Yale University
Department of Computer Science

On Backtracking Resistance in Pseudorandom Bit Generation (preliminary version)

Michael J. Fischer   Mike Paterson   Ewa Syta

YALEU/DCS/TR-1466
October 24, 2012
(Revised December 4, 2012)

On Backtracking Resistance in Pseudorandom Bit Generation[*]

Michael J. Fischer[†] Mike Paterson[‡] Ewa Syta[†]

Abstract

An *incremental* pseudorandom bit generator (iPRBG) is *backtracking resistant* if a state compromise does not allow the attacker a non-negligible advantage at distinguishing previously-generated bits from uniformly distributed random bits. Backtracking resistance can provide increased security in cryptographic protocols with long-term secrecy requirements. While a necessary condition for an iPRBG to be backtracking resistant is that the next-state function be one-way, we show that this condition is not sufficient. To do this, we assume that an iPRBG based on a one-way next-state permutation exists. We convert such an iPRBG into a new iPRBG that generates the same output sequence and is also based on a one-way next-state permutation, but the new generator is provably not backtracking resistant. We leave open the important question of whether cryptographically secure backtracking resistant iPRBGs exist, even assuming that cryptographically strong PRBGs and one-way permutations exist.

Keywords: backtracking resistance; pseudorandom generator; computational indistinguishability; one-way permutation; security

1 Introduction

Random numbers are widely applicable in many fields such as statistical sampling, computer simulation and gambling. However, they play an especially important role in cryptography as almost all cryptographic protocols require some amount of randomness. Since random numbers are impossible to generate on deterministic computers, pseudorandom bit generators (PRBGs) are used in practice to provide bits that are computationally indistinguishable from random bits. This property is desired for cryptographic uses since cryptographic protocols often rely on the unpredictability of pseudorandom numbers to maintain their security properties. Pseudorandom numbers are used in the key generation of encryption algorithms, random challenges in authentication protocols, salts for passwords, cryptographic nonces, etc [6].

It has become customary to consider the security of a PRBG primarily in terms of the quality of the output it produces, more specifically, its indistinguishability from random output. Therefore, the notion of computational indistinguishability has become equated with cryptographic security of a generator and has become the gold standard of PRBG security. However, there are two other highly

[*] This material is based upon work supported by the Defense Advanced Research Agency (DARPA) and SPAWAR Systems Center Pacific, Contract No. N66001-11-C-4018.

[†] Department of Computer Science, Yale University, CT, USA

[‡] Department of Computer Science, University of Warwick, Coventry, UK

desirable and often overlooked properties: *backtracking resistance* and *prediction resistance* [1]. Both properties provide resistance to attacks under the more general security model in which an adversary is able to obtain the internal state of the generator, potentially in addition to some subset of the past output and an ability to observe some future output. Informally, a PRBG is *backtracking resistant* if any unseen past output remains indistinguishable from a random sequence even to an adversary who obtains the internal state of a generator. A complementary notion of *prediction resistance* ensures that the unseen future output remains indistinguishable as well. It has been conjectured that PRBGs based on a one-way function achieve backtracking resistance while prediction resistance is ensured by providing additional entropy [1].

In this paper we focus on the importance of backtracking resistance to the security of pseudorandom bit generators given the more general security model. We make the following contributions. First, we give a motivational example to bring attention to the importance of backtracking resistance. Then, we provide a formal treatment of this property. We prove that one-wayness of the next-state function is a necessary but not a sufficient condition to ensure backtracking resistance. Finally, we formulate a number of important questions regarding backtracking resistance for future exploration.

The rest of the paper is structured as follows. Section 2 discusses the importance of backtracking resistance to security of PRBGs. Section 3 gives the required definitions. Sections 4 and 5 provide proofs of our claims. Section 6 outlines future work and Section 7 concludes.

2 Backtracking Resistance

In this section, we give a motivational story, informally discuss backtracking resistance, specify the security model, present the current approach to backtracking resistance, and review related work.

2.1 Motivational Story

Alice and Bob exchange messages on a daily basis. Because of their sensitive nature, they decide to encrypt them and use a fresh encryption key every day. Initially, they thought to use a public key encryption scheme but quickly got tired of the need to exchange their public keys so frequently. Then, Alice suggested they each use a cryptographically secure pseudorandom bit generator initialized with the same secret seed to generate the same key for a symmetric encryption algorithm. After exchanging the secret seed, Alice and Bob each used the pseudorandom bit generator on their respective computers and decided to generate a fresh AES [3] key every day.

