Kriptografi Bobot Ringan
(*Lightweight Cryptography*)

Oleh: Dr. Ir. Rinaldi, M.T

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika (STEI)
Institut Teknologi Bandung

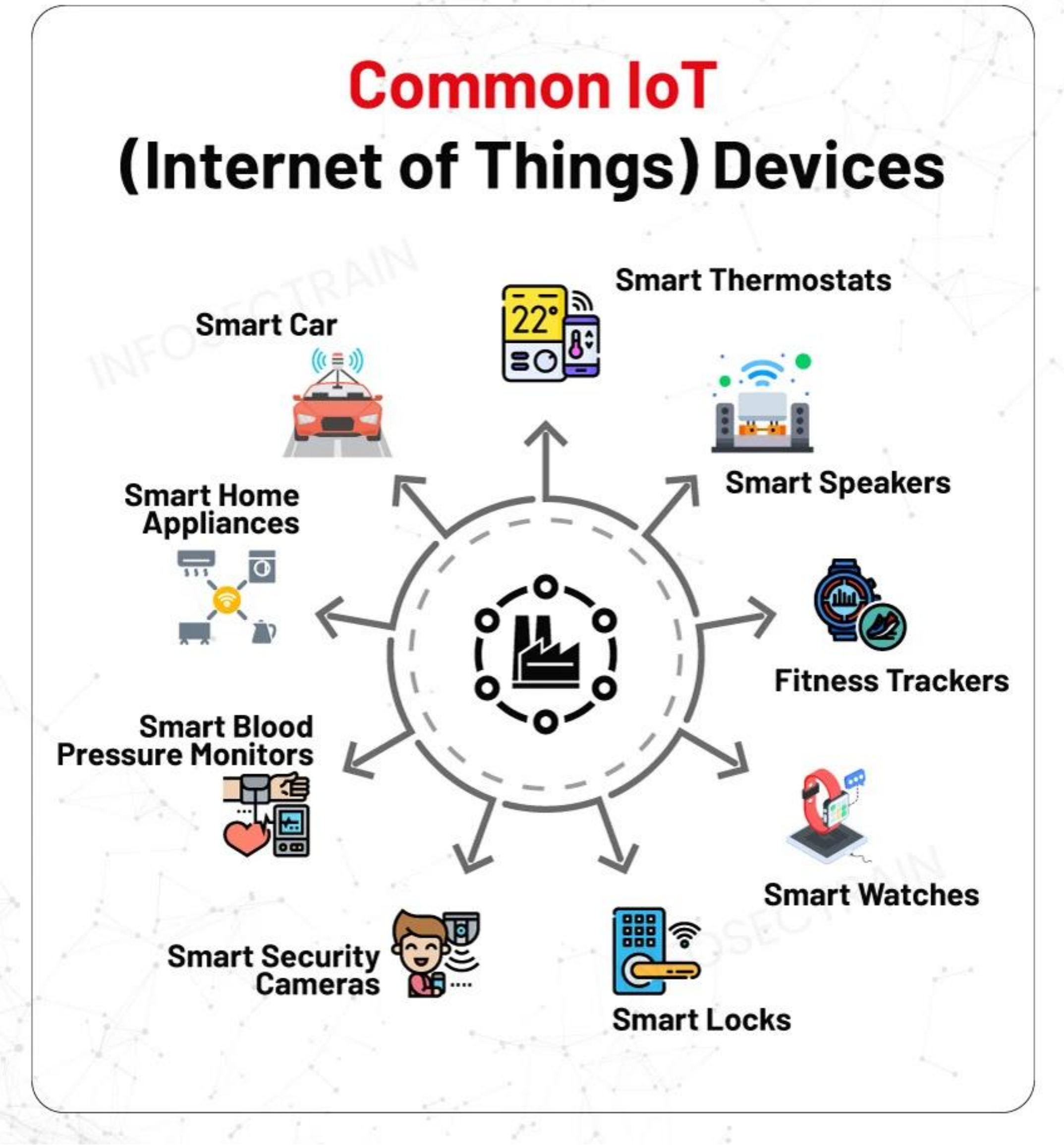
Lightweight Cryptography (LWC)

- *Lightweight Cryptography (LWC)* adalah cabang kriptografi yang dirancang untuk perangkat dengan sumberdaya sangat terbatas (*low resource*) seperti:
  - IoT devices (sensor, tag, node jaringan kecil)
  - Smart card, RFID tag
  - Medical implants
  - Wireless sensor networks
  - Embedded microcontrollers 8-bit / 16-bit

- Perangkat tersebut sering memiliki:
  - Memori kecil (RAM/ROM sangat terbatas)
  - Daya rendah (baterai kecil)
  - Prosesor sederhana
  - Koneksi lambat


SMART CARD
07/17


RFID


Common IoT
(Internet of Things) Devices
Smart Thermostats
Smart Car
Smart Speakers
Smart Home Appliances
Fitness Trackers
Smart Blood Pressure Monitors
Smart Watches
Smart Security Cameras
Smart Locks

- Oleh karena itu, algoritma kriptografi konvensional seperti AES, Twofish, Serpent, ElGamal, RSA, atau ECC terlalu berat untuk perangkat seperti itu.

- LWC menawarkan algoritma yang:
  - ringan (ringan dalam memori dan komputasi)
  - hemat energi
  - tetap aman terhadap serangan

Mengapa *Lightweight Cryptography* Dibutuhkan?

- **Kendala perangkat *low-resource***
  - AES biasa butuh ~20–30k *gate equivalent* (GE) → terlalu besar untuk RFID (4–8k GE).
  - RSA/ECC terlalu lambat dan butuh memori besar.
  - Banyak algoritma konvensional tidak mendukung kebutuhan komunikasi yang sangat kecil.

- **Lingkungan sangat rentan**

  Banyak perangkat IoT:
  - tidak bisa diperbarui (*firmware locked*)
  - berada di lokasi publik sehingga mudah diserang secara fisik
- Sehingga algoritma harus:
  - Sangkil (efisien)
  - mampu mendeteksi manipulasi (*authenticated encryption*)
  - mudah diimplementasi dalam *hardware* kecil

Karakteristik Lightweight Cryptography

1. Area kecil (GE rendah)

   Target umum: <10k GE (atau bahkan <2k GE untuk RFID).
2. Memori sangat kecil

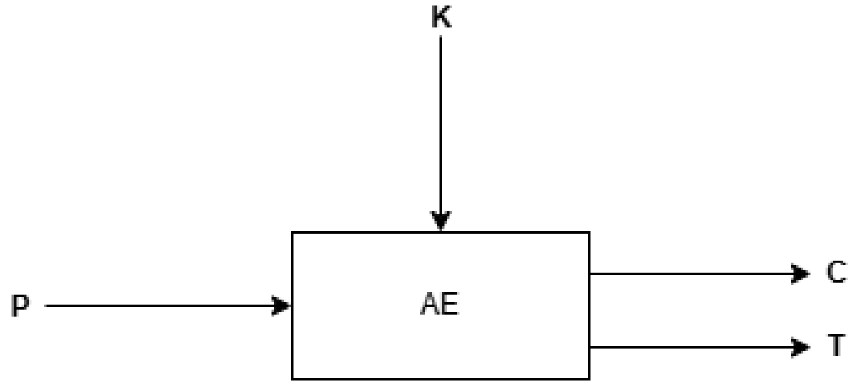
   RAM beberapa ratus byte saja.
3. Keamanan sekelas *modern cryptography*, meski ringan.
4. Mendukung *Authenticated Encryption*

   Mayoritas aplikasi IoT butuh: *confidentiality* + *integrity* → AEAD (*Authenticated Encryption with Associated Data*) menjadi standar.

