Format-Preserving Encryption

Mihir Bellare[1], Thomas Ristenpart[1], Phillip Rogaway[2], and Till Stegers[2]

[1] Dept. of Computer Science & Engineering, UC San Diego, La Jolla, CA 92093, USA
[2] Dept. of Computer Science, UC Davis, Davis, CA 95616, USA

Abstract. Format-preserving encryption (FPE) encrypts a plaintext of some specified format into a ciphertext of identical format—for example, encrypting a valid credit-card number into a valid credit-card number. The problem has been known for some time, but it has lacked a fully general and rigorous treatment. We provide one, starting off by formally defining FPE and security goals for it. We investigate the natural approach for achieving FPE on complex domains, the "rank-then-encipher" approach, and explore what it can and cannot do. We describe two flavors of unbalanced Feistel networks that can be used for achieving FPE, and we prove new security results for each. We revisit the cycle-walking approach for enciphering on a non-sparse subset of an encipherable domain, showing that the timing information that may be divulged by cycle walking is not a damaging thing to leak.

1 Introduction

Background. During the last few years, *format-preserving encryption* (FPE) has emerged as a useful tool in applied cryptography. The goal is this: under the control of a symmetric key K, deterministically encrypt a plaintext X into a ciphertext Y that has the same *format* as X. Examples include encryption of US social security numbers (SSNs), credit card numbers (CCNs) of a given length, 512-byte disk sectors, postal addresses of some particular country, and jpeg files of some given length. In our formalization of FPE, the format of a plaintext X will be a name N describing a finite set $\mathcal{X}_N$ over which the encryption function induces a permutation. For example, with SSNs this is the set of all nine-decimal-digit numbers.

The FPE goal is actually quite old. For one thing, a blockcipher itself can be seen as one kind of FPE: each N-bit string, where N is the block size, is mapped to some N-bit string. But what makes FPE an interesting and powerful idea is that the notion reaches far beyond blockciphers, which normally encipher strings of some one, convenient length.

Some prior work. In FIPS 74 (1981) [27], a DES-based approach is described to encipher strings over some fixed alphabet, say the decimal digits $\mathsf{D} = \{0, 1, \ldots, 9\}$. Each plaintext $X \in \mathsf{D}^N$ would be mapped to a ciphertext $Y \in \mathsf{D}^N$. Here each plaintext $X \in \mathsf{D}^*$ has a unique format $N = |X|$ and we must encipher X relative to the set $\mathcal{X}_N = \mathsf{D}^N$.

M.J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini (Eds.): SAC 2009, LNCS 5867, pp. 295–312, 2009.

Brightwell and Smith (1997) [6] considered a more general scenario, identifying what they termed *datatype-preserving encryption.* They wanted to encrypt database entries of some particular datatype without disrupting that datatype. A field containing an SSN (a nine-digit decimal string) should get mapped to another SSN. The authors colorfully explain the difficulty of doing this, saying that, with conventional encryption schemes, a "Ciphertext ... bears roughly the same resemblance to plaintext ... as a hamburger does to a T-bone steak. A social security number, encrypted using the DES encryption algorithm, not only does not resemble a social security number but will likely not contain any numbers at all" [6, p. 142]. The authors provide a proposed solution, though, as with FIPS 74, definitions or proofs for it are not likely or claimed.

Black and Rogaway [4] provided a provable-security investigation of a special case of FPE, asking how to make a cipher $E\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ with an arbitrary domain $\mathcal{X}$. Their solutions focused on $\mathcal{X} = \mathbb{Z}_N$, the integers $\{0, 1, \ldots, N-1\}$. The authors offer no general definition for FPE but they clearly intend that ciphers with domains of $\mathbb{Z}_N$ be used to construct schemes with other domains, like the set of valid CCNs of a given length.

The term *format-preserving encryption* is due to Terence Spies, Voltage Security's CTO [40]. Voltage, Semtek and other companies have been active in productizing FPE and explaining its utility [39]. FPE can enable a simpler migration path when encryption is added to legacy systems and databases, as required, for example, by the payment-card industry's data security standard (PCI DSS) [34]. Use of FPE enables upgrading database security in a way transparent to many applications, and minimally invasive to others. Spies has gone on to submit to NIST a proposed mechanism, FFSEM, that combines cycle walking and an AES-based balanced Feistel network [41].

Syntax. The current paper aims to help cryptographic theory "catch up" with cryptographic practice in this FPE domain. We initiate a general treatment of the problem, doing this within the provable-security tradition of modern cryptography.

We begin with a very general definition for FPE. Unlike a conventional cipher, an FPE scheme has associated to it a *collection* of domains, $\{\mathcal{X}_N\}_{N \in \mathcal{N}}$. We call each $\mathcal{X}_N$ a *slice* (the overall domain is their union, $\mathcal{X} = \bigcup_N \mathcal{X}_N$). The set $\mathcal{N}$ is the *format space*. For every key K, format N, and tweak T the FPE scheme E names a permutation $E_K^{N,T}$ on $\mathcal{X}_N$. We are careful to make FPEs tweakable [20] because, in this context, use of a tweak can significantly enhance security.

Returning to the CCN example, suppose we want to do FPE of CCNs with a zero Luhn-checksum [18]. Let's assume that the map should be length-preserving and that the possible lengths range from 12 to 19 decimal digits. Then we could let $\mathcal{N} = \{12, \ldots, 19\}$ and let $\mathcal{X}_N$ be the set of all N-digit numbers X such that $\mathsf{LuhnOK}(X)$ is true. Now an FPE scheme E with slices $\{\mathcal{X}_N\}_{N \in \mathcal{N}}$ does the job. You encrypt CCN X with key K and tweak T by letting $Y = E_K^{N,T}(X)$, where $N = \mathsf{len}(X)$.

Security notions. We define multiple notions of security for FPE schemes. Our strongest adapts the traditional PRP notion to capture the idea that FPE is a good approximation for a family of uniform permutations on the slices. Our weaker notions are denoted SPI, MP, and MR. SPI (single-point indistinguishability) is a variant of the PRP notion in which there is a only a single challenge point. MP (message privacy) lifts semantic security to the FPE setting by adapting earlier notions of deterministic encryption [3,5]. MR (message recovery) formalizes an adversary's inability to recover a challenge message, in its entirety, from the message's ciphertext. All of these notions can be made with respect to an adaptive or nonadaptive adversary, and can also be strengthened to allow chosen-ciphertext attacks (for PRP, this would result in what is called a *strong* PRP).

Why bother with SPI, MP, and MR when they are implied by PRP? SPI is useful because it is easy to work with and implies MP and MR with a tight bound. MP and MR are interesting because they, even in their nonadaptive form, are what an application will most typically need. An attack against the PRP notion may be no threat in practice, and achieving good PRP security may be overkill. Good concrete security bounds become particularly a focus when slices are small: a bound permitting $q \approx 2^{n/4}$ queries provides limited assurance when $n = 20$ bits.

Constructions. We next investigate the construction of FPE schemes. Suppose we wish to build an FPE scheme $\mathcal{E}$ with a complex *specification*—the slices $\{\mathcal{X}_N\}$ on which it should encipher. A natural approach is to arbitrarily number the points in each $\mathcal{X}_N$, say $\mathcal{X}_N = \{X_0, X_1, \ldots, X_{n-1}\}$ where $n = |\mathcal{X}_N|$. Then, to encipher $X \in \mathcal{X}_N$, find its index i in the enumeration, encipher i to j in $\mathbb{Z}_n$, and then return X_j as the encryption of X. We call this strategy the *rank-then-encipher* approach. It's the obvious, one could say folklore, approach. To implement it, we need an *integer* FPE that can encipher on $\mathbb{Z}_n$ for any needed n, as well a *ranking function*, *rank*, that maps each (N, X) with $X \in \mathcal{X}_N$ to a point in $\mathbb{Z}_n$ with $\mathit{rank}(N, \cdot)\colon \mathcal{X}_N \to \mathbb{Z}_n$ a bijection for all $N \in \mathcal{N}$.

