# Format-Preserving Encryption

Mihir Bellare[*] Thomas Ristenpart[†]

May 2009

**Abstract**

In the encryption of credit-card data as well as other applications, we may want to encipher in such a way that a certain property of the plaintext is preserved in the ciphertext. This paper initiates a treatment of this type of format-preserving encryption. We introduce a primitive that we call a general cipher that allows us to capture encryption preserving *arbitrary* formats. We specify an as-strong-as-possible notion of security for it that says that none but the desired property is leaked. We then provide an efficient construction of a general cipher that we call FPF.

**Keywords:** credit-card encryption, format-preserving encryption

[*] Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`.

[†] Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: `tristenp@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/tristenp`.

## Contents

# 1 Introduction

A *cipher* with domain $D$ is a deterministic function $E\colon \mathcal{K} \times D \to D$ that associates to any key $K \in \mathcal{K}$ a permutation $E_K\colon D \to D$. It is traditional to ask for ciphers whose ciphertexts hide *all* properties of the input, as formalized by asking that ciphers be secure pseudorandom permutations (PRP). However, important applications mandate that some properties of a plaintext *are* leaked by a ciphertext. The best known example of this is when format needs to be preserved, meaning a certain property of the plaintext must also be present in the ciphertext. This paper initiates a treatment of this type of format-preserving encryption. We introduce a primitive that we call a general cipher that allows us to capture encryption preserving *arbitrary* formats. We specify an as-strong-as-possible notion of security for it that says that none but the desired property is leaked. We then provide an efficient construction of a general cipher. Before giving more details, let us step back and explain the need for this new primitive via example.

**Format-preserving encryption.** Information stored in database records is often expected to have a certain format, meaning be of a certain type or be structured in a certain way. Software that operates on the data will complain and not function properly if it encounters data not of the expected format. An important example is credit card data. Here customer records include such data as names, addresses, and the credit card number itself, each expected to meet certain restrictions. For example, a name $\mathsf{Name}$ must consist of two strings of letters, each being an upper case letter followed by lower case letters. An address $\mathsf{Addr}$ must be a sequence of alphanumeric characters, spaces, or commas followed by a valid postal code. A credit card number $\mathsf{CC}$ must be 8–19 decimal digits followed by a valid Luhn digit [10]. (The Luhn digit is an error-correcting code computed as a function $\mathsf{L}$ of the preceding digits of the credit card number.) Two example records are shown below.

| | | |
|---|---|---|
| John Doe | 1234 Market Street, San Francisco, CA 94105 | 5588219104134593 |
| Sarah Jane | 5678 7th Avenue, New York, NY 10027 | 370371000012341 |

Software managing (or using) the database will fail (perhaps subtly) if any of these restrictions are not met. For example, the database software might check validity of Luhn digits before allowing new records to be written to it.

Now enter the security concerns. Compromise of servers storing databases such as the above results in credit card fraud that is costing the industry millions of dollars per year. Encryption has become not just desirable but mandatory due to recent standards such as the payment card industry's data security standard [22]. But an arbitrary encryption method won't do. There is a new constraint, namely that the encryption be format-preserving encryption (FPE). This means that if the plaintext has some prescribed format, so must the ciphertext. For our example above, we need a cipher that maps an input (plaintext) of the form $\mathsf{Name}$, $\mathsf{Addr}$, $\mathsf{CC}$ to an output (ciphertext) of the form $\mathsf{Name}^*, \mathsf{Addr}^*, \mathsf{CC}^*$. $\mathsf{Name}^*$, like $\mathsf{Name}$, must consist of two strings of letters each beginning with an upper case letter. $\mathsf{Addr}^*$, like $\mathsf{Addr}$, must consist of alphanumeric characters, spaces, or commas followed by a valid postal code. $\mathsf{CC}^*$, like $\mathsf{CC}$, must consist of 8–19 decimal digits followed by the output of the function $\mathsf{L}$ when applied to those digits. Furthermore, the ciphertext must be of the same length as the plaintext. This rules out randomized encryption such as CBC or CTR modes, where an IV must be included in the ciphertext.

Why format preservation? The complex systems that process financial transactions impose a powerful legacy constraint. Using a classical encryption scheme would require changing these systems, which is costly and error-prone. Instead, we want to be able to replace the plaintext by the ciphertext in the database without having the system "break". That is, the ciphertext must occupy the same space as the plaintext and have the format necessary to be accepted by the software. Format-preserving encryption accomplishes this and thus provides security with minimal cost overhead.

General ciphers. To solve the FPE problem we introduce a new primitive that we call a general cipher. Unlike a conventional cipher, a *general cipher* has associated to it a family $\{\mathsf{Dom}(\ell)\}_{\ell \in I}$ of domains where $I$ is some index set. For every key $K$, index $\ell$, and tweak $T$ the cipher specifies a permutation $\mathcal{E}_K^{T,\ell}$ on $\mathsf{Dom}(\ell)$. (We make general ciphers tweakable [12] because, as we will show, this feature will provide enhanced security.) We then provide a construction (called FPF) of a general cipher. We stress that the construction is able to produce a cipher over an *arbitrary* given domain $\{\mathsf{Dom}(\ell)\}_{\ell \in I}$, which (as we will now explain) is what enables us to implement FPE preserving *arbitrary* formats.

So how does a general cipher allow us to do FPE? Let us illustrate by restricting attention to the format-preserving encryption of credit card numbers from our example above. Recall we want the ciphertext $\mathsf{CC}^*$, like the plaintext $\mathsf{CC}$, to be a sequence of digits whose last digit is the value of the function $\mathsf{L}$ applied to the other digits. We also want the length $\mathsf{len}(\mathsf{CC}^*)$ of $\mathsf{CC}^*$ to be the same as the length $\mathsf{len}(\mathsf{CC})$ of $\mathsf{CC}$. Say the length ranges from 8 to 20 digits. Let $I = \{8, 9, \ldots, 20\}$ and let $\mathsf{Dom}(\ell)$ be the set of all $\ell$-digit numbers whose last digit is the value of $\mathsf{L}$ applied to the other digits. Now a general cipher $\mathcal{E}$ over $\{\mathsf{Dom}(\ell\}_{\ell \in I}$ does the job! Encrypt $\mathsf{CC}$ (under key $K$ and tweak $T$) by letting $\ell = \mathsf{len}(\mathsf{CC})$ and letting the ciphertext be $\mathsf{CC}^* = \mathcal{E}_K^{T,\ell}(\mathsf{CC})$, which has the desired format.

What is less obvious is that this approach extends to cover more complex formats. In Section 3 we show how to specify a domain that captures the full example we discussed above. (Case of letters and validness of postal codes are amongst the things to preserve.) From this it should be clear that one can do format-preserving encryption in general.

FPF. What remains is to be able to construct a general cipher over an arbitrary, given domain. Our FPF (Format-Preserving Feistel) construction does just this. Our starting point is the arbitrary domain cipher of Black and Rogaway [6] which combines a generalization of an unbalanced Feistel network with a technique from [24] called cycle walking. FPF extends this to handle multiple domains with the *same* key and also to incorporate tweaks. FPF is flexible, efficient, and customizable. The round function is a parameter and can be based on a block cipher such as AES or DES, or on a cryptographic hash function such as SHA-256.

Security notions. The first issue in security is to pick and then formally define the security goals of a general cipher. The novel element here is that some information about the plaintext (namely the format) *is* leaked by the ciphertext. Our strongest notion of security adapts the traditional PRP notion to general ciphers to capture no more than the format being leaked. But we also provide a weaker message privacy notion MP adapted from [4, 9] and a still weaker notion MR of security against message recovery. Why bother with MP and MR when they are implied by PRP? The reason is that MP and MR are what applications really need (an attack against the PRP notion may be absolutely no threat in practice!) and PRP can be a costly overkill. This could be because MP and MR are more efficiently achievable. It could also be because of concrete security. (Whether provable or as indicated by cryptanalysis.) Even for a particular construction, the concrete security under MP and MR may be better than PRP. (In the latter case a lot of security is often lost due to birthday attacks that don't threaten MP or MR.) This is particularly important in our context because domains may be small. (For example we may want to encrypt only 12 digits of a credit card number.) We need all the security we can get. We will see all these issues show up for FPF.

Security of FPF. We would ideally like to prove security of FPF under suitable assumptions on the round function. But we run into a limitation, namely that the proven strength of Feistel ciphers [13, 15, 16, 19, 20, 6], in terms of quality of bounds, falls short of what seems to be the actual strength indicated by resistance to attacks. (At least, for Feistel with enough rounds.) This is well understood in the community. We address this in a couple of ways.

First, proofs have always targeted PRP. Instead we target MP and MR and prove better bounds. We prove that FPF, with only 5 rounds, hides all partial information about inputs. To the best of our knowledge, our bound is the only known positive security result for unbalanced Feistel networks with constant number of rounds and message size that are meaningful beyond $q = \sqrt{2^{\kappa}}$ queries where $\kappa$ is the logarithm of the size of the round function (with smallest range). (See Section 5.3 for a detailed discussion of the prior known bounds from [17, 19, 20, 6, 15, 13].) Due to the wide use of Feistel networks, we expect that this result is of independent interest.

Even with this, we feel that being guided purely by what can be proved would lead us to pessimistic security estimates. We believe the most realistic picture is attained by 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 Feistel networks that achieves message recovery with success probability exponentially small in the number of rounds. (The attack can only be damaging if the number of rounds is too small.)

In conclusion, and broadly speaking, the status of FPF is that with enough rounds it appears to be secure. If the Feistel is balanced or almost balanced, we would suggest a minimum of 7 rounds, and preferably more, say a dozen. As the imbalance increases, so should the number of rounds.

ADE. Arbitrary-Domain Encryption (ADE) is the problem of building a *classical* cipher $E\colon \mathcal{K} \times D \to D$ over a single but arbitrary, given domain $D$. A general cipher yields this as a special case by letting the domain $\mathsf{Dom}$ consist of the single set $D$. FPF in this way yields a classical cipher over an arbitrary domain. This turns out to have some advantages over known constructs. To say more let us give some background.

There have been two approaches to the problem of designing ciphers over arbitrary (and small) domains. The first is to design the cipher from scratch. This is represented by Hasty Pudding [24]. The drawback is that users (such as financial institutions, which are very conservative) prefer something based on well-known and standardized algorithms. The second approach, due to Black and Rogaway [6] and mentioned above, is to combine a Feistel construction with a technique from [24] called cycle walking. The round function would be based on a standard algorithm such as AES. A prominent instantiation of this method is Spies' FFSEM which has been proposed to NIST as a standard [25].

The problem considered by these methods is to build a cipher with domain $\mathbb{Z}_n$ for some (arbitrary) positive integer $n$. FFSEM lets $l$ be the smallest integer such that $2^{2l} \geq n$ and builds a balanced Feistel cipher with domain $\{0,1\}^{2l}$. FFSEM applies the Feistel cipher to its input, and, if the output is not in $\mathbb{Z}_n$, it iterates, applying the Feistel cipher to this output, continuing until it obtains a point in the desired domain. This is the cycle walking, which is shown by [6] to always terminate.

FPF does not use cycle-walking. The technique to avoid cycle-walking in fact goes back to Black-Rogaway but was not used in the proposed standard FFSEM [25]. For the same security (meaning, the same number of Feistel rounds) this means that FPF is significantly faster. For example, suppose we are enciphering five digit numbers. FFSEM will let $l$ be the smallest integer such that $2^{2l} \geq 10^5$, obtaining $l = 9$, and then perform cycle-walking with a $2l$-bit Feistel cipher. The expected number of applications of the Feistel cipher is $2^{2l}/10^5 \approx 2.6$. This means that FFSEM is 2.6 times more expensive than FPF on the average, and even more expensive than that in some cases. Indeed, the running time of FFSEM is a random variable depending on the input. Besides the uncertainties that this brings is the danger of increased vulnerability to timing attacks. FPF, in contrast, has a fixed running time.

## 2 General Ciphers

We first provide a formal definition and then explanations.

