The Mix-and-Cut Shuffle: Small-Domain Encryption Secure against N Queries

Thomas Ristenpart[1] and Scott Yilek[2]

[1] University of Wisconsin–Madison
rist@cs.wisc.edu
[2] University of St. Thomas
syilek@stthomas.edu

Abstract. We provide a new shuffling algorithm, called Mix-and-Cut, that provides a provably-secure block cipher even for adversaries that can observe the encryption of all $N = 2^n$ domain points. Such fully secure ciphers are useful for format-preserving encryption, where small domains (e.g., $n = 30$) are common and databases may well include examples of almost all ciphertexts. Mix-and-Cut derives from a general framework for building fully secure pseudorandom permutations (PRPs) from fully secure pseudorandom separators (PRSs). The latter is a new primitive that we treat for the first time. Our framework was inspired by, and uses ideas from, a particular cipher due to Granboulin and Pornin. To achieve full security for Mix-and-Cut using this framework, we give a simple proof that a PRP secure for $(1 - \epsilon)N$ queries (recently achieved efficiently by Hoang, Morris, and Rogaway's Swap-or-Not cipher) yields a PRS secure for N queries.

Keywords: shuffles, small-block encryption, tweakable block ciphers.

1 Introduction

Traditional block ciphers such as AES and DES work on fixed domain sizes (e.g., $n = 64$ or 128 bits). Some applications, however, require the ability to securely encipher bit strings of smaller sizes (e.g., $n = 30$ bits). The canonical example here being format-preserving encryption (FPE) [2, 4, 6, 7], which makes use of small-block-size block ciphers to perform in-place encryption of credit card numbers, social security numbers, and other sensitive data.

In this paper, we provide a new small-domain block cipher, called Mix-and-Cut, that achieves provable security up to $q = 2^n$ queries (the most possible). It was designed using a new methodology for building ciphers secure as pseudorandom permutations (PRPs) from pseudorandom separators (PRSs). The latter is a cryptographic primitive that we treat for the first time. This methodology was inspired by, and uses ideas from, a particular cipher construction due to Granboulan and Pornin (GP) [11].

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part I, LNCS 8042, pp. 392–409, 2013.

Small domain encryption. Before explaining our results in more detail, we describe further the motivation and related work. FPE has become popular in settings where ciphertexts must follow a proscribed format. For example, should credit card numbers (CCNs) already be stored in a database with a limit of 16 numerical digits, then encrypting the CCN with a conventional encryption scheme would result in a ciphertext that could not be placed back in the database column. Typically, in fact, the first 6 digits (being the issuer identification number) and the last digit (a Luhn checksum digit) must also be left in the clear, and so one would like to encrypt just the remaining 9 digits. This requires a cipher with domain of $10^9 \approx 2^{30}$. In cryptographic parlance, we seek a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ for some key space $\mathcal{K}$, $n = 30$, and which is indistinguishable from a random permutation.

Work on this small-space encryption problem can be traced back to, at least, Black and Rogaway [5]. They gave several approaches, the most efficient of which generalizes Luby and Rackoff's classic result [17] on balanced Feistel networks. This provides provable security, but only up to a number of encryptions below $q = 2^{n/4}$. In our example with $n = 30$, this is only $q \approx 128$ ciphertexts. Morris, Rogaway, and Stegers [19] uncover a connection between unbalanced Feistel networks and the Thorp shuffle, and use it to prove that maximally unbalanced Feistel networks achieve security up to about $q \approx 2^{n(1-\epsilon)}$ queries for a fraction ϵ inversely proportional to the the number of rounds of the construction. This approach, with similar bounds, was extended to arbitrary balanced Feistels by Hoang and Rogaway [14]. Most recently, Hoang, Morris, and Rogaway [13] introduced a new shuffling approach, called Swap-or-Not, and a cryptographic realization of it that provably achieves security up to $q \approx (1-\epsilon)2^n$.

None of the above approaches, however, provide guarantees of security should q get within a constant of $N = 2^n$. Such *full security* is desirable when large databases contain almost N encryptions. In our running example, this would mean only about 1 billion database entries, which is not particularly large considering that processors using FPE deal with hundreds of millions of transactions each month. Employing key rotation and tweaks [16] can ease this gap (by reducing the number of ciphertexts per key/tweak), but their use for this purpose is not always feasible or desirable, for example should one need to support search. What's more, choosing appropriate security parameters requires somehow predicting the number of ciphertexts an adversary obtains. A fully secure cipher, on the other hand, can often set parameters based solely on n.

There exist a handful of proven full security constructions, but they are all quite slow for moderately sized N. The shuffle popularized by Knuth [9, 10, 15] provides a cipher with full security, but requires $\mathcal{O}(N)$ computation time, as does the simple construction that uses lookup tables (c.f., [5]). Granboulan and Pornin (GP) [11], build a cipher based on a shuffle introduced by Czumaj, Kanarek, Kutylowski, and Lory [8]. It requires $\mathcal{O}(\log^3 N)$ operations and repeated samplings from the hypergeometric distribution. The Thorp shuffle was shown to provide full security [18], but also with $\mathcal{O}(\log^3 N)$ rounds. Stefanov and Shi [23] offer a variant of the GP scheme that improves performance for very small

algorithm $\mathsf{MixAndCut}(D)$:
$n \leftarrow \log |D|$
Repeat r times:
  $K \leftarrow\!\!\$\ \{0,1\}^n$
  for each pair of positions $\{X, K \oplus X\}$
    $b \leftarrow\!\!\$\ \{0,1\}$
    If $b = 1$ swap the cards at positions X and $K \oplus X$
$(D_1, D_2) \leftarrow \mathsf{Cut}(D)$
$\mathsf{MixAndCut}(D_1)$
$\mathsf{MixAndCut}(D_2)$
$\mathsf{Gather}(D_1, D_2)$

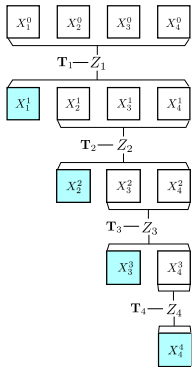

Fig. 1. (Left) The Mix-and-Cut shuffle on $N = 2^n$ cards with Swap-or-Not for the mixing. D is the size N deck of cards, $|D|$ is the number of cards in D, $\mathsf{Cut}(D)$ cuts the deck exactly in half, and $\mathsf{Gather}(D_1, D_2)$ stacks two piles on top of each other. **(Right)** Diagram of the Icicle construction with four stages ($s = 4$) and using pseudorandom separators $Z_1, \ldots, Z_4$. Each of the X_j^i have length γ bits. The leftmost γ bits (the shaded boxes) of each stage "drip" down to the final ciphertext $X_1^1 \,\|\, X_2^2 \,\|\, X_3^3 \,\|\, X_4^4$. Each $\mathbf{T}_i$ is a tweak that includes the X_j^j values output in previous stages $j < i$.

domains, but it requires $\tilde{\Theta}(N)$ time for key setup and $\tilde{\Theta}(N^{1/2})$ time and space for encryption.

Mix-and-Cut. We offer a new cipher, which, following [13,14,19] can be viewed as a card shuffle. Think of each of the 2^n domain points as a card. Our new shuffling algorithm intermingles two kinds of shuffles on these cards. The first is the recursive shuffling procedure underlying GP, in which at each stage one splits the deck of cards into two halves, then splits each half into two smaller halves for a total of four halves, and so on until one can split no further. GP uses a recursive hypergeometric sampling routine to make the split perfectly random. We instead use another shuffling routine to provide a split that is cryptographically indistinguishable from a random one. For this we use a number of rounds of the Swap-or-Not shuffle sufficient to ensure the top and bottom halves of the deck appear to have been chosen randomly (but not yet nearly enough rounds to prove a full uniform shuffling of the deck). The complete shuffling algorithm is given on the left of Figure 1. The full shuffle, then, will *mix* the deck using Swap-or-Not, then *cut* it into two, recursively shuffle each one independently, and so on.

