# CRYPTOGRAPHIC ALGORITHMS: PROPERTIES, DESIGN AND ANALYSIS *

Josef Pieprzyk†

Department of Computer Science,
Centre for Computer Security Research,
University of Wollongong,
Wollongong, NSW 2500, AUSTRALIA,
*e-mail: josef@cs.uow.edu.au*

**Abstract**

The paper presents an overview of recent developments in the design of cryptographic algorithms. A short historical introduction sheds a ray of light on some events which contributed to the advancement of cryptology. Modern cryptology is intimately tied up to the fundamental Shannon's work on secrecy systems. First modern cryptographic algorithms (Lucifer and DES) are discussed in terms of their impact on the next generation of conventional crypto-algorithms. Next algebraic structures of both conventional and conditionally secure crypto-algorithms are investigated and an account of the results achieved is provided. Later provably secure crypto-algorithms are explored including pseudorandom bit generators, one-way hashing and pseudorandom functions. The work concludes with the review of main results in the design of S-boxes.

## 1 Introduction

Secret writing was probably the first widely used method for secure communication via insecure channel. The secret text was invisible to an unsuspicious reader. This method of secure communication was rather weak if the document found its way to an attacker who was an expert in secret writing. Cryptology in its early years resembled very much secret writing – the well-known Caesar cipher [56] is an excellent example of concealment by ignorance. This cipher was used to encrypt military orders. This time the ciphertext was not hidden but characters were transformed using a very simple substitution. It was reasonable to assume that the cipher was "strong" enough as most of the potential attackers were illiterate and hopefully the rest thought that the document was written in an unknown foreign language.

It was quickly realized that the assumption about an ignorant attacker was not realistic. Most early European ciphers were designed to withstand attacks of educated opponents who knew the encryption process but did not know the cryptographic key. Additionally it was requested that encryption and decryption processes could be done quickly usually by hand or with the aid of mechanical devices such as the cipher disk invented by Leon Battista Alberti [29]. At

*Invited lecture, PRAGOCRYPT'96, the 1st International Conference on the Theory and Applications of Cryptology, September 30 - October 3, 1996, Prague, Czech Republic

†Support for this project was provided in part by the Australian Research Council under the reference number A49530480 and the ATERB grant

the beginning of the nineteenth century first mechanical-electrical machines were introduced for "fast" encryption. This was the first breakthrough in cryptography. Cryptographic operations (in this case encryption and decryption) could be done automatically with a minimal involvement of the operator. Cipher machines could handle relatively large volume of data. The German ENIGMA and Japanese PURPLE are examples of cipher machines. They were used to protect military and diplomatic information.

The basic three rotor ENIGMA was broken by Rejewski, Rozycki and Zygalski, a team of three Polish mathematicians. Their attack exploited weaknesses of the operating procedure used by the sender to communicate the settings of machine rotors to the receiver (see [14]). The British team with Alan Turing at Bletchley perfected the attack and broke the strengthened five rotor ENIGMA. Japanese PURPLE was broken due to the effort of Friedman [29]. These remarkable feats were possible due to careful analysis of the cryptographic algorithms, predictable selection of cipher machine parameters (bad operational procedures), and significant improvement of computational power. Cryptanalysis was first supported by application of the so-called crypto bombs which were copies of the original cipher machines used to test some of the possible initial settings. Later cryptanalysts applied early computers to speed up computations.

The advent of computers gave both the designers and cryptanalysts a new powerful tool for fast computations. New cryptographic algorithms were designed and new attacks were developed to break them. New impetus for Cryptology was not given by new designing tools but rather by new emerging applications of computers and new requirements for the protection of information. Distributed computations and sharing information in computer networks are among those new applications which demonstrated, sometimes very dramatically, the necessity of providing tools for reliable and secure information delivery. Recent progress in Internet applications illustrates the fact that new services can be put on the net only after a careful analysis of their security features. Secrecy is no longer the most important security issue. In the network environment, authenticity of messages and correct identification of users became two most important requirements.

The scope of Cryptology has increased dramatically. It is now seen as the field which provides the theory and a practical guide for the design and analysis of cryptographic tools which then can be used to build up complex secure services. The secrecy part of the field, traditionally concentrated around the design of new encryption algorithms, was enriched by the addition of authentication, cryptographic hashing, digital signature and secret sharing schemes.

The paper focuses on a small part of Cryptology namely the field of the design and analysis of cryptographic block algorithms. The aim of the work is to show different aspects of the field and how they overlap and interrelate. We also point out possible future developments in the area. Section 2 depicts main concepts and issues. Section 3 presents algebraic properties of conventional and public-key crypto-algorithms. Provably secure crypto-algorithms are discussed in Section 4. Finally, the main results in the S-box theory are reviewed in Section 5.

## 2 Modern Cryptographic Algorithms

Shannon in his seminal work [59] laid the theoretical foundations of modern cryptography. He used information theory to analyse ciphers. He defined *the unicity distance* in order to characterize the strength of a cipher against an opponent with unlimited computational power. He also considered the so-called *product ciphers*. Product ciphers use small substitution boxes connected by larger permutation boxes. Substitution boxes (also called S-boxes) are controlled by a relatively short cryptographic key. They provide confusion (because of the unknown secret key). Permutation boxes (P-boxes) have no key – their structure is fixed and they provide

diffusion. Product ciphers are also termed *substitution-permutation (S-P) networks*. As the decryption process applies the inverses of S-boxes and P-boxes in the reverse order, decryption in general cannot be implemented using the encryption routine. This is expensive in terms of both hardware and software.

Feistel [22] used S-P network concept to design the Lucifer encryption algorithm. It encrypts 128-bit messages into 128-bit cryptograms using 128-bit cryptographic key. The designers of the Lucifer algorithm was able to modify the S-P network in such a way that both the encryption and decryption algorithms could be implemented by a single program or a piece of hardware. Encryption (or decryption) is done in sixteen iterations (also called rounds). Each round acts on 128-bit input $(L_i, R_i)$ and generates 128-bit output $(L_{i+1}, R_{i+1})$ using 64-bit partial key $K_i$. A single round can be described as

$$\begin{aligned} R_{i+1} &= L_i \oplus f(R_i, K_i) \\ L_{i+1} &= R_i \end{aligned} \tag{1}$$

where $L_i$ and $R_i$ are 64-bit long sequences, $f(R_i, K_i)$ is a cryptographic function which represents a simple S-P network. In literature, the transformation defined by (1) is referred to as the Feistel permutation. Note that a round in the Lucifer algorithm always is a permutation no matter what is the form of the function $f()$. Also the inverse of a round can use the original round routine with the swapped input halves. The strength of the Lucifer algorithm directly relates to the strength of the cryptographic function $f()$. Another interesting observation is that the structure of a single round has a "cryptographic amplification" property – the design of a Lucifer-type cryptosystem is equivalent to the design of its $f()$ function which operates on shorter sequences.

