How to Construct Pseudorandom Permutations from Single Pseudorandom Functions

Josef Pieprzyk *

Department of Computer Science
University College
University of New South Wales
Austarlian Defence Force Academy
Canberra, ACT 2600, AUSTRALIA

Abstract

The paper examines permutation generators which are designed using four rounds of the Data Encryption Standard and a single pseudorandom function. We have proved that such generators are pseudorandom only if the pseudorandom function is used internally at least five times. The proof is given using two different approaches: deterministic and probabilistic. Some cryptographic implications are also discussed.

1 Introduction

Random number generators are commonly used in many different areas of science. However truly random generators create some problems. The most evident is the unreproducibility of generated numbers, so that it is impossible to repeat the same experiment. Another problem is related to the assessment of randomness of generators.

Classical pseudorandom generators are deterministic algorithms that provide numbers which "look" like random ones. As they have well-defined mathematical structures, they can be analysed easily. The basic measurement of classical pseudorandom generators is the similarity of generated numbers to truly random ones. Yao [8] redefined the notion of pseudorandomness in terms of complexity theory. A generator is said to be pseudorandom if it is "indistinguishable" from the truly random one, assuming polynomially bounded computing resources.

Cryptographers have always been interested in how to extend truly random "seeds" (n-bit long) into n^k-bit output strings in such a way that the output is indistinguishable from a truly random string ($k = 2, 3, \cdots$). Blum and Micali [1] introduced the notion of cryptographically strong pseudorandom bit generators (CSB). Levin [3] proved that such generators exist if and only if one-way functions exist. There are several implementations of CSB generators (for details see for example [2]).

*Support for this project was provided in part by TELECOM Australia under the contract number 7027 and by the Australian Research Council under the reference number A48830241.

I.B. Damgard (Ed.): Advances in Cryptology - EUROCRYPT '90, LNCS 473, pp. 140-150, 1991.

Goldreich, Goldwasser and Micali [2] showed that it is possible to construct pseudorandom functions using CSB generators. The next step in the theory of pseudorandomness is due to Luby and Rackoff [4]. They used pseudorandom functions to generate pseudorandom permutations using three rounds of the Data Encryption Standard (DES) and three different pseudorandom functions.

Ohnishi [9] proved that pseudorandom permutations can be obtained using three rounds of the DES and two different pseudorandom functions. During Eurocrypt'89 Zheng, Matsumoto and Imai [9] presented some new results about pseudorandom permutations. They gave a construction of distinguishing circuits for all permutation generators which use three rounds of DES and a single pseudorandom function. They also posed the following problem:

Prove or disprove that from one pseudorandom function, one can obtain in some way a pseudorandom (invertible) permutation applying four rounds of DES.

In this paper we solve the problem.

2 Notations and definitions

Let $I_n = \{0,1\}^n$ be the set of all 2^n binary strings of length n. For $a, b \in I_n$, $a \oplus b$ stands for bit-by-bit exclusive-or of a and b. The set of all functions from I_n to I_n is F_n, i.e.

$$F_n = \{f \mid f : I_n \to I_n\}$$

It consists of 2^{n2^n} elements. If we have two functions $f, g \in F_n$, their composition $f \circ g$ is defined as

$$f \circ g(x) = f(g(x))$$

for all $x \in I_n$. The set of all permutations from I_n to I_n is P_n, i.e.

$$P_n \subset F_n$$

Definition 2.1 *A permutation generator p with a key k of length $l(n)$ ($l(n)$ is a polynomial in n) is a (poly) collection of permutations*

$$p = \{p_n \mid n \in N, p_n \subseteq P_n\}$$

where for each key k of length $l(n)$ and any value of $\alpha \in I_n$, we can compute in polynomial time in n the element $p_{n,k}(\alpha)$ ($p_{n,k} \in p_n$) and N is the set of all positive integers.

