Generalized Format-Preserving Encryption for Character Data

Yanyu Huang[1], Bo Li[1], Shuang Liang[1], Haoyu Ma[2], and Zheli Liu[1](✉)

[1] College of Information Technical Science, Nankai University, Tianjin, China
onlyerir@163.com, nankailibo@163.com, nk_liangshuang@163.com,
liuzheli1978@163.com

[2] College of Network and Information, Xidian University, Xi'an, China
ma-haoyu@163.com

Abstract. We studied the problem on applying format-preserving encryption (FPE) to character data, specifically the uncertainty of the binary size of ciphertexts caused by variable-width encoding. In this paper, we suggested a extended rank-then-encipher approach for character data which connects character strings with numbers under mixed-radix numeral system. Based on this method, we proposed a generic character FPE scheme that deals with mixed-radix numerals, by introducing a customized "dynamic modulo addition" into unbalanced Feistel construction. Our work showed a new way of designing encryption methods for arbitrary message spaces which involves no tradeoff between efficacy and efficiency. Besides describing our design, security of our schemes are also analyzed.

Keywords: Block ciphers · Format-preserving encryption · Feistel networks · FFX mode · Mixed-radix numeral systems

1 Introduction

1.1 Problem of Applying FPE on Character Data

In recent years researches on applied cryptography have developed several practical enciphering methods, a paradigm is the so called *format-preserving encryption* (FPE). FPE aims to encipher messages of some specified format without disrupting it as achieving an acceptable level of security. Despite many efforts of designing FPE schemes, the work that has been done so far simply reduces concept "format" to "arbitrary domain", while other aspects other than the value of messages do not receive sufficient attention. In this paper, we emphasize the *variable-width encoding* of character data, and how it affects FPE application.

1.2 Related Work

Since FPE was first proposed in 1981, there have been plenty of researches on the subject [1–6,8]. In 2002, Black and Rogaway [1] provided a series of FPE methods on enciphering integers, and suggested that such ciphers can be used to construct FPE schemes on any arbitrary domain. In 2009, Bellare et al. [2] defined the *rank-then-encipher* approach (or RtE for short), and suggested that it's possible to construct any FPE scheme based on integer FPEs.

Some previous FPE schemes work on the message space $\mathcal{X} = \mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ for any desired N. Such schemes include both Feistel-based schemes like FFSEM [3], and other constructions such as card shuffle [4,5]. Some researchers present a method for keeping the database structure and supporting efficient SQL-based queries [16]. There are some application build the structure to achieve the function [13–15]. For Within these existing works, the FFX mode, proposed in 2010, is of the best generality [6]. Some researchers present a method for keeping the database structure and supporting efficient SQL-based queries [16]. FFX specifically aims on encrypting strings of some arbitrary alphabet Σ and works on the message space $\mathcal{X} = \Sigma^n$ for any desired string length n.

In a word, through all the current works on FPE, there is still no satisfying method in dealing character FPE. Clearly we need some better solutions.

1.3 Our Contributions

In this paper, we provide an effective and efficient solution for character FPE problem that can encipher character strings while preserving their length and memory consumption. In detail, our contributions are:

Firstly, we suggest that character alphabets can be ranked using an improved RtE method, where the characters are represented by extended position notation called *mixed-radix numeral systems*.

Secondly, we propose and analyze a character FPE scheme based on Feistel, and extend from the FFX mode to be able to use mixed-radix numerals.

2 Preliminaries

2.1 Format-Preserving Encryption

We start with a brief review to the classical definition of format-preserving encryption [2], described as:

Definition 1 (Format-preserving encryption). *A format-preserving encryption scheme is a function*

$$\boldsymbol{F} : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X} \cup \{\perp\}, \tag{1}$$

where $\mathcal{K}$, $\mathcal{N}$, $\mathcal{T}$, $\mathcal{X}$ are called the key space, format space, tweak space and domain, respectively. All of them are nonempty and $\perp \notin \mathcal{X}$.

All FPEs work on some subspaces of the domain $\mathcal{X}$, determined by a certain format in $\mathcal{N}$, named *slice*:

Definition 2 (Slice on a message space). *Given a concrete format $N \in \mathcal{N}$, the N-indexed slice is defined as*

$$\mathcal{X}_N = \{X \in \mathcal{X} | \forall \{K, T\} \in \mathcal{K} \times \mathcal{T}, E_K^{N,T}(X) \in \mathcal{X}\}, \tag{2}$$

where $E_K^{N,T}(X)$ returns the ciphertext of X.

