How to Enrich the Message Space of a Cipher

Thomas Ristenpart[1] and Phillip Rogaway[2]

[1] Dept. of Computer Science & Engineering, University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404, USA
tristenp@cs.ucsd.edu, http://www-cse.ucsd.edu/users/tristenp
[2] Dept. of Computer Science, University of California Davis
One Shields Avenue, Davis, CA 95616, USA
rogaway@cs.ucdavis.edu, http://www.cs.ucdavis.edu/~rogaway/

Abstract. Given (deterministic) ciphers $\mathcal{E}$ and E that can encipher messages of l and n bits, respectively, we construct a cipher $\mathcal{E}^* = \mathrm{XLS}[\mathcal{E}, E]$ that can encipher messages of $l + s$ bits for any $s < n$. Enciphering such a string will take one call to $\mathcal{E}$ and two calls to E. We prove that $\mathcal{E}^*$ is a strong pseudorandom permutation as long as $\mathcal{E}$ and E are. Our construction works even in the tweakable and VIL (variable-input-length) settings. It makes use of a multipermutation (a pair of orthogonal Latin squares), a combinatorial object not previously used to get a provable-security result.

Key words: Deterministic encryption, enciphering scheme, symmetric encryption, length-preserving encryption, multipermutation.

1 Introduction

Domain extension. Consider a cryptographic scheme with a message space $\mathcal{M} = \bigcup_{l \in \mathcal{L}} \{0,1\}^l$ for some set $\mathcal{L}$ of *permissible message lengths*. The scheme can handle any message of $l \in \mathcal{L}$ bits but it can't handle messages of $l^* \notin \mathcal{L}$ bits. Often the set of permissible message lengths $\mathcal{L}$ is what worked out well for the scheme's *designers*—it made the scheme simple, natural, or amenable to analysis—but it might not be ideal for the scheme's *users* who, all other things being equal, might prefer a scheme that works across arbitrary-length messages. To address this issue, one may wish to *extend* the scheme to handle more message lengths. Examples are extending CBC encryption using ciphertext stealing [21] and extending a pseudorandom function F with message space $(\{0,1\}^n)^+$ by appropriately padding the message and calling F.

Our work is about extending the domain of a *cipher*. When we speak of a *cipher* in this paper we mean a *deterministic* map $\mathcal{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{M} = \bigcup_{l \in \mathcal{L}} \{0,1\}^l$ and $\mathcal{E}_K(\cdot) = \mathcal{E}(K, \cdot)$ is a *length-preserving permutation*. Such an object is also called an *enciphering scheme*, a *pseudorandom permutation*, an *arbitrary-input-length blockcipher*, or a *deterministic* cipher / encryption scheme. Our goal is to extend a cipher $\mathcal{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ with permissible message lengths $\mathcal{L}$ to a cipher $\mathcal{E}^*\colon \mathcal{K}^* \times \mathcal{M}^* \to \mathcal{M}^*$ with an enlarged set $\mathcal{L}^* \supseteq \mathcal{L}$ of permissible message lengths. Being an *extension* of $\mathcal{E}$, what $\mathcal{E}^*$ does on a string of length $l \in \mathcal{L}$ and key $\langle K, K' \rangle$ must be identical to what $\mathcal{E}$ would do on key K. Note that padding-based methods will not work: even if there *is* a point in the message space of $\mathcal{E}$ that one can pad a plaintext to, padding M to M^* and then applying $\mathcal{E}$ would be length-increasing, and so not a cipher. Unlike signatures, MACs, pseudorandom functions, and semantically secure encryption, there is no obvious way to extend a cipher's domain.

Our contribution. We show how, with the help of an n-bit blockcipher E, to extend a cipher's set of permissible message lengths from $\mathcal{L} \subseteq [n\,..\,\infty)$ to $\mathcal{L}^* = \mathcal{L} + [0\,..\,n-1] = \{\ell + i \,|\, \ell \in \mathcal{L} \text{ and } i \in [0\,..\,n-1]\}$. In other words, we enlarge the message space from $\mathcal{M}$ to $\mathcal{M}^* = \mathcal{M} \,||\, \{0,1\}^{<n}$ where $\mathcal{M} \subseteq \{0,1\}^{\geq n}$.

We call our construction XLS (eXtension by Latin Squares). Its overhead is two blockcipher calls, eight xor instructions, and two one-bit rotations. This is the work beyond enciphering (or deciphering) a single l-bit string that is needed to encipher (or decipher) an $l+s$ bit one, where $s \in [1\,..\,n-1]$. If the message is in the original domain there is no overhead beyond determining this. As an example, if $\mathcal{E} = E$ is an n-bit blockcipher then it will take three blockcipher calls to encipher a $2n-1$ bit string.

The XLS method is described in Fig. 1. For a message M already in the domain of $\mathcal{E}$, just apply it. Otherwise, suppose that M has length $l+s$ where $l \in \mathcal{L}$ and $s \in [1\,..\,n-1]$. To encipher M: apply the blockcipher E to the last full n-bit block of M; mix together the last $2s$-bits; flip the immediately preceding bit; encipher under $\mathcal{E}$ the first l bits; mix together the last $2s$-bits; flip the immediately preceding bit; then apply E to the last full n-bit block. Our recommended instantiation of the mixing step uses three xors and a single one-bit circular rotation.

We prove that XLS works. More specifically, if $\mathcal{E}$ is secure in the sense of a strong pseudorandom permutation (a strong PRP) [17] then $\mathcal{E}^*$ inherits this property. This assumes that the blockcipher E is likewise a strong PRP. The result holds even in the variable-input-length (VIL) setting [3]: if $\mathcal{E}$ is VIL-secure then so is $\mathcal{E}^*$. See Theorem 2. It also holds in the tweakable-enciphering-scheme setting [16]: if $\mathcal{E}$ is tweakable then $\mathcal{E}^*$ inherits this. See Section 7. If one makes the weaker assumption that $\mathcal{E}$ and E are ordinary (not necessarily strong) PRPs, then one can conclude that $\mathcal{E}^*$ is a PRP. See Section 8.

While XLS is relatively simple, it is surprisingly delicate. We show that natural alternative ways of mixing do not work. We show that omitting the bit flip does not work. And attempting to get by without any mixing—say by enciphering the last n bits, the first l bits, then the last n bits—doesn't work even if one demands that the "overlap" in what is enciphered is $n/2$ bits: there is an attack of complexity $2^{n/4}$.

All that said, we develop sufficient conditions on the mixing function that *are* enough to guarantee security, and we provide a mixing function based on multipermutations (also called orthogonal Latin squares [7]). Though conceptually elegant, implementing multipermutations in this setting is slightly complicated, so we provide an alternate mixing function that approximates multipermutations via bit rotations. This comes at the (insignificant) cost of a slightly larger constant in the security reduction. XLS is the first mode of operation to employ multipermutations or approximate multipermutations to yield a provable-security guarantee. Indeed, such mixing functions may prove to be useful in further provable-security contexts.

We comment that we cannot handle messages of length less than n bits (the blocklength of the blockcipher that we use)—for example, we don't know how to encipher a 32-bit string using AES (in an efficient way and with a known and desirable security bound). This is a long open problem [6, 12].

Related work. There are several known methods for turning a blockcipher with message space $\mathcal{M} = \{0,1\}^n$ into a cipher with some message space $\{0,1\}^{\geq n}$. Halevi does this in his EME* and TET constructions [12, 13]; Fluhrer and McGrew do it (without a provable-security guarantee) with XCB [20]; Wang, Feng, and Wu do it in HCTR [30]; and Chakraborty and Sarkar do it in HCH [9]. All of these constructions are somewhat complex, and their methods for dealing with "inconvenient-length" strings are non-generic. Constructions of ciphers from n-bit blockciphers that result in a message space like $(\{0,1\}^n)^+$ are offered by Zheng, Matsumoto, and Imai [31], Naor and Reingold [22], Halevi and Rogaway [14, 15], Patel, Ramzan, and Sundaram [24], and Chakraborty and Sarkar [8]. One can even view Luby and Rackoff [17] in this light.

Anderson and Biham [2] and Lucks [18, 19] make a wide-blocksize cipher out of a stream cipher and a hash function, and Schroeppel provides a cipher [28] that works on an arbitrary message space *de novo*.

When $\mathcal{E} = E$ is an n-bit blockcipher, the XLS construction solves the *elastic blockcipher* problem of Cook, Yung, and Keromytis [10, 11], where one wants to extend a blockcipher from n bits to $[n \mathinner{..} 2n-1]$ bits. The Cook *et. al* solution is heuristic—there is no proof of security—but with XLS we have, for example, an "elastic AES" that provably preserves the security of AES.

When a cipher like CMC or EME [14, 15] plays the role of $\mathcal{E}$ in XLS, one gets a cipher with efficiency comparable to that of a mode like EME* [12].

