

On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses

(Extended Abstract)

Yuliang Zheng
Tsutomu Matsumoto
Hideki Imai

*Division of Electrical and Computer Engineering
Yokohama National University
156 Tokiwadai, Hodogaya, Yokohama, 240 Japan*

Abstract *One of the ultimate goals of cryptography researchers is to construct a (secrete-key) block cipher which has the following ideal properties: (1) The cipher is provably secure, (2) Security of the cipher does not depend on any unproved hypotheses, (3) The cipher can be easily implemented with current technology, and (4) All design criteria for the cipher are made public. It is currently unclear whether or not there really exists such an ideal block cipher. So to meet the requirements of practical applications, the best thing we can do is to construct a block cipher such that it approximates the ideal one as closely as possible. In this paper, we make a significant step in this direction. In particular, we construct several block ciphers each of which has the above mentioned properties (2), (3) and (4) as well as the following one: (1') Security of the cipher is supported by convincing evidence. Our construction builds upon profound mathematical bases for information security recently established in a series of excellent papers.*

1. Motivations and Summary of Results

Data Encryption Standard (DES) designed by IBM about *fifteen* years ago is the first modern (secrete-key) block cipher whose algorithm is publicly available [NBS]. It is a kind of product ciphers with Lucifer as its direct predecessor [FNS] [K]. A little more specifically, both DES and Lucifer consist of 16 rounds of *Feistel-type transformations (FTT's)* which are invented by and named (by us) after Feistel.

From the beginning of DES, however, there had a lot of controversy about its security, and especially, about its design criteria [K] which have been classified by NSA and its designer IBM. Many computer scientists and cryptography experts

were concerned about the possibilities that DES may possess weaknesses only NSA and IBM are aware of, and that trap-doors may have been inserted into the S-boxes of DES which would give a cryptanalytic advantage to a knowledgeable party. For these reasons, a great amount of effort has been invested in attempting to break the cipher, or to find its weaknesses. And many researchers have tried revealing the myths around the design criteria.

In their nice paper [LR], Luby and Rackoff showed that DES would be provably secure if its f -functions were secure pseudorandom ones. Unfortunately, the f -functions of DES cannot be secure in any reasonable sense. In the same paper, Luby and Rackoff proved also a result about FTT's: A function consisting of three rounds of randomly and independently chosen FTT's, which is in fact a permutation, cannot be efficiently distinguished from a truly random one. This result is very appealing, since it relies on no unproved hypotheses, and more importantly, it suggests that there is an extremely *simple* constructive method for designing a theoretically secure block cipher which does not rely on any unproved hypotheses. However, it is practically impossible to construct such a cipher, simply because it takes a huge amount of memory to implement the cipher.

Therefore both practical needs and theoretical interest encourage us to seek for an ideal block cipher having the following properties:

- (1) The cipher is provably secure,
- (2) Security of the cipher does not depend on any unproved hypotheses,
- (3) The cipher can be easily implemented with current technology, and
- (4) All design criteria for the cipher are made public.

It is still an open problem whether or not there really exists such a block cipher. The best thing we can do currently is to construct a block cipher such that it approximates the ideal one as closely as possible.

In this paper, we make a significant step in this direction. In particular, we propose a kind of transformations — *Generalized Type-2 transformations*, and show that it is an excellent building block for cryptosystems. Utilizing this type of transformations, we construct several concrete block ciphers which have the above mentioned properties (2), (3) and (4) as well as the following one:

- (1') Security of the cipher is supported by convincing evidence.

Our results build upon profound mathematical bases for information security recently established in a series of excellent papers such as [BM],[Y],[L],[GGM],[S]¹ and especially [LR].

The remaining part of the paper is organized as follows: Section 2 defines terminology used later, reviews one of the main design rules for DES — FTT's, and introduces the result of Luby and Rackoff on the rule. Section 3 proposes various types of transformations and shows that all these transformations can be used to construct permutations not efficiently distinguishable from a truly random

¹ The main result of [S] had been found to be false [O] [R] [ZMI]. But here the correct version of the result is used.

one. Among the transformations, Generalized Type-2 ones are proved to be most preferable. Section 4 constructs a theoretically provably secure block cipher (PSBC) by the use of Generalized Type-2 transformations. Section 5 presents a variant of PSBC. Section 6 proposes four concrete block ciphers based on theoretical results of Sections 2 – 5. Finally in Appendices A, B and C, three issues — minimum rounds for security, optimal transformations and super-security are discussed. For the sake of space, most results are presented without proofs. We will enclose proofs necessary in the full paper.

2. Preliminaries

This section defines the notions of pseudorandom number/function generators, and introduces the result of Luby and Rackoff on FTT's. Readers who are not interested in the definitions can jump over Section 2.1.

2.1 Pseudorandom Number/Function Generators

For purposes which will become clear later, our notions introduced below are slight generalizations of those given in [Y], [GGM] and [LR], mainly in the following aspect: In contrast to those in [Y], [GGM] and [LR], we will not impose polynomial bound upon the running time of an algorithm realizing a pseudorandom number/function generator or on the size of a (local) statistical test for strings/functions.

2.1.1 Pseudorandom Number Generators

The set of positive integers is denoted by \mathcal{N} . By a string we mean a binary string over the alphabet $\{0, 1\}$. For each $n \in \mathcal{N}$, denote by I_n the set of all 2^n strings of length n . For $s_1, s_2 \in I_n$, let $s_1 \oplus s_2$ denote the bit-wise XOR of the two strings. Denote by H_n the set of all $2^{n \cdot 2^n}$ functions and by Sym_n the set of all $2^n!$ permutations on I_n . The *composition* of two functions f and g in H_n , denoted by $f \circ g$, is defined by $f \circ g(x) = f(g(x))$ for all $x \in I_n$. By $x \in_R X$ we mean that x is drawn randomly and uniformly from a finite set X , and by a function in n (or t etc.) we mean, unless otherwise specified, a function from \mathcal{N} to \mathcal{N} .

Let P be a function in n with $P(n) > n$. A *pseudorandom number generator* (PNG) is a collection of functions $S = \{S_n \mid n \in \mathcal{N}\}$, where each function S_n maps an n -bit string *seed* into a $P(n)$ -bit string $S_n(\text{seed})$ and it can be computed by some deterministic algorithm. Security (or strength) of PNG's is defined in terms of local statistical tests for strings.

[Definition 1] Let Θ and \mathcal{L} be sets of functions in n , and Υ a set of functions from \mathcal{N} to $[0, 1]$. Let P be a function in n with $P(n) > n$, and let $\theta \in \Theta$ and $L \in \mathcal{L}$ with $0 < L(n) \leq P(n)$. A family of circuits $T^* = \{T_n^* \mid n \in \mathcal{N}\}$ is called a *local* (θ, L) *statistical test for strings* if each T_n^* is of size $\theta(n)$,² and on input an $L(n)$ -bit fixed portion of a $P(n)$ -bit string x , outputs a single bit $T_n^*[x]$. Call θ the size of

² The size of a circuit is the total number of connections in the circuit.

T^s . Now let $S = \{S_n \mid n \in \mathcal{N}\}$ be a PNG where S_n maps an n -bit string into a $P(n)$ -bit one. We say that

- (1) S *locally ε -passes the test T^s* if for all sufficiently large n , $|Pr\{T_n^s[r] = 1\} - Pr\{T_n^s[S_n(t)] = 1\}| < \varepsilon(n)$, where $r \in_R I_{P(n)}$, $t \in_R I_n$ and $\varepsilon \in \Upsilon$;
- (2) S is *locally (θ, L, ε) -secure* if it locally ε -passes all (θ, L) tests;
- (3) S is *locally $(\Theta, \mathcal{L}, \Upsilon)$ -secure* if it is locally (θ, L, ε) -secure for any $\varepsilon \in \Upsilon$ and any $(\theta, L) \in \Theta \times \mathcal{L}$ with $0 < L(n) \leq P(n)$.

Especially, a locally $(\Theta, \mathcal{L}, \Upsilon)$ -secure PNG S is said

- (4) *locally $(\infty, \mathcal{L}, \Upsilon)$ -secure* if Θ is the set of all functions in n , and
- (5) *strong* if, furthermore, \mathcal{L} is the infinite set of all polynomials in n and Υ that of all inverse polynomials in n . (An inverse polynomial in n is a function like $1/Q(n)$ where Q is a polynomial.)

