An Extension of the FF2 FPE Scheme

Submission to NIST

Joachim Vance
VeriFone

Mihir Bellare
UCSD

July 2, 2014

Contents

1 Introduction

A Format-Preserving Encryption (FPE) scheme [1] is a cipher that preserves in the ciphertext some structure of the plaintext. It can be used to encrypt credit-card numbers in such a way that the ciphertext is also a credit-card number.

An FPE scheme takes a key K, tweak T and plaintext X and deterministically computes a ciphertext Y. The map from plaintext to ciphertext is a permutation for each choice of key and tweak. Possession of the key and tweak allows decryption, meaning X can be recovered from K, T, Y.

$\mathsf{FF2}$ is an FPE scheme in the NIST draft SP 800-38G [2]. Like the other schemes in [2], $\mathsf{FF2}$ is a Feistel construction where the round function is based on a blockcipher CIPH with a 128-bit block length. $\mathsf{FF2}$ also assumes the blockcipher has a 128-bit key. Plaintext, ciphertext and tweak are all strings over the alphabet $\Sigma = \{0, 1, \ldots, \mathsf{radix} - 1\}$ where $\mathsf{radix} \geq 2$ is an integer called the radix.

$\mathsf{FF2}$ distinguishes itself from the other schemes by its *delegation* feature. It associates to each key K and tweak T a subkey $J(K, T)$, and the ciphertext corresponding to K, T, X is a function of $J(K, T), X$ alone.

Delegation is valuable because it limits direct use of the base key. Many sidechannel attacks gain in effectiveness at recovering a key as they obtain more encryptions under it. With delegation, the loss will be limited to the subkey so that even if encryption is compromised, it is only under a particular tweak, not under all tweaks as would happen with compromise of the base key. By limiting use of the base key, delegation also extends its lifetime, so that key changes, which are cumbersome to perform, are needed less frequently.

NIST/NSA have communicated anl attack on $\mathsf{FF2}$ [3] that we will call the subkey attack. Given encryptions of a single plaintext X under different tweaks $T_1, \ldots, T_Q$, the attack returns an index i and the subkey $J(K, T_i)$, in the time for around $2^{128}/Q$ evaluations of CIPH. NIST refer to it as a theoretical attack [3]. Indeed, this attack does not appear to compromise anticipated uses of $\mathsf{FF2}$ for credit-card encryption, and would appear to be infeasible to mount in practice in anticipated use cases. However it shows that $\mathsf{FF2}$ with a 128-bit key cipher does not provide 128 bits of security for all use cases. On this basis NIST has elected to finalize SP 800-38G without $\mathsf{FF2}$ while giving VeriFone the opportunity to propose a modification.

This document describes such a modification to $\mathsf{FF2}$. It is aimed at countering the subkey attack while preserving delegatability. The modification is small, local and cheap, so that most of the structure of $\mathsf{FF2}$ is preserved.

To better explain and conceptualize the issues, schemes and choices, we take a broad perspective. We describe an FPE scheme $\mathsf{DFF}[\mathsf{OFF}]$ ("delegatable FF"), that is parameterized by an *offset function* OFF. The latter takes the base key K and tweak T to return a 128-bit binary string $\mathsf{OFF}(K, T)$. Thus DFF specifies a family of FPE schemes, one for each choice of OFF. By making a particular choice of OFF we get a particular, specific FPE scheme. $\mathsf{FF2}$ is one of these, corresponding to the trivial choice of $\mathsf{OFF}(K, T) = 0^{128}$. We propose that the amended standard be obtained as DFF with a different choice of OFF, namely one that makes the offset depend on both K and T in an unpredictable way via CIPH, specifically $\mathsf{OFF}(K, T) = \mathsf{CIPH}(K, T')$ where T' is derived from T.

DFF serves thus to unify the prior and new versions of delegatable FPE, both of which appear as special cases. Regardless of the choice of OFF, the scheme $\mathsf{DFF}[\mathsf{OFF}]$ has the delegatability feature. Susceptibility to the subkey attack is however sensitive to the choice of OFF. Making its role explicit allows us to better see how the attack might extend and make choices, such as the one suggested, which we believe thwarts the attack.

