XLS is not a Strong Pseudorandom Permutation

Mridul Nandi

Indian Statistical Institute, Kolkata
mridul@isical.ac.in,

Abstract. In FSE 2007, Ristenpart and Rogaway had described a generic method XLS to construct a length-preserving strong pseudorandom permutation (SPRP) over bit-strings of size at least n. It requires a length-preserving permutation $\mathcal{E}$ over all bits of size multiple of n and a blockcipher E with block size n. The SPRP security of XLS was proved from the SPRP assumptions of both $\mathcal{E}$ and E. In this paper we disprove the claim by demonstrating a SPRP distinguisher of XLS which makes only three queries and has distinguishing advantage about $1/2$. XLS uses a multi-permutation linear function, called mix2. In this paper, we also show that if we replace mix2 by any invertible linear functions, the construction XLS still remains insecure. Thus the mode has inherit weakness.

Keywords: XLS, SPRP, Distinguishing Advantage, length-preserving encryption.

1 Introduction

The notion of domain extension arises in many areas of cryptography such as hash function, pseudorandom function or PRF, strong pseudorandom permutation or SPRP [12] etc. Usually, we design a building block defined for a small and fixed bit size domain. Then, by applying the building block iteratively, we obtain a similar kind of function defined over arbitrary domain. For example, a blockcipher defined on n bits can be used to define an encryption algorithm which can encrypt any message of size multiple of n. To define a ciphertext for a message whose size is not a multiple of n, one can first apply some padding rule to make the (padded) message of size multiple of n. This method can not preserve length as it expands ciphertext length. A length-preserving encryption is called an **enciphering scheme**. The length-preserving property makes our task more difficult and restricted than length expanding encryptions. On the other hand, designing enciphering schemes over all bit strings of size multiple of block-size (i.e., n) seems to be easier than defining over arbitrary bit strings. Many such enciphering schemes have been defined [10, 9].

Non-Generic Methods. There are several known methods for turning a blockcipher into an enciphering schemes over arbitrary bit strings. One can apply the underlying block cipher twice and use the intermediate output as an one-time pad for partial block (used in EME [7], TET [8], HEH [16] etc.); The other constructions e.g., HCTR [17], HCH [3], XCB [13] use hash-then counter

paradigm. A standard trick like ciphertext stealing can also be applied to specific constructions (e.g., AEZ [1]). However, all these approaches are not generic.

> We call a method **domain completion** (or generic domain completion) if it converts any enciphering scheme over bit strings of size multiple of n into an enciphering scheme over any bit strings (possibly of size at least n).

To our best knowledge, so far only two domain completions have been proposed.

1. A popular, efficient and neatly defined domain completion method is XLS (eXtended by Latin Square) designed by Ristenpart and Rogaway [15]. The design rational of XLS is similar to that of elastic blockcipher as both follow encrypt-then-mix paradigm.
2. Following hash-counter-hash paradigm, Nandi proposed a domain completion method in [14].

In addition to these, a heuristically described method, called **Elastic blockcipher** [4], was proposed by Cook, Yung and Keromytis. Later elastic blockcipher, defined over all bits of sizes in between n and $2n$, has been more formally defined in [5].

Applications of Domain Completion Method. While primarily interested in the theoretical question of how to obtain domain extension for ciphers, arbitrary-input-length enciphering is a problem with many applications. A well-known application is disk-sector encryption in which size of ciphertext and plaintext remain same as the sector size of the disk. In general, enciphering scheme is easy to define for input sizes of multiple of n (block size of the underlying blockcipher). Domain completion methods can be used to define the enciphering schemes for arbitrary bit strings. It is also used in other symmetric key algorithms such as authenticated encryption. For example, XLS is widely adapted in many authenticated encryptions, e.g. AES-COPA [2], Deoxys, Joltik, KIASU, Marble, SHELL etc. [1].

1.1 Our Contribution

In this paper we demonstrate a chosen plaintext-ciphertext adversary (CPCA) distinguisher against XLS (see Algorithm $\mathcal{A}_0$ in section 3.2). The attack makes only three encryption and decryption oracle queries in an interleaved manner and has distinguishing advantage about $1/2$. Thus, the security claim of XLS is wrong.

XLS uses a linear multi-permutation (very efficiently computable) mix2 which satisfies some property. Authors called any such linear permutation satisfying the property a *good mixing function*. It is natural to think a possible remedy of XLS to replace mix2 by other good mixing function or some other stronger linear permutations. Unfortunately, we show that these remedies would not work. To establish our claim, we consider a generalized version of XLS (we call it GXLS)

which applies any arbitrary linear permutations instead of mix2. Moreover, we consider keys of two invocations of the underlying blockcipher E to be independently chosen. We demonstrate similar CPCA-distinguishers (in section 4) for GXLS having advantage at least $1/4$. So we conclude that XLS has design flaws in its modes not in the choice of linear mixing functions.

