PAEQ: Parallelizable Permutation-based Authenticated Encryption

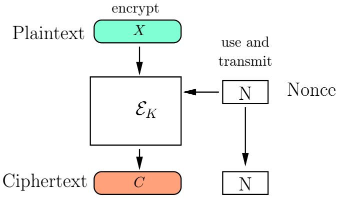
Alex Biryukov and Dmitry Khovratovich

University of Luxembourg

12 October 2014

Authenticated encryption

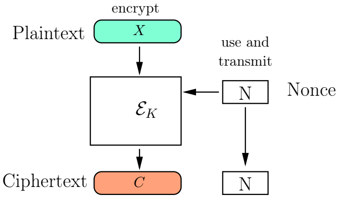
If you just want to protect confidentiality of your data, you use (simple) symmetric encryption:



- Agree on the key $\mathcal{K}$;
- Choose nonce N uniquely for each piece of data;
- Encrypt and send.

Good encryption scheme makes ciphertexts look random (even if plaintexts repeat).

If you just want to protect confidentiality of your data, you use (simple) symmetric encryption:

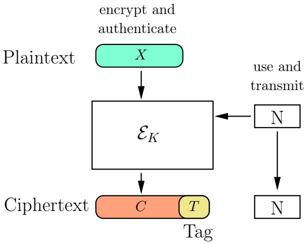


- Agree on the key $\mathcal{K}$;
- Choose nonce N uniquely for each piece of data;
- Encrypt and send.

Good encryption scheme makes ciphertexts look random (even if plaintexts repeat).

No integrity protection.

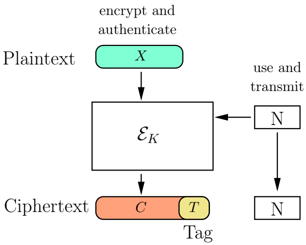
If you also want to protect integrity of your data (i.e. authenticate the message), you use *authenticated encryption*:



- Tag T is added to each ciphertext;
- Adversary can not modify $C||T$ without getting noticed.

Good encryption scheme should decrypt forged ciphertext to $\perp$ (invalid).

If you also want to protect integrity of your data (i.e. authenticate the message), you use *authenticated encryption*:

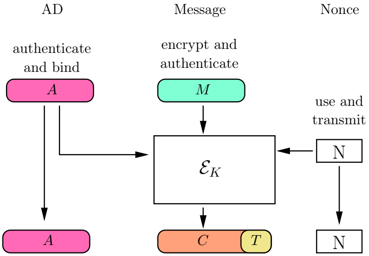


- Tag T is added to each ciphertext;
- Adversary can not modify $C||T$ without getting noticed.

Good encryption scheme should decrypt forged ciphertext to $\perp$ (invalid).

We might also want to authenticate some data without encrypting it (associated data).

Authenticated encryption with associated data



Confidentiality:

- Ciphertexts indistinguishable from random strings;

Data integrity:

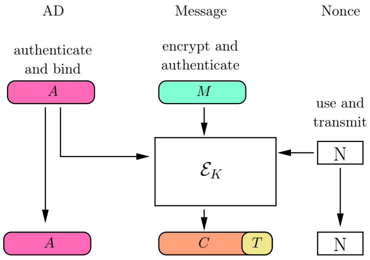
- Most of seemingly valid ciphertexts decrypt to $\perp$.

Desirable features

Non-exhaustive list of authenticated encryption features:

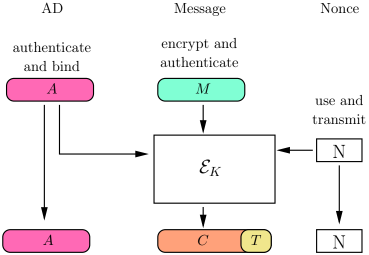
- Parallelizability to fully use multi-core CPU;
- Incremental tags to avoid recomputing the entire ciphertext;
- Security proof;
- Reasonable performance;
- Compact implementation.

What we also want



Some extra features:

- Easy to understand and implement.
- Security level equal to the key length (does not hold for AES-CBC/GCM/OCB).
- More compact and verifiable security proofs.
- No extra operations like key derivation, field multiplications etc. (makes the design more complex).



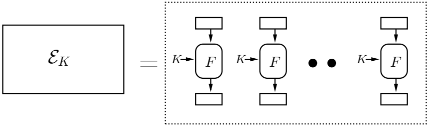
Some extra features:

- Easy to understand and implement.
- Security level equal to the key length (does not hold for AES-CBC/GCM/OCB).
- More compact and verifiable security proofs.
- No extra operations like key derivation, field multiplications etc. (makes the design more complex).