Bellare and Rogaway first defined VIL ciphers [3] and built one (although it is not secure as a *strong* PRP). An and Bellare [1] offer the viewpoint that cryptographic constructions are often aimed at adjusting the domain of a primitive. This viewpoint is implicit in our work.

Applications. While primarily interested in the "theoretical" question of how to accomplish domain extension for ciphers, arbitrary-input-length enciphering is a problem with many applications. A well-known application is disk-sector encryption, the problem being addressed by the IEEE Security in Storage Work Group P1619. Another application is saving bandwidth in network protocols: if one has a 53-byte payload to be enciphered, and no IV or sequence number to do it, the best that can be done without increasing the size of the datagram is to encipher this 53-byte string. A related application is the security-retrofitting of legacy communications protocols, where there is a mandated and immutable allocation of bytes in a datagram, this value not necessarily a multiple of, say, 16 bytes. Another application is in a database setting where it should be manifest when two confidential database records are identical, these records having arbitrary length that should not be changed, but nothing else about the records should be leaked. Arbitrary-length enciphering enables bandwidth-efficient use of the encode-then-encipher paradigm of Bellare and Rogaway [4], where one gets authenticity by enciphering strings encoded with redundancy and semantic security by enciphering strings that rarely collide.

2 Preliminaries

Basics and notation. For strings $X, Y \in \{0,1\}^*$, we use $X \,\|\, Y$ or $X\,Y$ to denote concatenation. We write $X[i]$ for $i \in [1\,..\,|X|]$ to represent the i^{th} bit of X (thus $X = X[1]X[2]\cdots X[s]$). The complement of a bit b is $\texttt{flip}(b)$. For a set $\mathcal{C}$ and element X we write $\mathcal{C} \xleftarrow{\cup} X$ for $\mathcal{C} \leftarrow \mathcal{C} \cup \{X\}$. We require that for any set of bit strings $\mathcal{S} \subseteq \{0,1\}^*$, if $X \in \mathcal{S}$ then $\{0,1\}^{|X|} \subseteq \mathcal{S}$.

A *cipher* is a map $\mathcal{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}$ is a nonempty set, $\mathcal{M} \subseteq \{0,1\}^*$ is a nonempty set, and $E_K(\cdot) = E(K, \cdot)$ is a length-preserving permutation. The set $\mathcal{K}$ is called the *key space* and the set $\mathcal{M}$ is called the *message space*. We can view the message space as $\cup_{l \in \mathcal{L}} \{0,1\}^l$ where $\mathcal{L} = \{l \mid \exists X \in \mathcal{M} \text{ s.t. } |X| = l\}$. Let $\mathcal{D}$ be the cipher with the same signature as $\mathcal{E}$ and defined by $\mathcal{D}_K(Y) = X$ iff $\mathcal{E}_K(X) = Y$. A *blockcipher* is a cipher with message space $\mathcal{M} = \{0,1\}^n$ for some $n \geq 1$ (the blocksize). For $\mathcal{M} \subseteq \{0,1\}^*$ let $\mathrm{Perm}(\mathcal{M})$ be the set of all length-preserving permutations on $\mathcal{M}$. By selecting $\mathcal{K} = \mathrm{Perm}(\mathcal{M})$ we have a cipher for which a uniformly chosen permutation on $\{0,1\}^l$ is selected for each $l \in \mathcal{L}$. Let $\mathrm{Func}(\mathcal{M})$ be the set of all length-preserving functions on $\mathcal{M}$. Write $\mathrm{Perm}(\ell)$ and $\mathrm{Func}(\ell)$ for $\mathrm{Perm}(\{0,1\}^\ell)$ and $\mathrm{Func}(\{0,1\}^\ell)$, respectively.

Let $\mathcal{S} \subseteq \{0,1\}^{\geq 1}$. Then define $\mathcal{S}^2 = \{XY \mid X, Y \in \mathcal{S} \wedge |X| = |Y|\}$. Let $f\colon \mathcal{S}^2 \to \mathcal{S}^2$ be a length-preserving function. We define the *left projection* of f as the function $f_{\mathrm{L}}\colon \mathcal{S}^2 \to \mathcal{S}$ where $f_{\mathrm{L}}(X)$ is equal to the first $|X|/2$ bits of $f(X)$. We define the *right projection* of f as the function $f_{\mathrm{R}}\colon \mathcal{S}^2 \to \mathcal{S}$ where $f_{\mathrm{R}}(X)$ is equal to the last $|X|/2$ bits of $f(X)$. Of course $f(X) = f_{\mathrm{L}}(X) \,\|\, f_{\mathrm{R}}(X)$.

When we say "Replace the last ℓ bits of M, Last, by $F(\mathrm{Last})$" we mean (1) parse M into $X \,\|\, \mathrm{Last}$ where $|X| = |M| - \ell$ and $|\mathrm{Last}| = \ell$; (2) let Z be $F(\mathrm{Last})$; and (3) replace M by $X \,\|\, Z$. We define the semantics of similar uses of "Replace ..." in the natural way.

The notation "$XYZ \leftarrow M$ *of lengths* x, y, z" for any string M with $|M| = x + y + z$ means parse M into three strings of length x, y, and z and assign these values to X, Y, and Z, respectively. The notation is extended to the case of parsing M into two halves in the natural way.

Finally, an *involution* is a permutation g which is its own inverse: $g(g(x)) = x$.

Security notions. When an adversary $\mathcal{A}$ is run with an oracle $\mathcal{O}$ we let $A^{\mathcal{O}} \Rightarrow 1$ denote the event that $\mathcal{A}$ outputs the bit 1. Let $\mathcal{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ be a cipher.

Then we define the following advantages for an adversary $\mathcal{A}$:

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\mathrm{prp}}(\mathcal{A}) = \Pr\Big[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K,\, \mathcal{D}_K} \Rightarrow 1\Big] - \Pr\Big[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{M}) : A^{\pi,\, \pi^{-1}} \Rightarrow 1\Big]$$

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\mathrm{prf}}(\mathcal{A}) = \Pr\Big[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K,\, \mathcal{D}_K} \Rightarrow 1\Big] - \Pr\Big[\rho, \sigma \xleftarrow{\$} \mathrm{Func}(\mathcal{M}) : A^{\rho,\, \sigma} \Rightarrow 1\Big]$$

where the probabilities are over the choice of K or choice of π (resp. ρ, σ) and the coins used by $\mathcal{A}$. The first experiment represents distinguishing $\mathcal{E}$ and its inverse from a random length-preserving permutation and its inverse and the second experiment represents distinguishing $\mathcal{E}$ and its inverse from two random length-preserving functions. In both settings, we demand that the adversary $\mathcal{A}$, given oracles f, g, does not repeat any query, does not ask $g(Y)$ after receiving Y in response to some query $f(X)$, and does not ask $f(X)$ after receiving X in response to some query $g(Y)$. Such forbidden queries are termed *pointless*.

While the above formalization allows *variable input length* (VIL) adversaries, we can also restrict adversaries to only query messages of a single length. We call such adversaries *fixed input length* (FIL) adversaries.

Informally, a cipher is called a "strong pseudorandom permutation" if no reasonable adversary $\mathcal{A}$ can distinguish the enciphering and deciphering functions, randomly keyed, from a randomly selected permutation and its inverse: $\mathbf{Adv}_{\mathcal{E}}^{\pm\mathrm{prp}}(\mathcal{A})$ is small. Our theorems make concrete statements about this and so we will not have to formalize "reasonable" or "small." Resources we pay attention to are the adversary's maximum running time (which, by convention, includes the length of the program); the number of queries it asks; and the lengths of the queries. For any cipher $\mathcal{E}$ with inverse $\mathcal{D}$, define $\mathsf{Time}_{\mathcal{E}}(\mu) = \max\{T_{key}, T_{\mathcal{E}}, T_{\mathcal{D}}\}$ where T_{key} is the maximum time required to generate a key L for the scheme, $T_{\mathcal{E}}$ is the maximum time to run $\mathcal{E}_L$ on a message of at most μ bits, and $T_{\mathcal{D}}$ is the maximum time to run $\mathcal{D}_L$ on a ciphertext of at most μ bits.

3 The XLS Construction

Fix a blocksize n. Let $\mathcal{E}$: $\mathcal{K}_{\mathcal{E}} \times \mathcal{M} \to \mathcal{M}$ be a cipher with $\mathcal{M} \subseteq \{0,1\}^{\geq n}$ and let E: $\mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Finally, define a length-preserving permutation mix: $\mathcal{S}^2 \to \mathcal{S}^2$ where $\mathcal{S} \supseteq \cup_{i=1}^{n-1}\{0,1\}^i$. Then we define a cipher $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mix}, \mathcal{E}, E]$ with key space $\mathcal{K}^* = \mathcal{K}_{\mathcal{E}} \times \mathcal{K}_E$ and message space $\mathcal{M}^* = \mathcal{M} \mid\mid \{0,1\}^{<n}$. For keys $L \in \mathcal{K}_{\mathcal{E}}$ and $K \in \mathcal{K}_E$ we have $\mathcal{E}^*_{L,K}(\cdot) = \mathcal{E}^*((L,K), \cdot)$. See Fig. 1 for the definition.