We will show how to build ranking functions for any FPE problem whose domain is a regular language (the slices being strings of each possible length). This includes many practical problems. This can be extended to domains that are context-free languages having unambiguous grammars.

Our starting point for building integer FPEs is the construction of Black and Rogaway [4], which combines a generalization of an unbalanced Feistel network (the left and right hand side are numbers in $\mathbb{Z}_a$ and $\mathbb{Z}_b$ rather than strings) and a technique the authors call *cycle walking*, a method apparently going back to the rotor machines of the early 1900's [37]. We extend their work to handle multiple slices with the same key, and to incorporate tweaks.

The type of unbalanced Feistel network that was extended in [4] is the type due to Lucks [22]. It is not the only kind of unbalanced Feistel network. An equally natural possibility is the unbalanced Feistel design of Schneier and Kelsey [36]. Extended to $\mathbb{Z}_N$ where $N = ab$, we call this a *type-1* Feistel, as opposed to the *type-2* unbalanced Feistel network of [4,22]. Our FPE schemes FE1 and FE2,

based on type-1 and type-2 unbalanced Feistel networks, comprise a flexible, efficient, and customizable means for enciphering domains $\mathbb{Z}_N$ where $N = ab$ is the product of integers greater than one. Its round function can be based, for example, on AES. Combining FE1 or FE2 with the rank-then-encipher approach lets one achieve FPE in a wide variety of contexts.

Security. Ideally, we would like to prove good bounds on the strong-PRP security for FE1 and FE2, assuming the round function to be a good PRF. But we run into a limitation, namely that the proven strength of Feistel ciphers [4,21,24,26,28,29,30,31,32,33], in terms of quality of bounds, falls short of what is wanted, and what appears to be the actual strength of the techniques. We address this in a couple of ways.

First, proofs have always targeted PRP. Instead, we target MP and MR, thereby getting better bounds more easily. We prove that FE2 with only *three* rounds hides all partial information with respect to a nonadaptive chosen-plaintext attack: one achieves nonadaptive SPI, MP, and MR security with reasonable bounds. Even then, we feel that being guided purely by what can be proved would lead to an overly quite pessimistic security estimate. The most realistic picture may be obtained by also assessing resistance to attacks. We consider known attacks and discuss their implications for our parameter choices (principally the number of rounds). We also provide a novel attack against (heavily) unbalanced type-2 Feistel networks, one that achieves message recovery with success probability exponentially small in the number of rounds. The attack is damaging if the number of rounds is too small.

Finally, reaching beyond PRP/SPI/MP/MR security, we consider a particular kind of side-channel attack. The use of cycle-walking in the rank-then-encipher approach raises the fear of timing attacks: might the number of times one has to apply the underlying cipher leak adversarially valuable information? We prove that cycle-walking will *not*, on its own, give rise to timing attacks. This is because the correct distribution on the number of iterations of the cipher on any input can be computed by a simulator that does not attend to the inputs. Due to space constraints, we present this result in the full version only [1].

The future. We expect FPE to be increasingly deployed. The complex systems that process financial transactions impose a powerful legacy constraint. Using classical blockcipher-based modes would require far larger changes to these systems, which is costly and error-prone. FPE can be realized by simple, AES-based modes of operation, avoiding the need to design and review any fundamentally new primitive. Besides the enciphering of database fields, FPE may prove useful in networking applications, allowing datagrams to have their fields protected without changing their format. What one might lose in security when employing a *deterministic* encryption scheme can often be erased by sensibly tweaking the FPE scheme [20]. Moreover, such loss of security may be entirely overshadowed by the reduced need for random bits and disruption in infrastructure, protocols, and code.

2 FPE Syntax

Syntax. A scheme for *format-preserving encryption* (FPE) is a function $E\colon \mathcal{K}\times \mathcal{N}\times\mathcal{T}\times\mathcal{X}\to\mathcal{X}\cup\{\perp\}$ where the sets $\mathcal{K}$, $\mathcal{N}$, $\mathcal{T}$, and $\mathcal{X}$ are called the *key space*, *format space*, *tweak space*, and *domain*, respectively. All of these sets are nonempty and $\perp\notin\mathcal{X}$. We write $E_K^{N,T}(X)=E(K,N,T,X)$ for the encryption of X with respect to key K, format N, and tweak T. We require that whether or not $E_K^{N,T}(X)=\perp$ depends only on N,X and not on K,T, and let

$$\mathcal{X}_N=\{X\in\mathcal{X}\colon E_K^{N,T}(X)\in\mathcal{X}\text{ for all }(K,T)\in\mathcal{K}\times\mathcal{T}\}$$

be the N-indexed *slice* of the domain. We demand that a point $X\in\mathcal{X}$ live in at least one slice, $X\in\mathcal{X}_N$ for some N (if X is in no slice it should not be included in E's domain). We demand that there be finitely many points in each slice, meaning $\mathcal{X}_N$ is finite for all $N\in\mathcal{N}$. We require that $E_K^{N,T}(\cdot)$ be a permutation on $\mathcal{X}_N$ for any $(K,T)\in\mathcal{K}\times\mathcal{T}$. Its inverse $D\colon \mathcal{K}\times\mathcal{N}\times\mathcal{T}\times\mathcal{X}\to\mathcal{X}\cup\{\perp\}$ is defined by $D_K^{N,T}(Y)=D(K,N,T,Y)=X$ if $E_K^{N,T}(X)=Y$. In summary, an FPE enciphers the points within each of the (finite) slices that collectively comprise its domain.

A practical FPE scheme $E\colon \mathcal{K}\times\mathcal{N}\times\mathcal{T}\times\mathcal{X}\to\mathcal{X}\cup\{\perp\}$ must be realizable by efficient algorithms: an algorithm E to encrypt, an algorithm D to decrypt, and an algorithm to sample uniformly from the key space $\mathcal{K}$. Thus $\mathcal{K}$, $\mathcal{N}$, $\mathcal{T}$, and $\mathcal{X}$ should consist of strings or points easily encoded as strings, and E and D should return $\perp$ when presented a point outside of $\mathcal{K}\times\mathcal{N}\times\mathcal{T}\times\mathcal{X}$. We will not draw any distinction between an integer element of $\mathcal{X}$, say, and a string that encodes such a point.

The format of a point. Let $E\colon \mathcal{K}\times\mathcal{N}\times\mathcal{T}\times\mathcal{X}\to\mathcal{X}\cup\{\perp\}$ be an FPE scheme. Then we can speak of $X\in\mathcal{X}$ as having *format* N if $X\in\mathcal{X}_N$. One could associate to E a *format function* $\varphi\colon \mathcal{X}\to\mathcal{P}(\mathcal{N})\setminus\{\emptyset\}$ that maps each $X\in\mathcal{X}$ to its possible formats; formally, $\varphi(X)=\{N\in\mathcal{N}\colon X\in\mathcal{X}_N\}$.

Note that, under our definitions, a point may have multiple formats. But often this will not be the case: each $X\in\mathcal{X}$ will belong to exactly one $\mathcal{X}_N$. In that case we can regard the format function as mapping $\varphi\colon \mathcal{X}\to\mathcal{N}$ and interpret $\varphi(X)$ as *the* format of X. FPE is somewhat simpler to understand for such *unique-format* FPEs: you can examine an X and know from it the slice $\mathcal{X}_{\varphi(X)}$ on which you mean to encipher it. For a unique format FPE one can write $E_K^{T}(X)$ rather than $E_K^{N,T}(X)$ since N is determined by X.

