Sometimes-Recurse Shuffle

Almost-Random Permutations in Logarithmic Expected Time

Ben Morris[1] Phillip Rogaway[2]

[1] Dept. of Mathematics, University of California, Davis, USA
[2] Dept. of Computer Science, University of California, Davis, USA

August 17, 2013

Abstract. We describe a security-preserving construction of a random permutation of domain size N from a random function, the construction tolerating adversaries asking all N plaintexts, yet employing just $\Theta(\lg N)$ calls, on average, to the one-bit-output random function. The approach is based on card shuffling. The basic idea is to use the *sometimes-recurse* transformation: lightly shuffle the deck (with some other shuffle), cut the deck, and then recursively shuffle one of the two halves. Our work builds on a recent paper of Ristenpart and Yilek.

Keywords: Card shuffling, format-preserving encryption, PRF-to-PRP conversion, mix-and-cut shuffle, pseudorandom permutations, sometimes-recurse shuffle, swap-or-not shuffle.

1 Introduction

Format-preserving encryption. Suppose you are given a blockcipher, say AES, and want to use it to efficiently construct a cipher on a smaller domain, say the set of $N = 10^{16}$ sixteen-digit credit card numbers. You could, for example, use AES as the round function for several rounds of a Feistel network, the approach taken by emerging standards [1, 6]. But information-theoretic security will vanish by the time the adversary asks $\sqrt{N}$ queries, which is a problem on these small-sized domains.[3] Or you could precompute a random permutation on N points, but spending $\Omega(N)$ time in computation will become undesirable before $\sqrt{N}$ adversarial queries becomes infeasible.

This paper provides a new solution to this problem of *format-preserving encryption*, where we aim to build ciphers with an arbitrary finite domain [3–5, 7], frequently $[N] = \{0, 1, \ldots, N-1\}$ for some N. Our solution lets you encipher a sixteen-digit credit card with about 1000 expected AES calls,[4] getting an essentially ideal provable-security claim. In particular, the adversary can ask any number of queries—including all N of them—and its advantage in distinguishing the constructed cipher from a random permutation will be insignificantly more than its ability to break the underlying primitive (in our example, AES) with a like number of queries.

Cast in more general language, then, this paper is about constructing ciphers (information theoretic or complexity theoretic PRPs) on an arbitrary domain $[N]$, starting with a PRF.[5] As in other recent work [8, 10, 13], our ideas are motivated by card shuffling and its cryptographic interpretation. This connection was observed by Naor [14, p. 62], [16, p. 17], who explained that

[3] It is a problem from the point of view of having a satisfying, information-theoretic provable-security claim. Likely it is not a problem with the actual, complexity-theoretic security of the construction.
[4] This comes to about 80K clock cycles, or 25 μsec, on a recent Intel processor.
[5] If starting from AES, only a single bit of each 128-bit output will be used. A random permutation on 128 bits that gets truncated to a single bit is extremely close to a random function [2].

when a card shuffle is *oblivious*—when you can trace the trajectory of a card without attending to the trajectories of *other* cards in the deck—then it may be useful as a cipher. We will move back and forth between the language of encryption and that of card shuffling: a PRP/cipher is a shuffle; a plaintext x encrypts to ciphertext y if the card initially at position x ends up at position y; the PRP's key is the randomness underlying the shuffle.

Ristenpart and Yilek paper. Our work follows on the heels of an upcoming Crypto paper [15] by Tom Ristenpart and Scott Yilek. That paper describes the following approach for turning one card shuffle into another. Assume that we want to shuffle $N = 2^n$ cards. Then the *Icicle* construction first mixes the cards using some given (we'll call it the *inner*) shuffle. Then we cut the deck into two piles and recursively shuffle each. Ristenpart and Yilek explain that if the inner shuffle is a good *pseudorandom separator* (PRS), then the constructed shuffle mixes all N cards well, a condition they term *full* security. A shuffle is a good PRS (a 2-PRS in the original terminology) if, after shuffling, the (unordered) *set* of cards ending up in each of the two piles is indistinguishable from a uniform partitioning of the cards into two equal-sized sets.[6]

Ristenpart and Yilek apply the Icicle construction to the swap-or-not shuffle[7] of Hoang, Morris, and Rogaway [10], a combination they call *mix-and-cut*.[8] The shuffle achieves full security in $\Theta(\lg^2 N)$ rounds. Replacing the random function of swap-or-not with a single-bit-output PRF, say AES, one gets a cipher on N points achieving full security with $\Theta(\lg^2 N)$ AES calls.[9, 10] While full security is directly achieved by some other oblivious shuffles [8, 9, 12], mix-and-cut would seem to be faster.

Contributions. We begin by reconceptualizing what is going on in mix-and-cut. Instead of thinking of the underlying transformation as turning a PRS into a PRP, we think of it as turning a mediocre PRP into a better one. If the inner shuffle is good enough to mix half the cards—in the inverse shuffle, any $N/2$ cards end up in almost uniform positions—then the constructed shuffle will achieve full security.

After this shift in viewpoint, we make a simple change to mix-and-cut that dramatically improves its expected running time. As before, one begins by applying the inner shuffle to the N cards. Then one splits the deck and recursively shuffles *one* (rather than both) of the two halves. Using swap-or-not (SN) for the inner shuffle we now get a PRP over $[N]$ enjoying full security and computable in $\Theta(\lg N)$ expected time.[11] We call the SN-based construction SR, for *sometimes recurse*. The underlying transformation we call **SR**.

