks follows from a straightforward generalization of Theorem 3.1.

Corollary 3.3 *Suppose the BBS pseudorandom generator with j output bits per iteration is vulnerable against a $(T_{\mathcal{A}}, \epsilon)$-state recovery attack $\mathcal{A}$. Then there exists a probabilistic algorithm $\mathcal{F}'$ that factors the modulus N in expected time $T_{\mathcal{F}'} = 2^j\epsilon^{-1}T_{\mathcal{A}} + O(\epsilon^{-1}n^2M)$.*

PROOF. The proof of this statement is similar to the proof of Corollary 3.2. When proving Corollary 3.2 we use Theorem 3.1 to construct algorithm $\mathcal{S}$ that inverts the Rabin function in time $T_{\mathcal{S}} = 2T_{\mathcal{A}} + O(n^2M)$ with probability ϵ. On Step 2 (see the

proof of Theorem 3.1) $\mathcal{S}$ runs algorithm $\mathcal{A}$ twice. In case of the BBS generator with j output bits per iteration the inversion algorithm $\mathcal{S}'$ on Step 2 has to run $\mathcal{A}$ for 2^j times and therefore $T_{\mathcal{S}'} = 2^j T_{\mathcal{A}} + O(n^2 M)$. Similarly to the proof of Corollary 3.2 $T_{\mathcal{F}'} = 2\epsilon^{-1} T_{\mathcal{S}'} = 2^j \epsilon^{-1} T_{\mathcal{A}} + O(\epsilon^{-1} n^2 M)$. □

3.2 The Pseudorandom Generator Based on the Elliptic Curve Discrete Logarithm

A pseudorandom generator based on the elliptic curve discrete logarithm (ECDL generator) is proposed by Kaliski [Ka88].

Let p be a prime, $p \equiv 2 \bmod 3$. Consider a curve $E(\mathbb{F}_p)$ that consists of points $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ such that

$$y^2 = x^3 + c,$$

where $c \in \mathbb{F}_p^*$. Kaliski [Ka88] shows that the points of $E(\mathbb{F}_p)$ together with a point at infinity $\mathcal{O}$ form a *cyclic* additive group of order $p + 1$. Let Q be a generator of this group. The security of the pseudorandom generator [Ka88] is based on the assumption that solving the discrete log problem over $E(\mathbb{F}_p)$ (given points Q, Y, find $\alpha \in \mathbb{Z}_{p+1}$ such that $Y = \alpha Q$) is infeasible. Let

$$\phi(P) = \begin{cases} y, \text{ if } P = (x, y); \\ p, \text{ if } P = \mathcal{O}. \end{cases}$$

The ECDL generator is a BM generator with $D = E(\mathbb{F}_p)$, $f(P) = \phi(P)Q$,

$$b(P) = \begin{cases} 1, \text{ if } \phi(P) \geq (p+1)/2; \\ 0, \text{ otherwise.} \end{cases}$$

The seed P_1 of the ECDL generator is a random point on the curve.

Corollary 3.4 *Suppose the ECDL pseudorandom generator is vulnerable against a $(T_{\mathcal{A}}, \epsilon)$-state recovery attack $\mathcal{A}$. Then there exist a probabilistic algorithm $\mathcal{L}$ that, given $Y \in E(\mathbb{F}_p)$, calculates $\alpha \in \mathbb{Z}_{p+1}$ such that $Y = \alpha Q$ in expected time $T_{\mathcal{L}} = 2\epsilon^{-1} T_{\mathcal{A}} + O(\epsilon^{-1} (\log p)^3 M)$.*

PROOF. The proof is based on Theorem 3.1 and on random-self-reducibility of the discrete logarithm problem. The details follow.

According to Theorem 3.1 there exists an algorithm $\mathcal{S}$ that on input $Z \in_R E(\mathbb{F}_p)$ calculates $X = f^{-1}(Z)$ in time $T_{\mathcal{S}} = 2T_{\mathcal{A}} + O((\log p)^3 M)$ with probability ϵ.