Enciphering a message M with $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mix}, \mathcal{E}, E]$ is straightforward. If $M \in \mathcal{M}$, then simply apply $\mathcal{E}$. Otherwise, apply E to the last full n-bit block of M and replace those bits with the result. Then 'mix together' the last $2s$ bits, again replacing the appropriate bits with the resulting mixture. Flip bit $|M| - 2s$, which is the first bit from the right not affected by mix. Apply $\mathcal{E}$ to as many bits as possible, starting from the left. Finally, just repeat the first three steps in reverse order. Deciphering is equally simple, and in fact, if one implements mix with an involution, as we suggest, then the inverse of $\mathcal{E}^*$ is just $\mathcal{D}^* = \mathrm{XLS}[\mathsf{mix}, \mathcal{D}, D]$.

Algorithm $\mathcal{E}^*_{L,K}(M)$

If $M \in \mathcal{M}$ then return $\mathcal{E}_L(M)$
Let $l < |M|$ be largest number such that $\{0,1\}^l \subseteq \mathcal{M}$; $s \leftarrow |M| - l$
If $s \geq n$ or $l < n$ then Return $\perp$
Replace the last full n-bit block of M, LastFull, with $E_K(\text{LastFull})$
Replace the last $2s$ bits of M, Last, with $\mathsf{mix}(\text{Last})$
Replace the $(|M| - 2s)$'th bit of M, b, with $\texttt{flip}(b)$
Replace the first l bits of M, First, with $\mathcal{E}_L(\text{First})$
Replace the $(|M| - 2s)$'th bit of M, b, with $\texttt{flip}(b)$
Replace the last $2s$ bits of M, Last, with $\mathsf{mix}(\text{Last})$
Replace the last full n-bit block of M, LastFull, with $E_K(\text{LastFull})$
Return M

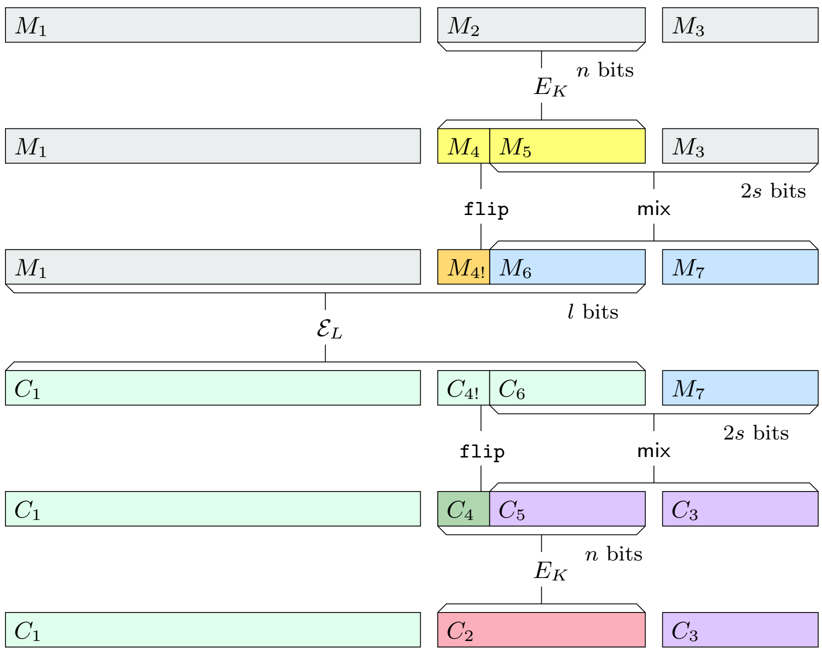

Fig. 1. Top: Enciphering algorithm $\mathcal{E}^* = \text{XLS}[\mathsf{mix}, \mathcal{E}, E]$. **Bottom:** Enciphering $M = M_1 \,||\, M_2 \,||\, M_3$ under $\mathcal{E}^*$ where M is not in $\mathcal{M}$; $l < |M|$ is the largest value such that $\{0,1\}^l \subseteq \mathcal{M}$; $s = |M| - l$; $M_1 \in \{0,1\}^{l-n}$; $M_2 \in \{0,1\}^n$; and $M_3 \in \{0,1\}^s$.

Why, intuitively, should XLS work? "Working" entails that each output bit strongly depends on each input bit. Since $\mathcal{E}$ presumably already does a good job of this we need only worry about mixing in the "leftover" s bits for $M \notin \mathcal{M}$. We mix in these bits utilizing the mixing function mix. But since mix will be a simple combinatorial object—it is unkeyed and will have no "cryptographic" property—we need to "protect" its input with the blockcipher. The "symmetrizing" of the protocol—repeating the blockcipher call and the mixing step in the reverse order so that lines 03 $\rightarrow$ 09 are identical to lines 09 $\rightarrow$ 03—helps achieve *strong* PRP-security: each input bit must strongly depend on each output bit, as queries can be made in the forward *or* backward direction. Finally, the bit-flipping step is just a symmetry-breaking technique to ensure that different-length messages are treated differently.

If mix does a "good" job of mixing, then XLS will in fact be secure, as we prove in Section 6. But what is the meaning of "good," and how do we make a mixing function that is simultaneously good, efficient, and easy to implement? We now turn towards answering these questions.

4 The Mixing Function

We now look at several possible ways of implementing mix, to build intuition on what properties are needed for the security of XLS. In the end we formally define, quantitatively, the sufficient condition of interest. For ease of exposition we will often silently parse the input to mix into its two halves, i.e., $\mathsf{mix}(AB)$ means that $A \,||\, B \in \mathcal{S}^2$ and that $|A| = |B|$. Also we interchangeably write $\mathsf{mix}(AB)$ and $\mathsf{mix}(A, B)$, which are equivalent.

A NAIVE APPROACH. Let's start with a natural construction that, perhaps surprisingly, does not lead to a secure construction. Suppose we define $\mathsf{mixWrong}$ by saying that $\mathsf{mixWrong}(AB) = A \oplus B \,||\, B$ for equal-length A, B: the mixing function xors the right half of the input into the left half, outputting the result and the original right half. Clearly $\mathsf{mixWrong}$ is a length-preserving permutation. Furthermore, it might seem sufficient for XLS because it will mix the "leftover" bits into those handled by $\mathcal{E}$. But this intuition is flawed: $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mixWrong}, \mathcal{E}, E]$ is easily distinguished from a length-preserving permutation on $\mathcal{M}^*$. An adversary can simply query $0^n \,||\, 0^{n-1}$ and $1^n \,||\, 0^{n-1}$. As one can easily verify, both $\mathcal{E}^*(0^n \,||\, 0^{n-1})$ and $\mathcal{E}^*(1^n \,||\, 0^{n-1})$ will have output with the last $n-1$ bits equal to 0^{n-1}. This would be true of a random permutation with probability at most $1/2^{n-1}$, and so the adversary's advantage is close to one. In fact $\mathsf{mixWrong}$ does not do a good job of mixing: the right half of the output is only a function of the right half of the input. One can try various fixes, but ultimately it appears that using *just* xors is inherently inadequate.