Mix-and-Cut may appear ad-hoc, fitting together two prior shuffles in an arbitrary fashion. But in fact it was discovered using, and is just the best instantiation we could find of, a new paradigm for building ciphers that is inspired by the construction of the GP cipher. We describe this paradigm from the bottom up, starting with a new cryptographic primitive that we define, called a Γ-pseudorandom separator.

Γ-PSEUDORANDOM SEPARATORS. A Γ-PRS, from a shuffling point of view, can "split a deck" pseudorandomly into Γ piles. Cryptographically speaking, a permutation Z on $\{0,1\}^n$ is a secure Γ-PRS if no computationally bounded adversary can distinguish the first $\gamma = \log \Gamma$ bits of its output from that of a randomly chosen, regular function $\{0,1\}^n \to \{0,1\}^\gamma$. That the ideal object is regular is important: we will be often interested in n small and q large, and here the difference between a regular random function and arbitrary random function is readily apparent even to a computationally bounded adversary. For large n and small q, however, Γ-PRS security can be shown to be equivalent to security in the sense of a PRF with range size γ bits using techniques first used to analyze the PRF security of truncated PRPs [1, 12].

Unlike a PRP, a Γ-PRS provides no security guarantees about the remaining $n-\gamma$ bits of its output (when $\gamma < n$). This means that Γ-PRS security is strictly weaker than PRP security in general; when $\gamma = n$ the two notions coincide.

In fact, we focus on tweakable separators, which like tweakable block ciphers in the sense of Liskov, Rivest and Wagner [16], are families of permutations indexed by both a key and a tweak. Different tweaks should give rise to independent-looking permutations. Tweaks will be critical to our use of PRSs in building PRPs.

Separators were inspired by the use, in the GP cipher, of an algorithm for perfectly splitting a deck into two halves using a recursive sampling from the hypergeometric distribution. Our treatment here draws out the implicit cryptographic primitive underlying GP's algorithm: in our terminology, the GP hypergeometric separator is a fully secure 2-PRS.

FROM PRSS TO PRPS. Fix a value γ and target block size n, with $s = n/\gamma$. We provide a new construction, that we call an icicle, that uses a set of tweakable permutations $Z_1, \ldots, Z_s$ to build a secure PRP on n bits. See the right hand side of Figure 1. Each Z_i should be a secure $\Gamma = 2^\gamma$-PRS on $n - \gamma(i-1)$ bits. An icicle is simple: apply an n-bit Γ-PRS, output the first γ bits, apply to the remaining $n-\gamma$ bits a Γ-PRS on $n-\gamma$ bits with tweak being the output thus far, add the first γ bits of the result to the output, and so on for s stages, in order to produce a sequence of γ-bit outputs that is the ciphertext. The use of tweaks is requisite for security: they ensure that the PRSs lower in the icicle provide independent behavior for different prefixes of the ciphertext. An icicle for $\Gamma = 2$ is exactly the recursive shuffling procedure used in the GP cipher; their cipher we can view now as an icicle using hypergeometric 2-PRSs. Our proof of the icicle construction modularizes their result, and generalizes it to work with imperfect, computational PRSs for arbitrary Γ. The proof conserves full security, as well, so if the underlying PRSs are fully secure, so too is the resulting PRP.

Overall, this can be seen as a new paradigm for building PRPs. Unlike Luby-Rackoff that starts with PRFs, we go a different route, starting with PRSs. This may seem to not buy much; in particular, building an n-bit cipher using an icicle requires an n-bit permutation (the first stage) and also a γ-bit PRP (the last stage). But we only require the first γ bits of the first stage to be random-looking,

and we can arrange that γ is small enough in the last stage to make a PRP there trivial (e.g., $\gamma = 1$).

A simple but useful lemma. This begs the question of how to build PRSs, particularly ones that are faster than the GP hypergeometric 2-PRS. We show that the inverse E_K^{-1} of any cipher E_K that is a secure PRP for $(\Gamma - 1)N/\Gamma$ queries is a good Γ-PRS for all N queries. The proof is straightforward (see Section 4), and we explain it informally here for the case of $\Gamma = 2$ which we will use later. The reduction must simulate all N 2-PRS queries given only $N/2$ evaluations of E. But to do so requires only returning the first bit of each value $E_K^{-1}(X_1), \ldots, E_K^{-1}(X_N)$, and this is learnable by querying only half the domain, say by querying $E_K(0 \,\|\, y)$ for all $y \in \{0,1\}^{n-1}$. If a value X is in the set of returned points, then we know that the first bit of $E_K^{-1}(X)$ is zero and otherwise that it is one. The result extends easily to handle when the PRS adversary queries all N domain points for each of some number of tweaks.

The lemma shows that one can get a fully secure Γ-PRS using any construction that achieves only $(1 - \epsilon)N$ PRP security. In particular this implies that Swap-or-Not is a fully secure 2-PRS for a number of rounds a small fraction of that needed to make it a fully secure PRP (provably under the Hoang et al. result).

Putting it all together. The Mix-and-Cut cipher uses the icicle construction with fully secure 2-PRSs built from Swap-or-Not. Our simple lemma described above ensures that we can use the number of Swap-or-Not rounds suggested by Hoang et al. for $N/2$ queries to establish 2-PRS security for N queries. Back to the shuffling interpretation, Swap-or-Not need only shuffle enough to ensure that the "deck" can be cut into two piles with a pseudorandom assignment of cards to piles. We then cut the deck, and focus on each pile independently.

The resulting cipher provides full security, using only simple operations: Swap-or-Not can be instantiated with two AES calls per round. That said, the new cipher does require a large number of rounds. For $N = 2^{30}$ and an advantage of less than 10^{-10} we need around 10,000 rounds. By comparison, using Swap-or-Not directly under the bounds[1] of Hoang et al. requires about 126×10^9 rounds to achieve the same advantage for $N = 2^{30}$. On an Intel Core i5 with AES-NI, a full application of Mix-and-Cut should take less than a millisecond for $N = 2^{30}$. Improved analyses for Swap-or-Not or another algorithm (directly as a 2-PRS or as a PRP) can be used immediately by Mix-and-Cut in order to increase efficiency.

Adding tweaks and CCA security. It is easy to ensure that icicle produces a tweakable block cipher: just prepend the tweak T to the tweak used in each stage. We have also described all the above in terms of achieving CPA security. But CPA security and CCA security are equivalent when $q = N$ — another advantage of targeting full security.

[1] Their bound is vacuous if one sets $q = N$, but we can apply it with $q = N - 1$.

Separators for generalized domains. Largely for pedagogical reasons, we focus in this abstract on the case where domains have sizes that are powers of 2. However, many small-domain encryption settings require domains that operate over non-binary digits (e.g., base 10 for credit cards). We can generalize our results to work with radixes other than two in a straightforward way. This natural generalization leads to slightly weaker bounds than the binary case. It is also possible to treat the most general case, which uses PRSs to build tweakable ciphers for completely arbitrary domains. We give details in the full version.