2 Notation

We let $\Sigma = \mathbb{Z}_{\mathsf{radix}} = \{0, 1, \ldots, \mathsf{radix} - 1\}$ be an alphabet. Members of Σ are referred to as digits or symbols. The size radix of Σ is referred to as the number of digits or the radix. A *string* is a finite sequence of symbols from Σ, and the set of all strings is denoted Σ^*. By $\mathsf{len}(X)$ we denote the length, meaning number of symbols, in a string X. The i-th digit of a string X will be denoted $X[i]$, for $i = 1, \ldots, \mathsf{len}(X)$, and we let $X[a \ldots b] = X[a] \ldots X[b]$ for $a \le b$. By Σ^n we denote the set of all strings of length n. Plaintexts, ciphertexts and tweak will be strings over Σ. (For simplicity we set the plaintext and tweak alphabets to be the same, although the original $\mathsf{FF2}$ specification allowed them to differ.)

If $s_1, s_2, \ldots, s_n$ are binary strings then $s_1 \| s_2 \| \ldots \| s_n$ denotes their concatenation. If $N < 2^{8i}$ is a non-negative integer then $[N]^i$ denotes its encoding as a string of $i \ge 1$ bytes. If $X \in \Sigma^*$ and $m = \mathsf{len}(X)$ then $\mathsf{NUM}_{\mathsf{radix}}(X)$ is the integer representing X, namely

$$\mathsf{NUM}_{\mathsf{radix}}(X) = \sum_{i=0}^{m-1} X[m-i] \cdot \mathsf{radix}^i .$$

By convention, $\mathsf{NUM}_{\mathsf{radix}}(X) = 0$ if $X = \varepsilon$ is the empty string, meaning the string of length 0.

If X is a binary string then $\mathsf{NUM}_2(X)$ is the integer it represents, and again $\mathsf{NUM}_2(X) = 0$ if X is the empty binary string. If $0 \le c < \mathsf{radix}^m$ is an integer then $\mathsf{STR}^m_{\mathsf{radix}}(c)$ is the string X in Σ^m that represents c, namely such that $\mathsf{NUM}_{\mathsf{radix}}(X) = c$.

Function RAD^m take input a 128-bit string S and return its representation as a string in Σ^m, computed by first converting S to an integer $I = \mathsf{NUM}_2(S)$, computing the integer I modulo radix^m to get a remainder R, and then returning $\mathsf{STR}^m_{\mathsf{radix}}(R)$.

$\mathsf{CIPH} : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ denotes an approved blockcipher such as AES.

3 The DFF FPE scheme

The $\mathsf{DFF}[\mathsf{OFF}]$ scheme is shown in Fig. 1. Parameters of the algorithm include:

- $\mathsf{radix} \in [2 \ldots 2^8]$ defining the input, output and tweak alphabet $\Sigma = \{0, \ldots, \mathsf{radix} - 1\}$
- Integer $\mathsf{minlen} \ge 2$ such that $\mathsf{radix}^{\mathsf{minlen}} \ge 100$
- Integer $\mathsf{maxlen} \ge \mathsf{minlen}$ such that $\mathsf{maxlen} \le 2\lfloor 120/\log_2(\mathsf{radix})\rfloor$ if radix is a power of two, and $\mathsf{maxlen} \le 2\lfloor 98/\log_2(\mathsf{radix})\rfloor$ if radix is not a power of two
- $\mathsf{maxTlen} < \lfloor 104/\log_2(\mathsf{radix})\rfloor$
- Function OFF that given key K and tweak T returns a string $\mathsf{OFF}(K, T) \in \{0,1\}^{128}$. Different specific FPE schemes are obtained by different choices of OFF.

```
Algorithm DFF[OFF].Enc(K, T, X)
$n \leftarrow \mathsf{len}(X)$ ; $t \leftarrow \mathsf{len}(T)$
$P \leftarrow [\mathsf{radix}]^1 \| [t]^1 \| [n]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$T' \leftarrow [0]^3 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$J \leftarrow \mathsf{CIPH}(K, P)$ ; $J' \leftarrow \mathsf{OFF}(K, T)$
$Z \leftarrow \mathsf{FEISTEL}(J, J', X)$
Return $Z$

Algorithm FEISTEL(J, J', X)
$u \leftarrow \lfloor n/2 \rfloor$ ; $v \leftarrow n - u$
$A \leftarrow X[1 \ldots u]$ ; $B \leftarrow X[u+1 \ldots n]$
For $i = 0, \ldots, 9$ do
    $Q \leftarrow [i]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(B)]^{15}$
    $Y \leftarrow \mathsf{CIPH}(J, J' \oplus Q)$ ; $y \leftarrow \mathsf{NUM}_2(Y)$
    If $i$ is even then $m \leftarrow u$ else $m \leftarrow v$
    $c \leftarrow (\mathsf{NUM}_{\mathsf{radix}}(A) + y) \bmod \mathsf{radix}^m$
    $C \leftarrow \mathsf{STR}^m_{\mathsf{radix}}(c)$
    $A \leftarrow B$ ; $B \leftarrow C$
Return $A \| B$
```

```
Algorithm DFF[OFF].Dec(K, T, Z)
$n \leftarrow \mathsf{len}(Z)$ ; $t \leftarrow \mathsf{len}(T)$
$P \leftarrow [\mathsf{radix}]^1 \| [t]^1 \| [n]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$T' \leftarrow [0]^3 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$J \leftarrow \mathsf{CIPH}(K, P)$ ; $J' \leftarrow \mathsf{OFF}(K, T)$
$X \leftarrow \mathsf{FEISTEL}^{-1}(J, J', Z)$
Return $X$

Algorithm FEISTEL^{-1}(J, J', Z)
$u \leftarrow \lfloor n/2 \rfloor$ ; $v \leftarrow n - u$
$A \leftarrow Z[1 \ldots u]$ ; $B \leftarrow Z[u+1 \ldots n]$
For $i = 9, \ldots, 0$ do
    $Q \leftarrow [i]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(B)]^{15}$
    $Y \leftarrow \mathsf{CIPH}(J, J' \oplus Q)$ ; $y \leftarrow \mathsf{NUM}_2(Y)$
    If $i$ is even then $m \leftarrow u$ else $m \leftarrow v$
    $c \leftarrow (\mathsf{NUM}_{\mathsf{radix}}(A) - y) \bmod \mathsf{radix}^m$
    $C \leftarrow \mathsf{STR}^m_{\mathsf{radix}}(c)$
    $A \leftarrow B$ ; $B \leftarrow C$
Return $A \| B$
```

Figure 1: **The $\mathsf{DFF[OFF]}$ FPE scheme.** The enciphering function is $\mathsf{DFF[OFF].Enc}$ and the deciphering function is $\mathsf{DFF[OFF].Dec}$. $\mathsf{FEISTEL}, \mathsf{FEISTEL}^{-1}$ are subroutines. The function OFF is a parameter.

	$\mathsf{OFF}(K, T)$	Remarks
OFF1	0^{128}	FF2, subject to subkey attack
OFF2	$\mathsf{CIPH}(K, T')$	Designed to resist subkey attack

Figure 2: **Some choices of OFF for $\mathsf{DFF[OFF]}$.** Here $T' = [0]^3 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$. The proposal for the standard is $\mathsf{DFF[OFF2]}$ with $\mathsf{OFF2}$ being the second choice shown.

Inputs to the encipher algorithm are:

- 128-bit key K, binary string
- Plaintext $X \in \Sigma^*$ such that $\mathsf{len}(X) \in [\mathsf{minlen} \ldots \mathsf{maxlen}]$
- Tweak $T \in \Sigma^*$ such that $\mathsf{len}(T) \in [0 \ldots \mathsf{maxTlen}]$

Inputs to the decipher algorithm are:

- 128-bit key K, binary string
- Ciphertext $Z \in \Sigma^*$ such that $\mathsf{len}(Z) \in [\mathsf{minlen} \ldots \mathsf{maxlen}]$
- Tweak $T \in \Sigma^*$ such that $\mathsf{len}(T) \in [0 \ldots \mathsf{maxTlen}]$

```
Algorithm DFF[OFF2].Enc($K, T, X$)
$n \leftarrow \mathsf{len}(X)$ ; $t \leftarrow \mathsf{len}(T)$
$P \leftarrow [\mathsf{radix}]^1 \| [t]^1 \| [n]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$T' \leftarrow [0]^3 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$J \leftarrow \mathsf{CIPH}(K, P)$ ; $J' \leftarrow \mathsf{CIPH}(K, T')$
$Z \leftarrow \mathsf{FEISTEL}(J, J', X)$
Return $Z$

Algorithm FEISTEL($J, J', X$)
$u \leftarrow \lfloor n/2 \rfloor$ ; $v \leftarrow n - u$
$A \leftarrow X[1 \ldots u]$ ; $B \leftarrow X[u+1 \ldots n]$
For $i = 0, \ldots, 9$ do
    $Q \leftarrow [i]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(B)]^{15}$
    $Y \leftarrow \mathsf{CIPH}(J, J' \oplus Q)$ ; $y \leftarrow \mathsf{NUM}_2(Y)$
    If $i$ is even then $m \leftarrow u$ else $m \leftarrow v$
    $c \leftarrow (\mathsf{NUM}_{\mathsf{radix}}(A) + y) \bmod \mathsf{radix}^m$
    $C \leftarrow \mathsf{STR}^m_{\mathsf{radix}}(c)$
    $A \leftarrow B$ ; $B \leftarrow C$
Return $A \| B$
```

```
Algorithm DFF[OFF2].Dec($K, T, Z$)
$n \leftarrow \mathsf{len}(Z)$ ; $t \leftarrow \mathsf{len}(T)$
$P \leftarrow [\mathsf{radix}]^1 \| [t]^1 \| [n]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$T' \leftarrow [0]^3 \| [\mathsf{NUM}_{\mathsf{radix}}(T)]^{13}$
$J \leftarrow \mathsf{CIPH}(K, P)$ ; $J' \leftarrow \mathsf{CIPH}(K, T')$
$X \leftarrow \mathsf{FEISTEL}^{-1}(J, J', Z)$
Return $X$

Algorithm FEISTEL$^{-1}$($J, J', Z$)
$u \leftarrow \lfloor n/2 \rfloor$ ; $v \leftarrow n - u$
$A \leftarrow Z[1 \ldots u]$ ; $B \leftarrow Z[u+1 \ldots n]$
For $i = 9, \ldots, 0$ do
    $Q \leftarrow [i]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(B)]^{15}$
    $Y \leftarrow \mathsf{CIPH}(J, J' \oplus Q)$ ; $y \leftarrow \mathsf{NUM}_2(Y)$
    If $i$ is even then $m \leftarrow u$ else $m \leftarrow v$
    $c \leftarrow (\mathsf{NUM}_{\mathsf{radix}}(A) - y) \bmod \mathsf{radix}^m$
    $C \leftarrow \mathsf{STR}^m_{\mathsf{radix}}(c)$
    $A \leftarrow B$ ; $B \leftarrow C$
Return $A \| B$
```

Figure 3: **The** DFF[OFF2] **FPE scheme** obtained by instantiating the offset function of DFF by the OFF2 function of Fig. 2. The enciphering function is DFF[OFF2].Enc and the deciphering function is DFF[OFF2].Dec. The FEISTEL, FEISTEL^{-1} subroutines are the same as in Fig. 1, shown again for completeness.

DFF is a generalization of FF2, capturing the latter as DFF[OFF] with $\mathsf{OFF}(K, T) = 0^n$. Fig. 2 shows this together with our new, suggested choice of OFF. The FF2 scheme is subject to the subkey attack of Section 4. The new choice of OFF is designed to evade the attack and the suggestion for the standard is DFF[OFF2]. The resulting FPE scheme is shown explicitly in Fig. 3 for completeness.

DFF[OFF] supports delegation, regardless of the choice of OFF. The delegated information would be $(J, \mathsf{OFF}(K, T))$. Plaintext X and this delegated information suffice to compute DFF[OFF].Enc(K, T, X), the base key K not being directly used. The delegated information can be precomputed for a given tweak T and then DFF[OFF].Enc(K, T, X) can be computed for any given X without access to K. If a terminal is encrypting with a fixed choice of tweak, it need not even store K. This limits the use of the base key, with ensuing increase in resistance to certain kinds of attacks including sidechannel attacks.