Finally, assume that $S = \{S_n \mid n \in \mathcal{N}\}$ is a PNG where S_n can be computed in deterministic *polynomial* time in n . Then

- (6) S is called *locally polynomially secure* if it is locally $(\Theta, \mathcal{L}, \Upsilon)$ -secure where both Θ and \mathcal{L} are the infinite set of all polynomials in n , and Υ that of all inverse polynomials in n .

Note that Yao's definition for *polynomial size statistical tests for strings* [Y] [GGM] is obtained from ours by letting P , θ and L be polynomials in n with $P = L$. Now assume, as at the end of Definition 1, that $S = \{S_n \mid n \in \mathcal{N}\}$ is a PNG where S_n can be computed in deterministic *polynomial* time in n . For such a PNG S , Yao defined that it *passes a polynomial size statistical test for strings* $T^s = \{T_n^s \mid n \in \mathcal{N}\}$ if for any polynomial P_1 and for all sufficiently large n , $|Pr\{T_n^s[r] = 1\} - Pr\{T_n^s[S_n(t)] = 1\}| < 1/P_1(n)$, where $r \in_R I_{P(n)}$ and $t \in_R I_n$, and that S is *polynomially secure* if it passes all polynomial size statistical tests for strings.

The following fact is an immediate consequence of Yao's famous theorem on statistical tests [Y] [GGM]: Assume that $S = \{S_n \mid n \in \mathcal{N}\}$ is a PNG where S_n can be computed in deterministic polynomial time in n . Then S is polynomially secure iff it is locally polynomially secure.

2.1.2 Pseudorandom Function Generators

Let P be an increasing function in n . A *pseudorandom function generator* (PFG) is a collection of functions $F = \{F_n \mid n \in \mathcal{N}\}$, where F_n specifies for each $P(n)$ -bit string *key*, (the description of) a function $F_n(\text{key}) \in H_n$ that can be computed by some deterministic algorithm.

Security of a PFG is defined in terms of statistical tests for functions, and the latter uses the concept of oracle circuits which are counterparts of often used *oracle Turing machines*. An *oracle circuit* C_n is an acyclic circuit which contains, in addition to ordinary AND, OR, NOT and constant gates, also a particular kind of gates — *oracle gates*. Each oracle gate has an n -bit input and an n -bit output, and it is evaluated using some function from H_n . The output of C_n , a single bit, is denoted by $C_n[f]$ when a function $f \in H_n$ is used to evaluate the oracle gates.

[Definition 2] Let Θ and \mathcal{Q} be sets of functions in n , and Υ a set of functions from \mathcal{N} to $[0, 1]$. Let $\theta \in \Theta$ and $Q \in \mathcal{Q}$ be two functions with $0 \leq Q(n) < \theta(n)$. A family of circuits $T^f = \{T_n^f \mid n \in \mathcal{N}\}$ is called a (θ, Q) *statistical test for functions* where T_n^f is an oracle circuit which is of size $\theta(n)$ and has $Q(n)$ oracle gates. Let P be an increasing function in n , and $F = \{F_n \mid n \in \mathcal{N}\}$ a PFG where F_n specifies for each $P(n)$ -bit string *key* a function $F_n(\text{key}) \in H_n$. We say that

- (1) F ε -passes the test T^f if for all sufficiently large n , $|\Pr\{T_n^f[r] = 1\} - \Pr\{T_n^f[F_n(g)] = 1\}| < \varepsilon(n)$ where $r \in_R H_n$, $g \in_R I_{P(n)}$ and $\varepsilon \in \Upsilon$.
- (2) F is (θ, Q, ε) -secure if it ε -passes all (θ, Q) tests.
- (3) F is $(\Theta, \mathcal{Q}, \Upsilon)$ -secure if it is (θ, Q, ε) -secure for any $\varepsilon \in \Upsilon$ and any $(\theta, Q) \in \Theta \times \mathcal{Q}$ with $0 \leq Q(n) < \theta(n)$.

Especially,

- (4) a $(\Theta, \mathcal{Q}, \Upsilon)$ -secure PFG F is said $(\infty, \mathcal{Q}, \Upsilon)$ -secure when Θ is the set of all functions in n .

Finally assume that for each n and for each *key* $\in I_{P(n)}$, the function $F_n(\text{key})$ can be computed in deterministic polynomial time in n . (This implies that P is a polynomial in n .) Then

- (5) F is called *polynomially secure* when it is $(\Theta, \mathcal{Q}, \Upsilon)$ -secure for Θ and \mathcal{Q} being the infinite set of all polynomials in n and Υ being the infinite set of all inverse polynomials in n .

We are mainly interested in a special kind of PFG's — pseudorandom permutation generators which are invertible. Let P be an increasing function in n . A *pseudorandom permutation generator* is a pseudorandom function generator $F = \{F_n \mid n \in \mathcal{N}\}$, where F_n specifies for each $P(n)$ -bit string *key* a *permutation* $F_n(\text{key}) \in \text{Sym}_n$ that can be computed by some deterministic algorithm. A pseudorandom permutation generator $F = \{F_n \mid n \in \mathcal{N}\}$ is called *invertible* if there is a pseudorandom permutation generator $\tilde{F} = \{\tilde{F}_n \mid n \in \mathcal{N}\}$ such that for each $P(n)$ -bit string *key*, \tilde{F}_n specifies the inverse of $F_n(\text{key})$. Security of (invertible) pseudorandom permutation generators is defined in exactly the same way as for pseudorandom function generators.

2.2 Feistel-Type Transformation (FTT)

For a function $f_i \in H_n$, we associate with it a function $g_i \in H_{2n}$ defined by

$$g_i(B_1, B_2) = (B_2 \oplus f_i(B_1), B_1)$$

where $B_1, B_2 \in I_n$. Note that g_i is obtained from f_i by applying one of the main design rules for DES, and it corresponds roughly to a layer of DES (Figure 1). Since the design rule was due to Feistel, we call g_i a *Feistel-type transformation* (FTT).

For $f_1, f_2, \dots, f_s \in H_n$, let $\psi(f_s, \dots, f_2, f_1) = g_s \circ \dots \circ g_2 \circ g_1$. We say that $\psi(f_s, \dots, f_2, f_1)$ consists of s rounds of FTT's. Obviously, g_i is an invertible permutation, and hence so is $\psi(f_s, \dots, f_2, f_1)$.

Luby and Rackoff proved the following result which was called *Main Lemma* in [LR] but is called *FTT Lemma* in this paper: For independent random functions

$f_1, f_2, f_3 \in H_n$, it is infeasible to distinguish $\psi(f_3, f_2, f_1)$ from a function drawn randomly and uniformly from H_{2n} . (See Figure 2.)

[FTT Lemma] (Version 1, [LR]) *Let Q be a polynomial in n and C_{2n} be an oracle circuit with $Q(n) < 2^n$ oracle gates. Then $|\Pr\{C_{2n}[r] = 1\} - \Pr\{C_{2n}[\psi(f_3, f_2, f_1)] = 1\}| \leq \frac{Q(n)^2}{2^n}$, where $r \in_R H_{2n}$ and $f_1, f_2, f_3 \in_R H_n$.*

FTT Lemma is surprising in the sense that it does not depend on any unproved hypotheses. It implies that we can construct as follows a block cipher which does not rely on any assumption and is provably secure against chosen-plaintext attack: Let the length of a plaintext be $2n$. Choose randomly and uniformly from H_n three functions f_1, f_2 and f_3 , and let the enciphering algorithm be $\psi(f_3, f_2, f_1)$ and the deciphering algorithm be the inverse of $\psi(f_3, f_2, f_1)$.

However one soon finds that such an approach is impractical: To make the cipher secure against some trivial attacks such as exhaustive search, $2n$ must be sufficiently large, say ≥ 64 , i.e., $n \geq 32$. When $n = 32$, specifying $\psi(f_3, f_2, f_1)$ takes at least $3 \cdot 32 \cdot 2^{32} \approx 4 \cdot 10^{11}$ bits, which is infeasible currently and even in the foreseeable future. In other words, there is still a big gap between practically constructing a provably secure block cipher and the nice theory initiated by Luby and Rackoff. In the following sections we will examine various types of transformations, and fill the gap greatly.

