A Secured Smart Card Using a Pseudorandom Affine Transformation Based Cipher and a Secured LIRKES

Ehab Mahmoud Mohamed ehab@mobcom.is.kyushu-u.ac.jp
Faculty of Engineering/
Advanced Information Technology Dept/
Wireless Communication Section/Kyushu University
Motooka 744, Nishi-ku, Fukuoka-city 819-0395, Japan
Phone +81-92-802-3573, Fax +81-92-802-3572,

Yassin Mahmoud Yassin Hasan ymyhasan@aun.edu.eg
Faculty of Engineering /Electrical Dept/
Electronics and Communication Section
Assuit University
Assuit, Egypt.

Hiroshi Furukawa furuhiro@is.kyushu-u.ac.jp
Faculty of Engineering/
Advanced Information Technology Dept/
Wireless Communication Section/Kyushu University
Motooka 744, Nishi-ku, Fukuoka-city 819-0395, Japan
Phone +81-92-802-3573, Fax +81-92-802-3572,

Abstract

The RKES (Remotely Keyed Encryption Schemes) are greatly useful in solving the vital problem of how to do bulk encryption/ decryption for high-bandwidth applications (like multimedia and video encryption) in a way that takes advantage of both the superior power of the host and the superior security of the smart card. According to this issue, we propose a novel length increasing (LI) RKES, in which, the output ciphertext length is larger than input plaintext length. In this scheme, an extra ciphertext block is used as a self validation or signature of the whole ciphertext, so an adversary can't forge the scheme.

The proposed LIRKES needs a strong pseudorandom permutation (PRP) as its basic building block, so we introduce a new symmetric-key block cipher, with variable block and key lengths, referred to as PATFC (Pseudorandom Affine Transformation based Feistel Cipher), appropriate for software and hardware implementations. PATFC adopts the 3-round Luby-Rackoff construction (a compact form of the Feistel network structures) for fusing pseudorandom functions of the plaintext partitions to obtain a pseudorandom permutation.

PATFC mainly makes use of a novel keyed pseudorandom function (PRF) that is based on a pseudorandom affine transformation (constructed using a highly nonlinear pseudorandom sequence generator) followed by a data and key dependent encoding and a simple hashing scheme.

Extensive statistical tests of PATFC and its underlying round function consistently demonstrated their competitive diffusion, confusion and pseudorandomness characteristics. Furthermore, PATFC is provably secure and not vulnerable to known/chosen/adaptive plaintext/ ciphertexts attacks.

At the end of this paper, we show how we can apply PATFC as a strong PRP in the suggested LIRKES to be used for smart cards.

Keywords: pseudorandom function (PF), pseudorandom permutation (PRP), Luby-Rackoff ciphers, Feistel Network (FN), LIRKES.

1. INTRODUCTION

Smart cards provide an effective tool for portable safe hardware storage of secret keys critically needed in many recent multimedia applications such as real time access control, software license management, e-technology, e-commerce and e-services [1]. Smart cards are mainly reliable because of their distinctive features of tamper-resistant packaging, loose coupling to the host and low cost [2]. However, with their computationally limited resources, smart cards cannot process large data blocks as fast as the host may need.

The Remotely Keyed Encryption Protocol (RKEP), first introduced by Blaze, addressed how to do bulk encryption/decryption taking advantage of both the superior computational power, speed and resources of the (high bandwidth) host (trusted with plaintexts/ciphertexts) and the superior security of the slow (low bandwidth) smart-card (trusted with the key) [2]. Although of the interesting approach of Blaze, it suffers from some drawbacks. Its drawbacks basically result from the low security of the protocol. Lucks gave three attacks on the blaze's RKEP, namely a chosen plaintext attack, a two sided attack and a forgery attack (working on the decrypt only smart-card) [3]. In addition, Lucks specified three conditions, that Blaze's RKEP does not satisfy any of them, to make a secure RKE scheme (RKES). Moreover, Lucks suggested the RaMaRK "Random Mapping based RKES" which is based on the Luby-Rackoff construction. Although RaMaRK is based upon Lucks' criteria, a critical weakness was found in RaMaRK [4]. Consequently, Blaze, Feigenbaum and Naor suggested an efficient Length Preserving (LP) RKES named BFN-LPRKES, in which the length of the output ciphertext is equal to the length of the input plaintext. Although the BFN-LPRKES is the most efficient scheme from the security point of view, it is not efficient from card computations and keys storages point of views which are critical requirements for inexpensive smart cards. The authors suggested a new LPRKS based upon a general view of the Feistel Network (FN), in which they only used 2-round PRP instead of the 4-round used by Luby-Rackoff. Our proposed LPRKES is more secure than the previous literature, and more efficient from complexity, card computations, and keys storages point of views [5] [6].

In addition to the BFN-LPRKES, Blaze, Feigenbaum and Naor suggested the length Increasing (LI) RKES named BFN-LIRKES as an alternative to solve the RKES problem. Their proposal is based upon adding a signature of the whole ciphertext to the output ciphertext which cannot be computed by an adversary without running the encryption protocol. So, the length of the resulting ciphertext is larger than the length of the input plaintext that why it is called LIRKES [4]. Although Blaze, Feigenbaum and Naor are considered the pioneers in introducing the LIRKES schemes, their proposal contains some security and complexity drawbacks that get it a little bit efficient solution for smart cards security problem.
In this research, we propose a secure and computationally efficient LIRKES. The proposed scheme withstands dictionary attack which can be easily applied to the BFN-LIRKES. In addition, it is suitable for cheap smart cards with a limited computational power.