The Data Encryption Standard (DES) was developed from Lucifer (see [62]) and very soon became a standard for encryption in banking and other non-military applications. It uses the same Feistel structure with shorter 64-bit message/cryptogram blocks and shorter 64-bit key. As a matter of fact the key contains 56 independent and 8 parity-check bits. Due to its wide utilization, the DES was extensively investigated and analysed. Differential cryptanalysis invented by Biham and Shamir [2] were first applied for the DES. Also linear cryptanalysis by Matsui [34] was first tested on the DES.

The experience with the analysis of the DES gave a valuable insight into design properties of cryptographic algorithms. Successors of the DES whose structure was based on Feistel permutation are amongst many Japanese Fast Encryption Algorithm (FEAL) [60] and the Australian LOKI algorithm [8].

Cryptographic hashing became an important component of cryptographic primitives especially in the context of efficient generation of digital signatures. MD4 and its more secure version MD5 [50] are examples of the design which combines Feistel structure with C language bitwise operations for fast hashing. Although both MD4 and MD5 were shown to have security flaws (see Dobbertin's attacks [19],[20]), their design principles were used to create more secure hashing algorithms such as HAVAL [70].

Both encryption and hashing algorithms can be designed using one-way functions. These constructions are conditionally secure as the security of the algorithms depends upon the difficulty of reversing the underlying one-way functions. This concept was articulated by Diffie and Hellman in their visionary paper [18] in 1976. Soon after in 1978 two practical implementations of public-key cryptosystems were published. Rivest, Shamir and Adleman [51] based their algorithm (RSA system) on two one-way functions: factorization and discrete logarithm. Merkle and Hellman [35] used the knapsack function. The Merkle-Hellman cryptosystem was broken six years later by Brickell [7]

The conventional approach to the design of cryptographic algorithms exploits Shannon S-P networks. The outcome is always a single crypto-algorithm with a fixed security parameter

(the size of input or output). The DES is an example of a such design. Its security parameter is $n = 64$ (or $n = 56$). On the other hand, the number-theoretical (or conditionally secure) approach uses specific one-way functions. As the result of the design process in the number-theoretical approach, a family of cryptographic algorithms (with a variable size of its input and output) is produced. The RSA can be seen as a family of crypto-algorithms. The members can be indexed by the modulus.

Conventional cryptographic algorithms have a limited life time – an algorithm "dies" if the exhaustive attack [1] has become possible due to the progress in computing technology. Conditionally secure cryptographic algorithms are insensitive to the increment of computational power of the attacker. It is enough to select larger security parameters for the algorithm and be sure that the algorithm is still secure.

Note that the design and analysis of conditionally secure cryptographic algorithms have very strong links with Complexity Theory and Number Theory. Surprisingly, some fields of Number Theory are now considered parts of Cryptology (for instance factorization algorithms, primality testing algorithms, etc.). To prove that a cryptographic algorithm based on one-way functions is secure, it is enough to show that the attacker faces a computational problem from the class **NP-P** (see [25]) provided the well known open question: Is **NP=P** ? will not be answered positively (as then the class **NP-P** is empty).

Brassard argues in [6] that if the quantum computer becomes a reality, a new complexity hierarchy will emerge with the discrete logarithm and factorization problems in the polynomial-time class.

# 3 Algebraic Structures

## 3.1 Conventional cryptographic algorithms

An encryption algorithm should allow a user to select an encryption function from a large enough collection of all possible functions by a random selection of a cryptographic key. Note that for a plaintext/ciphertext block of the size $n$ bits, the collection of all possible permutations contains $2^n!$ elements and is called the symmetric group. If we assume that the size of the key block is also $n$ bits, then the selection of permutations is restricted to $2^n$ out of $2^n!$ by random selection of the key. To generate a random permutation efficiently, it is enough to iterate simple (and possibly insecure) permutations many times (S-P network). The single iteration is controlled by a short cryptographic key. Therefore the iteration has to be seen as a collection of permutations each of which is indexed by the cryptographic key. The structure of iterations is crucial for the security of the algorithm.

Coppersmith and Grossman [15] studied iterations of basic permutations and their suitability to encryption. They defined the so-called *k-functions*. Each $k$-function along with its connection topology produces a single iteration permutation which can be used as a generator of other permutations by composing them. The authors proved that these generators produce at least the alternating group using a finite number of their compositions. It means that using composition and with generators of relatively simple structure, it is possible to produce at least half of all the permutations. Even and Goldreich [21] proved that the DES-like connection topology along with $k$-functions (Feistel permutation) can also generate the alternating group.

Consider a Feistel permutation described by Equation (1). The core cryptographic element is the function $g(R_i, K_i)$. Pieprzyk and Zhang [48] studied Feistel permutations with the function $g()$ restricted to one-to-one mappings. They proved that Feistel permutations with a one-to-one

[1] In the case of encryption algorithms, this means that the secret key space can be exhaustively searched. In the case of hashing algorithms, this means that the birthday attack becomes viable.

function $g()$ generate the alternating group. They showed that having $(2^{n/2})!$ generators, it is possible to produce $\frac{(2^n)!}{2}$ different permutations.

Bovey and Williamson reported in [5] that an ordered pair of generators can produce either the alternating group $\mathbf{A}_{V_n}$ or the symmetric group $\mathbf{S}_{V_n}$ with the probability greater than $1 - exp(-log^{1/2}2^n)$. So if we select the pair at random, there is a high probability that it generates at least $\mathbf{A}_{V_n}$.