3. Cryptographically Useful Transformations

This section introduces various types of transformations, and generalizes FTT Lemma in many directions. First we introduce two operations on strings in I_{kn} — the ρ -position left rotation and the ρ -position right rotation. These two operations are denoted by $L_{rot}^{(\rho)}$ and $R_{rot}^{(\rho)}$, and defined as

$$\begin{aligned} L_{rot}^{(\rho)}(B_1, B_2, \dots, B_k) &= (B_{\rho+1}, \dots, B_k, B_1, B_2, \dots, B_\rho), \\ R_{rot}^{(\rho)}(B_1, B_2, \dots, B_k) &= (B_{k-\rho+1}, \dots, B_k, B_1, B_2, \dots, B_{k-\rho}) \end{aligned}$$

respectively, where $1 \leq \rho < k$ and $B_j \in I_n$. Note that both $L_{rot}^{(\rho)}$ and $R_{rot}^{(\rho)}$ are permutations on I_{kn} , and that $L_{rot}^{(\rho)}$ is the inverse of $R_{rot}^{(\rho)}$ and vice versa.

3.1 Various Transformations

3.1.1 Type-1 Transformations

Following [FNS, pp.1547-1549] and [S], we associate with an $f_i \in H_n$ a function $g_{1,i} \in H_{kn}$ defined by

$$g_{1,i}(B_1, B_2, \dots, B_k) = (B_2 \oplus f_i(B_1), B_3, \dots, B_k, B_1),$$

where $B_j \in I_n$. Functions obtained in such a way are called *Type-1 transformations*.

Note that $g_{1,i}$ can be decomposed into $g_{1,i} = L_{rot}^{(1)} \circ \pi_{1,i}$ where $\pi_{1,i}$ is defined by $\pi_{1,i}(B_1, B_2, \dots, B_k) = (B_1, B_2 \oplus f_i(B_1), B_3, \dots, B_k)$. (See Figure 3.) It is easy to check that $\pi_{1,i} \circ \pi_{1,i}$ is the *identity transformation* on I_{kn} , i.e., $\pi_{1,i}$ is the inverse of itself. Such a function is usually called an *involution* [K]. Now we see that $g_{1,i}$ is an invertible permutation on I_{kn} , and its inverse, denoted by $\tilde{g}_{1,i}$, is given by $\tilde{g}_{1,i} = \pi_{1,i} \circ R_{rot}^{(1)}$.

For $f_1, f_2, \dots, f_s \in H_n$, define $\psi_1(f_s, \dots, f_2, f_1) = g_{1,s} \circ \dots \circ g_{1,2} \circ g_{1,1}$. $\psi_1(f_s, \dots, f_2, f_1)$ is also an invertible permutation on I_{kn} , and by definition, its inverse is $\tilde{\psi}_1(f_s, \dots, f_2, f_1) = \pi_{1,1} \circ R_{rot}^{(1)} \circ \dots \circ \pi_{1,s-1} \circ R_{rot}^{(1)} \circ \pi_{1,s} \circ R_{rot}^{(1)}$.

3.1.2 Type-2 Transformations

Let $k = 2\ell$, where $\ell \in \mathcal{N}$. Associate with a function-tuple $h_i = (f_{i,1}, f_{i,3}, \dots, f_{i,2\ell-1})$, where $f_{i,j} \in H_n$, a function $g_{2,i} \in H_{kn}$ defined by

$$g_{2,i}(B_1, B_2, \dots, B_k) = (B_2 \oplus f_{i,1}(B_1), B_3, B_4 \oplus f_{i,3}(B_3), \dots, B_{k-1}, B_k \oplus f_{i,k-1}(B_{k-1}), B_1).$$

$g_{2,i}$ is called a *Type-2 transformation*, and can be decomposed into $g_{2,i} = L_{rot}^{(1)} \circ \pi_{2,i}$ where $\pi_{2,i}$ is defined by $\pi_{2,i}(B_1, B_2, \dots, B_k) = (B_1, B_2 \oplus f_{i,1}(B_1), B_3, \dots, B_{k-1}, B_k \oplus f_{i,k-1}(B_{k-1}))$. (See Figure 4.) Obviously, $\pi_{2,i}$ is an involution. For s function-tuples h_1, h_2, \dots, h_s , define $\psi_2(h_s, \dots, h_2, h_1) = g_{2,s} \circ \dots \circ g_{2,2} \circ g_{2,1}$. The inverse of $\psi_2(h_s, \dots, h_2, h_1)$ is $\tilde{\psi}_2(h_s, \dots, h_2, h_1) = \tilde{g}_{2,1} \circ \dots \circ \tilde{g}_{2,s-1} \circ \tilde{g}_{2,s}$, where $\tilde{g}_{2,i} = \pi_{2,i} \circ R_{rot}^{(\rho)}$.

3.1.3 Type-3 Transformations

Associate with a function-tuple $h_i = (f_{i,1}, f_{i,2}, \dots, f_{i,k-1})$, where $f_{i,j} \in H_n$, a function $g_{3,i} \in H_{kn}$ defined by

$$g_{3,i}(B_1, B_2, \dots, B_k) = (B_2 \oplus f_{i,1}(B_1), B_3 \oplus f_{i,2}(B_2), \dots, B_k \oplus f_{i,k-1}(B_{k-1}), B_1).$$

Call $g_{3,i}$ a *Type-3 transformation*. We decompose $g_{3,i}$ into $g_{3,i} = L_{rot}^{(1)} \circ \pi_{3,i}$ where $\pi_{3,i}$ is defined by $\pi_{3,i}(B_1, B_2, \dots, B_k) = (B_1, B_2 \oplus f_{i,1}(B_1), B_3 \oplus f_{i,2}(B_2), \dots, B_k \oplus f_{i,k-1}(B_{k-1}))$. See Figure 5. $\pi_{3,i}$ is a permutation and its inverse is given by $\tilde{\pi}_{3,i}(C_1, C_2, \dots, C_k) = (B_1, B_2, \dots, B_k)$, where $B_1 = C_1$ and $B_j = C_j \oplus f_{i,j-1}(B_{j-1})$ for each $2 \leq j \leq k$. One can soon find that $\pi_{3,i}$ is *not* an involution (Figure 6).

For s function-tuples h_1, h_2, \dots, h_s , define $\psi_3(h_s, \dots, h_2, h_1) = g_{3,s} \circ \dots \circ g_{3,2} \circ g_{3,1}$. Since both $\pi_{3,i}$ and $L_{rot}^{(1)}$ are permutations, hence so are $g_{3,i}$ and $\psi_3(h_s, \dots, h_2, h_1)$. The inverse of $\psi_3(h_s, \dots, h_2, h_1)$ is $\tilde{\psi}_3(h_s, \dots, h_2, h_1) = \tilde{\pi}_{3,1} \circ R_{rot}^{(1)} \circ \dots \circ \tilde{\pi}_{3,s-1} \circ R_{rot}^{(1)} \circ \tilde{\pi}_{3,s} \circ R_{rot}^{(1)}$.

3.1.4 Generalized Transformations

From its definition, we see that $\pi_{1,i}$ can be obtained from $\pi_{3,i}$ by dropping functions $f_{i,j}$ in $h_i = (f_{i,1}, f_{i,2}, \dots, f_{i,k-1})$ for all $2 \leq j \leq k-1$. Similarly,

when k is even, $\pi_{2,i}$ can also be obtained from $\pi_{3,i}$ by dropping functions $f_{i,j}$ in $h_i = (f_{i,1}, f_{i,2}, \dots, f_{i,k-1})$ for all even $1 < j < k-1$.

Denote by $\pi_{\tau,i}$ a permutation obtained from $\pi_{3,i}$, by dropping certain functions $f_{i,j}$ in $h_i = (f_{i,1}, f_{i,2}, \dots, f_{i,k-1})$. (Note: $\pi_{\tau,i} = \pi_{3,i}$ when dropping no function.) Define $g_{\tau,i}^{(\rho)} = L_{\tau,i}^{(\rho)} \circ \pi_{\tau,i}$, where $1 \leq \rho \leq k-1$. Call transformations so obtained *Generalized Type- τ transformations*. Likewise, for s functions/function-tuples h_1, h_2, \dots, h_s , define $\psi_{\tau}^{(\rho)}(h_s, \dots, h_2, h_1) = g_{\tau,s}^{(\rho)} \circ \dots \circ g_{\tau,2}^{(\rho)} \circ g_{\tau,1}^{(\rho)}$.

3.2 Theorems on the Transformations

Let E be a permutation consisting of $2k-1$ rounds of Type-1, or $k+1$ rounds of Type-2, or $k+1$ rounds of Type-3 transformations, each of which is chosen randomly and independently. The following Theorems 1-3 say that *no* oracle circuit with polynomially many oracle gates can distinguish between E and a truly random function.

