ERICSSON

Date: April 4, 2025

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on NIST 800-38G
Methods for Format-Preserving Encryption

Dear NIST,

Thanks for your continuous efforts to produce well-written, user-friendly, and open-access security documents. Format-Preserving Encryption (FPE) has emerged as a very useful cryptographic tool for encryption of fields in legacy databases, protocols, and formats, that could otherwise not be encrypted without complex and costly changes.

Since FF2 and FF3 have been broken, NIST now approves only a single Format-Preserving Encryption (FPE) algorithm, limited to 128-bit block ciphers. We think NIST should initiate a "competition" to foster research in FPE and identify new algorithms suitable for standardization.

While NIST currently has an open call for block cipher modes [1], we propose a broader approach that welcomes FPE designs based on AES, Rijndael-256, the AES round function, Keccak, Ascon, stream ciphers, and other primitives. Instead of a resource-intensive competition, NIST could issue an open call for FPE proposals, allowing submissions to be evaluated for potential standardization.

Please find below Ericsson's detailed comments on NIST 800-38G Rev. 1 (2nd Public Draft):

- We agree with the main technical changes to the original publication [2].

- The Acknowledgments and Purpose sections contain historical information that would be more appropriately placed in an appendix.

- We think the introduction should discuss tokenization as an alternative or complement to FPE, highlighting their distinct deployment characteristics and security properties. Tokenization can be implemented using NIST-approved RNGs or PRFs. In use cases where FPE is used with a fixed tweak or a tweak with low entropy, tokenization with frequently refreshed tokens can offer enhanced privacy.

- *"The previously approved modes for encryption are transformations on binary data"*

  For binary data, it would be helpful to clarify whether FPE offers any security benefits or disadvantages compared to previously approved modes.

- It would be beneficial for most readers if the introduction mentioned the performance differences between FF1 and previously approved modes like AES-CTR and AES-GCM. Our understanding is that FF1 is typically several orders of magnitude slower due to its Feistel-based structure, whereas AES-CTR and AES-GCM benefit from fewer AES invocations and parallelization. Highlighting this contrast upfront would provide useful context for readers.

- *"The choice and implementation of a one-to-one correspondence between a given alphabet and the set of base radix numerals that represents the alphabet is outside of the scope of this publication."*

  It would be good to clarify that a complex one-to-one mapping between alphabet and radix numerals does not enhance security. Additionally, the encoding shall not depend on the key K. The alphabet and radix are typically also configurable choices. For example, when encrypting the middle six digits of a credit card number (e.g., 123456), the user can choose from 4 different radixes (10, 10^2, 10^3, 10^6). It would be helpful to clarify whether these choices have any impact on security.

- *"the encryption function is denoted by FF1.Encrypt(K, T, X), and the decryption function is denoted by FF1.Decrypt(K, T, X)"*

  What is the rationale behind this notation? Why does Decrypt() not take Y as input? Additionally, why aren't X and Y not just called P and C?

- *"the "tweak," which is not necessarily secret."*

  "The tweak does not need to be kept secret."

  "Usually, tweaks are non-secret information"

  It would be good to explain to the reader if there are any security benefits with secret tweaks similar to secret nonces [3–4].

- *"Ideally, the non-secret tweak associated with a plaintext is associated only with that plaintext"*

  Our understanding is that this also applies to secret tweaks. We suggest removing "non-secret."

- *"Although implementations **may** fix the value of the tweak, variable tweaks **should** be used as a security enhancement (see Appendix C)."*

  *"The tweak T is **optional** in that it **may** be the empty string with byte length t = 0."*

  *"implementations **should** enforce the use of different tweaks for different instances where feasible"*

  *"the tweak **should** vary with each instance of the encryption whenever possible"*

*"In general, if there is information available and statically associated with a plaintext, it is **recommended** to use that information as a tweak"*

The terms "may", "optional", "should", and "recommended" appear undefined in the document and should be clarified. In many SDOs, such as IETF [5] and 3GPP, something that is "should" or "recommended" cannot simultaneously be "may" or "optional." Based on this and security considerations, we suggest classifying fixed or empty tweaks as "should not" or "not recommended". The security of FF1 with a fixed tweak is quite low, as it always encrypts the same plaintext to the same ciphertext. This makes it vulnerable to pattern analysis and likely insufficient to meet requirements such as GDPR's 'pseudonymization' standard. We think this should be clarified.