Feistel permutations are also applicable for hashing (with and without cryptographic key). Rivest [50] used them in the MD4 and MD5 hashing algorithms. The single iteration (the generator) is controlled by a message block $X_i$ and is defined as

$$G_{X_i}(A,B,C,D) = (B,C,D,A + X_i + g(B,C,D))$$

where $(A,B,C,D)$ is the initial vector input ($A,B,C,D$ are 32-bit words), $X_i$ is a 32-bit message block, and the function $g()$ is a "mixing" S-box. The digest $MD$ in MD5 is obtained by applying 64 iterations so

$$MD = \underbrace{G_{X_{15}} \circ \ldots \circ G_{X_0}}_{64}$$

Tillich and Zemor based their $SL_2$ hashing scheme on two generators $A$ and $B$ (see [63]). Assume that $\pi : \{0,1\} \rightarrow \{A,B\}$ which takes 0 to A and 1 to B. The digest of a binary message of arbitrary length $x_1, \ldots, x_k$ is the product $\pi(x_1)\pi(x_2)\ldots\pi(x_k)$ in the $SL(2,2^n)$ group. Algebraic properties of the $SL_2$ hashing were investigated in [11], [12] and [26].

### 3.2 Conditionally secure cryptographic algorithms

Most of the conditionally secure cryptographic algorithms use exponentiation as the basic operation. The base of the exponentiation is a generator which defines a single iteration. The exponent specifies the number of iterations. There are two types of exponentiation

- Diffie-Hellman (DH) exponentiation (in $GF(p)$),
- RSA exponentiation (in $\mathcal{Z}_N$; $N = p \times q$, where $p$ and $q$ are primes).

Diffie and Hellman used exponentiation to design their public-key distribution scheme [18]. Using it, two parties can establish a secret key via a public exchange of messages. An attacker who has access to public channel, sees an integer $y = g^k \pmod p$. Knowing $y$ and two public elements: the generator $g$ and the modulus $p$, the attacker wants to compute the secret integer $k$. In other words, they face an instance of the well-known discrete logarithm problem which is believed to be intractable [42].

The DH exponentiation induces the multiplication group whose cycle is $p-1$. If $p$ is a prime, the cycle $p-1$ has at least two factors 2 and one or more other primes - the multiplicative group decomposes into two or more subgroups. An interesting case is when $p = 2^n$ and the cycle $p-1$ is a Mersenne prime. In this case any nonzero exponent has its inverse modulo $p-1$. As the result, this type of DH exponentiation can be used to convert linear equations into their exponent equivalents. This is useful if some of the elements need to be kept secret while still being accessible for computations. (see [13]).

In the DH exponentiation everybody knows the cycle of the multiplicative group – this is not the case in RSA exponentiation. The public modulus $N = p \times q$ has two factors $p$ and $q$. So the multiplicative group has the cycle $\gamma(N) = lcm(p-1, q-1)$. As the primes $p$ and $q$ are secret so is $\gamma(N)$. Knowing $C = M^e \pmod N$, the holder of the factorization of $N$ can recover $M$ by applying the exponent $d$ so $C^d = M^{e \times d} = M \pmod N$ and $d \times e = 1 \pmod{\gamma(N)}$.

The factorization of $\gamma(N)$ defines the algebraic structure of the multiplicative group. If $\gamma(N)$ contains many factors, there are many subgroups with short cycles. This leads to inherited weaknesses such as the lack of concealment of messages (i.e. there is a substantial fraction of elements for which $M^e = M \pmod N$ – see [3]) or vulnerability to the iteration attack [61].

The RSA exponentiation may use a very short exponent (see for example [32]). This can be very useful when the computing power of a party who applies the exponentiation is very limited (for example in smart cards).

The knapsack problem belongs to the **NP-complete** class. So its difficulty is higher than the difficulty of the discrete logarithm problem. Despite of this, very early application of knapsack for encryption turned to be a failure. Knapsack is a good example that the use of **NP-complete** problems does not guarantee secure cryptographic algorithms. **NP-complete** characterization of a problem is based on the existence of intractable instances. To get a secure crypto-algorithm, the designer has to prove that all instances employed in the algorithm (or in the worst case overwhelming majority) are intractable.

What would happen if complexity theory proved that **NP** = **P** ? Although this result looks unlikely, it is not unreasonable to consider some possible repercussions for Cryptology. The immediate consequence is that all crypto-algorithms based on **NP** problems would be insecure. The way out would be to design crypto-algorithms using problems whose complexity is higher than **NP**. The class of undecidable problems could be of a special significance. Their intractability is especially strong. There is no algorithm which solves an undecidable problem. Wagner and Magyarik [64] suggested the word problem in groups (this problem is undecidable) to design public-key crypto-algorithm.

# 4 Provably Secure Constructions

Crypto-algorithms use relatively simple transformations which are repeated many times during the cryptographic process. The selection of building blocks is to some extent arbitrary. All conventional crypto-algorithms without exceptions were designed without formal proof of their security. Even worse, as conventional crypto-algorithms have their parameters fixed for their life-time, any progress in computing technology tends to weaken them. To keep up with the progress in computing, it is necessary to design an infinite family of crypto-algorithms whose members are indexed by security parameters such as the length of message block. This approach has been already adopted in public-key algorithms. So for example, the RSA algorithm is immune against progress in computing technology. On the other hand, the RSA and other public-key algorithms are considered to be secure because the underlying one-way function is believed to be intractable. If the underlying function is proved to belong the class **P**, the system which uses it is insecure. All public-key algorithms are sensitive to progress in Complexity Theory[2].

A solution to this dilemma would be to design algorithms whose constructions are based on the assumption of existence of one-way function only. Even if it turned out that **P**=**NP**, the crypto-algorithms would be still secure. The only necessary modification would be the substitution of the compromised no-longer-one-way function by a new one-way function.

## 4.1 Pseudorandom Bit Generators

Yao in [66] considered the following scenario. Assume that we have two different bit generators $\mathcal{S} = \{S_n \mid n = 1, 2, \ldots\}$ and $\mathcal{S}' = \{S'_n \mid n = 1, 2, \ldots\}$. Each generator is a collection of instances indexed by the size of the output $n$. One of them, say $\mathcal{S}$, produces bits randomly

[2] New computing tools like quantum computers will certainly extend the class **P**.

with uniform probability. The other $\mathcal{S}'$ generates bits using a probabilistic polynomial time algorithm. An observer who has access to the output of one of the generators, tries to identify it. The observer uses a probabilistic polynomial-time algorithm called the *distinguisher*. The distinguisher collects large enough (but polynomial-size) sample of output bits and makes a decision: either "0" (the tested generator is $S$) or "1" (the tested generator is $S'$).

