仮想環境下における擬似乱数生成器 -Whirlwind- の安全性に関する考察

佐藤 洋之 †　　宮地 充子 ‡　　蘇 春華 ‡

† 北陸先端科学技術大学院大学 情報科学研究科
923-1292 石川県能美市旭台 1-1
{ s1310028,miyaji,chsu }@jaist.ac.jp

あらまし 擬似乱数生成器（PRNG）が満たすべき要件は，出力値が第三者から見てランダムに見えること（Resilience），内部状態が漏洩しても過去の出力値が予測不能であること（Forward security）や未来の出力値が予測不能であること（Backward security）である．加えて，PRNG の内部状態が暴露されても，出力値が一様ランダムな乱数と見分けることができないこと（Robustness）の概念が Boaz らによって提案された [1]．これを満たす PRNG のモデル化および安全性の証明が注目されている．しかし，近年，仮想環境下において，スナップショットの使用による PRNG の新たな脆弱性が発表された [5]．本研究では，上記の要件を満たすように新たに設計された PRNG（Adam らによる提案 [6]）の安全性に関する考察を行う．
キーワード 擬似乱数生成器（PRNG）, 仮想化, Yarrow-160, Whirlwind，原像攻撃

A Study on the safety of the pseudo-random number generator -Whirlwind- in the virtual environment

Hiroyuki Sato†　　Atsuko Miyaji‡　　SU, Chunhua‡

†Japan Advanced Institute of Science and Technology.
1-1, Asahidai, Nomi-shi, Ishikawa 923-1292, JAPAN
{ s1310028,miyaji,chsu }@jaist.ac.jp

Abstract Requirements that the pseudo-random number generator (PRNG) should meet are as follows. They are Resilience, Forward security, Backward security and Robustness[1]. Proof of safety and modeling of PRNG to meet these requirements has been attracting attention. However, in recent years, in the virtual environment, a new vulnerability of PRNG through the use of snapshots has been announced[5]. In this work, we make a study on the safety of the PRNG -Whirlwind- in the virtual environment[6].
Key words Pseudo Random Number Generator（PRNG）, Virtualization, Yarrow-160, Whirlwind，Preimage attack

1 はじめに

近年の特徴である情報化社会では，ネットワークを通じた安全なデータ通信が求められる．しかし，通信路上を流れるデータは盗聴される可能性があり，さらに悪意のある第三者によりデータの改ざん等がなされる危険性がある．こ

のような状況下において，暗号理論・技術の果たす役割は極めて重要である．通信内容の秘匿，認証を行うための基礎技術として用いられることから，安全性を担保するために必要不可欠な項目の一つに，擬似乱数の安全性が挙げられる．

その擬似乱数を生成する PRNG が満たすべき要件は，以下のとおりである．

1. Resilience : 出力値が第三者から見てランダムに見えること
2. Forward security : 内部状態が漏洩しても過去の出力値が予測不能であること
3. Backward security : 内部状態が漏洩しても未来の出力値が予測不能であること
4. Robustness : 内部状態が漏洩しても出力値が一様ランダムな乱数と見分けることができないこと

この概念は，Boaz らによって提案された [1]．また，出力に関しては，明確に定義されていないものの，NIST が公表している統計的な乱数検定に合格することが標準とされている [2]．これらを満たす PRNG のモデル化および安全性の証明が注目されている．しかし，現在使用されている PRNG において，上述の要件を全て満たすとされるモデルの報告は確認できていない．既存研究において，Linux や Windows の PRNG に関する脆弱性が報告されているのが現状である [3]．さらに，近年では，高信頼性，リソース最適化や運用効率化の観点から仮想化が進んでいるが，Ristenpart らによって仮想環境下における PRNG の脆弱性（VM Reset Vulnerability）が発表された [5]．その対策を考慮した PRNG（Whirlwind）が，Adam らによって提案された [6]．

本研究では，Guo らによる原像攻撃 [7] をもとに，Whirlwind の安全性に関する考察を行った．そして，全数探索よりも少ない計算量で，ハッシュ関数に対する入力が推測可能であることが確認できた．

本論文の構成は，次のとおりである．まず，2 章で擬似乱数生成器および仮想化について説明し，3 章で既存研究の紹介を行う．次に，4 章で Whirlwind を考察するうえで使用した原像攻撃について説明し，5 章で Whirlwind に対する考察を行う．そして，6 章で結論を述べる．