Let $Y \in E(\mathbb{F}_p)$. Algorithm $\mathcal{L}$ works as follows. Select $b \in_R \mathbb{Z}_{p+1}^*$. Use $\mathcal{S}$ to determine $X = f^{-1}(bY)$. If $\mathcal{S}$ succeeds calculate $a = b^{-1} \bmod (p+1)$ and $\alpha = a\phi(X)$ (in this case $bY = f(X) = \phi(X)Q$ and thus $Y = a\phi(X)Q$). If $\mathcal{S}$ fails select $b \in_R \mathbb{Z}_{p+1}^*$ once again etc. On average $\mathcal{L}$ repeats the above procedure ϵ^{-1} times. Therefore $T_{\mathcal{L}} = \epsilon^{-1} T_{\mathcal{S}} = 2\epsilon^{-1} T_{\mathcal{A}} + O(\epsilon^{-1} (\log p)^3 M)$. □

4 Concrete Result for the BBS generator

Consider the BBS pseudorandom generator with 1 output bit per iteration (see Section 3.1). Suppose our goal is to generate a pseudorandom sequence of length $M = 10^7$ such that no distinguishing attack $\mathcal{D}$ can distinguish the sequence from truly random sequence in time $T_{\mathcal{D}} = 2^{80}$ with advantage $\epsilon = 0.01$ (we use the same choice of parameters as in [FS00, Ge05, Kn97]). The question is what bit length n of the modulus N guarantees this level of security.

The security of the BBS generator relies on the intractability of factoring large Blum integers. The fastest general-purpose factoring algorithm today is the Number Field Sieve. According to [LV01] on heuristic grounds the Number Field Sieve is expected to require time proportional to $\gamma \exp((1.9229 + o(1))(\ln N)^{1/3}(\ln \ln N)^{2/3})$ for a constant γ. Following [LV01] we make an assumption that the $o(1)$-term can be treated as zero. Let $T_{\mathcal{F}}(n)$ be the number of clock cycles needed to factor an n-bit integer. Then

$$T_{\mathcal{F}}(n) \approx \gamma \exp(1.9229(n \ln 2)^{1/3}(\ln(n \ln 2))^{2/3}).$$

We use one clock cycle as a unit of time. Experience from available data points suggests that $T_{\mathcal{F}}(512) \approx 3 \cdot 10^{17}$ [LV01]. Therefore $\gamma \approx 2.8 \cdot 10^{-3}$ and

$$T_{\mathcal{F}}(n) \approx 2.8 \cdot 10^{-3} \cdot \exp(1.9229(n \ln 2)^{1/3}(\ln(n \ln 2))^{2/3}).$$

ASSUMPTION 4.1 *No algorithm can factor a randomly chosen n-bit Blum integer faster than in $T_{\mathcal{F}}(n)$ clock cycles.*

The strongest security proof for of the BBS generator is proposed by Fischlin and Schnorr [FS00].

Lemma 4.2 (Fischlin and Schnorr) *Under Assumption 4.1, no $(T_{\mathcal{D}}, \epsilon)$-distinguishing attack on the BBS generator exists if*

$$T_{\mathcal{D}} \leq \frac{T_{\mathcal{F}}(n)}{6n(\log_2 n)\epsilon^{-2}M^2} - 2^7 n\epsilon^{-2}M^2 \log_2(8n\epsilon^{-1}M).$$

We observe that $T_{\mathcal{F}} \gg T_{\mathcal{D}}$. Hence the security reduction is not tight.

Lemma 4.2 implies that for $M = 10^7$ no $(2^{80}, 0.01)$-distinguishing attack on the BBS exists if $n \geq 6800$. On the other hand due to Corollary 3.2 the security against $(2^{80}, 0.01)$-state recovery attacks is guaranteed for $n \geq 1100$. Since the security reduction in case of the state recovery attacks is tighter than in case of the distinguishing attacks the length of the modulus that guarantees security against the state recovery attacks is significantly smaller than the one that guarantees security against distinguishing attacks.