Specifications. An FPE problem, as needed by some application, will specify the desired collection of slices, $\{\mathcal{X}_N\}_{N\in\mathcal{N}}$. It will also specify the desired tweak space $\mathcal{T}$. Typically it is easy to support whatever tweak space one wants, but it may be quite hard to support a given collection of slices $\{\mathcal{X}_N\}_{N\in\mathcal{N}}$ (indeed it may be hard to accommodate a single slice, depending on what it is). We therefore call the collection of slices $\{\mathcal{X}_N\}_{N\in\mathcal{N}}$ the *specification* for an FPE scheme. We will write $\mathcal{X}=\{\mathcal{X}_N\}_{N\in\mathcal{N}}$ for a specification, only slightly abusing notation because the domain $\mathcal{X}$ *is* the union of slices in $\{\mathcal{X}_N\}_{N\in\mathcal{N}}$. The question confronting the

cryptographer is *how to design an FPE scheme with a given specification*. We now provide some example possibilities.

EXAMPLES. **(1)** AES-128 can be regarded as an FPE with a single slice, $\{0,1\}^{128}$. The key space is $\mathcal{K} = \{0,1\}^{128}$ and the format space and tweak space are trivial (have size one). **(2)** To encipher 16-digit decimal numbers, take $\mathcal{X} = \{0,1,\ldots,9\}^{16}$ and just the one slice. **(3)** To encipher 512-byte disk sectors using an 8-byte sector index as the tweak, let $\mathcal{X} = \{0,1\}^{4096}$, $\mathcal{T} = \{0,1\}^{64}$, and just the one slice. **(4)** Suppose you want to encipher CCNs of 12–19 digits with a proper Luhn checksum, the ciphertext having the same length as the plaintext. Then the specification could be $\mathcal{X} = \{\mathcal{X}_N\}_{N\in\mathcal{N}}$ where $\mathcal{N} = \{12,13,\ldots,18,19\}$ and $\mathcal{X}_N$ is the set of all strings $X \in \{0,1,\ldots,9\}^N$ satisfying the predicate $\mathsf{LuhnOK}(X)$. Here $|\mathcal{X}_N| = 10^{N-1}$. **(5)** One nice FPE has slices that are $\{0,1\}^N$ for each $N \geq 0$. It allows length-preserving encryption of any binary string. **(6)** One can FPE rather unusual spaces. For example, slice $\mathcal{X}_N$ could encode all N-vertex graphs. Or $\mathcal{X}_N$ could be all valid C-programs on N bytes. Designing an efficient FPE with this specification might be impossible. All of the examples just given are unique-format FPEs. The following example is not.

INTEGER FPES. The specification for a particularly handy kind of FPE is the following. The slices are $\mathcal{X}_N = \mathbb{Z}_N$, for $N \in \mathcal{N} \subseteq \mathbb{N}$. This allows enciphering natural numbers with respect to any permitted modulus N. Assuming the tweak space is similarly rich, say $\mathcal{T} = \{0,1\}^*$, we call such scheme an *integer FPE*. When used within the rank-then-encipher paradigm, integer FPEs enable the construction of FPEs with quite complex specifications.

3 FPE Security Notions

GAMES. Our definitions and proofs use *code-based games* [2], so we first review that material. A game has an **Initialize** procedure, an optional **Finalize** procedure, and any number of additional procedures. A game G is executed with an adversary $\mathcal{A}$ as follows. First, **Initialize** executes, possibly returning an output s, and then $\mathcal{A}(\mathsf{run}, s)$ is run ($s = \varepsilon$ if **Initialize** returns no string). As $\mathcal{A}$ executes it may call any procedure G (but not **Initialize** or **Finalize**) provided by G. If there is no **Finalize** procedure, the output of $\mathcal{A}$ is the output of the game. If the game does specify a **Finalize**, then, when $\mathcal{A}$ terminates, $\mathcal{A}$'s output is **Finalize**'s input and the game's output is that of **Finalize**. Game procedures may call $\mathcal{A}(\mathsf{identifier}[,\ x])$, which invokes an instance of the caller with distinct coins for each distinct identifier. Conceptually, then, each identifier thus names a separate adversarial algorithm. State is not shared among them. Let $\mathrm{G}^{\mathcal{A}} \Rightarrow y$ denote the event that the game outputs y. We write $S \xleftarrow{\cup} x$ as shorthand for $S \leftarrow S \cup \{x\}$. Later we write $c \xleftarrow{+} d$ for $c \leftarrow c + d$.

Boolean variables, including *bad*, are silently initialized to FALSE, set variables to $\emptyset$, integer variables to 0. Games G and H are said to be identical-until-*bad* if their code differs only in the sequel of statements that first set *bad* to true. We say that "$\mathrm{G}^{\mathcal{A}}$ sets *bad*" for the event that game G, when

executed with adversary $\mathcal{A}$, sets *bad* to true. If G, H are identical-until-*bad* and $\mathcal{A}$ is an adversary then $\Pr\left[\mathrm{G}^{\mathcal{A}} \text{ sets } bad\right] = \Pr\left[\mathrm{H}^{\mathcal{A}} \text{ sets } bad\right]$. It is also standard ("the fundamental lemma") that if G, H are identical-until-*bad* then $\Pr\left[\mathrm{G}^{\mathcal{A}} \Rightarrow y\right] - \Pr\left[\mathrm{H}^{\mathcal{A}} \Rightarrow y\right] \leq \Pr\left[\mathrm{G}^{\mathcal{A}} \text{ sets } bad\right]$.

Security notions. We will extend the standard PRP notion to our setting, but we will also describe notions weaker than it, because they can be achieved with better proven concrete security for the same efficiency and, at the same time, they suffice for typical applications. Coming at it from the latter perspective, the most basic and often sufficient requirement is security against message recovery (MR), under either an adaptive or nonadaptive attack. We define this as well as a stronger notion of message privacy (MP) that requires that partial information about the message is not leaked by the ciphertext. We also consider a weakening of the PRP notion that we call SPI. The reason for considering this notion is that it is simpler than MP and MR to work with yet implies them; at the same time, it can be achieved with better concrete security bounds than we currently know how to get for the ordinary PRP notion.

In the following let $E\colon \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ be an FPE scheme. We consider the games in Figure 1. It is assumed that any query of the form (N, T, X) satisfies $N \in \mathcal{N}$, $X \in \mathcal{X}_N$, and $T \in \mathcal{T}$.

PRP security. The standard notion of PRP security is extended to FPE schemes via game PRP_E and the corresponding adversary advantage is

$$\mathbf{Adv}^{\mathrm{prp}}_E(\mathcal{A}) = 2 \cdot \Pr\left[\mathrm{PRP}^{\mathcal{A}}_E \Rightarrow \mathsf{true}\right] - 1 \,.$$

In the game $\mathrm{Perm}(\mathcal{X}_N)$ is the set of all permutations on $\mathcal{X}_N$.

SPI *security.* Single-point indistinguishability (SPI) requires that the adversary be unable to distinguish between the encryption of a single chosen message or a random range point, even when given adaptive access to a true encryption oracle. The formalization is based on game SPI_E. An adversary $\mathcal{A}$ is allowed to make only a single **Test** query, and this must be its first oracle query. Its associated advantage is

$$\mathbf{Adv}^{\mathrm{spi}}_E(\mathcal{A}) = 2 \cdot \Pr\left[\mathrm{SPI}^{\mathcal{A}}_E \Rightarrow \mathsf{true}\right] - 1 \,.$$

The SPI notion is closely related to (and inspired by) a definition originally from [12], variants of which were also considered in [9,25]. It is easy to see that PRP security implies SPI security, but there is an additive loss of q/M in the advantage bound, where q is the number of queries by the adversary and M is the minimum size of $\mathcal{X}_N$ over all $N \in \mathcal{N}$. This is perhaps unfortunate, but SPI is only used as a tool anyway. A hybrid argument following [9,12] shows that SPI security likewise implies PRP security. Here, $\mathbf{Adv}^{\mathrm{spi}}_E(\mathcal{A}) \leq q \cdot \mathbf{Adv}^{\mathrm{prp}}_E(B) + q^2/M$ where q is the number of **Enc** queries of starting prp adversary $\mathcal{A}$, and constructed spi adversary B makes $q - 1$ **Enc** queries.