USING ORTHOGONAL LATIN SQUARES. The failure above suggests that what is needed is a mixing function with symmetry, in the sense that both the left and right halves of the output are dependent on both the left and right halves of the input. To achieve such a goal we can turn to the classical combinatorial objects known as a pair of *orthogonal Latin squares* [7], also called a *multipermutation* [27, 29]. This is a permutation $\mathsf{mix}\colon \mathcal{S}^2 \rightarrow \mathcal{S}^2$ such that, for any $C \in S$, $\mathsf{mix}_{\mathrm{L}}(C, \cdot)$, $\mathsf{mix}_{\mathrm{L}}(\cdot, C)$, $\mathsf{mix}_{\mathrm{L}}(C, \cdot)$, and $\mathsf{mix}_{\mathrm{R}}(\cdot, C)$ are all permutations, where $\mathsf{mix}_{\mathrm{L}}$ and $\mathsf{mix}_{\mathrm{R}}$ denote the projection of mix onto its first and second component. Let us describe a concrete realization. Fix a finite field $\mathbb{F}_{2^s}$ for each s and view each s-bit string as an element of this field. Then we can build a mixing function $\mathsf{mix1}$

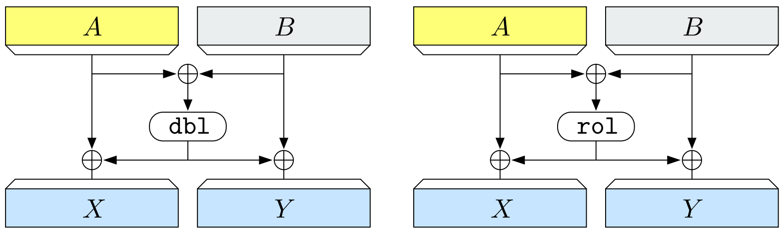

Fig. 2. The mixing functions $\mathsf{mix1}$ (left) and $\mathsf{mix2}$ (right). Here $A, B \in \{0,1\}^{<n}$ with $s = |A| = |B|$. The operation $\mathtt{dbl}$ is a multiplication by $\mathbf{2} = 0^{s-2}10 = \mathsf{x}$ in the finite field $\mathbb{F}_{2^s}$ and $\mathtt{rol}$ is a circular left rotation by one bit.

by saying that

$$\mathsf{mix1}(AB) = (\mathbf{3}A + \mathbf{2}B) \mid\mid (\mathbf{2}A + \mathbf{3}B) = (A + \mathbf{2}(A+B), B + \mathbf{2}(A+B))$$

for equal-length strings A and B (we will assume the length to be at least 2). Here addition and multiplication are over $\mathbb{F}_{2^s}$ and $\mathbf{2} = 0^{s-2}10 = \mathsf{x}$ and $\mathbf{3} = 0^{s-2}11 = \mathsf{x} + 1$. Addition is bitwise xor and multiplication by $\mathbf{2}$, which we also denote $\mathtt{dbl}$, can be implemented by a shift and a conditional xor. The mixing function $\mathsf{mix1}$ has several nice properties. First, it is a permutation and, in fact, an involution (meaning $\mathsf{mix} = \mathsf{mix}^{-1}$). Moreover, for any $C \in \{0,1\}^s$ we have that $\mathsf{mix1}_{\text{L}}(C, \cdot)$, $\mathsf{mix1}_{\text{R}}(C, \cdot)$, $\mathsf{mix1}_{\text{L}}(\cdot, C)$, and $\mathsf{mix1}_{\text{R}}(\cdot, C)$ are all permutations on $\{0,1\}^s$.

We show later that, when used in XLS, mixing function $\mathsf{mix1}$ leads to a secure construction. Moreover, it is fast and relatively simple. But implementing it in XLS requires a table of constants corresponding to irreducible polynomials, one for each $s \le n-1$. As it turns out, we can do better.

The simplified mixing function. We now simplify the mixing function $\mathsf{mix1}$. Let $\mathtt{rol}(X)$ represent left circular bit-rotation, that is, for any string X of length s let $\mathtt{rol}(X) = X[2]X[3]\cdots X[s]X[1]$. Then define $\mathsf{mix2}$ by

$$\mathsf{mix2}(AB) = (A \oplus \mathtt{rol}(A \oplus B)) \mid\mid (B \oplus \mathtt{rol}(A \oplus B))$$

where A and B are equal-length strings. See Fig. 2. Notice the similarity with $\mathsf{mix1}$: we replaced multiplication by two with a left circular rotation. The bit rotation "approximates" a proper multiplication, eliminating, in an implementation, the conditional xor and the table of constants. As before, $\mathsf{mix2}$ is an involution.

Quantifying the quality of mixing functions. We now formalize the properties of a mixing function that are needed in the proof of XLS.

Definition 1. *Fix a set $\mathcal{S} \subseteq \{0,1\}^{\ge 1}$, let* mix*: $\mathcal{S}^2 \to \mathcal{S}^2$ be a length-preserving permutation, and let $\epsilon \colon \mathbb{N} \to [0,1]$. We say that* mix *is an $\epsilon(s)$-**good mixing function** if, for all s such that $\{0,1\}^s \subseteq \mathcal{S}$, we have that*

(1) $\mathsf{mix}_{\mathrm{L}}(A,\cdot)$ *is a permutation for all* $A \in \{0,1\}^s$*,*

(2) $\mathsf{mix}_{\mathrm{R}}(\cdot,B)$ *is a permutation for all* $B \in \{0,1\}^s$*,*

(3) $\Pr[R \stackrel{\$}{\leftarrow} \{0,1\}^s : C = \mathsf{mix}_{\mathrm{L}}(R,B)] \le \epsilon(s)$ *for all* $B, C \in \{0,1\}^s$*, and*

(4) $\Pr[R \stackrel{\$}{\leftarrow} \{0,1\}^s : C = \mathsf{mix}_{\mathrm{R}}(A,R)] \le \epsilon(s)$ *for all* $A, C \in \{0,1\}^s$ *.* □

The best one can hope for is a 2^{-s}-good mixing function. In fact mix1 is such a function, while the mix2 function is just a factor of two off.

Lemma 1. *The mixing function* mix1 *is a* 2^{-s}*-good mixing function. The mixing function* mix2 *is a* 2^{-s+1}*-good mixing function.* □

Proof: Since for any $s \in [1 \mathinner{..} n-1]$ and any $C \in \{0,1\}^s$ we have that $\mathsf{mix1}_{\mathrm{L}}(C,\cdot)$, $\mathsf{mix1}_{\mathrm{R}}(C,\cdot)$, $\mathsf{mix1}_{\mathrm{L}}(\cdot,C)$, and $\mathsf{mix1}_{\mathrm{R}}(\cdot,C)$ are all permutations, the first half of the lemma is clear.

That mix2 meets parts 1 and 2 of the definition is clear. For the third part, we have that

$$\mathsf{mix2}_{\mathrm{L}}(R,B) = R \oplus \mathtt{rol}(R \oplus B) = R \oplus \mathtt{rol}(R) \oplus \mathtt{rol}(B)$$

and so we bound the number of values R such that $C \oplus \mathtt{rol}(B) = R \oplus \mathtt{rol}(R)$. Let $C' = C \oplus \mathtt{rol}(B)$, which is a constant. Then we have that

$$\begin{aligned} R[1] \oplus R[s] &= C'[1] \\ R[2] \oplus R[1] &= C'[2] \\ &\vdots \\ R[s-1] \oplus R[s-2] &= C'[s-1] \\ R[s] \oplus R[s-1] &= C'[s] \end{aligned}$$

Note that there only exists a string R that satisfies the above equalities if $C'[1] \oplus C'[2] \oplus \cdots \oplus C'[s] = 0$. If there exists a solution, then pick a value for $R[1]$. That choice and the equations above combine to specify $R[2], \ldots, R[s]$. This means that there are at most two possible values of R and so the probability that $C = \mathsf{mix2}_{\mathrm{L}}(R,B)$ is at most $2/2^s$. The proof of the fourth part is symmetric. ▮

We point out that the properties of circular rotations combined with xors as utilized in mix2 have been used before in different settings, such as [23].

We have introduced two mixing functions for the following reason: mix1 is conceptually more elegant, while mix2 is operationally more elegant. In addition, mix2 is in effect an approximation of mix1, making the latter an important conceptual building block. Such mixing functions might prove useful in future provable-security results.

Note that our definition of an $\epsilon(s)$-good mixing function is general, but XLS requires a mixing function for which $\mathcal{S} \supseteq \cup_{i=1}^{n-1}\{0,1\}^i$. This is clearly the case for mix1 and mix2. For the rest of the paper when we refer to an $\epsilon(s)$-good mixing function, we implicitly require that this function is well-defined for such an $\mathcal{S}$.

5 The Bit Flips

In steps 05 and 07 of XLS (see Fig. 1) we flip a single bit. Flipping bits in this manner is unintuitive and might seem unimportant for the security of XLS. However, the bit flips are actually crucial for the security of the scheme when in the VIL setting. Let $\mathcal{E}^\dagger$ be the cipher defined by running the algorithm of Fig. 1 except with lines 05 and 07 omitted. Then the following VIL adversary $\mathcal{A}$ easily distinguishes $\mathcal{E}^\dagger$ from a family of random permutations. The adversary $\mathcal{A}$ makes two enciphering queries on $M = 0^{n+1}$ and $M' = 0^{n+2}$, getting return values C and C' respectively. If the first n bits of C and C' are equal, then $\mathcal{A}$ outputs 1 (the oracles are likely the construction) and otherwise outputs 0 (the oracles are likely a random permutation). We have that $\Pr[K \xleftarrow{\$} \mathcal{K}^* : \mathcal{A}^{\mathcal{E}^\dagger(\cdot),\mathcal{D}^\dagger(\cdot)} \Rightarrow 1] = 1$. This is so because for both queries the inputs to $\mathcal{E}$ are necessarily the same (as one can verify quickly by following along in the diagram in Fig. 1; remember to omit the flip steps). Clearly $\Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{M}^*) : \mathcal{A}^{\pi(\cdot),\pi^{-1}(\cdot)} \Rightarrow 1] = 2^{-n}$ and so $\mathcal{A}$ has large advantage.