This approach has been working very well. Unfortunately, one day Alice's device was stolen and ended up in Eve's hands. Eve has been secretly intercepting messages exchanged between Alice and Bob but did not have much luck opening them because AES resists Eve's cryptanalytic skills. Since she could not break the encryption scheme, she decided to try to recover the encryption keys from the pseudorandom bit generator. Because she was in possession of Alice's computer, she knew the generator used and its internal state after generating the last key. Her next step was to try to even is case use this information to recover previously generated bits, thereby giving her full access to all messages previously encrypted.

Alice and Bob knew that they had nothing to worry about only if their pseudorandom bit generator was backtracking resistant. If it was, then Eve would not be able to backtrack to previous output bits ensuring the secrecy of past messages. However, if it was not, then Eve would be enjoying their private correspondence after decrypting their messages using keys recovered from the previous output of the generator.

The goal of the above example is to illustrate the need to consider the security of pseudorandom generators under a more general security model in which to an adversary has the ability to compromise the generator itself and obtain its entire internal state.

2.2 New Approach to PRBG Security

Pseudorandom generators are rarely, if ever, used on their own; they are mostly used as a part of more complex systems and directly affect the security properties offered by those systems. In practice, most people only consider the quality of the output a generator produces as an indicator of whether or not they can deem the generator “secure”. However, there is more to the security of pseudorandom generators than computational indistinguishability of its output.

As illustrated in our story, the security of the system Alice came up with might be at risk because of the properties (or lack of thereof) of the generator they chose. She surely knew to use a cryptographically secure generator to ensure that Eve cannot learn about the encryption keys based on any transmissions she was able to intercept. However, it was not entirely obvious to consider how a compromise of the system a generator is running on may aid an adversary in breaking the (past) security of the entire system.

We believe that considering security of pseudorandom generators with respect to more powerful adversaries is more important than ever. Nowadays, not only well-secured dedicated computers run sensitive application requiring strong security guarantees; portable devices such as laptops, tablets, or even mobile phones perform similar tasks as well. Such devices can be easily lost or stolen, creating a very plausible scenario for state compromise attacks. The approach to PRBG security must reflect that.

2.3 Importance of Backtracking Resistance

The backtracking resistance property ensures that the output generated prior to a compromise of the generator remains secure. In other words, if a generator is backtracking resistant, then obtaining its internal state does not provide additional useful information about the past output and therefore is not an advantage in breaking the “past security” of an application that employs such a generator.

Backtracking resistance is critical to applications requiring long-term security of past outputs. Backtracking resistance offers protection similar to but more general than backward secrecy, which ensures that past keys remain secure even in case of a compromise of future keys. Key generation and key renewal schemes are two obvious examples of tasks in which backtracking resistance is extremely important if not essential.

2.4 Security Model

Traditionally, when considering the security of a pseudorandom generator, an adversary is assumed to be outside of the system on which an instantiation of a pseudorandom generator is running. Such an adversary might observe (some) output of a generator, but never its internal state, while trying to either distinguish the generator’s output from truly random output or predict future output based on previously seen output.

In this paper we emphasize the need to consider a more general security model which permits a more powerful adversary whose capabilities include compromising a device (or a system) on which a generator is running. In such a scenario, an adversary obtains full knowledge of the internal state of the generator at the time of the compromise. In addition, the adversary might have additional knowledge about the previous output generated prior to compromising the system.

Therefore, in backtracking attacks, we shall always consider an adversary whose knowledge about the generator consists of its internal state at the time of compromise and possibly a subset of the output generated prior to an attack.

The notion of the internal state of a generator is crucial to understanding backtracking attacks and devising backtracking resistant generators.

2.5 Internal State of a Generator

The need to consider the internal state as an important and security-related element of a generator arises from the incremental way in which pseudorandom generators work in practice. Once a generator is instantiated, it produces output bits whenever requested based on its current internal state. It then updates its internal state. Informally speaking, an internal state of a generator consists of all information required for the generator to produce future output bits and internal states.

An internal state may include a varying amount of information and be specific to a particular type of a generator or even a particular implementation. In the formal treatment of the backtracking property (Section 3), we assume that the internal state consists of secret information that the generator acts upon. In practice, additional secret and non-secret information may be stored along with the state. Retaining unnecessary information may cause a generator to succumb to state compromise attacks even though the underlying generator algorithm is backtracking resistant. Therefore, the information kept around as a part of an internal state should be carefully analyzed and be a part of the generator's security considerations.