Message recovery. An FPE scheme secure against message recovery is one for which an adversary is unable to recover plaintexts from ciphertexts, even given an encryption oracle and a favorable distribution of plaintexts, formats, and tweaks.

Initialize	**Enc**(N, T, X)	**Finalize**(b') // Game PRP_E
$b \xleftarrow{\$} \{0,1\}$; $K \xleftarrow{\$} \mathcal{K}$	if $b = 1$ then ret $E_K^{N,T}(X)$	ret $(b = b')$
for $(N, T) \in \mathcal{N} \times \mathcal{T}$	if $b = 0$ then ret $\pi_{N,T}(X)$	
do $\pi_{N,T} \xleftarrow{\$} \mathrm{Perm}(\mathcal{X}_N)$		

Initialize	**Test**(N^*, T^*, X^*)	**Finalize**(b') // Game SPI_E
$b \xleftarrow{\$} \{0,1\}$; $K \xleftarrow{\$} \mathcal{K}$	if $(N^*, T^*, X^*) \in S$ then	ret $(b = b')$
	ret $\perp$	
Enc(N, T, X)	$S \xleftarrow{\cup} (N^*, T^*, X^*)$	
if $(N, T, X) \in S$ then	if $b = 1$ then	
ret $\perp$	$Y^* \leftarrow E_K^{N^*,T^*}(X^*)$	
$S \xleftarrow{\cup} (N, T, X)$	else $Y^* \xleftarrow{\$} \mathcal{X}_{N^*}$	
ret $E_K^{N,T}(X)$	ret Y^*	

Initialize	**Enc**(N, T, X)	**Test** // Game MP_E
$K \xleftarrow{\$} \mathcal{K}$	ret $E_K^{N,T}(X)$	ret Y^*
$(N^*, T^*, X^*) \xleftarrow{\$} \mathcal{A}(\mathsf{dist})$		
$Y^* \leftarrow E_K^{N^*,T^*}(X^*)$	**Eq**(X)	**Finalize**(Z)
ret (N^*, T^*)	ret $(X = X^*)$	ret $(Z = \mathcal{A}(\mathsf{func}, X^*))$

Initialize	**Enc**(N, T, X)	**Test** // Game MR_E
$K \xleftarrow{\$} \mathcal{K}$	ret $E_K^{N,T}(X)$	ret Y^*
$(N^*, T^*, X^*) \xleftarrow{\$} \mathcal{A}(\mathsf{dist})$		
$Y^* \leftarrow E_K^{N^*,T^*}(X^*)$	**Eq**(X)	**Finalize**(X)
ret (N^*, T^*)	ret $(X = X^*)$	ret $(X = X^*)$

Fig. 1. Games used for defining FPE security notions SPRP, PRP, SPI, MP, and MR. Procedure $\mathcal{A}$, invoked by games MP and MR, denotes the caller of the game.

If the encryption were randomized we would require that the target ciphertext Y^* and encryption oracle E_K be of *no* use in recovering the plaintext, but this is too much to ask for with a deterministic encryption scheme, as an adversary can always encrypt candidate messages $X_1, \ldots, X_q$ to ciphertexts $Y_1, \ldots, Y_q$ and, if $Y_i = Y^*$ for some i, it will know that the target plaintext is $X^* = X_i$. Our security definition will formalize that this attack is (up to the adversary's advantage) the best one possible.

The idea is formalized as game MR_E in Figure 1. An MR-adversary $\mathcal{A}$ must begin with a **Test** query and have $Q_{\mathrm{Test}}(\mathcal{A}) = 1$ and $Q_{\mathrm{Eq}}(\mathcal{A}) = 0$, while a simulator $\mathcal{S}$ for $\mathcal{A}$ is an adversary that has $\mathcal{S}(\mathsf{dist}) = \mathcal{A}(\mathsf{dist})$, $Q_{\mathrm{Test}}(\mathcal{S}) = Q_{\mathrm{Enc}}(\mathcal{S}) = 0$ and $Q_{\mathrm{Eq}}(\mathcal{S}) = Q_{\mathrm{Enc}}(\mathcal{A})$. Here $Q_{\mathrm{Proc}}(\mathcal{C})$ is the maximum number of calls that adversary $\mathcal{C}$ might make to procedure **Proc**, the maximum over all coins of $\mathcal{C}$ and all possible oracle responses. The MR-advantage of adversary $\mathcal{A}$ is then defined as

$$\mathbf{Adv}^{\mathrm{mr}}_E(\mathcal{A}) = \Pr\left[\mathrm{MR}^{\mathcal{A}}_E \Rightarrow \mathsf{true}\right] - p_{\mathcal{A}}$$

where $p_{\mathcal{A}} = \max_{\mathcal{S}} \Pr\left[\mathrm{MR}^{\mathcal{S}}_E \Rightarrow \mathsf{true}\right]$ with the maximum over all simulators $\mathcal{S}$ for $\mathcal{A}$. Translating our formalism into English, an adversary making a **Test** query

and some number of **Enc**-queries could do just as well forgoing its **Test** query and trading its **Enc** queries for **Eq** queries.

In our experiment defining $p_{\mathcal{A}}$ it is easy to see *what* strategy an optimal $\mathcal{S}$ should use: it makes q **Eq**-queries, $X_1, \ldots, X_q$, where X_1 is a most likely point output by $\mathcal{A}(\mathsf{dist})$ for the known (N^*, T^*); X_2 is a second most likely point ($X_2 \neq X_1$); X_3 is a third most likely point ($X_3 \notin \{X_1, X_2\}$); and so on. If the **Eq**-oracle returns true for some X_i then $\mathcal{S}$ calls **Finalize**(X_i); otherwise, it calls **Finalize**(X_{q+1}) where $X_{q+1} \notin \{X_1, \ldots, X_q\}$ is the next most likely point after X_q. In this way $\mathcal{S}$ will win with probability $p_{\mathcal{A}} = \sum_{i=1}^{q+1} p_i$ where $p_i = \Pr[\mathcal{A}(\mathsf{dist}) \Rightarrow (N, T, X_i) \mid (N, T) = (N^*, T^*)]$.

Message privacy. In message privacy we are trying to measure the ability of an adversary with an encryption oracle to compute some function of a challenge plaintext X^* from its encryption C^*. If the encryption is randomized we would require that the challenge ciphertext C^* is of no use in such an attack. The formalization of this is semantic security [13]. For deterministic encryption, the intuition we aim to capture is that the adversary should do no better than it could if the encryption were ideal. In this case, the encryption oracle provides no more than the capability of testing whether a message of the adversary's choice equals the challenge message.