- *"However, if the leading six digits and the trailing four digits of the CCN had been used as the tweak, then the corresponding plaintexts would almost certainly be different."*

  This would be correct if the digits were uniformly distributed. However, credit card numbers are not random, the first six or eight digits represent the issuer identification number, and the last digit is a check digit. Given this structure, we believe "almost certainly be different" is too strong.

  Using a tweak that is correlated to the plaintext provides very low security when the same key is reused, as it always produces the same ciphertext for identical plaintexts. This makes the encryption highly vulnerable to pattern analysis. We believe this should be clarified. The document should recommend using a variable tweak independent of the plaintext whenever possible.

- The document should explain the recommendations and implications of different tweak constructions and lengths. For example, what are the security implications of using random tweaks versus deterministic tweaks, such as a counter? For nonces, the impact can be significant [6].

- *"If changes to the length of the data field are supported by the application, the data can be padded with a fixed sequence of symbols, such as adding redundant zeros to the beginning of a credit card number (CCN). Padding can provide resistance against ciphertext guessing attacks, as modifications of the FPE-encrypted data will be correctly padded upon decryption only with some probability, depending on the length of the padding. Additionally, padding the data field can hide the length of the data to mitigate plaintext guessing attacks (see Appendix A.1)."*

  Our understanding is that this type of encode-then-encrypt padding [7–8] would provide strong integrity if implemented correctly and that the forgery probability is well understood, assuming that the FPE is secure. However, we think the details of padding and integrity should not be left to the application. If the document allows encode-then-encrypt padding for integrity, it should provide clear specifications rather than briefly mentioning it in the introduction. Furthermore, if data field length changes are supported, encryption with AES-CCM or AES-GCM becomes feasible by encoding the ciphertext as a radix string. The document should explain that the FF1 algorithm without padding does not provide any integrity and explain what the implications of an attacker modifying the ciphertext have.

- *"FF1 has two parameters — minlen and maxlen — that determine the lengths for the numeral strings that are supported by an implementation of the encryption or decryption function for the mode. FF1 also has a parameter — maxTlen — that indicates the maximum supported length of a tweak string. The selection of these parameters may also affect interoperability."*
  "maxTlen should be chosen to accommodate the desired tweaks for the implementation."

  It is unclear why the parameters *maxlen* and *maxTlen* in NIST SP 800-38G are implementation-defined prerequisites, whereas the maximum values of *len(P)*, *len(A)*, and *len(IV)* in NIST SP 800-38D are explicitly specified in the publication. While the documentation states that *maxlen* must be less than 2^{32}, it does not define a maximum value for *maxTlen*. We think the specification should provide more guidance to readers and developers by defining an explicit maximum or default value for *maxTlen*. Additionally, it is likely unclear to most readers whether, for example, *maxTlen* = 2^{64} is an acceptable choice and whether different values of *maxTlen* have security implications.

John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols, Ericsson

References

[1] Block Cipher Techniques Modes Development
https://csrc.nist.gov/projects/block-cipher-techniques/bcm/modes-development

[2] NIST SP 800-38G Rev. 1 (2nd Public Draft) Announcement
https://csrc.nist.gov/pubs/sp/800/38/g/r1/2pd

[3] The Multi-User Security of Authenticated Encryption: AES-GCM in TLS 1.3
https://eprint.iacr.org/2016/564.pdf

[4] Nonces are Noticed: AEAD Revisited
https://eprint.iacr.org/2019/624.pdf

[5] Key words for use in RFCs to Indicate Requirement Levels
https://www.rfc-editor.org/rfc/rfc2119.html

[6] Ericsson comments on 800-38D
https://csrc.nist.gov/files/pubs/sp/800/38/d/r1/upd/iprd/docs/sp800-38d-pre-draft-public-comments.pdf

[7] Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography
https://link.springer.com/chapter/10.1007/3-540-44448-3_24

[8] Proposal of Requirements for an Accordion Mode
https://csrc.nist.gov/files/pubs/other/2024/04/10/proposal-of-requirements-for-an-accordion-mode-dis/iprd/docs/proposal-of-requirements-for-an-accordion-mode-discussion-draft.pdf