[6] Said differently, the first bit of a PRS on $\{0,1\}^n$ should be indistinguishable from a random regular function from $\{0,1\}^n$ to $\{0,1\}$.

[7] In the binary-string setting ($N = 2^n$), a round of swap-or-not is: for a random string $K_i \in \{0,1\}^n$, replace X by $K_i \oplus X$ iff $F(i, \hat{X}) = 1$, where F is a random function to bits and $\hat{X} = \max(X, X \oplus K_i)$. The final value of input X is its image after shuffling. For an arbitrary domain $[N]$, a round is: replace X by $K_i - X$ iff $F(i, \hat{X}) = 1$, where $\hat{X} = \max(X, K_i - X)$. Subtraction is in a group of size N, say $\mathbb{Z}_N$, and K_i is selected uniformly from this group.

[8] We will use the name *mix-and-cut* somewhat more generically, as both the name for the SN-based construction and as the transformation that essentially *is* with Icicle, but which differently conceptualizes the requirements on the inner shuffle.

[9] Time bounds in this paragraph ignore dependency on the error bound, ε, which measures how close the constructed object is to a uniform permutation. The bounds apply for any constant ε.

[10] In speaking of time we assume a model of computation with unit-time evaluation of the underlying PRF. This is realistic in the setting of "small domain" FPE, where each PRF is likely to be instantiated with a blockcipher call.

[11] The expectation is over the input. In the case of binary-string inputs, slow-to-compute plaintexts, under our conventions, will be those whose ciphertexts begin with many leading 0-bits.

Our definitions and results apply to arbitrary N (it need not be a power of two). We emphasize that the adversary may query *all* points in the domain. We give numerical examples to illustrate that the improvement over mix-and-cut is large. We also explain why, with SR, having the running time depend on the key and plaintext does *not* give rise to side-channel attacks. Finally, we explain how to cheaply tweak [11] the construction, degrading neither the run-time nor the security bound compared to the untweaked counterpart.[12]

Additional related work. The work of Ristenpart and Yilek [15] built on earlier work of Granboulan and Pornin [8]. The latter contains a shuffle that can now be seen as the application of the Icicle construction to a particular PRS. But the chosen PRS is expensive to realize, involving extensive use of arbitrary-precision floating-point arithmetic to do approximate sampling from a hypergeometric distribution. Both mix-and-cut and sometimes-recurse are far more practical.

For realistic domain sizes N, both mix-and-cut and sometimes-recurse are also much faster than the method of Stefanov and Shi [17], which spends $\tilde{\Theta}(N)$ time to preprocess the key into a table of size $\tilde{\Theta}(\sqrt{N})$ that supports $\tilde{\Theta}(\sqrt{N})$-time evaluation of the constructed cipher.

2 Mix-and-Cut Shuffle

This section reviews and reframes the prior work of Ristenpart and Yilek [15]; the new contents of our paper begins in Section 3.

The mix-and-cut transformation can be described recursively as follows. Assume we want to shuffle $N = 2^n$ cards. If $N = 1$ then we are done; a single card is already shuffled. Otherwise, to mix-and-cut shuffle $N \geq 2$ cards,

1. shuffle the N cards using some other, inner shuffle; and then
2. cut the deck into two halves and, recursively, shuffle each half.

The method can be seen as an operator, **MC**, that maps a shuffle SH on a power-of-two cards to a shuffle $\mathrm{SH}' = \mathbf{MC}[\mathrm{SH}]$ on the same number of cards. A sufficient condition for SH' to achieve full security is for SH to *lightly shuffle* the deck. Informally, to lightly shuffle the deck means that if one identifies some $N/2$ positions of the deck, then the cards that land in these positions should be nearly uniform. More formally, we say that SH *ε-lightly shuffles* if SH^{-1} sends any $N/2$ cards to something within total-variation distance ε of $N/2$ uniformly random but distinct positions. Note that if the shuffle SH is swap-or-not (SN) then it is equivalent to ask that SH itself send $N/2$ cards to something ε-close to inform, as SN is identical in its forward and backward direction, up to the naming of keys.

Let's consider the speed of **MC** with SN as the underlying shuffle, a combination we'll write as $\mathrm{MC} = \mathbf{MC}[\mathrm{SN}]$. First some preliminaries. For a round-parameterized shuffle SH that approaches the uniform distribution, let $\tau_q^r(N)$ be the induced distribution after r rounds on some q distinct cards $(x_1, \ldots, x_q) \in Z_N^q$ from a deck of size N, and let $\pi_q(N)$ be the uniform distribution on q distinct points from $[N]$. Let $\Delta_{\mathrm{SH}}(N, q, r) = \|\tau_q^r(N) - \pi_q(N)\|$ be the statistical distance between these two distributions. Hoang, Morris, and Rogaway show that, for the swap-or-not shuffle, SN,

$$\Delta_{\mathrm{SN}}(N, q, r) \;\leq\; \frac{2N^{3/2}}{r+2}\left(\frac{q+N}{2N}\right)^{r/2+1} \;=\; \Delta_{\mathrm{SN}}^{\mathrm{ub}}(N, q, r)\,. \tag{1}$$

[12] Ristenpart and Yilek likewise support tweaks [15], but their quantitative bounds give up more, and each round key needs to depend on the tweak. Our improvements in this direction apply just as well to mix-and-cut.