Because of the requirement for a strong PRP in the proposed LIRKES, we introduce PATFC: Pseudorandom Affine Transformation Based Feistel Cipher as variable block-size symmetric-key block cipher. Block cipher is a PRP that maps a block of bits called plaintext into another block called ciphertext using the key bits. Pseudorandomness implies being not distinguishable form truly random permutation (TRP). In a well designed block cipher, a plaintext bit change should change each bit of the output ciphertext with a probability of 0.5. Also, there should be no plaintext/ciphertext-to-ciphertext correlations. Thus, secure block ciphers should essentially exhibit high degree of pseudorandomness, diffusion, and confusion [7]. In addition, a block cipher

is most practically qualified as secure if it has survived after being extensively exposed to proficient cryptanalysis. The structure of a block cipher may be a substitution-permutation network (SPN) or Feistel network (FN). The Advanced Encryption Standard AES-Rijndael is currently the most famous SPN cipher [8]. Alternatively, the FN structure, which is a universal method for converting a round function into a permutation, is adopted in several ciphers such as the DES, DESX, DEAL, FEAL, GOST, Khufu and Khafre, LOKI, CAST, and Blowfish [7], [8]. Rather than the use of many rounds, such as 16 in the DES, Luby and Rackoff introduced a 3-round FN construction used in designing a provably secure PRP from pseudorandom functions (PRF) [9]. Further analysis and several block ciphers are designed based on the Luby-Rackoff construction [5], [10]–[13].
By adopting the Luby-Rackoff construction, we propose PATFC which is a novel variable block-size symmetric-key block cipher. PATFC mainly makes use of a novel keyed PRF that is based upon a PR affine transformation (PRAT), constructed using a highly nonlinear Pseudorandom Number Generator (PRNG), and followed by a data and key dependent encoding and simple hashing scheme.
Extensive confusion, diffusion and pseudorandomness tests based upon the NIST statistical tests on PATFC and its underlying PRAT-based PRF consistently demonstrated their effectiveness. Furthermore, PATFC is not practically vulnerable to known, chosen and adaptive plaintext/ciphertext as well as dictionary and brute force attacks. It is also suitable for both software and hardware implementations.

Although PATFC is introduced to be used in the proposed LIRKES, it can be used to strengthen wireless mesh networks clients security by applying it as a candidate with a good pseudorandom and security properties in the well known WPA2 protocol used in IEEE 802.11i standard [14], [15]. In addition, we can exploit the whole scheme (PATFC and the LIRKES) to build a smart card based wireless mesh network to enhance its authentication and security in general [16].

The rest of the paper is organized as follows. Section 2 describes the Luby-Rackoff construction in more details, section 3 introduces PATFC and its experimental work, section 4 gives the suggested LIRKES with its cryptanalysis, section 5 shows how we can apply PATFC in the LIRKES, and section 6 gives the conclusions and future work.

2. PRELIMINARIES

Let "$\oplus$"denote the bit-wise XOR operation and $f_1, f_3 : \{0,1\}^r \to \{0,1\}^l$ and $f_2 : \{0,1\}^l \to \{0,1\}^r$ be a keyed PRFs. Given a *k*-bit key $K \in \{0,1\}^k$, a plaintext message $P = (L,R) \in \{0,1\}^{l+r}$ is divided into an *l* -bit (left) block L and *r*-bit (right) block R. Let $C = (U,T) \in \{0,1\}^{l+r}$ be its corresponding ciphertext. In case of *l*=*r* (balanced structure), Luby and Rackoff described how to construct a secure (against known / chosen plaintext attacks) PRP $\psi(f_1, f_2, f_3)(L,R) = (U,T)$ over $\{0,1\}^{l+r}$, from *r*-bit PRF's using a 3-round balanced Feistel network, rather than the use of 16 rounds as in the DES algorithm [9], with U and T computed as follows Fig.1:

$$\begin{aligned} S &= L \oplus f_1(K_1, R), \\ T &= R \oplus f_2(K_2, S), \\ \text{and } U &= S \oplus f_3(K_3, T) \end{aligned} \qquad (1)$$

where $S, U \in \{0,1\}^l$ and $T \in \{0,1\}^r$. Likewise, $\psi(f_3, f_2, f_1)$ yields the inverse PRP.
Note that because the entropy of the required permutation is (*l*+*r*)-bit, at least two rounds of PRFs are needed. But, using two rounds only, the attacker can distinguish the outputs from truly random permutation, if he simply chooses two different inputs with the same R. Luby and Rackoff even suggested the use of 4 rounds to prevent adaptive hosen plaintext-ciphertext attacks. Also unbalanced Luby-Rackoff construction $l \neq r$ is presented [10].

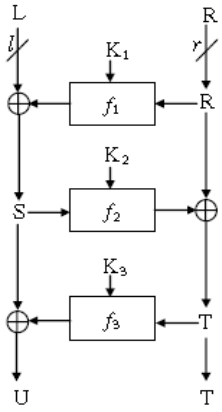

FIGURE 1: Luby-Rackoff cipher construction

3. The Proposed Cipher: PATFC

PATFC is a 3-round balanced (*l=r*) FN cipher, like Luby-Rackoff construction, based on a core PRAT based PRF. The following has motivated building PATFC using the proposed core PRF:

- In matrix-vector multiplication, it is evident that a change in an element of the input vector or major changes in the elements of the transformation matrix diffuse in all elements of the obtained output vector.
- Highly nonlinear PRNG's generate PR sequences that are very sensitive to the key. So, matrices successively constructed form such PR sequences dynamically pseudorandomly change their elements and significantly change with the key.

Thus, PR matrix-vector multiplication implies pseudorandomness, diffusion, and hence confusion [5] [17]. Pre-XORing the input vector with a PR vector (before matrix multiplication) yields the overall PRAT. Actually, the PRAT pseudorandomly substitutes the binary input vector with a vector of PR decimal numbers. Consequently, processing the obtained vector of PR decimal values to obtain the binary output, which incorporating proper additional nonlinearities to the PRF, complicates the cryptanalysis of the (underlying) PRF and the overall FN cipher constructed from it.