2 XLS and Its General Form GXLS

Basics and Notation.

1. An s-bit string X is denoted as $X = X[1]X[2]\cdots X[s]$ where $X[i] \in \{0,1\}$. We denote $X[i..j] = X[i]\cdots X[j]$ and $|X| = s$.
2. A length-preserving function f satisfies $|f(X)| = |X|$ for all X.
3. Any linear function from $\{0,1\}^s$ to $\{0,1\}^t$ can be represented by a $t \times s$ binary matrix. Let $\mathsf{rol}(X)$ represent left circular bit-rotation, that is, for any bit-string $X := X[1]X[2]\cdots X[s]$ of length s, let $\mathsf{rol}(X) = X[2]X[3]\cdots X[s]X[1]$. Note that rol is a linear invertible function and is represented by the following $s \times s$ invertible matrix:
$$\mathbf{R} = \begin{pmatrix} 0\,1\,0\cdots 0\,0 \\ 0\,0\,1\cdots 0\,0 \\ \vdots \\ 0\,0\,0\cdots 0\,1 \\ 1\,0\,0\cdots 0\,0 \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{I}_{s-1} \\ 1 & 0 \end{pmatrix}.$$
Here, $\mathbf{I}_{s-1}$ represents the identity matrix of size $s-1$.
4. Throughout the paper, let n be a fixed integer representing the block-size of the underlying blockcipher E.

2.1 XLS and GXLS on $\{0,1\}^{2n-1}$

In this section we describe how XLS has been defined for bit strings of size $2n-1$. Later we show distinguishing attack of XLS by making queries of size $2n-1$ only. We refer readers to the original paper [15] for the definition of XLS over arbitrary bit strings. We first define a linear function $\mathsf{mix2} : \{0,1\}^{2n-2} \to \{0,1\}^{2n-2}$ as below

$$\mathsf{mix2}(AB) = (A \oplus \mathsf{rol}(A \oplus B), B \oplus \mathsf{rol}(A \oplus B)) = ((\mathbf{R}+\mathbf{I})\cdot A + \mathbf{R}\cdot B,\ \ \mathbf{R}\cdot A + (\mathbf{R}+\mathbf{I})\cdot B)$$
$$= \begin{pmatrix} \mathbf{R}+\mathbf{I} & \mathbf{R} \\ \mathbf{R} & \mathbf{R}+\mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} A \\ B \end{pmatrix}$$

where $|A| = |B| = n-1$ and $\mathbf{I}$ is the identity matrix of size $n-1$. It is easy to see that the inverse of the linear map mix2 is itself. Such a permutation is also called an involution. Now we describe the XLS algorithm [15] over the set of all $2n-1$ bit strings based on two n-bit (random) permutations E and $\mathcal{E}$ and the linear permutation mix2. We would like to note that we express the input, output and intermediate variables with different notations from those of [15] which would be used to describe our attack and analysis.

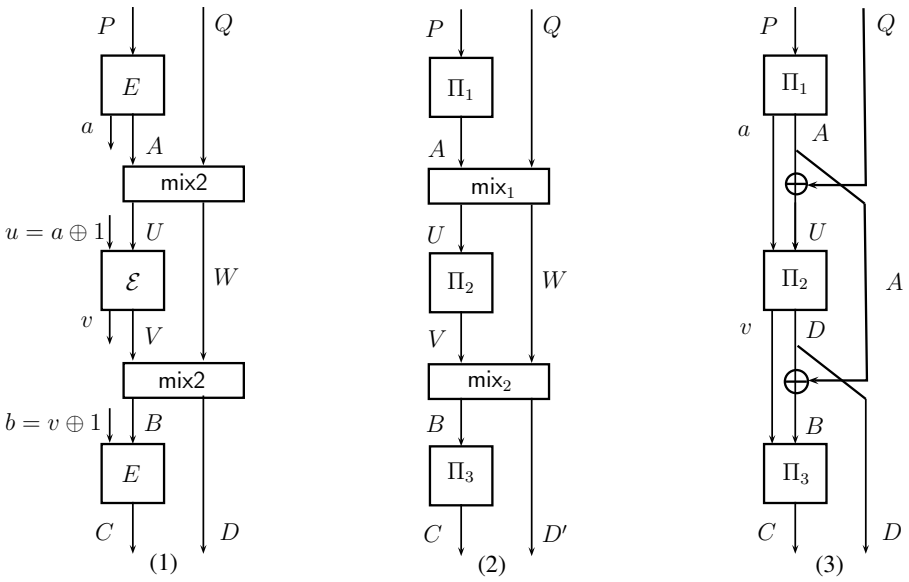

Fig. 2.1. Illustration of (1) XLS, (2) GXLS and (3) 3-round Elastic Blockcipher. The XLS and 3-round Elastic blockcipher are special cases of GXLS.