Assuming even N, setting $q = N/2$ in this equation gives

$$\Delta_{\mathrm{SN}}(N, N/2, r) \le N^{3/2} \left(\frac{3}{4}\right)^{r/2}$$

and so $\Delta_{\mathrm{SN}}(N, N/2, r) \le \varepsilon$ if

$$\frac{3}{2} \lg N + \frac{r}{2} \lg(3/4) \le \lg \varepsilon,$$

which occurs if

$$\begin{aligned} r &\ge \frac{\lg \varepsilon - (3/2) \lg N}{(1/2) \lg(3/4)} \\ &\ge 7.23 \lg N - 4.82 \lg \varepsilon \\ &\in \Theta(\lg N - \lg \varepsilon)\ . \end{aligned} \tag{2}$$

For a round-based shuffle SH monotonically approaching the uniform distribution, let $T_{\mathrm{SH}}(N, q, \varepsilon)$ be the minimum number r such that $\Delta_{\mathrm{SH}}(N, q, r) \le \varepsilon$. Let $T_{\mathrm{SH}}(N, \varepsilon) = T_{\mathrm{SH}}(N, N, \varepsilon)$ be the time to mix all the cards to within ε. For MC = $\mathbf{MC}$[SN] to mix all $N = 2^n$ cards to within ε it will suffice if we arrange that each invocation of SN mixes half the cards to within ε/n. Assuming this strategy, the total number of needed rounds will be

$$\begin{aligned} T_{\mathrm{MC}}(2^n, \varepsilon) &\le \sum_{\ell=1}^{n} T_{\mathrm{SN}}(2^\ell, 2^{\ell-1}, \varepsilon/n) \\ &\le \sum_{\ell=1}^{n} \left(7.23\,\ell - 4.82\ \lg(\varepsilon/n)\right) \qquad \text{(from (2))} \\ &\le 14.46\, n^2 + 4.82\, n \lg n - 4.82\, n \lg \varepsilon \\ &\in \Theta(\lg^2 N - \lg N \lg \varepsilon) \end{aligned}$$

Interpreting, the MC construction can encipher n-bit strings, getting to within any fixed statistical distance ε of uniform, by using $\Theta(n)$ stages of $\Theta(n)$ rounds, so $\Theta(n^2)$ total rounds. The round functions here are assumed uniform and independent. Replacing them by a complexity-theoretic PRF, we are converting a PRF into a PRP on domain $\{0,1\}^n$ with $\Theta(n^2)$ calls, achieving tight provable security and no limit on the number of adversarial queries.

3 Sometimes-Recurse Shuffle

Our main observation is this: after lightly shuffling the deck and cutting it in half, there is no need to recurse on one of the two halves; either pile can be declared finished, recursing only on the other. The reason is that once $N/2$ of the cards are distributed to within ε of uniform, so too is the joint distribution of this half of the cards and the unordered *set* of remaining cards. If the latter set of cards is shuffled to within δ of uniform then the entire sequence of N cards will be shuffled to within $\varepsilon + \delta$ of uniform. Shuffling the unfinished set of cards by starting at whatever configuration they were left in will do no harm, as we will shuffle the cards to within distance δ of uniform starting from *any* initial ordering.

Power-of-two domains. The *sometimes-recurse* (**SR**) transform can thus be described as follows. Assume for now that want to shuffle $N = 2^n$ cards (we will generalize afterward). If $N = 1$ then we are done; a single card is already shuffled. Otherwise, to **SR** shuffle $N \geq 2$ cards,

1. shuffle the N cards using some other, inner shuffle; and then
2. cut the deck into two halves and, recursively, shuffle the first half.[13]

The method can be seen as an operator, **SR**, that maps a shuffle SH on any power-of-two cards to a shuffle $\mathrm{SH}' = \mathbf{SR}[\mathrm{SH}]$ on any power-of-two cards.

Recasting the method into more cryptographic language, we are given a (variable-input-length) PRP $E\colon \mathcal{K} \times \{0,1\}^* \to \{0,1\}^*$. Each $E_K(\cdot)$ is a length-preserving permutation. (We will routinely write some or all arguments as subscripts, as in this example.) We construct from E a PRP $E' = \mathbf{SR}[E]$ as follows. First, assert that $E'_K(\Lambda) = \Lambda$, where Λ is the empty string. Otherwise, let $E'_K(X) = Y$ if $Y = E_K(X) = 1 \,\|\, Y'$ begins with a 1-bit, and let $E'_K(X) = 0 \,\|\, E_K(Y')$ if $Y = E_K(X) = 0 \,\|\, Y'$ begins with a 0-bit.

The **SR** transformation. The description above assumes a power-of-two number of cards and an even cut of the deck. The first assumption runs contrary to our intended applications, and dropping this assumption necessitates dropping the second assumption as well. Here then is the **SR** transform stated more broadly. Assume an inner shuffle, SH, that can mix an arbitrary number of cards. Let $p\colon \mathbb{N} \to \mathbb{N}$, the *split*, be a function with $1 \leq p(N) < N$. We'll write p_N for $p(N)$. We construct a shuffle $\mathrm{SH}' = \mathbf{SR}_p[\mathrm{SH}]$. Namely, if $N = 1$, we are done; a single card is shuffled. Otherwise,

1. shuffle the N cards using the inner shuffle, SH; and then
2. cut the deck into a first pile having p_N cards and a second pile having $q_N = N - p_N$ cards. Recursively, shuffle the first pile.