Solution: design a permutation-based mode, not a blockcipher one.

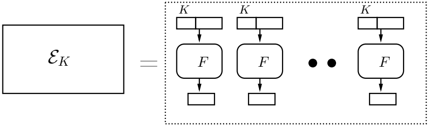
Permutation-based

How to construct a variable-length cipher:



- Each component is keyed function F_K;
- Security reduces to pseudorandomness of F (unpredictable under a random key).

How to construct a variable-length cipher:



- Each component is a fixed public function F;
- Security proven if F is randomly chosen (while in fact it is not).

Why permutation-based?

- A wide permutation can take key, nonce, counter, intermediate values, or a message block altogether as input.
- Plenty of designs: different widths and optimizations;
- The underlying permutation is easier to design and analyze (no need to care of key schedule, mask generation, nonce formatting, etc.).

Why permutation-based?

- A wide permutation can take key, nonce, counter, intermediate values, or a message block altogether as input.
- Plenty of designs: different widths and optimizations;
- The underlying permutation is easier to design and analyze (no need to care of key schedule, mask generation, nonce formatting, etc.).

Cons:

- Weaker security model (random permutation);
- Lower throughput (larger calls/byte ratio).

80- and 128-bit security

Most popular modes suggest using AES (128-bit block) as the underlying blockcipher.

Most popular modes suggest using AES (128-bit block) as the underlying blockcipher.

No security guaranteed as the number of invocations q approaches $2^{n/2} = 2^{64}$.

Beyond 64-bit security

Most popular modes suggest using AES (128-bit block) as the underlying blockcipher.

No security guaranteed as the number of invocations q approaches $2^{n/2} = 2^{64}$.

We want to offer a higher security margin.

PAEQ

Our new scheme **PAEQ** has

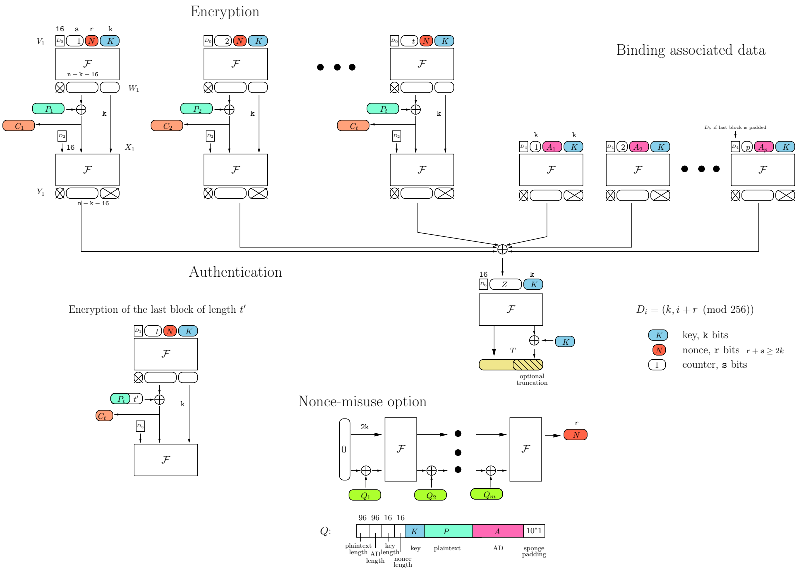
Basic features:

- Fully parallelizable;
- Handles associated data;
- Variable key/nonce/tag length;
- Patent-free;
- Online encryption and authentication, no length awareness;
- Byte-oriented.
- Incremental tag (for max tag length).

Extra features:

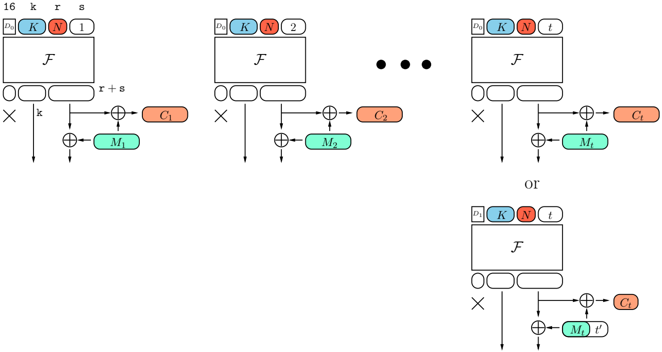
- Security level up to 128 bits and higher (up to $w/3$) and equal to the key length;
- Compact security proof in the random permutation setting;
- Permutation inputs and outputs are linked by only XORs and counters, no extra operations;
- Only forward permutation calls.

