Update on the Lightweight Cryptography Project

AEAD Security Requirements (2/2)

- Family of algorithms
  - One primary member with key, nonce and tag lengths of 128, 96 and 64 bits, respectively.
  - Limits on the input sizes for the primary member **shall not** be smaller than $2^{50} - 1$.
  - Family can include at most 10 members.
- Keys sizes **shall** at least be 128 bits. Attacks **shall** require at least 2^{112} computations. If larger key sizes are supported, it is recommended that at least one member has key size of 256 bits (and resistance to attacks **shall** at least be 2^{224} computations).
- Well-understood, and analyzed. Submissions are expected to have third-party analysis.

Hash Function Requirements

A hash function is a function with one byte-string input and one byte-string output. The input is a variable-length message. The output is a fixed-length hash value.

- Computationally infeasible to find a collision or a (second) preimage. Resistance to length extension attacks.
- Cryptanalytic attacks on the hash function **shall** require at least 2^{112} computations on a classical computer.
- The hash function **shall not** specify hash values that are smaller than 256 bits.
- Family of algorithms
  - One primary member has a hash size of 256 bits.
  - Limits on the input sizes for the primary member **shall not** be smaller than $2^{50} - 1$.
  - Family can include at most 10 members.

Additional Requirements for Submissions with AEAD and Hashing

- Submissions **shall** state which design components the AEAD and hashing algorithms have in common, and explain how these common components lead to a reduced implementation cost.
- Submissions **shall** specify list of pairs of AEAD and hash function family members to be evaluated jointly. This list is permitted to be as short as one recommendation. Primary member of the AEAD family and primary member of the hash function family **shall** be paired together. This list **shall not** be longer than ten recommendations.

Design Requirements

- Submissions **shall** perform significantly better in constrained environments (HW and SW platforms) compared to NIST standards.
- Optimized to be efficient for short messages.
- Implementations should lend themselves to countermeasures against various side channel attacks.
- Designs can make tradeoffs between performance metrics, and submitters are allowed to prioritize certain performance requirements over others.

Implementation Requirements

- Reference software implementation in C, to support public understanding.
- An implementation **shall** be provided for all variants.
- Code **shall not** contain compiler intrinsics, platform-specific headers, or compiler-specific features.
- API is compatible with eBACS: ECRYPT Benchmarking of Cryptographic Systems.
- Submission may include optimized implementations that use the same API, or additional implementations that highlight specific implementation features of the algorithms. There are no restrictions on the API for the additional implementations.
- The correctness of the reference implementation will be verified on the NIST test vector verification platform.

Evaluation Process and Tentative Timeline

- Submissions will be analyzes based on security, performance and also side channel resistance.
- Submissions that have significant third-party analysis or leverage components of existing standards will be favored for selection.

Tentative timeline

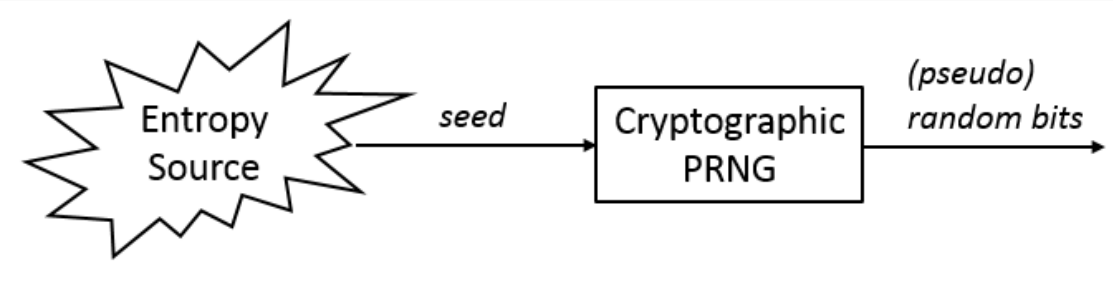
- March 2018, publish draft call for submission.
- June 2018, publish final call for submision.
- December 2018, deadline for submission (tight deadline).
- NIST will publish the complete and proper submissions.
- Initial evaluation 12 months.
- Workshop will be held ten to twelve months after the submission deadline.
- Standardization within two to four years, after the public analysis starts.