Algorithmic Definitions of XLS and GXLS Now we describe the algorithms XLS and GXLS which are defined on $2n-1$ bits.

Algorithm $\mathsf{XLS}^{E,\mathcal{E}}$
Input: $(P,Q) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n-1}$
Output: $(C,D) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n-1}$.
01 $E(P) = a \| A,\ \ a \in \mathbb{F}_2$.
02 $u = a!,\ \ (U,W) = \mathsf{mix2}(A,Q)$.
03 $\mathcal{E}(u\|U) = v\|V$.
04 $b = v!,\ \ (B,D) = \mathsf{mix2}(V,W)$.
05 $E(b\|B) = C$.
06 return (C,D).

Algorithm $\mathsf{GXLS}[\Pi_1, \mathsf{mix}_1, \Pi_2, \mathsf{mix}_2, \Pi_3]$
Input: $(P,Q) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n-1}$
Output: $(C,D) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n-1}$.
01 $\Pi_1(P) = A$.
02 $(U,W) = \mathsf{mix}_1(A,Q)$.
03 $\Pi_2(U) = V$.
04 $(B,D) = \mathsf{mix}_2(V,W)$.
05 $\Pi_3(B) = C$.
06 return (C,D).

Here ! denotes bit complement. Here mix_1 and mix_2 are two invertible linear functions on $2n-1$ bits and mix2 is a linear invertible function over $\{0,1\}^{2n-2}$ bits as described before. The Π_i's are independent uniform random (or pseudorandom) permutations whereas in XLS $\mathcal{E}$ and E are independent uniform random (or pseudorandom) permutations. We also denote the generalized-XLS as $\mathsf{GXLS}[\Pi_1, \mathsf{mix}_1, \Pi_2, \mathsf{mix}_2, \Pi_3](P,Q) = (C,D)$ as above (in the right hand side of Fig. 2.1). Note that the XLS algorithm is nothing but $\mathsf{GXLS}[E, !\|\mathsf{mix2}, \mathcal{E}, !\|\mathsf{mix2}, E]$ where $(!\|f)(b,X) = b!\|f(X)$. In order for GXLS to be invertible, mix_1 and mix_2 should be invertible.

2.2 Elastic Blockcipher

The three round Elastic blockcipher can also be viewed as a $\mathsf{GXLS}[\Pi_1, \mathsf{mix3}, \Pi_2, \mathsf{mix3}, \Pi_3]$ where $\mathsf{mix3}(A, B) = ((A[i_1] \cdots A[i_s]) \oplus B, A[i_1] \cdots A[i_s])$, $|A| = n$, $|B| = s$ and $1 \leq i_1 < \cdots < i_s \leq n$ are some fixed integers (specific choices of these values depend on the underlying blockcipher). We illustrate this method in Fig 2.1 when $i_1 = n-s+1, \ldots, i_s = n$. Basic mix function of it can be defined as $(X \| Y) \mapsto X \oplus Y \| X$ where $|X| = |Y| = s$. Similarly, four or higher number of rounds can be defined. So all of these follow the encrypt-mix paradigm iterated several rounds. We capture this paradigm for three iterations in GXLS. In the following sections, we prove that three rounds are not sufficient for having SPRP.

3 Insecurity of XLS

In this section we show that XLS is not SPRP (strong pseudorandom permutation). In fact we establish a distinguisher making only three oracle queries having distinguishing advantage about $1/2$. Moreover, if we repeat this attack independently, we can amplify the distinguishing advantage close to one. We first briefly define basics of security notions related to distinguishing advantages.

3.1 Security Definitions

Let $\mathtt{R}_i$ denote the uniform random function from $\{0,1\}^i$ to $\{0,1\}^i$, i.e., the uniform distribution on the set $\mathrm{Func}(\{0,1\}^i, \{0,1\}^i)$ of all functions from $\{0,1\}^i$ to itself. Given a set $L \subseteq \mathbb{N} := \{1, 2, \cdots\}$, we denote $\mathtt{R}_L$ for the tuple $\langle \mathtt{R}_i \rangle_{i \in L}$ of random functions where $\mathtt{R}_i$'s are jointly independently distributed. We call the set L length set. We call $\mathtt{R}_L$ a *length-preserving uniform random function* on $\{0,1\}^L := \cup_{i \in L} \{0,1\}^i$. Similarly, let $\mathtt{P}_i$ denote the uniform random permutation on $\{0,1\}^i$, i.e., the uniform distribution on the set $\mathrm{Perm}(\{0,1\}^i, \{0,1\}^i)$ of all permutations on $\{0,1\}^i$. Note that the inverse random permutation, $\mathtt{P}_i^{-1}$, is also an uniform random permutation. We similarly define length-preserving uniform random permutation $\mathtt{P}_L$ on $\{0,1\}^L$ which is independent composition of $\mathtt{P}_i$ for all $i \in L$.