Initial and generated N-values. A potential point of confusion is that, above, the name "N" effectively has two different meanings. One is the *initial* N, call it N_0, that specifies the domain $[N_0]$ on which we seek to encipher. The other meaning is as a generic name for any of the N-values that can arise in recursive calls that begin with the initial N. These are the *generated* N-values, a set of numbers $\mathcal{G}(N_0) = \mathcal{G}_p(N_0)$. We count the initial N among the generated N-values. As an example, if the initial N is $N_0 = 10^{16}$, and if $p_N = \lfloor N/2 \rfloor$, then there are 54 generated N-values, which are $\mathcal{G}(10^{16}) = \{10^{16}, 10^{16}/2, 10^{16}/4, \ldots, 71, 35, 17, 8, 4, 2, 1\}$. In general, $\mathcal{G}(N_0) = \{N_0, N_1, \ldots, N_n\}$ where $N_i = p_{N_{i-1}}$ and $N_n = 1$. We call n the number of *stages*.

The transformation works. Let $q\colon \mathbb{N} \to \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \to [0,1]$ be functions, $0 \leq q(n) \leq N$. Let SH be a shuffle that can mix any number of cards. We say that SH is (q,ε)-*good* if for all $N \in \mathbb{N}$, for any distinct $y_1, \ldots, y_{q(N)} \in [N]$, the total-variation distance between $(\mathrm{SH}^{-1}(y_1), \ldots, \mathrm{SH}^{-1}(y_{q(N)}))$ and the uniform distribution on $q(N)$ distinct points from $[N]$ is at most $\varepsilon(N)$. A shuffle is ε-*good* if it is (q,ε)-good for $q(N) = N$. We have the following:

Theorem 1. *Let $p : \mathbb{N} \to \mathbb{N}$ be a function, $1 \leq p_N < N$, and let $q_N = N - p_N$. Let* SH *be an (q,ε)-good shuffle. Then $\mathbf{SR}_p[\mathrm{SH}]$ is δ-good shuffle on domain $[N_0]$ where $\delta = \sum_{N \in \mathcal{G}(N_0)} \varepsilon(N)$.*

The proof is just the repeated application of the observation in the paragraph that began Section 3.

[13] The first (or left) pile of cards are those at positions in $\{0, 1, \ldots, 2^{n-1} - 1\} = [2^{n-1}]$; the second (or right) pile of cards are those at positions in $\{2^{n-1}, 2^{n-1} + 1, \ldots, 2^n - 1]$. The convention generalizes: the first/left pile has the cards with smaller indexes.

```
10  procedure E^N_{KF}(X)                                  //invariant: X ∈ [N]
11  if N = 1 then return X                                 //a single card is already shuffled

20  for i ← 1 to t_N do                                    //SN, for t_N-rounds
21      X' ← K_i − X (mod N)                               //X' is the "partner" of X
22      X̂ ← max(X, X')                                     //canonical name for {X, X'}
23      if F(i, X̂) = 1 then X ← X'                         //maybe swap X and X'

30  if X < p_N then return E^{p_N}_{KF}(X)                 //recursively shuffle the first pile
31  if X ≥ p_N then return X                               //but second pile is done
```

Fig. 1. Construction SR = **SR**[SN]. The method enciphers on $[N_0]$ (the initial value of N), each stage (recursive invocation) employing t_N-rounds of SN (lines 20–23). The split values, p_N, are a second parameter on which SR depends. The randomness for SN is determined by $F\colon \mathbb{N} \times \mathbb{N} \to \{0,1\}$ and $K\colon \mathbb{N} \to \mathbb{N}$.

Using SN as the inner shuffle. We'll write SR (no bold) for **SR**[SN], the sometimes-recurse transformation applied to the swap-or-not shuffle. The algorithm is shown in Fig. 1, now written out in the manner of a cipher, where the trajectory of a single card X is followed. Of course $\mathrm{SN} = \mathrm{SN}_t$ depends on the round count and $\mathbf{SR} = \mathbf{SR}_p$ depends on the split, so $\mathrm{SR} = \mathrm{SR}_{t,p}$ depends on both. The canonical choice for the split p_N is $p_N = \lfloor N/2 \rfloor$; when no mention of p_N is made, this is assumed. There is no default for the round counts t_N; we must select these values with care.