Definition 2.2 *A distinguishing circuit C_n is an acyclic circuit which consists of Boolean gates (AND, OR, NOT), constant gates ("0" and "1") and oracle gates. The circuit has one bit output only. Oracle gates accept inputs of length n and generate outputs of the same length. Each oracle gate can be evaluated using some permutation from P_n.*

Definition 2.3 *A family of distinguishing circuits for a permutation generator p is an infinite sequence of circuits $C_{n_1}, C_{n_2}, \cdots$ ($n_1 < n_2 < \cdots$) such that for two constants c_1 and c_2 and for each parameter n, there exits a circuit C_n which has the following properties:*

- *The size of C_n is smaller than n^{c_1} (the size is defined as the number of all connections between gates).*

- *Let $Pr[C_n(P_n)]$ be the probability that the output bit of C_n is one when a permutation is randomly selected from P_n and used to evaluate the oracle gates. Let $Pr[C_n(p_n)]$ be the probability that the output bit of C_n is one when a key k of length $l(n)$ is randomly selected and $p_{n,k} \in p_n$ is used to evaluate the oracle gates. The distinguishing probability for C_n:*

$$| Pr[C_n(P_n)] - Pr[C_n(p_n)] | \geq \frac{1}{n^{c_2}}$$

Definition 2.4 *A permutation generator p is pseudorandom if there is no distinguishing circuit family for p.*

Definition 2.5 *For a function $f \in F_n$, we define the DES-like permutation associated with f as*

$$D_{2n,f}(L,R) = (R \oplus f(L), L)$$

where R and L are n-bit strings ($R, L \in I_n$) and $D_{2n,f} \in P_{2n}$.

Having a sequence of functions $f_1, f_2, \cdots, f_i \in F_n$, we can determine the concatenation of their DES-like permutations ψ and

$$\psi(f_1, f_2, \cdots, f_i) = D_{2n,f_i} \circ D_{2n,f_{i-1}} \circ \cdots \circ D_{2n,f_1}$$

Of course, $\psi(f_1, f_2, \cdots, f_i) \in P_{2n}$.

We can now rephrase the previous results. Luby and Rackoff [4] proved that $\psi(g,h,f)$ is a pseudorandom permutation generator if g, h, f are different pseudorandom functions. Ohnishi [9] showed that both $\psi(g,f,f)$ and $\psi(f,f,g)$ are pseudorandom generators as well, where g, f are different pseudorandom functions. Note that $\psi(f,g,f)$ is not pseudorandom as it is symmetric. Zheng, Matsumto and Imai [9] considered permutations based on a single function $f \in F_n$ and they gave the description of distinguishing circuits for $\psi(f^i, f^j, f^k)$, where f^i is a composition $f \circ f^{i-1}$ and i, j, k are arbitrary positive integers.

Definition 2.6 *A function f is said to be (truly) random (denote this as $f \in_R F_n$) if for a fixed argument $x \in \{0, \cdots, 2^n - 1\}$, $f(x)$ is an independent and uniformly distributed random variable.*

Example 2.1 *Assume that $f \in_R F_2$. Then we have four independent uniformly distributed random variables $f(x_i)$ $(i = 0,1,2,3)$ and*

$$Pr[f(x_i) = j] = \frac{1}{4} \text{ for } j = 0,1,2,3$$

In this paper we are going to prove that $\psi(f,f,f,f^i)$ is pseudorandom permutation generator for $i = 2, 3, \cdots$. To shorten our considerations, we are going to use a reference permutation generator (instead of a truly random one). Its structure should be as close as possible to the structure of $\psi(f,f,f,f^i)$. As $\psi(f,f,g)$ is pseudorandom, we take $\psi(f,f,f,g)$ as the reference one. Obviously, it is pseudorandom (the proof is omitted). Intuitively, taking out the first f from $\psi(f,f,g)$, we get $\psi(f,g)$ which is no longer pseudorandom. Therefore, putting up an additional round for f to $\psi(f,f,g)$, we obtain $\psi(f,f,f,g)$ which must be pseudorandom (the additional round does not introduce symmetry).

The proof of pseudorandomness of $\psi(f,f,f,f^i)$ for $i = 2, 3, \cdots$ will be given in two different ways. In the next Section, it will be carried out using some deterministic arguments. Later on we show that the proof can also be based on some probabilistic ones.

3 Deterministic distinguishing circuits

To illustrate how deterministic distinguishing circuits (DDC) work, consider the following example.