Now let $\mathcal{A}$ be an oracle algorithm which has access of two length-preserving oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. Suppose $\mathcal{A}$ makes queries from the set $\{0,1\}^L$ for both oracles. We define **SPRP-advantage** of $\mathcal{A}$ for a length-preserving random permutation $\mathtt{F}_L$ (not necessarily uniform) by

$$\mathbf{Adv}^{\mathrm{sprp}}_{\mathtt{F}_L}(\mathcal{A}) = \mathsf{Pr}[\mathcal{A}^{\mathtt{F}_L, \mathtt{F}_L^{-1}} = 1] - \mathsf{Pr}[\mathcal{A}^{\mathtt{P}_L, \mathtt{P}_L^{-1}} = 1].$$

In general, we can define advantage for two pairs of tuples of length-preserving random functions $(\mathtt{F}_L, \mathtt{F}'_L)$ and $(\mathtt{G}_L, \mathtt{G}'_L)$ as

$$\mathbf{Adv}_{\mathcal{A}}((\mathtt{F}_L, \mathtt{F}'_L), (\mathtt{G}_L, \mathtt{G}'_L)) = \mathsf{Pr}[\mathcal{A}^{\mathtt{F}_L, \mathtt{F}'_L} = 1] - \mathsf{Pr}[\mathcal{A}^{\mathtt{G}_L, \mathtt{G}'_L} = 1].$$

If $\mathcal{A}$ interacts with a length-preserving random permutation $\mathcal{O}_1$ and its inverse $\mathcal{O}_2$ then we can assume the following:

1. $\mathcal{A}$ is not making any repetition query.
2. If x_i is $\mathcal{O}_1$-query and y_i is its response then there is no $\mathcal{O}_2$-query $x_j = y_i$ for some $j > i$ and vice-versa.

We can assume these since the responses are determined for these types of queries. An adversary satisfying the above conditions is called an *allowed adversary.*

Theorem 1. *[11] Let $\mathtt{R}_L$ and $\mathtt{R}'_L$ be independently chosen length-preserving uniform random functions and let $\mathtt{P}_L$ be length-preserving uniform random permutation. Then for any allowed adversary $\mathcal{A}$ which makes at most Q queries, we have,*

$$\mathbf{Adv}_{\mathcal{A}}((\mathtt{P}_L, {\mathtt{P}_L}^{-1}), (\mathtt{R}_L, \mathtt{R}'_L)) \leq \frac{Q(Q-1)}{2^{m+1}}$$

where $m = \min\{\ell : \ell \in L\}$.

The above result says that an uniform length-preserving random permutation is very close to an uniform length-preserving random function. Thus if we want to prove that an enciphering scheme is not SPRP-secure by small number of queries then it would be enough to compute the distinguishing advantage from uniform random functions for an allowed adversary. For example, when $Q = 3$, if for length-preserving construction $\mathtt{F}_L$, $\mathbf{Adv}_{\mathcal{A}}((\mathtt{P}_L, {\mathtt{P}_L}^{-1}), (\mathtt{F}_L, \mathtt{F}_L^{-1})) := c$ is significant for an allowed adversary then $\mathbf{Adv}^{\mathrm{sprp}}_{\mathtt{F}_L}(\mathcal{A})$ is at least $c - 2^{-n+2}$ which is also significant.

Remark 1. The above is one side of the implication of the Theorem 1. The other application is to show a construction $\mathtt{F}_L$ SPRP by showing $\mathbf{Adv}_{\mathcal{A}}((\mathtt{F}_L, {\mathtt{F}_L}^{-1}), (\mathtt{R}_L, \mathtt{R}'_L))$ is negligible.

3.2 SPRP Distinguishing Algorithm

Distinguishing Algorithm $\mathcal{A}_0$ for XLS.

1. Make encryption query (P_1, Q_1) and obtains response (C_1, D_1).
2. Make decryption query $(C_2 := C_1, D_2 := D_1 + 1)$ and obtains response (P_2, Q_2).
3. Make encryption query $(P_3 = P_2, Q_3)$ and obtains response (C_3, D_3) where
$$Q_3 = Q_1 + (\mathbf{I} + \mathbf{R}^{-2}) \cdot (Q_1 + Q_2 + 1).$$
4. **return** 1 if $D_3 = Q_1 + Q_3 + D_1$, 0 otherwise.

3.3 Analysis of Attack

To see why our attack works, let us first observe some useful relations among internal variables in the computations of XLS.