Our formalization closely resembles that for MR. A difference is that $\mathcal{A}$ is asked not only to come up with the distribution on plaintexts, but also the function on which it hopes to do well. See game MP in Figure 1. An MP-adversary $\mathcal{A}$ must begin with a **Test** query and have $Q_{\mathrm{Test}}(\mathcal{A}) = 1$ and $Q_{\mathrm{Eq}}(\mathcal{A}) = 0$, while a simulator $\mathcal{S}$ for $\mathcal{A}$ is an adversary that has $\mathcal{S}(\mathsf{dist}) = \mathcal{A}(\mathsf{dist})$, $Q_{\mathrm{Test}}(\mathcal{S}) = Q_{\mathrm{Enc}}(\mathcal{S}) = 0$, $Q_{\mathrm{Eq}}(\mathcal{S}) = Q_{\mathrm{Enc}}(\mathcal{A})$ and $\mathcal{S}(\mathsf{func}) = \mathcal{A}(\mathsf{func})$. The advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{mp}}_{E}(\mathcal{A}) = \Pr\left[\mathrm{MP}^{\mathcal{A}}_{E} \Rightarrow \mathsf{true}\right] - p_{\mathcal{A}}$$

where $p_{\mathcal{A}} = \max_{\mathcal{S}} \Pr\left[\mathrm{MP}^{\mathcal{S}}_{E} \Rightarrow \mathsf{true}\right]$ with the maximum over all simulators $\mathcal{S}$ for $\mathcal{A}$. Translating our formalism into English, an adversary making a **Test** query and some number of **Enc**-queries could do just as well in guessing $Z = \mathcal{A}(\mathsf{func}, X^*)$ forgoing its **Test** query and trading its **Enc** queries for **Eq** queries. Note that MR-security amounts to a special case of MP-security where the function $\mathcal{A}(\mathsf{func}, \cdot)$ is the identity function.

Relations between notions. One can pictorially describe the relationships between our four security notions like this:

$$\mathrm{PRP} \;\dashleftarrow\!\!\dashrightarrow\; \mathrm{SPI} \;\rightleftharpoons\; \mathrm{MP} \;\rightleftharpoons\; \mathrm{MR}$$

The solid arrows indicate tight implications and the broken arrows indicate lossy ones. We already noted the implications between PRP and SPI above. These can be shown to be the best possible, with the counter-example in the first case being a perfect FPE scheme and in the second case following [9]. We also noted that MP tightly implies MR. The non-obvious implication is that SPI tightly

implies MP, and is proved below. Finally, MP does not imply SPI, and MR does not imply MP. For the former separation, consider an FPE scheme that has a fixed point for all keys; for the latter separation, consider an FPE that always leaks a single bit of the plaintext. The proof of the following is given in the full version [1].

Proposition 1. **[SPI ⇒ MP]** *Let* $E\colon \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ *be an FPE scheme and let* $\mathcal{A}$ *be an MP adversary. Then there is an SPI adversary* $\mathcal{B}$ *such that* $\mathbf{Adv}^{\mathrm{mp}}_{E}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{spi}}_{E}(\mathcal{B})$. *In addition, adversary* $\mathcal{B}$ *runs in time that of* $\mathcal{A}$ *and* $Q_{\mathrm{Enc}}(\mathcal{B}) = Q_{\mathrm{Enc}}(\mathcal{A})$. □

Nonadaptive security, strong security. We expect that nonadaptive adversaries (the "static" security setting) are sufficient for many applications of FPE—the constructed scheme is not so much a tool as an end. We consider the class of static adversaries $\mathcal{S}$. An adversary $\mathcal{A} \in \mathcal{S}$, on input run, decides at the beginning of its execution the sequence of queries it will ask, their number and their kind being fixed. The relations between the non-adaptive notions of security remain the same as for their adaptive counterparts as described above.

In the other direction, the notions can be strengthened to require CCA-security. This is done by adding to the games a decryption procedure. In the PRP case, procedure $\mathbf{Dec}(N, T, Y)$ would return $D_K^{N,T}(Y)$ if $b = 1$ and $\pi^{-1}_{N,T}(Y)$ otherwise, where $D = E^{-1}$ denotes the inverse of E, as defined earlier. The resulting notion is the FPE analog of what is sometimes called strong-PRP (SPRP). In the games for SPI, MP and MR, $\mathbf{Dec}(N, T, Y)$ would return $D_K^{N,T}(Y)$. The adversary is not allowed to call it on inputs N^*, T^*, Y^* and the simulator is not allowed to call it at all.

Asymptotic notions. We can adapt our definitions to the asymptotic setting. We illustrate this for PRP-security. Recall first that, in speaking of complexity, we assume that $\mathcal{K}$, E, and D are all given by algorithms. Also, algorithm $\mathcal{K}$ took no input. We must slightly adjust the syntax of our FPE schemes. In particular, we provide $\mathcal{K}$ an input of the form 1^k. The algorithm must run in probabilistic polynomial time. Algorithm E and its inverse D must run in deterministic polynomial time in the sum of their input lengths. We then say that E is *PRP-secure* if, for any PPT adversary $\mathcal{A}$, the function $\varepsilon(k) = \mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{A}(1^k))$ is negligible, meaning $\varepsilon(k) \in k^{-\omega(1)}$. We emphasize that it is the key K output by $\mathcal{K}$ that, presumably, grows with the security parameter k; the specification $\mathcal{X} = \{\mathcal{X}_N\}$ does not grow with or otherwise depend on the security parameter.

4 The Rank-then-Encipher Approach

The idea. Suppose we want to build an FPE scheme $\mathcal{E}$ the slices of which may be quite complex. As an example, we might want to do length-preserving encryption of credit cards of various lengths, the CCNs of each length having a particular checksum and satisfying specified constraints on allowable substrings. It would be undesirable to design an encryption schemes whose internal

workings were tailored to the specialized task in hand. Instead, what one can do is this. First, arbitrarily order and then number the points in each slice, $\mathcal{X}_N = \{X_0, X_1, \ldots, X_{n-1}\}$ where $n = |\mathcal{X}_N|$. Then, to encipher $X \in \mathcal{X}_N$, find its index i in the enumeration, encipher i to j in $\mathbb{Z}_n$ using an integer FPE scheme, and then return X_j as the encryption of X. We call this strategy the *rank-then-encipher* approach. The method will be efficient if there is an efficient way to map each point X to its index i, to encipher i to j, and to map j back to the corresponding point X_j. Details now follow, attending more closely to formats and tweaks, and also allowing the enumeration used for mapping j to X_j to differ from that used for ranking.

Definitions. To formalize RtE encryption, we first define a *ranking* and an *unranking* function for a specification $\mathcal{X} = \{\mathcal{X}_N\}$. A ranking function is a map $\mathit{rank}\colon \mathcal{N} \times \mathcal{X} \to \mathbb{N} \cup \{\perp\}$ for which $\mathit{rank}_N(\cdot) = \mathit{rank}(N, \cdot)$ is a bijection from $\mathcal{X}_N$ to $\mathbb{Z}_{|\mathcal{X}_N|}$. In addition, $\mathit{rank}_N(X) = \perp$ if $N \notin \mathcal{N}$ or $X \notin \mathcal{X}_N$. An unranking function is a map $\mathit{unrank}\colon \mathcal{N} \times \mathbb{N} \to \mathcal{X} \cup \{\perp\}$ for which $\mathit{unrank}_N(\cdot) = \mathit{unrank}(N, \cdot)$ is a bijection from $\mathbb{Z}_{|\mathcal{X}_N|}$ to $\mathcal{X}_N$. In addition, $\mathit{unrank}_N(i) = \perp$ if $i \notin \mathbb{Z}_{|\mathcal{X}_N|}$.

For the asymptotic tradition, we say that a specification $\mathcal{X} = \{\mathcal{X}_N\}$ can be *efficiently ranked* if there are (deterministic) polynomial-time computable ranking and unranking functions for $\mathcal{X} = \{\mathcal{X}_N\}$. Polynomiality is in the sum of the input lengths. Note that the security parameter is not an input to the ranking or unranking functions, but it is already built in that larger slices may take more time to rank and unrank, as the input to these functions includes the format N.