2 準備

2.1 擬似乱数生成器（PRNG）

擬似乱数とは，$\{0,1\}^n$ 上の確率分布:X_n が一様分布 : $\{U_n\}$ と識別不可能であることを意味する．ここで，$\{U_n\}$ が $\{0,1\}^n$ 上で一様分布であるとは，$\forall a \in \{0,1\}^n$ に対して，$Pr_{x\in\{U_n\}}[x = a] = \frac{1}{2^n}$ となることである．

擬似乱数の厳密な定義は以下のとおりである．

$\{0,1\}^n$ 上の確率分布 : X_n が擬似乱数であるとは，任意の多項式時間アルゴリズム A に対して，$\exists\varepsilon_0 > 0$ が存在し，$\varepsilon_0 > \forall\varepsilon > 0$ に対して，以下を満たすことである．

$$|Pr_{t\in X_n}[A(t) = 1] - Pr_{t\in U_n}[A(t) = 1]| < \varepsilon \tag{1}$$

その擬似乱数を生成するのが PRNG である．

2.2 仮想化（Virtualization）

プロセッサやメモリ，ディスク，通信回線など，コンピュータシステムを構成する資源（および，それらの組み合わせ）を，物理的構成に拠らず柔軟に分割したり統合したりすることをいう．

仮想化技術における仮想マシン（Virtual Machine:VM）は，OS が動作する実際のコンピュータをソフトウェア（ハイパーバイザー）によって仮想的に構築したものである．1 台のコンピュータを複数の VM に分割することで，複数の利用者が同時に利用したり，異なる OS を並列に実行させたりすることができる．仮想化による主な利点は，高信頼性，運用効率化，リソース最適化等が挙げられる．

3　PRNG に関する既存研究

3.1　Yarrow-160[4]

1990 年代，PRNG に対する攻撃が注目されるようになった．そこで，Schneier らによって提案された PRNG が Yarrow-160 であり，以下のような特徴がある．

1. 2 つのエントロピープールの設定（fast，slow）
2. 各プールにおいて，異なる更新間隔を設定
3. 内部状態の更新
4. 既存の暗号化技術を使用
5. 各プールの容量：160 ビット

1. では，エントロピーを直接入力するのではなく，ハッシュ関数（SHA-1）を使用し，その出力をどちらか一方のプールに格納する．ここで，2 つのプールを設定することで，入力データを盗聴可能な攻撃者に対する対策がなされている．

2. では，更新間隔を別々にすることで，各プールに格納中のエントロピーを推測するのを困難にしている．

3. では，出力したブロック数をカウントし，ある値：p_g（$1 \leq p_g \leq 2^{n/3}$）に到達した場合，内部状態の更新を実施する（例：$p_g = 10$）．この値は，暗号化に 3-DES を使用していることから，バースデーパラドックスによる衝突を避けるために設定されている．

4. では，既存の暗号化技術を使用していることから，他のブロック暗号に変更することが可能である．

しかし，次のような問題点が指摘されている．一つ目は，5. のとおり，各プールで高々160 ビットのエントロピーしか格納できないこと．二つ目は，一度プール内のエントロピーが使用されると，0 にリセットするという設定のため，頻繁に擬似乱数生成の要求がなされた場合，各プールのエントロピーが不足し，内部状態の更新ができなくなるということである．

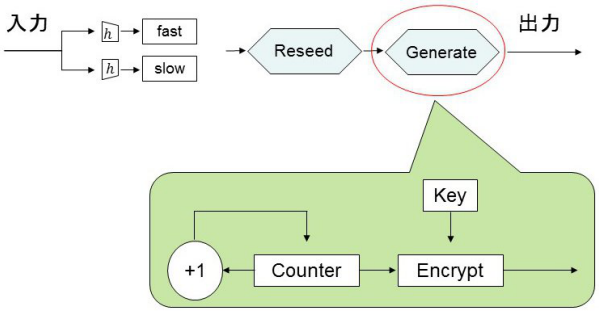

図 1：　Yarrow-160

3.2　VM Reset Vulnerability[5]

VM Reset Vulnerability は Ristenpart らによって発見された脆弱性である．これは，VM 上で擬似乱数を生成する場合，前回の遮断時にサーバーに保存されたスナップショットを再利用することに起因する．入力ソースが限られている VM 上では，再起動時は特にエントロピーが不足しがちなため，ある時点での全状態を保存したスナップショット（メモリ，CPU レジスター等を含む）が使用される．ここで以前のスナップショットをそのまま使うという設定の場合，攻撃者がある時点 i のスナップショットを入手したとすると，$i-1$ での擬似乱数が推測可能となる．Adam らもこの脆弱性を実験的に発見した [6]．