We proceed to analyze SR, for the canonical split, with the help of Proposition 1 and equation (2). We aim to shuffle N cards to within a target statistical distance ε. Assume we run each stage (that is, each SN shuffle) with t_N adequate to achieve error ε/n for any half, rounded up, of the cards. When N is a power of 2, the expected total number of rounds to encipher a point will then be

$$\begin{aligned}\mathbf{E}[T_{\mathrm{SR}}(N,\varepsilon)] &\le T_{\mathrm{SN}}(N, \tfrac{N}{2}, \varepsilon/\lg N) + \frac{T_{\mathrm{SN}}(\frac{N}{2}, \frac{N}{4}, \varepsilon/\lg N)}{2} + \frac{T_{\mathrm{SN}}(\frac{N}{4}, \frac{N}{8}, \varepsilon/\lg N)}{4} + \cdots \\ &\le 2(7.23\ \lg N + 4.82 \lg\lg N - 4.82 \log \varepsilon) \qquad \text{from (2)}\end{aligned}$$

For arbitrary N (not necessarily a power of two), simply replace N by $2N$ in the equation just given to get an upper bound. This is valid because the sequence of generated N-values for N_0 are bounded above by the sequence of generated N-values for N_0' the next higher power of two, and, additionally, our bound on $T_{\mathrm{SN}}(N,\varepsilon)$ is strictly increasing. Thus, for any N,

$$\begin{aligned}\mathbf{E}[T_{\mathrm{SR}}(N,\varepsilon)] &\le 14.46 \lg N + 4.82 \lg\lg 2N - 4.82 \lg \varepsilon + 14.46 \\ &\in \Theta(\lg N - \lg \varepsilon)\end{aligned} \tag{3}$$

The worst-case number of rounds is similarly bounded. We summarize the result as follows.

Theorem 2. *For any $N \ge 1$ and $\varepsilon \in (0,1)$, the* SR *construction enciphers points on $[N]$ in $\Theta(\lg N - \lg \varepsilon)$ expected rounds and $\Theta(\lg^2 - \lg N \lg \varepsilon)$ rounds in the worst case. No adversary can distinguish the construction from a uniform permutation on $[N]$ with advantage exceeding ε. This assumes uniformly random round keys and round functions for* SN*, appropriate round counts t_N, and the canonical split.*

As a numerical example, equation (3) gives $\mathbf{E}[T_{\mathrm{SR}}(10^{16}, 10^{-10})] \le 1159$. In the next section we will do better than this—but not by much—by doing calculations directly from equation (1) and by partitioning the error ε so as to give a larger portion to earlier (that is, larger) generated N.

d	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	18	20	30
min-1	187	213	239	264	289	313	337	362	386	410	435	459	483	507	531	580	628	869
mean-1	359	412	464	514	563	612	660	710	758	806	856	904	952	1000	1048	1145	1242	1723
max-1	1110	1802	2442	3475	4411	5295	6402	7762	8885	10235	11842	13197	14790	16639	18239	22158	26069	51453
min-2	218	219	225	248	272	295	318	342	365	389	413	436	460	484	507	555	602	840
mean-2	427	436	450	496	544	590	636	684	730	778	826	872	920	968	1014	1110	1204	1680
max-2	1308	1971	2701	3968	5168	6491	7951	9918	11681	13616	16107	18313	20701	23716	26365	32745	39131	83160

Fig. 2. Speed of SR **shuffle**. Minimum, mean (rounded to nearest integer), and maximum number of rounds to SR-encipher a d-digit decimal string with error $\varepsilon \le 10^{-10}$ and round counts t_N selected by strategy 1 or strategy 2, as marked. The split is $p_N = \lfloor N/2 \rfloor$. Round-counts for MC always coincide with the max-labeled rows.

4 Parameter Optimization

Round counts. Let us continue to assume the canonical split of $p_N = \lfloor N/2 \rfloor$ and look at the optimization of round counts t_N under this assumption.

In speaking below of the number p of nontrivial stages of SR, we only count generated N-values with $N \ge 3$. This is because we will always select $t_2 = 1$, as this choice already contributes zero error, and the degenerate SR stage with $N = 1$ contributes no error and needs no t_1 value (let $t_1 = 0$). Corresponding to this convention for counting the number of nontrivial stages, we let $\mathcal{G}'(N_0) = \mathcal{G}(N_0) \setminus \{1, 2\}$ be the generated N-values when starting with N_0 but excluding $N = 1$ and $N = 2$.

Given an initial N_0 and a target ε, we consider two strategies for computing the round counts t_N for $N \in \mathcal{G}'(N_0)$. Both use the upper bound $\Delta^{\mathrm{ub}}_{\mathrm{SN}}(N, q, r) = (2N^{3/2}/(r+2)) \cdot ((q+N)/(2N))^{r/2+1}$ on $\Delta_{\mathrm{SN}}(N, q, r)$ given by equation (1).

1. *Split the error equally.* Let $n = |\mathcal{G}'(N_0)| \approx \lg N_0$ be the number of nontrivial stages. For each $N \in \mathcal{G}'(N_0)$ let t_N be smallest number r such that $\Delta^{\mathrm{ub}}_{\mathrm{SN}}(N, \lceil N/2 \rceil, r) \le \varepsilon/n$. This will result in rounds counts t_N that diminish with diminishing N, each stage contributing about the same portion to the error.
2. *Constant round count.* Let r_0 be the smallest number r where $\sum_{N \in \mathcal{G}'(N_0)} \Delta^{\mathrm{ub}}_{\mathrm{SN}}(N, \lceil N/2 \rceil, r) < \varepsilon$ and let $t_N = r_0$ for all $N \in \mathcal{G}'(N_0)$. This will result in stages that contribute a diminishing amount to the error.