General ciphers. A general cipher is a a tuple $\mathcal{GE} = (\mathcal{E}, \mathsf{KeySp}, \mathsf{TwSp}, I, \mathsf{Dom})$. The sets $\mathsf{KeySp}$ and $\mathsf{TwSp}$ are the key space and tweak space, respectively. The *domain* $\mathsf{Dom}$ is a map that given a value $\ell \in I \subseteq \{0,1\}^*$ returns a finite, non-empty set $\mathsf{Dom}(\ell)$ where $I$ is an index set. For each $K \in \mathsf{KeySp}$, $T \in \mathsf{TwSp}$, and $\ell \in I$ we let the function $\mathcal{E}_K^{T,\ell}$ be defined by $\mathcal{E}_K^{T,\ell}(x) = \mathcal{E}(K, T, \ell, x)$ for all $x \in \mathsf{Dom}(\ell)$. We require that $\mathcal{E}_K^{T,\ell}$ is a permutation on $\mathsf{Dom}(\ell)$. We say that $\mathcal{GE}$ has a numeric domain if there is a function $s\colon\ I \to \mathbb{N}$, called the size function, such that $\mathsf{Dom}(\ell) = \mathbb{Z}_{s(\ell)}$ for all $\ell \in I$.

Discussion. The main novelty here is that the domain is a collection of sets rather than a single one as in conventional definitions of a cipher. For any fixed key and tweak, a general cipher must induce a permutation on the domain set associated to any given index. Why this model? Think of the index as specifying some to-be-preserved property of the plaintext. That is, associate to any input plaintext $x$ a "property" $P(x)$ that can take many possible values $V_1, V_2, \ldots$. Then we can let $\mathsf{Dom}(\ell)$ be the set of all inputs $x$ such that $x$ has this property, i.e. $P(x) = V_\ell$. Inputs having a certain property are then mapped to outputs with the same value of $P$ so that the cipher preserves property $P$. One example is that the inputs are drawn from $\Sigma^*$ for some alphabet $\Sigma$ and $P(x)$ is the length of $x$. In this case the cipher is length-preserving. It is in this way that we capture the length-preserving encryption of variable-length inputs. However, the framework is clearly more general, allowing us to talk of the preservation of other properties as well. The examples in Section 3 will illustrate further.

The role of tweaks. Before continuing, we first pause to explain the role of tweaks in providing improved security. For format-preserving encryption, there will often be public data associated with records (e.g. a record number or the last four digits of a credit card number, which is often necessarily left public). This associated data can be input as a tweak to encryption and decryption. This can significantly improve security, because brute-force message privacy or recovery attacks will have to attack each tweak independently, slowing down attacks in practice significantly. This is analogous to the concept of salting in UNIX password hashing.

Encode-then-encipher. It is convenient for constructions to work with numeric domains. Arbitrary domains are handled via an encode-then-encipher paradigm. Namely, suppose we have a general cipher $\mathcal{GE} = (\mathcal{E}, \mathsf{KeySp}, \mathsf{TwSp}, I, \mathsf{Dom})$ with numeric domain $\mathsf{Dom}(\ell) = \mathbb{Z}_\ell$ for $\ell \in I$. We want to build a general cipher $\overline{\mathcal{GE}} = (\bar{\mathcal{E}}, \mathsf{KeySp}, \mathsf{TwSp}, I, \overline{\mathsf{Dom}})$. We first specify an injective map $\mathsf{Enc}$ such that $\mathsf{Enc}(\ell, \cdot)\colon\ \overline{\mathsf{Dom}}(\ell) \to \mathbb{Z}_{s(\ell)}$, where $s(\ell) = |\overline{\mathsf{Dom}}(\ell)|$, for all $\ell \in I$. For $K \in \mathsf{KeySp}$, $T \in \mathsf{TwSp}$, $\ell \in I$, and $X \in \overline{\mathsf{Dom}}(\ell)$ we then let

$$\bar{\mathcal{E}}_K^{T,\ell}(X) = \mathsf{Dec}(\ell, \mathcal{E}_K^{T,\ell}(\mathsf{Enc}(\ell, X)))$$

where $\mathsf{Dec}(\ell, \cdot)$ is the inverse of $\mathsf{Enc}(\ell, \cdot)$ for $\ell \in I$. In this way, our task is broken up into building general ciphers over numeric domains and then capturing specific applications via appropriate encoding functions. The examples of Section 3 will illustrate.

# 3 Format-Preserving Encryption using General Ciphers

In this section, we give several examples of format-preserving encryption using a general cipher. These will show how general ciphers can capture requirements and format-preservation constraints that cannot be captured in the framework of enciphering over a (single) finite set [6, 25]. We present three examples. The first performs length-preserving encryption of (varying length) credit card numbers. The second provides a general method for encrypting multi-alphabet strings. The third shows how to use this multi-alphabet string cipher to perform format-preserving encryption of user records as per the example discussed in the introduction. Our technique throughout will be to utilize the encode-then-encipher paradigm discussed earlier. For this section we assume access to a general cipher $\mathcal{GE} = (\mathcal{E}, \mathsf{KeySp}, \mathsf{TwSp}, I, \mathsf{Dom})$ with $\mathsf{TwSp} \subseteq \{0,1\}^*$ and with index set and domain sufficient for the examples. (In the next section, we show how to realize such a general cipher.)

Example 1: credit card numbers. Here the inputs are strings over $\Sigma = \{0, 1, \ldots, 9\}$ where length $\ell$ can very, say from 4 to 20. (One might want to encrypt varying amounts of the credit card numbers and discretionary data, leaving the rest in place.) The output must be a string over $\Sigma^*$ of the same length as the input. (Note we are ignoring the functioning of the Luhn digit here. In Example 3 we will account for requiring valid Luhn digits.) Tweaks would be some of the (un-encrypted) credit card number digits encoded into bit strings in a natural way. We can capture the requirements of this setting by letting $\mathsf{Dom}_{cc}(\ell) = \Sigma^\ell$ for all $\ell \in I = \{4, \ldots, 20\}$. We seek a general cipher $\mathcal{GE}_{cc} = (\overline{\mathcal{E}}, \mathsf{KeySp}, \mathsf{TwSp}, I, \mathsf{Dom}_{cc})$. (Notice that we use the ability to have a domain with multiple sets in a crucial way to capture the length-preservation requirement. This cannot be done in the previous frameworks [6, 25].)

We build $\mathcal{GE}_{cc}$ using encode-then-encipher with the enciphering via general cipher $\mathcal{GE}$. Let $s(\ell) = 10^\ell$ for $\ell \in I$. Thus our indexes are just the number of digits in the credit card number. Now we need to specify the encoding function $\mathsf{Enc}$ and decoding function $\mathsf{Dec}$. But these are easy as $\mathsf{Enc}(\ell, X)$, where $X$ is a sequence of $\ell$ decimal digits, simply views $X$ as the decimal representation of an integer $N$ and returns $N$. Conversely for $\mathsf{Dec}$. Our final enciphering algorithm is $\overline{\mathcal{E}}_K^{T,\ell}(X) = \mathsf{Dec}(\ell, \mathcal{E}_K^{T,\ell}(\mathsf{Enc}(\ell, X))$. In Section B we provide a fully instantiated version of this construction.

Example 2: multi-alphabet strings. We construct a general cipher for multi-alphabet strings. In a moment we will show how this cipher can be used to handle multi-field records. A multi-alphabet string is a sequence of digits $p_k \cdots p_1$ where $p_i \in \Sigma_i$ for $1 \le i \le k$ and alphabets $\Sigma_k, \ldots, \Sigma_1$. The index set for our cipher is any sequence of (uniquely encoded descriptions of) alphabets, e.g. $\Sigma_k \Sigma_{k-1} \cdots \Sigma_1$ for any alphabets $\Sigma_k$ for any $k > 1$. (For simplicity here we do not handle single-character strings.) To any alphabet $\Sigma$ we associate an injective map $\mathsf{num}_\Sigma \colon \Sigma \to [0 \mathinner{..} |\Sigma| - 1]$. Its inverse we denote via $\mathsf{char}_\Sigma$.

We now build a general cipher $\mathcal{GE}_{mas} = (\tilde{\mathcal{E}}, \mathsf{KeySp}, \mathsf{TwSp}, I, \mathsf{Dom}_{mas})$ that inherits the key space and tweak space of the underlying general cipher $\mathcal{GE}$, has index set $I$ specifying the space of arbitrary string formats (the sequence of alphabets digits are taken from), and $\mathsf{Dom}_{mas}$ that maps an index to a set $\mathbb{Z}_{s(\ell)}$. We use the encode-then-encipher paradigm with encoding and decoding functions as given in Figure 1. Given an index $\ell$ encoding alphabets $\Sigma_k, \ldots, \Sigma_1$, the routines map a $k$-digit multi-alphabet string (from said alphabets) to a numerical value between 0 and $s(\ell) - 1$ where $s(\ell) = \prod_{i=1}^{k} |\Sigma_i|$. The domain map $\mathsf{Dom}_{mas}(\ell)$ outputs $\mathbb{Z}_{s(\ell)}$ Notice that many values of $\ell$ will lead to the same $\mathbb{Z}_n$, but this will not be a problem either for security or functionality. Our final enciphering algorithm is again $\tilde{\mathcal{E}}_K^{T,\ell}(X) = \mathsf{Dec}(\ell, \mathcal{E}_K^{T,\ell}(\mathsf{Enc}(\ell, X))$.

Example 3: multi-field records. Now we return to the complex example from the introduc-

```
$\mathsf{Enc}(\ell, P)$
Parse $\ell$ as $\Sigma_k, \ldots, \Sigma_1$
Parse $P$ as $p_k, \ldots, p_1$ where $p_i \in \Sigma_i$
$x \leftarrow \mathsf{num}_{\Sigma_1}(p_1) + \sum_{i=2}^{k} \left( \mathsf{num}_{\Sigma_i}(p_1) \cdot \prod_{j<i} |\Sigma_j| \right)$
**Return** $x$
```

```
$\mathsf{Dec}(\ell, y)$
Parse $\ell$ as $\Sigma_k, \ldots, \Sigma_1$
**For** $i = k$ to $2$ **do**
    $d \leftarrow \prod_{j<i} |\Sigma_j|$ ; $c_i \leftarrow \lfloor y / d \rfloor$ ; $y \leftarrow y - c_i \cdot d$
**Return** $\mathsf{char}_{\Sigma_k}(c_k) \| \cdots \| \mathsf{char}_{\Sigma_2}(c_2) \| \mathsf{char}_{\Sigma_1}(y)$
```

Figure 1: Encoding and decoding algorithms for a multi-alphabet general cipher.

tion. We want a format-preserving encryption that maps input plaintext $\mathsf{Name}, \mathsf{Addr}, \mathsf{CC}$ to output ciphertext $\mathsf{Name}^*, \mathsf{Addr}^*, \mathsf{CC}^*$. $\mathsf{Name}$ and $\mathsf{Name}^*$ must consist of two strings of letters, each being an upper case letter followed by lower case letters. $\mathsf{Addr}$ and $\mathsf{Addr}^*$ must be alphanumeric characters, spaces, or commas followed by a valid postal code. $\mathsf{CC}$ and $\mathsf{CC}^*$ must be 8–19 decimal digits followed by a valid Luhn digit [10].

We can handle these formats by interpreting a record as a multi-alphabet string and using $\mathcal{GE}_{mas}$. All that is needed is how to derive an index for a given record. Let $\Sigma_{lc} = \{a, \ldots, z\}$, $\Sigma_{uc} = \{A, \ldots, Z\}$, $\Sigma_{let} = \Sigma_{lc} \cup \Sigma_{uc}$, $\Sigma_{num} = \{0, \ldots, 9\}$, and $\Sigma_{\alpha} = \Sigma_{let} \cup \Sigma_{num} \cup \{,\} \cup \{\sqcup\}$. Let $\Sigma_{zip}$ be the set of all valid (5-digit US) postal (ZIP) codes. Let $\Sigma_{cc}^{i}$ be the set of all $i$-digit credit card numbers with valid check digit. That is $\Sigma_{cc}^{i} = \{n \| \mathsf{L}(n) \mid n \in \Sigma_{num}^{i-1}\}$. We write $\langle \Sigma \rangle$ to mean an (unambiguous) encoding of $\Sigma$ into a string. We assume an encoding for which the concatenation of several such encodings can also be unambiguously decoded.

