Trends in Mathematics
Information Center for Mathematical Sciences
Volume 8, Number 1, June, 2005, Pages 1–11

SECURITY FRAMEWORKS FOR PSEUDORANDOM NUMBER GENERATORS

JU-SUNG KANG

Abstract. In the cryptographic system a pseudorandom number generator is one of the basic primitives. We survey theoretically secure pseudorandom bit generators which are provably secure under certain reasonable number theoretic assumptions and some practical pseudorandom number generators based on assumptions about symmetric crypto-primitives. Recently, there was a noticeable result for the concrete security analysis of pseudorandom number generators used in practice. Thus we have a closer look on the Desai-Hevia-Yin's security framework and consider the provable security for some practical pseudorandom number generators.

1. Introduction

A pseudorandom number generator is one of the basic primitives for most cryptographic systems. It is hard to imagine a cryptographic application that doesn't use random numbers. For example, seeds for key generation, session keys used for encryption and authentication, salts to be hashed with passwords, the private key in the DSA, the primes in the RSA, and nonces in protocols are all assumed to be random. Because generating the real random sequence is expensive, most applications use a pseudorandom number generator (PRNG). The PRNG tries to stretch a short string of random bits to a longer string that are indistinguishable from the truly random string.

On the other hand, a random bit generator can be used to generate random numbers. A random integer in the interval $[0, n]$ can be obtained by generating a random bit sequence of length $\lfloor \log n \rfloor + 1$, and converting it to an integer. If the resulting integer exceeds n, we may discard it and generate a new random bit sequence. A true random bit generator is a device or algorithm which outputs a sequence of probabilistically independent and unbiased binary digits. While a PRBG is defined by a deterministic algorithm which, given a truly random bit sequence of length k, outputs a binary sequence of length $l \gg k$ which appears to be random. The initial input to the PRBG is called the seed, and output of the PRBG is called a pseudorandom bit sequence.

This work was supported by the new faculty research program 2004 of Kookmin University.

©2005 Information Center for Mathematical Sciences

A rigorous theory of pseudorandom generators initiated by Blum and Micali [1], and Yao [2] in the early 1980's. Blum and Micali [1] presented a general construction for cryptographically secure PRBGs and proposed the first concrete instance of cryptographically secure PRBG, called the Blum-Micali generator. Yao [2] showed how to obtain a cryptographically secure PRBG using any one-way permutation. Håstad et al. [3] generalized this result and showed how to construct a cryptographically secure PRBG from any one-way function. Since it is easy to construct a one-way function from a PRBG, the result of [3] disclosed the necessary and sufficient condition for the existence of a PRBG. The one-way function is a function easy to compute, but hard to invert. However we don't know if theoretically perfect one-way functions exist. A good block cipher and cryptographic hash functions are regarded as strong candidates of the one-way function in real-world. It can also be noted that if one relaxes the requirement that the PRBG be efficiently computable, then cryptographically secure PRBGs do exist that pass all polynomial-time tests.

Kelsey et al. [4] discussed possible cryptanalytic attacks against their proposed model for PRNGs. They considered PRNGs from an attacker's perspective and presented the strengths and weaknesses of four real-world PRNGs: the ANSI X9.17 PRNG, the DSA PRNG, the RSAREF PRNG, and CryptoLib. However Kelsey et al.'s analysis was mostly ad hoc and based on heuristic arguments. Desai, Hevia, and Yin [5] gave a general security framework for PRNGs, incorporating the attacks that users were typically concerned about and analyzed the ANSI X9.17 PRNG and the FIPS 186 PRNG. Their security framework did look at PRNGs from a provable-security viewpoint which is similar to the security treatment of Bellare et al. [6, 22]. Desai-Hevia-Yin's work provided a concrete security analysis for the PRNGs based on efficient cryptographic primitives, such as block ciphers and hash functions.

In this paper, we briefly survey theoretically secure PRBGs that are provably secure under certain reasonable number theoretic assumptions and some PRNG constructions based on assumptions about symmetric crypto-primitives. Recently, there was a noticeable result for the concrete security analysis of pseudorandom number generators used in practice. Thus we have a closer look on the Desai-Hevia-Yin's security framework and consider the provable security for some practical pseudorandom number generators.