FPE requires that any $X \in \mathcal{X}$ lives in at least one slice indexed by some $N \in \mathcal{N}$. It also requires that any slice $\mathcal{X}_N$ is finite for all $N \in \mathcal{N}$. For any $\{K, T\} \in \mathcal{K} \times \mathcal{T}$, both the encipher and decipher process should be permutations on $\mathcal{X}_N$, and whether $E_K^{N,T}(X) = \perp$ or not depends only on the format and the plaintext, but not on K and T.

2.2 Security Notion

Throughout all the security notions that FPEs could be after, the PRP notion is mostly used. Let $\mathcal{E} : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a block cipher, and let $\mathcal{A}^{E(\cdot)}$ indicates an adversary $\mathcal{A}$ with an oracle E, which may ask any encryption query $E(\cdot)$. Denote $\varepsilon \overset{\$}{\leftarrow} \mathcal{E}_K$ as to pick a key K randomly from $\mathcal{K}$ and return $\mathcal{E}_K(\cdot)$, and denote $\pi \overset{\$}{\leftarrow} Perm(\mathcal{X})$ as to pick a permutation π on $\mathcal{X}$ randomly and return $\pi(\cdot)$. Then the adversary's advantage is given by

$$\mathbf{Adv}_E^{PRP}(\mathcal{A}) \overset{def}{=} Pr[\varepsilon \overset{\$}{\leftarrow} \mathcal{E}_K, \mathcal{A}^{\varepsilon(\cdot)} = 1] - Pr[\pi \overset{\$}{\leftarrow} Perm(\mathcal{X}), \mathcal{A}^{\pi(\cdot)} = 1]. \tag{3}$$

3 Introducing Mixed-Radix Numeration to Character FPE

3.1 Notations

Denote the set of all possible characters as $Chars$. We know that $Chars$ is finite and $|Chars| = \boldsymbol{c}$. Given any two character strings $A, B \in Chars^*$ (by $*$ we mean that they each consists of any arbitrary number of characters), denote $A \parallel B$ (or AB in short) as their concatenation. Thus any character string X can be represented by $X = x_1 x_2 \cdots x_i, x_* \in Chars$. Additionally, we let $l(X)$ be the number of characters in string X (henceforth the *length* for short), and $s(X)$ be its binary size (henceforth the *size*), we believe that to fully describe the string Z, all of $Chars$, $l(X)$, and $s(X)$ are necessary. Obviously for single characters $x \in Chars$, the number of bytes needed in encoding it is given by $s(x)$. If let a set $\Psi = \{\psi_1, \cdots, \psi_I\}$ be all possible binary sizes of single characters, then Ψ determines a partition of $Chars$:

$$Chars = \bigcup_{i=1}^{I} C_i, \forall c \in Chars, c \in C_i \Leftrightarrow s(c) = \psi_i, \tag{4}$$

where each of C_i is a subset of $Chars$ in which the binary size of any character is ψ_i. We believe that the given partition could help reaching a satisfying solution for character FPE problem.

3.2 Mixed-Radix Numeration: A Promising Way

Character strings to mixed-radix numerals. As the matter of fact, character data is not the only one that results in a complex and irregular message space. For example, a full date of AD chronology can be considered as 3-digit numbers with each digit respectively in the domain $\mathbb{Z}_{31}$, $\mathbb{Z}_{12}$ and $\mathbb{Z}_{9999}$ (at least for now), thus the date 24-03-1998 is actually also a number $24_{31}3_{12}1998_{9999}$, where the subscripts are to represent the radixes of each digit. Similarly, suppose there are an octahedral dice, a hexahedral dice and a tetrahedral dice, the sample space of the statistical event "successively roll the three dices, and return the results in sequence" can also be described by a 3-digit number with each digit respectively in $\mathbb{Z}_8$, $\mathbb{Z}_6$ and $\mathbb{Z}_4$. Both the examples are to use a kind of non-standard positional numeral systems known as *mixed-radix numeral systems* [9], in which the numerical base varies from position to position. Such numeral systems are able to precisely represent any particular message space, as long as the amount of elements is a composite number of which all factors are known. Therefore, the mixed-radix numeration might just be what needed in building FPE schemes on arbitrary message spaces, like that of character data.