Authenticated Encryption (AE)

- *Authenticated Encryption* (AE) adalah *kelas skema kriptografi simetris* yang menggabungkan dua fungsi utama:
  - *Confidentiality* — menjaga kerahasiaan (data dienkripsi).
  - *Integrity & Authenticity* — memastikan data tidak diubah dan berasal dari pihak yang benar.

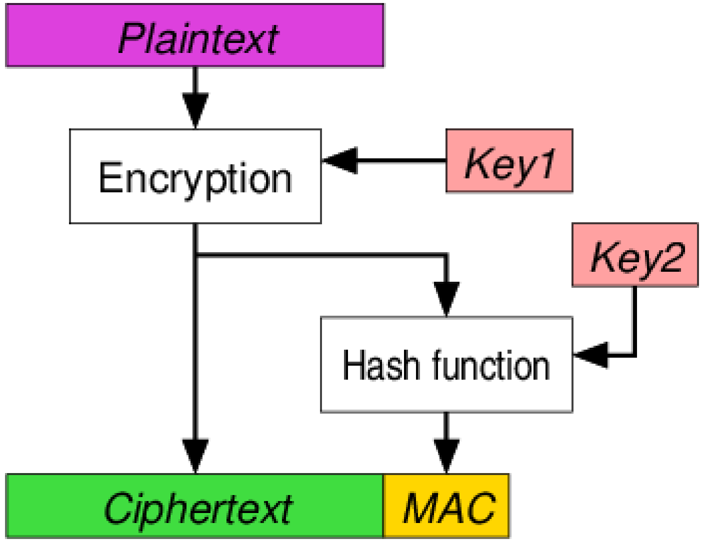
- AE menyatukan *encryption* + *authentication* (MAC) dalam satu skema yang aman.



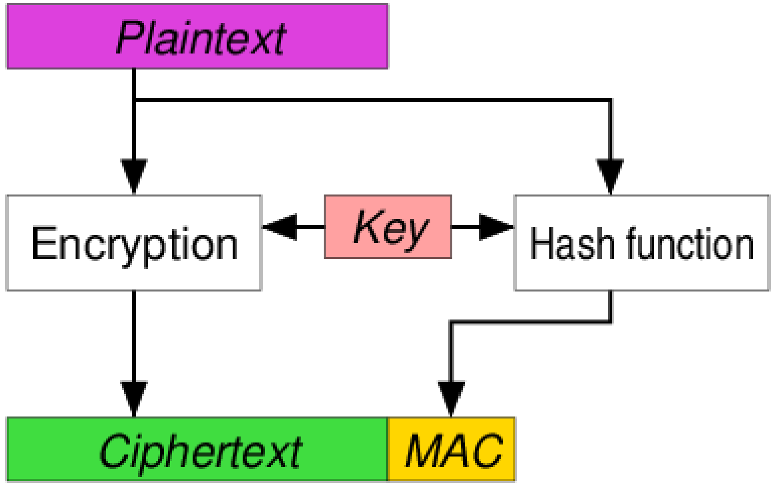
P = Plainteks
K = Kunci
C = Cipherteks
T = Tag otentikasi

- Bermacam-macam skema AE:

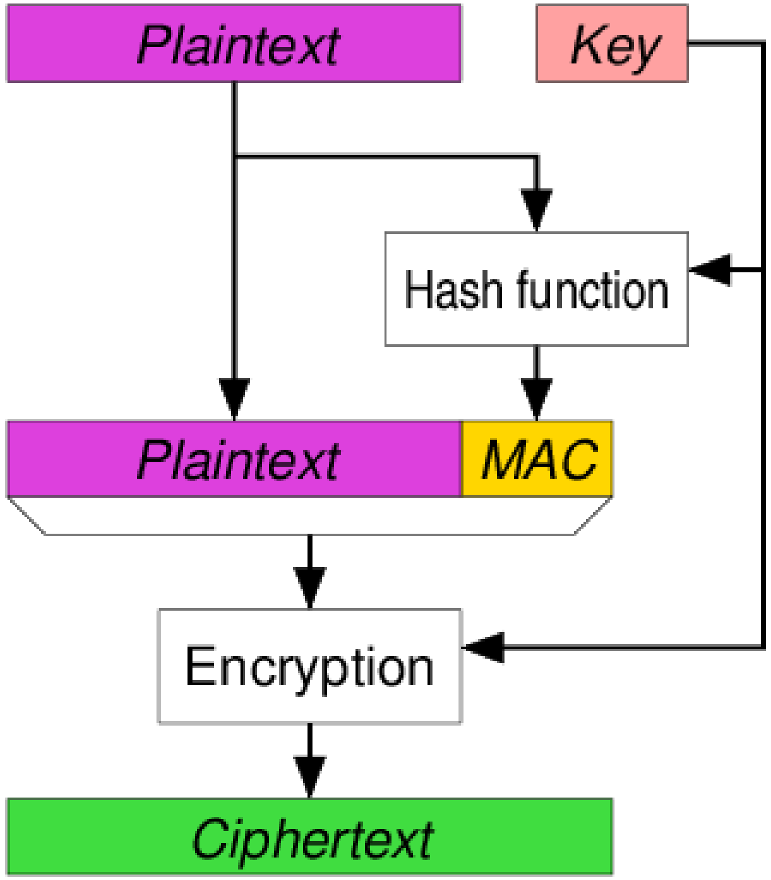
1)



2)



3)