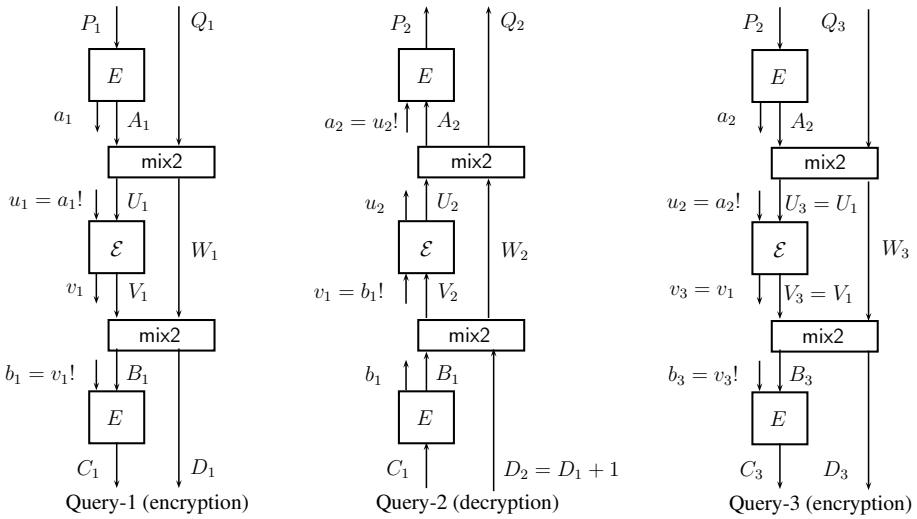

Fig. 3.1. $\mathcal{A}_0$ makes three queries and obtains collisions on U_1 and U_3 values with probability 1/2 (due to the event that $a_1 = a_2$).

Lemma 1. *With the notations as described in the algorithm* XLS*, we have* $A + B = (\mathbf{R}^{-1} + \mathbf{I}) \cdot (Q + D)$.

Proof. Since mix2 is inverse of itself we have $(V, W) = \mathsf{mix2}(B, D)$. By equating W with line 02 of XLS algorithm, we have

$$\mathbf{R} \cdot B + (\mathbf{R} + \mathbf{I}) \cdot D = \mathbf{R} \cdot A + (\mathbf{R} + \mathbf{I}) \cdot Q.$$

Thus, $\mathbf{R} \cdot (A + B) = (\mathbf{R} + \mathbf{I}) \cdot (Q + D)$ and so the result follows. □

Lemma 2. *With the notations as described in the algorithm* XLS*, we have* $U + V = \mathbf{R}^{-1} \cdot (Q + D)$.

Proof. Due to line 02 and 04, we have $\mathbf{R} \cdot U + (\mathbf{I} + \mathbf{R}) \cdot W = Q$ and $\mathbf{R} \cdot V + (\mathbf{I} + \mathbf{R}) \cdot W = D$. Thus, $\mathbf{R} \cdot (U + V) = (D + Q)$ and so the result follows. □

The basic idea of our attack is to obtain an internal collision. Suppose we have two queries (P_i, Q_i) with responses (C_i, D_i), $i = 1, 2$ such that the U_i values remain the same. So the outputs V_i are also same. Due to above lemma, we have $Q_1 \oplus D_1 = Q_2 \oplus D_2$. For a uniform random permutations this event can occur with a probability of about 2^{-n+1}. Now we show that in query-1 and query-3, U values collide with probability 1/2 and so we can distinguish XLS from uniform random permutation with advantage about 1/2 (for large n, 2^{-n+1} is negligible).

Theorem 2. *The Algorithm of* $\mathcal{A}_0$ *has distinguishing advantage about* 1/2 *against* XLS*.*

Proof. Note that $\mathcal{A}_0$ makes three encryption and decryption queries in an interleaved manner. Let us denote the intermediate variables of computations of i^{th} query by using suffix i, $1 \leq i \leq 3$. Let us denote $\mathbf{G} = \mathbf{R}^{-1} + \mathbf{I}$. By Lemma 1, we have $A_1 + B_1 = \mathbf{G} \cdot (Q_1 + D_1)$ and $A_2 + B_1 = A_2 + B_2 = \mathbf{G} \cdot (Q_2 + D_2)$ and so $A_1 + A_3 = A_1 + A_2 = \mathbf{G} \cdot (Q_1 + D_1 + Q_2 + D_2) = \mathbf{G} \cdot (Q_1 + Q_2 + 1)$. Now we make our main claim:

Claim: $U_1 = U_3$.

$$\begin{aligned} U_1 + U_3 &= (\mathbf{R} + \mathbf{I}) \cdot (A_1 + A_3) + \mathbf{R} \cdot (Q_1 + Q_3) \\ &= (\mathbf{R} + \mathbf{I}) \cdot \mathbf{G}(Q_1 + Q_2 + 1) + \mathbf{R} \cdot (Q_1 + Q_3) \\ &= (\mathbf{R} + \mathbf{R}^{-1}) \cdot (Q_1 + Q_2 + 1) + \mathbf{R} \cdot (Q_1 + Q_3) \end{aligned}$$

Since $Q_3 = Q_1 + (\mathbf{I} + \mathbf{R}^{-2}) \cdot (Q_1 + Q_2 + 1)$, we have $U_1 = U_3$. □