In implementing the PRAT, we use the RC4 PR bytes generator, which is a highly nonlinear generator with no public cryptanalytic results against it [7] [8]. The users of PATFC cipher could modify the algorithm to be suitable for any other PRNG such as the SEAL algorithm [7]. But we recommend RC4 because it makes use of a variable length key, and is PR (can likely be in 256×256^2 feasible states), fast, highly secure and invulnerable to linear/differential attacks [8].
Before presenting the PATFC internal construction, it is worth mentioning the balanced and homogeneous structure of PATFC. The FN is unbalanced or balanced if $r \neq l$ or $r = l$, respectively. Since the security of Luby-Rackoff ciphers depends on *min (l,r)*, the balanced structure provides more security [11][12]. Accordingly, to achieve optimal security, PATFC considers (without loss of generality) only the balanced structure, i.e., *l=r*. In addition, a FN is homogeneous if the same PRF is used in all rounds [18]. From the complexity (especially in hardware implementations) and computational burden points of view, it is recommended to reduce the number of different PRF's used in multi-round networks [11]. Therefore, to make the construction more efficient, PATFC uses the same PRF in the three rounds (but with different keys and consequently different PR sequences for the PRAT).

3.1 The Proposed PATFC Round Function *f*

The keyed PR round function f_{Ki} , $i \in \{1,2,3\}$, is the core of PATFC. f_{Ki} consists of PRAT followed by a dynamic (data and key dependent) *b*-bit encoding and simple hashing (Fig.2). Its inputs are an *r*-bit data, a 256-byte key K_i and a parameter representing the internally employed sub-block length $n \le min(r, N_{max})$, where $N_{max} \le r$ is a user specified number.

3.1.1 The PRAT construction

First, K_i is used to trigger an RC4 PRNG to successively generate *n* PR bits and n^2 PR bytes, i.e., $n+64n^2$ PR bits, to construct an $n \times 1$ -bit PR vector $\underline{\boldsymbol{v}}_{\boldsymbol{Ki}}$ and $n \times n$ byte PR matrix $\boldsymbol{G}_{\boldsymbol{Ki}}$.

The input *r*-bit is divided into sub-blocks each of size *n* bits. Then each *n*-bit sub-block of the input data is bitwise XORed with its associated RC4 generated PR bit-vector and then multiplied by its associated PR bytes matrix to obtain a vector of PR decimal values $\in \{0,1,...,255\boldsymbol{n}\}^n$. The actual maximum obtained value per sub-block dynamically changes depending on the patterns of the input data and the RC4 PR sequence.

To sum up, for the j^{th} *n*-bit input sub-block $\underline{\boldsymbol{x}}^{(j)}$, the PART can be represented as follows:

$$\underline{\boldsymbol{y}}^{(j)} = \boldsymbol{G}_{\boldsymbol{Ki}}^{(j)} (\underline{\boldsymbol{x}}^{(j)} \oplus \underline{\boldsymbol{v}}_{\boldsymbol{Ki}}^{(j)}) \tag{2}$$

Where $\underline{\boldsymbol{v}}_{\boldsymbol{Ki}}^{(j)}$ and $\boldsymbol{G}_{\boldsymbol{Ki}}^{(j)}$ are the j^{th} RC4 generated $n \times 1$ - bit PR vector and $n \times n$ byte PR matrix, respectively.

The bit wise XOR operation yields PR substitution whereas the multiplication by a PR matrix (which is actually equivalent to selective addition controlled by $\underline{\boldsymbol{x}}^{(j)} \oplus \underline{\boldsymbol{v}}_{\boldsymbol{Ki}}^{(j)}$) contributes to both diffusion and confusion.

Increasing the value of *n* results is achieving more randomness, diffusion, and confusion by the employed PRAT and the overall *f* (but with more computations).

3.1.2 Binary Encoding

We put each obtained decimal value from the PART into a binary format. The number of encoding bits used *b* dynamically changes from a sub-block to another, depending on its bit-pattern and the associated PR bit/byte patterns of the RC4 PR sequence, i.e., enforcing data and key dependency of the encoding process.

For the j^{th} sub-block, it is computed as

$$b^{(j)} = \max(1, \lceil \log_2 (y_{\max}^{(j)} + 1) \rceil) \tag{3}$$

Where $y_{\max}^{(j)}$ is the maximum decimal number in the j^{th} obtained vector. This step should yield a *br*-bit stream.

3.1.3 Simple Hash (Compression) Process

The *br*-bit stream R_1,obtained from the binary encoding step, is partitioned into *b r*-bit sections and compressed using *r b*-input XOR's working as a simple hash function to yield an *r*-bit output R' as follows:

$$\begin{aligned} R'(\xi)_{\xi=0\to r-1} &= \oplus_{j=0}^{b-1} R_1(\xi + jr) \\ &= R_1(\xi) \oplus R_1(\xi + r) \oplus .... \oplus R_1(\xi + (b-1)r) \end{aligned} \tag{4}$$

3.2 PATFC Key Scheduling Algorithm

The RC4 PRNG requires a 256-byte (2048-bit) key [7]. Hence, a total of 6144-bit key is required for the 3- round functions f_{K1}, f_{K2} and f_{K3}. So, PATFC can work in the 2048-bit key length mode in which the input key length is 2048-bit. Then, a simple key-scheduling algorithm, for example an RC4 based one, can be applied to generate the 3 round keys, K_1, K_2 and K_3, each of length 2048-bit. In other words, with K_1 only, the 3 PRF's may serially use a single RC4 PR sequence, while employing 3 distinct sections of the sequence. In addition, PATFC can work in the variable key length mode, in which the algorithm can accept any length key, and by the simple expansion

process suggested by the authors in [5], the key schedule algorithm can generate the 3-round keys used by PATFC.