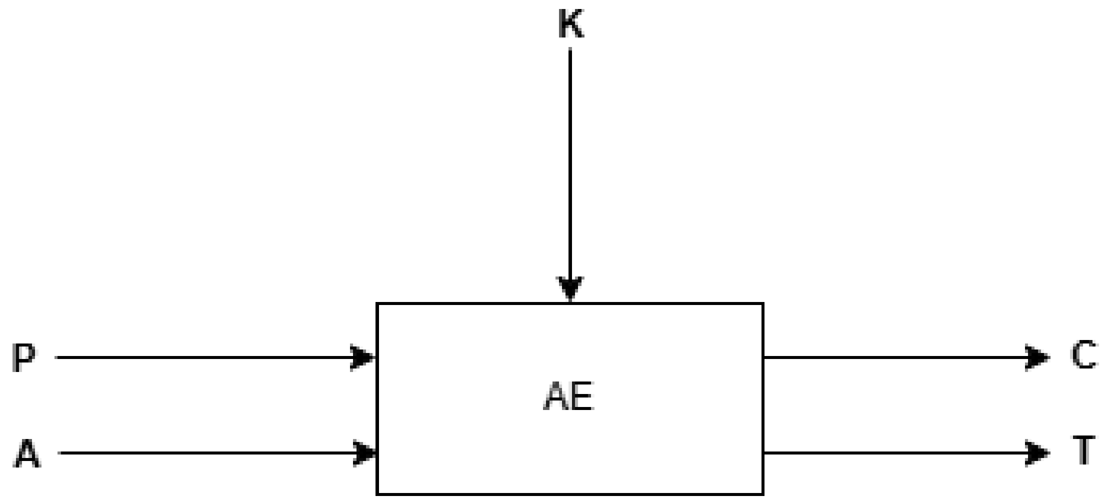


Sumber gambar: Wikipedia

Authenticated Encryption with Associated Data (AEAD).

- AEAD adalah perluasan dari AE yang memungkinkan untuk melampirkan data teks tambahan (yang tidak terenkripsi) ke ciphertext sedemikian sehingga jika data teks tambahan rusak/dimanipulasi, tag autentikasi tidak akan tervalidasi dan *ciphertext* tidak akan didekripsi. Data teks biasa tersebut di dalam AEAD disebut sebagai AD (*Associated Data*)
- Jadi, AEAD menyediakan:
  - Enkripsi (*confidentiality*)
  - Autentikasi & integritas (*authentication* + *integrity*)
  - Autentikasi untuk data tambahan (*Associated Data* / AD)
    → data yang tidak dienkripsi tetapi tetap dilindungi integritasnya.

- Dengan AEAD, kita mendapatkan dua perlindungan sekaligus: data tetap rahasia *dan* tidak dapat dimodifikasi / dipalsukan.



P = Plainteks
A = *Associated Data* (AD)
K = Kunci
C = Cipherteks
T = Tag otentikasi

Contoh AD:

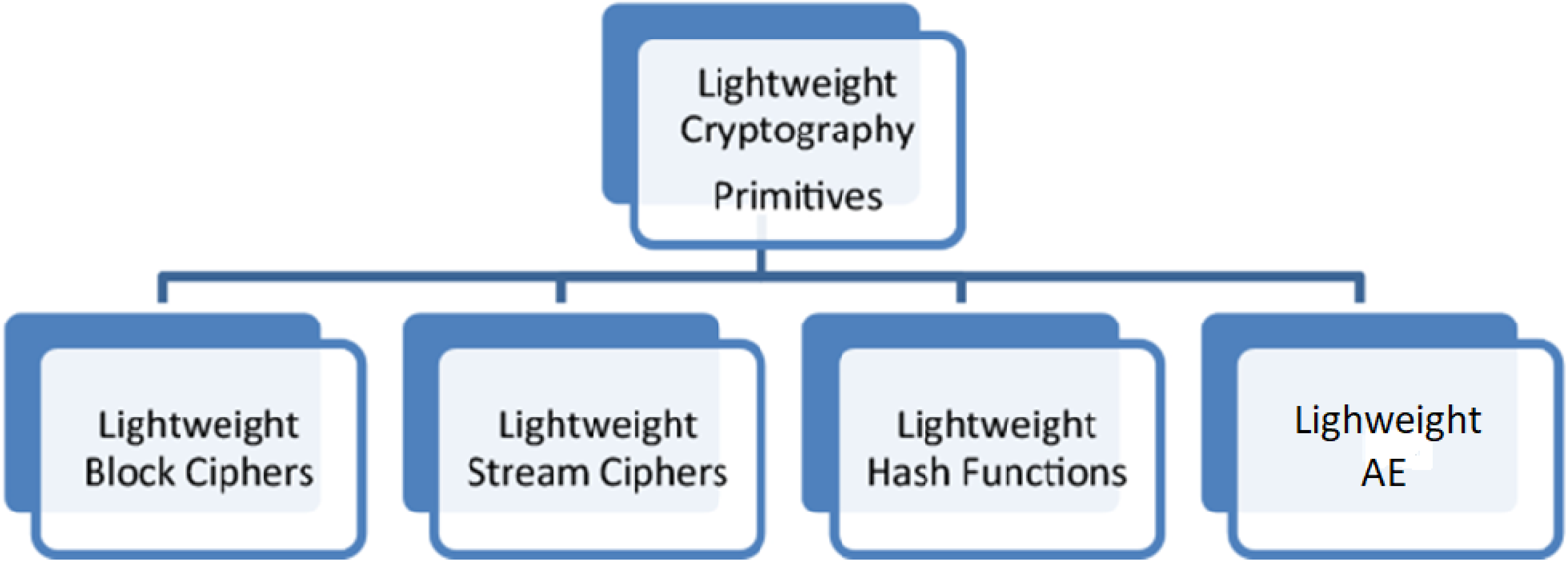
- *header* protokol (IP header, TLS header)
- metadata paket
- alamat, *sequence number*
- device ID, *nonce* eksternal, dsb.

AD tidak dienkripsi, tapi tidak boleh diubah.
Jika berubah sedikit saja → tag gagal → dekripsi ditolak.

- AEAD mengambil input:
    - *Key* (kunci rahasia)
    - *Nonce* (nilai unik per-enkripsi)
    - *Plaintext* (data yang akan dienkripsi)
    - *Associated Data* (AD) — tidak dienkripsi tapi harus dilindungi integritasnya

- AEAD menghasilkan output:
    - *Ciphertext* (hasil enkripsi plaintext)
    - *Authentication Tag* (tag / MAC / digest)
    - Ketika mendekripsi, AEAD akan mengecek tag, jika tag tidak cocok → dekripsi gagal, data tidak diberikan (integrity fail)

Klasifikasi Lightweight Cryptography



A. Block Cipher ringan

Contoh *block cipher lightweight*:

- *Block cipher* versi kecil dari AES (AES/LWC)
- **PRESENT**
- **KATAN / KATANTAN**
- **SIMON (by NSA)**
- **SPECK (by NSA, sekarang dihentikan penggunaannya)**
- **PRIDE**
- **RECTANGLE**
- **GIFT**

B. Stream Cipher ringan

Lebih kecil dan lebih cepat untuk hardware terbatas.

Contoh:

- **Trivium**
- **Grain**
- **Spritz**
- **Enocoro**

C. *Hash Function* ringan

Untuk autentikasi dan verifikasi integritas.

Contoh:

- **PHOTON**
- **SPONGENT**
- **Lesamnta-LW**

D. AEAD (*Authenticated Encryption with Associated Data*) ringan

Kategori paling penting dalam standardisasi modern.