2. Theoretically-secure PRBGs

A minimum security requirement for a PRBG is that the length k of the random seed should be sufficiently large so that a search over 2^k elements is infeasible for the adversary. Two general requirements are that the output sequences of a PRBG should be indistinguishable from truly random sequences, and the output bits should be unpredictable to an adversary with limited computational resources.

Definition 1. *(Yao [2], Goldreich-Goldwasser-Micali [8]) A PRBG is said to pass all polynomial-time statistical tests if no polynomial-time algorithm can correctly distinguish between an output sequence of the generator and a truly random sequence of the same length with probability significantly greater than* $1/2$.

Definition 2. *(Yao [2], Goldreich-Goldwasser-Micali [8]) A PRBG is said to pass the next-bit test if there is no polynomial-time algorithm which, on input of the first* l *bits of an output sequence* s*, can predict the* $(l+1)$*-st bit of* s *with probability significantly greater than* $1/2$.

For the formal definitions of a polynomial-time statistical test and next-bit test, see Yao [2] and Goldreich-Goldwasser-Micali [8]. Yao [2] showed that Definition 1 and 2 are equivalent and Goldreich, Goldwasser, and Micali [8] provided a generalization of Yao's result.

Theorem 1. *(Yao [2], Goldreich-Goldwasser-Micali [8]) A PRBG passes the next-bit test if and only if it passes all polynomial-time statistical tests.*

Blum-Micali [1] introduced the notion of a cryptographically strong PRBG as the following definition.

Definition 3. *A PRBG that passes the next-bit test is called a cryptographically strong PRBG.*

The pseudorandom sequence should have some statistical properties present in truly random sequences. Many statistical properties of the linear congruential pseudorandom number sequence

$$x_{i+1} = a \cdot x_i + b \pmod{n}$$

have been studied by Knuth [9]. However, unlike truly random sequences, the next number in a linear congruential sequence can be easily computed from the preceding ones, even when x_0, a, b, and n are not given [10]. While linear congruential generators pass some statistical tests, they are predictable and thus entirely insecure for cryptographic purposes. In this section we introduce three cryptographically strong PRBGs.

Blum-Micali PRBG [1]

- Summary: a pseudorandom bit sequence $y_1, y_2, \ldots, y_l$ of length l is generated.

(1) Let p be a large prime, and α a generator of Z_p^*.
(2) Select a random seed $x_0 \in Z_p^*$.
(3) For $1 \le i \le l$, do the following:
  (a) $x_i \leftarrow \alpha^{x_{i-1}} \pmod{p}$

(b) If $0 \leq x_{i-1} = \log_\alpha x_i \leq (p-1)/2$ then set $y_i \leftarrow 1$; otherwise set $y_i \leftarrow 0$.

(4) The output sequence is $y_1, y_2, \ldots, y_l$.

Assuming the intractability of the discrete logarithm problem in Z_p^*, the Blum-Micali PRBG was proven to satisfy the next-bit test. Long-Wigderson [11] improved the efficiency og the Blum-Micali PRBG by simultaneously extracting $O(\log \log p)$ bits from each x_i. Kaliski [12] modified the Blum-Micali PRBG so that the security depends on the discrete logarithm problem in the group of points on an elliptic curve defined over a finite field.

Blum-Blum-Shub PRBG [13]

- Summary: a pseudorandom bit sequence $y_1, y_2, \ldots, y_l$ of length l is generated.

(1) Generate two large secret random and distinct primes p and q, each congruent to 3 modulo 4, and compute $n = pq$.

(2) Select a random and secret seed $s \in [1, n-1]$ such that $\gcd(s, n) = 1$, and compute $x_0 \leftarrow s^2 \pmod{n}$.

(3) For $1 \leq i \leq l$ do the following:

(a) $x_i \leftarrow x_{i-1}^2 \pmod{n}$.