Our input is the triple of strings $(\mathsf{Name}, \mathsf{Addr}, \mathsf{CC})$. Produce an index string $\ell$ as follows. Initially set $\ell = \varepsilon$. Scan $\mathsf{Name}$ from left to right. For the first character, append $\langle \Sigma_{uc} \rangle$ to $\ell$. For each further non-space character in the first string, append $\langle \Sigma_{let} \rangle$. Append $\langle \{\sqcup\} \rangle$. Then append $\langle \Sigma_{uc} \rangle$ and append $\langle \Sigma_{let} \rangle$ for each subsequent character. Then scan $\mathsf{Addr}$, appending $\langle \Sigma_{\alpha} \rangle$ for each character stopping before the final 5 digit postal code. Append here $\langle \Sigma_{zip} \rangle$. Finally, append $\langle \Sigma_{cc}^{c} \rangle$ where $c$ is the number of decimal digits in $\mathsf{CC}$.

The result is a string encoding the appropriate alphabets of the characters used within the input record. (We are treating the ZIP code and credit card number as individual characters from particular alphabets.) We then encipher (for any desired tweak $T$) via $\tilde{\mathcal{E}}_{K}^{T,\ell}(\ell, (\mathsf{Name}, \mathsf{Addr}, \mathsf{CC}))$. The ciphertext will have the desired format. Note that ciphertexts contain all the information required to reproduce the appropriate $\ell$ for decryption.

## 4 Format-Preserving Feistel (FPF)

We present a construction of a general cipher $\mathcal{GE} = (\mathcal{E}, \mathsf{KeySp}, \mathsf{TwSp}, I, \mathsf{Dom})$ for key space $\mathsf{KeySp}$, tweak space $\mathsf{TwSp} \subseteq \{0,1\}^*$, and numeric domain with size function $s$ over index set $I$. (That is, $\mathsf{Dom}(\ell) = \mathbb{Z}_{s(\ell)}$ for all $\ell \in I$.) The construction is parameterized by the following.

- A function $\mathsf{split}$ that given $\ell \in I$ returns a pair $a, b \in \mathbb{N}$ such that $ab = s(\ell)$.
- A function $r\colon I \to \mathbb{N}$ specifying the number of rounds.
- A round function, which is a PRF $F\colon \mathsf{KeySp} \times D \to \mathbb{N}$ where $D = \mathsf{TwSp} \times \mathbb{N}^4$.

Figure 2 shows the enciphering algorithm $\mathcal{E}$ as well as describing the associated deciphering algorithm $\mathcal{D}$.

Discussion. Following [6], the internal Feistel computations are done modulo integers $a, b$ rather than over strings. To obtain a general cipher, not only the tweak $T$ but also the index $\ell$ of the domain set are input to the round function. This effectively “separates” instances of the cipher for different tweaks and indexes. To make the round functions independent while using a single key

```
algorithm $\mathcal{E}_K^{T,\ell}(X)$:
$a, b \leftarrow \mathsf{split}(\ell)$
$L_0 \leftarrow X \bmod a$ ; $R_0 \leftarrow \lfloor X/a \rfloor$
For $i = 1, \ldots, r(\ell)$ do
    $L_i \leftarrow R_{i-1}$
    $Z_i \leftarrow F_K(T, \ell, i, R_{i-1})$
    If $i \bmod 2 = 1$ then $R_i \leftarrow (Z_i + L_{i-1}) \bmod a$
    else $R_i \leftarrow (Z_i + L_{i-1}) \bmod b$
If $r(\ell) \bmod 2 = 1$ then Return $Y \leftarrow aL_{r(\ell)} + R_{r(\ell)}$
else Return $Y \leftarrow aR_{r(\ell)} + L_{r(\ell)}$
```

```
algorithm $\mathcal{D}_K^{T,\ell}(Y)$:
$a, b \leftarrow \mathsf{split}(\ell)$
If $r(\ell) \bmod 2 = 1$ then $R_{r(\ell)} \leftarrow Y \bmod a$ ; $L_{r(\ell)} \leftarrow \lfloor Y/a \rfloor$
Else $L_{r(\ell)} \leftarrow Y \bmod a$ ; $R_{r(\ell)} \leftarrow \lfloor Y/a \rfloor$
For $i = r(\ell), \ldots, 1$ do
    $R_{i-1} \leftarrow L_i$
    $Z_i \leftarrow F_K(T, \ell, i, L_i)$
    If $i \bmod 2 = 1$ then $L_{i-1} \leftarrow (R_i - Z_i) \bmod a$
    else $L_{i-1} \leftarrow (R_i - Z_i) \bmod b$
Return $aR_1 + L_1$
```

Figure 2: Encryption and decryption algorithms for the FPF general cipher where $K \in \mathsf{KeySp}$, $T \in \mathsf{TwSp}$, $\ell \in I$, and $X, Y \in \mathsf{Dom}(\ell)$.

the round number is also input to the round function. Cycle walking is eliminated by requiring that $ab = n$, where $n = s(\ell)$. (Black and Rogaway [6] need cycle walking because they only require $ab \geq n$.) The price we pay is that the Feistel can get unbalanced. As we discuss later in more depth, this can significantly impact security. But the fact is that in applications of interest, including encryption of (segments of) credit card numbers, we can always pick $a \approx b$ so the security is not impacted. Even if this is not possible, we can compensate with more rounds. With 7 or more rounds and relative balance it appears that security is achieved (there are no known and efficient attacks); we suggest 12 as a more conservative number.

How do we specify split? This is straightforward for our envisioned applications. Consider example 1 from the previous section. Here indices were the number of decimal digits of the input credit card. We have $\mathsf{split}(\ell)$ return $a = 10^{\ell_0}$ and $b = 10^{\ell_1}$ where $\ell_0 = \lfloor \ell/2 \rfloor$ and $\ell_1 = \lceil \ell/2 \rceil$. Notice that $|\ell_0 - \ell_1| \leq 1$ so that the Feistel will be almost balanced. For the second example from the previous section, an index $\ell$ is a description of alphabets $\Sigma_k, \ldots, \Sigma_1$ to which characters of the input belong. Recall that here $s(\ell) = \prod_{i=1}^{k} |\Sigma_i|$. Let $\mathsf{split}(\ell)$ first compute $s(\ell)$, then find the first value $h \in [1 \,..\, k]$ such that $s(\ell)/2 \leq \prod_{i=1}^{h} |\Sigma_i|$, and finally output $a = \prod_{i=1}^{h} |\Sigma_i|$ and $b = \prod_{i=h+1}^{k} |\Sigma_i|$. For many alphabet types, the resulting Feistel will be relatively balanced.

Round functions. FPF relies on a round function $F$ with non-standard domain ($\mathsf{TwSp} \times \mathbb{N}^4$) and range ($\mathbb{N}$). This function needs to be a secure PRF. Building suitable $F$ is straightforward given PRFs working over bit strings. We can therefore use well-known constructions of PRFs based on conventional block ciphers (e.g. 3DES or AES) or based on cryptographic hash functions (e.g. SHA-256). See Appendix A for 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).

The effect of imbalance. As promised above we look into this more closely. One might note that there seems here to be a degenerate case where the construction will fail to be secure, namely when $n$ is prime, so that either $a$ or $b$ is 1. But we can assume without loss of generality that $n$ is composite as follows: build the cipher for $n - 1$, which is composite, and use a technique from [6] to extend by one point. (The so-called one-off construction.)

The best attack we know against this kind of unbalanced Feistel network is presented in Section 5.2. Looking ahead, in extreme cases (such as when $n = 2p$ for a large prime $p$) our (new) attack succeeds at recovering messages with probability about $2^{-r(\ell)/2}$ using only a couple of queries. One can defend against it by increasing the number of rounds for at risk values of $n$, which is easily facilitated since the cipher can tailor the number of rounds to $n$. But as indicated above our experience is that these anomalous cases seldom arise in applications.

## 5 Security: Definitions and Analysis

In this section we formalize several security notions for general ciphers: the traditional notion of security as a pseudorandom permutation (PRP), suitably adapted to our setting, message privacy security (MP) (adapted from definitions from [4]), and security against message recovery (MR) attacks. The latter two notions quantify the ability of an adversary to learn partial information about (or even recover fully) plaintexts as a function of the adversary's a priori uncertainty about them. We show that PRP security implies MP security and that MP security implies MR security.

Why do we consider anything beyond PRP? We do so because the MP and MR notions seem sufficient for secure use of format-preserving encryption. These notions both model the following attack scenario. An intruder gains access to systems hosting data sets that were encrypted with the help of some (possibly remote) hardware security module (HSM). The intruder, in control of the entire system excepting the HSM, will have whatever proper credentials are required to submit encryption (but not necessarily decryption) requests to the HSM. MP measures the ability of this intruder to learn partial information about plaintexts. MR measures the ability of the intruder to recover plaintexts.

While we believe FPF is strong in the sense of PRP, the fact is that no one has yet been able to prove security of Feistel-based ciphers with bounds meaningful for the application of format-preserving encryption. (See the detailed discussion in Section 5.3.) We provide significantly more assurance about the security of FPF by proving MP and MR security with better bounds than any PRP analysis seems to have offered.

On leaking format. By design format-preserving encryption leaks to adversaries the format of underlying plaintexts. One must be careful to ensure that this information being leaked is not damaging in and of itself. In our application examples, the format is public (e.g. an adversary knows the database's structure and formatting requirements). Even here though some formats being preserved could be damaging if the set of possible messages of that format is small. For example, what if there is only one person with a two-letter name in the database? Since length is leaked, an adversary would know what entry in the database corresponds to this person. Note that our definitions do take account of this risk, since the adversary will be able to attack index sets of its choice. Security will only hold if challenge messages from this set are unpredictable.

Games. Our security definitions and proofs use code-based games [2], and so we recall some background from [2]. A game has an **Initialize** procedure, procedures to respond to adversary oracle queries, and (sometimes) a **Finalize** procedure. A game G is executed with an adversary $\mathcal{A}$ as follows. First, **Initialize** executes, and its outputs are the inputs to $\mathcal{A}$. Then $\mathcal{A}$ executes, its oracle queries being answered by the corresponding procedures of G. If there is no **Finalize** procedure, then when $\mathcal{A}$ terminates, its output is also called the output of the game. If there is a **Finalize** procedure, then when $\mathcal{A}$ terminates, its output becomes the input to the **Finalize** procedure. In this case the output of **Finalize** is called the output of the game. In either case, we let let $\mathrm{G}^{\mathcal{A}} \Rightarrow y$ denote the event that this game output takes value $y$. The boolean flag $\mathsf{bad}$ is assumed initialized to false. Games G, H are identical-until-$\mathsf{bad}$ if their code differs only in statements that follow the setting of $\mathsf{bad}$ to true. We say that "$\mathrm{G}^{\mathcal{A}}$ sets $\mathsf{bad}$" to denote the event that game G, when executed with adversary $\mathcal{A}$, sets bad to true. It is shown in [2] that if $\mathrm{G}, \mathrm{H}$ are identical-until-$\mathsf{bad}$ and $\mathcal{A}$ is an adversary, then $\Pr\left[\,\mathrm{G}^{\mathcal{A}} \text{ sets } \mathsf{bad}\,\right] = \Pr\left[\,\mathrm{H}^{\mathcal{A}} \text{ sets } \mathsf{bad}\,\right]$. The fundamental lemma of game-playing [2] says that if $\mathrm{G}, \mathrm{H}$ are identical-until-$\mathsf{bad}$ then $\Pr\left[\,\mathrm{G}^{\mathcal{A}} \Rightarrow y\,\right] - \Pr\left[\,\mathrm{H}^{\mathcal{A}} \Rightarrow y\,\right] \leq \Pr\left[\,\mathrm{G}^{\mathcal{A}} \text{ sets } \mathsf{bad}\,\right]$.