The scheme. Suppose one aims to create an FPE scheme $\mathcal{E}$ with specification $\mathcal{X} = \{\mathcal{X}_N\}_{N \in \mathcal{N}}$. Let the desired tweak space for $\mathcal{E}$ be the set $\mathcal{T}$. Let $\mathbb{N}_0 = \{|\mathcal{X}_N| \colon N \in \mathcal{N}\} \subseteq \mathbb{N}$ be the sizes of the different slices. Then we can construct our desired FPE scheme $\mathcal{E}$ if we have in hand: **(1)** an integer FPE scheme $E\colon \mathcal{K} \times \mathbb{N}_0 \times \{0,1\}^* \to \mathbb{N}$ (it enciphers points in $\mathbb{Z}_n$ for each $n \in \mathbb{N}_0$), and **(2)** a ranking function *rank* and an unranking function *unrank* for $\mathcal{X} = \{\mathcal{X}_N\}_{N \in \mathcal{N}}$. Given such objects, define $\mathcal{E} = \mathrm{RtE}[E, \mathit{rank}, \mathit{unrank}]$ as the map $\mathcal{E}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ with

$$\mathcal{E}_K^{N,T}(X) = \mathit{unrank}_N(E_K^{|\mathcal{X}_N|, \langle N, T \rangle}(\mathit{rank}_N(X)))$$

when $X \in \mathcal{X}_N$, and $\mathcal{E}_K^{N,T}(X) = \perp$ otherwise. We call this *rank-then-encipher* approach. In words: convert the N-formatted string X to its corresponding number i; encipher $i \in \mathbb{Z}_{|\mathcal{X}_N|}$ to some $j \in \mathbb{Z}_{|\mathcal{X}_N|}$, employing a tweak that encodes both the format N of X and the tweak of $\mathcal{E}$; finally, convert j back to a domain point in $Y \in \mathcal{X}_N$ using a possibly unrelated enumeration of points.

We will omit formalizing and proving the rather obvious statements that, if E is secure with respect to the strong-PRP, PRP, SPI, MP, or MR notion of security, then so too will be the FPE scheme $\mathcal{E} = \mathrm{RtE}[E, \mathit{rank}, \mathit{unrank}]$, the reduction being tight and having time complexity that is approximately the sum of the times to perform the ranking and unranking.

By way of the rank-then-encipher approach, one can take an integer FPE (based, e.g., on the techniques described in [4]) and create from it an FPE with a quite intricate specification $\mathcal{X} = \{\mathcal{X}_N\}_{N\in\mathcal{N}}$.

For many specifications the needed ranking and unranking functions are simple to design and fast to compute: an *ad hoc* approach will work fine. But what can one say in general about the power of the rank-then-encipher FPE approach? We now turn our attention to this.

5 FPE for Arbitrary Regular Languages

The problem. Let Σ be a (finite) alphabet and let $L \subseteq \Sigma^*$ be a language over it. We say that an FPE scheme $E\colon \mathcal{K}\times\mathcal{N}\times\mathcal{T}\times\mathcal{X} \to \mathcal{X}\cup\{\perp\}$ is an FPE scheme *for* L if $\mathcal{X} = L$, $\mathcal{N} = \mathbb{N}$, and the slices are $\mathcal{X}_n = L_n = L \cap \Sigma^n$ for all $n \in \mathbb{N}$. In this section we show how to build an FPE for an arbitrary regular language L by describing how to compute a corresponding ranking and unranking function.

Why attend to regular languages? Many FPE specifications can be cast as asking for an FPE for a regular language. This is trivially true when the domain is finite. Some important domains are finite and without an easily summarized structure; a domain like "a valid postal address" is likely to be defined by a database such as the US Address Information System (AIS) and, given such a database, ranking is easy. Other finite domains are large but have a concise description as a regular language, either in terms of a regular expression or a DFA. For example, a US social security number is a string in the regular language $(0 \cup 1 \cup \cdots \cup 9)^9$. Alternatively, one may subtract from this any set of numbers that have not been assigned, such as those starting with an 8 or 9, having 0000 as the last four digits, or having 00 as the preceding two digits, but the resulting set will again have a concise description. For credit card numbers, a simple 20-state DFA M recognizes the language Luhn^R of strings that are the reversals of numbers with a valid Luhn checksum [18]. Namely, the DFA is $M = (Q, \Sigma, \delta, q_0, F)$ with states $Q = \mathbb{Z}_{10} \times \mathbb{Z}_2$, final states $F = \{0\} \times \mathbb{Z}_2$, start state $q_0 = (0, 0)$, and transition rule $\delta((a, b), d) = (a+2d+a\lceil d/5\rceil \bmod 10, 1-b)$. We will continue to use the $M = (Q, \Sigma, \delta, q_0, F)$ syntax below, following the convention of Sipser's book [38].

Rank computation for regular languages. We will describe efficient ranking and unranking functions for the specification $\mathcal{X} = \{\mathcal{X}_M\}$ where M is a DFA and $\mathcal{X}_M = L(M)$ is its language. First impose a total order $a_1 \prec \cdots \prec a_{|\Sigma|}$ on the elements of the alphabet $\Sigma = \{a_1, \ldots, a_{|\Sigma|}\}$ and extend this to the lexicographic order $\prec$ on each Σ^n. For $a \in \Sigma$ let $\mathsf{ord}(a)$ be the index i such that $a = a_i$ and for every $n \in \mathbb{N}$ let the ranking function be given by $\mathit{rank}_L(X) = |\{Y \in L\colon |X| = |Y| = n \text{ and } Y \prec X\}|$. We omit the argument $n = |X|$ because it is determined by X. Assume we have an integer FPE scheme E. Provided that we can efficiently compute each $\mathit{rank}_L(\cdot)$ and its inverse $\mathit{unrank}_L(\cdot)$, applying the RtE paradigm gives a practical FPE $\mathcal{E} = \mathrm{RtE}[E, \mathit{rank}_L, \mathit{unrank}_L]$ with $\mathcal{E}\colon \mathcal{K}\times\mathcal{N}\times\mathcal{T}\times L \to L\cup\{\perp\}$.

```
algorithm BuildTable(n)
for q ∈ Q do
   if q ∈ F then T[q, 0] ← 1
for i ← 1, ..., n do
   for q ∈ Q do
      for a ∈ Σ do
         T[q, i] +← T[δ(q, a), i − 1]
```

```
algorithm rank(X)
q ← q0 ; c ← 0 ; n ← |X|
for i ← 1, ..., n do
   for j ← 1, ..., ord(X[i]) − 1 do
      c +← T[δ(q, aj), n − i]
   q ← δ(q, X[i])
ret c
```

```
algorithm unrank(c)
X ← ε ; q ← q0 ; j ← 1
for i ← 1, ..., n do
   while c ≥ T[δ(q, aj), n − i] do
      c ← c − T[δ(q, aj), n − i] ; j +← 1
   X[i] ← aj ; q ← δ(q, x[i]) ; j ← 1
ret X
```

Fig. 2. Bottom left: Algorithm for computing the rank of a word in the regular language L of a DFA $M = (Q, \Sigma, \delta, q_0, F)$. **Top:** Initializing the table T. Each $T[\cdot, \cdot]$ starts at zero. **Bottom right:** How to compute the inverse of the ranking function.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing the regular language $L \subseteq \Sigma^*$. Let $X[i]$ denote the i-th character of $X \in \Sigma^*$ (numbering from the left and starting at 1). Extend δ to $Q \times \Sigma^*$ so that $\delta(q, X)$ is the state we end up in by starting from q and following $X \in \Sigma^*$. Formally, set $\delta(q, \epsilon) = q$ for all $q \in Q$ and recursively define $\delta(q, x) = \delta(\delta(q, X[1] \cdots X[n-1]), X[n])$ for all $q \in Q$ and all $X \in \Sigma^*$ with $n = |X| \geq 1$.

We compute the ranking function for M by dynamic programming, following [11]. Let $T[q, n]$ be the number of strings $X \in \Sigma^n$ such that $\delta(q, X) \in F$. The first algorithm of Figure 2, on input n, uses dynamic programming to compute, for all $q \in Q$ and $j \in [1\,..\,n]$, the number $T[q, j]$ of accepting paths of length j that start at q. The rank of a word in L can be computed based on T as shown by the second algorithm in Figure 2. The third algorithm in the figure computes the inverse, deriving a word in L by its rank. In the unit-cost model of computation, where arbitrary integer multiplications and additions are performed in unit time, rank_M and unrank_M can be computed in $O(|\Sigma| \cdot n)$ time, while the preprocessing step BuildTable(n) takes time $O(|Q| \cdot |\Sigma| \cdot n)$ time.