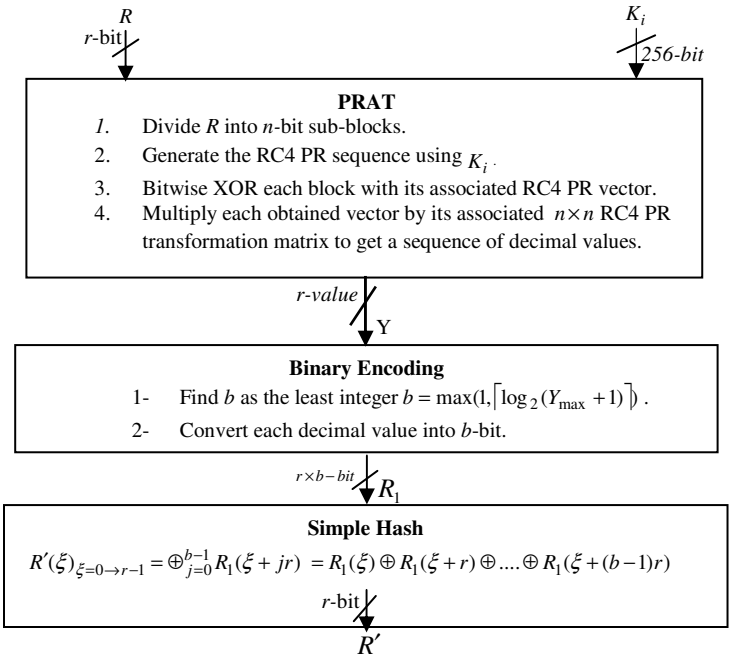

Figure 2: The proposed PATFC round function (*f*)

3.3 PATFC Cryptanalysis

In this section, we consider the performance of PATFC under several attacks types.

1- Possibility to attack the round function *f* :

In equation 2, $\underline{v}_{Ki}$ and G_{Ki} are successively generated using the PRNG, so they differ from a block to another. Choosing the PRNG to be highly nonlinear and secure leaves no helpful information to the attacker to predict its output sequence without knowing the seed value. i.e., the key used. Based on equation 2, the mean value of the elements of $\underline{y}$ is as follows:

$$E[y_i] = E\ [\boldsymbol{G}_K(\boldsymbol{i},:)\underline{\boldsymbol{v}}_K] = E\ [\sum_{j=1}^{n} G_K(i,j)v_K(j)] =$$

$$\sum_{j=1}^{n} E[G_K(i,j)v_K(j)] = \sum_{j=1}^{n} E[G_K(i,j)]E[v_K(j)] = \sum_{j=1}^{n} \frac{255}{2}\left(\frac{1}{2}\right) = \frac{255n}{4}, \qquad 1 \le i \le n\ . \tag{5}$$

Where *E* means the expected value.

It is appeared from equation 5 that the values of $\underline{y}$ almost widely spreads around $\frac{255n}{4}$. Therefore, it is a formidable task for the attacker to guess the values of $\underline{y}$ especially for large values of *n*, so every *r*-bit block output of the *f* function has almost the probability of $\frac{1}{2^r} + \varepsilon$, where ε depends upon the PRNG used.

2- Exhaustive key search attack (brut search attack):

In this attack, the attacker has many plaintext-ciphertext pairs encrypted under the same key and his job is to search all possible keys to find the key used in the encryption process. But, PATFC can accept a variable key ≤ 2048 -bit. So, practically, PATFC can effectively withstand the exhaustive key search attack.

3- Dictionary attack:
In this attack, the attacker makes a look up table (LUT) containing all possible plaintexts/ciphertexts pairs encrypted under all possible keys. Due to PATFC design as a variable block-size and key-length cipher, in case of dividing the plaintext into randomly sized blocks, the attacker neither knows the input plaintext length nor the key length. So he cannot practically make such a dictionary. Also if the whole plaintext is encrypted as a single block, the block size (and hence the codeword and number of entries of the LUT) is too large to practically try to construct the needed LUT.

4- Linear and Differential Cryptanalysis:
After the RC4 PRAT, the data/key-dependent encoding followed by the hashing scheme all together represents a highly PR nonlinear operation. So, even if some plaintext-ciphertext pairs (for the same key) are available to the attacker, the high PR nonlinearity of PATFC makes it invulnerable to linear and differential cryptanalysis. However, more analysis needs to be done to confirm our claim. On the other hand, since in linear and differential attacks [7], the attacker wants to know multiple distinct plaintexts-ciphertexts pairs for the same key, to know some of the key bits, we can encrypt the whole message at once using a different key each time or simply keep the employed PRNG in the PRAT step running and use its successive outputs for encoding the successive blocks.

5- Adaptive chosen plaintext/ciphertext attack:
The 3-round Luby-Rackoff ciphers may not prevent the adaptive chosen plaintext/ciphertext (two-sided) attack, which is the strongest attack against any symmetric key block cipher (despite being of little practical availability where the attacker can reach both the encryption and decryption engines). So, as suggested by Luby and Rackoff [9], a 4-round PATFC successfully prevents such type of attack.

Input plaintext 64-bit (In Hex)	User-key 64-bit (In Hex)	Output ciphertext64-bit (In Hex)
{0000-0000-0000-0000}	{0000-0000-0000-0000}	{78 EC-00C1-8915-8318}
{0000-0000-0000-0000}	{0000-0000-0000-0001}	{D67B-52F4-0F3F-E73E}
{0000-0000-0000-0001}	{0000-0000-0000-0000}	{3161-B32C-88BE-98D6}
{0000-0000-0000-0000}	{FFFF-FFFF-FFFF-FFFF}	{B3B9-3458-9307-D1E7}
{FFFF-FFFF-FFFF-FFFF}	{FFFF-FFFF-FFFF-FFFF}	{F2BA-89F5-6A60-4383}
{0000-0000-0000-0001}	{FFFF-FFFF-FFFF-FFFF}	{AE8F-0874-354D-F6B6}
{FFFF-FFFF-FFFF-FFFE}	{FFFF-FFFF-FFFF-FFFF}	{277F-0BDE-66E5-7926}
{FFFF-FFFF-FFFF-FFFF}	{FFFF-FFFF-FFFF-FFFE}	{AE04-F8DB-37F2-A7E5}
{FFFF-FFFF-FFFF-FFFF}	{0000-0000-0000-0000}	{3570-B0DA-3126-B6A3}

TABLE 1: Examples of 64-bit test vectors (in Hex) for PATFC

3.4 PATFC Experimental Work

We fully software implemented PATFC as a variable block-size variable key-length cipher with a simple effective key scheduling scheme. Table.1 presents examples of plaintext-key-ciphertext PATFC test vectors, especially including low and high density and correlated plaintext and key patterns, assuming 64-bit plaintext/key that shows PATFC excellent diffusion and confusion properties.