## 5.1 Security Notions for General Ciphers

Let $\mathcal{GE} = (\mathcal{E}, \mathsf{KeySp}, \mathsf{TwSp}, I, \mathsf{Dom})$ be a general cipher.

PRP SECURITY. A PRP adversary $A$ has access to an oracle that accepts inputs $T \in \mathsf{TwSp}$, index $\ell \in I$, and message $X \in \mathsf{Dom}(\ell)$. It returns a value $Y \in \mathsf{Dom}(\ell)$. In one world $Y$ is computed by applying $\mathcal{E}_K(T, \ell, X) = \mathcal{E}_K^{T,\ell}(X)$ for a random $K$. In another world $Y$ is computed by applying $\pi(T, \ell, X)$, where the map $\pi(T, \ell, \cdot)$ is a random permutation on $\mathsf{Dom}(\ell)$ for all $T \in \mathsf{TwSp}$ and $\ell \in I$. Formally we define the advantage of $A$ against $\mathcal{E}$ by

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{prp}}(A) = \Pr\left[\,\mathrm{Real}_{\mathcal{E}}^{\mathcal{A}} \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{Rand}_{\mathcal{E}}^{\mathcal{A}} \Rightarrow 1\,\right] .$$

where the games $\mathrm{Real}_{\mathcal{E}}$ and $\mathrm{Rand}_{\mathcal{E}}$ are defined in Figure 3. The definition explicitly models the requirement that different length inputs should appear to be enciphered by independent random permutations.

MP SECURITY. A *MP-adversary* $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ is a pair of algorithms. The message sampler $\mathcal{A}_m$ takes no input and outputs a tweak $T^* \in \mathsf{TwSp}$, index $\ell^* \in I$, and a pair of challenge messages $X_0^*, X_1^* \in (\mathsf{Dom}(\ell))^2$. The guessing adversary $\mathcal{A}_g$ takes input a challenge ciphertext and has access to an encryption oracle. We define security of a scheme $\mathcal{GE}$ against MP-adversaries via the game $\mathrm{MP}_{\mathcal{GE}}^{\mathcal{A}}$ shown in Figure 3. We define the MP-advantage of an MP-adversary $\mathcal{A}$ against $\mathcal{E}$ by

$$\mathbf{Adv}_{\mathcal{GE}}^{\mathrm{mp}}(\mathcal{A}) = 2 \cdot \Pr\left[\,\mathrm{MP}_{\mathcal{GE}}^{\mathcal{A}} \Rightarrow \textsc{true}\,\right] .$$

An MP adversary $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ has *min-entropy* $\mu$ if

$$\Pr\left[\,X_b = X' \,:\, (T, \ell, X_0, X_1) \xleftarrow{\$} \mathcal{A}_m\,\right] \leq 2^{-\mu}$$

holds for all $b \in \{0,1\}$ and $X' \in \mathsf{Dom}(\ell)$. An MP adversary $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ is *legitimate* if: $(X_0, X_1) \in (\mathsf{Dom}(\ell))^2$ for any $\ell, X_0, X_1$ output by $\mathcal{A}_m$, all queries $T, \ell, X$ by $\mathcal{A}_g$ to **Enc** are such that $T \in \mathsf{TwSp}$, $\ell \in I$, and $X \in \mathsf{Dom}(\ell)$ and $\mathcal{A}_g$ never repeats a query to **Enc**.

Note that this definition (and the next) only considers block sources (see [4, 9] for a discussion). We can also give a multiple challenge message notion of MP for more general (non-block) sources, following [4]. (We can also adapt the next definition analogously.) The proofs for FPF security in the MP (and MR) sense carry over to this more general setting.

MR SECURITY. Let $\mathcal{E}$ be a general cipher with key space $\mathsf{KeySp}$, tweak space $\mathsf{TwSp}$, and domain $\mathsf{Dom}$ with associated index set $I$. An *MR-adversary* $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ is a pair of algorithms. The message sampler $\mathcal{I}_m$ takes no input and outputs a tweak $T^* \in \mathsf{TwSp}$, index $\ell^* \in I$, and a message $X^* \in \mathsf{Dom}(\ell)$. The message recoverer $\mathcal{I}_g$ takes input a challenge ciphertext and has access to an encryption oracle. We define security of general cipher $\mathcal{GE}$ against MR-adversaries via the game $\mathrm{MR}_{\mathcal{GE}}^{\mathcal{I}}$ shown in Figure 3. We define the MR-advantage of an MR-adversary $\mathcal{I}$ against $\mathcal{E}$ by

$$\mathbf{Adv}_{\mathcal{GE}}^{\mathrm{mr}}(\mathcal{A}) = \Pr\left[\,\mathrm{MR}_{\mathcal{GE}}^{\mathcal{I}} \text{ sets } \textsc{win}\,\right] .$$

An MR adversary $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ has *min-entropy* $\mu$ if

$$\Pr\left[\,X = X' \,:\, (T, \ell, X) \xleftarrow{\$} \mathcal{I}_m\,\right] \leq 2^{-\mu}$$

holds for all $X' \in \mathsf{Dom}(\ell)$. An MR adversary $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ is *legitimate* if: $X \in \mathsf{Dom}(\ell)$ for any $\ell, X$ output by $\mathcal{A}_m$, all queries $T, \ell, X$ by $\mathcal{I}_g$ to **Enc** are such that $T \in \mathsf{TwSp}$, $\ell \in I$, and $X \in \mathsf{Dom}(\ell)$ and $\mathcal{I}_g$ never repeats a query to **Enc**.

IMPLICATIONS. A random permutation is optimally MP secure. Here optimal means that an adversary with min-entropy $\mu$ requires on the order of $q = 2^{\mu}$ queries to the **Enc** oracle to infer partial information about the challenge message (i.e. a brute force search is the only strategy). Thus, any general cipher that is secure as a PRP also hides all partial information about (high

```
procedure Initialize                              Game Real$_{\mathcal{GE}}$
  $K \xleftarrow{\$} \mathsf{KeySp}$

procedure Enc($T, \ell, X$)
  Return $\mathcal{E}_K^{T,\ell}(X)$
```

```
procedure Initialize                              Game Rand$_{\mathcal{GE}}$
  $\pi, \mathtt{R}$ everywhere $\perp$

procedure Enc($T, \ell, X$)
  If $\pi[T,\ell,X] \neq \perp$ Return $\pi[T,\ell,X]$
  $\pi[T,\ell,X] \xleftarrow{\$} \mathsf{Dom}(\ell)/\mathtt{R}[T,\ell]$
  $\mathtt{R}[T,\ell] \xleftarrow{\cup} \pi[T,\ell,X]$
  Return $\pi[T,\ell,X]$
```

```
procedure Initialize                              Game MP$^{\mathcal{A}}_{\mathcal{GE}}$
  $K \xleftarrow{\$} \mathsf{KeySp}$ ; $b \xleftarrow{\$} \{0,1\}$
  $(T^*, \ell^*, X_0^*, X_1^*) \xleftarrow{\$} \mathcal{A}_m$
  $C^* \leftarrow \mathcal{E}_K^{T^*,\ell^*}(X_b^*)$
  Return $(T^*, \ell^*, C^*)$

procedure Enc($T, \ell, X$)
  Return $\mathcal{E}_K^{T,\ell}(X)$

procedure Finalize($b'$)
  Return $(b' = b)$
```

```
procedure Initialize                              Game MR$^{\mathcal{I}}_{\mathcal{GE}}$
  $K \xleftarrow{\$} \mathsf{KeySp}$ ; $(T^*, \ell^*, X^*) \xleftarrow{\$} \mathcal{I}_m$
  Return $(T^*, \ell^*, \mathcal{E}_K^{T^*,\ell^*}(X^*))$

procedure Enc($T, \ell, X$)
  If $(T, \ell, X) = (T^*, \ell^*, X^*)$ then WIN $\leftarrow$ TRUE
  Return $\mathcal{E}_K^{T,\ell}(X)$
```

Figure 3: **(Top)** The games Real and Rand for defining PRP security of a general cipher $\mathcal{GE}$. **(Middle)** The MP security game for MP-adversary $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ and general cipher $\mathcal{GE}$. defining message privacy for a general cipher. **(Bottom)** The MR security game for MR-adversary $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ and general cipher $\mathcal{GE}$.

min-entropy) plaintexts. Any general cipher secure in the MP sense is just as secure against message recovery. The following proposition, whose proof is straightforward and omitted, captures these relationships formally.

**Proposition 5.1 [PRP security ⇒ MP security ⇒ MR security]** *Let $\mathcal{GE}$ be a general cipher. Let $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ be an MP adversary with min-entropy $\mu$ that makes at most $q$ queries. Then there exists a PRP-adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathrm{mp}}_{\mathcal{GE}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{\mathcal{GE}}(\mathcal{B}) + \frac{q}{2^\mu} .$$

*$\mathcal{B}$ runs in time that of $\mathcal{A}$ and makes at most $q$ queries. Let $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ be an MR adversary with min-entropy $\mu$ that makes at most $q$ queries. Then there exists an MP adversary $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ such that*

$$\mathbf{Adv}^{\mathrm{mr}}_{\mathcal{GE}}(\mathcal{I}) \leq \mathbf{Adv}^{\mathrm{mp}}_{\mathcal{GE}}(\mathcal{A}) .$$

*$\mathcal{A}$ runs in time that of $\mathcal{I}$, has min-entropy $\mu$, and makes at most $q$ queries.* □

## 5.2 Best Known Attacks against FPF

We first analyze the security of FPF in terms of previously proposed attacks and novel attacks.

Patarin's attack. Here we discuss Patarin's attack [17] on Feistel networks, which is the only previous attack we know of that applies to Feistel networks with rounds greater than 6. (We recommend an absolute minimum number of rounds for FPF be 7.) This attack attempts to distinguish between a Feistel network and a random permutation by making many queries. The attacker then computes all possible round functions, and checks if the queries and their responses are consistent with any one of the functions, outputting 1 if so. This attack requires an intractable amount of computation, even for small sets. Consider attacking FPF using $r$ rounds on domain $\mathbb{Z}_n$ for $n = ab$. Then the number of possible instances of FPF is $C = (a^b)^{\lfloor r/2 \rfloor} \cdot (b^a)^{r - \lfloor r/2 \rfloor}$. For simplicity (it won't affect the implications significantly) let's assume $a = b$, in which case $C = a^{ra}$. The probability that a random function from $\mathbb{Z}_n$ to $\mathbb{Z}_n$ (we ignore permutivity constraints for simplicity) matches at least one of these functions for any set of $q$ distinct domain points is at most $C/n^q$. To achieve advantage one, then, the adversary needs to choose $q$ so that $C/n^q = 1$. Rearranging we have that $a^{ra} = n^q = a^{2q}$ and this implies that $q = ra/2$. Say $a = b = 10$, which corresponds to using FPF to encrypt 2-digit numbers, and $r = 7$. Then $q = 70/2 = 35$, which is pretty small. Fortunately, the running time in this case is about $qC = 35 \cdot 10^{70} \approx 2^{237}$, making the attack practically intractable.