We say that a distinguisher $\mathcal{C} = \{C_n \mid n = 1, 2, \ldots\}$ is able to distinguish $\mathcal{S}$ from $\mathcal{S}'$ if there is an infinite sequence of indices $n = n_1, n_2, \ldots$ for which the distinguisher instance $C_n$ can tell apart $S_n$ from $S'_n$ with an arbitrary high probability using polynomial size of observed bits.

From a cryptographic point of view, a bit generator which is distinguishable from truly random one is insecure. Yao [66] proved that if a bit generator is indistinguishable from the truly random one, then the generator passes *the next bit test* and vice versa. It is said that a generator passes the next bit test if knowing a polynomial size of output bits of the generator, it is impossible to predict (using a probabilistic polynomial-time algorithm) the next bit better then by flipping an unbiased coin.

A bit generator which is indistinguishable from a truly random source (or equivalently passes the next bit test) is called *pseudorandom*. Levin [31] proved that there is a pseudorandom bit generator (PBG) if there exists a one-way function. Blum and Micali gave a construction for a PBG from a one-way function (see [4]).

### 4.2 Hard Bits

Pseudorandom generators yield a sequence of bits each of which is indistinguishable from the random bit. They are called the *hard bits*. A typical one-way function contains a mixture of easy-to-compute and other bits. The other bits usually cannot be distinguished from "biased" random bits. Goldreich and Levin [28] noted that hard bits *concentrate* the one-wayness of the function. They showed how to build a hard-core predicate for one-way functions and proved that it is always possible to extract $O(\log n)$ hard bits from any one-way function. If the number of hard bits needs to be larger, then a one-way function needs to be used as a building block in more complex constructions (see [55]).

### 4.3 One-way Hashing

Hashing is a cryptographic algorithm that creates an unforgeable fingerprint (or digest) of a message. Unforgeability means that it is computationally infeasible to find messages with the same digest. If two messages have the same digest, they are said to *collide*. Hashing algorithms can be divided into two categories: collision-free hash functions (or strong one-way hash functions) and one-way hash functions (or weak one-way hash functions). *A strong one-way hash function* $h()$ is a function which (1) can be applied to a message of arbitrary size, (2) produces a fixed size output, (3) given the message $M$ and the description of $h()$, it is easy to compute $h(M)$, (4) it is computationally intractable to find two distinct messages which hash to the same digest. *A weak one-way hash function* differs from a strong one in point (4) which should be rephrased as (4') given a randomly chosen message $M$, it is computationally infeasible to find a colliding message $M'$ such that $h(M) = h(M')$.

More precisely, $H$ is a family of hash functions $H = \{H_n \mid n \in \mathcal{N}\}$ where $\mathcal{N}$ is the set of natural numbers and $H_n : \Sigma^{l(n)} \to \Sigma^n$. $l(n)$ is a monotone increasing function $\mathcal{N} \to \mathcal{N}$. Let $F$ be a *collision finder*. $F$ is a probabilistic polynomial time algorithm such that in input $x \in \Sigma^{l(n)}$ and $h \in H_n$ outputs either "?" (cannot find) or a string $y \in \Sigma^{l(n)}$ such that $x \neq y$ and $h(x) = h(y)$. A formal definition of a universal one-way hash function (UOWHF) can be as follows.

Let $H$ be a computable and accessible hash function compressing $l(n)$-bit input into $n$-bit output strings and $F$ be a collision string finder. $H$ is a universal one-way hash function if for each $F$, and for each polynomial $Q$ and for all sufficiently large $n$,

$$\text{Prob}\{F(x,h) \neq ?\} < \frac{1}{Q(n)}$$

where $x \in \Sigma^{l(n)}$ and $h \in_r H_n$. The probability is computed over all $h \in H_n$, $x \in \Sigma^{l(n)}$ and the random choice of all finite strings that $F$ could have chosen.

A strong one-way hash function was defined by Damgard [16]. Note that in this case the finder is not given any input message. $H$ is called a collision free hash function if for each $F$, and for each polynomial $Q$, and for all sufficiently large $n$,

$$\text{Prob}\{F(h) \neq ?\} < \frac{1}{Q(n)}$$

where $h \in H_n$, and the probability $\text{Prob}\{F(h) \neq ?\}$ is computed over all $h \in H_n$ and the random choice of all finite strings that $F$ could have chosen.

Naor and Yung [36] introduced the concept of UOWHF and suggested a construction based on one-way permutations. Their construction is based on a UOWHF which compresses a single bit. Also they argued that the existence of secure signature scheme reduces to the existence of a UOWHF. Rompel [52] managed to design a UOWHF from any one-way function. Zheng, Matsumoto and Imai observed [69] that there is a duality between PBG and UOWHF. They presented a construction of UOWHF from Blum-Micali pseudorandom bit generators. For more details see [45].

Zheng, Hardjono and Pieprzyk [67] used UOWHFs to build a family of one-way functions which are polynomial-time computable and they have *the collision accessibility property*. The sibling intractable function family ($k$-SIFF) allows the design of functions with $k$ colliding input strings. Note that finding more than $k$ collisions amounts to the ability to reverse the underlying UOWHF.

### 4.4 Pseudorandom Functions

A function generator is a collection of functions with two properties: indexing and polynomial time evaluation. Let $l(n)$ be a polynomial in $n$, a function generator $F = \{F_n \mid F_n : \Sigma^n \rightarrow \Sigma^n, n \in N\}$ is a collection of functions with the following properties:

- Indexing: Each $F_n$ specifies for each $k$ of length $l(n)$ a function $f_{n,k} \in H_n$ where $H_n$ is the set of all functions from $\Sigma^n$ to $\Sigma^n$.
- Polynomial-time evaluation: Given a key $k \in \Sigma^{l(n)}$, and a string $x \in \Sigma^n$, $f_{n,k}(x)$ can be computed in polynomial time in $n$.