As in all Luby-Rackoff ciphers, security and pseudorandomness of the cipher is based upon the PR of the employed keyed round PRF f_K. The diffusion and confusion properties as well as pseudorandomness of the proposed PRF and the overall PATFC have been verified using extensive statistical diffusion and confusion as well as NIST tests [19].

Diffusion Test: 100 64-bit (32-bit for testing the round function) PR plaintexts P_i, i=1,2,.. ..,100 and 100 64-bit key K_i, i=1,2,.. .., 100, are generated using the SEAL algorithm. For each P_i, 64 1-perturbed-bit plaintexts $\{P_{i,j}, j=1,2,.. ..,64\}$, with the *jth* bit inverted, are generated. Then, the histogram, mean value and variance of the 6400 hamming distances $d_{i,j}=\sum(E_{Ki}(P_i) \oplus E_{Ki}(P_{i,j}))$ are computed, where $E_{Ki}(P_i)$ means the encryption of plaintext P_i using the K_i key.

Confusion Test: For the $P_{i,j}$*'s* mentioned above, the histogram, mean value and variance of the 6400 plaintext-ciphertext correlation coefficients $\rho_{i,j}= corr(P_{i,j},E_{Ki}(P_{i,j}))$ are computed. Also, for the P_i*'s* and $P_{i,j}$*'s* the histogram, mean value and variance of the 6400 ciphertext-ciphertext (of correlated plaintexts) correlation coefficients $\rho_{ij} = corr(E_{Ki}(Pi),E_{Ki}(P_{i,j}))$ are computed.

The results of the confusion and diffusion tests (summarized in Table.2 and Fig.3, 4 and 5) illustrate the competitive performance of PATFC compared with the DES and IDEA ciphers [7] as the correlations are almost zero and the percentage of the changing bits due to 1-bit perturbations is almost 50%.

NIST Pseudorandomness tests: The NIST Test Suite is a statistical package composed of 16 tests, basically developed to test the randomness of PRNG sequences. To use the NIST tests for testing the pseudorandomness (and implicitly the diffusion and confusion) of a block cipher, 7 data types are generated, following the procedure suggested in [20]. Of each data type, 100 4096-bit binary sequences were analyzed. These data types include: Plaintext-Avalanche, Key-Avalanche, Plaintext-Ciphertext Correlation, Low-Density Plaintext, Low-Density Key, High-Density Plaintext and High-Density Key data types.

The following 13 tests, with 32 *p*-values, of the 16 NIST tests were applied, namely the frequency (monobit), frequency within a Block (using a 128-bit block length), runs, longest run-of-1's in a block (using a 128-bit block length), binary matrix rank (with a 3×3 size), discrete Fourier transform, overlapping template matching (using a template of 9 1's, with a block length of 512-bit), Maurer's "universal statistical" (with 4-bit per block with 60 blocks for the initialization sequence), linear complexity (with a 20-bit block length), serial (with a 3-bit block length), approximate entropy (with a 2-bit block length), cumulative sums (Cusums), and random excursions variant tests.

Cipher Algorithm	Diffusion block length=64	Confusion tests block length=64	
		plain /cipher texts Corr.	Ciphertexts Corr.
	mean/64, var/64	Mean, var	Mean, var
PATFC	0.49, 0.24	2.546e-4, 9.82e-4	8.93e-5, 9.65e-4
DES	0.50, 0.24	-1.05e-5, 9.46e-4	-2.93e-4, 9.67e-4
IDEA	0.50, 0.25	-4.43e-4, 9.65e-4	-6.17e-4, 9.78e-4

TABLE 2: Comparison between the PATFC, DES, and IDEA.

Significance level of 0.01 indicates that one would expect 1 sequence out of 100 sequences to be rejected. A *p*-value ≥ 0.01 means that the sequence can be considered as random with a confidence of 99%. For each *p*-value, either success or failure evaluation was made based on being either above or below the pre-specified significance level of α=0.01 [19]. For each 100

sequences, two quantities were determined: the proportion of binary sequences passing the statistical test and an extra uniformity *p*-value based on a chi *χ2* test (with 9 degree of freedom) applied to the *p*-values of the 100 sequences. A sample (of 100 sequences) was considered to be passed a statistical test if its proportion of success exceeded

$$(1-\alpha)-\sqrt[3]{\frac{\alpha(1-\alpha)}{\boldsymbol{m}}}=0.99-\sqrt[3]{\frac{0.99\times 001}{100}}\approx 0.94 \quad (6)$$

i.e., 94%, and the uniformity test *p*-value exceeds 0.0001 [19]. The obtained results of the 32 *p*-values of the NIST tests successfully verified the pseudorandomness, diffusion and confusion properties of the proposed PRF and the overall PATFC with more than 94% proportion of succeeded sequences. Figures 6-8 illustrate samples of the obtained results, specifically the proportion of succeeded sequences for the 32 NIST tests applied to PATFC with Plaintext-Avalanche, Key-Avalanche, and Plaintext-Ciphertext Correlation generated data types.