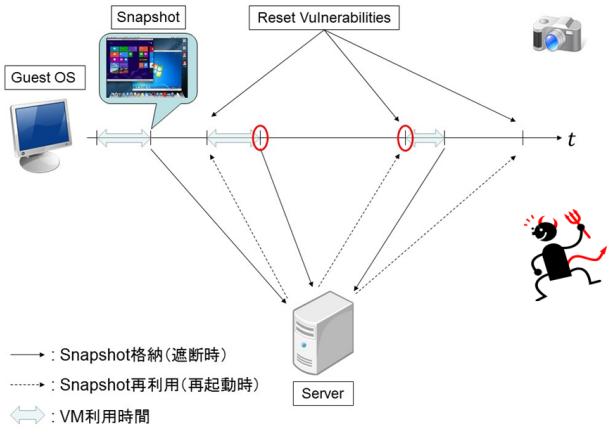

図 2：　VM Reset Vulnerability

3.3 Whirlwind[6]

[4] および [5] をもとにして，Adam らによって新たに提案された PRNG であり，仮想環境下においても Robustness を満たすとされる．主に Yarrow-160 をもとに改良がなされており，以下のような特徴がある．

1. 複数の入力を想定
2. Boot Security を担保
3. 2 つのエントロピープールの設定（fast，slow）
4. 各プールにおいて，異なる更新間隔を設定
5. 内部状態の更新
6. 既存の暗号化技術を使用
7. バッファ容量：1024 ビット，各プールの容量：512 ビット

1. では，複数の入力を使用し，それらを統合している．（Algorithm 1，2）

2. では，VM Reset Vulnerability を考慮し，以下の Algorithm 2 で対策がなされている．

3. では，Yarrow-160 と同様，ハッシュ関数（ここでは SHA-2）を使用し，その出力をどちらか一方のプールに格納する．

4. では，図 3 のように分岐点（$d = 10$）を設定している．これにより，各プールへの入力回数に偏りを持たせている．つまり，この場合は，10 回の入力のうち 9 回は S_{fast} プールへ分岐されることになる．また，S_{slow} プールでは更新間隔をある入力回数時に設定（$p = 50$）することで，格納中のエントロピーを推測するのをさらに困難にしている．この場合は，S'_{slow} に 50 回の入力後，初めて S_{slow} にエントロピーが格納されることになる．

5. では，擬似乱数出力の要求毎に実施する．（Algorithm 1，2）

7. では，SHA-2 に対応可能な構造に設定されている．

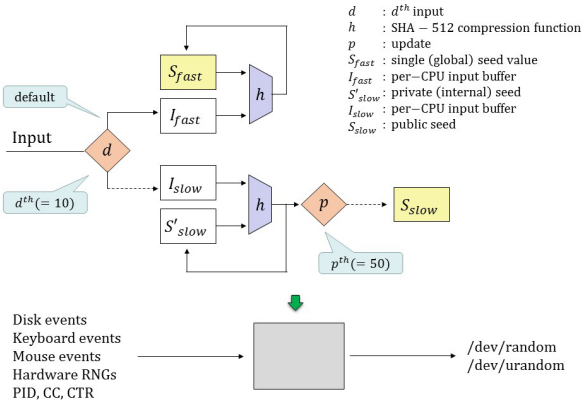

図 3： Whirlwind

3.3.1 内部状態更新のアルゴリズム

Algorithm 1 ww_generate_bytes(b)

$s_1 \leftarrow S_{fast}$
$s_2 \leftarrow S_{slow}$
$t \leftarrow \lceil 8b/n \rceil$
$ctr \leftarrow$ atomic_inc$(Ctr, t) - t$
$hw \leftarrow$ read_hw_random()
ww_add_input()
for $i = 0$ **to** t **do**
  $CC \leftarrow$ get_cycle_counter()
  output[i] $\leftarrow H(3 \parallel s_1 \parallel s_2 \parallel (ctr + i) \parallel CC \parallel P \parallel hw\ )$
end for
ww_add_input()
$S_{fast} \leftarrow h(S_{fast}, 0^{1024})$
return first b bytes of output