Other uses of Γ-PRSs. We believe that the new Γ-PRS primitive will find application in contexts beyond our goal here of full security ciphers. As one example, we show in the full version that 2 rounds of balanced Feistel gives a good $2^{n/2}$-PRS, though with security only for $q \approx 2^{n/4}$. While therefore not directly useful for full security applications, we show that one can recast the original Luby-Rackoff (LR) result [17] that 3-rounds provides a secure PRP as a composition of any $2^{n/2}$-PRS with one round of Feistel. Combining the two results gives a bound that matches the original LR result.

Further discussion. Should one be interested in just partial security ($q \ll N$), then our approach does not provide the most efficient solution. This limitation extends as well to very large block sizes, where partial security is inherently the goal (since, e.g., $q = 2^{128}$ is unrealistic); our lemma described above scales exponentially in q. In these settings one would do best to stick with (say) Swap-or-Not up to $n = 64$ and from there use traditional block ciphers within a suitable domain extension transform (e.g., [22]).

The icicle construction, being built from any Γ-PRS, can of course be used in a multitude of ways. A two stage icicle extends the domain of any fixed-length PRP by γ bits. One can also build a full cipher using multiple different kinds of separators across different icicle stages. For example, one could use several stages of (say) Swap-or-Not before applying a different shuffle for a few stages, before then applying at the bottom of the icicle a permutation that works well for very small domains. While the resulting cipher would be more complicated than Mix-and-Cut with its homogeneous set of separators, it suggests the existence of a wide space of possible designs from which one might use proven-secure ciphers for the domain sizes for which they work best to improve overall efficiency.

Finally, we note that in practice small-block encryption uses constructions that have weak bounds or even have no security proofs entirely. The proposed FFX [3,4] standard for FPE suggests 10 rounds of Feistel for domain size around 2^{30}. The choice of rounds is based on a heuristic; no proofs providing reasonable bounds are known for this choice and in fact no proofs are likely given current techniques (see [19] for more discussion). That said, no (computationally reasonable) attacks are known, and we have no reason to believe that efficient attacks will arise. (The best is due to Patarin [21], but the round choice was made to defeat this.) But that is not proof of the absence of attacks, and so we view closing the (large) performance gap between Mix-and-Cut and ciphers such as FFX an important open research problem.

2 Preliminaries

TWEAKABLE BLOCK CIPHERS. A tweakable block cipher is a family of functions $E : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{K}$ is a non-empty, finite set called the key space, $\mathcal{T}$ is a non-empty, finite set called the tweak space, and where for every $K \in \mathcal{K}$ and $T \in \mathcal{T}$, $E_K(T,\cdot) = E(K,T,\cdot)$ gives a permutation on $\{0,1\}^n$. We let $E_K^{-1}(T,\cdot)$ denote the inverse block cipher of E. When $\mathcal{T}$ is a singleton, we have a block cipher, and write instead $E_K(\cdot) = E(K,\cdot)$ and $E_K^{-1}(\cdot) = E^{-1}(K,\cdot)$.

We target tweakable block ciphers that are secure even under chosen-ciphertext attack. This is sometimes called strong pseudorandom permutation (SPRP) security. Given tweakable block cipher $E : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ and an adversary A, the cca-advantage of A with respect to E is

$$\mathbf{Adv}^{\mathrm{cca}}_E(A) = \Pr\left[A^{E(K,\cdot,\cdot),E^{-1}(K,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\pi(\cdot,\cdot),\pi^{-1}(\cdot,\cdot)} \Rightarrow 1\right]$$

where the first probability is over $K \xleftarrow{\$} \mathcal{K}$ and the coins used by A and the second probability is over $\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{T},n)$ and the coins used by A. Here $\mathsf{Perm}(\mathcal{T},n)$ is the set of families of n-bit permutations. That means picking π gives a family of uniformly chosen permutations, one for each $T \in \mathcal{T}$. The inverse of π is denoted π^{-1}. When $\mathcal{T} = \{0,1\}^t$ we will write $\mathsf{Perm}(t,n)$. A cpa adversary simply makes no queries to its second oracle, and for such adversaries we denote their advantage by $\mathbf{Adv}^{\mathrm{cpa}}_E(A)$ as a reminder that A makes no inverse queries. We call a cipher that is secure under cpa attack a good PRP.

We will need as well non-adaptive cpa security, which we define as follows. A non-adaptive cpa adversary A is given access to one of two different oracles to which it can query a single time a pair of vectors $(T_1,\ldots,T_q)$ and $(M_1,\ldots,M_q)$. The oracle $\boldsymbol{E}(K,(T_1,\ldots,T_q),(M_1,\ldots,M_q))$ computes $C_i = E_K^{T_i}(M_i)$ for all i and returns the resulting ciphertexts. The oracle $\boldsymbol{\pi}((T_1,\ldots,T_q),(M_1,\ldots,M_q))$ computes $C_i = \pi(T_i,M_i)$ for a random tweakable permutation π, and returns the results. We define advantage as

$$\mathbf{Adv}^{\mathrm{ncpa}}_E(A) = \Pr\left[A^{\boldsymbol{E}(K,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\boldsymbol{\pi}(\cdot,\cdot)} \Rightarrow 1\right]$$

where the first probability is over the choice of $K \xleftarrow{\$} \mathcal{K}$ and the coins used by A and the second probability is over $\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{T},n)$ and the coins used by A.

FULL SECURITY. We target full security, meaning that cca advantage should be low even for adversaries that make $q = N = 2^n$ queries for some number w of tweaks. (Clearly $q = N - 1$ is also sufficient, but the difference matters little.) Thus, full security requires security to hold for a total of wN queries. When security holds only for $q \ll N$ we say that the tweakable cipher instead achieves only *partial security*. Partial security suffices when, as with standard block ciphers with $n = 128$, the domain is so large that no adversary could feasibly obtain, let alone compute over, anywhere remotely close to N queries. In small domain encryption settings, however, full security is important as applications may apply a cipher to most of the domain (for some set of tweaks). Another advantage of targeting full security is that cpa security and cca security are

equivalent when $q = N$. The following formalizes this fact and allows us to focus on cpa adversaries in the remainder of the paper.

Lemma 1. *Let* $E\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *and* $N = 2^n$. *Let* A *be a cca adversary making queries for* w *distinct tweaks. Then for the cpa adversary* B *specified in the proof below it holds that* $\mathbf{Adv}^{\mathrm{cca}}_E(A) \le \mathbf{Adv}^{\mathrm{cpa}}_E(B)$. *Moreover* B *makes at most* wN *queries and runs in time that of* A *plus* $\mathcal{O}(wN \log wN)$ *time.*

Proof. The adversary B runs adversary A. When A makes either a forward or inverse query on a not-before-seen tweak T, B immediately queries values $(T, X_1), \ldots, (T, X_N)$. That is, it queries the entire domain. Then B uses the resulting values $Y_1, \ldots, Y_N$ to respond to the query, and to respond to future forward or inverse oracle queries using T. ∎

Note that the variable w in the above only measures the number of distinct tweaks queried, not the total number of queries made by A. Thus, even if A makes $2^n - 1$ queries on each tweak, the bound holds as shown. Also, when we use big-O notation, i.e., $\mathcal{O}(w2^n \log w2^n)$ in the lemma above, this hides only small, fixed constants.