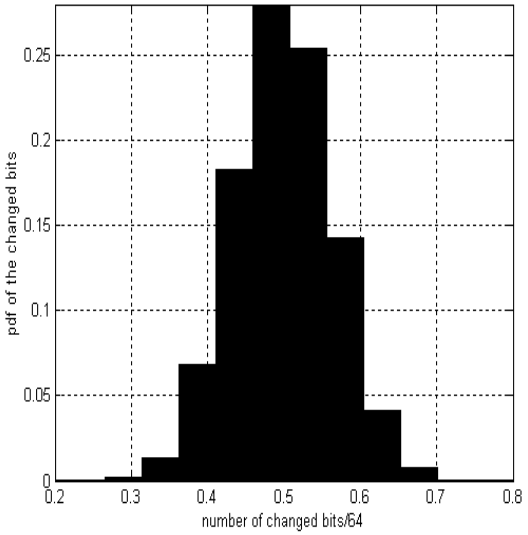

FIGURE 3: Diffusion test histogram: PATFC

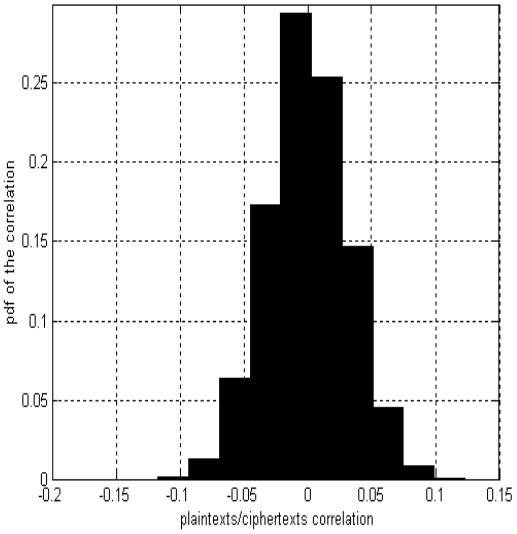

FIGURE 4: Confusion test: PATFC Plaintexts-Ciphertexts Correlations histogram

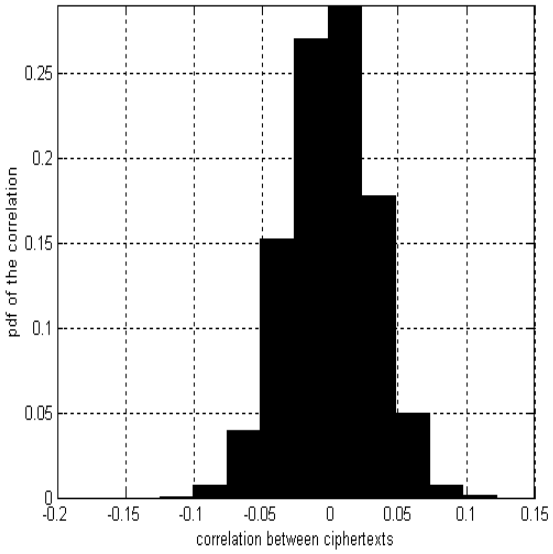

FIGURE 5: Confusion test: PATFC ciphertexts Correlations histogram

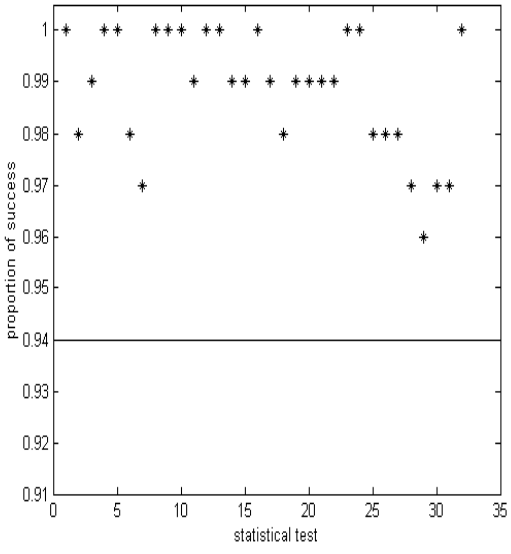

FIGURE 6: NIST tests using Plaintext-Avalanche data: Proportion of succeeded sequences for PATFC

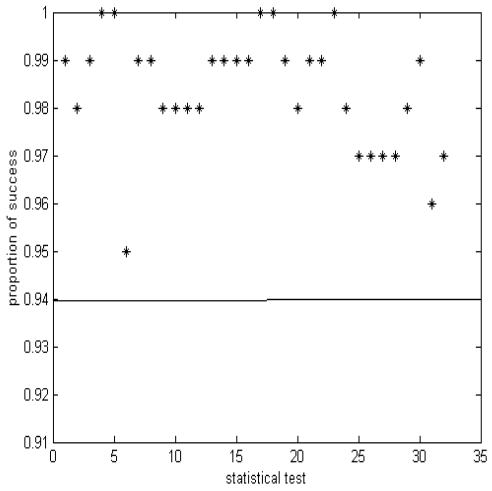

FIGURE 7: NIST tests using
Plaintext- Ciphertext correlation:
Proportion of succeeded sequences for PATFC

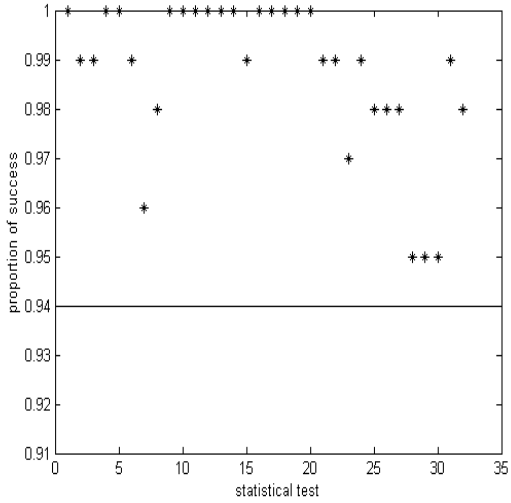

FIGURE 8: NIST tests using
key-Avalanche data:
Proportion of succeeded sequences for PATFC

4. A Novel LIRKES for Smart Cards

4.1 Overview of the BFN-LIRKES

As an attempt to solve the RKES problem, Blaze, Feigenbaum and Naor suggested a new trend different from the previous proposals [4]. Their trend is based upon the idea of self validation; Self validation means adding a signature ciphertext block to the original ciphertext, so, the resulting ciphertext length after adding the signature block is larger than the input plaintext length; as a result, their scheme is a length increasing (LI) RKES.

By using this idea, they suggested two schemes; one of them is insecure that any adversary can easily forge, and the other is a secure one that an adversary cannot forge. We will focus on the secure one.

The details of this scheme are as follows:

Secured BFN-LIRKES

Encryption protocol: input $P_1, P_2, ..., P_n$;output $t, C_0, C_1, C_2, ..., C_n$.