6 Security of XLS

We are now ready to prove the security of XLS. The proof is broken into two parts: first we show that XLS is secure in an information-theoretic setting (i.e., using actual random permutations as components). Afterwards we pass to a complexity-theoretic setting to get our main result.

Theorem 1. *Fix n and an $\epsilon(s)$-good mixing function* mix. *Let $\mathcal{M} \subseteq \{0,1\}^{\geq n}$ and $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mix}, \mathrm{Perm}(\mathcal{M}), \mathrm{Perm}(n)]$. Then for any adversary $\mathcal{A}$ that asks at most q queries we have that $\mathbf{Adv}^{\pm\mathrm{prf}}_{\mathcal{E}^*}(A) \leq 5q^2\,\epsilon(s)/2^{n-s} + 3q^2/2^n$ for any $s \in [1\,..\,n-1]$, and so, by Lemma 1,*

$$\mathbf{Adv}^{\pm\mathrm{prf}}_{\mathcal{E}^*}(A) \leq \frac{8q^2}{2^n} \quad \text{and} \quad \mathbf{Adv}^{\pm\mathrm{prf}}_{\mathcal{E}^*}(A) \leq \frac{13q^2}{2^n}$$

for $\mathsf{mix} = \mathsf{mix1}$ *and* $\mathsf{mix} = \mathsf{mix2}$*, respectively.* $\square$

Proof. Due to space constraints, we only present a self-contained chunk of the proof together with a sketch of the other portion of the proof. See the full version [25] for the complete proof. Fix n and let $\mathsf{mix}\colon \mathcal{S}^2 \to \mathcal{S}^2$ be an $\epsilon(s)$-good mixing permutation. Let $\mathcal{E}\colon \mathrm{Perm}(\mathcal{M}) \times \mathcal{M} \to \mathcal{M}$ be a cipher with message space $\mathcal{M}$ and let $E\colon \mathrm{Perm}(n) \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Note that these last two simply implement a family of random length-preserving permutations on $\mathcal{M}$ and a random permutation on $\{0,1\}^n$, respectively. Let $\mathcal{A}$ be a $\pm$prf adversary against $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mix}, \mathcal{E}, E]$. We therefore must bound

$$\mathbf{Adv}^{\pm\mathrm{prf}}_{\mathcal{E}^*}(\mathcal{A}) = \Pr\Big[K \xleftarrow{\$} \mathcal{K}^* : A^{\mathcal{E}^*_K, \mathcal{D}^*_K} \Rightarrow 1\Big] - \Pr\Big[\rho, \sigma \xleftarrow{\$} \mathrm{Func}(\mathcal{M}^*) : A^{\rho,\,\sigma} \Rightarrow 1\Big]\,.$$

Recall that we disallow $\mathcal{A}$ from making pointless queries. We utilize a game-playing argument [5] and the first two games are G0 and G1, shown in Fig. 3.

procedure Choose$\mathcal{E}$(X) $\quad$ **procedure** Choose$\mathcal{D}$(Y) $\quad$ $\boxed{\text{G0}}$ G1

procedure Choose$\mathcal{E}$(X)	**procedure** Choose$\mathcal{D}$(Y)
$Y \xleftarrow{\$} \{0,1\}^{\lvert X\rvert}$	$X \xleftarrow{\$} \{0,1\}^{\lvert Y\rvert}$
If $Y \in \mathcal{RE}$ then $bad \leftarrow$ true $\boxed{, Y \xleftarrow{\$} \overline{\mathcal{RE}}}$	If $X \in \mathcal{DE}$ then $bad \leftarrow$ true $\boxed{, Y \xleftarrow{\$} \overline{\mathcal{DE}}}$
If $X \in \mathcal{DE}$ then $bad \leftarrow$ true $\boxed{, Y \leftarrow \mathcal{E}(X)}$	If $Y \in \mathcal{RE}$ then $bad \leftarrow$ true $\boxed{, X \leftarrow \mathcal{D}(Y)}$
$\mathcal{E}(X) \leftarrow Y$; $\mathcal{D}(Y) \leftarrow X$	$\mathcal{E}(X) \leftarrow Y$; $\mathcal{D}(Y) \leftarrow X$
$\mathcal{RE} \xleftarrow{\cup} Y$; $\mathcal{DE} \xleftarrow{\cup} X$; Return Y	$\mathcal{RE} \xleftarrow{\cup} Y$; $\mathcal{DE} \xleftarrow{\cup} X$; Return X
procedure Choose$E$(X)	**procedure** Choose$D$(Y)
$Y \xleftarrow{\$} \{0,1\}^{\lvert X\rvert}$	$X \xleftarrow{\$} \{0,1\}^{\lvert Y\rvert}$
If $Y \in RE$ then $bad \leftarrow$ true $\boxed{, Y \xleftarrow{\$} \overline{RE}}$	If $X \in DE$ then $bad \leftarrow$ true $\boxed{, Y \xleftarrow{\$} \overline{DE}}$
If $X \in DE$ then $bad \leftarrow$ true $\boxed{, Y \leftarrow E(X)}$	If $Y \in RE$ then $bad \leftarrow$ true $\boxed{, X \leftarrow D(Y)}$
$E(X) \leftarrow Y$; $D(Y) \leftarrow X$	$E(X) \leftarrow Y$; $D(Y) \leftarrow X$
$RE \xleftarrow{\cup} Y$; $DE \xleftarrow{\cup} X$; Return Y	$RE \xleftarrow{\cup} Y$; $DE \xleftarrow{\cup} X$; Return X

procedure Enc(M)
$j \leftarrow j+1$; $M^j \leftarrow M$; If $M^j \in \mathcal{M}$ then Return $C^j \leftarrow$ Choose$\mathcal{E}$(M^j)
Let s be smallest number s.t. $\{0,1\}^{\lvert M^j\rvert - s} \in \mathcal{M}$
$m \leftarrow \lvert M^j\rvert - n - s$; $M_1^j\ M_2^j\ M_3^j \leftarrow M^j$ *of lengths* m, n, s
Let $i \in [1\,..\,j]$ be smallest index s.t. $M_2^j = M_2^i$
If $i < j$ then $M_4^j \leftarrow M_4^i$; $M_5^j \leftarrow M_5^i$
Else $M_4^j\ M_5^j \leftarrow$ Choose$E$(M_2^j) *of lengths* $n-s$, s
$M_6^j\ M_7^j \leftarrow \mathsf{mix}(M_4^j, M_3^j)$ *of lengths* s, s
$C_1^j\ C_{4!}^j\ C_6^j \leftarrow$ Choose$\mathcal{E}$($M_1^j \parallel \texttt{flip1}(M_4^j) \parallel M_6^j$) *of lengths* m, $n-s$, s
$C_5^j\ C_3^j \leftarrow \mathsf{mix}(C_6^j, M_7^j)$ *of lengths* s, s
$C_2^j \leftarrow$ Choose$E$($\texttt{flip1}(C_{4!}^j) \parallel C_5^j$)
Return $C_1^j\ C_2^j\ C_3^j$

procedure Dec(C)
$j \leftarrow j+1$; $C^j \leftarrow C$; If $C^j \in \mathcal{M}$ then Return $M^j \leftarrow$ Choose$\mathcal{D}$(C^j)
Let s be smallest number s.t. $\{0,1\}^{\lvert C^j\rvert - s} \in \mathcal{M}$
$m \leftarrow \lvert C^j\rvert - n - s$; $C_1^j\ C_2^j\ C_3^j \leftarrow C^j$ *of lengths* m, n, s
Let $i \in [1\,..\,j]$ be smallest index s.t. $C_2^j = C_2^i$
If $i < j$ then $C_4^j \leftarrow C_4^i$, $C_5^j \leftarrow C_5^i$
Else $C_4^j\ C_5^j \leftarrow$ Choose$D$(C_2^j) *of lengths* $n-s$, s
$C_6^j\ M_7^j \leftarrow \mathsf{mix}(C_4^j, C_3^j)$ *of lengths* s, s
$M_1^j\ M_{4!}^j\ M_6^j \leftarrow$ Choose$\mathcal{D}$($C_1^j \parallel \texttt{flip1}(C_4^j) \parallel C_5^j$) *of lengths* m, $n-s$, s
$M_5^j\ M_3^j \leftarrow \mathsf{mix}(M_6^j\ M_7^j)$ *of lengths* s, s
$M_2^j \leftarrow$ Choose$D$($\texttt{flip1}(M_{4!}^j) \parallel M_5^j$)
Return $M_1^j\ M_2^j\ M_3^j$

Fig. 3. Games G0 (boxed statements included) and G1 (boxed statements dropped) used in the proof of Theorem 1. Initially, $j = 0$ and $\mathcal{DE}, \mathcal{RE}, DE, RE$ are empty sets and the partial functions $\mathcal{E}, \mathcal{D}, E, D$ are everywhere undefined. The function $\texttt{flip1}(X)$, for any bit string $X = X[1]\cdots X[s]$, outputs the string with last bit complemented: $X[1]\cdots X[s-1]\texttt{flip}(X[s])$.