Algorithm 2 ww_bootstrap()

for $i \leftarrow 0$ **to** ℓ **do**
  $CC \leftarrow$ get_cycle_counter()
  ww_add_input()
  $k \leftarrow CC \bmod \ell_{max}$
  for $j \leftarrow 0$ **to** k **do**
    $a \leftarrow (j/(CC + 1)) - (a * i)$
  end for
end for

4 Whirlwind への攻撃

以下のような攻撃者を想定し，考察を行う．

4.1 攻撃者の定義

本攻撃では，攻撃者は VM 上でのホストサーバーを想定する．つまり，攻撃開始の時点で，攻撃者は物理的に隔離されたターゲット VM の

スナップショットを読み取り，その時点 i の内部状態に関する情報を入手できるものとする．そして，それをもとに $i-1$ 時点の内部状態を推測し，Forward security を破ることを目標とする．

4.2 攻撃手順

上述の定義から，攻撃者は以下の 1. を入手可能なものとし，以降の手順 2，3 を行う．

1. i 時点の内部状態を入手
   $s1$ および $s2$: 512 ビット
   $ctr+i, CC, P$ および hw : 64 ビット
2. 1. をもとに，$i-1$ 時点の内部状態を推測
3. $i-1$ 時点の不明な情報（$s1, s2$）に対して，原像攻撃（中間一致攻撃）[7] を使用し，推測

2. では，$ctr+i-1$ は容易に判明する（1. で得た $ctr+i$ を使用）．また，CC, P, hw は，ホストサーバーシステム共有であり，隔離されていないことから，攻撃者はそれを追跡記録し，入手可能である．

3. では，その時点で不明な情報は $s1, s2$ のみである．これは，各ゲスト OS のメモリ上にあるため，ホストサーバーには不明である．そのため，以下の Guo らによる原像攻撃 [7] をもとに推測する．

4.3 原像攻撃 [7]

ハッシュ関数の特性である原像計算困難性（一方向性）に対する攻撃のことをいう．原像攻撃に対する安全性は，ハッシュ関数の出力長 : n ビットに対し，2^n の計算量が要求されるため，原像が全数探索より少ない計算量で求めることができれば，攻撃は成功といえる．

原像攻撃 [7] をする際，Guo らは Fact 9.99[8] を使用しており，本研究でも同様に使用する．

4.4 攻撃手順

原像攻撃の一例として，中間一致攻撃がある．攻撃手順は以下のとおりである．

1. 2 つの Chunk（1st chunk，2nd chunk）に分割（10 ラウンドのうち，ラウンド 1 から分割）
2. W_p, W_q を設定
3. 2 つの表 : L_p, L_q を作成
4. L_p, L_q を比較し，部分的な一致を探索
5. 手順（1〜4）を反復
6. 擬原像から原像への変換

2. で，W_p，W_q は以下のように設定する．なお，W_p，W_q 以外のビットは固定する．
W_p : $\{w_0, w_1, w_2, w_4\}$
W_q : $\{w_6, w_7, w_8, w_9\}$
3. で，L_p, L_q は，各 chunk の計算結果
5. で，双方から得られた値を比較し，値が一致する組を探すことで，擬原像を見つける．
6. で，求まった擬原像を原像に変換する．