PRFs. Let $F\colon \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^n$ be a family of functions. For an adversary A, the prf security of F with respect to A is $\mathbf{Adv}^{\mathrm{prf}}_F(A) = \Pr\left[A^{F(K,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\rho(\cdot)} \Rightarrow 1\right]$ where the first probability is over $K \xleftarrow{\$} \mathcal{K}$ and the coins used by A and the second probability is over $\rho \xleftarrow{\$} \mathsf{Func}(\ell, n)$ and the coins used by A. Here $\mathsf{Func}(\ell, n)$ is the set of all functions $\{0,1\}^\ell \to \{0,1\}^n$. It will be convenient to speak of PRFs that accept tweaks as input in addition to messages and keys, so a map $F\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^\ell \to \{0,1\}^n$. It is easy to build such an F that is a good PRF using an untweaked PRF. We write $\mathsf{Func}(\mathcal{T}, \ell, n)$ to denote the set of all such functions and $\mathsf{Func}(t, \ell, n)$ should $\mathcal{T} = \{0,1\}^t$.

3 Pseudorandom Separators

We define a new security goal for tweakable permutations. Informally speaking, a family of permutations is a good Γ-*pseudorandom separator* or simply a good Γ-PRS if no adversary can distinguish the first $\gamma = \log \Gamma$ bits of its outputs from a random, regular function $\{0,1\}^n \to \{0,1\}^\gamma$. (Regular just means that each range point has $2^{n-\gamma}$ preimages.) The shuffling-based interpretation is that the permutation does a good job of separating the domain into Γ different piles with random domain points (cards) assigned to each pile.

More formally, a *tweakable separator* is a map $Z\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is the key space, $\mathcal{T}$ is the tweak space, and we require that for all $K \in \mathcal{K}$ and $T \in \mathcal{T}$, $Z(K, T, \cdot)$ is a permutation with inverse $Z^{-1}(K, T, \cdot)$. Let $\mathsf{RegFunc}(\mathcal{T}, n, \gamma)$ be the set of all functions $f\colon \mathcal{T} \times \{0,1\}^n \to \{0,1\}^\gamma$ where $f(T, \cdot)$ is regular for any $T \in \mathcal{T}$. Then for $\gamma \le n$, the Γ-prs advantage of an adversary A is defined by

$$\mathbf{Adv}^{\Gamma\text{-prs}}_Z(A) = \Pr\left[A^{Z(K,\cdot,\cdot)[\gamma]} \Rightarrow 1\right] - \Pr\left[A^{\hat{\rho}(\cdot,\cdot)} \Rightarrow 1\right]$$

where the first probability is over $K \xleftarrow{\$} \mathcal{K}$ and the coins used by A and the second probability is over $\hat{\rho} \xleftarrow{\$} \mathsf{RegFunc}(\mathcal{T}, n, \gamma)$ and the coins used by A. The oracle $Z(K, \cdot, \cdot)[\gamma]$ on input (T, x), returns the first γ bits of $Z(K, T, x)$.

We say, informally, that a tweakable separator Z is Γ-PRS secure if the advantage is low for all adversaries given reasonable resources. Looking ahead, we will use separators in building fully secure, small-block ciphers, and so reasonable here means $q = N$ queries can be made for each of some number ω of tweaks.

It is easy to see that any separator that is Γ-PRS secure is also Γ'-PRS secure for any $\gamma' < \gamma$. The converse is also true, for any $\gamma' < \gamma$ one can construct a Γ'-PRS that is not a Γ-PRS. (Briefly, use the hypergeometric sampler described below within the icicle construction of Section 5 for just γ' stages, and output the result without further processing of the low $n - \gamma'$ bits.)

Separators for generalized domains. In the above, and in the next few sections, we restrict attention to the case of bit-string domains. We find this to be pedagogically appealing. However, many small-domain encryption settings require domains that operate over non-binary digits (e.g., base 10 for credit cards). Generalizing the above formulation and our results in Sections 4, 5, and 6 to work over domains being strings over an arbitrary alphabet Σ is straightforward. We do caution that the natural generalization leads to slightly weaker bounds than the binary case; see the full version for the details. There we also treat the most general case, using PRSs to build tweakable ciphers for domains of any size.

PRS versus PRF. It may seem that we have, above, just tediously repeated with different language the classical PRF security notion for the special case of a truncated permutation. However, there is indeed a gap between the two notions, due to the fact that we require the random function in the PRS setting to be regular. In fact, PRF security is not helpful to us in our full security setting. Formally:

Proposition 1. *Let $\rho \xleftarrow{\$} \mathsf{Func}(\mathcal{T}, n, \gamma)$ for $n \geq \gamma$ and finite set $\mathcal{T}$. There exists an adversary A making $N = 2^n$ queries such that $\mathbf{Adv}_{\rho}^{\Gamma\text{-prs}}(A) > 1 - e/\left((\sqrt{2\pi})^{\Gamma} \cdot N^{(\Gamma-1)/2} \cdot \Gamma^{(N-\Gamma)/2}\right)$ where $\Gamma = 2^{\gamma}$.*

The attack is simple: just query all points, and determine if the function is regular by inspecting preimage sets. The probability that a random function is regular is at most the fraction shown, derived straightforwardly using Stirling's approximation. Even for $n = 2$ and $\gamma = 1$, we have the advantage of A being about 0.9.

We note, however, that for $q \ll N$ and for certain ranges of values of γ and n, there exist upper bounds showing that PRF security implies PRS security. For example, a simple birthday bound argument shows this for reasonably large γ (and so n) and relatively small q. Better bounds for some parameters by Hall, Wagner, Kelsey, and Schneier [12] as well as Bellare and Impagliazzo [1] arise in their analyses of truncated PRPs. Their results also yield analogues to Proposition 1 with improved bounds when A is allowed fewer than N queries.

PRS VERSUS PRP. When $\gamma = n$, PRS security is equivalent to PRP security. For $\gamma < n$, however, PRS is strictly weaker. The reason is that the high bits of the output of a separator are not given to the adversary, and so these may be entirely non-random. We next give an example of a secure separator that is easily distinguished from a PRP.

THE HYPERGEOMETRIC 2-PRS. As an example of a PRS, we turn to the Granboulan and Pornin [11] splitting function that we denote $Z_{\mathrm{GP}}(x)$. Their algorithm cleverly uses repeated sampling from the hypergeometric distribution. We provide a fuller description in the full version.

GP show that Z_{GP} is (in our terms) a secure 2-PRS. It is not a secure PRP. More specifically, Z_{GP} has the property that if $x < x'$ and x and x' are mapped to the same half of the output space, then $Z_{\mathrm{GP}}(x) < Z_{\mathrm{GP}}(x')$. In other words, points that are mapped to the same half of the range retain their relative ordering. Because of this, for example, $Z_{\mathrm{GP}}(0^n)$ will always be equal to either 0^n or 10^{n-1}, and so Z_{GP} is easily distinguished from a random permutation.

The GP 2-PRS is, unfortunately, slow when implemented due to the expensive floating point computations needed to perform the repeated hypergeometric samplings. We will therefore seek other ways of constructing pseudorandom separators.

4 Full PRS Security from Partial CPA Security

In this section we prove a relationship between the cpa advantage and the Γ-prs advantage for a block cipher E. In short, we show that if a block cipher E is CPA secure against $(\Gamma-1)N/\Gamma$ queries, then its inverse E^{-1} is a secure Γ-PRS against N queries. Later in the paper, we will be particularly interested in the $\Gamma = 2$ case, since there a block cipher need only be CPA secure against $N/2$ queries. The following theorem captures this result.