Structure of character strings. With set Ψ given in the previous section, we suggest a notion called the *structure* to describe the format of character data in the sense of mixed-radix numeration:

Definition 3 (Structure of character data). *Without loss of generality, for a character string $X \in Chars^n = x_1x_2\cdots x_n$, let*

$$\omega_i = |\{x_j \in X | s(x_j) = \psi_i, 1 \leqslant j \leqslant n\}|, i \in \{1, 2, \ldots, I\} \tag{5}$$

be the number of characters in X that is encoded with ψ_i bytes, the structure of X is therefore defined as

$$\Omega(X) = \{\omega_{1\psi_1}, \omega_{2\psi_2}, \ldots, \omega_{I\psi_I}\}. \tag{6}$$

Figure 1 gives a intuitive example on how the structure we defined works on the character strings. Additionally, the structure refines the notion of "format" for character data, because of the following properties it has:

Proposition 1 (Properties of the structure). *For the structure $\Omega(\cdot)$ of character data, the following statements holds:*

1. *For any $X \in Chars^*$, $\Omega(X)$ is unique.*
2. *For any $A, B \in Chars^*$, $X = A \parallel B \Leftrightarrow \Omega(X) = \Omega(A) + \Omega(B)$.*
3. *For any $A, B \in Chars^*$, $\Omega(A) = \Omega(B) \Leftrightarrow (l(A), s(A)) = (l(B), s(B))$ because:*

$$\forall X \in Chars^*, \{l(X) = \sum_{i=1}^{I} \omega_i s(X) = \sum_{i=1}^{I} \omega_i \psi_i,\ \omega_i \in \Omega(X), \psi_i \in \Psi. \tag{7}$$

Therefore, we believe that it is the key basis to extend RtE approach for character data, and to further build character FPE schemes on.

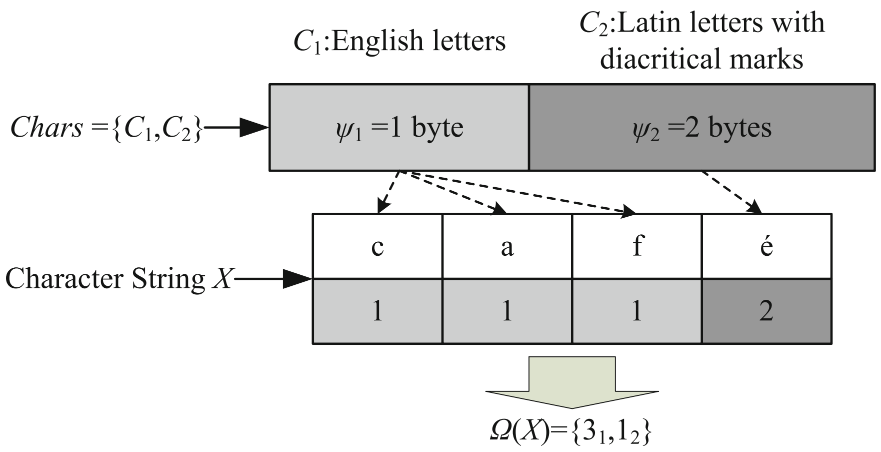


Fig. 1. This figure gives an illustration of the structure of character strings. Assume the overall character set *Chars* consists of 2 subsets: C_1 containing the modern English alphabet and C_2 containing Latin letters with diacritical marks, respectively takes 1 byte and 2 bytes to be represented (i.e. $\Psi = \{\psi_1 = 1, \psi_2 = 2\}$). Then since the string $X =$ "café" has 3 characters from C_1 and 1 character from C_2, its structure is given by $\Omega(X) = \{3_1, 1_2\}$, from which we know that this 4-character string takes a memory space of 5 bytes.

3.3 The Extended Rank-Then-Encipher Approach

In order to build character FPE schemes that work under mixed-radix numeration, we extend the RtE approach, to rank character strings with mixed-radix notation. Note that though the structure of character data is the basis of our work, it cannot directly do this kind of ranking. To do that we mainly exploited the partition on *Chars* given at Sect. 3.1. Recall that *Chars* is divided into subsets $C_i(i = 1, 2, \cdots, I)$, in which all characters are of the same binary size ψ_i. Obviously a bijective mapping can be built between elements in C_i and the integer domain $\mathbb{Z}_{|C_i|}$, i.e. each character in C_i can be mapped to a integer in $\{0, 1, \cdots, |C_i| - 1\}$. Thus when characters are treated as digits of a mixed-radix number in RtE, a feasible way is to decide the radixes according to the subset where the characters belong to. This is done by denoting all $c \in Chars$ with the following 2-tuple:

$$c = (v(c), t(c)) : c \in C_i \Leftrightarrow t(c) = iv(c) \in \mathbb{Z}_{|C_i|}, \tag{8}$$

where $t(c)$ is the *tag* of the character that marks the subset it belongs to, and $v(c)$ is the value of it. Correspondingly, for the subsets C_*, a table is kept to record $|C_*|$, so that the radix of a character can be easily determined by its tag. Take the demonstration in Fig. 1 as an example, suppose the character "é" is the 10-th character in the subset C_2, which contains a total of 100 characters (i.e. $|C_2| = 100$), then the ranking routine will recognize "é" as $(9, 2)$, and thus find its radix 100. Moreover, with digits in the form of such 2-tuples, the structure of a character string is also indicated by the tags in its ranking result, thus the proposed ranking approach is a important reference in preserving the format of character strings.

4 C-FFX: A Generic Solution for Character FPE

Under such guiding ideology, we propose a generic and efficient scheme for character FPE using the unbalanced Feistel construction [11]. Built based on the FFX mode construction, in this scheme (which we call the "C-FFX") we extende the RtE approach to rank character strings with mixed-radix numerals, and applied a specially designed dynamic modulo addition in our construction, which is able to operate between k-digits mixed-radix numerals, to ensure the format of the scheme's output remains the same as its input.

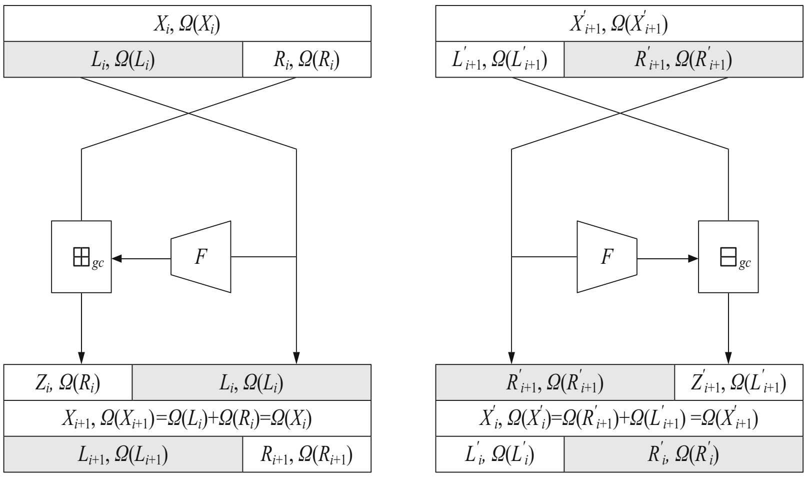
4.1 Feistel-Based Construction with Dynamic Modulo Addition

Recall that our goal of character FPE designing is to preserve the structure of character strings, i.e. for an input string X and its corresponding output string Y, a character FPE should ensure that $\Omega(X) = \Omega(Y)$ always hold.

For concision, given a character string $X \in Chars^n$, let $|X| = n$ be its length, and denote the i-th digit of X as $X[i]$ $(i = 1, 2, \dots, n)$. For $1 \leqslant i < j \leqslant n$, let $X[i..j] = X[i] \parallel X[i+1] \parallel \cdots \parallel X[j]$. Also we denote $\boldsymbol{cmin} = min(|C_i|, i = 1, 2, \dots, I), \boldsymbol{cmax} = max(|C_i|, i = 1, 2, \dots, I)$ respectively as the amount of elements of the smallest and largest subset of $Chars$, and define $\mathcal{X} = |Chars^n_{\Omega(X)}|$ as the number of character strings with the structure of string X. It's easy to know that $\boldsymbol{cmax}^{|X|} \geqslant \mathcal{X} \geqslant \boldsymbol{cmin}^{|X|}$. Additionally, since in each round of any Feistel, the input string X is split into two substrings (denoted as L and R, for the left part and the right part), again assume $|X| = n$, we also let $l = |L|$ and $n - l = |R|$.