2.5.1 Atomicity of Operations

A generator goes through different stages while (incrementally) producing output bits. At any given stage i, a pseudorandom generator has an internal state s_i. It produces a pseudorandom bit r_i using the output function and goes from the current state s_i to a new state s_{i+1} using the next-state function. During this transition, the current (s_i) and future (s_{i+1}) states and the current (r_i) and future (r_{i+1}) output bits are simultaneously available. We assume that the operation of computing the next-state and output functions is atomic; otherwise, breaking backtracking resistance is trivial because of the additional information available during the transition.

To justify the reasonableness of this assumption, imagine that a system goes through active and idle periods. During an active period, the generator is in use and continuously produces new output bits and updates its state. During an idle period, the generator is not in use and only stores its current state. In our previous example, Alice only requests bits from the generator (putting it into an active state) when she is physically present at her computer and in need of a new key to encrypt a message. Therefore, when Eve accesses Alice's computer, the system is in an idle state, ensuring that the only information available is a single internal state.

Voting machines are another example. While in use, they are physically guarded in a voting location, but when stored, they are more vulnerable to physical attacks. Smart phones, tablets, and other personal devices would exhibit the same "active-or-idle" usage pattern.

2.6 Achieving Backtracking Resistance

To achieve backtracking resistance, some restrictions must be placed on a generator, more specifically on the initial-state, next-state, and/or output functions. The goal is to limit the information carried in the internal state in a way that allows producing future output without revealing information about past output.

It has been suggested in [1] that requiring the underlying generation algorithm to be one-way is a sufficient condition to ensure backtracking resistance.

Backtracking resistance is often thought of as an inability of an adversary to differentiate previously unseen output of a generator produced prior to stage i from random output after compromising an internal state s_i at some stage i. One approach to differentiate such outputs is to retrieve previous states of the generator from the current one. This way an adversary can himself generate the previous output which would clearly translate into his ability to differentiate the outputs.

Therefore, requiring the underlying algorithm to be one-way is necessary to prevent inverting the current state and (trivially) breaking the backtracking resistance. However, it is not sufficient. The state may carry enough information to retrieve past outputs without the need to fully recover past states.

2.7 Related Work

Backtracking resistance of pseudorandom generators has not been widely studied. The first mention and discussion of this property is due to Kelsey et. al. [5]. The notion of backtracking resistance has been incorporated into the NIST Special Publication 800-90 “Recommendation for Random Number Generation Using Deterministic Random Bit Generators” [1].

3 Definitions

This section provides the necessary definitions needed for constructions given in Sections 4 and 5.

3.1 Incremental Pseudorandom Bit Generator

A *pseudorandom bit generator (PRBG)* (sometimes called a deterministic random bit generator (DRBG) [1]), is a deterministic polynomial-time algorithm G that maps a seed z of length m to an output string r of length $n > m$. To be *cryptographically secure*, the ensemble of output strings $G(U_m)$ should be computationally indistinguishable from U_n, where U_m and U_n are the uniform distributions over strings of length m and n, respectively.[1]

In practice, pseudorandom bits are generated on demand and the output string is built incrementally.

Definition 1. *An* incremental PRBG (iPRBG) *G is defined by a tuple $(m, N, S, \iota, \delta, \rho)$. m is the length of the seed, N is the length of the output sequence, S is a finite set of* states *of the generator, and ι, δ, and ρ are functions. The* initial-state function *ι maps a seed to an initial state $s_0 \in S$. The* next-state function *δ is a permutation on S. The output function ρ maps S to* $\{0, 1\}$.

[1] The notion of computational indistinguishability, introduced by Yao [7], means that any probabilistic polynomial-time algorithm behaves essentially the same whether supplied with inputs from the one distribution or the other. See Goldreich [4] for further details.

Starting with a seed $z \in \{0,1\}^m$, G computes a sequence of states $s_0, s_1, \ldots, s_N$ and a sequence of bits $r_0 r_1 \ldots r_{N-1}$, where $s_0 = \iota(z)$, $s_k = \delta(s_{k-1})$ for $1 \leq k \leq N$, and $r_k = \rho(s_k)$ for $0 \leq k \leq N-1$. The output $G(z) = r_0 r_1 \ldots r_{N-1}$.