We comment that ranking can be further sped up to require about n sums instead of $n|\Sigma|$ by precomputing the needed partial sums, adding a third coordinate to T. The unranking function would need a binary search, or some other method, to map a number into the corrected (precomputed) interval $[0..\beta_1)$, $[\beta_1..\beta_2), \ldots, [\beta_{\sigma-1}, \beta_\sigma)$ that contains it, where $\sigma = |\Sigma|$. Regardless, ranking and unranking are linear-time for any regular language L, with modest constants in terms of the DFA representation of L.

On the importance of representations. It is important that we represented our regular language in terms of a DFA; had L been represented in terms of an NFA or a regular expression, we could not have efficiently computed the

ranking and unranking functions. In particular, remember that it is NP-hard (even PSPACE-hard) to decide if the language of an NFA M (or a regular expression α) is Σ^* [10, #AL1], [14]. Consequently, if P $\neq$ NP, we can't compute $unrank(2^n - 1)$ efficiently for all n, as such functionality would provide immediate means to decide if $L(M) = \Sigma^*$. Formally, if P$\neq$NP then $\mathcal{X}_M$ can't be efficiently ranked, where $\mathcal{X}_M = L(M)$ is the language of the NFA M. Note, however that this does not imply an inability to make an efficient FPE scheme for this specification—it only means that such a scheme could not use the RtE approach.

Ranking non-regular languages. Beyond regular languages, we can also apply the RtE approach with Mäkinen's ranking algorithm for the language generated by an unambiguous context-free grammar [23]. Efficient ranking algorithms exist for various other classes of combinatorial objects. For example, if we wish to encrypt the domain $\mathcal{X}_{n!}$ consisting of the set of permutations on n elements, the Lucas-Lehmer encoding [16] provides an efficient ranking. Other examples are spanning trees of a graph [7], B-trees [19], and Dyck languages [17]. Efficient rankings have also been studied in coding theory, starting with [8].

Given the ease of ranking regular languages and beyond, it is natural to ask if *every* language for which there is an efficient FPE scheme admits an RtE-style one. In the full version [1] we show that the answer is *no*. More specifically, we exhibit a specification $\mathcal{X} = \{\mathcal{X}_N\}_{N \in \mathcal{N}}$ where efficient FPE is possible but efficient ranking is not.

6 Feistel-Based Integer FPEs

We present two Feistel-based constructions of integer FPE schemes $E\colon \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ with format space $\mathcal{N} = \mathbb{N} \times \mathbb{N}$ and $\mathcal{X}$ such that $\mathcal{X}_N = \mathbb{Z}_{ab}$ for $N = (a, b)$ with $a \leq b$. Both are parameterized by the following: (1) a round function $F\colon \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$; and (2) a function $r\colon \mathcal{N} \to \mathbb{N}$ specifying the number of rounds.

Figure 3 defines encryption and decryption for the two integer FPE schemes FE1 and FE2. We refer to Feistel networks, such as FE1, that utilize the same kind of round function every round as type-1. Type-1 Feistel networks were previously treated in [26,36] for the case of bit strings. We refer to Feistel networks, such as FE2, that alternate the kind of round function as type-2. Type-2 Feistel networks for the case of bit strings are due to Lucks [22]. Type-2 Feistel networks with modular arithmetic were first used in [4].

Round functions. The round functions should be PRFs. It is not clear what this means when the range is the infinite set $\mathbb{N}$. To specify a round function, we will first specify a range function $w\colon \mathcal{N} \to \mathbb{N}$ such that for all $N \in \mathcal{N}$ we have $w(N) \geq b$ where $N = (a, b)$. The PRF advantage of an adversary $\mathcal{A}$ is then defined by

$$\mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{A}) = \Pr\left[\mathcal{A}^{F(K,\cdot,\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot,\cdot,\cdot)} \Rightarrow 1\right]$$

algorithm $\mathbf{FE1}_K^{N,T}(X)$

$(a,b) \leftarrow N\,;\ X_0 \leftarrow X$
for $i = 1,\ldots,r(N)$ do
  $L_{i-1} \leftarrow X_{i-1} \operatorname{div} b$
  $R_{i-1} \leftarrow X_{i-1} \bmod b$
  $W_i \leftarrow L_{i-1} + F_K(N,T,i,R_{i-1}) \bmod a$
  $X_i \leftarrow aR_{i-1} + W_i$
ret $X_{r(N)}$

algorithm $\mathbf{FD1}^{N,T}(Y)$

$(a,b) \leftarrow N\,;\ Y_{r(N)} \leftarrow Y$
for $i = r(N),\ldots,1$ do
  $W_i \leftarrow Y_i \bmod a$
  $R_{i-1} \leftarrow Y_i \operatorname{div} b$
  $L_{i-1} \leftarrow W_i - F_K(N,T,i,R_{i-1}) \bmod a$
  $Y_{i-1} \leftarrow aR_{i-1} + Z_i$
ret Y_0

algorithm $\mathbf{FE2}_K^{N,T}(X)$

$(a,b) \leftarrow N$
$L_0 \leftarrow X \bmod a\,;\ R_0 \leftarrow X \operatorname{div} a$
for $i = 1,\ldots,r(N)$ do
  If $i \bmod 2 = 1$ then $s \leftarrow a$ else $s \leftarrow b$
  $L_i \leftarrow R_{i-1}$
  $R_i \leftarrow L_{i-1} + F_K(N,T,i,R_{i-1}) \bmod s$
ret $sL_{r(N)} + R_{r(N)}$

algorithm $\mathbf{FD2}_K^{N,T}(Y)$

$(a,b) \leftarrow N$
If $r(N) \bmod 2 = 1$ then $s \leftarrow a$ else $s \leftarrow b$
$R_{r(N)} \leftarrow Y \bmod s\,;\ L_0 \leftarrow Y \operatorname{div} s$
for $i = r(N),\ldots,1$ do
  If $i \bmod 2 = 1$ then $s \leftarrow a$ else $s \leftarrow b$
  $R_{i-1} \leftarrow L_i$
  $L_{i-1} \leftarrow R_i - F_K(N,T,i,R_{i-1}) \bmod s$
ret $sR_0 + L_0$

Fig. 3. Top: Encryption and decryption algorithms for the integer FPE scheme FE1 where $K \in \mathcal{K}$, $T \in \mathcal{T}$, $F \in \mathcal{N}$, and $X, Y \in \mathcal{X}_F$. Here $x \operatorname{div} y$ is short-hand for $\lfloor x/y \rfloor$. **Bottom:** Encryption and decryption algorithms for the integer FPE scheme FE2.