In G0 we build $\mathcal{E}$ and E lazily using the appropriate Choose procedures, and so $\mathcal{E}$ and E are partial functions in this context. Note that $\mathcal{DE}, \mathcal{RE}, DE, RE$ are initially empty and the functions $\mathcal{E}, \mathcal{D}, E, D$ are everywhere undefined. As usual, $\mathcal{D}$ and D represent the inverses of $\mathcal{E}$ and E. While game G0, which includes the boxed statements, enforces that $\mathcal{E}$ and E be length-preserving permutations, game G1 dispenses with that requirement (the boxed statements are not included in G1). A flag *bad* is initially false and set to true when, in the course of building $\mathcal{E}$ and E, a duplicate domain or range point is initially selected. In G0 these points are not used (enforcing that the functions are permutations), but in G1 we use them and thus duplicate points can be added to $\mathcal{DE}$, $\mathcal{RE}$, DE, and RE. A *collision* is just a pair of equal strings in one of the sets. Note that for $\mathcal{DE}$ and $\mathcal{RE}$, only strings of the same length can collide.

Game G0 exactly simulates $\mathcal{E}^*$ and its inverse while G1 always returns random bits. This second statement needs to be justified for the case of a query $M \notin \mathcal{M}$ (or $C \notin \mathcal{M}$). Particularly, if the j^{th} query is to encipher $M^j \notin \mathcal{M}$, then the last s bits returned are $C_3^j = \mathsf{mix}_{\mathrm{R}}(C_6^j, M_7^j)$. Here C_6^j is uniformly selected, and by the definition of an $\epsilon(s)$-good mixing function, we have that $\mathsf{mix}_{\mathrm{R}}(C_6^j, M_7^j)$ is a permutation of C_6^j. So C_3^j inherits its distribution. The same reasoning justifies the distribution of deciphering queries $C \notin \mathcal{M}$. We can therefore replace the oracles $\mathcal{A}$ queries with the two described games and apply the fundamental lemma of game playing [5] to get

$$\mathbf{Adv}_{\mathcal{E}^*}^{\pm\mathrm{prf}}(\mathcal{A}) = \Pr\big[\mathcal{A}^{\mathrm{G0}} \Rightarrow 1\big] - \Pr\big[\mathcal{A}^{\mathrm{G1}} \Rightarrow 1\big] \le \Pr\big[\mathcal{A}^{\mathrm{G1}} \text{ sets } \mathit{bad}\big]\,. \tag{1}$$

The following lemma captures the bound on the ability of $\mathcal{A}$ to set *bad*.

Lemma 2. $\Pr\big[\mathcal{A}^{\mathrm{G1}} \text{ sets } \mathit{bad}\big] \le 5q^2\,\epsilon(s)/2^{n-s} + 3q^2/2^n$ *for any* $s \in [1\,..\,n-1]$. $\square$

Combining Lemma 2 with Equation 1 implies the theorem statement, and a full proof of the lemma appears in [25]. Here we informally sketch one of the more interesting cases for proving the lemma above. In particular, we reason about the probability that $\mathcal{A}$ can set *bad* by causing a collision in the set $\mathcal{DE}$, which represents the domain of $\mathcal{E}$. Note that in the full proof in [25], we go through several game transitions before reasoning about this case—here we do it in the context of game G1 and thus end up being a bit informal. For simplicity we'll just focus on enciphering queries. Suppose that the i^{th} and j^{th} (with $i < j$) enciphering queries result in applying $\mathcal{E}$ to the same domain point. That is, if we let X^i and X^j be the bit strings added to $\mathcal{DE}$ during queries i and j, then a collision in $\mathcal{DE}$ occurs if $X^i = X^j$ and $l \equiv |X^i| = |X^j|$. If such a collision occurs with high probability, then $\mathcal{A}$ would be able to distinguish easily. There are two main cases to consider, based on the lengths of the queried messages M^i and M^j. The cases are marked by triangles.

$\triangleright$ Suppose that $|M^i| \notin \mathcal{M}$ and $|M^j| \notin \mathcal{M}$. Then the two domain points are $X^i = M_1^i \,||\, \mathtt{flip1}(M_4^i) \,||\, M_6^i$ and $X^j = M_1^j \,||\, \mathtt{flip1}(M_4^j) \,||\, M_6^j$. Let $s^i = |M_6^i|$ and $s^j = |M_6^j|$ (necessarily we have that $|M^i| - s^i = |M^j| - s^j$). We break down the analysis into two subcases:

- If $M_2^i \neq M_2^j$ then $M_4^j \,||\, M_5^j$ are selected uniformly and independently from any random choices made during query i. We have then that $\texttt{flip1}(M_4^j)$ consists of $n - s^j$ randomly selected bits, which will collide with the appropriate $n - s^j$ bits of X^i with probability 2^{-n+s}. Furthermore, $M_6^j = \mathsf{mix}_{\text{L}}(M_5^j, M_3^j)$ where M_5^j is a string of s^j random bits. We can apply the definition of an $\epsilon(s)$-good mixing function to get that the probability that M_6^j collides with some other value is at most $\epsilon(s)$. Combining the two probabilities, we see that the probability that $X^i = M_1^j \,||\, \texttt{flip1}(M_4^j) \,||\, M_6^j$ is at most $\epsilon(s)/2^{n-s}$.
- If $M_2^i = M_2^j$ then we can show that the probability that $X^i = X^j$ is zero. First consider if $s \equiv s^i = s^j$. Then we have that $M_4^i = M_4^j$ and $M_5^i = M_5^j$. For a collision to occur it must be that $M_1^i = M_1^j$ and $M_6^i = M_6^j$. But because of the permutivity of $\mathsf{mix}_{\text{L}}(A, \cdot)$, as given by Definition 1 part 1, this last equivalence implies that $M_3^i = M_3^j$. In turn this means that $M^i = M^j$, which would make query j pointless. But since we disallow $\mathcal{A}$ from making pointless queries, we have a contradiction. Second consider, without loss of generality, that $s^i < s^j$. Then we have that X^i can not equal X^j (recall that $|X^i| = |X^j| = l$) because flip1 ensures that $X^i\left[l - s^j\right] \neq X^j\left[l - s^j\right]$. Thus, in either situation, the probability of a collision is zero.

$\triangleright$ Now suppose that $|M^i| \in \mathcal{M}$ and $|M^j| \notin \mathcal{M}$. The two domain points are M^i and $X^j = M_1^j \,||\, \texttt{flip1}(M_4^j) \,||\, M_6^j$. Let $s^j = |M_6^j|$. We have that M_4^j and M_5^j are uniformly selected after the adversary has specified M^i (since $i < j$). Thus, $\texttt{flip1}(M_4^j)$ will collide with the appropriate bits of M^i with probability 2^{-n+s}. Since $M_6^j = \mathsf{mix}_{\text{L}}(M_5^j, M_3^j)$ we can apply the definition of a good mixing function. This gives that the probability of M_6^j colliding with the appropriate s^j bits of M^i is at most $\epsilon(s)$. Therefore the probability that $M^i = X^j$ is at most $\epsilon(s)/2^{n-s}$.

If on the other hand $|M^i| \notin \mathcal{M}$ while $|M^j| \in \mathcal{M}$, then we can only apply similar reasoning if we show that $\mathcal{A}$ learns nothing about certain random choices made in the course of answering query i. We do just that rigorously in the full proof.

So in the cases above the probability of a collision is no greater than $\epsilon(s)/2^{n-s}$, where $s \in [1 \mathinner{\ldotp\ldotp} n-1]$. Because each query adds one string to $\mathcal{DE}$, we have that $|\mathcal{DE}| = q$. Thus, the total probability of *bad* being set due to a collision in the domain of $\mathcal{E}$ is at most

$$\binom{q}{2} \frac{\epsilon(s)}{2^{n-s}} \leq \frac{q^2\,\epsilon(s)}{2^{n-s+1}}\,.$$

Combining this (via a union bound) with analyses of the other ways in which *bad* can be set yields a sketch of the lemma. ∎

The next theorem captures the security of XLS in a complexity-theoretic setting. It's proof is by a standard hybrid argument, which utilizes as one step Theorem 1. See the full version for details [25].