Contoh AEAD ringan:

- **ASCON** → Pemenang NIST LWC 2023
- **ACORN**
- **CAESAR candidates** (AES-GCM-SIV, OCB)

Contoh Algoritma Lightweight Cryptography Terpopuler

Kategori	Algoritma	Keterangan
Block Cipher	**PRESENT**	Salah satu cipher ringan paling populer (<2k GE)
Block Cipher	**SIMON**	Desain NSA, cepat dan kecil
AEAD	**ASCON**	Standar NIST LWC
AEAD	**ACORN**	Finalis NIST LWC
Hash	**PHOTON**	Hash super ringan
Hash	**SPONGENT**	Mirip SHA-3 tapi lebih kecil
Stream Cipher	**Trivium**	Bagian dari eSTREAM
Stream Cipher	**Grain**	Sangat ringan untuk RFID

Mengapa tidak ada LWC untuk Public-Key Cryptography?

1. Algoritma kriptografi kunci publik jauh lebih kompleks daripada *symmetric-key cryptography*, baik kompleks secara *hardware* maupun *software*

Perbandingan kompleksitas hardware:

Jenis Kripto	**Perkiraan GE**
AES	20k–30k GE
PRESENT	1.5k GE
ASCON	2.6–9.4k GE
RSA-1024	**120k–200k GE**
ECC-160	**40k–80k GE**
ECC-256	**100k+ GE**
PQC (Kyber/Dilithium)	**ratusan ribu GE** (jauh lebih besar)

2. Algoritma kriptografi kunci publik sangat boros energi

Pada perangkat ultra-terbatas (RFID pasif, sensor energi-*harvesting*):

- Operasi AES/PRESENT butuh mikro-Joule
- Operasi ECC butuh mili-Joule, ribuan kali lebih besar

RFID tidak memiliki baterai → energinya hanya dari medan elektromagnetik *reader*.
Mengerjakan ECC/RSA tidak mungkin secara praktis.

3. Algoritma kriptografi kunci-publik sangat lambat pada perangkat lemah

Operasi perkalian scalar 256-bit dengan titik di dalam ECC bisa memakan:

- ratusan ribu hingga jutaan *clock cycle*
- bahkan pada mikrokontroler 8-bit → > 1 detik

Sedangkan AES/LWC hanya:

- beberapa ribu cycle
- cocok untuk latency rendah

Untuk RFID yang harus merespons < 10 ms, kriptografi kunci publik adalah *mission impossible*.

5. *Use-case lightweight* biasanya tidak membutuhkan public-key

- *Use ca*se penggunaan LWC adalah adalah untuk autentikasi sederhana, enkripsi cepat, integritas data. Semua ini bisa dilakukan oleh kriptografi simetris saja. Tidak perlu *public-key cryptography*.
- *Public-key cryptography* dibutuhkan untuk *key exchange* dan *digital signature*, tapi perangkat ultra-terbatas biasanya disiapkan dengan kunci terlebih dahulu (*pre-shared key*), sehingga tidak butuh *public-key.*

6. Standar *NIST Lightweight Cryptography* (LWC) memang hanya untuk *symmetric cryptography* saja

- Dokumen NIST LWC (SP 800-232) secara eksplisit menyatakan:

Lightweight Cryptography focuses on symmetric primitives (AEAD and hash functions), not public-key cryptography.

- Karena public-key cryptography *overhead* terlalu besar, tidak sesuai target platform, dan tidak realistis untuk 8-bit microcontroller atau RFID

NIST Lightweight Cryptography Standard (2023)

- Pada tahun 2018–2023, divisi *computer security* NIST mengadakan kompetisi LWC untuk menemukan algoritma AEAD dan *hash function* yang aman untuk IoT.
- Pemenang Final (2023): ASCON Family
- ASCON family termasuk:
  - ASCON-128 (AEAD)
  - ASCON-128a (AEAD)
  - ASCON-80pq (AEAD, mendukung *post-quantum security* level kecil)
  - ASCON-HASH (hash)
  - ASCON-XOF (*extendable-output function*)
- Alasan terpilih:
  - Area kecil (≈2.5k GE)
  - Aman dari serangan klasik dan side-channel
  - Performa baik di hardware dan software
  - Struktur *sponge* yang sederhana

NIST

Computer Security Division (CSD)

NIST

Developing Crypto Standards

- International "competitions" e.g., AES, SHA-3, PQC, Lightweight Crypto
- Adoption of existing standards e.g., RSA, HMAC
- Open call for proposals: e.g., block cipher modes of operations

CSD Publications

- Federal Information Processing Standards (FIPS): Specify approved crypto standards
- NIST Special Publications (SPs): Guidelines, technical specifications, recommendations etc.
- NIST Internal or Interagency Reports (IR): Reports of research findings

Principles

Transparency, openness, balance, integrity, technical merit, global acceptability, usability, continuous improvement, innovation and intellectual property etc.

Sumber: Meltem Sonmez Turannist, Lightweight Cryptography Standardization, GLOBAL PLATFORM SECURITY TASK FORCE MEETING, MARCH 28, 2023, NIST

Lightweight Cryptography – Motivation

NIST

CONSTRAINED DEVICES

e.g., RFID tags, sensors, IoT devices

NEW APPLICATIONS

e.g., home automation, healthcare, smart city

PRIVATE INFORMATION

e.g., location, health data

LACK OF CRYPTOGRAPHY STANDARDS

NIST crypto standards are optimized for general-purpose computers.

Sumber: Meltem Sonmez Turannist, Lightweight Cryptography Standardization, GLOBAL PLATFORM SECURITY TASK FORCE MEETING, MARCH 28, 2023, NIST

Rounds of Evaluation

NIST

Round 1	Round 2	Round 3
April 2019 – August 2019	August 2019 – March 2021	March 2021 – February 2023
56 Round–1 Candidates	32 Round–2 Candidates	10 Finalists
Evaluation based on security	Evaluation based on security and performance	Evaluation based on security, performance (including protected implementations) and additional features

Finalists

NIST

ASCON	Elephant	GIFT-COFB	Grain-128aead	ISAP
Photon-Beetle	Romulus	Sparkle	TinyJambu	Xoodyak

Variants

Finalist	# Variants	Key size (bits)	Nonce size (bits)	Tag size (bits)	Digest size (bits)
Ascon	2 AEAD 2 hash	128 --	128 --	128 --	-- 256
Elephant	3 AEAD	128	96	64-128	--
GIFT-COFB	1 AEAD	128	128	128	--
Grain-128aead	1 AEAD	128	96	64	--
ISAP	4 AEAD	128	128	128	--
PHOTON-Beetle	2 AEAD 1 hash	128 --	128 --	128 --	-- 256
Romulus	3 AEAD 1 hash	128 --	128 --	128 --	-- 256
Sparkle	4 AEAD 2 hash	128-256 --	128-256 --	128-256 --	-- 256-384
TinyJambu	3 AEAD	128-256	96	64	
Xoodyak	1 AEAD 1 hash	128 --	128 --	128 --	-- 256

ASCON

- ASCON adalah keluarga algoritma kriptografi ringan yang terdiri dari:
  1. ASCON-128 (AEAD)
  2. ASCON-128a (AEAD, throughput lebih tinggi)
  3. ASCON-80pq (AEAD, resistansi post-quantum level ringan)
  4. ASCON-HASH, ASCON-XOF (hash & extendable output)

- Dirancang oleh kelompok peneliti Austria–Swiss (ETH Zürich, Graz University).