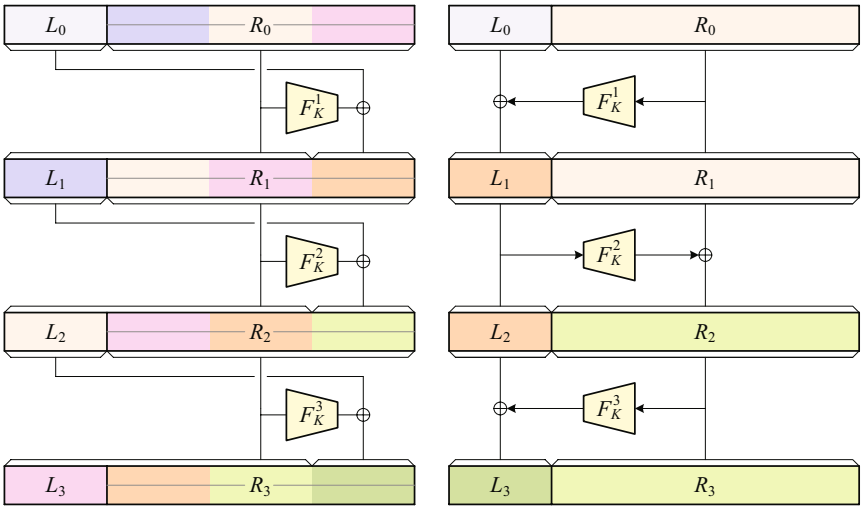

Fig. 4. Diagrams of three rounds of FE1 (**left**) and FE2 (**right**) for format $N = (a,b) = (2^{n_0}, 2^{n_1})$ and input $X \in \mathbb{Z}_{ab}$. For both mechanisms, $L_0, L_1, L_2, L_3 \in \mathbb{Z}_a$ and $R_0, R_1, R_2, R_3 \in \mathbb{Z}_b$.

where $\mathcal{A}$'s oracle in the second case returns a random point in $\mathbb{Z}_{w(N)}$ in response to a query F, T, i, X. Adversary $\mathcal{A}$ is not allowed to repeat an oracle query.

In cases of practical interest, we can build suitable round functions based on block ciphers (e.g. 3DES or AES) or cryptographic hash functions (e.g. SHA-256). In the full version [1] we detail example instantiations. We also discuss there the use of precomputation for speed improvements (deriving from the fact that several of the inputs to F are the same across all rounds).

Discussion. The round function takes as input the format and tweak, which effectively provides "separate" instances of the cipher for each format, tweak pair. To ensure independence between rounds, the round number is also input into the PRF.

FE1 and FE2 support domains of the form $\mathbb{Z}_{ab}$ and only provide security when $a > 1$. To handle arbitrary $\mathbb{Z}_n$ one can choose $N = (a, b)$ so that $ab > N$ and then utilize the cycle walking technique with FE1 or FE2 (see [4] for a treatment). Alternatively, one might utilize the off-by-one construction (see [4]) to avoid cycle-walking. But for typical applications like the encryption of credit card numbers, the requisite domains will be $\mathbb{Z}_n$ for which $n = ab$ for a and b that are almost balanced.

Security of FE1, FE2. In the full version [1] we discuss in detail the security of FE1 and FE2 in terms of best known attacks and proven security bounds. Beyond prior results, we give a novel MR attack that breaks FE2 when it is used with very unbalanced (a, b) and a relatively small number of rounds. We also give novel provable SPI security bounds for both schemes, which by Proposition 1 establishes MP and MR security.

Acknowledgments

Rogaway thanks Terence Spies for many useful discussions, and for sparking his interest in this topic. Rogaway and Stegers were supported by NSF grant CNS 0904380. Bellare and Ristenpart thank Clay Mueller, Lance Nakamura and Semtek for useful discussions and support.

References

1. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-Preserving Encryption. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 295–312. Springer, Heidelberg (2009)
2. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
3. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)
4. Black, J., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (2002)

5. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
6. Brightwell, M., Smith, H.: Using datatype-preserving encryption to enhance data warehouse security. In: 20th NISSC Proceedings, pp. 141–149 (1997), http://www.csrc.nist.gov/nissc/1997
7. Colbourn, C., Day, R., Nel, L.: Unranking and ranking spanning trees of a graph. Journal of Algorithms 10(2), 271–286 (1989)
8. Cover, T.: Enumerative source encoding. IEEE Transactions on Information Theory 19(1), 73–77 (1977)
9. Desai, A., Miner, S.: Concrete security characterizations of pRFs and pRPs: Reductions and applications. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 503–516. Springer, Heidelberg (2000)
10. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
11. Goldberg, A., Sipser, M.: Compression and Ranking. In: 17th Annual ACM Symposium on the Theory of Computing (STOC 1985), pp. 440–448. ACM Press, New York (1985)
12. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM 33(4), 792–807 (1986)
13. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
14. Hopcroft, J., Ullman, J.: Formal Languages and their Relation to Automata. Addison-Wesley, Reading (1969)
15. Jerrum, M.: A very simple algorithm for estimating the number of k-colorings of a low-degree graph. Random Structures and Algorithms 7(2), 157–165 (1995)
16. Knuth, D.: The Art of Computer Programming, 3rd edn. Seminumerical Algorithms, vol. 2. Addison-Wesley, Reading (1997)
17. Liebehenschel, J.: Ranking and unranking of a generalized Dyck language and the application to the generation of random trees. Séminaire Lotharingien de Combinatoire 43 (2000)
18. ISO/IEC 7812-1:2006. Identification cards – Identification of issuers – Part 1: Numbering system
19. Kelsen, P.: Ranking and unranking trees using regular reductions. In: Puech, C., Reischuk, R. (eds.) STACS 1996. LNCS, vol. 1046, pp. 581–592. Springer, Heidelberg (1996)
20. Liskov, M., Rivest, R., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
21. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal of Computing 17(2), 373–386 (1988)
22. Lucks, S.: Faster Luby-Rackoff ciphers. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 189–203. Springer, Heidelberg (1996)
23. Mäkinen, E.: Ranking and unranking left Szilard languages. Report A-1997-2, Department of Computer Science, University of Tampere (1997)
24. Maurer, U., Pietrzak, K.: The security of many-round Luby-Rackoff pseudo-random permutations. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 544–561. Springer, Heidelberg (2003)
25. Morris, B., Rogaway, P., Stegers, T.: How to encipher messages on a small domain: deterministic encryption and the Thorp shuffle. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009)

26. Naor, M., Reingold, O.: On the construction of pseudorandom permutations: Luby-Rackoff revisited. Journal of Cryptology 12(1), 29–66 (1999)
27. National Bureau of Standards. FIPS PUB 74. Guidelines for Implementing and Using the NBS Data Encryption Standard (April 1, 1981)
28. Patarin, J.: New results on pseudorandom permutation generators based on the DES Scheme. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 301–312. Springer, Heidelberg (1992)
29. Patarin, J.: Generic attacks on Feistel schemes. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 222–238. Springer, Heidelberg (2001)
30. Patarin, J.: Luby-Rackoff: 7 rounds are enough for $2^{n(1-\epsilon)}$ security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 513–529. Springer, Heidelberg (2003)
31. Patarin, J.: Security of random Feistel schemes with 5 or more rounds. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 106–122. Springer, Heidelberg (2004)
32. Patarin, J., Nachef, V., Berbain, C.: Generic attacks on unbalanced Feistel schemes with contracting functions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 396–411. Springer, Heidelberg (2006)
33. Patel, S., Ramzan, Z., Sundaram, G.: Efficient constructions of variable-input-length block ciphers. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 326–340. Springer, Heidelberg (2004)
34. PCI Security Standards Council. Payment Card Industry Data Security Standard Version 1.2, `https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml`
35. Petrank, E., Rackoff, C.: CBC MAC for real-time data sources. J. of Cryptology 13(3), 315–338 (2000)
36. Schneier, B., Kelsey, J.: Unbalanced Feistel networks and block cipher design. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 121–144. Springer, Heidelberg (1996)
37. Schroeppel, R.: Personal communication, approximately (2001)
38. Sipser, M.: Introduction to the Theory of Computation, 2nd edn. Thomson Press (2006)
39. Spies, T.: Format preserving encryption. Unpublished white paper, `www.voltage.com` Database and Network Journal (December 2008), Format preserving encryption: `www.voltage.com`
40. Spies, T.: Personal communications (February 2009)
41. Spies, T.: Feistel finite set encryption mode. Manuscript, posted on NIST's website on (February 6, 2008), `http://www.csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffsem/ffsem-spec.pdf`
42. Valiant, L.: The complexity of computing the permanent. Theoretical Computer Science 8, 189–201 (1979)