(b) $y_i \leftarrow$ the least significant bit of x_i.

(4) The output sequence is $y_1, y_2, \ldots, y_l$.

Blum-Blum-Shub [13] showed that their PRBG is a cryptographically strong assuming the intractability of the quadratic residuosity problem. Vazirani-Vazirani [14] established a stronger result regarding the security of the Blum-Blum-Shub PRBG by proving it cryptographically strong under the weaker assumption that integer factorization is intractable.

RSA PRBG [15]

- Summary: a pseudorandom bit sequence $y_1, y_2, \ldots, y_l$ of length l is generated.

(1) Generate two secret RSA-like primes p and q, and compute $n = pq$ and $\phi = (p-1)(q-1)$. Select a random integer e, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.

(2) Select a random seed $x_0 \in [1, n-1]$.

(3) For $1 \leq i \leq l$ do the following:

(a) $x_i \leftarrow x_{i-1}^e \pmod{n}$.

(b) $y_i \leftarrow$ the least significant bit of x_i.

(4) The output sequence is $y_1, y_2, \ldots, y_l$.

Alexi et al. [15] proved that the RSA PRBG is a cryptogrphically strong under the assumption that the RSA problem is intractable. Moreover they provided that the above RSA PRBG can be improved by extracting the $j = c \log\log n$ least significant bits of x_i, where c is a constant. Micalli-Schnorr [16] proposed a modification improving the efficiency of the RSA PRBG. In their PRBG, one can extact the $k = \lfloor N(1 - 2/e) \rfloor$ least significant bits of x_i, where $N = \lfloor \log n \rfloor + 1$. However Micali-Schnorr PRBG is cryptographically strong under the assumption which stronger than requiring that the RSA problem be intractable.

3. Practical PRNGs

In section 2, we have introduced some constructions for pseudorandom number generation that are provably secure under certain reasonable number theoretic assumptions. However these theoretically secure PRBGs are still impractical. For practical reasons, the PRNGs, that are in prevalent use today, are typically based on efficient cryptographic primitives such as block ciphers and hash functions. In this section we describe two most widely-used PRNGs, the ANSI X9.17 PRNG [17] and the FIPS 186 PRNG [18].

ANSI X9.17 PRNG [17]

- Input: a random and secret 64-bit seed s, integer m, and Triple DES E-D-E encryption key k.
- Output: m pseudorandom 64-bit strings x_1, x_2, ..., x_m.

(1) Compute $I = E_k(T)$, where T is a 64-bit representation of the date/time to as fine a resolution as is available.
(2) For $1 \leq i \leq m$ do the following:
  (a) $x_i \leftarrow E_k(I \oplus s)$.
  (b) $s \leftarrow E_k(x_i \oplus I)$.
(3) Return $(x_1, x_2, \ldots, x_m)$.

The ANSI X9.17 PRNG is a part of a popular banking standard for the purpose of pseudorandomly generating keys and initialization vectors for the use with DES. In the above algorithm, E_k denotes DES E-D-E two-key triple-encryption under a key k. Two-key triple-encryption is defined by $E(x) = E_{k_1}(D_{k_2}(E_{k_1}(x)))$, where $D_k = E_k^{-1}$.

FIPS 186 PRNG using SHA-1 for DSA private keys [18]

- Input: a 160-bit prime q and an integer m.
- Output: m pseudorandom integers a_1, a_2, ..., a_m in the interval $[0, q-1]$ which may be used as DSA private keys.

(1) Select an arbitrary integer b, $160 \leq b \leq 512$.
(2) Generate a random and secret b-bit seed s.
(3) Define the 160-bit string in hexadecimal notation, t = 67452301 efcdab89 98badcfe 10325476 c3d2e1f0.
(4) For $1 \leq i \leq m$ do the following:
    (a) (Optional user input) Either select a b-bit string y_i, or set $y_i \leftarrow 0$.
    (b) $x_i \leftarrow (s + y_i) \pmod{2^b}$.
    (c) $a_i \leftarrow H(t, x_i) \pmod{q}$.
    (d) $s \leftarrow (1 + s + a_i) \pmod{2^b}$
(5) Return $(a_1, a_2, \ldots, a_m)$.

