Authentication and Encryption

How do these two concepts mix

The basic concepts

- **Encryption**

Ensures confidentiality of the transmitted information.

Does not ensure integrity or identity.

(Sometimes, identity proofs may be outright undesirable. See: TOR).

- **Authentication**

Ensures identity and, usually, integrity of the transmitted information.

Does not interfere with the transmitted information itself.

May be *short-time* (session) or *long-time* (digital signature).

Usual mode of employment

- During a typical Alice-2-Bob key exchange, a common shared secret *S* is established and separate keys negotiated for encryption and authentication.
- An authentication tag is computed for the plaintext *and possibly some extra data*, then the plaintext is encrypted, and the authentication tag appended to the ciphertext.
- Examples: SSL/TLS, PGP, SRTP, OTR.

Authenticated Encryption

- We want to have it all.
- As usual, such a state is hard to reach.
- One of the answers: OCB (Offset Codebook Mode).
  - Developed since 2001 as „OCB1“.
  - The current version is „OCB3“ (2011).
  - Adds iterative „offset“ to a classical ECB.

OCB – basic properties

- Very good performance (about same as CTR)
  - Mainly by careful design, which limits many calls to the underlying block cipher.
  - For the runtime of CTR, we get ***two*** effects:
    - Encryption
    - Authentication.
- Parallel processing of subsequent messages possible and easy.
- Effective for authentication of fixed headers.
- Basically the very simple ECB and so-called „offset“.

OCB3 – inputs (1)

- A block cipher BC of block length 128 bits.
  - KEYLENGTH *k* in bits (128, 192, 256…)
  - ENCIPHER(K,P) … the encryption procedure
  - DECIPHER(K,C) … the inverse (decryption) procedure

  Both the ENCIPHER and DECIPHER procedures operate on 128-bit blocks.

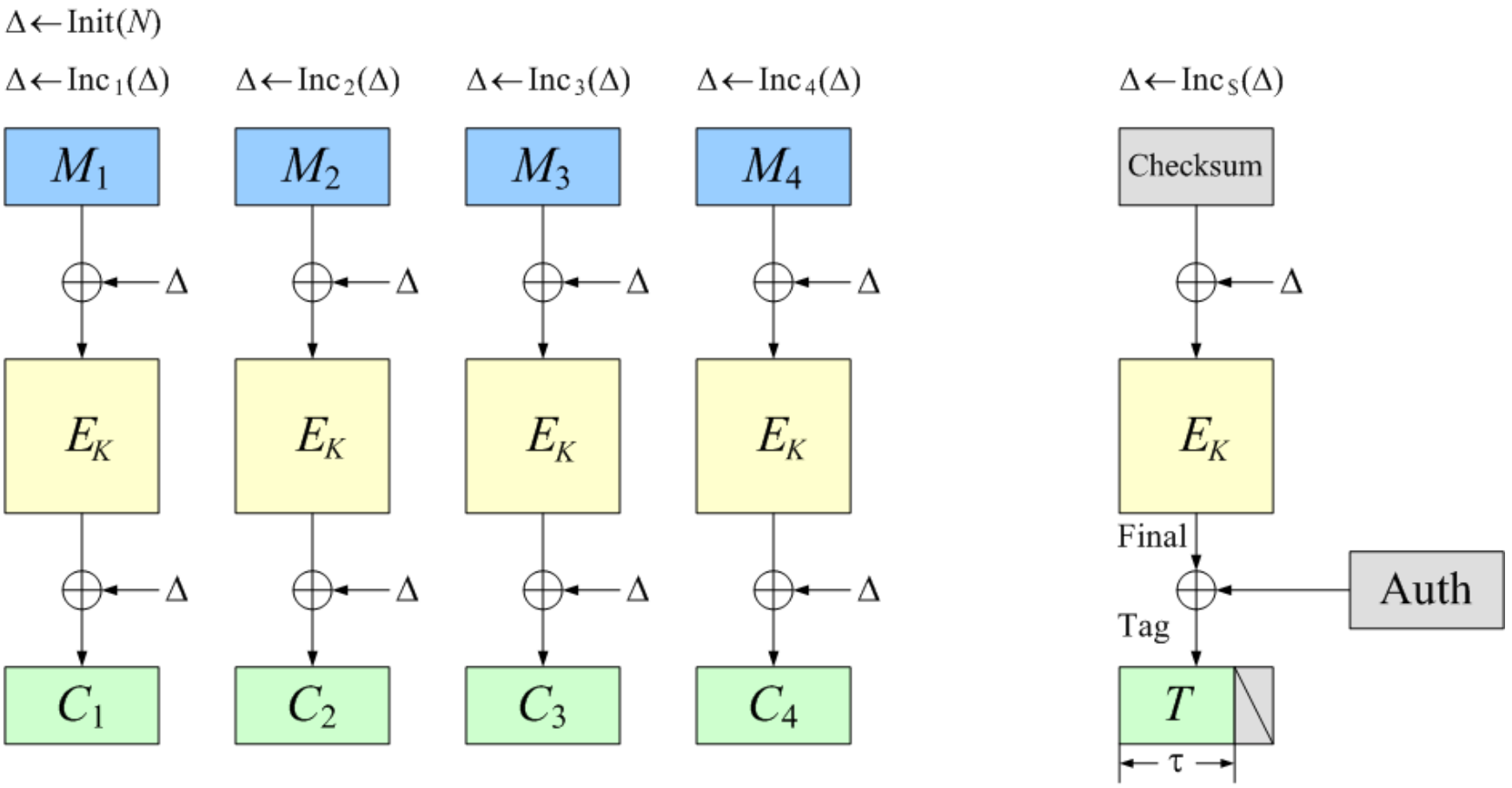
OCB3 – inputs (2)