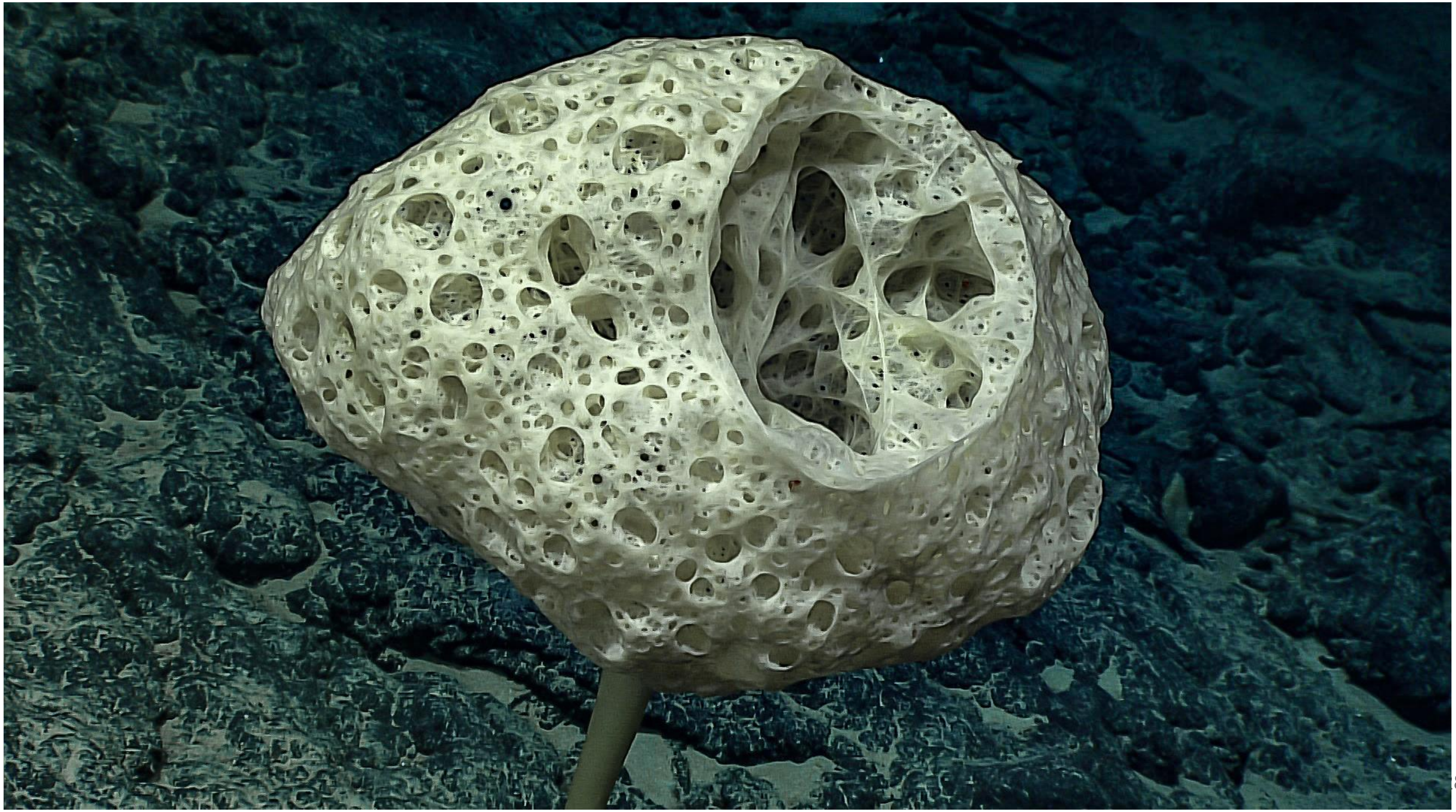
- Pada tahun 2023, ASCON dipilih sebagai NIST Lightweight Cryptography Standard, mengalahkan puluhan finalis lain.

The Ascon Team

- Martin Schläffer
- Florian Mendel
- Christoph Dobraunig
- Maria Eichlseder

(c) Lunghammer, TU Graz

- ASCON menggunakan struktur **Sponge,** mirip SHA-3, namun lebih kecil, round function lebih ringan, desain difokuskan untuk AEAD

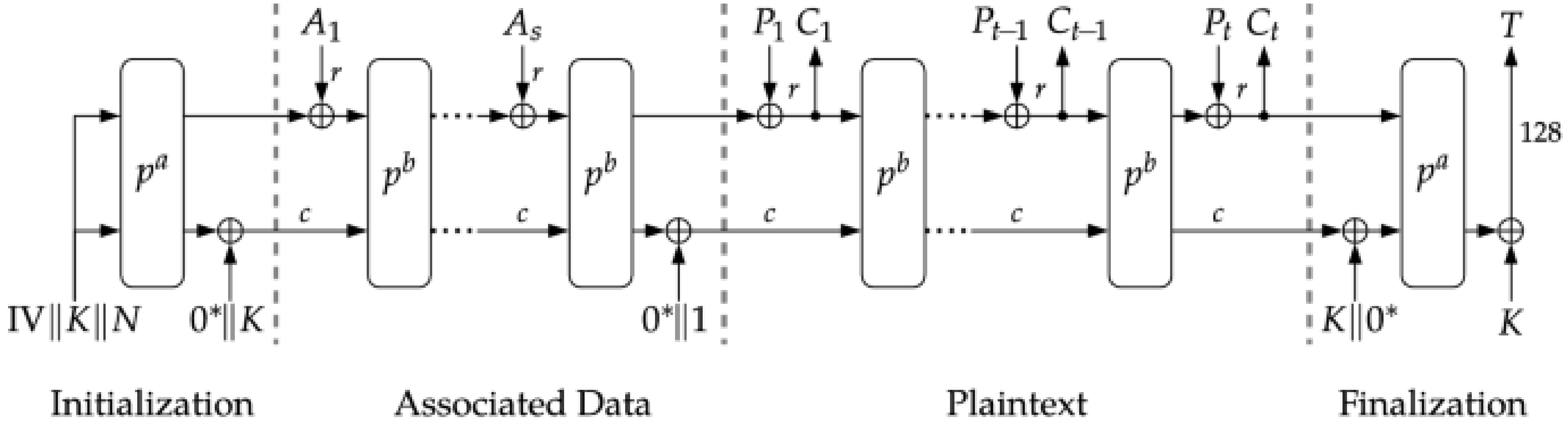


- ASCON menggunakan pemutasi ringan yang menerapkan jaringan subtitusi-permutasi (SPN) dengan jumlah transformasi *round* hingga 16.