A pseudorandom function generator is a function generator that cannot be distinguished from a truly random one. In other words, it is a collection of functions on $n$-bit strings that cannot be distinguished from the set of all functions on $n$-bit strings. To determine whether a collection of functions can be distinguished from the set of all functions, distinguishing circuits for functions are used, which are similar to distinguishing circuits for bit generators but are more powerful. They can be modeled by oracle circuits. The exact definitions of oracle circuits and distinguishing circuits for functions and pseudorandom function generators can be found in [27] and [45]. Goldreich, Goldwasser and Micali [27] were able to construct a pseudorandom function generator, given a pseudorandom bit generator which stretched an $n$-bit seed to a $2n$-bit string.

### 4.5 Pseudorandom Permutations

Pseudorandom permutation generator (a family of permutations) cannot be distinguished from truly random permutations by any probabilistic polynomial time algorithm. Permutation generators can be designed using different building blocks such as one-way permutations, pseudorandom functions, etc. A construction based on a one-way permutation was given in [55].

Assume that we define the DES-type permutation as

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

where $R$ and $L$ are $n$-bit strings and $\|$ stands for concatenation of two strings. Having a sequence of functions $f_1, \ldots, f_i$, we can define the composition of their DES-type permutations as

$$\psi(f_1, \ldots, f_i) = D_{2n,f_i} \circ \ldots \circ D_{2n,f_1}$$

where $f_j : \Sigma^n \to \Sigma^n; j = 1, \ldots, i$ and $\Sigma = \{0, 1\}$. Luby and Rackoff [33] showed that $\psi(f_1, f_2, f_3)$ is pseudorandom family of permutations (the index $n$ specifies the member of the family) if $f_1, f_2, f_3$ are three independent pseudorandom functions. Ohnishi [43] improved their results and demonstrated that both $\psi(f, f, g)$ and $\psi(f, g, g)$ are pseudorandom if $f$ and $g$ are two different pseudorandom functions. Zheng, Matsumoto and Imai [68] gave construction for a distinguisher for any generator of the form $\psi(f^k, f^i, f^k)$ proving that it is impossible to design pseudorandom permutations using three rounds of DES with a single pseudorandom function $f$ ($f^k$ means the composition of the function $k$ times). Pieprzyk [46] proved that $\psi(f, f, f, f^i)$ is pseudorandom ($i \geq 2$) if $f$ is pseudorandom function.

As before the distinguisher is a probabilistic polynomial time algorithm which can query the tested generator about cryptograms (ciphertext) for chosen messages (plaintext). This process is implemented by so-called *normal oracle gates*. In other words, pseudorandom permutations are secure against *the chosen plaintext attack*.

### 4.6 Super Pseudorandom Permutations

The distinguisher can be made more powerful by allowing it to query not only about ciphertext for chosen plaintext but also about plaintext for a chosen ciphertext. In this case, the distinguisher has two kinds of oracle gates: normal oracle gates (to query about ciphertext) and inverse oracle gates (to query about plaintext). The total number of queries has to be polynomial in $n$ ($n$ indexes instance generator and corresponding instance distinguisher).

If a permutation generator cannot be distinguished from the truly random one by any probabilistic polynomial time distinguisher with normal and inverse oracle gates, then it is called *super pseudorandom*. Luby and Rackoff [33] proved that $\psi(e, f, g, h)$ is super pseudorandom as long as $e, f, g, h$ are different pseudorandom functions. Using two pseudorandom functions $f, g$, it is possible to get a super pseudorandom permutation $\psi(f, f, g, g)$. Sadeghiyan and Pieprzyk [54] showed that the generator $\psi(f, 1, f^2, f, 1, f^2)$ based on a single pseudorandom function, is super pseudorandom.

## 5 S-box Theory

The design of a good cryptographic algorithm of the DES structure is equivalent to the design of a single random function $f$. The number of necessary iterations is at least six (four iterations with $f$ and two with the identity permutation). Consider a random function $f = \{f_k : \Sigma^n \to \Sigma^n; k \in GF(2^{2n})\}$. For any instance of the key $k$, there should be an instance of a random

function $f_k$. To store all the $2^{2n}$ instances of random function, it is necessary to store $2^n \times 2^{2n}$ $n$-bit strings. This is impractical even for small $n$.

As the function $f$ is requested to be fast and to be implementable with a limited memory, it has to be non-random. Now we face some questions. How to assess the quality of the function $f$ when we have already assumed that it fails most of the statistical tests ? Which tests are crucial to ensure that the function $f$ captures "typical features" of random functions so as the resulting algorithms will be fast and cryptographically strong ?

The DES was the first cryptographic algorithm that was extensively scrutinized by the international community. It is still a good example of how to design a relatively strong algorithm from a non-random function $f()$. The function $f()$ was built using a collection of eight S-boxes concatenated by the $P$-box (S-P network). Although the DES algorithm was made public, the collection of tests used to select $S$-boxes and the $P$-box was never revealed. The collection of tests is equivalently referred in the literature as *the design criteria/properties*. It took a lot of effort by the international community to identify some of the design criteria. A summary of the identified design properties used during the design of S-boxes and the P-box can be found in [9] and [40].

S-box design criteria can be defined using the information theory concept of mutual information. This approach was applied by Forre in [23] and Dawson and Tavares in [17]. They argued that the mutual information between inputs and outputs of S-boxes should be as small as possible.

### 5.1 Design Criteria

The list of S-box design criteria usually includes

(1) balancedness,

(2) nonlinearity,

(3) strict avalanche criterion (SAC),

(4) higher-order SAC or propagation criterion and

(5) completeness.