- *K* … the key string of KEYLENGTH bits,
- *M* … the message to be encrypted,
- *N* … the nonce, a 96-bit string,
- *A* … the *associated data*, a string which need to be authenticated, but not encrypted.

Note that *A* may be an empty string.

Nevertheless, in real world it often isn't. For example, *A* may be headers or routing information.

The overall design scheme



The „Offset“

- Core element of the design. Denoted by Δ.
- Let every 128-bit binary string represent an element of GF(128).
  - Addition: simple XOR.
  - Multiplication: if a suitable polynomial is used, we have easy multiplication by *x*.
  - Additionally, if we choose a primitive polynomial, then *x* is generator of the group and thus the length of the cycle r, rx, rx^2, rx^3 … is maximal for every nonzero *r*.

Performance

- A 128-bit Gray code is used to permutate the set r, rx, rx^2, rx^3 …
- With use of the Gray code, iterating the offset is extremely computationally easy:
  - At the beginning of the encryption process, pre-compute a table of values $L_0, L_1, L_2 \ldots L_{127}$.
  - Then, just iterate Δ by XOR-ing the current value of Δ with a suitable L.

OCB3 – component functions

- **Init()**: does not depend on the message, only on N. Done before the actual encryption.
- **Inc()**: an “offset iteration” operation. Does not depend on plaintext or ciphertext.
- **Checksum()**: a simple binary addition of all plaintext message blocks.
- **Auth()**: the final authentication tag computation, including the associated data **A**.

The Init() function

- Depends on *N* and *K,* but not *M.*
- Performed before start of the encryption process of a new message M (when the counter *N* is usually incremented).
- If nonce *N* is counter, then Init() only uses block cipher once per 64 invocations.
  - For multiple messages in a session, saves up to 63 block cipher operations. Massive performance improvement, esp. for shorter messages.
- Produces the initial offset value Δ and the tables L_0, L_1, L_2 … L_{127}.
  - L_i are produced by bit shifts and binary additions – fast.

Init() inside:

- Concatenate 0x00000001 || N (128 bits).
- Mask out the lower 6 bits, remember them as „Bottom“. The masked-out value is „Top“
- Calculate Ktop = ENCIPHER(K, Top).
  - Note that for N a counter, Ktop changes only once per 64 values of N.
- Stretch Ktop to 256 bits: Stretch = Ktop || (Ktop$\oplus$(Ktop<<8))
- The initial Δ is equal to Stretch << Bottom.