1. Generate session key S.
2. Host: $C_i = E_S^i(P_1, P_2, ..., P_n)$, $i \in \{1, ..., n\}$.
3. Host: $h = H(C_1, C_2, ..., C_n)$.
4. Host $\rightarrow$ Card: S, h.
5. Card: $C_0 \leftarrow E_{K_1}(S)$.
6. Card: $t \leftarrow F_{K_4}(F_{K_3}(C_0) \oplus F_{K_2}(h))$.
7. Card $\rightarrow$ Host: C_0, t .

Decryption protocol: input $t, C_0, C_1, C_2, ..., C_n$; output $P_1, P_2, ..., P_n$ or “invalid”.

1. Host: $h = H(C_1, C_2, ..., C_n)$.
2. Host $\rightarrow$ Card: C_0, h, t .
3. Card: if $t \neq F_{K_4}(F_{K_3}(C_0) \oplus F_{K_2}(h))$ Then $S \leftarrow$ “invalid”

Else $S \leftarrow D_{K_1}(C_0)$.

4. Card $\rightarrow$ Host: S.
5. Host: if S $\neq$ "invalid"
   Then $\{P_i = D_S^i(C_1, C_2, ..., C_n)$; output $(P_1, P_2, ..., P_n)\}$.
   Else output "invalid".

4.2 A Novel LIRKES

In this section, we will introduce a new LIRKES that overcomes the drawbacks of the BFN-LIRKES. Our scheme is also based upon the idea of self validation, but it is more secure and more efficient from card computations and key storages point of views than the BFN-LIRKES.

The proposed LIRKES

Encryption protocol: input $P_1, P_2, ..., P_n$; output $C_0, C_1, C_2, ..., C_n$.

1. Generate session key S by a best disposal.
2. Host: $C_i = E_S^i(P_1, P_2, ..., P_n)$, $i \in \{1, ..., n\}$.
3. Host: $h = H(C_1, C_2, ..., C_n)$.
4. Host $\rightarrow$ Card: S, h.
5. Card: $Z = H(K_1 \mid h)$. | Means concatenation
6. Card: $C_0 = E_Z(S)$.
7. Card $\rightarrow$ Host: C_0.

Decryption protocol: input $C_0, C_1, C_2, ..., C_n$; output $P_1, P_2, ..., P_n$.

1. Host: $h = H(C_1, C_2, ..., C_n)$.
2. Host $\rightarrow$ Card: C_0, h.
3. Card: $Z = H(K_1 \mid h)$.
4. Card: $S = D_Z(C_0)$.
5. Card $\rightarrow$ Host: S.
6. Host: $P_i = D_S^i(C_1, C_2, ..., C_n)$.

4.3 Security Analysis and Advantages of the proposed LIRKES

Theorem 1: The proposed LIRKES is forgery secure with probability $\frac{q(q-1)/2}{2^s} + \varepsilon$.

The proposed length increasing scheme is forgery secure in the sense that any probabilistic polynomial time adversary who access to the scheme during the HOST phase and makes *q* encryptions/decryptions with arbitrarily chosen inputs, can know no more than *q* valid plaintexts/ciphertexts pairs.

Proof: The collision resistance of *H* implies that $H(X_1) \neq H(X_2)$, for $X_1 \neq X_2$. So the chance for the adversary to find $(C_1, ..., C_n) \neq (C_1', ..., C_n')$ such that $H(C_1, ..., C_n) = H(C_1', ..., C_n')$ is negligibly small. Then the probability that $Z = Z'$, for $h \neq h'$ is also negligibly small. Also by assuming that the encryption function *E* is a strong invertible pseudorandom permutation. Then the probability that: $P_r(E_Z(S) = E_{Z'}(S')) = \frac{1}{2^s} + \varepsilon$, where $E(.) = \{0,1\}^z \times \{0,1\}^s \rightarrow \{0,1\}^s$, *z* and *s* are the lengths of Z and S respectively, and ε is a small number depends upon the pseudorandomness of *E*, and If *E* is truly random then ε=0. In addition, If the attacker makes *q* encryptions then there are *q(q-1)/2* different messages pairs then $P_r(E_Z(S) = E_{Z'}(S')) \leq \frac{q(q-1)/2}{2^s} + \varepsilon$.

Theorem 2: The proposed LIRKES is pseudorandom
Proof: From the above analysis, we can conclude that our proposed scheme is also pseudorandom.

Advantages of the proposed scheme over the BFN-LIRKES

1. The length of output ciphertext in the proposed scheme is shorter than the BFN-LIRKES. While the BFN-LIRKES uses two fields (t, C_0) to define the self validation (ciphetrext signature), we only use on field (C_0) to do that.
2. Our scheme is a self checking scheme; that is the checking step is inherited in the signature block C_0, i.e., if C_0 is incorrect the decryption protocol will output a random plaintext other than the correct one. consequently, there is no need for the checking steps used in the BFN-LIRKES which increases the complexity of the scheme.
3. We can attack BFN-LIRKES scheme by using a dictionary attack, i.e. if an adversary can access to the scheme many times, assuming the card uses the same K_1 every time, he can make a dictionary contains all values of S and its corresponding values of C_0. The size of this dictionary is 2^S, where *s* is the length of the session key S. So if the attacker successes in making such a dictionary, he can easily get the value of S for any C_0, so he can decrypt any message contains C_0. Therefore in BFN-LIRKES S must be very large. In contrast, in the proposed scheme the value of C_0 don't depend only on S but also upon *h*, for constant K_1, where *h* is the output length of the hash function, so the dictionary size will be $2^S \times 2^h$. As a result, the dictionary size is very large which gets such type of attacks computationally infeasible.
4. The proposed scheme is more efficient than the BFN-LIRKES from the card computation and key storage point of views. In BFN-LIRKES, the card uses four different keys but in our scheme we only use one key. In addition, the BFN-LIRKES requires from the card to evaluate four different functions, but in our scheme we require from the card to evaluate only two functions. In conclusion, the proposed scheme is suitable for cheap smart cards while BFN-LIRKES requires expensive ones.

5. The Application of PATFC in the Proposed LIRKES.

Figure 9 shows how we can apply PATFC as a strong PRP in the suggested LIRKES.