When the seed z is chosen according to the uniform distribution U_m, G induces a distribution L_n^G on the state s_n at stage n. Namely, $L_n^G = \delta^n(\iota(U_m))$.

3.2 Backtracking Resistance

A backtracking attack applies to an iPRBG G whose internal state has been compromised. We assume that an adversary compromises the internal state of G at stage n after r_{n-1} has been produced and the internal state has been replaced by s_n. Informally, we say that G resists backtracking at stage n if the bit string $r_0 \ldots r_{n-1}$ is computationally indistinguishable from a truly random string of the same length even given the state s_n. G resists backtracking if it resists backtracking at any stage. This implies that a polynomially-bounded adversary has only a negligible advantage at guessing any of the bits produced by G before the attack.

Definition 2. *(Backtracking Resistance) Let $G = (m, N, S, \iota, \delta, \rho)$ be an incremental PRBG, and let $0 \leq n \leq N$. Consider a run $G(z)$ on seed z. An adversary who corrupts G at stage n receives the pair $(r, s) \in \{0,1\}^n \times S$, where $r = r_0 r_1 \ldots r_{n-1}$ and $s = s_n$. When z is chosen randomly, this gives rise to a probability distribution $G_n^*(U_m)$, where U_m is the uniform distribution over seeds. We say that G is* backtracking resistant *if $G_n^*(U_m)$ is computationally indistinguishable from the distribution $U_n \times L_n^G$ for all $0 \leq n \leq N$.*

Intuitively, the only difference between cryptographic security and backtracking resistance is that in the latter case, the distinguishing judge is given either the output and state of G up to a given stage, or she is given a random output and random state, where the output is uniformly distributed and the state is correctly distributed for that same stage. Backtracking resistance implies that knowing the state of the generator gives the adversary no useful information about the previous output bits.

4 One-wayness is Necessary

In this section we show that one-wayness of the next-state function is a necessary condition to achieve backtracking resistance.

Theorem 1. *Let $G = (m, N, S, \iota, \delta, \rho)$ be a backtracking resistant iPRBG. Then δ is a one-way function.*

Proof. Let $G = (m, N, S, \iota, \delta, \rho)$ be a backtracking resistant iPRBG. We show that δ is a one-way function.

Assume to the contrary that δ is not a one-way function. Then there exists a probabilistic polynomial-time algorithm *Inv* for inverting δ with a non-negligible success probability. In greater detail, when given a uniformly distributed random state $s \in S$,

$$Prob[\delta(\mathit{Inv}(\delta(s)) = \delta(s)]$$

is non-negligible.

We construct a polynomial-time judge $\mathcal{J}$. Given inputs $r \in \{0,1\}^n$ and $s \in S$, the judge computes $\hat{s} = \mathit{Inv}(s)$. If $\delta(\hat{s}) = s$, then she compares $\rho(\hat{s})$ with r_{n-1} and outputs 1 if they are equal and 0 otherwise. If $\delta(\hat{s}) \neq s$, she outputs a random uniform bit b.

Suppose (r, s) results from a random run of G. If $\hat{s}$ is in the preimage of s under δ, then $\mathcal{J}$ outputs 1. Otherwise, $\mathcal{J}$ outputs 1 or 0 with equal probability. Since *Inv* succeeds with non-negligible success probability ϵ, then $\mathcal{J}$ outputs 1 with probability $1/2 + \epsilon/2$. On the other hand, if r_{n-1} is a uniformly and independently chosen random bit, then $\mathcal{J}$ outputs 1 with probability exactly $1/2$. Hence, J distinguishes the pseudorandom output from truly random with advantage $\epsilon/2$. This implies that G is not backtracking resistant, contradicting the assumption that it is.

Hence, δ must be a one-way function. □

5 One-wayness is Not Sufficient

In this section we show that one-wayness of the next-state function is not a sufficient condition to ensure backtracking resistance.

Theorem 2. *Given an iPRBG $G = (m, N, S, \iota, \delta, \rho)$ where δ is a one-way permutation, there is an iPRBG $H = (m, N, \hat{S}, \hat{\iota}, \hat{\delta}, \hat{\rho})$ such that $\hat{\delta}$ is a one-way permutation, $H(z) = G(z)$ for all seeds z, and H is not backtracking resistant.*

This theorem implies that if G is also cryptographically secure, then so is H. This is because cryptographic security depends only on the generated output sequences and not on the internal structure of the generator itself, so the output sequences of H and G are the same.