Example 3.1 *Take a permutation generator $\psi(f, f, f^2)$. It is possible to show [9] that it has a DDC family for all n. The structure of such a distinguishing circuit is given in Figure 1.*

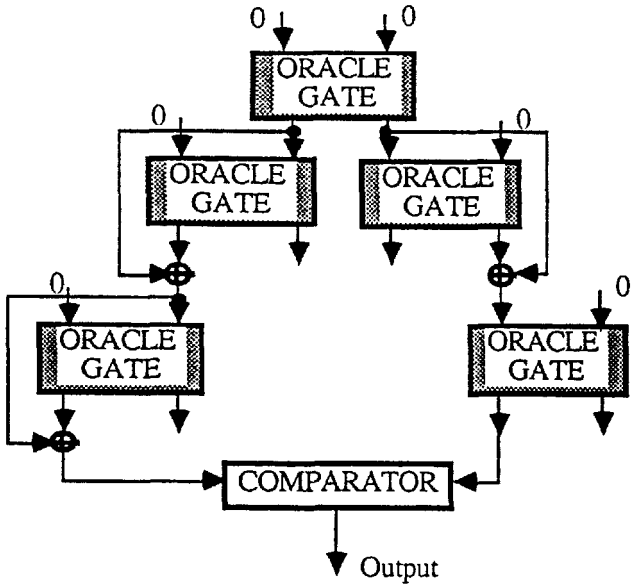

Figure 1. Deterministic distinguishing circuit for $\psi(f, f, f^2)$

The generator produces the following outputs:

$$
\begin{array}{ll}
(f^2(0),\ \ f(0) \oplus f^4(0)) & \textit{for input } (0,0) \\
(f(L \oplus f(0)),\ \ L \oplus f(0) \oplus f^3(L \oplus f(0))) & \textit{for input } (L,0) \\
(R \oplus f^2(R),\ \ f(R) \oplus f^2(R \oplus f^2(R))) & \textit{for input } (0,R)
\end{array}
$$

If $\psi(f, f, f^2)$ is used to evaluate the oracle gates, the comparator inputs can be described as $f^8(0)$. It is interesting what happens if we apply $\psi(f, f, g)$ for oracle gate evaluation. It can be verified that the left hand side input to the comparator does not change its description and is $f^8(0)$, while the right hand side one is represented as $fgfgf^2(0)$.

This example shows us some general properties of a DDC, namely:

P1 As it must work for all possible selections of f, it can be seen as a network which generates the same signal using two different paths. Each path can be expressed by equivalent algebraic formulae.

P2 If there is a DDC, it can always be designed using the oracle gate with input $(0,0)$ as the top one which initiates both paths.

P3 Modulo-2 operation $\oplus$ used outside oracle gates, removes input expression from the output.

Now we will prove the main lemma of the section. Without a loss of generality, we can consider generator $\psi(f,f,f,f^2)$ instead of $\psi(f,f,f,f^i)$ for $i=2,3,\cdots$.

Lemma 3.1 *If $\psi(f,f,f,g)$ is pseudorandom, then there is no deterministic distinguishing circuit for a permutation generator*

$$\psi(f,f,f,f^2)$$

Proof (by contradiction): Assume that there is a DDC for $\psi(f,f,f,f^2)$. It may be represented as shown in Figure 2.

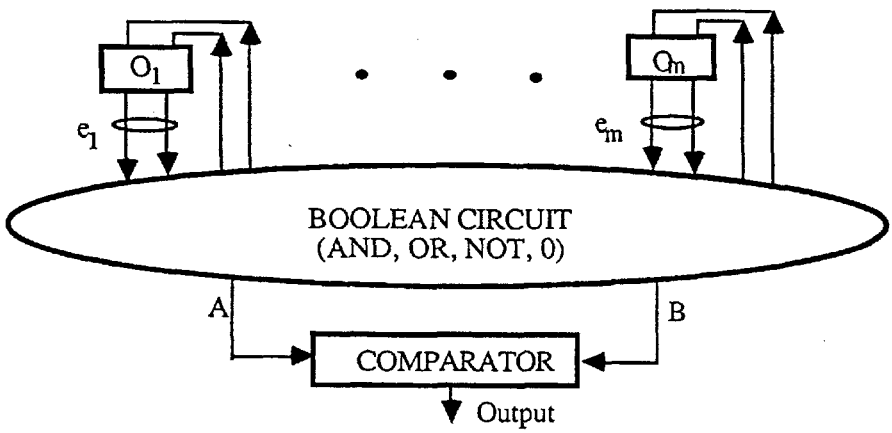

Figure 2. A general diagram of a deterministic distinguishing circuit (OG stands for oracle gate)