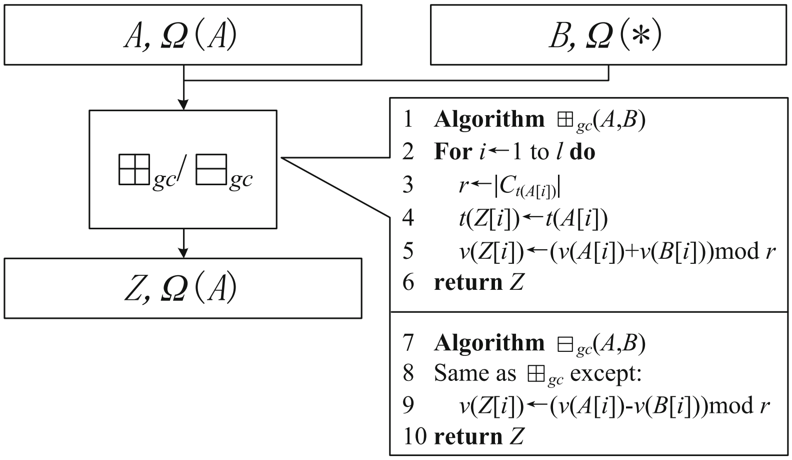
In Fig. 2 we show the structure of one round of C-FFX, where F is the round function and $\boxplus_{gc}/\boxminus_{gc}$ is the modulo addition/subtraction module we designed. Since, as mentioned, in each round the input string X is split into two substrings L and R, obviously $\Omega(X) = \Omega(R) + \Omega(L)$. Without loss of generality, suppose L directly goes to the right of the output Y, then certainly $\Omega(Y) = \Omega(L) + \Omega'$. Therefore to make $\Omega(Y) = \Omega(X)$, it's easy to know that $\Omega' = \Omega(R)$. As we know, the rest part of Y is generated by $L \boxplus_{gc} F(R)$ or $L \boxminus_{gc} F(R)$, therefore to ensure $\Omega(Y) = \Omega(X)$ is to let $\Omega(L \boxplus_{gc} F(R))/\Omega(L \boxminus_{gc} F(R)) = \Omega(R)$. It's not quite realistic to design a round function that returns a mixed-radix numeral pseudo-randomly, while the radixes of the output varies at each round. Thus in order to do achieve out goal, we made some major improvement on the modulo addition function.

The modified modulo addition and its inverse, which we called the "dynamic modulo addition" $\boxplus_{gc}$ and $\boxminus_{gc}$, are digit-wise operations, meaning that they process the input pair digit by digit. Assume that string A and B are the two inputs of $\boxplus_{gc}$ or $\boxminus_{gc}$, and suppose we demand that the output of the operations


$X_i, \Omega(X_i)$
$L_i, \Omega(L_i)$
$R_i, \Omega(R_i)$
$\boxplus_{gc}$
F
$Z_i, \Omega(R_i)$
$L_i, \Omega(L_i)$
$X_{i+1}, \Omega(X_{i+1})=\Omega(L_i)+\Omega(R_i)=\Omega(X_i)$
$L_{i+1}, \Omega(L_{i+1})$
$R_{i+1}, \Omega(R_{i+1})$
$X'_{i+1}, \Omega(X'_{i+1})$
$L'_{i+1}, \Omega(L'_{i+1})$
$R'_{i+1}, \Omega(R'_{i+1})$
F
$\boxminus_{gc}$
$R'_{i+1}, \Omega(R'_{i+1})$
$Z'_{i+1}, \Omega(L'_{i+1})$
$X'_i, \Omega(X'_i)=\Omega(R'_{i+1})+\Omega(L'_{i+1})=\Omega(X'_{i+1})$
$L'_i, \Omega(L'_i)$
$R'_i, \Omega(R'_i)$

(a) The enciphering process (b) The deciphering process

Fig. 2. A single round of the C-FFX construction.



```
1  Algorithm $\boxplus_{gc}(A,B)$
2  For $i \leftarrow 1$ to $l$ do
3      $r \leftarrow |C_{t(A[i])}|$
4      $t(Z[i]) \leftarrow t(A[i])$
5      $v(Z[i]) \leftarrow (v(A[i])+v(B[i])) \bmod r$
6  return $Z$

7  Algorithm $\boxminus_{gc}(A,B)$
8  Same as $\boxplus_{gc}$ except:
9      $v(Z[i]) \leftarrow (v(A[i])-v(B[i])) \bmod r$
10 return $Z$
```

Fig. 3. Mechanism of the modulo addition function $\boxplus_g c$ and $\boxminus_g c$. Both functions traverse the two input strings A and B (line 2), in the i-th loop, line 3 uses tag $t(A[i])$ to determine the radix r for the modulo operation, thus the content of the output digit $v(Z[i])$ computed in line 5 or 9 is kept in the range of subset $C_{t(A[i])}$. Since line 4 sets tag $t(Z[i])$ to be the same as $t(A[i])$. Therefore according to Eq. 8, this ensures that $Z[i]$ belongs to $C_{t(A[i])}$.