The proof is structured as follows. Starting from an iPRBG $G = (m, N, S, \iota, \delta, \rho)$ based on a one-way permutation δ, we construct an iPRBG $H = (m, N, \hat{S}, \hat{\iota}, \hat{\delta}, \hat{\rho})$ which is the same as G except that we augment the state with an extra bit of information that does not affect the output sequence. We show that $\hat{\delta}$ is a one-way permutation, but H is not backtracking resistant. Therefore, $\hat{\delta}$'s one-wayness is not sufficient to ensure backtracking resistance.

5.1 The Generator H

Assume that an iPRBG $G = (m, N, S, \iota, \delta, \rho)$ exists with δ a one-way permutation. Define $H = (m, N, \hat{S}, \hat{\iota}, \hat{\delta}, \hat{\rho})$, as follows:

$$\begin{aligned} \hat{S} &= S \times \{0,1\} \\ \hat{\iota}(z) &= (\iota(z), 0) \\ \hat{\delta}(s,b) &= (\delta(s), b \oplus \rho(s)) \\ \hat{\rho}(s,b) &= \rho(s) \end{aligned}$$

In the following, s_k and r_k refer to G, and $\hat{s}_k$ and $\hat{r}_k$ refer to H.

Fact 1. *Let $0 \leq k \leq N$, let $b_0 = 0$, and let $b_k = r_0 \oplus r_1 \oplus \ldots \oplus r_{k-1}$ for $1 \leq k \leq N$. Then*

1. *$\hat{s}_k = (s_k, b_k)$;*
2. *$\hat{r}_k = r_k$;*
3. *$H(z) = G(z)$ for all seeds z.*

Proof. Immediate by induction on k. □

To know that H is an iPRBG, we must verify that $\hat{\delta}$ is a permutation on $\hat{S}$.

Lemma 1. *$\hat{\delta}$ is a permutation on $\hat{S}$.*

Proof. We show that $\hat{\delta}$ is one-to-one and onto.

Suppose $\hat{\delta}(s', b') = (s, b) = \hat{\delta}(s'', b'')$. Then $s' = s''$ since δ is one-to-one and $\delta(s') = s = \delta(s'')$. Moreover, $\rho(s') \oplus b' = b = \rho(s'') \oplus b''$. Since $s' = s''$, then $\rho(s') = \rho(s'')$ so also $b' = b''$. Hence, $\hat{\delta}$ is one-to-one.

Now let $(s, b) \in \hat{S}$. Since δ is a permutation, there exists s' such that $\delta(s') = s$. Let $b' = \rho(s') \oplus b$. Then $\hat{\delta}(s', b') = (s, b)$, so $\hat{\delta}$ is onto. □

Lemma 2. *H is not backtracking resistant.*

Proof. We construct a polynomial-time judge $\mathcal{J}$. On input $(r, (s, b))$, where $r = r_0, r_1, \ldots, r_{n-1}$, $\mathcal{J}$ outputs $r_0 \oplus \ldots r_{n-1} \oplus b$.

Let $1 \leq n \leq N$ and suppose a random run of H results in output $r = r_0, r_1, \ldots, r_{n-1}$ and state (s_n, b_n). Then $\mathcal{J}$ on input $(r, (s_n, b_n))$ outputs 0 since $b_n = r_0 \oplus r_1 \oplus \ldots \oplus r_{n-1}$. On the other hand, if $(r, (s, b))$ is chosen according to the distribution $U_n \times L_n^H$, then $\mathcal{J}$ outputs 1 with probability exactly $1/2$. Hence, J distinguishes the pseudorandom output from truly random with advantage $1/2$. This implies that H is not backtracking resistant. □

Lemma 3. *The next-state function $\hat{\delta}$ is one-way.*

Proof. Assume that $\hat{\delta}$ is not one-way. Then there exists a probabilistic polynomial-time algorithm $\widehat{\mathit{Inv}}$ that inverts $\hat{\delta}$ with non-negligible success probability, We construct a probabilistic polynomial-time algorithm *Inv* that inverts δ with non-negligible success probability.

Given a state $s \in S$, *Inv* computes $\widehat{\mathit{Inv}}(s, 0)$ and $\widehat{\mathit{Inv}}(s, 1)$ to see if either results in an inverse of $\hat{\delta}$. If an inverse (s', b') is found, then *Inv* outputs s' (which is a δ-inverse of s). Algorithm 1 describes this strategy in greater detail.