The table of Fig. 2 illustrates the expected and worst-case number of rounds that result from these two strategies if we encipher on a domain of $N_0 = 10^d$ points and cap the error at $\varepsilon = 10^{-10}$. The pronounced differences between mean and max round counts (a factor exceeding 17 when $n = 16$) coincides with the saving of SR over MC. In contrast, there is only a modest difference in mean round-counts between the two round-count selection strategies.

In numerical experiments, more complex strategies for determining the round counts did not work better.

Non-equal splits. Besides the split of $p_N = \lfloor N/2 \rfloor$, we considered splits of $p_N = \lfloor \alpha N \rfloor$ for $\alpha \in (0, 1)$. For example, if the input is a decimal string then a selection of $\alpha = 0.1$ corresponds to using SN until a 90% fraction of the cards are (almost) properly distributed, at which point there would be only a 10% chance of needing to recurse. When a recursive call is made, it would be on

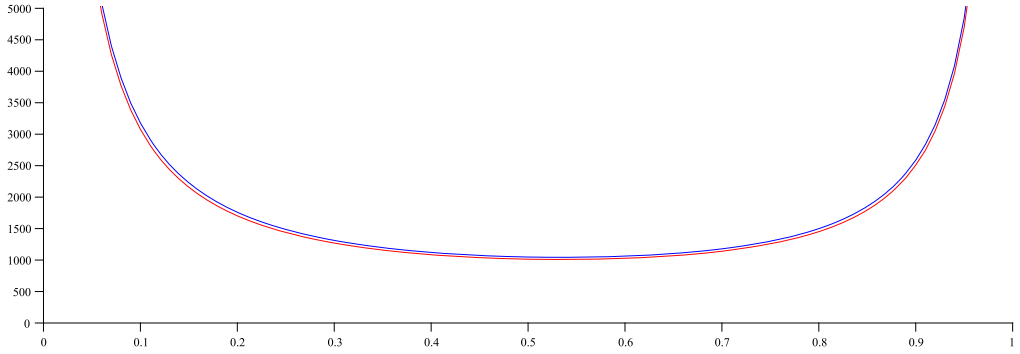

Fig. 3. Selecting the split. Expected number of rounds (the y-coordinate) to encipher $N = 10^{16}$ points using SR and a split of $p_N = \lfloor \alpha N \rfloor$ for various α (the x-axis). The total variation distance is capped at $\varepsilon = 10^{-10}$. The top (blue) curve is with round counts t_N determined by for strategy 1; the bottom (red) curve for strategy 2. In both cases the smallest expected number of rounds occurs with a non-canonical split: 1048 rounds ($\alpha = 0.5$) reduced to 1043 rounds ($\alpha = 0.53$) for strategy 1; and 1014 rounds ($\alpha = 0.5$) reduced to 1010 rounds ($\alpha = 0.52$) for strategy 2.

a string of length one less than before. But splits this uneven turn out to be inefficient; see Fig. 3. On the other hand, when the split $p_N = \lfloor \alpha N \rfloor$ has α close to 1/2, the expected number of rounds is not very sensitive to α; again see the figure. Small α make each SN stage slower, but there will be fewer of them; large α make each SN stage faster, but there will be more.

Given the similar mean round counts for strategies 1 and 2, the similar mean round counts all α near 1/2, the implementation simplicity of dividing by 2, and the better maximum rounds counts of strategy 1, the choice of strategy 1 and $\alpha = 1/2$ seems best.

5 Incorporating Tweaks

The possibly-small domain for FPE makes it important, in applications, to have the constructed cipher be *tweaked*: an additional argument T, the tweak, names the desired permutation in a family of keyed permutations [11]. In the reference experiment that defines security one asks for indistinguishability (complexity theoretic or information theoretic) from a family of tweak-indexed, uniformly random permutations, each tweak naming an independent permutation from the collection. As an example of a tweak's use, in the context of enciphering a credit card number, one might encipher only the middle six digits, using the first six and last four digits as the tweak.

The obvious way to incorporate a tweak in SR is to make the round constants K_i (line 21 of Fig. 1) depend on it, and to make the round functions $F(i, \hat{X})$ (line 23 of Fig. 1) depend on it. Note, however, that an inefficiency emerges when the former is done: if there is a large space of possible tweaks, it will no longer be possible to precompute the round constants K_i. In addition, we do not want to get a security bound that gives up a factor corresponding to the number of tweaks used, which would be a potentially major loss in quantitative security.

As it turns out, neither price need be paid. In particular, it is fine to leave the round constants independent of the tweak T, and, even when doing so, there need be no quantitative security loss in the bound from making this change. What we call tweaked-SR, then, is identical to Fig. 1 except that the tweak T is added to the scope of F at line 23.

To establish security for this scheme, obtaining the same bounds as before, we go back to the swap-or-not shuffle and show that, in that context, if the round constants are left untweaked but

the round function is tweaked, then equation (1) continues to hold. The result is as follows. In its statement and proof, for μ and ν probability distributions on Ω, denote the total variation distance between them by $\|\mu - \nu\| = (1/2)\sum_{x\in\Omega} |\mu(x) - \nu(x)|$.

