Addendum to
"The FFX Mode of Operation for Format-Preserving Encryption"

A parameter collection for enciphering strings of arbitrary radix and length

M. Bellare[1] P. Rogaway[2] T. Spies[3]

Draft 1.0
September 3, 2010

1 Introduction

Background. A scheme for *format-preserving encryption* (FPE) is supposed to do that which a conventional (possibly tweakable) blockcipher does—encipher messages within some message space $\mathcal{X}$—except that message space, instead of being something like $\mathcal{X} = \{0,1\}^{128}$, is more general [1, 3]. For example, the message space might be the set $\mathcal{X} = \{0,1,\ldots,9\}^{16}$, in which case each 16-digit plaintext $X \in \mathcal{X}$ gets enciphered into a 16-digit ciphertext $Y \in \mathcal{X}$. In a *string-based* FPE scheme—the only type of FPE that we consider here—the message space is of the form $\mathcal{X} = \{0,1,\ldots,\mathsf{radix}-1\}^n$ for some message length n and alphabet size radix.

One way to achieve FPE is with a *Feistel-based mode of operation.* The underlying round function can be based on a conventional blockcipher, say AES. Two proposals along these lines were recently submitted to NIST. One is FFX, by the present authors [2]. Another is BPS, by Brier, Peyrin, and Stern [4], which was developed independently and submitted soon after FFX.

The FFX scheme is more open-ended than BPS; by itself, it is more a framework than a mode. To turn FFX into a concrete mode one has to fix a variety of *parameters*—some nine parameters in all. To make things concrete, the authors of FFX suggest two *parameter collections*, named A2 and A10. Scheme FFX-A2 encrypts binary strings of 8–128 bits, while FFX-A10 encrypts decimal strings of 4–36 digits. For sufficiently long strings, both schemes use 12 rounds of Feistel. But the number of rounds escalate as messages get shorter, going as high as 24 rounds (for FFX-A10) or 36 rounds (for FFX-A2) for the shortest permitted message lengths.

BPS brings to the table two new characteristics that a user might like. First is a more aggressive round count: exactly eight rounds are used, regardless of the input's length. Fewer rounds mean

[1] University of California, San Diego, USA. e-mail: mihir@cs.ucsd.edu.
[2] University of California, Davis, USA. e-mail: rogaway@cs.ucdavis.edu.
[3] Voltage Security, USA. email: terence@voltage.com.

greater speed (and, of course, a less conservative design). Second, BPS includes a CBC-like chaining mode to allow for the encryption of very long strings.

We note that the CBC-like mode of BPS—or, more generally, any CBC-like mode—cannot possibly meet the security requirements specified in the literature for an FPE [1, 3]: you will *not* get a pseudorandom permutation (PRP). To begin with, the first bit of ciphertext does not depend on the last bit of plaintext. We feel that any scheme that calls itself a scheme for format-preserving encryption ought to achieve the PRP goal—and preferably the *strong* PRP goal as well.

THE FFX[radix] MODE. This document provides new FFX parameter collections. First, we expand the allowed radix values; now, any reasonable radix may be used, not just 2 or 10. Second, we enlarge the allowed message lengths, permitting effectively arbitrary strings to be enciphered. Finally, we select more aggressive round counts than we did for FFX-A2 and FFX-A10.

To accomplish the above, we define the FFX scheme we call FFX[radix]. The bracketed value compactly names an FFX parameter collection. The value of radix is a number between 2 and 2^{16}. Messages to be enciphered under FFX[radix] are regarded as strings of characters drawn from the alphabet $\mathsf{Chars} = \{0, 1, 2, \ldots, \mathsf{radix} - 1\}$. Scheme FFX[radix] does its work using an AES-based balanced Feistel network. If the message length is odd, an alternating, maximally-balanced Feistel scheme is used instead.

Mode FFX[radix] effectively unifies and extends FFX-A2 and FFX-A10; we suggest its use in lieu of the later modes. The radix is more general, messages can be longer, and the number of rounds is made constant, rather than depending on the message length n. We have not tried to maintain upward compatibility; mode FFX[2] and FFX[10] do not coincide with with FFX-A2 and FFX-A10, and they would not coincide even if the same number of rounds had been employed (although, in that case, there would be no cryptographically significant difference).