In general, for a given input (L,R), a oracle gate generates:

$$(\ L\oplus f(R)\oplus f(R\oplus f(L\oplus f(R))),$$
$$R\oplus f(L\oplus f(R))\oplus f^2(f(R)\oplus f(R\oplus f(L\oplus f(R))))\ ).$$

In particular, for input $(0,0)$, the oracle gate can be described as follows:

$$(\ f(0)\oplus f^3(0),\ \ f^2(0)\oplus f^2(f(0)\oplus f^3(0))\ ).$$

Take the last term of the second output of oracle gate for input (L, R) and denote it as $b(L, R)$, i.e.

$$b(L, R) = f^2(f(R) \oplus f(R \oplus f(L \oplus f(R)))).$$

Note that:

- f^2 in $b(L, R)$ does not commute with the function which is inside the brackets for all L and R (in particular, it does not commute for $L = 0$ and $R = 0$),
- it is impossible to separate a single expression $b(L, R)$ from the right hand side outputs of oracle gates so it always appears as a part of bigger expression which consists of at least two terms. In other words f^2 in $b(L, R)$ does not commute with the function f when $b(L, R)$ is a part of an expression placed inside the brackets of the function f.

It means that inputs to the comparator A and B are equal and can be presented as

$$A(e_1, ..., e_m) = A(e, b_1, ..., b_m)$$

where b_i is $b(L, R)$ for the i-th oracle gate ($i = 1, \cdots m$) and e stands for the rest of oracle gate outputs. f^2 in all expressions $b_1, \cdots, b_m$ does not "mix" and stays in its original form.

Now if the DDC is used for $\psi(f, f, f, g)$, the Boolean circuit is fed with the same vector e and both A and B can be expressed as:

$$A = B = A(e, g_1, \cdots, g_m)$$

where g_i substitutes f^2 in b_i. It means that $\psi(f, f, f, g)$ has a DDC. This is the contradiction which proves the lemma.

□

Theorem 3.1 *If $\psi(f, f, f, g)$ is pseudorandom, then $\psi(f, f, f, f^2)$ is pseudorandom.*

Proof: Luby and Rackoff noted in [4] that any probabilistic distinguishing circuit can be converted to deterministic one. So if a generator has no DDC (see the lemma), it has no probabilistic distinguishing circuit and the generator is pseudorandom.

□

4 Probabilistic distinguishing circuits

In this section we are going to prove that $\psi(f, f, f, f^2)$ is pseudorandom using probabilistic arguments.

Lemma 4.1 *Let C_{2n} be an distinguishing circuit with m oracle gates. Then*

$$| Pr[C_{2n}(\psi(F))] - Pr[C_{2n}(\psi(F, G))] | \leq \frac{9m^3}{4^n}$$

where $Pr[C_{2n}(\psi(F))] = Pr[C_{2n}(\psi(f, f, f, f^2))]$ is the probability that the oracle circuit generates "1" when its oracle gates are evaluated using $\psi(f, f, f, f^2)$ (the function $f \in_R F_n$) and $Pr[C_{2n}(\psi(F, G))] = Pr[C_{2n}(\psi(f, f, f, g))]$ is the probability that the oracle circuit generates "1" when its oracle gates are evaluated using $\psi(f, f, f, g)$ ($f, g \in_R F_n$).

Proof(sketch): Note that both f and g can be considered as two sequences of 2^n independent and uniformly distributed random variables. For two different arguments $a, b \in I_n$, $f(a)$ and $f(b)$ are independent random variables. When the input to an oracle gate is (L, R) $(L, R \in I_n)$, the gate produces the output:

$$f(R) \oplus f(R \oplus f(L \oplus f(R))), \quad f(L \oplus f(R)) \oplus f^2(L \oplus f(R) \oplus f(R \oplus f(L \oplus f(R))))$$

if it is evaluated using $\psi(f, f, f, f^2)$ or generates the output:

$$f(R) \oplus f(R \oplus f(L \oplus f(R))), \quad f(L \oplus f(R)) \oplus g(L \oplus f(R) \oplus f(R \oplus f(L \oplus f(R))))$$

when it is evaluated according to $\psi(f, f, f, g)$ (outputs of oracle gates are given after removing the input L and R, respectively). Of course the first components (random variables) of the outputs are the same.