A new distinguishing attack. Highly unbalanced Feistel networks are susceptible to highly efficient attacks that succeed with exponentially vanishing probability as the number of rounds increases. Still, for small, fixed round number, the attacks could be dangerous. We present a PRP adversary $A$ against FPF for some $\ell$ such that $\mathsf{split}(\ell)$ outputs $a, b$. Assume without loss of generality that $a \le b$. Let $T \in \mathsf{TwSp}$. Denote $r(\ell)$ by $r$ and assume $r$ is even (the attack easily extends to the case that $r$ is odd). Then adversary $A$ works as described below.

```
adversary $A^{\mathcal{O}(\cdot,\cdot,\cdot)}$:
$L_0 \xleftarrow{\$} \mathbb{Z}_a$ ; $L'_0 \leftarrow L_0$ ; $R_0 \xleftarrow{\$} \mathbb{Z}_b$ ; $R'_0 \xleftarrow{\$} \mathbb{Z}_b \backslash \{R_0\}$
$Y \leftarrow \mathcal{O}(T, \ell, (L_0 + a \cdot R_0))$ ; $Y' \leftarrow \mathcal{O}(T, \ell, (L'_0 + a \cdot R'_0))$
$D \leftarrow Y \bmod b$ ; $D' \leftarrow Y' \bmod b$
If $D - R_0 \equiv D' - R'_0 \pmod{b}$ then Ret 1 Else Ret 0
```

First we analyze $\Pr[A^{\mathrm{Rand}} \Rightarrow 1]$. Adversary $A$ outputs 1 exactly when $D - R_0 + R'_0 \equiv D' \pmod{b}$. Thus $\Pr[A^{\mathrm{Rand}} \Rightarrow 1] = \frac{1}{a} \cdot \frac{1}{b-1} + \frac{a-1}{a} \cdot \frac{1}{b}$. Now we analyze $\Pr[A^{\mathrm{Real}} \Rightarrow 1]$. Let $d = r/2$. This is the number of times a value $R_i$ is assigned in $\mathcal{E}$ for $i > 0$ and even. (Refer to Figure 2.) Let $Z_1, \ldots, Z_r$ be the outputs of the round function $F$ for rounds 1 to $r$ (respectively) when evaluating $\mathcal{E}_K^{T,\ell}(L_0, R_0)$ in response to $A$'s first query. Similarly let $Z'_1, \ldots, Z'_r$ be the outputs of the round function $F$ for rounds 1 to $r$ (respectively) when evaluating $\mathcal{E}_K^{T,\ell}(L'_0, R'_0)$ in response to $A$'s second query. Consider the situation in which $Z_i = Z'_i$ for all $i > 0$ and $i$ even. This occurs with probability at least $a^{-d}$. (This is true because the inputs to each of the relevant $d$ round function applications will collide with probability $1/a$.) Then in this case it holds with probability one that $D - R_0 \equiv D' - R'_0 \pmod{b}$ since

$$D \equiv R_r \equiv R_0 + \sum_{i \le r\,,\, i \text{ even}} Z_i \pmod{b} \quad \text{and} \quad D' \equiv R'_r \equiv R'_0 + \sum_{i \le r\,,\, i \text{ even}} Z'_i \pmod{b}\,.$$

Therefore we have $\Pr[A^{\mathrm{Real}} \Rightarrow 1] \ge a^{-d}$. Combining this with the upper bound on $\Pr[A^{\mathrm{Rand}} \Rightarrow 1]$ given above we get

$$\mathbf{Adv}^{\mathrm{prp}}_{\mathrm{FPF}}(A) \ge \frac{1}{a^d} - \frac{1}{a(b-1)} - \frac{a-1}{ab}\ .$$

For certain values of $a, b, r$ this is large. Say $r = 7$ and $N = 2p$ for some relatively large prime $p$. Then $a = 2$, $b = p$ and $A$'s advantage is $1/3 - 1/(2p-2) - 1/2p$.

Message recovery attack. We can adapt the above attack to mount message recovery attacks. The distinguishing attack establishes a relationship $D - R_0 \equiv D' - R'_0 \pmod{b}$ for distinct messages with high probability. If $R_0$ is unknown, one can recover it if $D$, $D'$, and $R'_0$ are known. This requires a single known-plaintext and its associated ciphertext, which will have the desired collisions with the unknown plaintext with probability $a^{-d}$. From the known plaintext, ciphertext pair one can recover the unknown plaintext portion $R_0$. Then $L_0$ can be guessed (with probability of success $1/a$). The attack succeeds in recovering the full plaintext with probability at least $a^{-d-1}$.

### 5.3 Provable Security Bounds for FPF

PRP SECURITY. There is a long line of work investigating the PRP security of Feistel networks. The original bounds are due to Luby and Rackoff [13] and bounds for the Feistel network underlying FPF are given by Black and Rogaway [6]. These establish security up to $q \approx \sqrt{\min_\ell\{a,b\}}$ where the minimization is taken over the $a, b$ output by $\mathsf{split}(\ell)$ for all $\ell \in I$. A line of papers by Patarin [17, 19, 20] gives analysis of balanced Feistel networks with a small constant number of rounds, but that are asymptotic in the block size. It is not apparent whether these bounds apply when considering concrete security for fixed, small blocks, which is our concern here. Maurer and Pietrzak [15] prove that as the number of rounds of a balanced Feistel network increases, one can asymptotically approach the optimal information theoretic bound. Naor and Reingold investigate Feistel networks and (unbalanced) variants of them [16], showing that some forms of imbalance (though not the one considered here) can, in fact, improve security bounds for sufficiently many rounds.

MP AND MR SECURITY. We investigate the MP and MR security of FPF, giving concrete security bounds up to $q \approx \min_\ell\{a,b\}$ for (unbalanced) Feistel networks with fixed message size and fixed number of rounds. Of course we could use Theorem 5.1 in conjunction with an adaptation (to the general cipher setting) of [6, Thm. 2], but as mentioned above this would only give security up to $q \approx \sqrt{\min_\ell\{a,b\}}$. Instead we provide a direct proof of the MR security of 5-round FPF, which gives better bounds. This is possible because, in particular, birthday-phenomena abusing attacks won't usually reveal information about a *specific* message. We state and prove the result in terms of MR security. This is for simplicity only and the result and proof lift straightforwardly to the MP setting.

**Theorem 5.2** *Let $F$: $\mathsf{KeySp} \times (\mathsf{TwSp} \times \mathbb{N}^4) \to \mathbb{N}$. Let $\mathcal{E}$ be the* FPF *cipher for index set $I$, domain $\mathsf{Dom}$, $r(\cdot) = 5$, round function $F$, and $\mathsf{split}$. Let $a_{min}, b_{min}$ be the smallest values output by $\mathsf{split}$ for any $\ell \in I$. Let $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ be an MR-adversary with min-entropy $\mu$. Then there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathrm{mr}}_{\mathcal{E}}(\mathcal{I}) \le \mathbf{Adv}^{\mathrm{prf}}_{F}(\mathcal{B}) + \frac{q}{2^\mu} + \frac{q}{a_{min}} + \frac{q}{b_{min}}\,.$$

*Adversary $\mathcal{B}$ runs in the time of $\mathcal{I}$ and makes at most $q$ queries.* □

A proof is given in Appendix C. Note that even though the bounds we prove are better than any known previously, they are still not sufficient for concluding security for small $a, b$. Indeed, finding even better proven security bounds represents an interesting open problem. As mentioned above, no known attacks perform better than brute force for sufficiently many rounds.

## References

[1] M. Bellare, K. Pietrzak, and P. Rogaway. Improved Security Analyses for CBC MACs. *Advances in Cryptology – CRYPTO '05*, LNCS vol. 3621, pp. 527–545, Springer, 2005.

[2] Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. *Advances in Cryptology – EUROCRYPT '06*, LNCS vol. 4004, pp. 409–426, Springer, 2006.

[3] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. *Advances in Cryptology – CRYPTO '07*, LNCS vol. 4622, pp. 535–552, Springer, 2007.

[4] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. *Advances in Cryptology – CRYPTO '08*, LNCS vol. 5157, pp. 360–378, Springer, 2008.

[5] J. Black and P. Rogaway. CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. *J. of Cryptology*, vol. 18, no. 2, pp. 111–131, 2005.

[6] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. *Topics in Cryptology – CT-RSA '02*, LNCS vol. 2271, Springer, pp. 114–130, 2002.

[7] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. *Advances in Cryptology – CRYPTO '99*, LNCS vol. 1666. pp. 216–233, Springer, 1999.

[8] J. Black and P. Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. *Advances in Cryptology – Eurocrypt '02*, LNCS vol. 2332, pp. 384–397, Springer, 2002.

[9] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. *Advances in Cryptology – CRYPTO '08*, LNCS vol. 5157, pp. 335–359 Springer, 2008.

[10] ISO/IEC 7812-1:2006. Identification cards – Identification of issuers – Part 1: Numbering system.

[11] T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. *Fast Software Encryption – FSE '03*, LNCS vol. 2887, pp. 129–153, Springer, 2003.

[12] M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. *Advances in Cryptology – CRYPTO 2002*, LNCS vol. 2442, Springer, pp. 31–46, 2002.

[13] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal of Computing*, vol. 17, no. 2, pp. 373–386, 1988.

[14] S. Lucks. Faster Luby-Rackoff ciphers. *Fast Software Encryption 1996*, LNCS vol. 1039, Springer, pp. 189–203, 1996.

[15] U. Maurer and K. Pietrzak. The Security of Many-Round Luby-Rackoff Pseudo-Random Permutations. *Advances in Cryptology – EUROCRYPT '03*, LNCS vol. 2656, pp. 544-561, Springer, 2003.

[16] M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, vol. 12, no. 1, pp. 29–66, 1999.

[17] J. Patarin. New Results on Pseudorandom Permutation Generators Based on the DES Scheme. *Advances in Cryptology – CRYPTO '91*, LNCS vol. 576, Springer, pp. 301–312, 1991.

[18] J. Patarin. Generic Attacks on Feistel Schemes. *Advances in Cryptology – ASIACRYPT '01*, LNCS vol. 2248, Springer, pp. 222–238, 2001.

[19] J. Patarin. Luby-Rackoff: 7 Rounds Are Enough for $2n(1-\epsilon)$ Security. *Advances in Cryptology – CRYPTO '03*, LNCS vol. 2729, Springer, pp. 513–529, 2003.

[20] J. Patarin. Security of Random Feistel Schemes with 5 or More Rounds. *Advances in Cryptology – CRYPTO '04*, LNCS vol. 3152, Springer, pp. 106–122, 2004.

[21] S. Patel, Z. Ramzan, and G. Sundaram. Efficient constructions of variable-input-length block ciphers. *Selected Areas in Cryptography 2004*, LNCS vol. 3357, pp. 326–340, 2004.

[22] PCI Security Standards Council. Payment Card Industry Data Security Standard Version 1.2. Available from `https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml`

[23] B. Schneier and J. Kelsey. Unbalanced Feistel networks and block cipher design. *Fast Software Encryption 1996*, LNCS vol. 1039, Springer, pp. 121–144, 1996.

[24] R. Schroeppel. Hasty pudding cipher specification. *First AES Candidate Workshop*, 1998.

[25] T. Spies. Feistel Finite Set Encryption Mode. `http://csrc.nist.gov/groups/ST/toolkit/`

BCM/documents/proposedmodes/ffsem/ffsem-spec.pdf

# A Building Efficient Round Functions

In this section we explore methods for instantiating efficient tweakable round functions based on a block cipher $E\colon \mathsf{KeySp}\times\{0,1\}^n \to \{0,1\}^n$. A simple and secure approach is to first build a variable-input-length PRF $F$ from $E$ using any of the many well known constructions (e.g. CMAC [5], PMAC [8], OMAC [11], UMAC [7], etc.). Then map tweak $T$ and message $M$ to a bit string via some injective encoding and apply $F$. Since each $T, M$ pair gives a unique input to $F$ we achieve the desired behavior, for any secure underlying PRF: independently chosen points for each pair $(T, M)$. We can achieve good efficiency by exploiting due the structure of the tweaks and messages used by FPF. We offer two constructions that do so.

In the following $\mathsf{StN}_l$ takes input an $l$-bit string $y$ and returns the integer $N$ $(0 \le N < 2^l)$ whose binary representation is $y$. For example, $\mathsf{StN}_6(000011) = 3$.