Theorem 1. *Fix $n \geq \gamma \geq 1$. Let $N = 2^n$ and $\Gamma = 2^\gamma$. Let $E : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher and $E^{-1} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ its inverse. Let A be an Γ-prs adversary making queries with ω distinct tweaks. Then for the cpa adversary B specified in the proof below it holds that $\mathbf{Adv}^{\Gamma\text{-prs}}_{E^{-1}}(A) \leq \mathbf{Adv}^{\mathrm{cpa}}_{E}(B)$. Moreover, B makes exactly $\omega \cdot N \cdot (\Gamma-1)/\Gamma$ cpa queries and runs in time $\mathcal{O}(\omega N(\Gamma-1)/\Gamma \cdot \log(\omega N(\Gamma-1)/\Gamma))$.*

Proof. Adversary B runs A and answers its oracle queries as follows. On query (T, x) from B, if T is a tweak that has never been previously queried, then B queries its own oracle on (T, y) for all $y \in \{0,1\}^n$ *except* those that begin with 0^γ. If there is a y such that the oracle query (T, y) returned x to A, then B replies to A with the first γ bits of y. Otherwise, A replies with 0^γ. When A finally outputs a final bit, B outputs this same value. In short, B is able to perfectly simulate the environment for A because it makes sure to make enough queries to its oracle to determine the first γ bits of all of A's queries. ∎

The above theorem and proof are perhaps most easily understood for the $\Gamma = 2$ case. There, to answer which half of the set $\{0,1\}^n$ the point x is mapped to, B queries the second half of $\{0,1\}^n$, i.e., all points starting with a 1. If one of the returned answers is x, then B knows to return 1. If none of the returned answers is x, then x must be mapped to the other half of $\{0,1\}^n$, so A knows it can return a 0.

5 Building PRPs from PRSs: The Icicle Construction

We now show how to use PRSs to build PRPs. The route is a new construction that we call Icicle, which recasts the Permutator algorithm of Granboulan and Pornin [11] to work for arbitrary Γ-PRSs. (In the full version we provide a generalization of Permutator that works for any domain size.) Icicle can be viewed as a way to shuffle a deck of $N = 2^n$ cards. First, use a Γ-PRS to separate the full deck of cards into Γ piles pseudorandomly. Then, recursively shuffle each of the Γ piles by separating each of them pseudorandomly into Γ smaller piles, and so on. Finally collect up all the final piles (in some fixed order) back to form a single deck. It is important that in each recursive step the shuffling is independent across all the different piles.

The cryptographic interpretation can be formalized as follows. Let $\gamma, n \in \mathbb{N}$ be numbers with $1 \le \gamma \le n$ and $\gamma \mid n$. Let $\Gamma = 2^\gamma$. Let $Z_i \colon \{0,1\}^{k_i} \times \mathcal{T}_i \times \{0,1\}^{n_i} \to \{0,1\}^{n_i}$ for $i \in [1\,..\,n/\gamma]$ and where $\mathcal{T}_i = \mathcal{T} \times \{0,1\}^{\omega_i}$ for $\omega_i = (i-1)\gamma$ and $n_i = n - (i-1)\gamma$. Note that when $i = 1$, $\mathcal{T}_i = \mathcal{T}$ and $n_i = n$. Let $\mathcal{Z} = \{Z_1, \ldots, Z_s\}$. The Icicle construction $\mathrm{Ic}_{\gamma,n}(\mathcal{Z})$ builds a tweakable block cipher $E \colon \{0,1\}^{ks} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ as defined by the pseudocode in Figure 2. We use some notation there that must be defined: $X^1 \,\|\, \cdots \,\|\, X^0$ is defined to be the empty string, while $\mathrm{par}_k(\cdot)$ takes an input string of ik bits and parses it into i strings of length k bits (and similarly for par_γ).

The Icicle uses $s = n/\gamma$ stages. In the first stage, it applies to the full n bit input a Γ-PRS on n bits. This fixes the least γ bits of the final ciphertext, which "drip" down to the output. These bits specify to which of the Γ piles the input

algorithm $E(K,T,M)$:	algorithm $E^{-1}(K,T,Y)$:
$s \leftarrow n/\gamma$	$s \leftarrow n/\gamma$
$K^1, \ldots, K^s \leftarrow \mathrm{par}_k(K)$	$K^1, \ldots, K^s \leftarrow \mathrm{par}_k(K)$
$X_1^0, \ldots, X_s^0 \leftarrow \mathrm{par}_\gamma(M)$	$X_1^1, X_2^2, \ldots, X_s^s \leftarrow \mathrm{par}_\gamma(Y)$
for $i = 1$ to s:	for $i = s$ down to 1:
$\mathbf{T}^i \leftarrow T \,\|\, X_1^1 \,\|\, \cdots \,\|\, X_{i-1}^{i-1}$	$\mathbf{T}^i \leftarrow T \,\|\, X_1^1 \,\|\, \cdots \,\|\, X_{i-1}^{i-1}$
$\mathbf{X}^{i-1} \leftarrow X_i^{i-1} \,\|\, \cdots \,\|\, X_s^{i-1}$	$\mathbf{X}^i \leftarrow X_i^i \,\|\, \cdots \,\|\, X_s^i$
$X_i^i, \ldots, X_s^i \leftarrow \mathrm{par}_\gamma(Z_i(K_i, \mathbf{T}^i, \mathbf{X}^{i-1}))$	$X_i^{i-1}, \ldots, X_s^{i-1} \leftarrow \mathrm{par}_\gamma(Z_i^{-1}(K_i, \mathbf{T}^i, \mathbf{X}^i))$
$Y \leftarrow X_1^1 \,\|\, X_2^2 \,\|\, \cdots \,\|\, X_s^s$	$M \leftarrow X_1^0 \,\|\, \cdots \,\|\, X_s^0$
return Y	return M

Fig. 2. The s-stage Icicle construction $\mathrm{Ic}_{\gamma,n}(\mathcal{Z})$ using 2^γ-PRSs $\mathcal{Z} = \{Z_1, \ldots, Z_s\}$

algorithm $\mathrm{Hyb}_j(K, T, X)$:
$s \leftarrow n/\gamma$; $K^1, \ldots, K^s \leftarrow \mathrm{par}_k(K)$; $X_1^0, \ldots, X_s^0 \leftarrow \mathrm{par}_\gamma(X)$
for $i = 1$ to j:
  $T^i \leftarrow T \,\|\, X_1^1 \,\|\, \cdots \,\|\, X_{i-1}^{i-1}$; $\mathbf{X}^{i-1} \leftarrow X_i^{i-1} \,\|\, \cdots \,\|\, X_s^{i-1}$
  $X_i^i, \ldots, X_s^i \leftarrow \mathrm{par}_\gamma(Z_i(K^i, T^i, \mathbf{X}^{i-1}))$
if $j = s$ then
  $Y \leftarrow X_1^1 \,\|\, \cdots \,\|\, X_j^j$
else
  $T^{j+1} \leftarrow T \,\|\, X_1^1 \,\|\, \cdots \,\|\, X_j^j$; $\mathbf{X}^j \leftarrow X_{j+1}^j \,\|\, \cdots \,\|\, X_s^j$
  $X_{j+1}^{j+1}, \ldots, X_s^{j+1} \leftarrow \mathrm{par}_\gamma(\pi_{j+1}(T^{j+1}, \mathbf{X}^j))$
  $Y \leftarrow X_1^1 \,\|\, \cdots \,\|\, X_j^j \,\|\, X_{j+1}^{j+1} \,\|\, \cdots \,\|\, X_s^{j+1}$
return Y

Fig. 3. Hybrid oracles for $j \in [0\,..\,s]$ where $s = n/\gamma$ as used in the proof of Theorem 2. Tweakable permutation π_j is selected randomly from $\mathsf{Perm}(\mathcal{T}_{\gamma j}, n - \gamma j)$.

was mapped by the stage. The second stage processes the remaining $n - \gamma$ bits using a Γ-PRS on n bits. The tweak includes the γ bits output from the first stage. The resulting output fixes the second γ bits of the final output, and so on. In the last stage, only γ bits remain, and these are handled by the last Γ-PRS. A pictorial diagram is shown on the right-hand-side of Figure 1 in the introduction.