Observe that if we have two oracle gates which are evaluated by $\psi(f, f, f, f^2)$ and whose inputs are different (an input (L, R) is different from (L', R') if either $L \neq L'$ or $R \neq R'$), their outputs yield four independent random variables.

A probabilistic distinguishing circuit generates 1 on its output with the same probability for both $\psi(f, f, f, f^2)$ and $\psi(f, f, f, g)$ if the random variables (the second part of the right hand side output of oracle gates) $f^2(\alpha_i)$ are independent from all internal random variables:

$$f(R_i);$$
$$f(L_i \oplus f(R_i));$$
$$\alpha_i = f(R_i \oplus f(L_i \oplus f(R_i)))$$

in all m oracle gates, where i is the number of the gate and (L_i, R_i) is the input to the i-th gate.

A probabilistic distinguishing circuit generates 1 on its output with different probabilities if there is at least one oracle gate in which $f^2(L_i \oplus f(R_i) \oplus \alpha_i)$ is expressable by a composition of two internal variables (there are $3m$ such random variables). Denote Pr_{diff} as the probability that there exists $f^2(L_i \oplus f(R_i) \oplus \alpha_i)$ that is expressable by two internal variables. Therefore:

$$\mid Pr[C_{2n}(\psi(F))] - Pr[C_{2n}(\psi(F, G))] \mid \leq Pr_{diff}$$

For fixed i $(1 \leq i \leq m)$, the probability that a single variable $f^2(L_i \oplus f(R_i) \oplus \alpha_i)$ can be constructed from two internal variables is:

$$\frac{9m^2}{4^n}$$

So:

$$Pr_{diff} = \frac{9m^3}{4^n}$$

and the final result follows. (A formal proof needs definitions of suitable probabilistic spaces to be introduced. Such spaces are the same as those in the paper by Luby and Rackoff [4]).

□

Denote that $p_F = Pr[C_{2n}(\psi(F))]$, $p_{F,G} = Pr[C_{2n}(\psi(F, G))]$ (those probabilities are defined above) and $p_R = Pr[C_{2n}(F_{2n})]$ is the probability that the oracle circuit generates "1" if the oracle gates are evaluated using a random function $f \in_R F_{2n}$.

Lemma 4.2 *If $\psi(f,f,f,g)$ is pseudorandom, then*

$$\mid p_F - p_R \mid \leq \frac{\gamma(m)}{2^n}$$

where $\gamma(m)$ is a polynomial in m (m is the number of oracle gates in the distinguishing circuit).

Proof. The previous lemma stated that

$$\mid p_F - p_{F,G} \mid \leq \frac{\gamma_1(m)}{2^n}$$

and the assumption about pseudorandomness of $\psi(f,f,f,g)$ is equivalent to

$$\mid p_{F,G} - p_R \mid \leq \frac{\gamma_2(m)}{2^n}.$$

where $\gamma_2(m)$ is a polynomial in m. The following is true

$$\begin{gathered}\mid p_F - p_R \mid = \mid p_F - p_{F,G} + p_{F,G} - p_R \mid \leq \\ \mid p_F - p_{F,G} \mid + \mid p_{F,G} - p_R \mid = \frac{\gamma(m)}{2^n}\end{gathered}$$

where $\gamma(m) = \gamma_1(m) + \gamma_2(m)$.