The rest of the proof is straightforward. As we have collision on U values, we have collision on V values, i.e., $V_1 = V_3$. But this can happen if the first bit of inputs of $\mathcal{E}$ in query 1 and 3 match which can happen with probability $1/2$. Assuming this, we can exploit the collision to make distinguishing attack as discussed before the theorem. We have $D_3 = Q_1 + Q_3 + D_1$. This can hold for a uniform random permutation $\mathcal{E}$ with probability about 2^{-n+1}. So the result follows. □

Remark 2. The same attack works for any length of the form $kn - 1$ with same advantage. We only need the size of the partial block to be $n - 1$. Note that we need the first bit of output of E in query 1 and 3 should match which can happen with probability $1/2$. For other length inputs, the distinguishing advantage reduces as we need more bits collision. In general, if we want to distinguish XLS only on $kn + s$ bits inputs then we need collision on the first $n - s$ bits of outputs of E in query 1 and 3 which can happen with probability about 2^{s-n}. So the distinguishing advantage would be about $2^{s-n} - 2^{1-n}$. So if the partial block size s is small the distinguishing advantage of our attack reduces. This is very natural as most of the intermediate bits are processed through $\mathcal{E}$.

4 Distinguishing attack on GXLS on $\{0,1\}^{2n-1}$

Now we demonstrate how we can modify the distinguishing attack for GXLS. This would suggest that any simple modification on XLS (such as changing mix functions with others) do not work. In other words, we show that the mode, not the mixing function, has inherent weakness. Behavior of this distinguishing attack depends on different cases of invertible mixing matrices mix_1 and mix_2. As we need to assume these as linear permutations, we can represent these by the following $(2n-1) \times (2n-1)$ invertible matrices.

$$\mathsf{mix}_1 = \begin{pmatrix} M[1]_{n \times n} & N[1]_{n \times (n-1)} \\ M[2]_{(n-1) \times n} & N[2]_{(n-1) \times (n-1)} \end{pmatrix},$$

$$\mathsf{mix}_2^{-1} = \begin{pmatrix} M'[1]_{n\times n} & N'[1]_{n\times(n-1)} \\ M'[2]_{(n-1)\times n} & N'[2]_{(n-1)\times(n-1)} \end{pmatrix}.$$

Before we demonstrate our attacks we state some notations and results which would be used.

Notations. Given a $r \times s$ matrix $\mathbf{A}$ we denote $\mathcal{C}(\mathbf{A})$ the column space of the matrix.

Lemma 3. *Let $M_{r\times s}$ and $N_{r\times t}$ be two matrices and $c_{r\times 1}$ is a vector such that $\mathcal{C}(N) \not\subseteq \mathcal{C}(M)$. For any two uniform random vectors $a \xleftarrow{\$} \{0,1\}^s$ and $q \xleftarrow{\$} \{0,1\}^t$ (not necessarily independent) $\mathsf{Pr}[M \cdot a = N \cdot q + c] \leq 1/2$.*

Proof. This is straightforward when M is of the form $\begin{pmatrix} \mathbf{I}_r & * \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$ where r is the rank of M and $*$ means that the sub matrix could be anything. In this case there must exist $i > r$ such that i^{th} row of N is non-zero. As q_i is uniform on $\{0,1\}$, by equating the event on i^{th} bit we get probability at most $1/2$ to achieve the event.

For a general matrix M, we can find two non-singular square matrices S and T such that $S \cdot M \cdot T = \begin{pmatrix} \mathbf{I}_r & * \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$. So the given probability p should be same as

$$\mathsf{Pr}[SMT \cdot (T^{-1}a) = SN \cdot q + S \cdot c].$$

Let us denote $M' = SMT$, $a' = T^{-1}a$, $c' = Sc$ and $N' = SN$. With this notation, we have $p = \mathsf{Pr}[M' \cdot a' = N' \cdot q + c']$. Now note that M' has the form as considered before. Due to invertible property of S and T, we have the property that $\mathcal{C}(N') \not\subseteq \mathcal{C}(M')$ and, a' and q' follow individually uniform distributions. □

Now we describe our attacks for different cases of the sub matrices of the mix functions. Conventionally, we use suffix 1, 2 and 3 to denote intermediate values for the first, second and third queries respectively.

4.1 $\mathbf{rank}(M[2]) \leq n - 2$

In this case we first claim that the column space of $N[2]$ must contain a vector which does not belong to the column space of $M[2]$. Otherwise, the rank of $(n-1) \times (2n-1)$ matrix $(M[2]\ N[2])$ is less than $n-2$ which contradicts that the matrix mix_1 is invertible.