Security. We now turn to analyzing the security of the Icicle construction. The following theorem shows its security assuming the underlying separators enjoy Γ-PRS security.

Theorem 2. *Fix $n \geq \gamma \geq 1$ with $\gamma \,|\, n$, let $s = n/\gamma$ and let $\mathcal{Z} = \{Z_i\}_{i=1}^s$ be an appropriate collection of permutations for $\mathrm{Ic}_{\gamma,n}(\mathcal{Z})$. Let A be a cpa adversary making q queries. Then for the Γ-prs adversaries $B_1, \ldots, B_s$ specified in the proof below it holds that $\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{Ic}_{\gamma,n}(\mathcal{Z})}(A) \leq \sum_{j=1}^s \mathbf{Adv}^{\Gamma\text{-prs}}_{Z_i}(B_s)$. Each adversary B_j makes q oracle queries, runs in time at most that of A plus $\mathcal{O}(jq \cdot \mathsf{Time}(\mathcal{Z}) + (s-j)q \log(s-j)q)$.*

Proof. We introduce a sequence of hybrid oracles $\mathrm{Hyb}_0, \ldots, \mathrm{Hyb}_s$ as defined in Figure 3. The j^{th} hybrid Hyb_j implements $\mathrm{Ic}_{\gamma,n}(\mathcal{Z})$ but with the last $s - j$ stages replaced by an appropriately sized, tweakable random permutation. By construction Hyb_0 implements a random permutation while Hyb_s implements $\mathrm{Ic}_{\gamma,n}(\mathcal{Z})$. Then

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{Ic}_{\gamma,n}(\mathcal{Z})}(A) \leq \sum_{j=0}^{s-1} \left| \Pr\left[A^{\mathrm{Hyb}_{j+1}} \Rightarrow 1 \right] - \Pr\left[A^{\mathrm{Hyb}_j} \Rightarrow 1 \right] \right| \qquad (1)$$

We will now upper bound each individual difference in the sum, by way of adversaries $B_1, \ldots, B_s$ that attack the Γ-PRS security of the s underlying permutations in $\mathcal{Z}$. The adversaries are defined in Figure 4. In it, the adversary runs

adversary B_j^{Sep}:
$\mathtt{V}$ everywhere zero ; $\mathtt{Z}$ everywhere $\perp$
$K^1, \ldots, K^s \leftarrow\!\!\$\ \{0,1\}^{ks}$
Run A^{Enc}, answering queries (T, M) by:
  $X_1^0, \ldots, X_s^0 \leftarrow \mathrm{par}_\gamma(M)$
  for $i = 1$ to j:
    $T^i \leftarrow T \,\|\, X_1^1 \,\|\, \cdots \,\|\, X_{i-1}^{i-1}$; $\mathbf{X}^{i-1} \leftarrow X_i^{i-1} \,\|\, \cdots \,\|\, X_s^{i-1}$
    if $i < j$ then $X_i^i, \ldots, X_s^i \leftarrow \mathrm{par}_\gamma(Z_i(K_i, T^i, \mathbf{X}^{i-1}))$
    else $X_j^j, \ldots, X_s^j \leftarrow \mathrm{par}_\gamma(\mathrm{SepSim}(T^j, \mathbf{X}^{j-1}))$
  if $j = s$ then
    $Y \leftarrow X_1^1 \,\|\, \cdots \,\|\, X_j^j$
  else
    $T^{j+1} \leftarrow T \,\|\, X_1^1 \,\|\, \cdots \,\|\, X_j^j$; $\mathbf{X}^j \leftarrow X_{j+1}^j \,\|\, \cdots \,\|\, X_s^j$
    $X_{j+1}^{j+1}, \ldots, X_s^{j+1} \leftarrow \mathrm{par}_\gamma(\pi_{j+1}(T^{j+1}, \mathbf{X}^j))$
    $Y \leftarrow X_1^1 \,\|\, \cdots \,\|\, X_j^j \,\|\, X_{j+1}^{j+1} \,\|\, \cdots \,\|\, X_s^{j+1}$
  return Y
A outputs b'
return b'

subroutine $\mathrm{SepSim}(T', \mathbf{X})$
if $\mathtt{Z}[T', \mathbf{X}] = \perp$ then
  $X_j^j \leftarrow \mathrm{Sep}(T', \mathbf{X})$
  $\mathtt{Z}[T', \mathbf{X}] \leftarrow X_j^j \,\|\, \langle \mathtt{V}[T', X_j^j] \rangle_{n-j\gamma}$
  $\mathtt{V}[T', X_j^j] \leftarrow \mathtt{V}[T', X_j^j] + 1$
return $\mathtt{Z}[T', \mathbf{X}]$

Fig. 4. Adversaries for the proof of Theorem 2

for the first $j-1$ stages as in $\mathrm{Ic}_{\gamma,n}(\mathcal{Z})$ and then queries its own Γ-PRS oracle for the j^{th} stage. Note that the oracle only gives back γ bits.

The adversary therefore simulates the behavior of Z_j using the procedure SepSim, defined as follows. On input T', X The first γ bits X_j are set by querying $X_j^j \leftarrow \mathrm{Sep}(T', \mathbf{X}^{j-1})$. The remaining $n-\gamma$ bits are set to an arbitrary value so that the $\mathrm{SepSim}(T', \cdot)$ implements a permutation for each T'. Specifically, we set the last $n-\gamma$ bits to be equal to the value $\mathtt{V}[T, X_j^j]$, which is then incremented. (Note that $\mathtt{V}$ is set everywhere to zero initially.) This process ensures that we build a simulation of $Z_j(T', \cdot)$ that is a permutation yet agrees on the first γ bits with the output of SepSim.

The output of the first j stages, the γ-bit value returned by the PRS oracle, and the output of π_{j+1} (should $j < s$) combines to give the oracle response. We will now argue that

$$\Pr\left[A^{\mathrm{Hyb}_j} \Rightarrow 1\right] = \Pr\left[B_j^{Z_j(K,\cdot,\cdot)[\gamma]} \Rightarrow 1\right] \quad \text{and} \tag{2}$$

$$\Pr\left[A^{\mathrm{Hyb}_{j+1}} \Rightarrow 1\right] = \Pr\left[B_j^{\rho_j(\cdot,\cdot)} \Rightarrow 1\right]. \tag{3}$$

where $\rho_j \xleftarrow{\$} \mathsf{RegFunc}(\mathcal{T}_j, n_j, \gamma)$ for $\mathcal{T}_j = \mathcal{T} \times \{0,1\}^{(j-1)\gamma}$ and $n_j = n - (j-1)\gamma$. The first equation may seem to be immediate, but in fact is not because B_j does not simulate exactly Z_j for A when $j < s$. (When $j = s$ the simulation is indeed exact.) Nevertheless, for $j < s$ the distribution of values observed by A when Sep returns $Z_j(T', \mathbf{X})[\gamma]$ is distributed identically to when A's oracle uses Z_j directly. This is because the discrepancy between SepSim's outputs and what would have been computed by Z_j are hidden by π_{j+1}, and SepSim implements a permutation. This justifies (2).