PAEQ



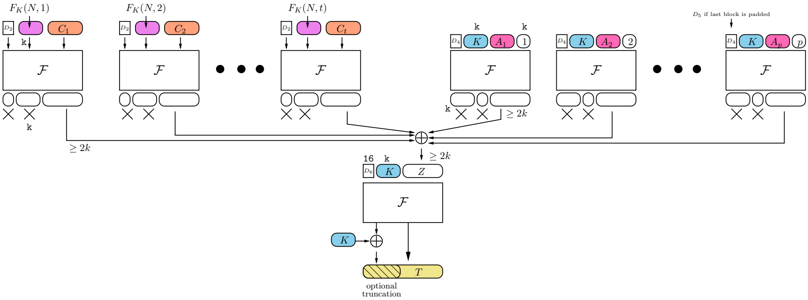
Encryption
Binding associated data
Authentication
Encryption of the last block of length t′
Nonce-misuse option
D_i = (k, i + r (mod 256))
key, k bits
nonce, r bits r + s ≥ 2k
counter, s bits
optional truncation
D₅ if last block is padded
Q:
plaintext length
AD length
key length
nonce length
key
plaintext
AD
sponge padding

Encryption:



- Counter mode with PRF;
- Confidentiality basically follows from the properties of CTR.

Authentication:

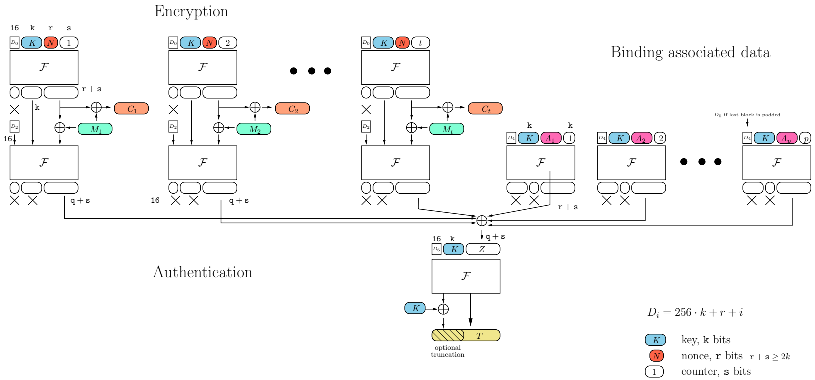


- PMAC style with additional input from the encryption part;
- If the tag has full length, it can be updated with a few extra calls.

PAEQ comes with several security proofs. Confidentiality and integrity are established up to 2^k total queries to $\mathcal{F}$:

$$\mathbf{Adv}_{\Pi}^{\mathrm{conf}}(\mathcal{A}) \leq \frac{3q}{2^k};$$
$$\mathbf{Adv}_{\Pi}^{\mathrm{int}}(\mathcal{A}) \leq \frac{q}{2^\tau} + \frac{4q}{2^k}.$$

where k — key length, τ — tag length, q — total number of queries to $\mathcal{F}$.
If the nonce is misused, integrity is still established up to $2^{k/2}$ queries.



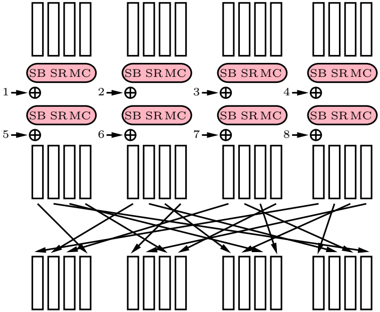
We use our own permutation — AESQ.

AESQ

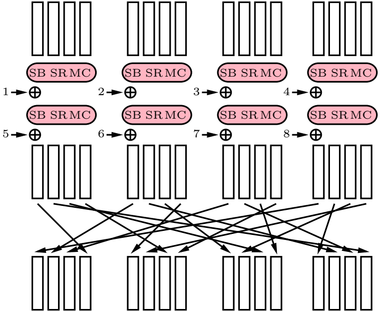
New 512-bit permutation aimed at modern CPUs:

- 4 parallel AES states;
- 2 AES rounds alternated with column shuffle;
- Simple round constants;
- 20 rounds in total.

2 rounds of AESQ:



Running two instances of AESQ in parallel yields highest throughput on Haswell processors.



Security of AESQ:

- Differential/linear properties disappear after 8 rounds;
- Rebound attacks stop at 12 rounds;
- Preimage/distinguishing attacks stop at 12-14 rounds.

Benchmarks on the Haswell CPU:

Security level / Key length	PAEQ (20 rounds, cycles per byte)
64	4.9
80	5.1
128	5.8
256	8.9

Questions?