Theorem 3. *Fix $q_1, \ldots, q_l$ with $\sum_{i=1}^{l} q_i = q$. Let $X_t^1, X_t^2, \ldots, X_t^l$ be SN shuffles on G driven by the same round constants $K_1, \ldots, K_r$, but independent round functions. Let $X_t = (X_t^1, \ldots, X_t^l)$. For i with $1 \le i \le l$, let π^i be the uniform distribution on q_i samples without replacement from G, and let $\pi = \pi^1 \times \pi^2 \cdots \times \pi^l$. That is, π is the distribution of l independent samples, one each from $\pi^1, \pi^2, \ldots, \pi^l$. Let τ be the distribution of X_r. Then*

$$\|\tau - \pi\| \le \frac{2N^{3/2}}{r+2}\left(\frac{q+N}{2N}\right)^{r/2+1}. \tag{4}$$

Proof. Let

$$\Delta(j) = \sum_{m=0}^{j-1} \frac{\sqrt{N}}{2}\left(\frac{m+N}{2N}\right)^{r/2}.$$

We show that

$$\|\tau - \pi\| \le \Delta(q)$$

from which (4) follows by way of

$$\begin{aligned}\|\tau - \pi\| &\le \sum_{m=0}^{q-1} \frac{\sqrt{N}}{2}\left(\frac{m+N}{2N}\right)^{r/2} \\ &\le N^{3/2}\int_0^{q/2N} (1/2+x)^{r/2}\,dx \\ &\le \frac{2N^{3/2}}{r+2}\left(\frac{q+N}{2N}\right)^{r/2+1}.\end{aligned}$$

For random variables $W_1, W_2, \ldots, W_j$, we write $\tau^i(\,\cdot \mid W_1, W_2, \ldots, W_j)$ for the conditional distribution of X_r^i given $W_1, W_2, \ldots, W_j$. Then we have

$$\|\tau - \pi\| \le \sum_{i=1}^{l} \mathbf{E}\left(\|\tau^i(\,\cdot \mid X_r^1, \ldots, X_r^{i-1}) - \pi^i\|\right). \tag{5}$$

We claim that

$$\mathbf{E}\left(\|\tau^i(\,\cdot \mid X_r^1, \ldots, X_r^{i-1}) - \pi^i\|\right) \le \Delta(q_i). \tag{6}$$

For distributions μ and ν the total variation distance $\|\mu - \nu\|$ is half the L^1-norm of $\mu - \nu$. Since the L^1-norm is convex, to verify the claim it is enough to show that

$$\mathbf{E}\left(\|\tau^i(\,\cdot \mid X_r^1, \ldots, X_r^{i-1}, K_1, \ldots, K_r) - \pi^i\|\right) \le \Delta(q_i).$$

But the X_r^i are conditionally independent given $K_1, K_2, \ldots, K_r$, so

$$\tau^i(\,\cdot \mid X_r^1, \ldots, X_r^{i-1}, K_1, \ldots, K_r) = \tau^i(\,\cdot \mid K_1, \ldots, K_r).$$

Thus it remains to show that

$$\mathbf{E}\left(\|\tau^i(\,\cdot \mid K_1, \ldots, K_r) - \pi^i\|\right) \leq \Delta(q_i),$$

but this is exactly what is shown in [10, fourth-to-last equation of p. 8]. This verifies (6). Combining this with (5) gives

$$\begin{aligned}\|\tau - \pi\| &\leq \sum_{i=1}^{l} \Delta(q_i) \\ &\leq \Delta(q),\end{aligned}$$

where the second inequality holds because the summands in the definition of $\Delta(j)$ are increasing. This completes the proof. □

Theorem 3 plays the same role in establishing the security for tweaked-SR as equation (1) played for establishing the security of the basic version. The values in the table of Fig. 2, for example, apply equally well to the tweakable-SR.

We comment that in the the tweakable version of SR, the round constants do depend on the generated N-values. This dependency can also be eliminated, but we do not pursue this for now.

6 Absence of Timing Attacks

With SR (and, more generally, with **SR**), the total number of rounds t^* used to encipher a plaintext $X \in [N_0]$ to a ciphertext $Y \in [N_0]$ will depend on X and the key $\boldsymbol{K} = KF$. This suggests that an adversary's acquiring t^*, perhaps by measuring the running time of the algorithm, could be damaging. But this is not the case, for the adversary knows Y, and t^* depends only on it.

It is easiest to describe this dependency when $N_0 = 2^n$ is a power of two. In that case the generated N-values are $2^n, 2^{n-1}, \ldots, 4, 2, 1$. Let $t'_0, t'_1, \ldots, t'_{n-2}, t'_{n-1}, t'_n$ be the corresponding round counts (the last two values are 1 and 0, respectively). Let $t^*_j = \sum_{i \leq j} t'_i$ be the cumulative round counts: the total number of SN rounds if we run for $j+1$ stages. Then t^* is simply t^*_ℓ where ℓ is the number of leading 0-bits in the n-bit binary representation of Y. The adversary holding a ciphertext of $Y = 0^z 1 Z$, knows that it was produced using $t^* = t^*_z$ rounds of SN. Ciphertext 0^n is the slowest to produce, needing t^*_n rounds.

The observation generalizes when N_0 is not a power of 2: the set $[N_0]$ is partitioned into easily-calculated intervals and the number of SN rounds that a ciphertext Y was subjected to is determined by the interval containing it.

7 Discussion

Alternative description. It is easy to eliminate the tail recursion of Fig. 1; no stack is needed. This and other changes are made to the alternative description of tweaked-SR given in Fig. 4. While the algorithm looks rather different from before, it is equivalent.