To show the second part, we must argue that, when SepSim uses an oracle ρ_j, the result is that B_j implements Hyb_{j+1}. Here, we require that the composition of SepSim and π_{j+1} in B_j's simulation yields a random tweakable permutation. For each tweak $T' = T \,\|\, X_1^1 \,\|\, \cdots \,\|\, X_{j-1}^{j-1}$, gives rise to a different random function $\rho_j(T', \cdot)$ and permutation $\pi_{j+1}(T', \cdot)$. Moreover, $\mathrm{SepSim}(T', \cdot)$ implements a permutation and so π_{j+1} ends up mapping counter values 1-1 to random values. This justifies (3).

Combining (2) and (3) with (1) and the definition of Γ-prs advantage yields the advantage statement given in the theorem. ∎

Discussion. It is easy to generalize the construction above to work for heterogeneous mixes of separators, meaning that γ varies across stages. Moreover, we have above used a different separator with its own key for each stage, but we can also extend our results to use a variable-input-length separator that can be securely used with a single key on inputs of differing sizes. Finally we note that a corollary of Theorem 2 is that the icicle construction with $s = 2$ gives a way to extend the domain of a PRP on m-bits to one on $m + \gamma$ bits given a Γ-PRS on $m + \gamma$ bits.

6 The Mix-and-Cut Cipher

We now use the results of the prior sections to construct a new, full-security tweakable block cipher that we call Mix-and-Cut. To do so, we show that the Swap-or-Not cipher [13] is a secure 2-PRS for N queries. We then apply the Icicle construction with $\Gamma = 2$, and use Swap-or-Not for each of the n stages.

The Swap-or-Not cipher. Let $F : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ be a PRF family. The Swap-or-Not tweakable block cipher $E_{\mathrm{SN}} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ built from F is described on the left side of Figure 5. We emphasize that each round of Swap-or-Not results in two calls to F, one to generate the round key and the other to decide whether or not to swap. If F is implemented using CBC-MAC with AES, then a single call to F may result in multiple AES calls, depending on the lengths of the PRF inputs and how they are encoded. (One must also take care to use F that is secure for variable-length inputs.)

Hoang, Morris, and Rogaway [13] analyzed an ideal, untweaked version of this cipher with K_j independently random and the swap decision in each round made by a call to a random function G_j (right side Figure 5). We use the following

algorithm $E_{\mathrm{SN}}(K, T, M)$:
for $j = 1$ to r:
  $K_j \leftarrow F(K, \mathsf{roundkey} \,\|\, j \,\|\, T)$
  $M' \leftarrow K_j \oplus M$
  $\hat{X} \leftarrow \max(M, M')$
  If $F(K, \mathsf{decision} \,\|\, j \,\|\, T \,\|\, \hat{X})[0] = 1$
    $M \leftarrow M'$
Return M

algorithm $E_{\mathrm{SN}^*}(KG, M)$:
$(K_1, \ldots, K_r) \leftarrow \mathrm{par}_n(KG)$
for $j = 1$ to r:
  $M' \leftarrow K_j \oplus M$
  $\hat{X} \leftarrow \max(M, M')$
  If $G_j(\hat{X}) = 1$
    $M \leftarrow M'$
Return M

Fig. 5. (Left) The Swap-or-Not algorithm [13] and **(Right)** its ideal counterpart

restatement of [13, Th. 2], which gives a bound on the non-adaptive cpa security of E_{SN^*}.

Theorem 3 (Hoang-Morris-Rogaway). *Let $N = 2^n$ and let E_{SN^*} be the ideal version Swap-or-Not block cipher with r rounds as defined above. Let A be a non-adaptive cpa adversary making q queries. Then* $\mathbf{Adv}^{\mathrm{ncpa}}_{E_{\mathrm{SN}^*}}(A) \leq \frac{2 \cdot N^{3/2}}{r+2} \cdot \left(\frac{q+N}{2N}\right)^{r/2+1}$.

Unfortunately, the bound above is not very useful when $q \geq N - 1$. We will instead use it with $q = N/2$ in order to help us prove the following theorem, which establishes that E_{SN} is a secure 2-PRS for wN queries for some number w of tweaks.

Theorem 4. *Fix γ, n with $\gamma \,|\, n$ and let $\Gamma = 2^\gamma$, $N = 2^n$. Let E_{SN} be the Swap-or-Not block cipher with r rounds as defined above and using a function F. Let A be a Γ-PRS adversary making N queries across ω distinct tweaks. Then there exists an explicit prf-adversary B for which it holds that*

$$\mathbf{Adv}^{\text{2-prs}}_{E_{\mathrm{SN}}}(A) \leq \frac{2 \cdot \omega \cdot N^{3/2}}{r+2} \cdot \left(\frac{3}{4}\right)^{r/2+1} + \mathbf{Adv}^{\mathrm{prf}}_{F}(B) \,.$$

The adversary B makes at most $2rq\omega$ queries and runs in time that of A plus $\mathcal{O}(2r\omega N)$.

Proof. By Theorem 1 the advantage of A is upper bounded by the cpa advantage of an adversary B' against E_{SN} which makes $N/2$ fixed queries for each (adaptively chosen) tweak T. (Note that $E_{\mathrm{SN}} = E^{-1}_{\mathrm{SN}}$.) Thus

$$\mathbf{Adv}^{\text{2-prs}}_{E_{\mathrm{SN}}}(A) \leq \mathbf{Adv}^{\mathrm{cpa}}_{E_{\mathrm{SN}}}(B')$$

for an adversary B' explicitly specified in the proof of Theorem 1. We now move to a setting where E_{SN} uses, instead of the function F, a random function. This is via a standard reduction yielding that

$$\mathbf{Adv}^{\text{2-prs}}_{E_{\mathrm{SN}}}(A) \leq \mathbf{Adv}^{\mathrm{cpa}}_{E_{\mathrm{SN}}[\rho]}(B') + \mathbf{Adv}^{\mathrm{prf}}_{F}(B)$$

where $E_{\mathrm{SN}}[\rho]$ is E_{SN} except with $F(K, \cdot)$ replaced by a random function ρ. Note that $E_{\mathrm{SN}}[\rho]$ is not yet E_{SN^*} since the latter does not take tweaks. However $E_{\mathrm{SN}}[\rho]$

is equivalent to using E_{SN^*} with a fresh key KG for every tweak T queried by B'. Note also that the set of messages queried to each instance of E_{SN^*} is fixed. We can therefore use a hybrid argument in which one repeatedly applies the ncpa advantage for each independent instance of E_{SN^*} to get that

$$\mathbf{Adv}^{\mathrm{cpa}}_{E_{\mathrm{SN}}[\rho]}(B') \le \sum_{i=1}^{\omega} \mathbf{Adv}^{\mathrm{ncpa}}_{E_{\mathrm{SN}^*}}(B_i) \le \frac{2\cdot\omega\cdot N^{3/2}}{r+2} \cdot \left(\frac{3}{4}\right)^{r/2+1} .$$

The last inequality uses Theorem 3 for $q = N/2$ (the number of queries used by each B_i). ∎