Theorem 2. *Fix n and an $\epsilon(s)$-good mixing function* mix. *Let $\mathcal{E}\colon \mathcal{K}_{\mathcal{E}} \times \mathcal{M} \to \mathcal{M}$ be a cipher and $E\colon \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Let $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mix}, \mathcal{E}, E]$ and let $\mathcal{A}$ be an adversary that runs in time t and asks at most q queries, each of at most μ bits. Then there exists adversaries $\mathcal{B}$ and $\mathcal{C}$ and an absolute constant c such that $\mathbf{Adv}^{\pm\mathrm{prp}}_{\mathcal{E}^*}(\mathcal{A}) \le \mathbf{Adv}^{\pm\mathrm{prp}}_{\mathcal{E}}(\mathcal{B}) + \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\mathcal{C}) + 5q^2\,\epsilon(s)/2^{n-s} + 4q^2/2^n$ for any $s \in [1\,..\,n-1]$, and so, by Lemma 1*

$$\mathbf{Adv}^{\pm\mathrm{prp}}_{\mathcal{E}^*}(\mathcal{A}) \le \mathbf{Adv}^{\pm\mathrm{prp}}_{\mathcal{E}}(\mathcal{B}) + \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\mathcal{C}) + \frac{9q^2}{2^n}$$

when $\mathsf{mix} = \mathsf{mix1}$, and the same expression, but with the 9 replaced by 14, when $\mathsf{mix} = \mathsf{mix2}$. Here $\mathcal{B}$ runs in time $t_B = t + c\mu q \log q$ and asks $q_B = q$ queries, each of length at most μ, and $\mathcal{C}$ runs in time $t_{\mathcal{C}} \le t + (q+1)\cdot \mathsf{Time}_{\mathcal{E}}(\mu) + c\mu q$ and asks $q_{\mathcal{C}} \le 2q$ queries, each of length at most n. $\square$

7 Supporting Tweaks

A *tweakable cipher* [15, 16] is a function $\widetilde{\mathcal{E}}\colon \mathcal{K}_{\mathcal{E}} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}_{\mathcal{E}} \neq \emptyset$ is the key space, $\mathcal{T} \neq \emptyset$ is the tweak space, and $\mathcal{M}$ is the message space. We require that $\widetilde{\mathcal{E}}^T_K(\cdot)$ is a length-preserving permutation for all $K \in \mathcal{K}_{\mathcal{E}}$ and $T \in \mathcal{T}$. We write the inverse of $\widetilde{\mathcal{E}}$ as $\widetilde{\mathcal{D}}$. A tweakable blockcipher is a tweakable cipher with $\mathcal{M} = \{0,1\}^n$ for some fixed n. Tweakable ciphers are useful tools for building higher-level protocols. The tweak of a cipher can be used as, for example, a sector index.

The security of a tweakable cipher is based on indistinguishability of the scheme and a tweakable random permutation. More formally, we define the following advantages

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\widetilde{\mathcal{E}}}(\mathcal{A}) = \Pr\Big[K \xleftarrow{\$} \mathcal{K}_{\mathcal{E}} : \mathcal{A}^{\widetilde{\mathcal{E}}_K, \widetilde{\mathcal{D}}_K} \Rightarrow 1\Big] - \Pr\Big[\widetilde{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : \mathcal{A}^{\widetilde{\pi}, \widetilde{\pi}^{-1}} \Rightarrow 1\Big]$$
$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prf}}}_{\widetilde{\mathcal{E}}}(\mathcal{A}) = \Pr\Big[K \xleftarrow{\$} \mathcal{K}_{\mathcal{E}} : \mathcal{A}^{\widetilde{\mathcal{E}}_K, \widetilde{\mathcal{D}}_K} \Rightarrow 1\Big] - \Pr\Big[\rho, \sigma \xleftarrow{\$} \mathrm{Func}^{\mathcal{T}}(\mathcal{M}) : \mathcal{A}^{\rho, \sigma} \Rightarrow 1\Big]$$

where the probabilities are over the choice of K or $\widetilde{\pi}$ (resp. ρ, σ) and the coins used by $\mathcal{A}$. Here $\mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ is the set of all functions $\widetilde{\pi}\colon \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\widetilde{\pi}(T, \cdot)$ is a length-preserving permutation and $\mathrm{Func}^{\mathcal{T}}(\mathcal{M})$ is the set of all functions $\rho\colon \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\rho(T, \cdot)$ is length-preserving.

XLS AND TWEAKS. The XLS construction works for tweaks. By this we mean that if the cipher $\mathcal{E}$ utilized in XLS is tweakable, then the resulting cipher with the enlarged message space is also tweakable. More specifically, fix an $\epsilon(s)$-good mixing function. If we construct $\widetilde{\mathcal{E}}^* = \mathrm{XLS}[\mathsf{mix}, \widetilde{\mathcal{E}}, E]$ where $\widetilde{\mathcal{E}}\colon \mathcal{K}_{\mathcal{E}} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ is a tweakable cipher and $E\colon \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ is a conventional blockcipher then the resulting scheme $\widetilde{\mathcal{E}}^*$ is tweakable with tweak space $\mathcal{T}$ and enlarged message space $\mathcal{M}^* = \mathcal{M} \,||\, \{0,1\}^{<n}$. Here XLS is implicitly changed by adding the tweak T as a superscript to $\mathcal{E}$ on lines 00 and 06 of Fig. 1. The following theorem statements, which are analogous to Theorem 1 and Theorem 2, establish the security of XLS with tweaks.

Theorem 3. *Fix n and an $\epsilon(s)$-good mixing function* mix. *Let $\mathcal{M} \subset \{0,1\}^{\geq n}$, let $\mathcal{T}$ be a nonempty set, and let $\widetilde{\mathcal{E}}^* = \mathrm{XLS}[\mathsf{mix}, \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}), \mathrm{Perm}(n)]$. Then for any adversary $\mathcal{A}$ that asks at most q queries we have that $\mathbf{Adv}^{\pm\widetilde{\mathrm{prf}}}_{\widetilde{\mathcal{E}}^*}(A) \leq 5q^2\,\epsilon(s)/2^{n-s} + 3q^2/2^n$ for any $s \in [1 \mathinner{..} n-1]$, and so, by Lemma 1*

$$\mathbf{Adv}^{\pm\mathrm{prf}}_{\mathcal{E}^*}(A) \leq \frac{8q^2}{2^n} \quad \text{and} \quad \mathbf{Adv}^{\pm\mathrm{prf}}_{\mathcal{E}^*}(A) \leq \frac{13q^2}{2^n}$$

for $\mathsf{mix} = \mathsf{mix1}$ *and* $\mathsf{mix} = \mathsf{mix2}$*, respectively.* $\square$

To prove this theorem, we can adjust the proof of Theorem 1 as follows. Replace $\mathcal{E}$ with $\widetilde{\mathcal{E}}$ throughout. Modify games G0 and G1 so enciphering and deciphering queries take a tweak, and lazily build separate random length-preserving functions $\widetilde{\mathcal{E}}$ for each tweak queried (in G0 they are permutations while in G1 they are just functions). Lemma 2, and its proof, can be modified in the natural way. See [25] for details. Combining Theorem 1 with a standard hybrid argument proves the next theorem.

Theorem 4. *Fix n and an $\epsilon(s)$-good mixing function* mix. *Let $\widetilde{\mathcal{E}}\colon \mathcal{K}_{\mathcal{E}} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be a tweakable cipher and $E\colon \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Let $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mix}, \widetilde{\mathcal{E}}, E]$ and let $\mathcal{A}$ be an adversary that runs in time t and asks at most q queries, with maximal query length being μ bits Then there exists adversaries $\mathcal{B}$ and $\mathcal{C}$ and an absolute constant c such that $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\widetilde{\mathcal{E}}^*}(\mathcal{A}) \leq \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\widetilde{\mathcal{E}}}(\mathcal{B}) + \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\mathcal{C}) + 5q^2\,\epsilon(s)/2^{n-s} + 4q^2/2^n$ where $s \in [0 \mathinner{..} n-1]$, and so, by Lemma 1*

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}^*}(\mathcal{A}) \leq \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\widetilde{\mathcal{E}}}(\mathcal{B}) + \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(\mathcal{C}) + \frac{9q^2}{2^n}$$

when $\mathsf{mix} = \mathsf{mix1}$*, and the same expression, but with the 9 replaced by 14, when* $\mathsf{mix} = \mathsf{mix2}$*. Here $\mathcal{B}$ runs in time $t_B = t + c\mu q \log q$ and asks $q_B = q$ queries, none longer than μ, and where $\mathcal{C}$ runs in time $t_{\mathcal{C}} \leq t + (q+1) \cdot \mathsf{Time}_{\widetilde{\mathcal{E}}}(\mu) + c\mu q$ and asks $q_{\mathcal{C}} \leq 2q$ queries, each of length at most n.* $\square$

Note that the concrete security bounds are the same in both the tweaked and untweaked settings. Intuitively this is because having a tweakable family of permutations is a stronger primitive than a normal family of permutations, and the adversary might as well just focus its attack on a single tweak.