The FIPS 186 PRNG [18] has standardized for generating randomness in DSA. The above algorithm generates DSA private keys, where H denotes the one-way function using SHA-1, and also there is slightly different algorithm for generating the per-message secrets. The PRNG for generating the DSA per-message secrets doesn't permit the optional user input in step 4.1 of the above algorithm. On the other hand, the one-way function H using SHA-1 can be replaced by the one-way function using DES.

Several years ago, an unpublished attack on DSA was found that relies on the non-uniformity of the PRNGs specified in the Appendix 3 of the FIPS 186 [18]. Then, on 2000, NIST announced the approval of FIPS 186-2 [19] with Change Notice 1 which includes algorithms modifying the previous PRNGs. These modifications reduce the non-uniformity of the PRNGs and do not affect interoperability. For example, the revised version of the FIPS 186 PRNG using SHA-1 for DSA private keys [18] is as follows:

FIPS 186-2 (Change Notice 1) PRNG using SHA-1 for DSA private keys [19]

- Input: a 160-bit prime q and an integer m.
- Output: m pseudorandom integers a_1, a_2, ..., a_m in the interval $[0, q-1]$ which may be used as DSA private keys.

(1) Select an arbitrary integer b, $160 \leq b \leq 512$.
(2) Generate a random and secret b-bit seed s.
(3) Define the 160-bit string in hexadecimal notation, t = 67452301 efcdab89 98badcfe 10325476 c3d2e1f0.
(4) For $1 \leq i \leq m$ do the following:
    (a) (Optional user input) Either select a b-bit string y_i, or set $y_i \leftarrow 0$.
    (b) For $j = 0$ to 1 do
        (i) $x_i \leftarrow (s + y_i) \pmod{2^b}$.
        (ii) $w_j \leftarrow H(t, x_i)$.
        (iii) $s \leftarrow (1 + s + w_j) \pmod{2^b}$.

(c) $a_i = (w_0||w_1) \pmod q$.

(5) Return $(a_1, a_2, \ldots, a_m)$.

Desai-Hevia-Yin [5] provided an analysis of the ANSI X9.17 and the FIPS 186 PRNGs by applying the practice-oriented provable security [21]. Their analysis was to formalize reasonable security assumptions on the underlying primitives. They indicated that the one-way property itself is not enough to ensure the security of the construction, and assumed that the underlying primitives are pseudorandom functions.

On the other hand, the assumption that the underlying primitive is a pseudorandom function is quite a strong one. A weaker assumption is that we have an one-way function which is hard to invert. Håstad-Nåslund [20] proposed the BMGL construction which used an optimization of Blum-Micali PRBG. The underlying primitive of the BMGL construction is the block cipher AES. Håstad-Nåslund [20] showed that if one assumes more than just one-wayness, e.g. that the underlying function is also a permutation, then the situation becomes much more favorable and much simpler constructions can be found.

4. Security framework for practical PRNGs

In this section, we discuss the practice-oriented provable security [21] framework for PRNGs. Desai-Hevia-Yin [5] gave a general security framework for PRNGs, incorporating the attacks that users are typically concerned about. They modelled a PRNG as an iterative algorithm, that in each iteration take three inputs: a key, a current state, and an auxiliary input.

Definition 4. *(Desai-Hevia-Yin [5]) A PRNG $\mathcal{GE} = (\mathcal{K}, \mathcal{G})$ consists of two algorithms.*

- *The seed generation algorithm $\mathcal{K}$ takes as input a security parameter k and returns a key K and an initial state s_0.*
- *For $i \geq 1$, the generation algorithm $\mathcal{G}$ takes as input the key K, the current state s_{i-1} and an auxiliary input t_i, and returns a PRNG output y_i and the next state s_i.*

Under the above PRNG model, Desai-Hevia-Yin [5] introduced three different attacks, CIA(Chosen-Input Attack), CSA(Chosen-State Attack), and KKA(Known-Key Attack). Under CIA, the key is hidden, the states are known, but not chosen, and auxiliary input may be chosen by attacker. CSA is similar, except that the auxiliary inputs are not allowed to be chosen while the states may be chosen. KKA is different in that it allows the key be known. However, under KKA, the states are hidden and the auxiliary inputs are not allowed to be chosen. These three attacks can be coupled with the notion of pseudorandomness as the following definitions.