Now we run the algorithm $\mathcal{A}_0$ only for the first two queries. As before, we have $M[2](A_1 + A_2) = N[2](Q_1 + Q_2) + N'[2] \cdot 1$. Using the lemma 3, we know that when the algorithm is interacting with uniform random permutation, the probability that $N[2](Q_1+Q_2)+N'[2]\cdot 1$ belongs to the column space of $M[2]$ is less than $1/2$. However, for the case of GXLS it occurs with probability one. So we can distinguish with advantage at least $1/2$. We formally describe the attack algorithm by $\mathcal{A}_1$ below.

Distinguishing Algorithm $\mathcal{A}_1$.

1. Make encryption query (P_1, Q_1) and obtain response (C_1, D_1).
2. Make decryption query $(C_2 := C_1, D_2 := D_1+1)$ and obtain response (P_2, Q_2).
3. **return** 1 if $N[2](Q_1 + Q_2) + N'[2] \cdot 1 \in \mathcal{C}(M[2])$, 0 otherwise.

Note that given a vector v and a matrix $M[2]$, there is an efficient algorithm to check whether a vector v belongs to the column space of $M[2]$. For this we essentially need to solve the system of equations $M[2] \cdot x = v$ and whenever we arrive contradiction a solution does not exist equivalently v is not a member of the column space. Alternatively we can first find some invertible matrices S and T (by some standard elementary operations) such that

$$S \cdot M[2] \cdot T = \begin{pmatrix} \mathbf{I}_r & * \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

where r denotes the rank of $M[2]$. So $M[2] \cdot x = v$ if and only if

$$\begin{pmatrix} \mathbf{I}_r & * \\ \mathbf{0} & \mathbf{0} \end{pmatrix} (T^{-1}x) = S \cdot v$$

which holds if and only if for all $i > r$, the i^{th} entry of $S \cdot v$ is zero.

Remark 3. Similar attack works when $\text{rank}(M'[2]) \leq n-2$. In this case we only need to interchange the role of encryption and decryption queries.

4.2 Case: $\text{rank}(M[2]) = \text{rank}(M'[2]) = n-1$, $\text{rank}(N[1]) \leq n-2$

As $N[1]$ does not have full rank, we can find $Q_1 \neq Q_2$ such that $N[1]Q_1 = N[1]Q_2$. So U values collide for two encryption queries (P, Q_1) and (P, Q_2). Now we write the relationship among intermediate variables. So $A_1 = A_2$ and due to choice of Q_1 and Q_2 we also have $U_1 = U_2$ and hence $V_1 = V_2$. Now, let us write mix_2 function as

$$\mathsf{mix}_2 = \begin{pmatrix} M''[1]_{n\times n} & N''[1]_{n\times(n-1)} \\ M''[2]_{(n-1)\times n} & N''[2]_{(n-1)\times(n-1)} \end{pmatrix}.$$

By the applications of mix_1 and mix_2 functions for two queries, we have

1. $W_1 = M[2] \cdot A_1 + N[2] \cdot Q_1$, $W_2 = M[2] \cdot A_2 + N[2] \cdot Q_2$ and
2. $D_1 = M''[2] \cdot V_1 + N''[2] \cdot W_1$, $D_2 = M''[2] \cdot V_2 + N''[2] \cdot W_2$.

So $W_1 + W_2 = N[2] \cdot (Q_1 + Q_2)$ and $D_1 + D_2 = N''[2] \cdot (W_1 + W_2) = N''[2] \cdot N[2] \cdot (Q_1 + Q_2)$. Note that for a random function, we observe this with probability 2^{-n+1}. We formally describe the attack algorithm by $\mathcal{A}_2$ below.

Distinguishing Algorithm $\mathcal{A}_2$.

1. Let $N[1]Q_1 = N[1]Q_2$.
2. Make encryption query (P_1, Q_1) and obtains response (C_1, D_1).
3. Make encryption query (P_1, Q_2) and obtains response (C_2, D_2).
4. **return** 1 if $D_2 = D_1 + N''[2] \cdot N[2] \cdot (Q_1 + Q_2)$, 0 otherwise.

4.3 Case: $\mathbf{rank}(M[2]) = \mathbf{rank}(M'[2]) = n-1$, $\mathbf{rank}(N[1]) = n-1$

We make three queries same as $\mathcal{A}_0$ except the choice of Q_3 whose value is determined below. We have

1. $U_1 + U_3 = M[1](A_1 + A_2) + N[1]Q_1 + N[1]Q_3$.
2. $M[2](A_1 + A_2) = N[2](Q_1 + Q_2) + N'[2](D_1 + D_2)$ (from the computations of W_1 and W_2).