Balancedness guarantees that S-boxes do not discriminate against any specific bit "0" or "1". S-boxes must not be affine or close to affine functions. The *nonlinearity* of a function $f : \Sigma^n \to \Sigma$ is defined as the Hamming distance between the function and the set of all affine functions [44]. The concept of nonlinearity can be extended to measure nonlinearity of arbitrary functions $f : \Sigma^n \to \Sigma^m$ including permutations ($n = m$) (see [47],[38],[58]). *Strict Avalanche Criterion* or SAC was introduced by Webster and Tavares [65]. A function $f : \Sigma^n \to \Sigma^m$ satisfies the SAC if $f(x \oplus \alpha)$ is balanced for all $x \in \Sigma^n$ and for all $\alpha$ whose weight is "1" ($wt(\alpha) = 1$). In other words, it characterizes the number of output bits which change their values in response to a single input bit change. *Higher order SAC* is a generalization of the SAC property. Both SAC and higher order SAC can be collectively called *propagation criteria* ([1],[49]). A function $f : \Sigma^n \to \Sigma^m$ satisfies a propagation criterion of degree $k$ if $f(x \oplus \alpha)$ is balanced for all $x \in \Sigma^n$ and for all $\alpha$ whose weight is k ($wt(\alpha) = k$). *Completeness* was defined by Kam and Davida [30] to characterize S-P network complexity - it requires each output to be dependent upon any input.

Additional useful design criteria are:

(6) linear nonequivalence,

(7) short algebraic normal forms,

(8) good XOR profile.

The collection of functions $F = \{f_1, \ldots, f_n\}$ (where each $f_i : \Sigma^n \to \Sigma$) is linearly nonequivalent if there is no affine transformation which converts a function $f_i$ into $f_j$ $(i \neq j)$ [10]. The requirement about a short algebraic normal form of a function becomes very important when the function is too big to be stored as the lookup table. In this case the output values of the functions are generated on the fly - the shorter the function is the quicker is evaluated (see hashing algorithms). Differential cryptanalysis [2] prompted a new general design criterion - a "good" XOR profile. A function $f : \Sigma^n \to \Sigma^m$ has a good XOR profile if for any fixed $\alpha$, $f(x) \oplus f(x \oplus \alpha)$ takes on $2^{m-1}$ values each $2^{n-m+1}$ times while the rest of $2^{m-1}$ values have empty entries $(x \in \Sigma^n)$.

### 5.2 Relation amongst S-box Design Criteria

In general, it is impossible to design a S-box which will meet the above criteria. For instance, bent functions have the maximum nonlinearity but are not balanced. To get a workable design, it is necessary to specify which criteria are crucial and what tradeoff is acceptable amongst them. The designer must also consider how S-boxes will be implemented. The implementation restrictions usually influence the size of S-boxes. Note that for some small sizes there exists no reasonable design. For instance any $(2 \times 2)$ S-box with balanced functions is always affine. In general, the bigger size of the S-box input $n$ the more flexibility is possible in the tradeoff amongst different criteria.

Balancedness is a criterion which must be satisfied unconditionally. Note that the nonlinearity of balanced functions is always smaller than the nonlinearity of bent functions which attain the maximum nonlinearity and satisfy SAC [53]. The tradeoff between nonlinearity and the propagation criterion (including the SAC) for balanced functions is discussed in [57] and [58]. Charnes and Pieprzyk [10] studied the relation between the nonlinearity and the linear nonequivalence. They showed that it is not possible to select five balanced, SAC satisfying, linear nonequivalent functions in five boolean variables without reduction of nonlinearity. Nyberg [37] discussed how to design S-boxes for which any linear combination of their outputs is a bent function, any output function has high nonlinearity and satisfy SAC, and additionally S-boxes are immune to differential cryptanalysis (have a good XOR profile).

### 5.3 New Trends in S-box Design

One may argue that S-boxes can be generated at random. The selected S-boxes can then be verified against a collection of the S-box criteria. O'Connor [41] analysed a class of such algorithms and concluded that they are infeasible for relatively small sizes of S-boxes. Systematic design of S-boxes is a growing area of the S-box theory (see for example [37],[57]). It produces a single "good" S-box.

There is a trend to design a family of crypto-algorithms with a security parameter $n$ instead of a single algorithm for fixed $n$. The parameter $n$ usually specifies the size (in bits) of the input and output. To claim that some security features hold for the whole family of algorithms, we need to know how to design a family of good S-boxes whose properties are "regular" and can be approximated from the S-box properties for small $n$. An important advantage of families of crypto-algorithms is that their security can be extensively tested for small $n$.

*An universal S-box* $S$ can be defined as a family of S-boxes, i.e. $S = \{S_n \mid n = n_1, n_2, \ldots\}$ where $n_1, n_2, \ldots$ is an infinite sequence of "acceptable" parameters. It is important to be able to

assess the tradeoff of design criteria for arbitrary $n$. The designers of the LOKI algorithm (see [8]) used exponentiation in $GF(2^8)$ to generate the S-boxes whose structure can be changed to make a private copy of the algorithm. Pieprzyk [47] proved that cubing and other related exponent permutations have high nonlinearity and their properties can be characterized for arbitrary $n$. Exponentiation can be a good source of universal S-boxes (see also [39]). In general, any polynomial $f(x) = x^a + \ldots + b_1 x + b_0$, or any function (for example $f(x) = g^x$) which generates a permutation can be a potential candidate for a good universal S-box. The difficult part is to prove to what degree the requested criteria are satisfied.

Universal S-boxes can also be used to accommodate cryptographic keys as a part of their input. The resulting S-box behaves as a S-box with a variable structure (which depends on the key).

## 6 Conclusions

There are the following classes of crypto-algorithms:

(1) conventional crypto-algorithms (security parameters are fixed),

(2) families of crypto-algorithms. (security parameters specify an instance),

(3) conditionally secure crypto-algorithms (such as public-key crypto-algorithms). The algorithm is based on a specific one-way function. An instance of the algorithm can be identified by the security parameter (like the modulus in the RSA),

(4) provably secure crypto-algorithms. These algorithms do not apply any specific one-way function.

S-box theory supports the design of crypto-algorithms from the classes (1) and (2). We have achieved an impressive progress in the design of single S-boxes. The author believes that the future work in S-box theory will address the problem of S-box design for the class (2). So the design of universal S-boxes seems to be the next step in advancement of the S-box theory.

In classes (3) and (4), One-way functions play the role of S-boxes. Note that all provably secure constructions (4) work under the assumption that there is a one-way function. There should be no surprise that most of algorithms in the class (4) are not very efficient - one has to sacrifice efficiency to gain generality (the constructions have to work for arbitrary one-way functions!).

Unfortunately, the collection of the known one-way functions is quite modest. Therefore, looking for new one-way functions seems to continue. At the same time, the known one-way functions can be used to produce one-way functions satisfying some security requirements. How to immunize public-key against some specific attacks, is an example of such research [24].