(denoted as Z) to have the same structure as A (i.e. $\Omega(Z) = \Omega(A)$), then for each digit i of Z, our operations first computes the sum of the corresponding digit $A[i]$ and $B[i]$, then modulo the result with the radix referenced by the tag $t(A[i])$. Since the tags of each digit of the output is exactly the same as the input

A, this procedure ensures that the output is of A's structure. The radix of input strings varies from digit to digit, so are the operations, thus they are considered "dynamic". As shown in Fig. 3, the 2-tuple notation in the extended RtE makes the operations easy to realize.

C-FFX gives a practically effective solution to character FPE problem, which is theoretically able to be applied on any character data, encipher/decipher them correctly in fixed rounds, without using cycle-walking or repeated encryption.

4.2 Security

Regardless the modification we made, by definition our construction is an unbalanced Feistel with an arbitrary alphabet. Thus same as other studies on the provable-security analysis of Feistel networks, the round functions used are assumed to be selected uniformly and independently at random.

Security bound. In [12], the authors mentioned that the CCA bound for binary unbalanced Feistels given in its Theorem 7 can be extended to unbalanced Feistels with arbitrary alphabet. Although C-FFX uses mixed-radix numeration and thus the radix of its alphabet is not fixed, its CCA bound can still be given by:

Theorem 1 (CCA security of C-FFX). *Denote a C-FFX as ϵ, given* $\boldsymbol{cmin}$ *and a fixed $\tau \geq 1$, while $l > n - l$, we have*

$$\mathbf{Adv}_{\epsilon}^{CCA}(q) \leq \frac{2q}{\tau+1}((3\lceil l/(n-l)\rceil + 3)q/\boldsymbol{cmin}^{l})^{\tau}, \tag{9}$$

while

$$r = \tau(4\lceil l/(n-l)\rceil + 4) \tag{10}$$

is the minimum number of Feistel rounds needed.

To proof the above result works for C-FFX, the only thing we need to do is to reinterpret Lemmas 11 and 12 in [12] to analyze the case when the construction works on mixed-radix numeration, since Theorem 7 in [12] is deduced based on these two lemmas:

Lemma 1 (reinterpretation of Lemma 11 in [12]). *In C-FFX, the chance that two distinct non-adaptive queries have the same coin at round $t \geq 1$ is at most* $\boldsymbol{cmin}^{l-n}$*.*

Proof. Since the scheme is designed to preserve the structure of character strings, in each round, the structure of both input and output of the Feistel is assumed to be known (this is due to that the only thing that a Feistel round will do to the structure of its input is to swap positions of the digits in a constant way).

Suppose that a C-FFX scheme receives distinct non-adaptive queries X_1 and X_2. For each $i \in \{1, 2\}$, let (L_i, R_i) be the output at round $t-1$ of X_i, where $|L_i| = l$ and $|R_i| = n - l$. The queries X_1 and X_2 collide at time t if and only if $R_1 \boxplus_{gc} F(L_1) = R_2 \boxplus_{gc} F(L_2)$, with F being the round function at round t. This occurs when R_1 and R_2 differ, with probability $|Chars^{n-l}_{\Omega(R)}|^{-1} = \mathcal{R}^{-1} \leq \boldsymbol{cmin}^{l-n}$, because F is uniformly random. If $R_1 = R_2$ then so are L_1 and L_2, which contradicts the hypothesis that the two queries are distinct.

Lemma 2 (reinterpretation of Lemma 12 in [12]). *In C-FFX, the chance that two distinct non-adaptive queries collide at time $t > \lceil l/(n-l) \rceil$ is at most $3/\boldsymbol{cmin}^b$.*

Proof. Suppose that a C-FFX scheme receives distinct non-adaptive queries X_1 and X_2. We shall prove by induction on b that for any $b \leq l$, the probability that outputs at round $t > \lceil b/(n-l) \rceil$ of the two queries have the same last b digits is at most $3/\boldsymbol{cmin}^b$. The claim of this lemma corresponds to the special case $b = l$.