Definition 5. *(Desai-Hevia-Yin [5]) Let $\mathcal{G} = (\mathcal{K}, \mathcal{G})$ be a PRNG with block-length n. Let $b \in \{0,1\}$. Let D be a distinguisher that runs in two stages. For $atk \in \{cia, csa, kka\}$, the experiment $\mathbf{EXP}^{prng-atk-b}_{\mathcal{G}E,m,D}$ is defined as follows:*

- $(K, s_0) \xleftarrow{R} \mathcal{K}(k)$. *A key and an initial state are generated.*
- $c_0 \leftarrow \{t_1, t_2, \ldots, t_m\}$. *$D$ is initialized with auxiliary input values.*
- *For $i = 1$ to m :* $y_i^0 \xleftarrow{R} \{0,1\}^n$. *A random mn-bit string is chosen.*
- $i \leftarrow 0$, $y_0^0 \leftarrow \epsilon$, $y_0^1 \leftarrow \epsilon$.
- *Repeat*
  - $i \leftarrow i + 1$.
  - *if $atk = cia$,* $(p_i, t_i, c_i) \leftarrow D(find, y_{i-1}^b, s_{i-1}, c_{i-1})$.
  - *if $atk = csa$,* $(p_i, s_{i-1}, c_i) \leftarrow D(find, y_{i-1}^b, s_{i-1}, c_{i-1})$.
  - *if $atk = kka$,* $(p_i, c_i) \leftarrow D(find, K, y_{i-1}^b, c_{i-1})$.
  - $(y_i^1, s_i) \leftarrow \mathcal{G}_K(s_{i-1}, t_i)$. *PRNG output and next state are generated.*
- *until* $(p_i = guess)$ *or* $(i = m)$.
- $d \leftarrow D(guess, c_i)$.
- *Return d.*

Definition 6. *The advantage of the distinguisher D is defined by*

$$ADV^{prng-atk}_{\mathcal{G}E,m,D} = Pr\left(\mathbf{EXP}^{prng-atk-1}_{\mathcal{G}E,m,D} = 1\right) - Pr\left(\mathbf{EXP}^{prng-atk-0}_{\mathcal{G}E,m,D} = 1\right) ,$$

and, for any integer t, the advantage function of $\mathcal{G}E$ is defined by

$$ADV^{prng-atk}_{\mathcal{G}E,m}(t) = \max_D \{ADV^{prng-atk}_{\mathcal{G}E,m,D}\} ,$$

where the maximum is over all D with time complexity t.

In [5], the ANSI X9.17 and FIPS 186 PRNGs were simplified by the author's representation. For a block cipher F, the simplified ANSI X9.17 PRNG $\mathcal{G}E^F_{ANSI} = (\mathcal{K}_{ANSI}, \mathcal{G}_{ANSI})$ is defined by

- $\underline{\mathcal{K}_{ANSI}(n)}$
  - $K \xleftarrow{R} \{0,1\}^n$, $s_0 \xleftarrow{R} \{0,1\}^n$.
  - Return (K, s_0).
- $\underline{\mathcal{G}_{ANSI}(s_{i-1}, t_i)}$
  - $y_i \leftarrow F_K(s_{i-1} \oplus F_K(t_i))$.
  - $s_i \leftarrow F_K(y_i \oplus F_K(t_i))$.
  - Return (y_i, s_i).

The PRNG $\mathcal{G}E^F_{ANSI}$ is insecure under any attack in which the key is known. An attacker, knowing the key and a single output, can completely determine subsequent outputs and states. Under the assumption that F is a finite pseudorandom function family [22], it has been shown that $\mathcal{G}E^F_{ANSI}$ is secure in the PRNG-CSA sense and also in the PRNG-CIA sense [5].