*ACKNOWLEDGMENT*

*The author thanks Chris Charnes, Rei Safavi-Naini, and Xian-Mo Zhang for their critical comments and discussions on the paper.*

## References

[1] C. Adams and S. Tavares. The structured design of cryptographically good S-boxes. *Journal of Cryptology*, 3:27–41, 1990.

[2] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[3] G.R. Blakley and I. Borosh. Rivest-Shamir-Adleman public-key cryptosystems do not always conceal messages. *Comp. and Maths with Appls*, 5:169–178, 1979.

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

[5] J. Bovey and A. Williamson. The probability of generating the symmetric group. *Bull. London Math. Soc.*, 10:91–96, 1978.

[6] G. Brassard. The impending demise of rsa. *RSA Newsletter*, Premier Issue, 1995.

[7] E.F. Brickell. Breaking iterated knapsacks. *Advances in Cryptology - CRYPTO'84, Lecture Notes in Computer Science, Blakley and Chaum (Eds)*, 196:342–358, 1984.

[8] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry. Improving resistance to differential cryptanalysis and the redesign of LOKI. *Advances in Cryptology - ASIACRYPT'91, Lecture Notes in Computer Science (H. Imai, R.L. Rivest and T. Matsumoto (Eds))*, 739:36–50, 1993.

[9] L.P. Brown. *Analysis of the DES and the design of the LOKI encryption scheme*. PhD Thesis, University of New South Wales, Canberra, Australia, 1991.

[10] C. Charnes and J. Pieprzyk. Linear nonequivalence versus nonlinearity. In *Avances in Cryptology - AUSCRYPT'92, Lecture Note s in Computer Science, Vol. 718, J.Seberry, Y.Zheng (Eds.)*, pages 156–164. Springer-Verlag, 1993.

[11] C. Charnes and J. Pieprzyk. Attacking the $SL_2$ hashing scheme. In *Advances in Cryptology - ASIACRYPT'94, J. Pieprzyk and R. Safavi-Naini (Eds), Lecture Notes in Computer Science, Vol.917*, pages 322–330. Springer Verlag, 1995.

[12] C. Charnes and J. Pieprzyk. Weak parameters for the $SL_2$ hash function. (in preparation), 1996.

[13] C. Charnes, J. Pieprzyk, and R. Safavi-Naini. Conditionally secure secret sharing schemes with disenrollment capability. In *Proceedings of the 2nd ACM Conference on Computer and Communication Security, November 2-4, 1994, Fairfax, Virginia*, pages 89–95, 1994.

[14] R.F. Churchhouse. The ENIGMA – some aspects of its history and solution. *IMA Bulletin*, 27:129–137, 1991.

[15] D. Coppersmith and E. Grossman. Generators for certain alternating groups with applications to cryptography. *SIAM Journal Appl. Math*, 29(4):624–627, 1975.

[16] I.B. Damgard. A design principle for hash functions. In *Advances in Cryptology, Proceedings of CRYPTO'89, Ed G. Brassard, Lecture Notes in Computer Science, Vol.435*, pages 416–427. Springer-Verlag, 1990.

[17] M.H. Dawson and S.E. Tavares. An expanded set of S-box design criteria based on information theory and its relation to differential-like attacks. In *Advances in Cryptology - EUROCRYPT'91, D.W. Davies (ED), Lecture Notes in Computer Science, Vol.547*, pages 352–367. Springer Verlag, 1991.

[18] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22(6):644–654, November 1976.

[19] H. Dobbertin. Cryptanalysis of MD4. In *Fast Software Encryption, Lecture Notes in Computer Science, Vol. 1039, D.Gollmann (Ed.)*, pages 71–82. Springer-Verlag, 1996.

[20] H. Dobbertin. Cryptanalysis of MD5 compress. Announcement on Internet, May 1996.

[21] S. Even and O. Goldreich. Des-like functions can generate the alternating group. *IEEE Transactions on Information Theory*, 29(6):863–865, November 1983.

[22] H. Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.

[23] R. Forre. Methods and instruments for designing S-boxes. *Journal of Cryptology*, 2(3):115–130, 1990.

[24] Y. Frankel and M. Yung. Cryptanalysis of the immunized LL public key systems. In *Advances in Cryptology CRYPTO'95, Proceedings of 15th Annual International Cryptology Conference, D. Coppersmith (Ed), Lecture Notes in Computer Science, Vol.963*, pages 287–296. Springer-Verlag, 1995.

[25] M. R. Garey and D. S. Johnson. *Computers and Intractibility A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, 1979.

[26] W. Geiselmann. A note on the hash function of tillich and zemor. In *Cryptography and Coding, C Boyd (Ed), LNCS, Vol.1025*, pages 257–263. Springer-Verlag, 1995.

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

[28] O. Goldreich and L.A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21th ACM Symposium on Theory of Computing*, pages 25–32, New York, 1989. ACM.

[29] D. Kahn. *The Codebreakers*. MacMillan, New York, 1967.

[30] J. Kam and G. Davida. Structured design of substitution-permutation networks. *IEEE Transactions on Computers*, C-28:747–753, 1979.

[31] L. A. Levin. One-way function and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[32] J.H. Loxton, D.S. Khoo, G.J. Bird, and J. Seberry. A cubic RSA code equivalent to factorization. *Journal of Cryptology*, 5:139–150, 1992.

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

[34] M. Matsui. Linear cryptanalysis method for DES cipher. Abstracts of EUROCRYPT'93, May 1993.

[35] R.C. Merkle and M.E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inform. Theory*, 24:525–530, 1978.

[36] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *The 21-st ACM Symposium on Theory of Computing*, pages 33–43, 1989.

[37] K. Nyberg. Perfect nonlinear S-boxes. In *Advances in Cryptology - EUROCRYPT'91, Lecture Notes in Computer Science, Vol.547*, pages 378–386. Springer Verlag, 1991.

[38] K. Nyberg. On the construction of highly nonlinear permutations. In *Advances in Cryptology - EUROCRYPT'92, Lecture Notes in Computer Science, Vol.658*, pages 92–98. Springer Verlag, 1992.