Algorithm 1 Algorithm *Inv*

```
Input: $s \in S$
Output: $s'$ s.t. $\delta(s') = s$

1. Compute: $(s'_0, b'_0) = \widehat{Inv}(s, 0)$ and $(s'_1, b'_1) = \widehat{Inv}(s, 1)$.

2. If $\delta(s'_0) = s$, then output $s'_0$.
   Else if $\delta(s'_1) = s$, then output $s'_1$.
   Else output "fail".
```

Inv's advantage in inverting δ is at least as great as the advantage of $\widehat{\mathit{Inv}}$. Since $\widehat{\mathit{Inv}}$ is a polynomial-time algorithm that succeeds with non-negligible probability, then *Inv* is also polynomial-time and succeeds with non-negligible probability. This is impossible since δ is one-way.

Therefore, $\widehat{\mathit{Inv}}$ does not exist and $\hat{\delta}$ is one-way. □

5.2 Proof of Theorem 2

Proof. Assuming an iPRBG G based on a one-way next-state permutation exists, we constructed a new iPRBG H in Section 5.1 such that $H(z) = G(z)$ for all seeds z. From Lemma 2, H is not backtracking resistant, but nevertheless, Lemmas 1 and 3 show that H is based on a one-way next-state permutation. □

6 Future Work

We do know how to implement iPRBGs that have a very weak form of backtracking resistance. Namely, if the adversary compromises the generator and gets only the current state s_n, then he cannot distinguish this case from a random pair $(r, s) \in U_n \times L_n^G$ with a non-negligible advantage. This is true of the Blum Blum Shub (BBS) [2] generator based on the assumed difficulty of the quadratic residuosity problem. Assuming the contrary, then an adversary for predicting r_{n-1} from s_n would give a means for constructing a quadratic residue tester with non-negligible success probability. We omit the details here.

We leave open the major problem of whether or not a cryptographically secure iPRBG exists, even assuming that one-way functions and cryptographically secure PRNGs exist. The seed in an iPRBG determines both the output sequence r and the final state s_n. Backtracking resistance requires showing the computational independence of r and s_n, something that the authors do not know how to do except in the very special case mentioned above.

The fact that this problem may be more difficult than is at first apparent can be seen from the following contrived example. Suppose we start with a cryptographically secure iPRBG G with 160-bit long states. We construct a new generator H that is identical to G except for, say, the first 160 bits of output. The way H generates those bits is to run G to the end to obtain s_N, then continue running G to generate the next 160 bits r'. Now, the first 160 bits of output from H are $r' \oplus s_0$. We have no a priori reason to believe that H is cryptographically secure because of the strange way those first 160 bits are generated, yet it seems intuitive that they should still look perfectly "random". Nevertheless, H is not backtracking resistant in a very strong sense – if an adversary gets both the first 160 output bits and the final state s_N of the generator, he can compute r', xor it with the first 160 output bits, and obtain s_0, completely breaking the generator.

7 Conclusions

In this paper we have explored a new approach to security of pseudorandom bit generators which involves a more general security model in which an adversary can compromise the internal state of a generator. We have formally defined an incremental pseudorandom bit generator (iPRBG) to better reflect the way pseudorandom bits are generated in practice, and we have defined the backtracking resistance property with respect to it. We have also provided proofs that one-wayness of the next-state function is a necessary but not a sufficient condition to ensure backtracking resistance. We have only begun to explore the complex notion of backtracking resistance and different ways of achieving it. We leave to future work the important question of whether backtracking resistance is achievable, either in theory or in practice.

References

[1] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. Technical report, National Institute of Standards and Technology, 2012.

[2] Lenore Blum, Manuel Blum, and Michael Shub. A simple unpredictable pseudo random number generator. *SIAM J. Comput.*, 15(2):364–383, May 1986.

[3] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Verlag, Berlin, Heidelberg, New York, 2002.

[4] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, 2001.

[5] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In *FAST SOFTWARE ENCRYPTION, FIFTH INTERNATIONAL PROCEEDINGS*, pages 168–188. Springer-Verlag, 1998.

[6] Bruce Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[7] Andrew C. Yao. Theory and application of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 80–91, Washington, DC, USA, 1982. IEEE Computer Society.