Some open research questions

Open Research Related to SP 800 90 Series

NIST SP 800 90 Series - Recommendations on Random Number Generation

- 90A - Deterministic RNGs
- 90B - Entropy Sources
- 90C - RNG Constructions



Entropy Estimation - 90B Perspective

- 90B aims to estimate entropy of noise/entropy source outputs.
- Black box analysis, based on different statistical assumptions, and minimum estimate is awarded.
  - Black box analysis is necessary for lab evaluation. Although it is not the optimal strategy, it improves the quality of the RNGs.
- We propsoed the predictors framework to estimate entropy [2].

Open issues include

- Designing noise source specific predictors, Simulating and modeling TRNG outputs, using multiple noise sources.

[2] Kelsey J., McKay K.A. and Sonmez Turan, M., Predictive Models for Min-Entropy Estimation, Workshop on Cryptographic Hardware and Embedded Systems, CHES 2015, Saint Malo, France, 2015

Circuit Complexity

Given a set of gate types, construct a circuit that computes f and is optimal to some criteria

Multiplicative Complexity is the minimum number of AND gates that are sufficient to evaluate the function over the basis (AND, XOR, NOT).

Why do we count the AND gates?

- Lightweight Cryptography: The technique of *minimizing the number of AND gates, and then optimizing the linear components* leads to the implementations with low gate complexity.
- Secure multi-party computation: Reducing the number of AND gates improves the efficiency of SMP protocols.
- Side channel attacks: Minimizing the number of AND gates is necessary when implementing a masking scheme to prevent side-channel attacks.

Multiplicative Complexity is affine invariant!

Affine Equivalance

An affine transformation from g to f in B_n is a mapping of the form

$$f(x) = g(Ax + a) + b \cdot x + c,$$

where A is a non-singular $n \times n$ matrix over $\mathbb{F}_2$; x, a are column vectors over $\mathbb{F}_2$; b is a row vector over $\mathbb{F}_2$.

- f, g are affine equivalent, if and only if there exists an affine transformation between them.
- Affine equivalent functions are said to be in the same equivalence class.
- All functions in an equivalence class have the same multiplicative complexity, i.e., multiplicative complexity is affine invariant.

Multiplicative Complexity of Boolean functions $n \leq 6$

Method

Exhaustively construct all Boolean circuit topologies with 1,2, 3, ... AND gates, and mark the Boolean functions that can be generated by the circuits until a function from each equivalence class is generated.

Multiplicative complexity is

- ≤ 3, for $n = 4$ (8 equivalence classes)
- ≤ 4, for $n = 5$ (48 equivalence classes)[3]
- ≤ 6, for $n = 6$ (150,357 equivalence classes)[4]

[3] M. Sonmez Turan, R. Peralta, *The Multiplicative Complexity of Boolean Functions on Four and Five Variables*, https://eprint.iacr.org/2015/848

[4] C. Calik and M. Sonmez Turan and R. Peralta, *The Multiplicative Complexity of 6-variable Boolean Functions*, https://eprint.iacr.org/2018/002

Multiplicative Complexity of Boolean functions $n \leq 6$

Method

Exhaustively construct all Boolean circuit topologies with 1,2, 3, ... AND gates, and mark the Boolean functions that can be generated by the circuits until a function from each equivalence class is generated.

Multiplicative complexity is

- ≤ 3, for $n = 4$ (8 equivalence classes)
- ≤ 4, for $n = 5$ (48 equivalence classes)[3]
- ≤ 6, for $n = 6$ (150,357 equivalence classes)[4]

How about for larger n? Vectorial Boolean functions? Multiplicative Complexity of AES S-box?

[3] M. Sonmez Turan, R. Peralta, *The Multiplicative Complexity of Boolean Functions on Four and Five Variables*, https://eprint.iacr.org/2015/848

[4] C. Calik and M. Sonmez Turan and R. Peralta, *The Multiplicative Complexity of 6-variable Boolean Functions*, https://eprint.iacr.org/2018/002

Thanks!

More information on Lightweight Cryptography Project available at
`https://www.nist.gov/programs-projects/lightweight-cryptography`

Subscribe to our mailing list: `lwc-forum@list.nist.gov`

Additional comments/questions?

Email the team at `lightweight-crypto@nist.gov`