[Theorem 1] *Let Q be a polynomial in n and C_{kn} be an oracle circuit with $Q(n) < 2^n$ oracle gates. Then $|\Pr\{C_{kn}[r] = 1\} - \Pr\{C_{kn}[\psi_1(f_{2k-1}, \dots, f_2, f_1)] = 1\}| \leq \frac{(k-1)Q(n)^2}{2^n}$, where $r \in {}_{\mathbb{R}}H_{kn}$ and $f_1, f_2, \dots, f_{2k-1} \in {}_{\mathbb{R}}H_n$.*

[Theorem 2] *Let Q be a polynomial in n and C_{kn} be an oracle circuit with $Q(n) < 2^n$ oracle gates where $k = 2\ell$. Then $|\Pr\{C_{kn}[r] = 1\} - \Pr\{C_{kn}[\psi_2(h_{k+1}, \dots, h_2, h_1)] = 1\}| \leq \frac{\ell^2 Q(n)^2}{2^n}$, where $r \in {}_{\mathbb{R}}H_{kn}$ and $h_i = (f_{i,1}, f_{i,3}, \dots, f_{i,k-1})$ with $f_{i,j} \in {}_{\mathbb{R}}H_n$.*

[Theorem 3] *Let Q be a polynomial in n and C_{kn} be an oracle circuit with $Q(n) < 2^n$ oracle gates. Then $|\Pr\{C_{kn}[r] = 1\} - \Pr\{C_{kn}[\psi_3(h_{k+1}, \dots, h_2, h_1)] = 1\}| \leq \frac{k(k-1)Q(n)^2}{2^{n+1}}$, where $r \in {}_{\mathbb{R}}H_{kn}$ and $h_i = (f_{i,1}, f_{i,2}, \dots, f_{i,k-1})$ with $f_{i,j} \in {}_{\mathbb{R}}H_n$.*

Theorem 2 can be proved by essentially the same technique developed in [LR] for proving FTT Lemma. Details will appear in the full paper. Proofs for Theorems 1 and 3 can be derived from the proof for Theorem 2.

For Generalized Type-2 transformations, we have the following theorem, which is crucial to our construction of block ciphers described in Sections 4-6, and can be proved by modifying the proof for Theorem 2. For the other types of generalized transformations we have no results similar to Theorem 2-G. For reasons see Appendix A where many other results are presented.

[Theorem 2-G] (Version 1) *Let $k = 2\ell$, where $\ell \in \mathcal{N}$, and let ρ be an odd integer in $[1, k]$. Let Q be a polynomial in n and C_{kn} be an oracle circuit with $Q(n) < 2^n$ oracle gates. Then $|\Pr\{C_{kn}[r] = 1\} - \Pr\{C_{kn}[\psi_2^{(\rho)}(h_{k+1}, \dots, h_2, h_1)] = 1\}| \leq \frac{\ell^2 Q(n)^2}{2^n}$, where $r \in {}_{\mathbb{R}}H_{kn}$ and $h_i = (f_{i,1}, f_{i,3}, \dots, f_{i,k-1})$ with $f_{i,j} \in {}_{\mathbb{R}}H_n$.*

3.3 Optimal Transformations

Let E be a permutation consisting of s rounds of randomly chosen Generalized Type- τ transformations. From Theorem A5 in Appendix A we see that $s \geq$

$k + 1$ is a necessary condition for E being indistinguishable from a truly random function by all oracle circuits with polynomially many oracle gates. Call a type of transformations *optimal* if

- (1) a permutation E consisting of $k + 1$ rounds of randomly chosen transformations is indistinguishable from a truly random function by all oracle circuits with polynomially many oracle gates, and
- (2) the inverse of E can be computed in the same parallel time as E .

For a rigorous definition of optimality, see Appendix B. The following theorem is proved in the same appendix.

[Theorem B1] *Among all types of transformations discussed in this paper, Generalized Type-2 transformations $g_{2,i}^{(\rho)} = L_{rot}^{(\rho)} \circ \pi_{2,i}^{(\rho)}$ with even k and odd ρ , are the only optimal ones.*

4. PSBC — A Provably Secure Block Cipher

Applying the only optimal Generalized Type-2 Transformations we construct a provably secure block cipher (PSBC) in this section.

4.1 A Few Observations

As pointed out in [S], FTT Lemma remains true even if the number of oracle gates is replaced by $Q(n) \leq 2^{o(n)}$ [S]. Here $Q(n) \leq 2^{o(n)}$ means that $Q(n) \leq 2^{f(n)}$ for some $f(n)$, which satisfies $\lim_{n \rightarrow \infty} \frac{cf(n)}{n} = 0$ for every positive constant c , i.e., $f(n) = o(n)$.

[FTT Lemma] (Version 2, [S]) *Let C_{2n} be an oracle circuit with $Q(n) \leq 2^{o(n)}$ oracle gates. Then $|Pr\{C_{2n}[r] = 1\} - Pr\{C_{2n}[\psi(f_3, f_2, f_1)] = 1\}| \leq \frac{Q(n)^2}{2^n}$, where $r \in_R H_{2n}$ and $f_1, f_2, f_3 \in_R H_n$.*

Schnorr's observation also applies to our Theorem 2-G (Version 1) stated in Section 3.

[Theorem 2-G] (Version 2) *Let $k = 2\ell$, where $\ell \in \mathcal{N}$, and let ρ be an odd integer in $[1, k]$. Let C_{kn} be an oracle circuit with $Q(n) \leq 2^{o(n)}$ oracle gates. Then $|Pr\{C_{kn}[r] = 1\} - Pr\{C_{kn}[\psi_2^{(\rho)}(h_{k+1}, \dots, h_2, h_1)] = 1\}| \leq \frac{t^2 Q(n)^2}{2^n}$, where $r \in_R H_{kn}$ and $h_i = (f_{i,1}, f_{i,3}, \dots, f_{i,k-1})$ with $f_{i,j} \in_R H_n$.*

Next we make a few more observations. Let $t \in \mathcal{N}$ and $n = \lceil (\log t)^{1+\epsilon} \rceil$ for some $\epsilon \in (0, 1]$, where the logarithm is taken to the base 2. Then for any constants c and ϵ' with $c > 0$ and $0 \leq \epsilon' < \epsilon$, we have $c \log t \leq c(\log t)^{1+\epsilon'} = o(\lceil (\log t)^{1+\epsilon} \rceil) = o(n)$, and $t^c = 2^{c \log t} \leq 2^{c(\log t)^{1+\epsilon'}} = 2^{o(n)}$. Thus we obtain from Theorem 2-G (Version 2) the following one.

[Theorem 2-G] (Version 3) *Let $k = 2\ell$, where $\ell \in \mathcal{N}$, and let ρ be an odd integer in $[1, k]$. Assume that $t \in \mathcal{N}$, $\epsilon \in (0, 1]$, $n = \lceil (\log t)^{1+\epsilon} \rceil$ and*

$Q(t) \leq 2^{o(n)}$ is a polynomial in t . (Notice that $2^n = 2^{\lceil (\log t)^{1+\epsilon} \rceil}$ is quasi-polynomial in t .)³ Let C_{2n} be an oracle circuit with $Q(t)$ oracle gates. Then $|\Pr\{C_{kn}[r] = 1\} - \Pr\{C_{kn}[\psi_2^{(\rho)}(h_{k+1}, \dots, h_2, h_1)] = 1\}| \leq \frac{t^2 Q(t)^2}{2^n}$, where $r \in_R H_{kn}$ and $h_i = (f_{i,1}, f_{i,3}, \dots, f_{i,k-1})$ with $f_{i,j} \in_R H_n$.

4.2 Enciphering/Deciphering Algorithms for PSBC

Theorem 2-G (Version 3) says that theoretically, if one is quasi-polynomially powerful, then one can construct a block cipher secure against any polynomially powerful adversary.

Let $n = \lceil (\log t)^{1+\epsilon} \rceil$, where $t \in \mathcal{N}$ and $\epsilon \in (0, 1]$. Let $\ell \in \mathcal{N}$, $k = 2\ell$ and ρ be an odd integer in $[1, k]$. Assume that the plaintext and ciphertext spaces are I_{kn} . Denote by $B = (B_1, B_2, \dots, B_k)$ a plaintext in I_{kn} and by $C = (C_1, C_2, \dots, C_k)$ the ciphertext of B , where $B_i, C_i \in I_n$.

PSBC consists principally of s rounds of Generalized Type-2 transformations where $s \geq k+2$. The reason for choosing $s \geq k+2$ is as follows: When $s = k+1$, our block cipher PSBC is secure against chosen plaintext attack, but not secure against chosen plaintext/ciphertext attack. When $s \geq k+2$, PSBC is secure against chosen plaintext/ciphertext attack. See Appendix C and [LR].

