Comments on NIST's Requirements for an Accordion Cipher

John Preuß Mattsson, Ben Smeets, Erik Thormarker
Ericsson

Accordion – Encryption for the next 50 years

- Well-designed accordions and derived functions will likely be extremely useful in many future use cases.

- The accordion ciphers and their derived functions should have:
  - very strong cryptographic properties
  - good usability and usable security
  - cryptographic agility
  - good performance

- We agree with NIST about tweakable VIL-SPRP, derived functions, and collaborative approach.

- Design for the next 50 years, not for the most limiting AES APIs.

Accordion – Underlying primitives

- We are not convinced by the suggestion to restrict the underlying primitives to only block ciphers or AES.
  - NIST has not provided any motivation for this.
- Suggested outcome of the 2023 workshop was to design accordions based on Rijndael-256-256 and TurboSHAKE.
  - We agree with this suggestion. Rijndael-256-256 gives high security. TurboSHAKE gives cryptographic agility.
- AEZ, AEGIS, and Rocca-S are based on the AES round function to significantly increase performance.
  - Performance has historically been very important for large-scale adoption.
- 3DES was disallowed as the block size is only 57% of the claimed security strength. AES-256 has the same problem.
  - For MILENAGE-256, AES-256 was a dead end that increased complexity, decreased performance, and/or limited security.

Accordion inputs

Key size.

— Fine with only 256-bit keys. Only reason to support 128-bit keys would be compatibility with AES-128.

Plaintext length.

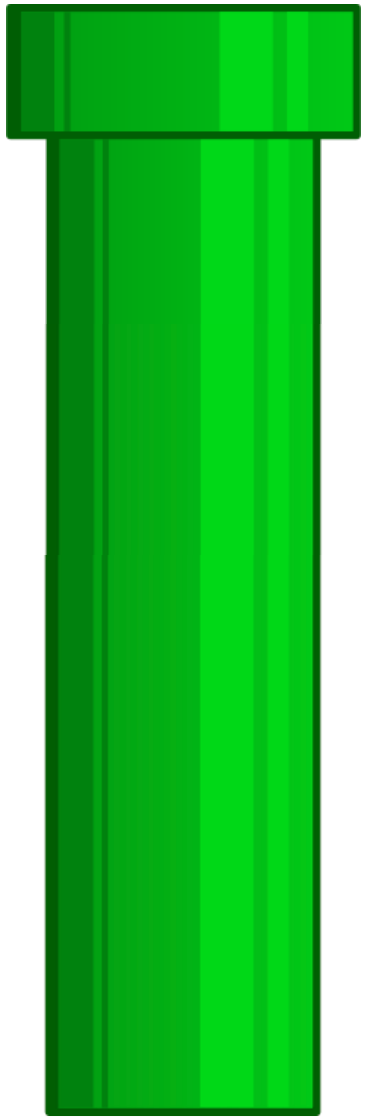
— We think it should be a strong requirement that $g \leq 8$ to support all byte strings within a certain range. A cipher with $g = 128$ would limit the number of use cases and require padding. For SW it might be beneficial with $g = 8$.

— A limit such as ag = 16 or 32 bytes is acceptable. It would be beneficial if the accordion supports plaintexts of length significantly greater than 2^{36}, e.g., bg = 2^{48} or 2^{64} bytes.

Tweak size.

— We agree with NIST that the accordion cipher should support long variable-length tweaks. This eliminates hashing in the derived functions.

— To securely handle encryption with random nonces and to protect against TMTO attacks, the tweak size needs to be 256 bits. The security (offline) of AES-256-GCM with r cleartext random nonces is only 97 - $\log_2 r$ 😱. Nonce hiding (AE2) increases the attack complexity for attacks like this and misuse resistance (MRAE) lowers the damage.



Strong security properties of derived functions

- We would like to see a Robust Authenticated Encryption (RAE), with security against release of unverified plaintext (RUP), nonce hiding, succinct commitment (AE3), and much better bounds than $\sigma^2 / 2^{129}$.
  - Many keys, many invocations per key, and long plaintexts.
- Robust Authenticated Encryption (RAE) is a good starting point, it includes misuse resistance (MRAE), reforgeability resilience, and concealment of plaintext length.
- Variable ciphertext expansion λ is required for RAE, concealment of plaintext length, and succinct commitment.
- We don't think the term "beyond birthday-bound security" should be used. The goal should be to have a high concrete security level, and the easiest way to achieve this is to use a block cipher with a 256-bit block length. A confidentiality advantage of $\sigma_r^2 / 2^{257}$, where σ_r is the number of encrypted 256-bit blocks, is a good solution but is not "beyond birthday-bound security".