[39] K. Nyberg and L.R. Knudsen. Provable security against differential attack. *Journal of Cryptology*, 8(1):27–38, 1995.

[40] L.J. O'Connor. An analysis of product ciphers based on the properties of Boolean functions. PhD thesis, the University of Waterloo, 1992. Waterloo, Ontario, Canada.

[41] L.J. O'Connor. An analysis of a class of algorithms for S-box construction. *Journal of Cryptology*, 7(3):133–152, 1994.

[42] A.M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. *Proceedings of EUROCRYPT'84, Lecture Notes in Computer Science, Advances in Cryptology, T. Beth, N. Cot, I. Ingemarsson (Eds)*, 209:224–314, 1985.

[43] Y. Ohnishi. A study on data security. Master's thesis, Tohoku University, Japan, 1988.

[44] J. Pieprzyk and G. Finkelstein. Towards effective nonlinear cryptosystem design. *IEE Proceedings-E, Computers and Digital Techniques*, 135(6):325–335, November 1988.

[45] J. Pieprzyk and B. Sadeghiyan. *Design of Hashing Algorithms*. Springer-Verlag, New York, 1993.

[46] J.P. Pieprzyk. How to construct pseudorandom permutations from single pseudorandom functions. In *Advances in Cryptology - EUROCRYPT'90, Lecture Notes in Computer Science, Vol.473*, pages 140–150. Springer Verlag, May 1990.

[47] J.P. Pieprzyk. On bent permutations. In *Proceedings of the International Conference on Finite Fields, Coding Theory, and Advances in Communications and Computing, Las Vegas*, August 1991.

[48] J.P. Pieprzyk and Xian-Mo Zhang. Permutation generators of alternating groups. In *Advances in Cryptology - AUSCRYPT'90, J. Seberry, J. Pieprzyk (Eds), Lecture Notes in Computer Science, Vol.453*, pages 237–244. Springer Verlag, 1990.

[49] B. Preneel, W. Van Leewijck, L. Van Linden, R. Govaerts, and J. Vandewalle. Propagation characteristics of boolean functions. In *Advances in Cryptology - EUROCRYPT'90, Lecture Notes in Computer Science, Vol.473*, pages 161–173. Springer Verlag, May 1990.

[50] R. Rivest. The MD5 message digest algorithm. Request for Comments, RFC 1321, 1992.

[51] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[52] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *The 22-nd ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[53] O.S. Rothaus. On bent functions. *Journal of Combinatorial Theory*, 20:300–305, 1976.

[54] B. Sadeghiyan and J. Pieprzyk. A construction for super pseudorandom permutations from a single pseudorandom function. In *Advances in Cryptology - EUROCRYPT'92, R. Rueppel (Ed), Workshop on the Theory and Application of Cryptographic Techniques, Balatonfured, Hungary, May 1992, Lecture Notes in Computer Science, Vol.658*, pages 267–284. Springer Verlag, 1993.

[55] B. Sadeghiyan, Y. Zheng, and J. Pieprzyk. How to construct a family of strong one way permutations. In *Advances in Cryptology - ASIACRYPT'91, H. Imai, R. Rivest, T. Matsumoto (Eds), Lecture Notes in Computer Science, Vol.739*, pages 97–110. Springer Verlag, 1993.

[56] J. Seberry and J. Pieprzyk. *Cryptography: An Introduction to Computer Security*. Prentice Hall International, 1989.

[57] J. Seberry, X.M. Zhang, and Y. Zheng. Systematic generation of cryptographically robust S-boxes. Proceedings of the 1st ACM Conference on Computer and Communication Security, November 1993.

[58] J. Seberry, X.M. Zhang, and Y. Zheng. Nonlinearly balanced boolean functions and their propagation characteristics. In *Advances in Cryptology - CRYPTO'93, Lecture Notes in Computer Science (D.R Stinson (Ed))*, volume 773, pages 49–60, New York, 1994. Springer Verlag.

[59] C. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28:656–715, 1949.

[60] A. Shimizu and S. Miyaguchi. Fast data encryption algorithm feal. Abstracts of EUROCRYPT'87, Amsterdam, April 1987.

[61] G.J. Simmons and M.J. Norris. Preliminary comments on the MIT public-key cryptosystem. *Cryptologia*, 1:406–414, 1977.

[62] Data Encryption Standard. Federal information processing standard (fips). Publication 46, National Bureau of Standards, US Department of Commerce, January 1977.

[63] J-P. Tillich and G. Zemor. Hashing with $SL_2$. In *Advances in Cryptology - CRYPTO'94, Lecture Notes in Computer Science (Y. Desmedt (Ed))*, volume 839, pages 40–49, New York, 1994. Springer Verlag.

[64] N.R. Wagner and M.R. Magyarik. A public-key cryptosystem based on word problem. In *Advances in Cryptology. Proceedings of CRYPTO'84, Lecture Notes in Computer Science (Blakley, Chaum (Eds))*, volume 196, pages 19–35, New York, 1985. Springer Verlag.

[65] A.F. Webster and S.E. Tavares. On the design of S-boxes. In *Lecture Notes in Computer Science, Advances in Cryptology, Proceedings of Crypto'85*, pages 523–534. Springer-Verlag, 1985.

[66] 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.

[67] Y. Zheng, T. Hardjono, and J. Pieprzyk. The sibling intractable function family (SIFF): Notion, construction and applications. *IEICE Trans. Fundamentals*, E76-A:4–13, 1993.

[68] Y. Zheng, T. Matsumoto, and H. Imai. Impossibility and optimality results on constructing pseudorandom permutations. In *Advances in Cryptology - EUROCRYPT'89, Lecture Notes in Computer Science, Vol.434*, pages 412–422. Springer Verlag, April 1989.

[69] Y. Zheng, T. Matsumoto, and H. Imai. Duality between Two Cryptographic Primitives. The 8-th International Conference on Applied Algebra, Algebraic Algorithms and Error Correcting Codes, 1990.

[70] Y. Zheng, J. Pieprzyk, and J. Seberry. HAVAL - A one-way hashing algorithm with variable length of output. In *Avances in Cryptology - AUSCRYPT'92, Lecture Notes in Computer Science, Vol. 718, J.Seberry, Y.Zheng (Eds.)*, pages 83–104. Springer-Verlag, 1993.