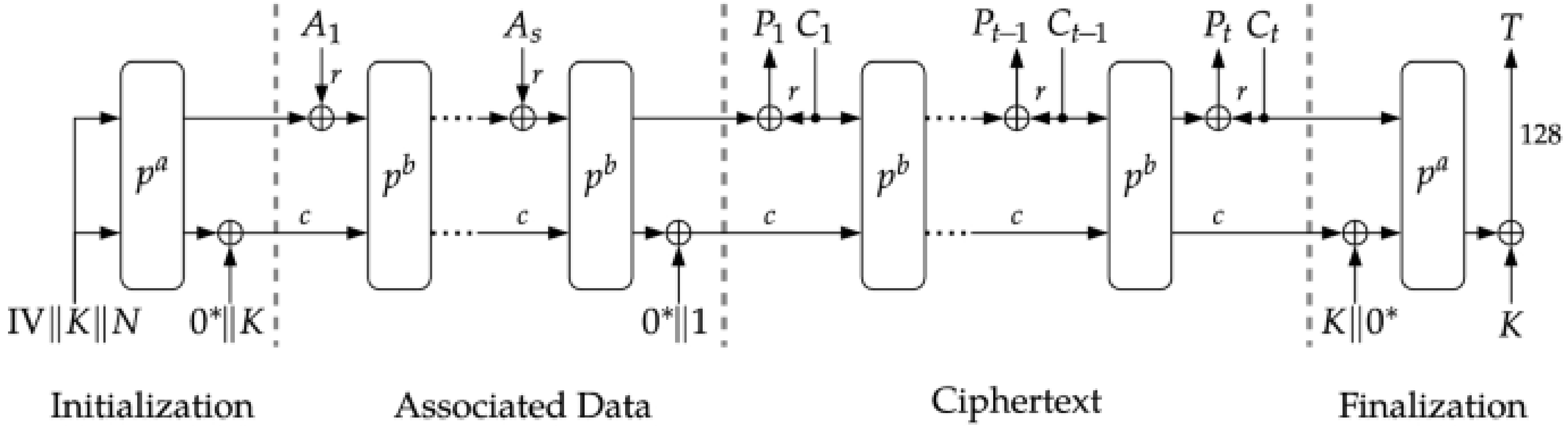
- Setiap tranformasi *round* mengoperasikan 320 bit *state* (state internal ASCON) yang dibagi menjadi 5 *words* yaitu *S0, S1, S2, S3, S4* yang setiap state panjangnya 64 bit.
- Overview Proses ASCON (AEAD), untuk ASCON-128:

  1. *Initialisation*
     Masukkan key + nonce + IV ke state.
  2. *Process Associated Data* (AD)
     Data tidak terenkripsi tapi diautentikasi.
  3. *Plaintext Encryption* → enkripsi plaintext menghasilkan ciphertext.
  4. *Finalisation*
     Masukkan key lagi (Hermetic sealing).
  5. Generate Tag (*authentication tag* 128-bit).

Semua langkah ini menggunakan fungsi permutasi p^a dan p^b.



(a) Encryption $\mathcal{E}_{k,r,a,b}$



(b) Decryption $\mathcal{D}_{k,r,a,b}$

Pada ASCON-128 (AEAD),
r = 64, c = 320 – 64 = 256

Sumber: https://link.springer.com/article/10.1007/s00145-021-09398-9

Algorithm 1: Authenticated encryption and decryption procedures

Authenticated Encryption
$\mathcal{E}_{k,r,a,b}(K, N, A, P)$

Input: key $K \in \{0,1\}^k$, $k \leq 160$,
nonce $N \in \{0,1\}^{128}$,
associated data $A \in \{0,1\}^*$,
plaintext $P \in \{0,1\}^*$

Output: ciphertext $C \in \{0,1\}^{|P|}$,
tag $T \in \{0,1\}^{128}$

Initialization
$S \leftarrow \mathrm{IV}_{k,r,a,b} \parallel K \parallel N$
$S \leftarrow p^a(S) \oplus (0^{320-k} \parallel K)$

Processing Associated Data
if $|A| > 0$ **then**
$A_1 \ldots A_s \leftarrow r$-bit blocks of $A\|1\|0^*$
for $i = 1, \ldots, s$ **do**
$S \leftarrow p^b((S_r \oplus A_i) \parallel S_c)$
$S \leftarrow S \oplus (0^{319} \parallel 1)$

Processing Plaintext
$P_1 \ldots P_t \leftarrow r$-bit blocks of $P\|1\|0^*$
for $i = 1, \ldots, t-1$ **do**
$S_r \leftarrow S_r \oplus P_i$
$C_i \leftarrow S_r$
$S \leftarrow p^b(S)$
$S_r \leftarrow S_r \oplus P_t$
$\tilde{C}_t \leftarrow \lfloor S_r \rfloor_{|P| \bmod r}$

Finalization
$S \leftarrow p^a(S \oplus (0^r \parallel K \parallel 0^{320-r-k}))$
$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
return $C_1 \parallel \ldots \parallel C_{t-1} \parallel \tilde{C}_t, T$

Verified Decryption
$\mathcal{D}_{k,r,a,b}(K, N, A, C, T)$

Input: key $K \in \{0,1\}^k$, $k \leq 160$,
nonce $N \in \{0,1\}^{128}$,
associated data $A \in \{0,1\}^*$,
ciphertext $C \in \{0,1\}^*$,
tag $T \in \{0,1\}^{128}$

Output: plaintext $P \in \{0,1\}^{|C|}$ or $\perp$

Initialization
$S \leftarrow \mathrm{IV}_{k,r,a,b} \parallel K \parallel N$
$S \leftarrow p^a(S) \oplus (0^{320-k} \parallel K)$

Processing Associated Data
if $|A| > 0$ **then**
$A_1 \ldots A_s \leftarrow r$-bit blocks of $A\|1\|0^*$
for $i = 1, \ldots, s$ **do**
$S \leftarrow p^b((S_r \oplus A_i) \parallel S_c)$
$S \leftarrow S \oplus (0^{319} \parallel 1)$

Processing Ciphertext
$C_1 \ldots C_{t-1}\tilde{C}_t \leftarrow r$-bit blocks of $C, 0 \leq |\tilde{C}_t| < r$
for $i = 1, \ldots, t-1$ **do**
$P_i \leftarrow S_r \oplus C_i$
$S \leftarrow C_i \parallel S_c$
$S \leftarrow p^b(S)$
$\tilde{P}_t \leftarrow \lfloor S_r \rfloor_{|\tilde{C}_t|} \oplus \tilde{C}_t$
$S_r \leftarrow S_r \oplus (\tilde{P}_t \parallel 1 \parallel 0^*)$