2 The Scheme FFX[radix]

NOTATION. We assume the notation from the FFX spec [2] but, for the user's convenience, we recall the most relevant definitions here. Plaintexts and ciphertexts are regarded as strings over the alphabet $\mathsf{Chars} = \{0, 1, \ldots, \mathsf{radix} - 1\}$. By $|X|$ we denote the length of string X, the number of characters in it. Let BYTE denote $\{0,1\}^8$, the set of 8-bit bytes. By $|T|_8$ we denote the length, in bytes, of the byte string $T \in \text{BYTE}^*$. With radix understood, the blockwise addition function $\boxplus$ is defined by $a_1 \cdots a_n \boxplus b_1 \cdots b_n = c_1 \cdots c_n$ where $c_1 \cdots c_n$ is the unique string such that $\sum c_i \,\mathsf{radix}^{n-i} = \left(\sum a_i \,\mathsf{radix}^{n-i} + \sum b_i \,\mathsf{radix}^{n-i}\right) \bmod \mathsf{radix}^n$. By $X \boxminus Y$ we mean the unique string Z such that $Y \boxplus Z = X$. By $[s]^i$ we mean the i-byte string that encodes the number $s \in [0..2^{8i}-1]$. The function $\text{NUM}_{\mathsf{radix}}(X)$ takes a nonempty string $X \in \{0, \ldots, \mathsf{radix} - 1\}^*$ and converts it to the corresponding number, where the number is interpreted in the given radix, most-significant character first. The function $\text{STR}^m_{\mathsf{radix}}(x)$ takes a number $x \in [0 \,..\, \mathsf{radix}^m - 1]$ and returns the m-character string that represents it in the the given radix, most significant character first.

When $K \in \{0,1\}^{128}$ and $X \in \{0,1\}^*$ and $|X|$ is divisible by 128, algorithm CBC-MAC$_K(X)$ is defined as follows. First, let $X_1 \cdots X_m \leftarrow X$ where $|X_i| = 128$ and let $Y \leftarrow 0^{128}$. Then, for $j \leftarrow 1$ **to** m, set $Y \leftarrow \text{AES}_K(Y \oplus X_i)$. Finally, return Y.

```
10  algorithm FFX.Encrypt(K, T, X)
11  if K ∉ Keys or T ∉ Tweaks or
12     X ∉ Chars* or |X| ∉ Lengths
13     then return ⊥
14  n ← |X|;  ℓ ← split(n);  r ← rnds(n)
15  A ← X[1 .. ℓ];  B ← X[ℓ + 1 .. n]
16  for i ← 0 to r − 1 do
17      C ← A ⊞ F_K(n, T, i, B)
18      A ← B;  B ← C
19  return A ‖ B
```

```
20  algorithm FFX.Decrypt(K, T, Y)
21  if K ∉ Keys or T ∉ Tweaks or
22     Y ∉ Chars* or |Y| ∉ Lengths
23     then return ⊥
24  n ← |Y|;  ℓ ← split(n);  r ← rnds(n)
25  A ← Y[1 .. ℓ];  B ← Y[ℓ + 1 .. n]
26  for i ← r − 1 downto 0 do
27      C ← B;  B ← A
28      A ← C ⊟ F_K(n, T, i, B)
29  return A ‖ B
```

radix	a number radix ∈ [2 .. 2^{16}]	alphabet is Chars = {0, 1, . . . , radix − 1}
Lengths	[minlen .. maxlen] where minlen = 2 if radix ≥ 10 and minlen = 8 otherwise; and maxlen = $2^{32} - 1$.	permitted message lengths
Keys	$\{0,1\}^{128}$	128-bit AES keys
Tweaks	$\text{BYTE}^{\leq \text{maxlen}}$ where maxlen = $2^{32} - 1$	tweaks are arbitrary byte strings
addition	1	blockwise addition
method	2	alternating Feistel
split (n)	$\lfloor n/2 \rfloor$	maximally balanced Feistel
rnds (n)	10	number of rounds
F	given below	AES-based round function

```
30  algorithm F_K(n, T, i, B)
31  vers ← 1;  t ← |T|_8;  β ← ⌈n/2⌉;  b ← ⌈⌈β log_2(radix)⌉ / 8⌉;  d ← 4⌈b/4⌉
32  if EVEN(i) then m ← ⌊n/2⌋ else m ← ⌈n/2⌉

33  P ← [vers]^1 ‖ [method]^1 ‖ [addition]^1 ‖ [radix]^3 ‖ [rnds(n)]^1 ‖ [split(n)]^1 ‖ [n]^4 ‖ [t]^4
34  Q ← T ‖ [0]^((−t−b−1) mod 16) ‖ [i]^1 ‖ [NUM_radix(B)]^b
35  Y ← CBC-MAC_K(P ‖ Q)
36  Y ← first d + 4 bytes of (Y ‖ AES_K(Y ⊕ [1]^16) ‖ AES_K(Y ⊕ [2]^16) ‖ AES_K(Y ⊕ [3]^16) · · · )
37  y ← NUM_2(Y)

38  z ← y mod radix^m
39  return STR^m_radix(z)
```