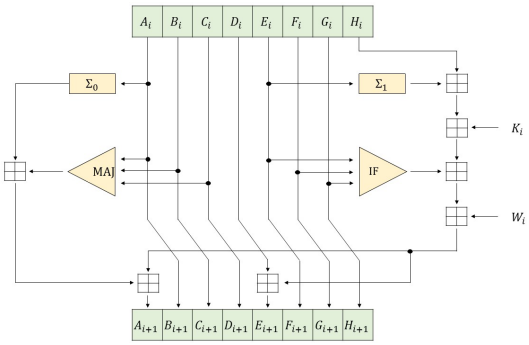

図 4： SHA-512

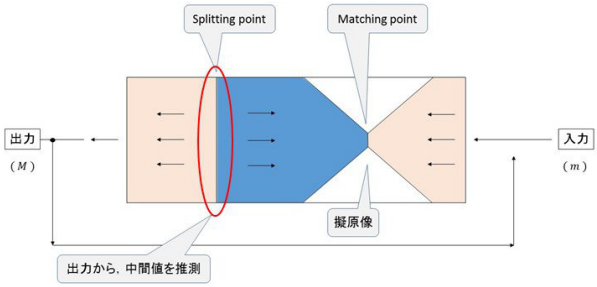

図 5： 中間一致攻撃の概要

5　攻撃に対する考察

5.1　ハッシュ関数に対する攻撃の計算量

目標とする結果が得られるまで，n ビットの出力長を持つハッシュ関数に対し，入力を選択して出力の計算を繰り返す場合，一般に以下の計算量が必要となる．

- 原像攻撃：2^n
- 第二原像攻撃：2^n
- 衝突攻撃：$2^{n/2}$

攻撃の目標は，ある時点 i で原像，第二原像，衝突等を見つけることである．一般に，ハッシュ関数全体の安全性は，その圧縮関数の安全性に強く依存するため，圧縮関数に対する攻撃が重要視されている．以下では，ハッシュ関数の圧縮関数を $H(\cdot)$，異なる 2 つのメッセージを X，X'，ある与えられたハッシュ値を Y とする．また，X_0，V_0 はそれぞれ与えられた入力，初期値とする．

- 原像（Preimage）
  原像を発見するということは，以下となるような X を探すことである．つまり，ハッシュ値が与えられたとき，その値からメッセージを探し出すことである．

$$H(V_0, X) = Y \tag{2}$$

- 擬原像（Pseudo-preimage）
  擬原像を発見するということは，以下を満たすような V, X を計算することである．つまり，与えられたハッシュ値からランダムな初期値とメッセージを計算することである．

$$H(V, X) = Y \tag{3}$$

- 第二原像（2nd preimage）
  第二原像を発見するということは，以下となるような X' を探すことである．つまり，あるメッセージが与えられたとき，それと同じハッシュ値を持つような異なるメッセージを探し出すことである．

$$H(V, X') = H(V_0, X_0) \tag{4}$$

- 衝突（Collision）
  一般に衝突とは，異なる 2 つのメッセージが同じハッシュ値を持つことである．

ここで，第二原像と衝突に関しては，元の値を復元できないので，今回は行わない．

5.2　Fact 9.99[8]

中間一致攻撃で得られた擬原像を用いて，計算量の算出過程において，次の結果（Fact 9.99[8]）を利用する．

もし，ハッシュ値が n ビットとなるような圧縮関数: f が，擬原像攻撃（pseudo-preimage attacks）に対して，2^n の計算量的なセキュリティを保持していないのであれば，原像（preimages）は 2^n よりも少ない計算量で求めることができる．これは，以下の結果による．

もし，擬原像が 2^s の計算量で求めることができるのであれば，$2^{(n+s)/2}$ forward points および $2^{(n-s)/2}$ backward points が利用される．つまり，原像は $2 \cdot 2^{(n+s)/2}$ の計算量で求めることができる．

原像攻撃に必要な計算量の算出過程 [7]：
（例）$n = 256$，$s = 245.32$

$$\begin{aligned} 2 \cdot 2^{(n+s)/2} &= 2 \cdot 2^{(256+245.32)/2} \\ &= 2^{(256+245.32)/2+1} \\ &= 2^{251.66} \end{aligned} \tag{5}$$

5.3　計算量の評価

原像攻撃による計算量（出力長: $n = 512$ ビットの場合）は，以下のとおりである．通常では，ある時点 i の内部状態から，直前の時点 $i-1$ の内部状態を復元するには，以下の全数探索が必要である．

- 全数探索：2^{512}
  （512 ビットのハッシュ関数に対する攻撃）
  ここで，攻撃者にとって不明な情報は s_1，s_2 の 2 つなので，$2 \cdot 2^{512}$ の計算量が必要となる．

Algorithm 1 の output[i] から，Whirlwind の全内部状態を考慮した場合の理論上の安全性は，以下のとおりである．

- 理論上の安全性：2^{1380}
  s_1，s_2：512 ビット
  $ctr + i, CC, P$ および hw：64 ビット

これに対して，我々の提案では，計算量は以下のようになる．

- 原像攻撃：$2 \cdot 10 \cdot 2^{(512+256)/2+1}$
  （512 ビットの s_1 および s_2 に対し，10 ラウンドを 10 回使用した場合）