Finalization
$S \leftarrow p^a(S \oplus (0^r \parallel K \parallel 0^{320-r-k}))$
$T^* \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
if $T = T^*$ **return** $P_1 \parallel \ldots \parallel P_{t-1} \parallel \tilde{P}_t$
else return $\perp$

1. Initialization

The 320-bit initial state of Ascon is formed by the secret key *K* of *k* bits and nonce *N* of 128 bits, as well as an IV specifying the algorithm (including the key size *k*, the rate *r*, the initialization and finalization round number *a*, and the intermediate round number *b*, each written as an 8-bit integer):

$$\mathrm{IV}_{k,r,a,b} \leftarrow k\|r\|a\|b\|0^{160-k} = \begin{cases} \texttt{80400c0600000000} & \text{for Ascon-128} \\ \texttt{80800c0800000000} & \text{for Ascon-128a} \\ \texttt{a0400c06} & \text{for Ascon-80pq} \end{cases}$$

$$S \leftarrow \mathrm{IV}_{k,r,a,b}\|K\|N$$

In the initialization, *a* rounds of the round transformation *p* are applied to the initial state, followed by an xor of the secret key *K*:

$$S \leftarrow p^a(S) \oplus (0^{320-k}\|K)$$

Sumber: https://link.springer.com/article/10.1007/s00145-021-09398-9

2. *Processing Associated Data*

Processing Associated Data ASCON processes the associated data A in blocks of r bits. It appends a single 1 and the smallest number of 0s to A to obtain a multiple of r bits and split it into s blocks of r bits, $A_1 \| \ldots \| A_s$. In case A is empty, no padding is applied and $s = 0$:

$$A_1, \ldots, A_s \leftarrow \begin{cases} r\text{-bit blocks of } A \| 1 \| 0^{r-1-(|A| \bmod r)} & \text{if } |A| > 0 \\ \varnothing & \text{if } |A| = 0 \end{cases}$$

Each block A_i with $i = 1, \ldots, s$ is XORed to the first r bits S_r of the state S, followed by an application of the b-round permutation p^b to S:

$$S \leftarrow p^b((S_r \oplus A_i) \| S_c), \qquad 1 \leq i \leq s$$

After processing A_s (also if $s = 0$), a 1-bit domain separation constant is XORed to S:

$$S \leftarrow S \oplus (0^{319} \| 1)$$

Sumber: https://link.springer.com/article/10.1007/s00145-021-09398-9

3. *Plaintext Encryption*

Processing Plaintext/Ciphertext ASCON processes the plaintext *P* in blocks of *r* bits. The padding process appends a single 1 and the smallest number of 0s to the plaintext *P* such that the length of the padded plaintext is a multiple of *r* bits. The resulting padded plaintext is split into *t* blocks of *r* bits, $P_1 \| \ldots \| P_t$:

$$P_1, \ldots, P_t \leftarrow r\text{-bit blocks of } P\|1\|0^{r-1-(|P| \bmod r)}$$

Encryption In each iteration, one padded plaintext block P_i with $i = 1, \ldots, t$ is XORed to the first *r* bits S_r of the internal state *S*, followed by the extraction of one ciphertext block C_i. For each block except the last one, the whole internal state *S* is transformed by the permutation p^b using *b* rounds:

$$C_i \leftarrow S_r \oplus P_i$$

$$S \leftarrow \begin{cases} p^b(C_i \| S_c) & \text{if } 1 \leq i < t \\ C_i \| S_c & \text{if } 1 \leq t \end{cases}$$

Sumber: https://link.springer.com/article/10.1007/s00145-021-09398-9

The last ciphertext block C_t is then truncated to the length of the unpadded last plaintext block-fragment so that its length is between 0 and $r-1$ bits, and the total length of the ciphertext C is exactly the same as for the original plaintext P:

$$\tilde{C}_t \leftarrow \lfloor C_t \rfloor_{|P| \bmod r}$$

Decryption In each iteration except the last one, the plaintext block P_i is computed by XORing the ciphertext block C_i with the first r bits S_r of the internal state. Then, the first r bits of the internal state, S_r, are replaced by C_i. Finally, for each ciphertext block except the last one, the internal state is transformed by the b-round permutation p^b:

For the last, truncated ciphertext block $\tilde{C}_t$ with $0 \leq \ell < r$ bits, the procedure differs:

$$\tilde{P}_t \leftarrow \lfloor S_r \rfloor_\ell \oplus \tilde{C}_t$$
$$S \leftarrow (S_r \oplus (\tilde{P}_t \| 1 \| 0^{r-1-\ell})) \| S_c$$

4. Finalization

Finalization In the finalization, the secret key K is XORed to the internal state and the state is transformed by the permutation p^a using a rounds. The tag T consists of the last 128 bits of the state XORed with the last 128 bits of the key K:

$$S \leftarrow p^a(S \oplus (0^r \| K \| 0^{c-k}))$$
$$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$$

The encryption algorithm returns the tag T together with the ciphertext $C_1 \| \ldots \| \tilde{C}_t$. The decryption algorithm returns the plaintext $P_1 \| \ldots \| \tilde{P}_t$ only if the calculated tag value matches the received tag value.