Theorem 2. *(Desai-Hevia-Yin [5]) Let $\mathcal{GE}$ be the ANSI X9.17 PRNG based on a function family $F = \{F_K\}$ where F_K maps n-bit to n-bit. Then*

$$ADV^{prng-csa}_{\mathcal{GE},m}(t) \leq 2 \cdot ADV^{prf}_{F}(t, 3m) + \frac{m(2m-1)}{2^n} ,$$

and

$$ADV^{prng-cia}_{\mathcal{GE},m}(t) \leq 2 \cdot ADV^{prf}_{F}(t, 3m) + \frac{(2m-1)^2}{2^n} .$$

In the theorem 3, $ADV^{prf}_{F}(t, q)$ [22] denotes the maximum advantage of any distinguisher for F versus a random function with time complexity t and query complexity q.

For a function H, the simplified FIPS 186 PRNG $\mathcal{GE}^{H}_{FIPS} = (\mathcal{K}_{FIPS}, \mathcal{G}_{FIPS})$ is defined by

- $\underline{\mathcal{K}_{FIPS}(n)}$
  - $K \leftarrow \{0,1\}^n$, $s_0 \xleftarrow{R} \{0,1\}^n$.
  - Return (K, s_0).
- $\underline{\mathcal{G}_{FIPS}(s_{i-1}, t_i)}$
  - $y_i \leftarrow H_K((s_{i-1} + t_i) \pmod{2^n})$.
  - $s_i \leftarrow s_{i-1} + y_i + 1 \pmod{2^n}$.
  - Return (y_i, s_i).

Note that in the FIPS 186 PRNG, unlike in the ANSI X9.17, the key K is known, and the values for K are fixed in the original specification [18]. Since the key K is known, the only secret in $\mathcal{GE}_{FIPS}$ is the initial state s_0. Thus Desai-Hevia-Yin [5] proposed an alternative view for the underlying primitive H_K. Let $\hat{H}_{s_0}(x) = H_K(s_0 + x \pmod{2^n})$, then we can rewrite the $\mathcal{GE}_{FIPS}$ as follows:

- $\underline{\mathcal{K}_{FIPS}(n)}$
  - $K \leftarrow \{0,1\}^n$, $s_0 \xleftarrow{R} \{0,1\}^n$.
  - Return (K, s_0).
- $\underline{\hat{\mathcal{G}}_{FIPS}(s_{i-1}, t_i)}$
  - $\Delta s_{i-1} \leftarrow s_{i-1} - s_0 \pmod{2^n}$.
  - $y_i \leftarrow \hat{H}_{s_0}((\Delta s_{i-1} + t_i) \pmod{2^n})$.
  - $s_i \leftarrow s_{i-1} + y_i + 1 \pmod{2^n}$.
  - Return (y_i, s_i).

Under the assumption that $\hat{H} = \{\hat{H}_{s_0}\}$ is a finite pseudorandom function family, it has been shown that $\mathcal{GE}^{\hat{H}}_{FIPS}$ is secure in the PRNG-KKA sense.

Theorem 3. *(Desai-Hevia-Yin [5]) Let $\mathcal{GE}$ be the FIPS 186 PRNG based on a function family $\hat{H} = \{\hat{H}_{s_0}\}$. Then*

$$ADV^{prng-kka}_{\mathcal{GE},m}(t) \leq 2 \cdot ADV^{prf}_{\hat{H}}(t, m) + \frac{m(m-1)}{2^{n-1}} .$$

5. Conclusion

The theoretically-secure constructions of PRBGs has up to now mostly focused on methods based on number theoretic assumptions. The existing schemes of cryptographically strong PRBGs are very inefficient, and can be used in the system require a lower expectations of the speed, but require stronger guarantees of security. Thus an improving efficiency of the previous theoretically-secure PRBGs would be an interesting research topics. On the other hand, the results on PRBGs from number theoretic primitives are only analyzed asymptotic terms. And an analysis in the model of exact security is of practical interest.