Prefix-free CBC-MAC. Recall that CBC-MAC is secure for variable-input-length inputs if a prefix-free encoding of messages is used. We will encode the tweaks in a manner that ensures prefix-freeness. Let $E\colon \mathsf{KeySp} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Fix some index set $I$. Let $\mathsf{pad}(T, \ell)$ output the string

$$\langle \ell \rangle_{\lceil \log |I| \rceil} \parallel \langle |T| \rangle_{\lceil \log |\mathsf{TwSp}| \rceil} \parallel T \parallel 0^p$$

where $p$ is the minimum number of bits needed to ensure that $\lceil \log |I| \rceil + \lceil \log |\mathsf{TwSp}| \rceil + p$ is a multiple of $n$. The round function $F\colon \mathsf{KeySp} \times \mathsf{TwSp} \times \mathbb{N}^4 \to \mathbb{N}$ is then defined by

$$F_K(T, \ell, i, X) = \mathsf{StN}_n(\mathsf{CBC\text{-}MAC}_K\,(\mathsf{pad}(T, \ell) \parallel \langle i \rangle_8 \parallel \langle X \rangle_{n-8}))$$

for any $K \in \mathsf{KeySp}$, $T \in \mathsf{TwSp}$, and $X \in \{0,1\}^{n-8}$.

The encoding ensures that one can do efficient precomputation, since $T, \ell$ do not change between rounds. One therefore precomputes $\tau \leftarrow \mathsf{CBC\text{-}MAC}_K(\mathsf{pad}(T, \mathsf{label}, a, b))$ first. During round $i$, applying the PRF to number $X$ can then be accomplished via a single call to $E$:

$$Z \leftarrow \mathsf{StN}_n(E_K(\tau \oplus (\langle i \rangle_8 \parallel \langle X \rangle_{n-8}))) \; .$$

That prefix-free CBC-MAC is a secure PRF was proven in [1]. They show that, for maximum message length $\ell$ and $q$ queries prefix-free CBC-MAC enjoys security bound approximately $\ell q^2/2^n$.

Rekeying by tweaks. We modify the above construction slightly to give an alternative construction that utilizes rekeying. Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher with $k \ge n$. Then define $G\colon \mathsf{KeySp} \times \mathsf{TwSp} \times \mathbb{N}^4 \to \mathbb{N}$ by

$$G_K(T, \ell, i, X) = E_{\mathsf{CBC\text{-}MAC}_K(\mathsf{pad}(T,\ell))|_k}(\langle i \rangle_8 \parallel \langle X \rangle_{n-8}) \; .$$

That is, $\mathsf{CBC\text{-}MAC}$ is applied to the padded tweak data to derive a key $\tau$, suitably truncated, for a final application of $E$ on the round number and message. Again precomputation is straightforward, meaning each round requires just a single block cipher application. Here, though, two keys are used with $E$.

Effect of modular arithmetic. Our PRFs are built from traditional ones that output bit strings, and are proven to be indistinguishable from random functions outputting bit strings. But our PRFs output numbers that are taken modulo $a$ or $b$ within FPF. Here we discuss how security is affected by this extra mod operation.

We begin by asking the following. Consider, on the one hand, the uniform distribution on $\mathbb{Z}_M$. Consider, on the other hand, the distribution on $\mathbb{Z}_M$ that is obtained by picking a random point $x$ in $\mathbb{Z}_N$ and returning $x \bmod M$. What is the statistical difference between these distributions?

To answer this, let IntDiv denote the integer division algorithm, which on inputs $N, M$ returns a quotient $q$ and remainder $r$ satisfying $N = Mq + r$ and $0 \leq r < M$. Then, we claim the following.

**Lemma A.1** *Let $N \geq M \geq 1$ be integers, and let $(q, r) \leftarrow \mathrm{IntDiv}(N, M)$. For $z \in \mathbb{Z}_M$ let*

$$P_{N,M}(z) \;=\; \Pr\left[ x \bmod M = z \;:\; x \xleftarrow{\$} \mathbb{Z}_N \right] .$$

*Then for any $z \in \mathbb{Z}_M$,*

$$P_{N,M}(z) \;=\; \begin{cases} \dfrac{q+1}{N} & \text{if } 0 \leq z < r \\ \dfrac{q}{N} & \text{if } r \leq z < M. \end{cases}$$

**Proof of Lemma A.1:** Let the random variable $X$ be uniformly distributed over $\mathbb{Z}_N$. Then

$$\begin{aligned} P_{N,M}(z) &= \Pr\left[ X \bmod M = z \right] \\ &= \Pr\left[ X < Mq \right] \cdot \Pr\left[ X \bmod M = z \mid X < Mq \right] \\ &\qquad + \Pr\left[ Mq \leq X < N \right] \cdot \Pr\left[ X \bmod M = z \mid Mq \leq X < N \right] \\ &= \frac{Mq}{N} \cdot \frac{1}{M} + \frac{N - Mq}{N} \cdot \begin{cases} \dfrac{1}{N - Mq} & \text{if } 0 \leq z < N - Mq \\ 0 & \text{if } N - Mq \leq z < M. \end{cases} \\ &= \frac{q}{N} + \frac{r}{N} \cdot \begin{cases} \dfrac{1}{r} & \text{if } 0 \leq z < r \\ 0 & \text{if } r \leq z < M. \end{cases} \end{aligned}$$

Simplifying yields the claimed equation. ▮

As a result, the statistical distance between the uniform distribution on $\mathbb{Z}_M$ and the distribution obtained by picking a random point $x$ in $\mathbb{Z}_N$ and returning $x \bmod M$ is

$$\frac{1}{2} \sum_{z=0}^{r-1} \left| \frac{q+1}{N} - \frac{1}{M} \right| + \frac{1}{2} \sum_{z=r}^{M-1} \left| \frac{q}{N} - \frac{1}{M} \right| \;=\; \frac{r(M-r)}{NM} \;\leq\; \frac{1}{4} \frac{M}{N} .$$

Typical maximum for the number of digits in the plaintext is 20. In this case, we have $M = 10^{10}$, so the above statistical distance is at most $10^{10}/2^{64} \approx 2^{-31}$. This is reasonably small, indicating that the mod operation does not dramatically affect the distribution. While one might be concerned that it is perhaps not small enough, the salient fact is that we do not know of any way in which this deviation affects security anyway.

# B Example Instantiation: VAES

Let $\Sigma = \{0, \ldots, 9\}$. In this section we describe a complete instantiation of FPF for domain $\mathsf{Dom}(\ell) = \mathbb{Z}_{10^\ell}$ for any $\ell \in I = \{\ell \mid 2 \leq \ell \leq 66\}$. We call this instance of FPF the VAES algorithm due to its use of AES as the underlying PRF. First, we need to define some notation and auxiliary functions.

Notation and auxiliary functions. $\mathsf{NtS}_l$ takes input an integer $N$ in the range $0 \leq N < 2^l$ and returns its encoding as a binary string of exactly $l$ bits. For example, $\mathsf{NtS}_6(3) = 000011$.

$\mathsf{StN}_l$ takes input an $l$-bit string $y$ and returns the integer $N$ $(0 \leq N < 2^l)$ whose binary representation is $y$. For example, $\mathsf{StN}_6(000011) = 3$.

$\mathsf{NtD}_l$ takes input an integer $N$ in the range $0 \leq N < 10^l$ and returns its representation as an $l$-digit number. For example, $\mathsf{NtD}_5(326) = 00326$. Thus, the operation consists merely of prepending enough zeros to bring the number of digits to exactly $l$.

$\mathsf{DtN}$ takes input an $l$-digit number $y$ and returns the corresponding integer $N$ $(0 \leq N < 10^l)$. For example, $\mathsf{DtN}(00326) = 326$. Thus, the operation consists merely of removing leading zeros.

$|X|_{10}$ denotes the number of digits in a digit-represented number. For example, $|00326|_{10} = 5$. Note that leading zeros are counted.

$N$ div $D$ returns the quotient obtained when integer $N$ is divided by integer $D$. For example $17 \operatorname{div} 2 = 8$ and $14 \operatorname{div} 2 = 7$.

$N$ mod $D$ returns the remainder obtained when integer $N$ is divided by integer $D$. This is an integer in the range $0, \ldots, D-1$. Note we can apply this operation even when $N$ is negative. For example $(30 - 70) \bmod 100 = 60$.

The algorithm. VAES is a concrete instantiation of the FPF algorithm. The general cipher defined by the enciphering function $\mathcal{E}$ and deciphering function $\mathcal{D}$ shown in Figure 4. The key space of VAES is that of AES, $\{0,1\}^{128}$. The tweak space of VAES is $\{0,1\}^{\leq 112}$. The numeric domain $\mathsf{Dom}$ is defined as $\mathsf{Dom}(\ell) = \mathbb{Z}_{10^\ell}$ for all $i \in I = \{2, \ldots, 66\}$. Hence the size function is $s(\ell) = 10^\ell$. The message space includes any decimal digit string of at least 2 digits and up to 66 digits, and encoding and decoding is done via $\mathsf{DtN}$ and $\mathsf{NtD}$. VAES utilizes the following split function, number of rounds, and round function PRF.

- (Split function) For any digit length $\ell \in \{2 \,..\, 66\}$, all inputs of that length can be represented by numbers from the set $\mathbb{Z}_{10^\ell}$. Since $10^\ell$ has natural and easily-computable factors, defining a split function is straightforward. Namely, the VAES split function $\mathsf{split}(\ell)$ returns, for any $\ell \in \{2 \,..\, 66\}$, the pair $(a, b) = (10^{d(0)}, 10^{d(1)})$ where $d(0) = \ell \operatorname{div} 2$ and $d(1) = \ell - d(0)$.
- (Number of rounds) VAES fixes the number of rounds to some constant, i.e. $r(\ell) = r$ for some fixed value $r$. We recommend no less than 7 rounds, and one can specify up to $2^8$ rounds.
- (Round PRF) The VAES round function is computed in two separate stages. An initial precomputation generates a key $K' = E_K(\mathsf{NtS}_8(|T|) \| \mathsf{NtS}_8(\ell) \| \mathsf{NtS}_{112}(T))$. The input to $\mathsf{AES}$ here is encodings of the tweak length, the digit length, and the tweak. Then during each round, the function $F_{K'}(i, x)$ for any $i \in [0 \,..\, 2^8 - 1]$ and $x \in \{0,1\}^{120}$ is computed by

  $$v \leftarrow \mathsf{NtS}_8(i) \| x \;\; ; \;\; y \leftarrow \mathsf{AES}(K_1, v) \;\; ; \;\; z \leftarrow \mathsf{StN}_{128}(y) \;\; ; \;\; \textbf{Ret } z$$

  As can be seen, the returned value is a number in $\mathbb{Z}_{2^{128}}$. Note that $r$ round VAES requires only $r+1$ calls to AES.

Design discussion. Here we explain various choices made in the design and discuss their rationale. We assess the security of the construction against known attacks, and also compare it to other alternative designs.

The VAES algorithm implements a Feistel network. The $i$-th round $(1 \leq i \leq r)$ splits its input into a left half of $d(0)$ bits and a right half of $d(1)$ bits if $i$ is odd, and vice versa if $i$ is even. The size of the splits is thus the same in every round if $\ell$ is even but varies if $\ell$ is odd, which is unusual. However, varying split sizes does not seem to degrade security.