The enciphering algorithm E for PSBC can be concisely expressed as

$$E = \pi_{2,s} \circ L_{rot}^{(\rho)} \circ \dots \circ L_{rot}^{(\rho)} \circ \pi_{2,2} \circ L_{rot}^{(\rho)} \circ \pi_{2,1}.$$

See Figure 7. The inverse of E is $\pi_{2,1} \circ R_{rot}^{(\rho)} \circ \dots \circ R_{rot}^{(\rho)} \circ \pi_{2,s-1} \circ R_{rot}^{(\rho)} \circ \pi_{2,s}$. So the deciphering algorithm D for PSBC is obtained from E by

- (1) interchanging h_i with h_{s+1-i} for each $1 \leq i \leq \lfloor \frac{s}{2} \rfloor$, and
- (2) changing the mapping (or wiring) representing $L_{rot}^{(\rho)}$ to the mapping (or wiring) representing $R_{rot}^{(\rho)}$.

Notice that when ℓ is odd and $\rho = \ell$, there is no need for changing the mapping (or wiring), since $L_{rot}^{(\rho)} = R_{rot}^{(\rho)}$ in this case.

5. A Variant of PSBC

The block cipher PSBC described in Section 4 requires quasi-polynomially many memory cells for both enciphering and deciphering procedures. Thus it is practically impossible to realize the cipher. This section presents a variant of PSBC, in order to pave the way to practically realizable ciphers. The variant is obtained by adding to PSBC a *key-expanding part*. The key-expanding part stretches a short string into a long one, i.e., is a PNG. The PNG we use is a *strong* one (see Definition 1), and it is essentially due to Ohnishi and Schnorr [O] [S].

5.1 A Strong Pseudorandom Number Generator

Ohnishi observed that FTT Lemma remains valid even when *two* independent random functions are available [O].

³ We call a function f *quasi-polynomial* in t if for any polynomial P , for any constant $c > 0$ and for all sufficiently large t , we have $P(t) < f(t) < 2^{t^c}$.

[FTT Lemma] (Version 3, [O]) Let Q be a polynomial in n , and let C_{2n} be an oracle circuit with $Q(n) \leq 2^{o(n)}$ oracle gates. Then $|\Pr\{C_{2n}[r] = 1\} - \Pr\{C_{2n}[\psi(f_2, f_1, f_1)] = 1\}| \leq \frac{2(Q(n)+1)^2}{2^n}$, and $|\Pr\{C_{2n}[r] = 1\} - \Pr\{C_{2n}[\psi(f_2, f_2, f_1)] = 1\}| \leq \frac{2(Q(n)+1)^2}{2^n}$, where $r \in {}_R H_{2n}$, $f_1, f_2 \in {}_R H_n$.

Schnorr [S] showed that FTT Lemma implies that we can explicitly construct a PNG without any hypotheses. Putting together observations made in [O] and [S], we have the following PNG.

First we note that there is a natural one-one correspondence between functions in H_n and strings in I_{n2^n} , i.e., a bijection Φ_n from H_n to I_{n2^n} . The bijection maps a function $f \in H_n$ into the concatenation of $\prod_{x \in I_n} f(x)$, where x ranges over all strings $x \in I_n$ in a predetermined (such as lexicographical) order, and \prod is the concatenation operation on more than two strings. By this bijection, $\psi(f_2, f_2, f_1)$ constructed from $f_1, f_2 \in H_n$ via FTT's yields a function $S_{2n2^n} : I_{2n2^n} \rightarrow I_{2n2^{2n}}$. S_{2n2^n} maps a string $x = x_1 x_2$ where $x_1, x_2 \in I_{n2^n}$, into a string $y \in I_{2n2^{2n}}$ in the following way: $S_{2n2^n}(x) = \Phi_{2n}(\psi(\Phi_n^{-1}(x_2), \Phi_n^{-1}(x_2), \Phi_n^{-1}(x_1)))$.

Now we describe concretely an algorithm G_n computing the function S_{2n2^n} . The algorithm follows a similar one in [S]. (See Figure 8.) We write a string $x \in I_{2n2^n}$ as the concatenation of two strings $x_1, x_2 \in I_{n2^n}$, each of which is written as the concatenation of 2^n strings in I_n , i.e., $x = x_1 x_2 = \prod_{i \in I_n} x_{1,i} \prod_{i \in I_n} x_{2,i}$, where $x_{1,i}, x_{2,i} \in I_n$. Likewise, we write a string $y \in I_{2n2^{2n}}$ as the concatenation of 2^{2n} strings in I_{2n} , i.e., $y = \prod_{i \in I_{2n}} y_i$, where $y_i \in I_{2n}$. For a string $y \in I_{2n}$, let $B_1(y)$ and $B_2(y)$ be the left and right half strings in I_n .

Algorithm $G_n(x)$

/* This algorithm outputs a $2n2^{2n}$ -bit string y

on input a $2n2^n$ -bit string $x = x_1 x_2 = \prod_{i \in I_n} x_{1,i} \prod_{i \in I_n} x_{2,i}$. */

- (1) For all $i \in I_{2n}$, let $y_i^0 := i$;
- (2) For all $i \in I_{2n}$ do
 - { $w := y_i^0$; $u := x_{1, B_1(w)}$;
 - $y_i^1 := (B_2(w) \oplus u, B_1(w))$ };
- (3) For $j = 1, 2$ do
 - For all $i \in I_{2n}$ do
 - { $w := y_i^j$; $u := x_{2, B_1(w)}$;
 - $y_i^{j+1} := (B_2(w) \oplus u, B_1(w))$ };
- (4) Output $y = \prod_{i \in I_{2n}} y_i^3$.

Let $S = \{S_{e(n)} | e(n) = 2n2^n, n \in \mathcal{N}\}$. From [S] we know that the PNG S passes all statistical tests for strings which receive at most $2^{o(n)}$ bits as input. In our terms, this can be formally stated as follows.

[Theorem 4] (Version 1) *The PNG $S = \{S_{e(n)} | e(n) = 2n2^n, n \in \mathcal{N}\}$ is locally $(\infty, \mathcal{L}, \Upsilon)$ -secure where \mathcal{L} is the infinite set of functions L in n with $L(n) \leq 2^{o(n)}$ and Υ that of all inverse polynomials in n .*

Proof: A local statistical test for strings $T' = \{T'_n \mid n \in \mathcal{N}\}$, where T'_n has a $Q(n)(\leq 2^{o(n)})$ -bit input, can be viewed as a statistical test for functions $T' = \{T'_n \mid n \in \mathcal{N}\}$, where T'_n has at most $Q(n) \leq 2^{o(n)}$ oracle gates that are evaluated using a function from H_{2n} . Thus the theorem is true by FTT Lemma (Version 2) in Section 4.1. ■

Applying our observation made in Section 4, this theorem can be translated into the following theorem.

[Theorem 4] (Version 2) *Let $n = \lceil (\log t)^{1+\varepsilon} \rceil$, where $t \in \mathcal{N}$ and $\varepsilon \in (0, 1]$. Then the PNG $S = \{S_{\bar{e}(t)} | \bar{e}(t) = e(n) = 2n2^n, n = \lceil (\log t)^{1+\varepsilon} \rceil, t \in \mathcal{N}\}$ where $S_{\bar{e}(t)}$ maps an $\bar{e}(t) = e(n) = 2n2^n$ -bit string into a $2n2^{2n}$ -bit one, is locally $(\infty, \mathcal{L}, \Upsilon)$ -secure where \mathcal{L} is the infinite set of all polynomials in t , and Υ that of all inverse polynomials in t . That is to say, S is a strong PNG.*

5.2 PSBC with Key-Expanding

Let $n = \lceil (\log t)^{1+\varepsilon} \rceil$, where $t \in \mathcal{N}$ and $\varepsilon \in (0, 1]$. Let I_{kn} be the plaintext/ciphertext spaces where $k = 2\ell$, $\ell \in \mathcal{N}$, and let ρ be an odd integer in $[1, k]$, s an integer with $s \geq k + 2$.

The enciphering algorithm consists of two parts: *the enciphering part* and *the key-expanding part* (Figure 9). The enciphering part, as PSBC, consists essentially of s rounds of Generalized Type-2 transformations. The key-expanding part is an algorithm G_m that computes a function $S_{\bar{e}(t)}$ from a strong PNG $S = \{S_{\bar{e}(t)} | \bar{e}(t) = 2m2^m, m = n \lceil \log n \rceil, n = \lceil (\log t)^{1+\varepsilon} \rceil, t \in \mathcal{N}\}$, and it can expand a $2m2^m$ -bit input string into a $2m2^{2m}$ -bit output string.