Not much theoretical work has been done on PRNGs based on efficient cryptographic primitives. The only reference of a theoretical work for practical PRNGs is Desai-Hevia-Yin [5], but its full version is not published yet. It may be necessary to examine the practice-oriented provable security for practical PRNGs different from the ANSI 9.17 and FIPS 186 PRNGs.

References

[1] M. Blum and S. Micali, "How to generate cryptographycally strong sequences of pseudorandom bits", SIAM Journal on Computing, Vol. 13, No. 4, 1986, pp. 850-864.

[2] A. C. Yao, "Theory and applications of trapdoor functions", 23rd IEEE FOCS, 1982, pp. 80-91.

[3] J. Håstad, R. Impagliazzo, L. A. Levin, M. Luby, "Pseudorandom number generators from any one-way function, SIAM Journal on Computing", Vol. 28, No. 4, 1999, pp. 1364-1396.

[4] J. Kelsey, B. Schneier, D. Wagner, C. Hall, "Cryptanalytic attacks on pseudorandom number generators", FSE'98.

[5] A. Desai, A. Hevia, Y. L. Yin, "A practice-oriented treatment of pseudorandom number generators", EUROCRYPT 2002, LNCS 2332, 2002, pp. 368-383.

[6] M. Bellare, A. Desai, E. Jokipi, P. Rogaway, A concrete security treatment of symmetric encryption, FOCS 1997.

[7] M. Bellare, J, Killian, P. Rogaway, "The security of the cipher block channing message authentication code", CRYPTO'94.

[8] O. Goldreich, S. Goldwasser, S. Micali, "How to construct random functions", Journal of the Association for Computing Machinery, Vol. 33, No. 4, 1986, pp. 792-807.

[9] D. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, Vol. 2, 2nd ed., Addison-Wesley, 1981.

[10] J. Plumstead, "Inferring a sequence generated by a linear congruence", Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, 1982, pp. 153-159.

[11] D. L. Long, A. Wigderson, "The discrete logarithm hides $O(\log n)$ bits", SAIM Journal on Computing, Vol. 17, 1988, pp. 363-372.

[12] B. S. Kaliski, "A pseudo-random bit generator based on elliptic logarithms", CRYPTO'86, LNCS 263, 1987, pp. 84-103.

[13] L. Blum, M. Blum, M. Shub, "A simple unpredictable pseudorandom number gerator", SIAM Journal on Computing, Vol. 15, 1986, pp. 364-383.

[14] U. V. Vazirani, V. V. Vazirani, "Efficient and secure pseudo-random number generation", CRYPTO'84, LNCS 196, 1985, pp. 193-202.

[15] W. Alexi, B. Chor, O. Goldreich, C. P. Schnorr, "RSA and Rabin functions: Certain parts are as hard as the whole", SIAM Journal on Computing, Vol. 17, 1988, pp. 194-209.

[16] S. Micali, C. P. Schnorr, "Efficient, perfect polynomial randomnumber generators", Journal of Cryptology, Vol. 3, 1991, pp. 157-172.

[17] ANSI X9.17, "American National Standard - Financial institution message authentication (wholesale)", 1986.

[18] FIPS 186, "Digital Signature Standard", National Institute of Standards and Technologies, 1994.

[19] FIPS 186-2, "Digital Signature Standard Change Notice 1", National Institute of Standards and Technologies, 2001.

[20] J. Håstad, M. Nåslund, "BMGL: Synchronous key-stream generator with provable security", NESSIE submission, http://www.cryptonessie.org, 2000.

[21] M. Bellare, "Practice-oriented provable-security", ISW'97, LNCS 1396, 1998.

[22] M. Bellare, J. Kilian, P. Rogaway, "The security of the cipher block chaining message authentication code", CRYPTO'94, LNCS 839, 1994.

Department of Mathematics, Kookmin University, Jeongreung3-Dong, Seongbuk-Gu, Seoul, 136-702, KOREA

E-mail address: jskang@kookmin.ac.kr