In order to be able to encipher digit sequences rather than bit sequences, the round function outputs digits. The traditional XORs have been replaced by additions and subtractions modulo appropriate powers of 10. The round function converts bits to digits by interpreting the $\mathsf{AES}$ output as an integer and then taking the remainder upon division by the appropriate power of 10.

```
algorithm $\mathcal{E}_K^{T,\ell}(P[0]\dots P[\ell-1])$:
$d(0) \leftarrow \ell \text{ div } 2$ ; $d(1) \leftarrow \ell - d(0)$
$w \leftarrow \mathsf{NtS}_8(|T|)\|\mathsf{NtS}_8(\ell)\|\mathsf{NtS}_{112}(T)$
$K' \leftarrow \mathsf{AES}(K, w)$
$L_0 \leftarrow \mathsf{DtN}(P[0]\dots P[d(0)-1])$
$R_0 \leftarrow \mathsf{DtN}(P[d(0)]\dots P[d-1])$
For $i = 1, \dots, r$ do
    $L_i \leftarrow R_{i-1}$
    $s \leftarrow 1 - (i \bmod 2)$
    $x \leftarrow \mathsf{NtS}_{120}(R_{i-1})$
    $R_i \leftarrow (F_{K'}(i, x) + L_{i-1}) \bmod 10^{d(s)}$
$s \leftarrow r \bmod 2$
$C[0]\dots C[d(s)-1] \leftarrow \mathsf{NtD}_{d(s)}(L_r)$
$C[d(s)]\dots C[d-1] \leftarrow \mathsf{NtD}_{d(1-s)}(R_r)$
Ret $C[0]\dots C[d-1]$
```

```
algorithm $\mathcal{D}_K^{T,\ell}(C[0]\dots C[d-1])$:
$d(0) \leftarrow \ell \text{ div } 2$ ; $d(1) \leftarrow \ell - d(0)$
$w \leftarrow \mathsf{NtS}_8(|T|)\|\mathsf{NtS}_8(\ell)\|\mathsf{NtS}_{112}(T)$
$K' \leftarrow \mathsf{AES}(K, w)$
$s \leftarrow r \bmod 2$
$L_r \leftarrow \mathsf{DtN}(C[0]\dots C[d(s)-1])$
$R_r \leftarrow \mathsf{DtN}(C[d(s)]\dots C[d-1])$
For $i = r, \dots, 1$ do
    $R_{i-1} \leftarrow L_i$
    $s \leftarrow 1 - (i \bmod 2)$
    $x \leftarrow \mathsf{NtS}_{120}(R_{i-1})$
    $L_{i-1} \leftarrow (R_i - F_{K'}(i, x)) \bmod 10^{d(s)}$
$P[0]\dots P[d(0)-1] \leftarrow \mathsf{NtD}_{d(0)}(L_0)$
$P[d(0)]\dots P[d-1] \leftarrow \mathsf{NtD}_{d(1)}(R_0)$
Ret $P[0]\dots P[d-1]$
```

Figure 4: The VAES algorithm. The enciphering function is $\mathcal{E}$ and the deciphering function is $\mathcal{D}$. The number of rounds is $r$.

To discuss this further, let $N = 2^{128}$ and let $M = 10^m$ where $m$ is either $d(0)$ or $d(1)$. Ideally, we would like the round function to implement a random function with range $\mathbb{Z}_M$. In Appendix **??** we assess the statistical difference between implementing such a round function and use of a round function that has range $\mathbb{Z}_N$. We show there that this difference is at most about $M/N$, and since $M$ can be at most $10^{14}$ we get that the statistical difference is at most about $10^{14}/2^{128} = 2^{-81}$, which is clearly negligible.

Comparison to other approaches. We focus on comparison with Spies' FFSEM [25], which is (to our knowledge) the only concrete instantiation of an arbitrary finite set cipher. We emphasize, however, that FFSEM is *not* a general cipher as it can only handle messages from a single domain set. We therefore provide a comparison for encryption of a single set.

The major difference between VAES and FFSEM is that the former is significantly more efficient. Other differences are that the former, unlike the latter, is tweaked. The major design difference is that VAES works directly with digits, while FFSEM works over bits and then uses cycle walking to zero in on the correct domain. Let us now provide some details about the efficiency claim.

As an example, suppose we want to encrypt five digit plaintexts. FFSEM lets $m$ be smallest integer such that $2^{2m} \geq 10^5$, which results in $m = 9$. It then builds a balanced Feistel network over $2m$ bits. It then uses cycle walking to convert the final $2m$ bit result to a 5 digit result. Cycle walking works by reapplying the cipher. The expected number of applications of the cipher is $2^{2m}/10^5 \approx 2.6$. For the sake of a fair comparison, let us assume the two constructions use the same number of rounds, and also consider the non-tweaked version of VAES. Then, with seven rounds, VAES uses 7 computations of the underlying block cipher, while FFSEM always uses at least 7 and on the average uses 19. VAES is thus significantly more efficient.

# C Proof of Theorem 5.2

We replace the round functions with true random functions in the standard way to pass to the information theoretic setting. Then we note that any queries to **Enc** that do not have $T = T^*$ and $\ell = \ell^*$ will not help the adversary, and so without loss of generality we assume all queries are such. This means we can (for the purposes of the rest of this analysis) ignore tweaks and assume all queries are for the same domain set. We therefore can focus on an MR adversary $\mathcal{I}$ attacking

```
procedure Initialize                                    Game G0
$(\rho_1,\rho_3,\rho_5) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_b,\mathbb{Z}_a))^3$
$(\rho_2,\rho_4) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_a,\mathbb{Z}_b))^2$
$X_0^* X_1^* \xleftarrow{\$} \mathcal{I}_m$
$Z_1^* \leftarrow \rho_1(X_1^*)$ ; $X_2^* \leftarrow Z_1^* + X_0^* \bmod a$
$Z_2^* \leftarrow \rho_2(X_2^*)$ ; $X_3^* \leftarrow Z_2^* + X_1^* \bmod b$
$Z_3^* \leftarrow \rho_3(X_3^*)$ ; $X_4^* \leftarrow Z_3^* + X_2^* \bmod a$
$Z_4^* \leftarrow \rho_4(X_4^*)$ ; $X_5^* \leftarrow Z_4^* + X_3^* \bmod b$
$Z_5^* \leftarrow \rho_5(X_5^*)$ ; $X_6^* \leftarrow Z_5^* + X_4^* \bmod a$
Return $X_5^* X_6^*$

procedure Enc$(X_0, X_1)$
$i \leftarrow i+1$ ; $X_0^i \leftarrow X_0$ ; $X_1^i \leftarrow X_1$
If $X_0^i X_1^i = X_0^* X_1^*$ then WIN ← TRUE
$Z_1^i \leftarrow \rho_1(X_1^i)$ ; $X_2^i \leftarrow Z_1^i + X_0^i \bmod a$
$Z_2^i \leftarrow \rho_2(X_2^i)$ ; $X_3^i \leftarrow Z_2^i + X_1^i \bmod b$
$Z_3^i \leftarrow \rho_3(X_3^i)$ ; $X_4^i \leftarrow Z_3^i + X_2^i \bmod a$
$Z_4^i \leftarrow \rho_4(X_4^i)$ ; $X_5^i \leftarrow Z_4^i + X_3^i \bmod b$
$Z_5^i \leftarrow \rho_5(X_5^i)$ ; $X_6^i \leftarrow Z_5^i + X_4^i \bmod a$
Return $X_5^i X_6^i$
```

```
procedure Initialize                                    Game $\boxed{\text{G1}}$ G2
$(\rho_1,\rho_3,\rho_5) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_b,\mathbb{Z}_a))^3$
$(\rho_2,\rho_4) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_a,\mathbb{Z}_b))^2$
$X_0^* X_1^* \xleftarrow{\$} \mathcal{I}_m$
$Z_1^* \leftarrow \rho_1(X_1^*)$ ; $X_2^* \leftarrow Z_1^* + X_0^* \bmod a$
$Z_2^* \xleftarrow{\$} \mathbb{Z}_b$ ; $X_3^* \leftarrow Z_2^* + X_1^* \bmod b$
$Z_3^* \xleftarrow{\$} \mathbb{Z}_a$ ; $X_4^* \leftarrow Z_3^* + X_2^* \bmod a$
$Z_4^* \leftarrow \rho_4(X_4^*)$ ; $X_5^* \leftarrow Z_4^* + X_3^* \bmod b$
$Z_5^* \leftarrow \rho_5(X_5^*)$ ; $X_6^* \leftarrow Z_5^* + X_4^* \bmod a$
Return $X_5^* X_6^*$

procedure Enc$(X_0, X_1)$
$i \leftarrow i+1$ ; $X_0^i \leftarrow X_0$ ; $X_1^i \leftarrow X_1$
If $X_0^i X_1^i = X_0^* X_1^*$ then WIN ← TRUE
$Z_1^i \leftarrow \rho_1(X_1^i)$ ; $X_2^i \leftarrow Z_1^i + X_0^i \bmod a$
$Z_2^i \leftarrow \rho_2(X_2^i)$
If $X_2^i = X_2^*$ then bad1 ← TRUE $\boxed{;\ Z_2^i \leftarrow Z_2^*}$
$X_3^i \leftarrow Z_2^i + X_1^i \bmod b$
$Z_3^i \leftarrow \rho_3(X_3^i)$
If $X_3^i = X_3^*$ then bad2 ← TRUE $\boxed{;\ Z_3^i \leftarrow Z_3^*}$
$X_4^i \leftarrow Z_3^i + X_2^i \bmod a$
$Z_4^i \leftarrow \rho_4(X_4^i)$ ; $X_5^i \leftarrow Z_4^i + X_3^i \bmod b$
$Z_5^i \leftarrow \rho_5(X_5^i)$ ; $X_6^i \leftarrow Z_5^i + X_4^i \bmod a$
Return $X_5^i X_6^i$
```

Figure 5: Games used in proof of Theorem 5.2.

the FPF Feistel network using some fixed $a$ and $b$, no tweaks, and five independent random round functions $\rho_1, \rho_2, \rho_3, \rho_4, \rho_5$. For notational simplicity we have $\mathcal{I}_m$ output a pair $(X_0, X_1) \in (\mathbb{Z}_a, \mathbb{Z}_b)$ and ciphertexts returned to the adversary are also pairs in $\mathbb{Z}_b, \mathbb{Z}_a$. We utilize games to do the analysis bounding $\mathcal{I}$'s advantage; see Figure 5 and Figure 6. Game G0 implements the MR security game so restricted. So far we have justified that

$$\mathbf{Adv}^{\mathrm{mr}}_{\mathrm{FPF}}(\mathcal{I}) \le \mathbf{Adv}^{\mathrm{prf}}_{F}(\mathcal{B}) + \Pr\left[\,\mathrm{G0}^{\mathcal{A}} \text{ sets WIN}\,\right]. \tag{1}$$

We will now justify that

$$\begin{aligned}
\Pr\left[\,\mathrm{G0}^{\mathcal{I}} \text{ sets WIN}\,\right] &= \Pr\left[\,\mathrm{G1}^{\mathcal{I}} \text{ sets WIN}\,\right] && (2)\\
&\le \Pr\left[\,\mathrm{G1}^{\mathcal{I}} \text{ sets } \mathsf{bad1}\,\right] + \Pr\left[\,\mathrm{G1}^{\mathcal{I}} \text{ sets } \mathsf{bad2}\,\right] && (3)\\
&= \Pr\left[\,\mathrm{G2}^{\mathcal{I}} \text{ sets } \mathsf{bad1}\,\right] + \Pr\left[\,\mathrm{G2}^{\mathcal{I}} \text{ sets } \mathsf{bad2}\,\right] && (4)\\
&= \Pr\left[\,\mathrm{G3}^{\mathcal{I}} \text{ sets } \mathsf{bad1}\,\right] + \Pr\left[\,\mathrm{G3}^{\mathcal{I}} \text{ sets } \mathsf{bad2}\,\right] && (5)\\
&= \Pr\left[\,\mathrm{G4}^{\mathcal{I}} \text{ sets } \mathsf{bad1}\,\right] + \Pr\left[\,\mathrm{G4}^{\mathcal{I}} \text{ sets } \mathsf{bad2}\,\right] && (6)\\
&\le \Pr\left[\,\mathrm{G5}^{\mathcal{I}} \text{ sets } \mathsf{bad1}\,\right] + \Pr\left[\,\mathrm{G5}^{\mathcal{I}} \text{ sets } \mathsf{bad2}\,\right] && (7)\\
&= \Pr\left[\,\mathrm{G6}^{\mathcal{I}} \text{ sets } \mathsf{bad1}\,\right] + \Pr\left[\,\mathrm{G6}^{\mathcal{I}} \text{ sets } \mathsf{bad2}\,\right]. && (8)
\end{aligned}$$