The deciphering algorithm is obtained by

- (1) reversing the portion, which is used by the enciphering part, of the output of the key-expanding part and
- (2) changing the mapping (or wiring) representing $L_{rot}^{(\rho)}$ to the mapping (or wiring) representing $R_{rot}^{(\rho)}$.

The following theorem implies that the block cipher PSBC with key-expanding is secure against any polynomial size adversary. It can be proved by making some obvious modifications on the proof for Theorem 1 of [LR].

[Theorem 5] *Let $k = 2\ell$ where $\ell \in \mathcal{N}$, and ρ be an odd integer in $[1, k]$. Also let $t \in \mathcal{N}$, $\varepsilon \in (0, 1]$, $n = \lceil (\log t)^{1+\varepsilon} \rceil$ and $S = \{S_{m(t)} | \bar{e}(t) = 2m2^m, m = n \lceil \log n \rceil, n = \lceil (\log t)^{1+\varepsilon} \rceil, t \in \mathcal{N}\}$ be the above constructed strong PNG. Assume that P and Q are polynomials in t and that C_{2n} is an oracle circuit with $Q(t)$ oracle gates. Then for any $r \in {}_R H_{kn}$, for any $x \in {}_R I_{\bar{e}(t)}$, and for any h_1, h_2, \dots, h_{k+1} where $h_i = (f_{i,1}, f_{i,3}, \dots, f_{i,k-1})$ and each $f_{i,j}$ corresponds to a distinct $n2^n (= 2^{o(m)})$ -bit portion of the output of $S_{\bar{e}(t)}(x)$, we have $|Pr\{C_{kn}[r] = 1\} - Pr\{C_{kn}[\psi_2^{(\rho)}(h_{k+1}, \dots, h_2, h_1)] = 1\}| < 1/P(t)$.*

6. Practical Block Ciphers

PSBC with key-expanding requires still quasi-polynomial amount of memory to specify an enciphering/deciphering algorithm. In addition, the enciphering/deciphering part uses only an extremely small portion of the output of the key-expanding part.

Experience tells us that concatenating a number of transformations, each of which may not be so cryptographically strong, can produce a very strong one [M]. This folklore has recently been proved to be correct by Luby and Rackoff. See Theorem 2 in the preliminary version of [LR].

Along this guideline, we consider how to modify PSBC with key-expanding so that it is practically secure and can be implemented with current technology. We focus on the following three aspects: the size of a key, the sizes of n and k , and the rounds of transformations.

1. A key should be relatively short to make the cipher easy to be implemented. However to beat back the exhaustive search attack, the key should not be too short.
2. n should not be too large since it takes $n2^n$ bits to specify a random function from H_n . However, kn and hence k should be sufficiently large, otherwise the cipher is insecure even against the trivial exhaustive search attack.
3. When a relatively short key and a small n are chosen, the strength of the cipher will be significantly reduced. An effective method of resolving the problem is increasing the number of rounds of transformations.

The remaining part of this section proposes four example ciphers which we hope are secure enough for practical applications. Main parameters of the ciphers are collected in Table 1. For completeness, the definitions of the parameters are summarized below the table.

These parameters are chosen according to the preceding three aspects. In addition, $n = 4$ and $n = 8$ are chosen for easier implementation by software and/or hardware. The key-expanding part of each example cipher is realized by the algorithm G_m expanding a key of length $2m2^m$ bits into a long string of length $2m2^{2m}$ bits. All output bits of G_m are used by the enciphering part.

Notice that in Examples 2 and 3, the output of G_m is only half of the bits required by the enciphering part. We take two $2m2^{2m}$ -bit strings, and use G_m to stretch them into $2m2^{2m}$ -bit ones. Then we combine the $2m2^{2m}$ -bit strings into a $4m2^{2m}$ -bit one. A recommended method for combining strings is concatenating them in bit/bits unit.

7. Conclusion

We have investigated various types of transformations, and showed that among them Generalized Type-2 transformations are the most preferable. Two provably secure block ciphers, PSBC and PSBC with key-expanding, have been constructed by the use of Generalized Type-2 transformations. And finally, based on PSBC with key-expanding, practically implementable block ciphers have been presented.

Table 1 Four Example Ciphers

Parameters	Example 1	Example 2	Example 3	Example 4
Length of Plaintext/ Ciphertext (bits)	64	96	112	128
Length of Key (bits)	768	1536 (= $2 \cdot 768$)*	3584 (= $2 \cdot 1792$)*	4096
Size of Enciphering Part (kilo-bytes)	6	12	56	128
s (rounds)	96	128	32	64
n (bits)	4	4	8	8
k	16	24	14	16
m	6	6	7	8

* (G_m is used twice)

Definitions of Parameters

- Length of Plaintext/Ciphertext = $n \cdot k$ (bits).
- Length of Key = $t \cdot 2 \cdot m \cdot 2^m$ (bits), where t is the number of times G_m is applied.
- Size of Enciphering Part = $\ell \cdot s \cdot n \cdot 2^n$ (bits) = $\ell \cdot s \cdot n \cdot 2^n / 2^{13}$ (kilo-bytes).
- s — the number of rounds of Generalized Type-2 transformations applied in the enciphering/deciphering part.
- n — the length of a substring B_i (or C_i).
- k — (= 2ℓ) the number of substrings B_i 's (or C_i 's).
- m — specifying the length, $2m2^m$, of an input to G_m .

References

- [BM] M. Blum and S. Micali: "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM Journal on Computing*, Vol. 13, No. 4, (1984), pp.850-864.
- [FNS] H. Feistel, W.A. Notz and J.L. Smith: "Some cryptographic techniques for machine-to-machine data communications," *Proceedings of IEEE*, Vol. 63, No. 11, (1975), pp.1545-1554.
- [GGM] O. Goldreich, S. Goldwasser and S. Micali: "How to construct random functions," *Journal of ACM*, Vol. 33, No. 4, (1986), pp.792-807.
- [K] A.G. Konheim: *Cryptography: A Primer*, John Wiley & Sons, Inc. (1981).
- [L] L.A. Levin: "One-way functions and pseudorandom generators," *Combinatorica*, Vol. 7, No. 4, (1987), pp.357-363.
- [LR] M. Luby and C. Rackoff: "How to construct pseudorandom permutations from pseudorandom functions," *SIAM Journal on Computing*, Vol. 17, No. 2, (1988), pp.373-386.

- [M] C.H. Meyer: "Ciphertext/plaintext and ciphertext/key dependence vs number of rounds for the data encryption standard," *AFIPS Conference Proceedings*, Vol. 47, (1978), pp.1119-1126.
- [NBS] *Data Encryption Standard*, Federal Information Processing Standards (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, (1977).
- [O] Y. Ohnishi: "A study on data security," *Master Thesis* (in Japanese), Tohoku University, Japan, (1988).
- [R] R.A. Rueppel: "On the security of Schnorr's pseudorandom generator," *Presented at EUROCRYPT'89*, Houthalen, (April 10-13, 1989).
- [S] C.P. Schnorr: "On the construction of random number generators and random function generators," *Advances in Cryptology — EUROCRYPT'88*, LNCS Vol. 330, Springer-Verlag, (1988), pp.225-232.
- [Y] A.C. Yao: "Theory and applications of trapdoor functions," *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, (1982), pp.80-91.
- [ZMI] Y. Zheng, T. Matsumoto and H. Imai: "Impossibility and optimality results on constructing pseudorandom permutations," *Presented at EUROCRYPT'89*, Houthalen, (April 10-13, 1989).

Appendix A — Minimum Rounds for Security

This appendix discusses minimum rounds for achieving security when a permutation is constructed from some kind of transformations.

First we consider transformations related to Type-3 ones. For these transformations we have the following useful lemma.

[Lemma A1] *Let P be a subset of H_{kn} . If for any function $p \in P$ and for any input $s = (s_i, s_\ell, s_j) \in I_{kn}$, the output of p takes the form of $(\dots, s_\ell \oplus (\dots \hat{s}_\ell \dots), \dots)$, where $0 \leq i, \ell, j \leq kn, \ell \geq 1, i + \ell + j = kn, s_i \in I_i, s_\ell \in I_\ell, s_j \in I_j$ and $(\dots \hat{s}_\ell \dots)$ means that the string does not depend on s_ℓ , then there is a simple oracle circuit distinguishing between a function $p \in P$ and a function randomly and uniformly selected from H_{kn} .*

Recall that for a function-tuple $h_i = (f_{i,1}, f_{i,2}, \dots, f_{i,k-1})$, where $f_{i,j} \in H_n$, a Type-3 transformation is defined as $g_{3,i} = L_{rot}^{(1)} \circ \pi_{3,i}$, where $\pi_{3,i}(B_1, B_2, \dots, B_k) = (B_1, B_2 \oplus f_{i,1}(B_1), B_3 \oplus f_{i,2}(B_2), \dots, B_k \oplus f_{i,k-1}(B_{k-1}))$, and that for s function-tuples h_1, h_2, \dots, h_s , a permutation consisting of s rounds of Type-3 transformations is defined as $\psi_3(h_s, \dots, h_2, h_1) = g_{3,s} \circ \dots \circ g_{3,2} \circ g_{3,1}$.