First consider the base case $b < n - l$. For each $i \in \{1, 2\}$, let (L_i, R_i) be the output at round $t-1$ of X_i, where $|L_i| = l$ and $|R_i| = n-l$. The last $(n-l)$-digit substring of the round-t output of X_i is $R_i \boxplus_{gc} F(L_i)$, with F being the round function at round t. If R_1 and R_2 differ then the probability that outputs at round t of the two queries have the same last b digits is at most $\boldsymbol{cmin}^{-b}$ (the same reason as in Lemma 2). If $R_1 = R_2$ then the two queries have the same coin at round $t-1$, which by Lemma 2 occurs with probability at most $\boldsymbol{cmin}^{l-n}$. Hence, by union bound, the chance that the two queries have the same last b digits is at most $\boldsymbol{cmin}^{-b} + \boldsymbol{cmin}^{l-n} \leq 3/\boldsymbol{cmin}^b$.

Next consider $b \geq n - l$ and assume that the chance round-$(t-1)$ outputs of the two queries have the same last $b - n + l$ digits is at most $3/\boldsymbol{cmin}^{b-n+l}$. The outputs at round t of the two queries have the same last b digits if and only if (i) they have the same coin at round t, which by Lemma 2 occurs with probability at most $\boldsymbol{cmin}^{l-n}$, and (ii) their output at round $t-1$ have the same last $b - n + l$ digits, which occurs with probability at most $3/\boldsymbol{cmin}^{b-n+l}$ by induction hypothesis. As the round functions in the network are independent, the chance that both (i) and (ii) occur is at most $\boldsymbol{cmin}^{l-n} \cdot 3/\boldsymbol{cmin}^{b-n+l} = 3/\boldsymbol{cmin}^b$.

Notice that according to [12], the above extension is only known to be work when the round functions are contracting, which is the reason that we set $l > n-l$ in out construction.

5 Conclusion

In this paper we stated the problem in applying format-preserving encryption on character data, as well as analyzed the fundamental reason of it. By introducing mixed-radix numeral systems to character FPE, we refined the format of character data, and extended the rank-then-encipher approach for character FPE. On the top of these, we proposed the C-FFX scheme as a generic character FPE solution, which adopts Feistel-based construction with a specially built dynamic modulo addition module, so that mixed-radix numerals can be processed. Analysis showed that our scheme provides solid security. In our future works, we plan to further design FPE schemes that work under mixed-radix numeration, based on other constructions.

Acknowledgment. This work is supported by National Natural Science Foundation of China (No. 61672300), National Natural Science Foundation of Tianjin (Nos. 16JCYBJC15500 and 14JCYBJC15300).

References

1. 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). https://doi.org/10.1007/3-540-45760-7_9
2. 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). https://doi.org/10.1007/978-3-642-05445-7_19
3. Spies, T.: Feistel finite set encryption. NIST submission (2008)
4. 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). https://doi.org/10.1007/978-3-642-03356-8_17
5. Hoang, V.T., Morris, B., Rogaway, P.: An enciphering scheme based on a card shuffle. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 1–13. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_1
6. Bellare, M., Rogaway, P., Spies, T.: The FFX mode of operation for format-preserving encryption. NIST submission (2010)
7. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Comput. **17**, 373–386 (1988)
8. Li, M., Liu, Z.L., Li, J.W., Jia, C.F.: Format-preserving encryption for character data. J. Netw. **7**, 1239–1244 (2012)
9. Fraenkel, A.S.: Systems of numeration. Am. Math. Mon. **92**, 105–114 (1985)
10. Patarin, J.: Generic attacks on Feistel schemes. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 222–238. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_14
11. 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). https://doi.org/10.1007/3-540-60865-6_49
12. Patarin, J.: About Feistel schemes with six (or more) rounds. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 103–121. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-69710-1_8
13. Rozenberg, B., Weiss, M.: Complex format-preserving encryption scheme. 14/296484 (2015)
14. Hoover, D.N.: Format-preserving encryption via rotating block encryption. US8948376 B2 (2015)
15. Spies, T., Pauker, M.J.: Format-preserving cryptographic systems. US8958562 B2 (2015)
16. Li, J., Liu, Z.L., Chen, X.F., Xhafa, F., Tan, X., Wong, D.S.: L-EncDB: a light-weight framework for privacy-preserving data queries in cloud computing. Knowl.-Based Syst. **79**, 18–26 (2015)