Figure 1: **Definition of FFX[radix].** The mode enciphers strings over Chars = {0, 1, . . . , radix − 1}. It does so using a maximally-balanced Feistel network and a round function based on the AES CBC-MAC.

We do not repeat further material from the FFX spec [2] except, for the reader's convenience, we do give the definition of FFX itself, as specialized to alternating Feistel (method = 2). This makes our specification terse but self-contained.

Specification. In Figure 1 we specify FFX and the parameter collection [radix]. When FFX is instantiated with this parameter collection one obtains the scheme FFX[radix]. In Figure 2 we graphically illustrate the latter mode.

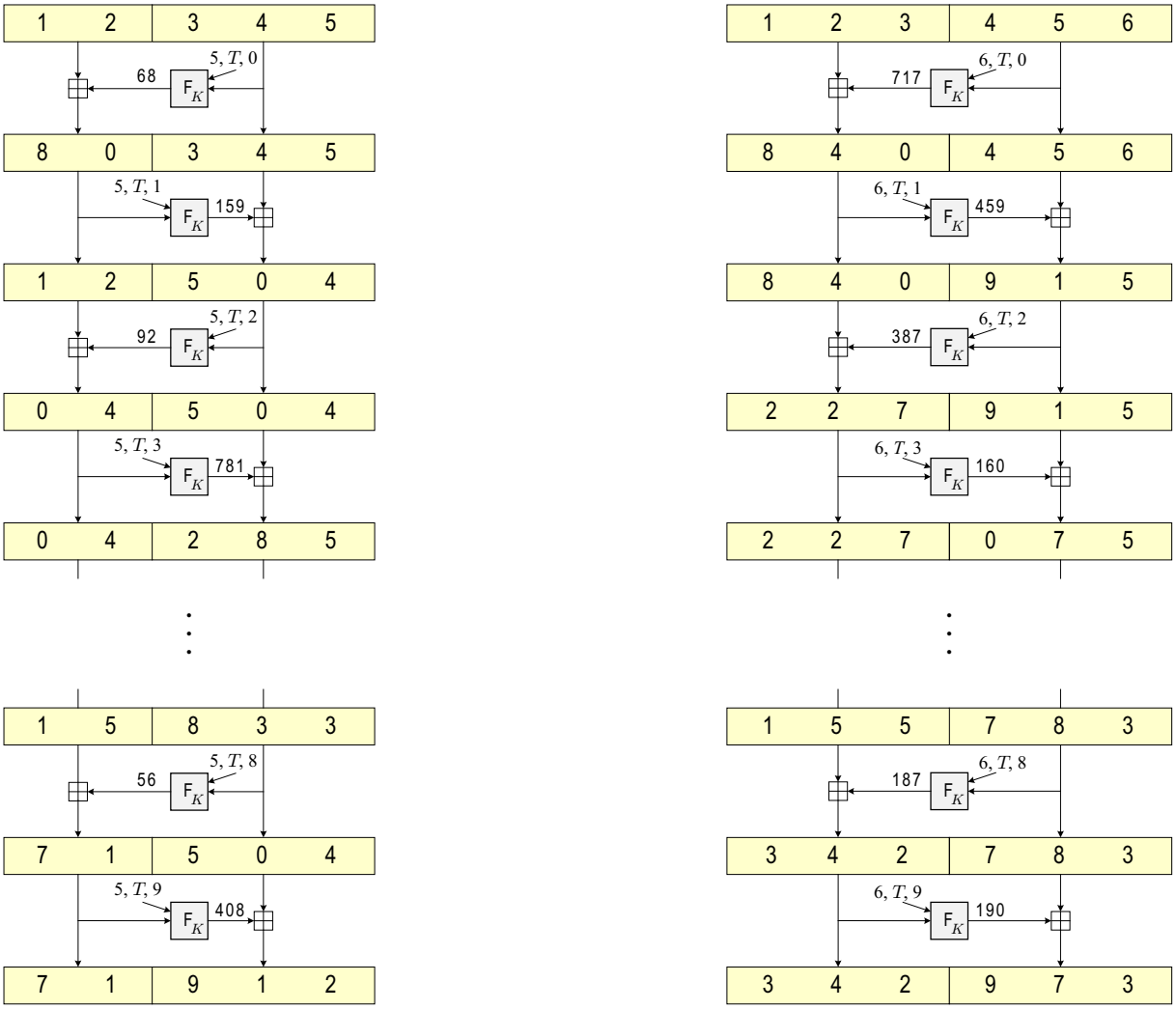

Figure 2: **Illustration of FFX[10].** The left-hand side depicts the behavior for an odd-length input (in which case alternating, almost-balanced Feistel is used); the right-hand side shows an even-length input (in which case balanced Feistel is employed). The indicated round-function outputs are fictitious; the actual values depend on the key K and tweak T, neither of which has been specified.

3 Comments

A thorough discussion of FFX is given in the spec to which that this documents is an addendum [2]; we confine ourselves here to a few quick comments.

First, we would emphasize that, while we have permitted enciphering very long strings, mode FFX[radix]—and indeed FFX in general—is intended for enciphering relatively short strings. For *binary* strings whose length exceeds 128 bits, EME2 [5] may be a better choice than FFX[radix]. For long *non*-binary strings, one can, similarly, do "better" than FFX[radix], both in terms of speed and proven-security results. However, nobody has written a spec for such a mode, one reason we have chosen to allow use of FFX[radix] even for quite long strings.

Second, we would like to briefly comment on the number of rounds, $\mathsf{rnds}(n) = 10$. Our earlier work, in parameter collections A2 and A10, had used a rather complicated function (we have come to call it 12^+) in which the number of rounds $\mathsf{rnds}(n)$ was four more than the minimal number of rounds r such that $r \cdot \mathsf{split}(n) \cdot \log_2(\mathsf{radix}) \geq 128$; but at least 12 and always a multiple of 6. The rule was heuristically motivated and extremely conservative. For $\mathsf{radix} = 10$ it translates to $\mathsf{rnds}(n) = 12$ when $n \geq 10$; $\mathsf{rnds}(n) = 18$ if $6 \leq n \leq 9$; and $\mathsf{rnds}(n) = 24$ if n is 4 or 5. The feedback we got—and our own sensibilities as well—was that a non-constant number of rounds was rather too complex. Also, it is unintuitive why one should *increase* the number of rounds as messages get *shorter*. While the explanation for these choices make sense (see the appendices in the FFX spec [2]), we have come to regard the conceptual cost as too high. In the meantime, there remains no real evidence that, once one overcomes the "meet-in-the-middle" attack, more rounds are actually needed for shorter strings. We have already noted that BPS [4] selected a round count of 8, independent of message lengths. With all these considerations in mind, we have elected to simplify and cut back the round count to 10, speeding up implementations yet leaving some margin of safety. The choice is certainly not *as* (hyper-)conservative as a round count of 12^+, but ultimately the designers of a mechanism must make a choice, based on their informed judgment and the cryptographic state-of-the-art.

Third, as discussed in the FFX spec [2], implementation choices will greatly affect performance. For example, the first AES computation implicit in line 35 need be done only once across all the rounds. When the radix is not a power of two, the mod at line 38 may require significant attention.

Finally, we re-emphasize that FFX[radix] is an instantiated version of FFX, and that other instantiations are compliant with the higher-level scheme.

References

[1] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. *Selected Areas in Cryptography* (SAC 2009), LNCS 5867, Springer, 2009. Also ePrint report 2009/251.

[2] M. Bellare, P. Rogaway, and T. Spies. The FFX mode of operation for format-preserving encryption. Draft 1.1. February 20, 2010. Manuscript, available on the NIST website at URL http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html.

[3] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. *RSA Data Security Conference, Cryptographer's Track* (RSA CT '02). LNCS vol. 2271, pp. 114–130, Springer, 2002.

[4] E. Brier, T. Peyrin, and J. Stern. BPS: a format-preserving encryption proposal. Available at URL http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html. Undated manuscript, published on the above website in April 2010.

[5] IEEE P1619.2. Draft standard architecture for wide-block encryption for shared storage media. 2008. Available from https://siswg.net.