Dua Tipe Permutasi: p^a dan p^b

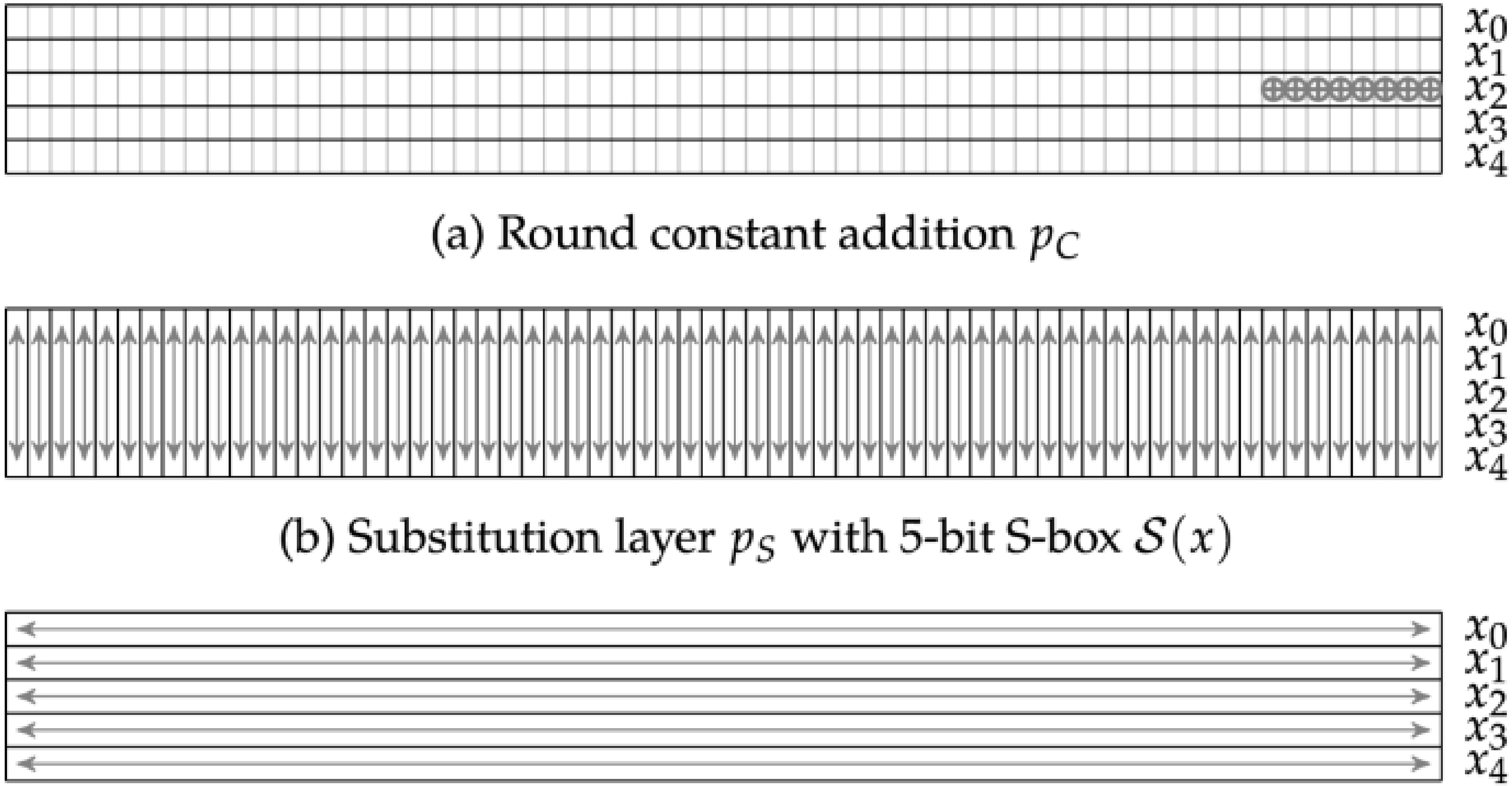
- p^a = 12 rounds
  (digunakan pada tahap *Initialization* & *Finalization*)
- p^b = 6 rounds
  (digunakan untuk *Associated Data* & *Plaintext*)

- Tiap round terdiri dari:
  1. *Add round constant*
  2. *Substitution lay*er (nonlinear S-box)
  3. *Linear diffusion layer*

  Semuanya beroperasi pada lima register 64-bit.

For the description and application of the round transformations, the 320-bit state S is split into five 64-bit registers words x_i, $S = x_0 \| x_1 \| x_2 \| x_3 \| x_4$ (see Fig. 3).

Fig. 3



(a) Round constant addition p_C

(b) Substitution layer p_S with 5-bit S-box $\mathcal{S}(x)$

(c) Linear layer with 64-bit diffusion functions $\Sigma_i(x_i)$

Addition of Constants The constant addition step p_C adds a round constant c_r to register word x_2 of the state *S* in round *i* (see Fig. 3a). Both indices *r* and *i* start from zero and we use $r = i$ for p^a and $r = i + a - b$ for p^b (see Table 6):

$$x_2 \leftarrow x_2 \oplus c_r .$$

Table 6 Round constants c_r used in each round *i* of p^a and p^b

From: ASCON v1.2: Lightweight Authenticated Encryption and Hashing

p^{12}	p^8	p^6	Constant c_r	p^{12}	p^8	p^6	Constant c_r
0			00000000000000f0	6	2	0	0000000000000096
1			00000000000000e1	7	3	1	0000000000000087
2			00000000000000d2	8	4	2	0000000000000078
3			00000000000000c3	9	5	3	0000000000000069
4	0		00000000000000b4	10	6	4	000000000000005a
5	1		00000000000000a5	11	7	5	000000000000004b

Substitution Layer The substitution layer p_S updates the state S with 64 parallel applications of the 5-bit S-box $\mathcal{S}(x)$ defined in Fig. 4a to each bit-slice of the five registers $x_0 \ldots x_4$ (Fig. 3b). It is typically implemented in this bitsliced form with operations performed on the entire 64-bit words, as in the example code in Fig. 5. The lookup table of $\mathcal{S}$ is given in Table 7, where x_0 is the MSB and x_4 the LSB.

Table 7 ASCON 's 5-bit S-box $\mathcal{S}$ as a lookup table

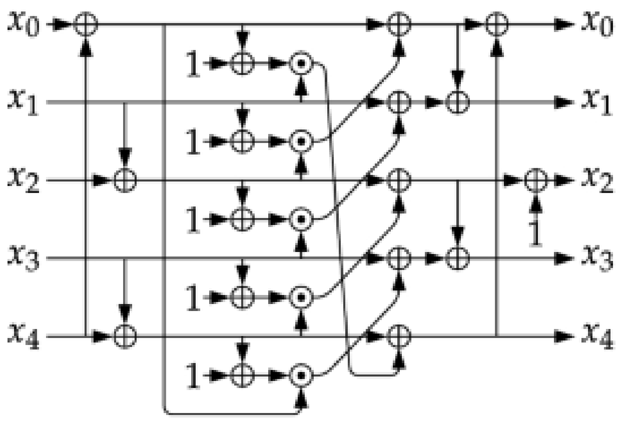
From: ASCON v1.2: Lightweight Authenticated Encryption and Hashing

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$\mathcal{S}(x)$	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

Linear Diffusion Layer The linear diffusion layer p_L provides diffusion within each 64-bit register word x_i (Fig. 3c). It applies a linear function $\Sigma_i(x_i)$ defined in Fig. 4b to each word x_i:

$$x_i \leftarrow \Sigma_i(x_i), \quad 0 \le i \le 4.$$

Fig. 4



(a) Ascon's 5-bit S-box $\mathcal{S}(x)$

$$\begin{aligned}
x_0 &\leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)\\
x_1 &\leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)\\
x_2 &\leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)\\
x_3 &\leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)\\
x_4 &\leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)
\end{aligned}$$

(b) Ascon's linear layer with 64-bit functions $\Sigma_i(x_i)$

ASCON 's substitution layer and linear diffusion layer

Referensi

Materi di dalam PPT ini diambil dari:

1. Ascon v1.2: Lightweight Authenticated Encryption and Hashing, https://link.springer.com/article/10.1007/s00145-021-09398-9
2. Mochammad Fatchur Rochman, Implementasi Kriptografi Hibrida ASCON AEAD dan ECDH Pada Sistem Pemantauan Elektrokardiogram Jarak Jauh, Tugas Akhir Informatika ITB
3. Meltem Sonmez Turannist, Lightweight Cryptography Standardization, GLOBAL PLATFORM SECURITY TASK FORCE MEETING, MARCH 28, 2023, NIST
4. Authenticated encryption, https://en.wikipedia.org/wiki/Authenticated_encryption#Authenticated_encryption_with_associated_data
5. OpenAI