8 XLS with Ordinary PRPs

If we apply the XLS construction to cipher $\widetilde{\mathcal{E}}$ and blockcipher E that are both secure as (ordinary) pseudorandom permutations (i.e., adversaries are restricted to chosen-plaintext attacks), then the resulting cipher with expanded message space is also secure as a PRP. More formally, we define the following advantage

$$\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{\mathcal{E}}}(\mathcal{A}) = \Pr\Big[K \xleftarrow{\$} \mathcal{K}_{\mathcal{E}} : \mathcal{A}^{\widetilde{\mathcal{E}}_K} \Rightarrow 1\Big] - \Pr\Big[\widetilde{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}) : \mathcal{A}^{\widetilde{\pi}} \Rightarrow 1\Big]$$

where the probability is over the random choice of K or π and the random coins used by $\mathcal{A}$. While the theorems from Section 7 do not imply the security of XLS using ordinary (tweaked) PRPs, their proofs can be (simplified) in a straightforward manner to derive the PRP security of XLS as captured by the next theorem statement.

Theorem 5. *Fix n and an $\epsilon(s)$-good mixing function* mix. *Let $\widetilde{\mathcal{E}}\colon \mathcal{K}_{\mathcal{E}} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be a tweakable cipher and $E\colon \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Let $\mathcal{E}^* = \mathrm{XLS}[\mathsf{mix}, \mathcal{E}, E]$ and let $\mathcal{A}$ be an adversary that runs in time t and asks at most q queries, with maximal query length being μ bits Then there exists adversaries $\mathcal{B}$ and $\mathcal{C}$ and an absolute constant c such that $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{\mathcal{E}}^*}(\mathcal{A}) \le \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{\mathcal{E}}}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{C}) + 5q^2\,\epsilon(s)/2^{n-s} + 4q^2/2^n$ where $s \in [0\,..\,n-1]$, and so, by Lemma 1*

$$\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\mathcal{E}^*}(\mathcal{A}) \le \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\mathcal{E}}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{C}) + \frac{9q^2}{2^n}$$

when $\mathsf{mix} = \mathsf{mix1}$, *and the same expression, but with the* 9 *replaced by* 14, *when* $\mathsf{mix} = \mathsf{mix2}$. *Here $\mathcal{B}$ runs in time $t_B = t + c\mu q \log q$ and asks $q_B = q$ queries, none longer than μ, and where $\mathcal{C}$ runs in time $t_{\mathcal{C}} \le t + (q+1) \cdot \mathsf{Time}_{\widetilde{\mathcal{E}}}(\mu) + c\mu q$ and asks $q_{\mathcal{C}} \le 2q$ queries, each of length at most n.* $\square$

An immediate corollary of the theorem is security of XLS for ordinary, untweaked PRPs (just set the tweak space $\mathcal{T}$ to a single value).

Acknowledgments

We thank Mihir Bellare, Jesse Walker, and the anonymous referees. This work was supported in part by NSF grants CCR-0208842, CNS-0524765, and a gift from Intel Corp; thanks to the NSF and Intel for their kind support.

References

1. J. An and M. Bellare. Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. *Advances in Cryptology – CRYPTO 1999*, LNCS vol. 1666, Springer, pp. 252–269, 1999.
2. R. Anderson and E. Biham. Two practical and provably secure block ciphers: BEAR and LION. *Fast Software Encryption 1996*, LNCS vol. 1039, Springer, pp. 113–120, 1996.
3. M. Bellare, and P. Rogaway. On the construction of variable-input-length ciphers. *Fast Software Encryption 1999*, LNCS vol. 1636, Springer, pp. 231–244, 1999.
4. M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. *Advances in Cryptology – ASIACRYPT 2000*, LNCS vol. 1976, Springer, pp. 317–330, 2000.
5. M. Bellare, and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. *Advances in Cryptology – EUROCRYPT 2006*, LNCS vol. 4004, Springer, pp. 409–426, 2006.

6. J. Black and P. Rogaway. Ciphers with arbitrary finite domains. *Topics in Cryptology – CT-RSA 2002*, LNCS vol. 2271, Springer, pp. 114–130, 2002.
7. A. Cayley. On Latin squares. *Oxford Cambridge Dublin Messenger Math.*, vol. 19, pp. 135–137, 1890.
8. D. Chakraborty and P. Sarkar. A new mode of encryption providing a strong tweakable pseudo-random permutation. *Fast Software Encryption – FSE 2006*, LNCS vol. 4047, Springer, pp. 293–309, 2006.
9. D. Chakraborty and P. Sarkar. HCH: A new tweakable enciphering scheme using the Hash-Encrypt-Hash approach. *Advances in Cryptology – INDOCRYPT 2006*, LNCS vol. 4329, Springer, pp. 287–302, 2006.
10. D. Cook, M. Yung, and A. Keromytis. Elastic AES. Cryptology ePrint archive, report 2004/141, 2004.
11. D. Cook, M. Yung, and A. Keromytis. Elastic block ciphers. Cryptology ePrint archive, report 2004/128, 2004.
12. S. Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. *Advances in Cryptology – INDOCRYPT 2004*, LNCS vol. 3348, Springer, pp. 315–327, 2004.
13. S. Halevi. TET: A wide-block tweakable mode based on Naor-Reingold. Cryptology ePrint archive, report 20007/14, 2007.
14. S. Halevi and P. Rogaway. A parallelizable enciphering mode. *Topics in Cryptology – CT-RSA 2004*, LNCS vol. 2964, Springer, pp. 292–304, 2004.
15. S. Halevi and P. Rogaway. A tweakable enciphering mode. *Advances in Cryptology – CRYPTO 2003*, LNCS vol. 2729, Springer, pp. 482–499, 2003.
16. M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. *Advances in Cryptology – CRYPTO 2002*, LNCS vol. 2442, Springer, pp. 31–46, 2002.
17. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal of Computing*, vol. 17, no. 2, pp. 373–386, 1988.
18. S. Lucks. BEAST: A fast block cipher for arbitrary blocksizes. *Communications and Multimedia Security*, IFIP vol. 70, Chapman & Hill, pp. 144–153, 1996.
19. S. Lucks. Faster Luby-Rackoff ciphers. *Fast Software Encryption 1996*, LNCS vol. 1039, Springer, pp. 189–203, 1996.
20. D. McGrew and S. Fluhrer. The extended codebook (XCB) mode of operation. Cryptology ePrint archive, report 2004/278, 2004.
21. C. Meyer and M. Matyas. *Cryptography: A New Dimension in Data Security*. John Wiley & Sons, New York, 1982.
22. M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, vol. 12, no. 1, pp. 29–66, 1999.
23. J. Patarin. How to construct pseudorandom and super pseudorandom permutations from one single pseudorandom function. *Advances in Cryptology – EUROCRYPT 1992*, LNCS vol. 658, pp. 256–266, 1992.
24. S. Patel, Z. Ramzan, and G. Sundaram. Efficient constructions of variable-input-length block ciphers. *Selected Areas in Cryptography 2004*, LNCS vol. 3357, pp. 326–340, 2004.
25. T. Ristenpart, and P. Rogaway. How to Enrich the Message Space of a Cipher (full version of this paper). http://www.cse.ucsd.edu/users/tristenp/
26. B. Schneier and J. Kelsey. Unbalanced Feistel networks and block cipher design. *Fast Software Encryption 1996*, LNCS vol. 1039, Springer, pp. 121–144, 1996.
27. C. Schnorr and S. Vaudenay. Black box cryptanalysis of hash networks based on multipermutations. *Advances in Cryptology – EUROCRYPT 1994*, LNCS vol. 950,

Springer, pp. 47–57, 1995.
28. R. Schroeppel. Hasty pudding cipher specification. *First AES Candidate Workshop*, 1998.
29. S. Vaudenay. On the need for multipermutations: cryptanalysis of MD4 and SAFER. *Fast Software Encryption 1994*, LNCS vol. 1008, Springer, pp. 286–297, 1995.
30. P. Wang, D. Feng, and W. Wu. HCTR: a variable-input-length enciphering mode. *Information Security and Cryptography, CISC 2005*, LNCS vol. 3822, Springer, pp. 175–188, 2005.
31. Y. Zheng, T. Matsumoto, and H. Imai. On the construction of block ciphers provably secure and not relying on any unproved hypotheses. *Advances in Cryptology – CRYPTO 1989*, LNCS vol. 435, Springer, pp. 461–480, 1990.