Let $\pi_{\tau,i}$ be a permutation obtained from $\pi_{3,i}$ by dropping certain functions $f_{i,j}$ in $h_i = (f_{i,1}, f_{i,2}, \dots, f_{i,k-1})$. ($\pi_{\tau,i} = \pi_{3,i}$ when dropping no function.) Define $g_{\tau,i} = L_{rot}^{(1)} \circ \pi_{\tau,i}$, and call it a Type- τ transformation. Also let $\psi_\tau(h_s, \dots, h_2, h_1)$ be a permutation in H_{kn} consisting of s rounds of Type- τ transformations.

Let λ_τ denote the minimum number s at which $\psi_\tau(h_s, \dots, h_2, h_1)$ is secure. Now we are in a position to prove that $\lambda_\tau \geq k + 1$, i.e., a necessary condition for

$\psi_\tau(h_s, \dots, h_1, h_1)$ to be secure is that $s \geq k + 1$. Formally, we have

[Theorem A2] (1) $\lambda_\tau \geq k + 1$. (2) $\lambda_1 \geq 2k - 1$.

A type of transformations is called *singular* if a transformation g of that type is defined as $g(B_1, \dots, B_i, \dots, B_k) = (C_1, \dots, C_{j-1}, C_j, C_{j+1}, \dots, C_k)$, where $C_j = B_i$ and neither (C_1, \dots, C_{j-1}) nor (C_{j+1}, \dots, C_k) depends on B_i . For example, Type-1 transformations are singular, but Type-2 and Type-3 ones are non-singular.

[Theorem A3] Let Q be a polynomial in n and C_{kn} be an oracle circuit with $Q(n) < 2^n$ oracle gates. Let $\psi_\tau(h_s, \dots, h_2, h_1)$ be a permutation consisting of s rounds of singular Type- τ transformations, where h_s, \dots, h_2, h_1 are independent random function tuples. Then (1) when $s \leq k + 1$, $\psi_\tau(h_s, \dots, h_2, h_1)$ is insecure, and (2) when $s = 2k - 1$, $|\Pr\{C_{kn}[r] = 1\} - \Pr\{C_{kn}[\psi_\tau(h_s, \dots, h_2, h_1)] = 1\}| \leq \frac{(k-1)^2 Q(n)^2}{2^n}$, where $r \in_R H_{kn}$.

[Theorem A4] Let Q be a polynomial in n and C_{kn} be an oracle circuit with $Q(n) < 2^n$ oracle gates, and let $\psi_\tau(h_s, \dots, h_2, h_1)$ be a permutation consisting of s rounds of non-singular Type- τ transformations, where h_s, \dots, h_2, h_1 are independent random function tuples. Then when $s = k + 1$, $|\Pr\{C_{kn}[r] = 1\} - \Pr\{C_{kn}[\psi_\tau(h_s, \dots, h_2, h_1)] = 1\}| \leq \frac{k(k-1)Q(n)^2}{2^{n+1}}$, where $r \in_R H_{kn}$.

Now we consider Generalized Type- τ transformations $g_{\tau,i}^{(\rho)} = L_{rot}^{(\rho)} \circ \pi_{\tau,i}$. Let $\psi_\tau^{(\rho)}(h_s, \dots, h_2, h_1)$ be a permutation in H_{kn} consisting of s rounds of Generalized Type- τ transformations. Denote by $\lambda_\tau^{(\rho)}$ the minimum number s at which $\psi_\tau^{(\rho)}(h_s, \dots, h_2, h_1)$ is secure, where h_i are independent random function tuples. In particular, $\lambda_\tau^{(\rho)}$ is defined to be $+\infty$ if $\psi_\tau^{(\rho)}(h_s, \dots, h_2, h_1)$ is insecure no matter how large s is. We have the following theorem which is easy to prove.

[Theorem A5] (1) $\lambda_\tau^{(\rho)} \geq \lambda_\tau \geq k + 1$. (2) $\lambda_2^{(\rho)} = \lambda_2$ when ρ is an odd integer in $[1, k]$, and $\lambda_2^{(\rho)} = +\infty$ when ρ is an even integer in $[1, k]$.

Appendix B — Optimal Transformations

The computing procedures for Generalized Type- τ transformations $g_{\tau,i}^{(\rho)}$, and hence for $\psi_\tau^{(\rho)}(h_s, \dots, h_2, h_1)$, can be represented by acyclic *computation graphs*. There are three kinds of nodes in a computation graph: input nodes, output nodes and internal nodes. Each internal node in a computation graph represents a generic operation: computing a function $f_{i,j}$ or XORing two strings.

The length of a path between two nodes is defined as the number of arcs in the path. Now assume that the length of the longest path(s) from input nodes to output nodes in a computation graph is L . Then the *depth* of the graph is defined to be $L - 1$. The *normal-delay* $D^+(\rho, \tau, s)$ of a permutation $\psi_\tau^{(\rho)}(h_s, \dots, h_2, h_1) \in H_{kn}$ is defined as the depth of the computation graph for the permutation, the *inverse-delay* $D^-(\rho, \tau, s)$ is defined as that for the inverse of the permutation, and the *sum-delay* $D(\rho, \tau, s)$ is defined as $D(\rho, \tau, s) = D^+(\rho, \tau, s) + D^-(\rho, \tau, s)$.

Clearly, $D^+(\rho, \tau, s) = 2s$, and $D^-(\rho, \tau, s) \geq D^+(\rho, \tau, s) \geq 2s$. Thus $D(\rho, \tau, s) = D^+(\rho, \tau, s) + D^-(\rho, \tau, s) \geq 2D^+(\rho, \tau, s) \geq 4s$.

Recall that $\lambda_\tau^{(\rho)}$ denotes the minimum number of rounds s at which $\psi_\tau^{(\rho)}(h_s, \dots, h_2, h_1)$ is secure. From Theorem A5, we have $\lambda_\tau^{(\rho)} \geq k + 1$. Hence, $D(\rho, \tau, \lambda_\tau^{(\rho)}) \geq 4(k + 1)$.

Call a Generalized Type- τ transformation *optimal* if $D(\rho, \tau, \lambda_\tau^{(\rho)}) = 4(k + 1)$.

Now we discuss the optimality of transformations. First we have two facts: (1) When $g_{\tau,i}^{(\rho)} = L_{rot}^{(\rho)} \circ \pi_{\tau,i}^{(\rho)}$ with $\pi_{\tau,i}^{(\rho)}$ being not an involution, we have $D^-(\rho, \tau, s) > 2s$. So, transformations like Type-3 cannot be optimal. (2) When $\pi_{\tau,i}^{(\rho)}$ is an involution but $g_{\tau,i}^{(\rho)}$ is singular, we have $\lambda_\tau^{(\rho)} > k + 1$, and hence $D(\rho, \tau, \lambda_\tau^{(\rho)}) > 4(k + 1)$. So transformations like Type-1 cannot be optimal.

Consider the following two cases: odd k and even k . In the former case, either $g_{\tau,i}^{(\rho)} = L_{rot}^{(\rho)} \circ \pi_{\tau,i}^{(\rho)}$ is singular or $\pi_{\tau,i}^{(\rho)}$ is not an involution. Thus by the above two facts, no optimal transformation can be obtained. In the latter case, it is not hard to verify that the only non-singular transformations $g_{\tau,i}^{(\rho)} = L_{rot}^{(\rho)} \circ \pi_{\tau,i}^{(\rho)}$ with $\pi_{\tau,i}^{(\rho)}$ being involutions are Generalized Type-2 ones with ρ odd. For such transformations we have $D(\rho, 2, \lambda_2^{(\rho)}) = 4(k + 1)$. Thus we have proved:

[Theorem B1] *Among all types of transformations discussed in this paper, Generalized Type-2 transformations $g_{2,i}^{(\rho)} = L_{rot}^{(\rho)} \circ \pi_{2,i}^{(\rho)}$ with even k and odd ρ , are the only optimal ones.*

Appendix C — Super-Security

Luby and Rackoff introduced also the notion of *super-secure pseudorandom permutation generators* in [LR]. Intuitively, a pseudorandom permutation generator is super-secure if no super-oracle circuit can tell a permutation randomly specified by the generator from a randomly and uniformly chosen one. A *super-oracle circuit* is an oracle circuit with two kinds of oracle gates. The first is called the *normal* oracle gates which are evaluated using some permutation, and the second the *inverse* oracle gates which are evaluated using the inverse of the permutation.