Game G1 (boxed statements included) modifies the way in which G0 utilizes $\rho_2$ and $\rho_3$. In particular, the values mapped to by $X_2^*$ and $X_3^*$ are never used; $Z_2^*$ and $Z_3^*$ are chosen independently from $\rho_1$ and $\rho_2$. The added conditionals in Enc ensure consistency with the behavior of G0. The

```
procedure Initialize                                          Game G3
$(\rho_1, \rho_3, \rho_5) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_b, \mathbb{Z}_a))^3$
$(\rho_2, \rho_4) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_a, \mathbb{Z}_b))^2$
$X_0^* X_1^* \xleftarrow{\$} \mathcal{I}_m$
$Z_1^* \leftarrow \rho_1(X_1^*)$ ; $X_2^* \leftarrow Z_1^* + X_0^* \bmod a$
$X_3^* \xleftarrow{\$} \mathbb{Z}_b$ ; $Z_2^* \leftarrow X_3^* - X_1^* \bmod b$
$X_4^* \xleftarrow{\$} \mathbb{Z}_a$ ; $Z_3^* \leftarrow X_4^* - X_2^* \bmod a$
$Z_4^* \leftarrow \rho_4(X_4^*)$ ; $X_5^* \leftarrow Z_4^* + X_3^* \bmod b$
$Z_5^* \leftarrow \rho_4(X_5^*)$ ; $X_6^* \leftarrow Z_5^* + X_4^* \bmod a$
Return $X_5^* X_6^*$

procedure Enc$(X_0, X_1)$
$i \leftarrow i + 1$ ; $X_0^i \leftarrow X_0$ ; $X_1^i \leftarrow X_1$
$Z_1^i \leftarrow \rho_1(X_1^i)$ ; $X_2^i \leftarrow Z_1^i + X_0^i \bmod a$
$Z_2^i \leftarrow \rho_2(X_2^i)$ ; $X_3^i \leftarrow Z_2^i + X_1^i \bmod b$
$Z_3^i \leftarrow \rho_3(X_3^i)$ ; $X_4^i \leftarrow Z_3^i + X_2^i \bmod a$
$Z_4^i \leftarrow \rho_4(X_4^i)$ ; $X_5^i \leftarrow Z_4^i + X_3^i \bmod b$
$Z_5^i \leftarrow \rho_5(X_5^i)$ ; $X_6^i \leftarrow Z_5^i + X_4^i \bmod a$
Return $X_5^i X_6^i$

procedure Finalize
bad1 $\leftarrow (\exists j \,.\, X_2^j = X_2^*)$
bad2 $\leftarrow (\exists j \,.\, X_3^j = X_3^*)$
```

```
procedure Initialize                                          Game G4
$(\rho_1, \rho_3, \rho_5) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_b, \mathbb{Z}_a))^3$
$(\rho_2, \rho_4) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_a, \mathbb{Z}_b))^2$
$X_3^* \xleftarrow{\$} \mathbb{Z}_b$
$X_4^* \xleftarrow{\$} \mathbb{Z}_a$
$Z_4^* \leftarrow \rho_4(X_4^*)$ ; $X_5^* \leftarrow Z_4^* + X_3^* \bmod b$
$Z_5^* \leftarrow \rho_5(X_5^*)$ ; $X_6^* \leftarrow Z_5^* + X_4^* \bmod a$
Return $(X_3^* X_4^*, X_5^* X_6^*)$

procedure Enc$(X_0, X_1)$
$i \leftarrow i + 1$ ; $X_0^i \leftarrow X_0$ ; $X_1^i \leftarrow X_1$
$Z_1^i \leftarrow \rho_1(X_1^i)$ ; $X_2^i \leftarrow Z_1^i + X_0^i \bmod a$
$Z_2^i \leftarrow \rho_2(X_2^i)$ ; $X_3^i \leftarrow Z_2^i + X_1^i \bmod b$
$Z_3^i \leftarrow \rho_3(X_3^i)$ ; $X_4^i \leftarrow Z_3^i + X_2^i \bmod a$
$Z_4^i \leftarrow \rho_4(X_4^i)$ ; $X_5^i \leftarrow Z_4^i + X_3^i \bmod b$
$Z_5^i \leftarrow \rho_5(X_5^i)$ ; $X_6^i \leftarrow Z_5^i + X_4^i \bmod a$
Return $X_5^i X_6^i$

procedure Finalize
$X_0^* X_1^* \xleftarrow{\$} \mathcal{I}_m$
$Z_1^* \leftarrow \rho_1(X_1^*)$ ; $X_2^* \leftarrow Z_1^* + X_0^* \bmod a$
bad1 $\leftarrow (\exists j \,.\, X_2^j = X_2^*)$
bad2 $\leftarrow (\exists j \,.\, X_3^j = X_3^*)$
```

```
procedure Initialize                                          Game G5
$(\rho_1, \rho_3) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_b, \mathbb{Z}_a))^2$
$\rho_2 \xleftarrow{\$} \mathrm{Func}(\mathbb{Z}_a, \mathbb{Z}_b)$
$X_3^* \xleftarrow{\$} \mathbb{Z}_b$
$X_4^* \xleftarrow{\$} \mathbb{Z}_a$
$Z_4^* \leftarrow \hat{X}_5^* - X_3^* \bmod b$
$Z_5^* \leftarrow \hat{X}_6^* - X_4^* \bmod a$
Return $(X_3^* X_4^*, \hat{X}_5^* \hat{X}_6^*)$

procedure Enc$(X_0, X_1, X_5, X_6)$
$i \leftarrow i + 1$ ; $(X_0^i, X_1^i, X_5^i, X_6^i) \leftarrow (X_0, X_1, X_5, X_6)$
$Z_1^i \leftarrow \rho_1(X_1^i)$ ; $X_2^i \leftarrow Z_1^i + X_0^i \bmod a$
$Z_2^i \leftarrow \rho_2(X_2^i)$ ; $X_3^i \leftarrow Z_2^i + X_1^i \bmod b$
$Z_3^i \leftarrow \rho_3(X_3^i)$ ; $X_4^i \leftarrow Z_3^i + X_2^i \bmod a$
$Z_4^i \leftarrow X_5^i - X_3^i \bmod b$
$Z_5^i \leftarrow X_6^i - X_4^i \bmod a$

procedure Finalize
$X_0^* X_1^* \xleftarrow{\$} \mathcal{I}_m$
$Z_1^* \leftarrow \rho_1(X_1^*)$ ; $X_2^* \leftarrow Z_1^* + X_0^* \bmod a$
bad1 $\leftarrow (\exists j \,.\, X_2^j = X_2^*)$
bad2 $\leftarrow (\exists j \,.\, X_3^j = X_3^*)$
```

```
procedure Initialize                                          Game G6
$X_3^* \xleftarrow{\$} \mathbb{Z}_b$
$X_4^* \xleftarrow{\$} \mathbb{Z}_a$
$Z_4^* \leftarrow \hat{X}_5^* - X_3^* \bmod b$
$Z_5^* \leftarrow \hat{X}_6^* - X_4^* \bmod a$
Return $(X_3^* X_4^*, \hat{X}_5^* \hat{X}_6^*)$

procedure Finalize$(\tau)$
$(\rho_1, \rho_3) \xleftarrow{\$} (\mathrm{Func}(\mathbb{Z}_b, \mathbb{Z}_a))^2$
$\rho_2 \xleftarrow{\$} \mathrm{Func}(\mathbb{Z}_a, \mathbb{Z}_b)$
$(X_0^1, X_1^1, X_5^1, X_6^1), \ldots, (X_0^q, X_1^q, X_5^q, X_6^q) \leftarrow \tau$
For $i = 1$ to $q$ do
    $Z_1^i \leftarrow \rho_1(X_1^i)$ ; $X_2^i \leftarrow Z_1^i + X_0^i \bmod a$
    $Z_2^i \leftarrow \rho_2(X_2^i)$ ; $X_3^i \leftarrow Z_2^i + X_1^i \bmod b$
    $Z_3^i \leftarrow \rho_3(X_3^i)$ ; $X_4^i \leftarrow Z_3^i + X_2^i \bmod a$
    $Z_4^i \leftarrow X_5^i - X_3^i \bmod b$
    $Z_5^i \leftarrow X_6^i - X_4^i \bmod a$
$X_0^* X_1^* \xleftarrow{\$} \mathcal{I}_m$
$Z_1^* \leftarrow \rho_1(X_1^*)$ ; $X_2^* \leftarrow Z_1^* + X_0^* \bmod a$
bad1 $\leftarrow (\exists j \,.\, X_2^j = X_2^*)$
bad2 $\leftarrow (\exists j \,.\, X_3^j = X_3^*)$
```

Figure 6: Games used in the proof of Theorem 5.2. In games G5 and G6, the values $\hat{X}_5^*$ and $\hat{X}_6^*$ are hard-coded.

distribution of all random variables in G1 is therefore equivalent to G0. (The flags bad1 and bad2 might be set but have no bearing on the game.) This justifies (2). Whenever WIN is set in G1 necessarily bad1 and bad2 are set. This justifies (3). Game G2 drops the boxed statements of game G1. The games are identical-until-bad1 and identical-until-bad2 and so the fundamental lemma of game-playing justifies (4). Game G3 moves the setting of bad1 and bad2 to Finalize and modifies the order in which $X_3^*$, $Z_2^*$ and $X_4^*$, $Z_3^*$ are chosen. The distribution of the involved variables remains unchanged, justifying (5). In game G3 nothing but the setting of bad1 or bad2 relies on the message selected by $\mathcal{I}_m$, and so its selection and the computation of $Z_1^*$ and $Y_2^*$ are deferred to Finalize in G4. Also, G4 returns not only $X_5^* X_6^*$ but also $X_3^* X_4^*$. This can only increase the adversary's success probability. We have justified (6). We partially derandomize the game in G5, allowing the adversary to (effectively) choose range points of the cipher. First, game G5 has hard-coded into it a best value of $\hat{X}_5^* \hat{X}_6^*$ for the challenge ciphertext. (Note this is fixed independently of choice of $X_3^* X_4^*$.) Second, game G5 allows the adversary to specify both the inputs *and* outputs of encryption queries. Encryption no longer returns values. Technically, these changes allow the adversary to partially control the selection of the outputs of $\rho_4$ and $\rho_5$. Since these changes can only help the adversary, (7) holds. Finally, in game G6 we remove the encryption oracle completely, having the adversary simply output a transcript of all its queries and their responses. Equation (8) follows since in game G5 the adversary learned nothing from its queries anyway. Note also that in G6 selection of $\rho_1, \rho_2, \rho_3$ only need occur in **Finalize**.

We now must bound the setting of bad1 and bad2. We do this via the following two claims.

**Claim C.1** $\Pr\left[\, \mathrm{G6}^{\mathcal{I}} \text{ sets bad1} \,\right] \le \dfrac{q}{2^\mu} + \dfrac{q}{a}$ □

**Proof:** We must bound the probability that $X_2^* = X_2^i$ for some query $i$. Suppose query $i$ did not have right component equal to $X_1^*$. Then in this case the probability that $X_2^* = X_2^i$ is at most $q/a$. Suppose query $i$ has $X_1^i = X_1^*$. But in this case we see then that $X_2^* = X_2^i$ only if also $X_0^i = X_0^*$. The min-entropy requirement for $\mathcal{I}_m$ implies that the probability this can occur is at most $q/2^\mu$. ▮

**Claim C.2** $\Pr\left[\, \mathrm{G4}^{\mathcal{I}} \text{ sets bad2} \,\right] \le \dfrac{q}{b}$ □

**Proof:** We must bound the probability that $X_3^i \equiv Z_2^i + X_1^i \equiv \rho_2(X_2^i) + X_1^i \equiv X_3^* \pmod{b}$ for some $i \in [1 \mathinner{..} q]$. The selection of $\rho_2$ occurs after $X_3^*$ and all $X_1^i$ are fixed. Let $S_{X_2}$ be the set of query indices for which $\rho_2$ was evaluated on point $X_2 \in \mathbb{Z}_a$. For each $X_2 \in \mathbb{Z}_a$, then the probability that bad2 is set by any of the queries that evaluated $\rho_2(X_2)$ is at most $|S_{X_2}|/2^b$. That is bad2 is set if $\rho_2(X_2)$ equals any of the $|S_{X_2}|$ values defined as $X_3^* - X_1^i \bmod b$ for $i \in S_{X_2}$. However, we know that that $\sum_{X_2} |S_{X_2}| \le q$. Thus a union bound gives that the probability of setting bad2 is at most $q/2^b$.

▮