□

Theorem 4.1 *$\psi(f,f,f,f^2)$ is pseudorandom permutation generator.*

Proof (by contradiction). Proof is similar to that given in [4]. Denote that $p_f = Pr[C_{2n}(\psi(f,f,f,f^2))]$, where $f \in F_n$ is a pseudorandom function. Assume that there is a distinguishing circuit family for $\psi(f,f,f,f^2)$. Let it be $C = \{C_{2n_1}, C_{2n_2}, \cdots\}$, where $n_1 < n_2 < \cdots$. It means that for large enough n, the following sequence is true:

$$\frac{1}{n^c} \leq \mid p_f - p_R \mid = \mid p_f - p_F + p_F - p_R \mid \leq \mid p_f - p_F \mid + \mid p_F - p_R \mid$$

where c is a constant. As $\mid p_F - p_R \mid$ is bounded by $\frac{\gamma(m)}{2^n}$, $\mid p_f - p_F \mid \geq \frac{1}{2n^c}$. In other words the family C defines a distinguishing circuit family for pseudorandom function f. This is the contradiction and the theorem is proved.

□

All above considerations are valid for $\psi(f,f,f,f^i)$ for fixed $i = 2,3,\cdots$.

5 Towards provably secure block cryptosystems

The results obtained in the theory of pseudorandomness may be interpreted differently. As we know (see [4]), permutation generators $\psi(f,g,h)$ are pseudorandom if f,g,h are different pseudorandom functions. In general, if we use random functions instead pseudorandom ones then a permutation generator for which there is no distinguisher, is called a randomizer. Now we can rephrase the previous theorems.

Theorem 5.1 (Luby and Rackoff [4]) *Let $f, g, h \in_R F_n$ and a distinguisher C_{2n} have m oracle gates ($m < 2^n$), then:*

$$| Pr[C_{2n}(F)] - Pr[C_{2n}(\psi(f,g,h))] | \leq \frac{m^2}{2^n} \tag{1}$$

and the randomizer ψ extends binary strings of length $3 \times n2^n$ into strings of length $2n2^{2n}$

To make the probability in (1) sufficiently small n should be larger than 64. For $n = 64$ the randomizer $\psi(f,g,h)$ extends 3.5×10^{21}-bit random string into a random string of length 4.4×10^{40} bits.

Theorem 5.2 (Ohnishi [9]) *Let $f, g \in_R F_n$ and a distinguisher C_{2n} have m oracle gates ($m < 2^n$), then:*

$$| Pr[C_{2n}(F)] - Pr[C_{2n}(\psi(f,f,g))] | \leq \frac{2(m+1)^2}{2^n} \tag{2}$$

and the randomizer ψ extends binary strings of length $2 \times n2^n$ into strings of length $2n2^{2n}$

Theorem 5.3 *Let $f \in_R F_n$ and a distinguisher C_{2n} have m oracle gates ($m < 2^n$), then*

$$| Pr[C_{2n}(F)] - Pr[C_{2n}(\psi(f,f,f,f^i))] | \leq \frac{2(m+1)^2}{2^n} + \frac{3^i m^{i+1}}{2^{ni}} \tag{3}$$

and the randomizer ψ extends binary strings of length $n2^n$ into strings of length $2n2^{2n}$

The last theorem gives the best known result and for $n = 32$, the randomizer $\psi(f,f,f,f^i)$ extends 1.4×10^{11} bits into binary string of length 3.5×10^{21}. Unfortunately the probability given by (3) can be too high even for relatively small m (for instance if $m = 10^5$ the probability is close to 1).

Zheng et al. [10] studied the practicability of provable secure cryptosystems (PSC) and they pointed out that there is still gap between the theory (which says that PSCs exist) and the practice (any implementation needs exponential size memory to memorize random functions).

Consider formulae (1,2,3) and note that both memory requirements and probabilities depend on the value of 2^n. Ideally, we would like to be able to select in some way memory requirements (the size of random function) and probabilities independently, possibly by introducing an additional parameter. In general, it is well known that increase of number of rounds in any cryptosystem improves its quality. Therefore, such a parameter may be the number of rounds.

There are three different approaches:

- direct concatenation of basic randomizers ψ, i.e. we create an randomizer $\Psi_j = \underbrace{\psi \circ \psi \circ \cdots}_{j}$;
- construction of complex randomizers using a basic randomizer ψ, i.e. $\theta = \psi(p,p,p,p^2)$ where $p = \psi(f,f,f,f^2)$ and f is a random function;

- concatenation of complex randomizers, i.e. $\Theta_j = \underbrace{\theta \circ \theta \circ \cdots}_{j}$, where θ is given above.