4 Potential attacks

We discuss potential attacks on DFF, including the subkey attack on some instances of DFF[OFF], including FF2, meaning the first choice of offset function from Fig. 2. We discuss how the suggested choice of offset function protects against the attack.

```
Adversary SKA($X, T_1, \ldots, T_Q, Z_1, \ldots, Z_Q$)
$i \leftarrow 0$ ; $c \leftarrow 0$
While ($i = 0$) do
    $J_c \leftarrow \mathsf{STR}_2^{128}(c)$ ; $Z \leftarrow \mathsf{FEISTEL}(J_c, X)$
    $i \leftarrow \mathsf{Find}(Z, Z_1, \ldots, Z_Q)$
    If $i > 0$ then $J \leftarrow J_c$ else $c \leftarrow c + 1$
Return $(T_i, J)$
```

Figure 4: The subkey attack on FF2.

4.1 Attack parameters and dictionary attack

The adversary is assumed to have access to an encryption oracle $\mathsf{DFF}[\mathsf{OFF}].\mathsf{Enc}(K, \cdot, \cdot)$, where K is the target key. It can invoke this oracle on any tweak T and input X of its choice to get back $Y = \mathsf{DFF}[\mathsf{OFF}].\mathsf{Enc}(K, T, X)$. We denote by $Q = 2^q$ the number of queries the adversary makes to this oracle, called encryption queries.

We regard the input length n and radix radix as fixed, these being any permissible choices. The space of possible inputs is then $\Sigma^n = \mathbb{Z}^n_{\mathsf{radix}}$. We let $R = \mathsf{radix}^n$ denote its size, meaning the number of possible inputs. We let $\{I_1, \ldots, I_R\}$ denote a listing of all possible inputs.

Any FPE scheme is subject to an unavoidable dictionary attack which breaks the scheme in R encryption queries and almost no offline work. In the case of DFF, it works as follows. The adversary picks any tweak T of its choice and queries $Y_i = \mathsf{DFF}[\mathsf{OFF}].\mathsf{Enc}(K, T, I_i)$ for all $i = 1, \ldots, R$. Having the table $(Y_1, \ldots, Y_R)$, it can subsequently decrypt any ciphertext Y encrypted under tweak T, by simply finding i such that $Y = Y_i$ and returning I_i as the decryption.

4.2 The subkey attack

Recall that $\mathsf{FF2} = \mathsf{DFF}[\mathsf{OFF}]$ for $\mathsf{OFF}(K, T) = 0^{128}$. We describe the NIST/NSA attack on $\mathsf{FF2}$ from [3]. We call it the subkey attack. We then discuss the effectiveness of the attack. Below we denote by $\mathsf{FF2.Enc}$ and $\mathsf{FF2.Dec}$ the encryption and decryption algorithms of $\mathsf{FF2}$, respectively.

The adversary picks a plaintext X and distinct non-empty tweaks $T_1, \ldots, T_Q$. It then uses its encryption oracle to obtain encipherings $Z_i = \mathsf{FF2.Enc}(K, T_i, X)$ for $i = 1, \ldots, Q$. Here $Q = 2^q$, the number of encryption queries, is the number of tweak-ciphertext pairs that the adversary has, and is a parameter of the attack.

The attack, denoted SKA, is shown in Fig. 4. Here $\mathsf{STR}_2^{128}(c)$ is the 128-bit binary string corresponding to integer c. Algorithm Find, on input $Z, Z_1, \ldots, Z_Q$, searches for Z in the list $Z_1, \ldots, Z_Q$, returning some value i such that $Z = Z_i$ if such a value exists, and returning 0 otherwise.

To explain the attack, first let

$$P_i = [\mathsf{radix}]^1 \| [\mathsf{len}(T_i)]^1 \| [\mathsf{len}(X)]^1 \| [\mathsf{NUM}_{\mathsf{radix}}(T_i)]^{13} \qquad \text{and} \qquad J(T_i) = \mathsf{CIPH}(K, P_i)$$

for $i = 1, \ldots, Q$. We call $J(T_1), \ldots, J(T_Q)$ the target subkeys. Knowledge of $J(T_i)$ allows decryption of any ciphertext enciphered under tweak T_i. Indeed, if $Z' = \mathsf{FF2.Enc}(K, T_i, X')$ is encrypted under

T_i, and one knows $J(T_i)$, one can recover $X' = \mathsf{FEISTEL}^{-1}(J(T_i), Z')$. However, possession of $J(T_i)$ does not allow decryption of ciphertexts enciphered under a tweak different from T_i, meaning $J(T_i)$ is of limited use to an attacker.