原像攻撃に必要な計算量の算出過程：
$n = 512$，$s = 256$

$$\begin{aligned} 2 \cdot 10 \cdot 2^{(512+256)/2+1} &= 2 \cdot (2 \cdot 5) \cdot 2^{384+1} \\ &= 2 \cdot 2 \cdot 5 \cdot 2^{385} \\ &= 5 \cdot 2^{385+2} \\ &= 5 \cdot 2^{387} \end{aligned} \tag{6}$$

以上から，本攻撃において必要な計算量（$5 \cdot 2^{387}$）は，理論上必要とされる計算量（2^{512}）よりも少ないことが分かる．

5.4 考察

4.2 節の攻撃手順 3 から，本攻撃において不明な情報は 512 ビットの s_1 および s_2 である．これを利用し，計算量を算出した．

本研究結果では，理論上必要とされる計算量より少ない計算量で攻撃が可能であることが確認でき，PRNG に要求される安全性を備えていないことが判明した．つまり，i 時点での内部状態から，$i-1$ 時点の内部状態が分かるので，攻撃者はその時点で出力される擬似乱数を知ることが可能となる．現時点では，喫緊の問題になるような計算量ではないが，別の攻撃手法で検討し，計算量について比較する必要がある．

ところで，本研究では触れていないが，各分岐点である d, P の値によって，内部状態の更新間隔等に特徴が表れるのではないかと思われる．Adam らの提案内容には，各値の設定に関して明確な定義がなされていない．また，もし各値が改ざん可能であれば，内部状態の更新間隔が操作可能になるのではないかと思われる（例：$d = 1, P = 10^{100}$）．

6 結論

6.1 まとめ

本研究では，VM 上での攻撃者を想定し，Whirlwind の安全性に関する考察を行った．そして，全数探索（2^{512}）よりも少ない計算量で，ハッシュ関数に対する入力が推測可能であることが確認できた．

6.2 今後の課題

今後は別の攻撃手法を用いた検討および比較をするとともに，攻撃に対する改善策についても検討する必要がある．また，考察でも述べたように，別の想定による被攻撃の可能性も考えられる．さらに，Dodis らによって定義されている証明可能安全性 [9] についても議論する余地があると考えられる．

参考文献

[1] Boaz Barak and Halevi, “A model and Architecture for Pseudo-Random Generation and Applications to /dev/random” , *12th Conference on Computer and Communications Security(ACM CCS 2005)* , pages 203-212, 2005.

[2] NIST(National Institute of Standards and Technology)，“A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications”，(NIST SP800-22 / Revision 1a)”，2010.

[3] Leo Dorrendorf, Zvi Gutterman, Benny Pinkas, “Cryptanalysis of the Windows Random Number Generator” , *14th ACM Conference on Computer and Communications Security (ACM CCS 2007)* , pages 476-485, 2007.

[4] J. Kelsey, B. Schneier, and N. Ferguson, “Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator” , *In Selected Areas in Cryptography* , pages 13-33. Springer, 2000.

[5] T. Ristenpart and S. Yilek. “When good randomness goes bad:Virtual machine reset vulnerabilities and hedging deployed cryptography.” *In NDSS.* ISOC, 2010.

[6] Adam Everspaugh, Yan Zhai, Robert Jellinek, Thomas Ristenpart, Michael Swift, “Not-So-Random Numbers in Virtualized Linux and the Whirlwind RNG” , *IEEE Computer Society Technical Committee on Security and Privacy (IEEE S & P 2014)* , 2014.

[7] Guo, J., Matusiewicz, K. “Preimages for Step-Reduced SHA-2”，*ASIACRYPT 2009.* LNCS, vol. 5912, pp. 578-597. Springer, 2009.

[8] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. “Handbook of Applied Cryptography”，CRC Press, 2001.

[9] Yevgeniy Dodis, Adi Shamir, Noah Stephens-Davidowitz, and Daniel Wichs, “How to Eat Your Entropy and Have it Too - Optimal Recovery Strategies for Compromised RNGs” , 2014.

[10] 宮地充子,『代数学から学ぶ暗号理論』，日本評論社，2012.