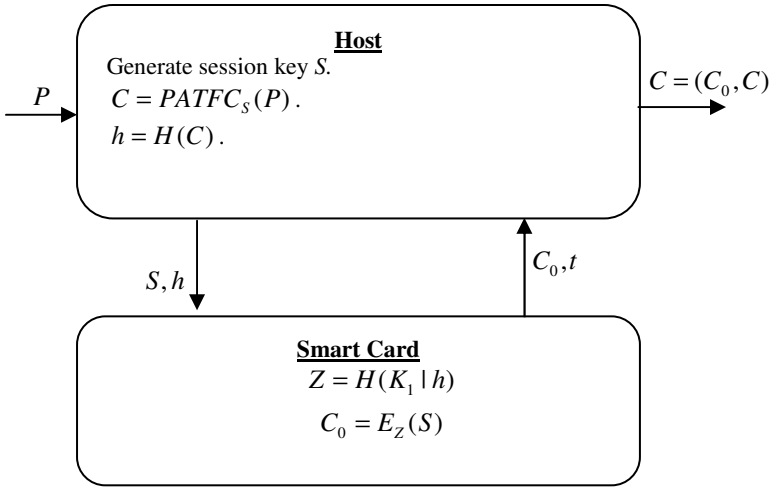

FIGURE 9: The proposed LIRKS using PATFC

6. CONSLUSION & FUTURE WORK

This paper deals with cryptographic smart cards protocols which are used to organize the bulk encryption process between the host and the smart card. In an attempt to solve this important issue, we introduce a new LIRKES that overcomes the drawbacks of the previous proposals. In addition we analyze this scheme from security and smart card efficiency point of views.
Because the suggested LIRKES is highly depending upon a strong PRP, we also present PATFC: Pseudorandom Affine Transformation based Feistel Cipher; a novel Luby-Rackoff construction-based variable block and key lengths symmetric-key block cipher. Its core function is a new pseudorandom function that consists of a pseudorandom affine transformation followed by binary encoding and a simple hashing scheme. Extensive simulations, diffusion, confusion, and NIST pseudorandomness tests proof that PATFC and its round function are good PRP and PR function respectively. However, PATFC needs a complexity analysis beside the security analysis, but we believe that PATFC is less complex.
Also, we show how PATFC can be applied as a PRP in the suggested LIRKES. For future development, we will try to apply our cipher and LIRKES in enhancing the security and authentication of the wireless mesh networks especially the wireless backhaul system.

7. REFERENCES

[1] S. Yuan and J. Liu, “Proceedings of the IEEE international conference on e-tech, e-commerce and e-services,” pp.91–110, 2004.

[2] M. Blaze, “High-bandwidth encryption with low-bandwidth smartcards,” Lecture Notes in Computer Science, vol.1039, pp.33–40, 1996.

[3] S. Lucks, “On the security of remotely keyed encryption,” Proceedings of the Fast Software Encryption Workshop, pp.219–229, Springer, 1997.

[4] M. Blaze, J. Feigenbaum, and M. Naor, “A formal treatment of remotely keyed encryption,” Lecture Notes in Computer Science, vol.1403, pp.251–265, 1998.

[5] E. M. Mohamed, Y. Hasan, H. Furukawa,” A Novel Luby-Rackoff Based Cipher in a New Feistel-Network Based LPRKES for Smart Cards”, International Journal of Computer Science and Security IJCSS, vol 3, pp 66- 81, 2009.

[6] Yasien M. Yasien, E. M. Mohamed “Two-Round Generalized FEISTEL Network Key-Linking Block Ciphers For Smart Card Applications”, Information Security Symposium (ISS), Al-Madinah Al-Munawwarah, Saudi Arabia, 2-4 May 2006.

[7] A. Menezes, P. Van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC press, 2001.

[8] A. Biryukov, “Block ciphers and stream ciphers: The state of the art,” Lecture Notes in Computer Science, Proc. COSIC Summer Course, 2003.

[9] M. Luby and C. Rackoff, “How to construct pseudorandom permutations from pseudorandom functions,” SIAM Journal on Computing, vol.17, no.2, pp.373–386, 1988.

[10] M. Naor, “On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited,” Journal of Cryptology, vol.12, no.1, pp.29–66, 1999.

[11] R. Anderson and E. Biham, “Two practical and provably secure block ciphers: BEAR and LION,” Lecture Notes in Computer Science, pp.113–120, 1996.

[12] P. Morin, “A critique of BEAR and LION,” Manuscript, citeseer. nj. nec. Com/124166. html.

[13] Y. Hasan, “YC: A Luby-Rackoff ciphers family driven by pseudorandom vector/matrix transformations,” Signal Processing and Its Applications, 2007. ISSPA 2007. 9th International Symposium on, pp.1–4, 2007.

[14] S. Frankel, B. Eydt, L. Owens, and K. Kent, “Guide to ieee 802.11 i: Establishing robust security networks,” Technical Report 800-97, National Institute of Standards and Technology Administration US Department of Commerce, Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8930, 2006.

[15] F. Martignon, S. Paris, and A. Capone, “MobiSEC: a novel security architecture for wireless mesh networks,” Proceedings of the 4th ACM symposium on QoS and security for wireless and mobile networks, pp.35–42, ACM New York, NY, USA, 2008.
[16] M. Siddiqui and C. Hong, “Security issues in wireless mesh networks,” IEEE intl. conf. on multimedia and ubiquitous engineering, 2007.
[17] Y. Hasan, “From stream to provably secure block ciphers based on pseudorandom matrix transformations,” Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on, pp.260–265, 2008.
[18] U. Maurer, “A simplified and generalized treatment of Luby- Rackoff pseudorandom permutation generators”, Proceedings Advances in Cryptology- EUROCRYPT 92, LNCS, vol.658, pp.239-255, Springer-Verlag, 1992.
[19] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” , 2001.
[20] J. Soto and L. Bassham, “Randomness Testing of the Advanced Encryption Standard Finalist Candidates. National Institute of Standards and Technology (NIST),” Computer Security Division, 2000.