As the rank of $M[2]$ is $n-1$ and the right hand side of item 2 above is known, we can guess $(A_1 + A_2)$ correctly with probability $1/2$ (since there are only two choices). So we can guess $M[1](A_1 + A_2)$ from $M[2](A_1 + A_2)$ with probability at least $1/2$. Let X be the guessed value of $M[1](A_1 + A_2)$. We now choose Q_3 such that $U_1 + U_3 = 0$ (i.e., $U_1 = U_3$). From the item 1 of above, we define $Q_3 = N[1]^{-1} \cdot X + Q_1$. Note that $N[1]$ is assumed to be invertible in this case. So $\mathsf{Pr}[U_1 = U_3] \geq 1/2$. This essentially leads to a similar distinguisher as in XLS. However, we need to compute the distinguishing event similar to the computation of the previous case. By the applications of mix_1, mix_2^{-1} and mix_2 functions for three queries, we have

1. $W_1 + W_2 = N'[2] \cdot (D_1 + D_2)$,
2. $W_2 + W_3 = N[2] \cdot (Q_2 + Q_3)$, and
3. $N''[2] \cdot (W_1 + W_3) = D_1 + D_3$.

So we have $D_3 = D_1 + N''[2] \cdot (N'[2] \cdot (D_1 + D_2) + N[2] \cdot (Q_1 + Q_3))$ which can be observed with probability 2^{-n+1} for a random function. We formally describe the attack algorithm by $\mathcal{A}_2$ below.

Distinguishing Algorithm $\mathcal{A}_3$.

1. Make encryption query (P_1, Q_1) and obtains response (C_1, D_1).
2. Make decryption query $(C_2 := C_1, D_2 := D_1 + 1)$ and obtains response (P_2, Q_2).
3. Guess $M[1](A_1 + A_2)$, denoted X, from $N[2](Q_1 + Q_2) + N'[2](D_1 + D_2)$
4. Choose Q_3 such that $N[1](Q_1 + Q_3) = X$.
5. Make encryption query $(P_3 = P_2, Q_3)$ and obtains response (C_3, D_3).
6. **return** 1 if $D_3 = D_1 + N''[2] \cdot (N'[2] \cdot (D_1 + D_2) + N[2] \cdot (Q_1 + Q_3))$,
7. **return** 0, otherwise.

5 Conclusion

In this paper we provide chosen plaintext and ciphertext distinguishing attack (i.e., SPRP distinguisher) of XLS. It makes three encryption and decryption calls and has distinguishing advantage about 1/2. This attack can be further extended to a general form of XLS following mix-then-encrypt paradigm. We believe that it can not be repaired without introducing any non-linear functionality, e.g. an additional blockcipher call. So we need four blockcipher calls to make this types of design secure. Both Elastic blokcipher and Nandi's construction make four calls of non-linear functions. However, Nandi's construction could be potentially faster, as it requires two universal hash invocations (which can be achieved by applying four rounds of AES [6]) and one call of weak-PRF (optimistically, one can apply eight rounds of AES) in addition with a full blokcipher call (which is e.g., ten rounds of AES). So in total it requires 26 rounds of AES which is much faster than four full invocations of AES.

Acknowledgement. This work is supported by Centre of Excellence in Cryptology at Indian Statistical Institute, Kolkata.

References

1. CAESAR submissions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.
2. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. Parallelizable and authenticated online ciphers. In *ASIACRYPT (1)*, pages 424–443, 2013.
3. Debrup Chakraborty and Palash Sarkar. HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *Lecture Notes in Computer Science*, pages 287–302. Springer, 2006.
4. Debra L. Cook, Moti Yung, and Angelos D. Keromytis. Elastic aes. *IACR Cryptology ePrint Archive*, 2004:141, 2004.
5. Debra L. Cook, Moti Yung, and Angelos D. Keromytis. Elastic block ciphers: method, security and instantiations. *Int. J. Inf. Sec.*, 8(3):211–231, 2009.
6. Joan Daemen, Mario Lamberger, Norbert Pramstaller, Vincent Rijmen, and Frederik Vercauteren. Computational aspects of the expected differential probability of 4-round aes and aes-like ciphers. *Computing*, 85(1-2):85–104, 2009.
7. Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2004.
8. Shai Halevi. TET: A wide-block tweakable mode based on Naor-Reingold. Cryptology ePrint Archive, Report 2007/014, 2007. http://eprint.iacr.org/.
9. Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2003.

10. Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004.
11. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.
12. M. Luby and C. Rackoff. How to construct pseudo-random permutations from pseudo-random functions. In *Advances in Cryptology – Crypto 1985*, number 218 in Lecture Notes in Computer Science, page 447, New York, 1984. Springer-Verlag.
13. David A. McGrew and Scott R. Fluhrer. The extended codebook (XCB) mode of operation. Cryptology ePrint Archive, Report 2004/278, 2004. http://eprint.iacr.org/.
14. Mridul Nandi. A generic method to extend message space of a strong pseudorandom permutation. *Computación y Sistemas*, 12(3), 2009.
15. Thomas Ristenpart and Phillip Rogaway. How to enrich the message space of a cipher. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 101–118. Springer, 2007.
16. Palash Sarkar. Improving upon the tet mode of operation. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2007.
17. Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A variable-input-length enciphering mode. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *CISC*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005.