Hedged key wrap instead of DAE

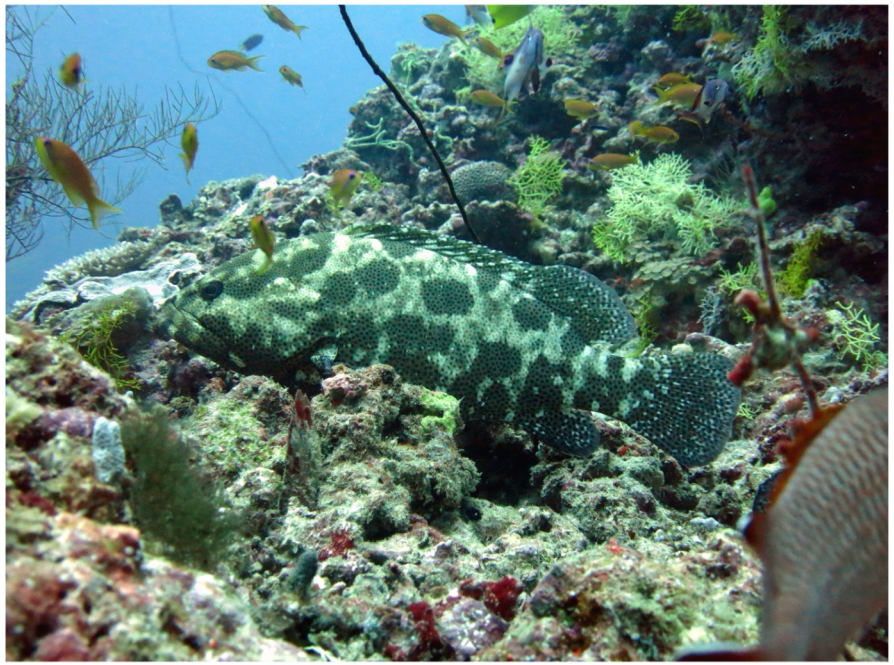
- The DAE solution NIST proposes for key wrap inherits issues from AES-KWP
  1. Equality of Plaintexts
  2. Implied Strength of Protected Keys
- An RAE is an MRAE with flexible ciphertext expansion λ and can solve both issues. Can also be implemented with a nonce-hiding MRAE.

  Hedged-KW(K, P, λ):

  1. Let N be a random nonce. Let A be the empty string.
  2. C' = RAE(K, N, P, A, λ)
  3. Return C = N || C'
- Even with a bad random number generator (RNG), a hedged key wrap gives IND-CCA2 security with DAE security as a worst case. The randomness likely increases security against side-channel attacks and fault attacks.
- While not basing security on a strong RNG is a requirement to key wrap, we don't think determinism is a requirement; it just happened to be a feature of the solutions.
- We think NIST should specify a hedged key wrap with with flexible ciphertext expansion instead of a DAE.

Derived functions with nonce hiding

- Cleartext nonces can enable tracking and identification of sender and receiver, reveal information to an attacker who backdoored the PRNG, and significantly speed up offline collision attacks on random nonces.
- Nonce hiding has seen broad adoption in modern security protocols such as TLS 1.3, DTLS 1.3, and QUIC.
  - The nonce typically consists of an implicit part and an explicit part (the sequence number).
- A nonce-hiding encryption API would need to only hide the explicit part of the nonce (the sequence number). The implicit part of the nonce can likely be associated data / tweak.
- The decryption API would need to output the sequence number, as that is used for replay protection and message ordering.
- A standardized nonce hiding AE would be welcome as it simplifies sequence number encryption in both security protocols and when the AE is used on its own.
- Fully Encrypted Protocols (FEP) are important to avoid network censorship and for military communication systems. A nonce hiding AEAD enables implementation of a FEP with less overhead.
- We think NIST should specify derived functions with nonce hiding.



Derived functions with replay protection

- An alternative to just nonce hiding that improves usable security are constructions such as Authenticated Encryption with Replay prOtection (AERO) which use a tweakable VIL-SPRP to provide both nonce hiding and replay protection. Basic idea is to encrypt the plaintext concatenated with the sequence number.

  E = EncryptionInit(K)
  D = DecryptionInit(K)
  C = E(P, A , λ)
  P = D(C, A)

- An AERO API offers several advantages over other methods: it is simpler to use, encourages the use of replay protection, and has more compact messages. Optimal stand-alone encryption envelope.
- We have in the past seen a lot of practical and serious vulnerabilities due to weak or missing replay protection. Higher-layer protocols are very often designed based on the assumption of strong replay protection in lower layers. Missing or weak replay protection in the lower layer typically compromises confidentiality, integrity, and availability at higher layers, i.e., the whole CIA triad. Replay protection should be a mandatory requirement in all modern security protocols.
- We think NIST should standardize an AERO derived function. Users should ideally not have to deal with nonces or replay protection.

Summary

— Well-designed accordions and derived functions will likely be extremely useful in the future improving both cryptographic security properties and usable security.

— AES-256 is likely not a good building block. Rijndael-256-256, TurboSHAKE, and the AES round function all seem more promising.

— NIST should standardize Rijndael-256-256 and TurboSHAKE.

— The accordion standardization project should analyze (and standardize) several accordions with a common set of derived functions.

— NIST should forbid the use of of GCM and CCM with random nonces in the ongoing revisions of SP 800-38C/D.

https://www.ericsson.com/en/security