Icicle applied to Swap-or-Not. We now explore the security of the icicle construction when E_{SN} is used as the underlying 2-PRS. Let $N = 2^n$ and let $\mathcal{Z}_{\mathrm{SN}} = \{E^i_{\mathrm{SN}}\}_{i=1}^n$ be a family of SN block ciphers where $E^i_{\mathrm{SN}} : \{0,1\}^k \times \mathcal{T}_i \times \{0,1\}^{n_i} \to \{0,1\}^{n_i}$ denotes the Swap-or-Not cipher as defined above where $n_i = n - (i-1)$ and $\mathcal{T}_i = \mathcal{T} \times \{0,1\}^{i-1}$. Given this, let $\mathrm{Ic}_{1,n}(\mathcal{Z}_{\mathrm{SN}}) : \{0,1\}^{kn} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ denote the block cipher on n bits with tweak space $\mathcal{T}$ constructed by using the icicle construction with $\mathcal{Z}_{\mathrm{SN}}$. For simplicity, we use this construction with the same number of rounds r of Swap-or-Not at each stage (meaning nr rounds of Swap-or-Not for the entire icicle construction). We refer to the resulting construction as the Mix-and-Cut cipher. We can prove the following:

Theorem 5. *Let $N = 2^n$ and $\mathrm{Ic}_{1,n}(\mathcal{Z}_{\mathrm{SN}})$ be the construction described above using r rounds for each Swap-or-Not cipher E^i_{SN}. Each cipher uses the same PRF F (with distinct keys). Let A be a cpa adversary making N queries with w distinct tweaks. Then for the prf adversary B given in the proof below it holds that*

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{Ic}_{1,n}(\mathcal{Z}_{\mathrm{SN}})}(A) < \frac{7\cdot w\cdot N^{3/2}}{r+2} \cdot \left(\frac{3}{4}\right)^{r/2+1} + n\cdot \mathbf{Adv}^{\mathrm{prf}}_F(B) .$$

B makes $2rnNw$ queries and runs in time that of A plus $\mathcal{O}(2rnNw)$.

Proof. Let $\alpha = (r+2)^{-1}\cdot(3/4)^{r/2+1}$. We first apply Theorem 2 and Theorem 4 to get that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{Ic}_{1,n}(\mathcal{Z}_{\mathrm{SN}})}(A) &\le \sum_{i=1}^{n} \mathbf{Adv}^{\text{2-prs}}_{E^i_{\mathrm{SN}}}(B_i) \\ &\le \sum_{i=1}^{n} 2\cdot w\cdot 2^{i-1}\cdot N_i^{3/2}\alpha + \sum_{i=1}^{n} \mathbf{Adv}^{\mathrm{prf}}_F(B_i) \qquad (4)\end{aligned}$$

where $N_i = 2^{n-(i-1)}$ and we have used that the number of tweaks queried against the i^{th} stage E^i_{SN} is $w\cdot 2^{i-1}$. First, a standard hybrid argument shows that the prf adversary B that chooses $j \leftarrow\!\!\$\ [1\,..\,n]$ and behaves as B_j is such that $\sum_{i=1}^n \mathbf{Adv}^{\mathrm{prf}}_F(B_i) \le n\cdot\mathbf{Adv}^{\mathrm{prf}}_F(B)$. Turning to analyze the first sum in the right hand side, we first have that

$$\sum_{i=1}^{n} 2^{i-1} \cdot N_i^{3/2} = \sum_{i=1}^{n} 2^{i-1} \cdot 2^{n-(i-1)} \cdot 2^{(n-(i-1))/2} = 2^n \sum_{i=1}^{n} 2^{i/2}$$

$$= 2^n \cdot \frac{2^{(n+1)/2} - \sqrt{2}}{\sqrt{2} - 1}$$

$$< 3.5N^{3/2} .$$

Plugging back into (4) yields the advantage statement of the theorem. ∎

Shuffling interpretation. We can view the cipher above as a shuffle by replacing all uses of the PRF with fresh random coin tosses. We refer to this as the Mix-and-Cut shuffle, and it is given in the left of Figure 1 in the introduction. To shuffle using Mix-and-Cut, "lightly" mix the entire deck in any fashion (e.g., using the Swap-or-Not shuffle for r rounds). Then, cut the deck in half. Lightly mix each half and then cut the halves, yielding four total piles. This process is repeated until all of the cards are in their own piles. At this point, the cards are simply gathered together to form one deck. This shuffle is oblivious assuming the mixing step is oblivious (in the sense of [20]).

Acknowledgements. The authors thank Phillip Rogaway for comments on an earlier draft of this paper as well as the anonymous Crypto 2013 reviewers for their valuable feedback. Ristenpart was supported in part by NSF grant CNS-1065134 and generous gifts from Microsoft and RSA Labs.

References

1. Bellare, M., Impagliazzo, R.: A tool for obtaining tighter security analyses of pseudorandom function based constructions, with applications to prp to prf conversion. Cryptology ePrint Archive, Report 1999/024 (1999), `http://eprint.iacr.org/`
2. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-preserving encryption. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 295–312. Springer, Heidelberg (2009)
3. Bellare, M., Rogaway, P., Spies, T.: Addendum to "the FFX mode of operation for format preserving encryption". Submission to NIST (September 2010)
4. Bellare, M., Rogaway, P., Spies, T.: The FFX mode of operation for format-preserving encryption. Submission to NIST (February 2010)
5. Black, J.A., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (2002)
6. Brier, E., Peyrin, T., Stern, J.: BPS: a format-preserving encryption proposal. Submission to NIST, `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf`
7. Brightwell, M., Smith, H.: Using datatype-preserving encryption to enhance data warehouse security. In: National Information Systems Security Conference, NISSC (1997)

8. Czumaj, A., Kanarek, P., Kutylowski, M., Lorys, K.: Fast generation of random permutations via networks simulation. In: European Symposium on Algorithms, pp. 246–260 (1996)
9. Durstenfeld, R.: Algorithm 235: Random permutation. Communications of the ACM 7(7), 420 (1964)
10. Fisher, R., Yates, F.: Statistical tables for biological, agricultural and medical research. Oliver & Boyd (1938)
11. Granboulan, L., Pornin, T.: Perfect block ciphers with small blocks. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 452–465. Springer, Heidelberg (2007)
12. Hall, C., Wagner, D., Kelsey, J., Schneier, B.: Building PRFs from PRPs. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 370–389. Springer, Heidelberg (1998)
13. Hoang, V.T., Morris, B., Rogaway, P.: An enciphering scheme based on a card shuffle. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 1–13. Springer, Heidelberg (2012)
14. Hoang, V.T., Rogaway, P.: On generalized feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 613–630. Springer, Heidelberg (2010)
15. Knuth, D.: The Art of Computer Programming, 3rd edn., vol. 2. Addison-Wesley (1997)
16. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
17. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing 17(2) (1988)
18. Morris, B.: Improved mixing time bounds for the Thorp shuffle. arXiv Technical Report 0912.2759 (2009), `http://arxiv.org/abs/0912.2759`
19. Morris, B., Rogaway, P., Stegers, T.: How to encipher messages on a small domain. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009)
20. Naor, M., Reingold, O.: On the construction of pseudorandom permutations: Luby-Rackoff revisited. Journal of Cryptology 12(1), 29–66 (1999)
21. Patarin, J.: Generic attacks on feistel schemes. Cryptology ePrint Archive, Report 2008/036 (2008), `http://eprint.iacr.org/2008/036`
22. Ristenpart, T., Rogaway, P.: How to enrich the message space of a cipher. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 101–118. Springer, Heidelberg (2007)
23. Stefanov, E., Shi, E.: Fastprp: Fast pseudo-random permutations for small domains. Cryptology ePrint Archive, Report 2012/254 (2012), `http://eprint.iacr.org/`