When a secure pseudorandom permutation generator is used to construct a block cipher, the cipher is secure against the chosen plaintext attack, but not necessarily secure against the chosen plaintext/ciphertext attack. When a super-secure pseudorandom permutation generator is used to construct a block cipher, the cipher is secure against the chosen plaintext/ciphertext attack [LR].

Luby and Rackoff showed that functions consisting of 4 rounds of FTT's are super-secure. We can generalize their result to the following one.

[Theorem C1] *Let $k = 2\ell$, where $\ell \in \mathcal{N}$, and let ρ be an odd integer in $[1, k]$. Assume that $\psi_2^{(\rho)}(h_s, \dots, h_2, h_1)$ consists of s rounds of Generalized Type-2 transformations, where $h_i = (f_{i,1}, f_{i,3}, \dots, f_{i,k-1})$ with $f_{i,j} \in_R H_n$. Then $\psi_2^{(\rho)}(h_s, \dots, h_2, h_1)$ is super-secure iff $s \geq k + 2$.*

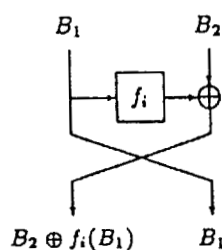


Figure 1: Feistel-Type Transformation (FTT)

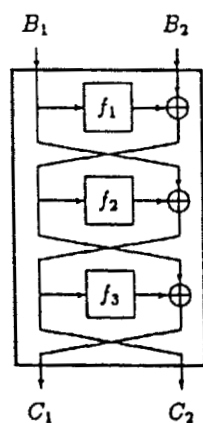


Figure 2: Feistel-Type Transformation (FTT) Lemma

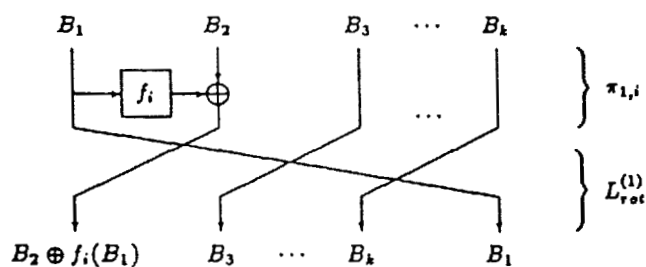


Figure 3: Type-1 Transformation

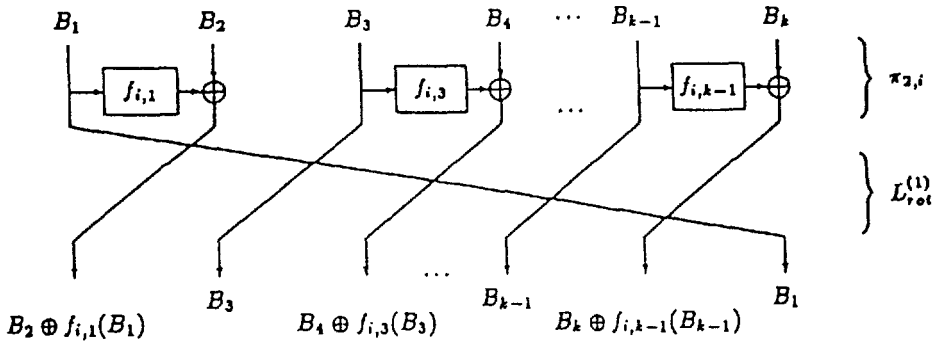


Figure 4: Type-2 Transformation

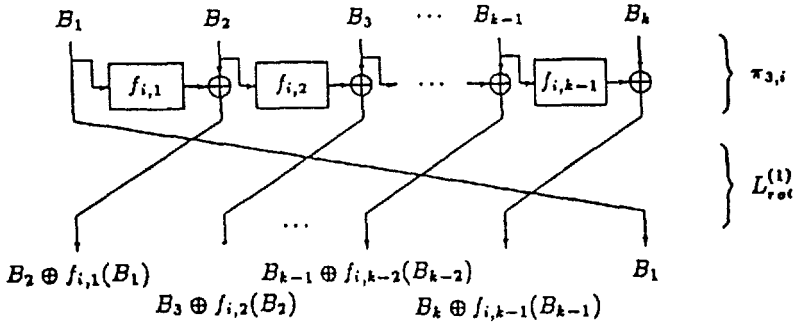


Figure 5: Type-3 Transformation

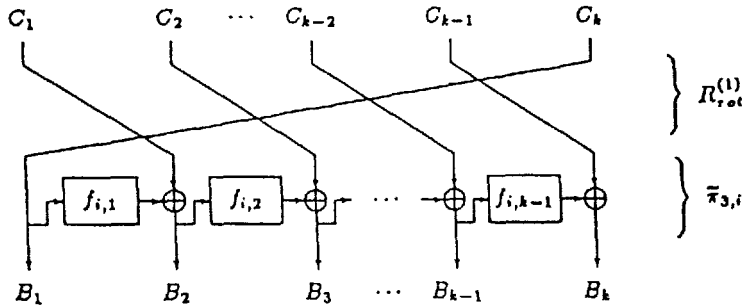


Figure 6: Inverse of Type-3 Transformation

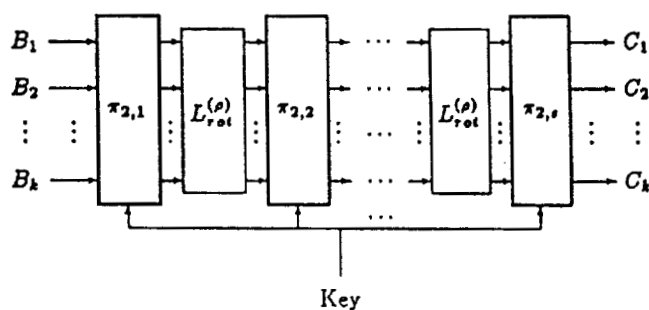


Figure 7: Enciphering Algorithm for PSBC

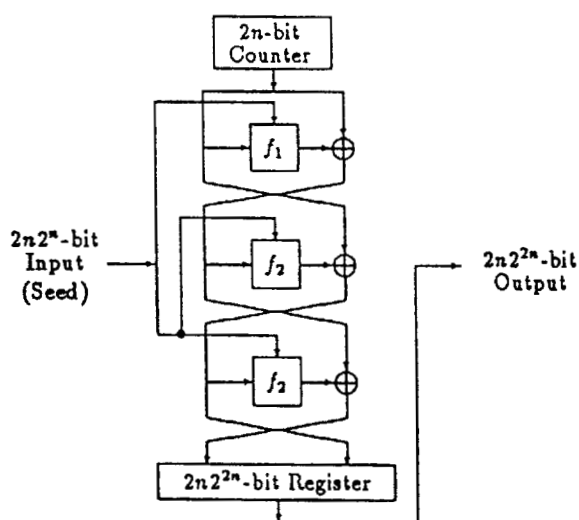
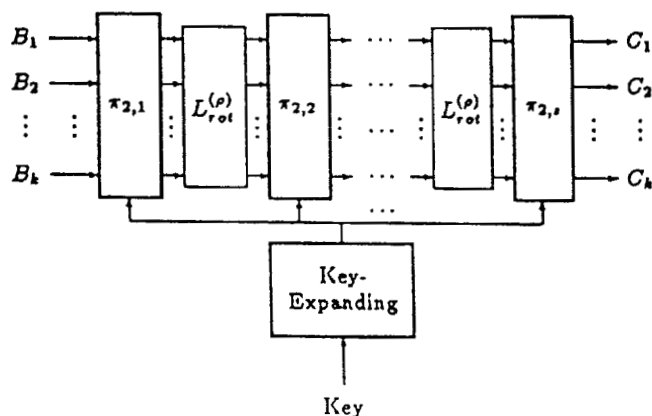
Figure 8: Algorithm G_n for Computing S_{2n2^n} 

Figure 9: Enciphering Algorithm for PSBC with Key-Expanding