The attack will try to find $J(T_i)$ for some i. The attacker will not be able to dictate the value of i for which it is successful. This will, rather, emerge from the attack, effectively being a random value in the range $1, \ldots, Q$. The attack searches through the key space for subkeys, testing each candidate subkey J_c by seeing if the result Z of FEISTEL on X with the candidate subkey J_c matches some ciphertext in its given list $Z_1, \ldots, Z_Q$. If so, meaning $Z = Z_i$, then it is likely that $J_c = J(T_i)$. The attack returns T_i and $J = J_c$.

4.3 Cost of the subkey attack

Since there are $Q = 2^q$ target subkeys, we expect that the loop will be successful after $2^{128}/Q = 2^{128-q}$ iterations. Each iteration costs 10 CIPH computations plus the time to run Find. The latter can be made much less than Q steps by using data structures such as hash tables or binary search, and we will accordingly neglect this cost entirely. (This is perhaps giving the attacker too much credit, for merely the memory management is likely to have some cost for large Q.) The result is a cost estimate of $10 \cdot 2^{128-q} \geq 2^{131-q}$ CIPH operations. The memory cost is $\mathcal{O}(Q)$.

NIST refer to the subkey attack as a theoretical one. Cryptographers generally consider an effort of 2^{80} to be prohibitive, and a system is considered secure if the adversary effort is estimated to be of this magnitude. This is the effort corresponding to the current parameter choices for pubic-key cryptography, namely 1024 bit RSA keys or 160-bit EC keys. If we adopt the same metric, then the subkey attack can be considered ineffective in practice as long as $2^{131-q} \geq 2^{80}$. In other words, for the attack to be effective, it must be that $q \geq 131 - 80 = 51$. That is, the adversary would need to collect at least $Q = 2^{51} \approx 2.251$ quadrillion tweak-ciphertext pairs, *all under one, same plaintext.* Obtaining this is not easy, particularly in the context of payment systems. It requires long-term access to an encryption device. For example Verifone devices allow at most 1,000 encryptions per hour, so obtaining even 2^{37} of them would take over 100,000 years. We also note that the attack does not recover the key K, but only allows decryption under some tweak over which the attacker has little control.

For the attack to be non-trivial, meaning better than the unavoidable dictionary attack discussed above, we need $Q \leq R$, which means that the attack is effective only when the input space is large, the opposite of the typical application setting for FPE. Also, the tweak space needs to be large.

NIST's position is that when using an underlying blockcipher CIPH with a 128-bit key, one should obtain 128 bits of security for the FPE scheme for all uses, a goal that FF2 does not meet. DFF[OFF] is here suggested as an extension that, for appropriate choices of OFF, is aimed at countering the attack while preserving delegatability.

4.4 The suggested offset choice and scheme

The subkey attack on FF2 crucially exploits the fact that the FEISTEL computation is not tweak dependent. To prevent the subkey attack or variants, we allow the Feistel round function to depend on the tweak via the offset. Specifically we consider DFF[OFF] with $\mathsf{OFF}(K, T) = \mathsf{CIPH}(K, T')$, the

2nd choice from Fig. 2. We believe this will circumvent the subkey attack, restoring the algorithm to a 128-bit strength for all uses, and making it suitable for inclusion in SP 800-38G. The reason is that the the Feistel computation must be repeated for each candidate tweak in the attack.

References

[1] M. Bellare, T. Ristenpart, P. Rogaway and T. Stegers. Format Preserving Encryption. Proceedings of SAC 2009, Springer LNCS Vol. 5867, 2009. IACR Cryptology ePrint Archive Report 2009/251.

[2] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. NIST Special Publication 800-38G. Draft, July 2013.

[3] NIST Computer Security Division News, Explanation of Changes to Draft Special Publication 800-38G, June 27, 2014. `csrc.nist.gov/news_events/`