Which pile to recurse on? The convention that **SR** recurses on the first (left) pile of cards, rather than on the second (right) pile of cards, simplifies bookkeeping: in this way, we will always be following a card $X \in [N]$ for decreasing values of N. Had we recursed on the second pile we

```
50  procedure E_KF^{T,N0}(X)                    //Encipher X ∈ [N0] with tweak T, key KF
51  N ← N0                                      //initial-N
52  for j ← 0 to ∞ do                           //for each stage, until we return
53      for i ← 1 to t'_j do                    //SN, for as many rounds as needed for this stage
54          X' ← K_i − X (mod N)                //X' is the partner of X
55          X̂ ← max(X, X')                      //canonical name for {X, X'}
56          if F(i, X̂, T) = 1 then X ← X'       //maybe swap X and X'
57      if X ≥ ⌊N/2⌋ then return X              //right pile is done
58      N ← ⌊N/2⌋                               //left pile is new domain to shuffle
```

Fig. 4. Alternative description of the tweaked construction. We eliminate the recursion and assume the canonical split.

would be following a card $X \in [N_0-N+1\,..\,N_0-1]$ for decreasing values of N. Concretely, the code in Figures 1 and 4 would become more complex with the recurse-right convention.

Multiple concurrent domains. Our assumption has been that the domain for the constructed cipher is $[N_0]$ for some N_0. As with variable-input-length (VIL) PRFs, it makes sense to seek security against adversaries that can simultaneously encipher points from any number of domains $\{[N_0] :\ N_0 \in \mathcal{N}\}$, as previously formalized [3]. This can be handled by having the round-function and round-keys depend on the description of the domain N_0. Once again it seems unnecessary to reflect the N_0 dependency in the round-keys. To prove the conjecture will take a generalization of Theorem 3.

Open question. The outstanding open question in this domain is whether there is an oblivious shuffle on N cards where a card can be tracked through the shuffle in *worst-case* $\Theta(\lg N)$-time. Equivalently, can we do information-theoretic PRF to PRP conversion with $\Theta(\lg N)$ calls, always, to the (constant-output-length) PRF?

Acknowledgments

This work was made possible by Tom Ristenpart and Scott Yilek generously sharing an early draft of their work [15]. Thanks also to Tom and Scott for their comments and interaction. Thanks to Terence Spies and Voltage Security, whose interest in FPE has motivated this line of work. Our work was supported under NSF grants CNS-0904380 and CNS-1228828.

References

1. Accredited Standards Committee X9, Incorporated (ANSI X9). X9.124: Symmetric key cryptography for the financial services industry — format preserving encryption. Manuscript, 2011.
2. M. Bellare and R. Impagliazzo. A tool for obtaining tighter security analyses of pseudorandom function based constructions, with applications to PRP to PRF conversion. ePrint report 1999/024.
3. M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. *Selected Areas in Cryptography* (SAC) 2009. Springer, pp. 295–312, 2009.
4. J. Black and P. Rogaway. Ciphers with arbitrary finite domains. *CT-RSA 2002*, LNCS vol. 2271, Springer, pp. 114–130, 2002.
5. M. Brightwell and H. Smith. Using datatype-preserving encryption to enhance data warehouse security. 20th National Information Systems Security Conference Proceedings (NISSC), pp. 141–149, 1997.
6. M. Dworkin. NIST Special Publication 800-38G: Draft. Recommendation for block cipher modes of operation: methods for format-preserving encryption. July 2013.

7. FIPS 74. U.S. National Bureau of Standards (U.S). Guidelines for implementing and using the NBS Data Encryption Standard. U.S. Dept. of Commerce, 1981.
8. L. Granboulan and T. Pornin. Perfect block ciphers with small blocks. *Fast Software Encryption* (FSE 2007), LNCS vol. 4593, Springer, pp. 452–465, 2007.
9. J. Håstad. The square lattice shuffle. *Random Structures and Algorithms*, 29(4), pp. 466–474, 2006.
10. V. Hoang, B. Morris, and P. Rogaway. An enciphering scheme based on a card shuffle. *CRYPTO 2012*, pp. 1–13.
11. M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. *J. of Cryptology*, 24(3), pp. 588–613, 2011.
12. B. Morris. The mixing time of the Thorp shuffle. *SIAM J. on Computing*, 38(2), pp. 484–504, 2008. Earlier version in *STOC 2005*.
13. B. Morris, P. Rogaway, and T. Stegers. How to encipher messages on a small domain: deterministic encryption and the Thorp shuffle. CRYPTO 2009, pp. 286–302, 2009.
14. M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. *J. of Cryptology*, 12(1), pp. 29-66, 1999.
15. T. Ristenpart and S. Yilek. The mix-and-cut shuffle: small-domain encryptions secure against N queries. To appear at *CRYPTO 2013*.
16. S. Rudich. Limits on the provable consequences of one-way functions. Ph.D. Thesis, UC Berkeley, 1989.
17. E. Stefanov and E. Shi. FastPRP: Fast pseudo-random permutations for small domains. Cryptology ePrint Report 2012/254.
18. E. Thorp. Nonrandom shuffling with applications to the game of Faro. *Journal of the American Statistical Association*, 68, pp. 842–847, 1973.