The basic randomizer uses a random function $f \in F_n$. The resulting randomizer Ψ_j generates a permutation from F_{2n}. Both randomizers θ and Θ_j are permutations in F_{4n}. The most important question is: *What are the corresponding probabilities bounds for the first two approaches ?* Ideally, we would expect that the probabilities are expressable as:

$$\frac{Q(n)}{2^{nj}}$$

where j is the number of iterations. In this case, it would be possible to select small enough n so the memory requirements could be met and at the same time, the probability (3) could be adjusted by selecting the number of concatenations.

Of course, all above randomizers are based on the DES structure. It is possible to generalize this structure as is shown in [10]. Maurer and Massey [5] considered codes as randomizers.

6 Conclusions

The generation of binary sequences that resemble truly random ones, is widely used in many different applications. Some of these applications impose especially strict requirements. One such application is cryptography. Most binary sequence generators turned to be useless from a cryptographic point of view.

Yao [8] defined a class of generators which are not distinguishable from truly random generators having polynomial size sample of output and polynomially-bounded computing resources. He called them pseudorandom.

The existance of PBG has been proved by Levin [3] providing existence of a way-one function. There several implementations of PBG using different one-way functions (see [2]). Goldreich et al. [2] showed that PBGs can be used to construct pseudorandom function generators. Later, Luby and Rackoff [4] described pseudorandom permutation generators using three DES rounds and three pseudorandom functions.

Ohnishi improved their result by proving that pseudorandom permutation generators (PPGs) can be made up from three DES rounds and two pseudorandom functions. This result is optimal in the sense that using a single pseudorandom function and three DES rounds, it is impossible to construct PPG (Zheng et al [9]).

Schnor [7] was the first to pose the problem of construction of pseudorandom permutation generators using single pseudorandom functions. Zheng et al. [9] and Rueppel [6] showed that Schnors generator $\psi(f, f, f)$ is not pseudorandom. In this paper we have proved that having a single pseudorandom function and four DES rounds it is possible to construct PPGs.

If we substitute pseudorandom functions by truly random ones in PPG, we obtain generators which are called randomizers. Their quality is expressable by their probability bound for distinguishing them from truly random permutations and does not rely on unproved assumptions (as the existance of one-way functions).

Randomizers of structure $\psi(f, f, f, f^2)$ (f is a random function from F_n) stretch $n \times 2^n$ input bits into $2n \times 2^{2n}$ output bits. Zheng et al [10] defined provable secure cryptosystems as randomizers. There is however, a gap between the practical implementation and

the theory. To make this idea implementable, more research is necessary. Especially, it is interesting to examine the influence of the number of rounds on the distinguishing probability. Also a search for new "more" efficient randomizers could bring us closer to a practical implementation of PSC. Maurer and Massey [5] pointed out codes as one of possible randomizers.

ACKNOWLEDGMENT

I would like to thank Cathy Newberry, Dr Rei Safavi-Naini and Xian-Mo Zhang for their comments, discussions and assistance during the preparation of this work.

References

[1] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, November 1984.

[2] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[3] L. A. Levin. One-way function and pseudorandom generators. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 363–365, New York, 1985. ACM.

[4] M. Luby and Ch. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, April 1988.

[5] U.M. Maurer and J.L. Massey. Perfect local randomness in pseudorandom sequences. Astracts of CRYPTO'89, Santa Barbara, CA, August 1989.

[6] R.A. Rueppel. On the security of Schnorr's pseudo random generator. Astracts of EUROCRYPT'89, Houthalen, Belgium, April 1989.

[7] C.P. Schnorr. On the construction of random number generators and random function generators. In *Proc. of Eurocrypt 88, Lecture Notes in Computer Science*, New York, 1988. Springer Verlag.

[8] Andrew C. Yao. Theory and application of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Fundation of Computer Science*, pages 80–91, New York, 1982. IEEE.

[9] Y. Zheng, T. Matsumoto, and H. Imai. Impossibility and optimality results on constructing pseudorandom permutations. Astracts of EUROCRYPT'89, Houthalen, Belgium, April 1989.

[10] Y. Zheng, T. Matsumoto, and H. Imai. On the construction of block ciphers provably secure and not relying on any unproved hypotheses. Astracts of CRYPTO'89, Santa Barbara, CA, July 1989.