5 Conclusion

Provable security is a substantial and desirable property of cryptographic primitives. To our knowledge, there exists no provably secure pseudorandom generator with a tight se-

curity reduction. For this reason the provably secure pseudorandom generators are slow. Often it makes them impractical.

One of the important examples of provably secure pseudorandom generators is the Blum-Micali family. They are known to be asymptotically secure, shown using polynomial time reductions which are far from tight. Moreover, in case of the BBS generator the reduction cannot be tighter [FS00]. In this paper we show that there exists a tight security reduction for a subclass of attacks, namely for state recovery attacks.

A way to improve tightness of the security reductions (and thus to make the pseudorandom generators more efficient) is to weaken Yao's definition of security for pseudorandom generators. The fact that there exists no efficient distinguishing attack on a pseudorandom generator means that every bit of the pseudorandom sequence is unpredictable. Is it necessary to demand such a strong property of pseudorandom generators? For example, suppose a pseudorandom generator is used for generating 1024-bit RSA keys. In this case leakage of a few bits of the secret key does not affect the security of the scheme, since the Number Field Sieve can not benefit from this information. Wanders [Wa87] points out that a small deviation from Golumb's randomness postulates results in even smaller information leak per bit (namely, the information leak is quadratic in the deviation). Our work raises an interesting open problem: is it possible to weaken the definition of security for pseudorandom generators in a reasonable way so that there exists a pseudorandom generator with tight security reduction?

References

[Al88] W. Alexi, B. Chor, O. Goldeich, and C. P. Schnorr. RSA and Rabin functions: Certain Parts are as Hard as the Whole. *SIAM Journal on Computing*, pages 194–209, 1988.

[BBS86] L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, pages 364–383, 1986.

[BM84] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[BR96] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Advances in Cryptology – Eurocrypt 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Berlin, 1996. Springer-Verlag.

[Ek03] P. Ekdahl. *On LFSR based Stream Ciphers – Analysis and Design*. PhD thesis, Lund University, Faculty of Technology, Lund, Sweden, 2003.

[FS00] R. Fischlin and C. P. Schnorr. Stronger Security Proofs for RSA and Rabin Bits. *Journal of Cryptology*, 13(2):221–244, 2000.

[Ge05] R. Gennaro. An Improved Pseudo-random Generator Based on the Discrete Logarithm Problem. *Journal of Cryptology*, 18(2):91–110, 2005.

[Go01] O. Goldreich. *Foundations of Cryprography. Basic Tools*. Cambridge University Press, Cambridge, United Kingdom, 2001.

[Ha99] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

[Ka88] B. S. Kaliski. *Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools*. PhD thesis, MIT, Cambridge, MA, USA, 1988.

[Kn98] L. R. Knudsen et al. Analysis Methods for (Alleged) RC4. In *Advances in Cryptology – Asiacrypt 1998*, volume 1514 of *Lecture Notes in Computer Science*, pages 327–341, Berlin, 1998. Springer-Verlag.

[Kn97] D. E. Knuth. *Seminumerical Algorithms*, volume 3. Addison-Wesley, Reading, MA, USA, third edition, 1997.

[LV01] A. K. Lenstra and E. R. Verheul. Selecting Cryptographic Key Sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[Ma05] I. Mantin. Predicting and Distinguishing Attacks on RC4 Keystream Generator. In *Advances in Cryptology – Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 491–506, Berlin, 2005. Springer-Verlag.

[Ma68] G. Marsaglia. Random Numbers Fall Mainly in the Planes. *Proceedings of the National Academy of Sciences*, 61(1):25–28, 1968.

[Ma69] J. L. Massey. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory*, 15:122–127, 1969.

[VV84] U. V. Vazirani and V. V. Vazirani. Effi cient and Secure Pseudo-Random Number Generation. In *IEEE Symposium on Foundations of Computing Science*, pages 458–463, 1984.

[Wa87] H. Wanders. Two Topics in Secrecy. Master's thesis, Eindhoven University of Technology, Department of Mathematics and Computing Science, Eindhoven, The Netherlands, 1987.

[Ya82] A. C. Yao. Theory and Application of Trapdoor Functions. In *IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.