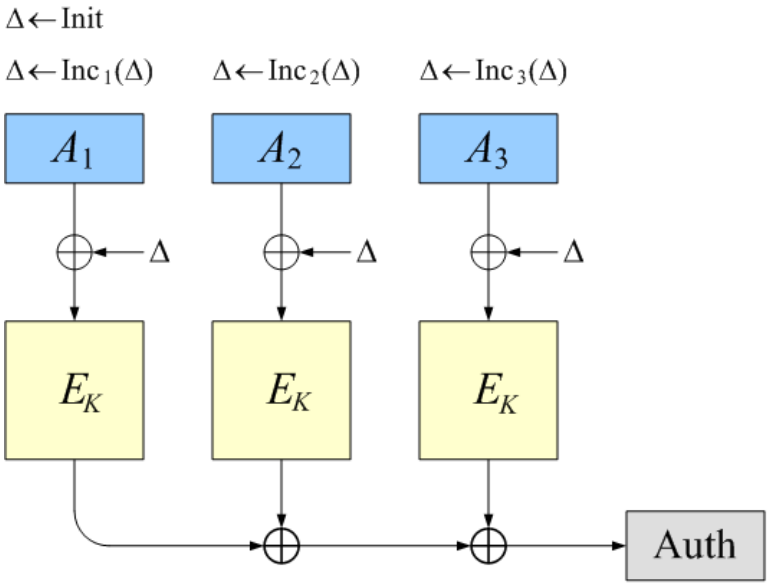
The Inc() function.

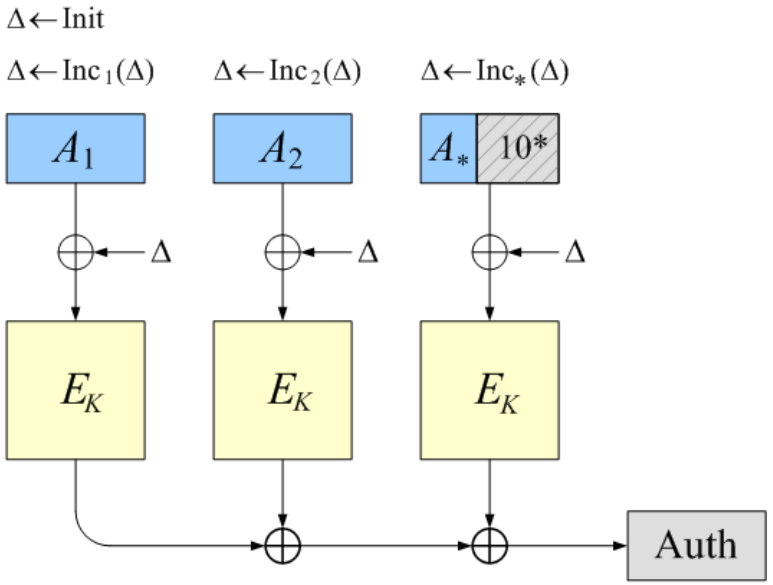
- Advances the offset, using a Gray code permutation.
- Performed before every block encryption.
- **$\Delta = \Delta$ xor L_j ,** where **j = NTZ(i)**
    - NTZ = number of trailing zeros.
- Very fast. 128-bit XORing is often directly supported in modern processor instruction sets.

The Checksum() and Auth() functions

- **Checksum = M_1 xor M_2 xor … xor M_m.**
- A simple XOR of the entire set of blocks, easy to compute.
- **Auth** involves the associated data (A).
- A simple hash, which produces 0 (= a 128-bit zero vector) for empty string.
- Uses the same Init() and Inc() as the encryption process.

Auth scheme


Δ ← Init
Δ ← Inc 1(Δ)
Δ ← Inc 2(Δ)
Δ ← Inc 3(Δ)
A1
A2
A3
Δ
Δ
Δ
EK
EK
EK
Auth


Δ ← Init
Δ ← Inc 1(Δ)
Δ ← Inc 2(Δ)
Δ ← Inc*(Δ)
A1
A2
A*
10*
Δ
Δ
Δ
EK
EK
EK
Auth

Performance II.

- Let us denote the length of M in blocks as *m*, the length of A in blocks as *a*.
- On average, the scheme uses $m + a + 1.016$ block cipher invocations.
- If A is constant throughout session, the Auth value needs to be computed just once.
- Unwieldy algorithms such as 128-bit addition are avoided.

Security

- By adding the offset both to the plaintext before encryption, and to the ciphertext after encryption, it can be shown that the scheme is resistant to chosen-plaintext and chosen-ciphertext attack.
- The concept of *tweakable blockcipher* with the above properties can be generalized.
- Proven in:
  - Krovetz, Rogaway: The Software Performance of Authenticated-Encryption Modes (2011)