Applied Cryptology

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
⟨csdsp@bristol.ac.uk⟩

April 24, 2024

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
   - the associated notes page may be pre-populated with extra, written explaination of material covered in lecture(s), plus
   - anything with a "grey'ed out" header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

- Agenda: explore **implementation attacks** via
  1. an "in theory", i.e., concept-oriented perspective,
     - 1.1 explanation,
     - 1.2 justification,
     - 1.3 formalisation.

     and
  2. an "in practice", i.e., example-oriented perspective,
     - 2.1 attacks,
     - 2.2 countermeasures.
- Caveat!

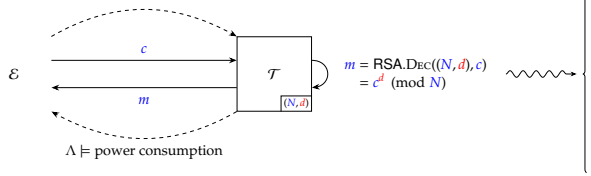
  ~ 2 hours ⇒ introductory, and (very) selective (versus definitive) coverage.

Notes:

Part 2.1: in practice (1)

Attacks: $\mathcal{T} \simeq$ RSA, Λ = power consumption

- Scenario:
  - given the following interaction between an **attacker** $\mathcal{E}$ and a **target** $\mathcal{T}$



$m = \text{RSA.Dec}((N,d),c)$
$= c^d \pmod N$

```
1  algorithm RSA.Dec((N, d), c) begin
2      Pre-compute Π = (N, ρ, ω) from N
3      ĉ ← MontMul(Π, c, ρ^2 mod N)
4      r̂ ← MontMul(Π, c, ρ^2 mod N)
5      for i = |d| − 2 downto 0 do
6          r̂ ← MontSqr(Π, r̂)
7          if d_i = 1 then
8              r̂ ← MontMul(Π, r̂, ĉ)
9          end
10     end
11     return MontMul(Π, r̂, 1)
12 end
```

  - and noting that
    - there are no countermeasures implemented,
    - a dedicated Montgomery squaring implementation, i.e.,

$$\text{MontSqr}(\Pi, \hat{r}) = \hat{r} \times \hat{r} \times \rho^{-1} \pmod N,$$

      is used, such that although

$$\text{MontSqr}(\Pi, \hat{r}) = \text{MontMul}(\Pi, \hat{r}, \hat{r}),$$

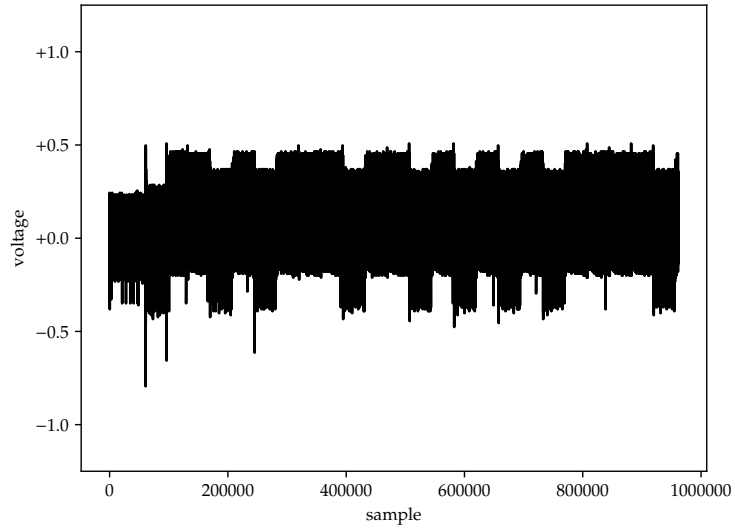
      the former is more efficient than the latter,
    - the Montgomery squaring and Montgomery multiplication implementations are FIOS-based [10],
  - how can $\mathcal{E}$ mount a successful attack, i.e., recover d ?

Notes:

Part 2.1: in practice (2)

Attacks: $\mathcal{T} \simeq$ RSA, $\Lambda =$ power consumption

- Example: ARM Cortex-M3, $|N| = 1024$.

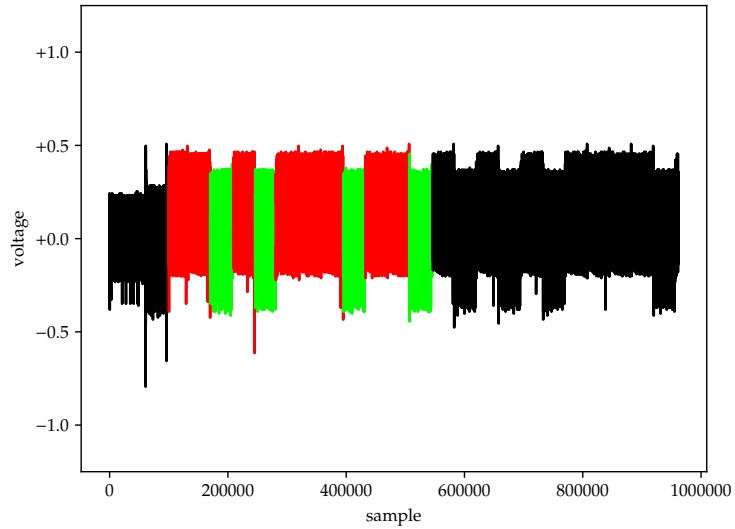


Notes:

Part 2.1: in practice (2)

Attacks: $\mathcal{T} \simeq$ RSA, $\Lambda =$ power consumption

- Attack [11]:



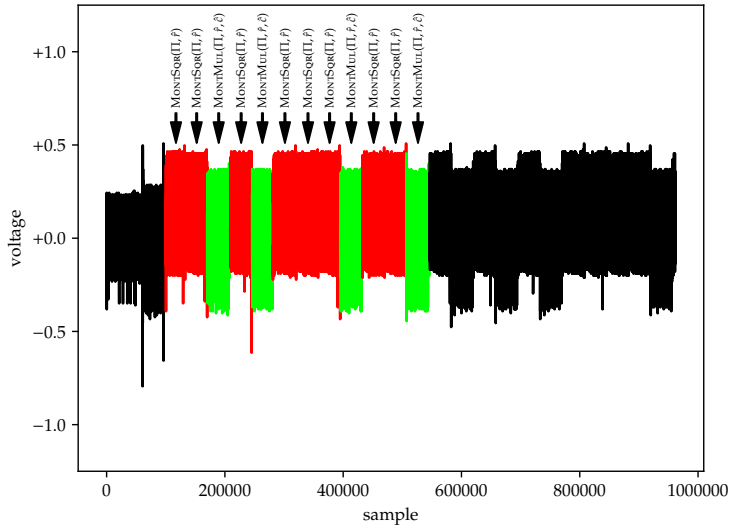
Notes:

© Daniel Page ⟨...⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Part 2.1: in practice (2)

Attacks: $\mathcal{T} \simeq$ RSA, Λ = power consumption

- Attack [11]:

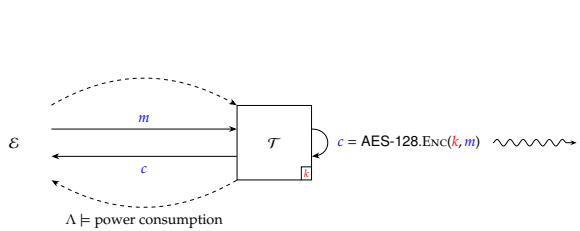


Notes:

Part 2.1: in practice (3)

Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Scenario:
  - given the following interaction between an **attacker** $\mathcal{E}$ and a **target** $\mathcal{T}$




```
1  algorithm AES-128.Enc(k, m) begin
2  |  s ← m
3  |  s ← AddRoundKey(s, k = rk^(0))
4  |  for r = 1 upto 9 step +1 do
5  |  |  k ← EvolveRoundKey(k, rc^(r))
6  |  |  s ← SubBytes(s)
7  |  |  s ← ShiftRows(s)
8  |  |  s ← MixColumns(s)
9  |  |  s ← AddRoundKey(s, k = rk^(r))
10 |  end
11 |  k ← EvolveRoundKey(k, rc^(10))
12 |  s ← SubBytes(s)
13 |  s ← ShiftRows(s)
14 |  s ← AddRoundKey(s, k = rk^(10))
15 |  c ← s
16 |  return c
17 end
```

  - and noting that
    - there are no countermeasures implemented,
    - in the first round, the implementation computes

$$\text{S-box}(m_t \oplus k_t)$$

    - for $0 \leq t < 16$: we can target this operation, and so recover each t-th byte independently,
    - power consumption can be modelled by Hamming weight, i.e.,

$$\text{HW}(\text{S-box}(m_t \oplus k_t))$$

    - models the power consumption of the target operation (or result of it),
  - how can $\mathcal{E}$ mount a successful attack, i.e., recover k ?

Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Attack [7]:

$$\begin{array}{ll} \mathcal{M}_0 & \text{AES-128.Enc}(k, \mathcal{M}_0) \\ \mathcal{M}_1 & \text{AES-128.Enc}(k, \mathcal{M}_1) \\ \vdots & \\ \mathcal{M}_{n-1} & \text{AES-128.Enc}(k, \mathcal{M}_{n-1}) \end{array} \begin{pmatrix} \mathcal{R}_{0,0} & \mathcal{R}_{0,1} & \cdots & \mathcal{R}_{0,l-1} \\ \mathcal{R}_{1,0} & \mathcal{R}_{1,1} & \cdots & \mathcal{R}_{1,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{R}_{n-1,0} & \mathcal{R}_{n-1,1} & \cdots & \mathcal{R}_{n-1,l-1} \end{pmatrix}$$

Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Attack [7]:

$$\begin{array}{ll} \mathcal{M}_0 & \text{AES-128.Enc}(k, \mathcal{M}_0) \\ \mathcal{M}_1 & \text{AES-128.Enc}(k, \mathcal{M}_1) \\ \vdots & \\ \mathcal{M}_{n-1} & \text{AES-128.Enc}(k, \mathcal{M}_{n-1}) \end{array} \begin{pmatrix} \mathcal{R}_{0,0} & \mathcal{R}_{0,1} & \cdots & \mathcal{R}_{0,l-1} \\ \mathcal{R}_{1,0} & \mathcal{R}_{1,1} & \cdots & \mathcal{R}_{1,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{R}_{n-1,0} & \mathcal{R}_{n-1,1} & \cdots & \mathcal{R}_{n-1,l-1} \end{pmatrix}$$

some $\tilde{j}$ indexes leakage from the target operation

© Daniel Page ⟨...⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

$$\begin{array}{ll} \mathcal{M}_0 & \text{AES-128.Enc}(k, \mathcal{M}_0) \rightsquigarrow \left(\mathcal{R}_{0,0} \;\; \mathcal{R}_{0,1} \;\; \cdots \;\; \mathcal{R}_{0,l-1} \right) \\ \mathcal{M}_1 & \text{AES-128.Enc}(k, \mathcal{M}_1) \rightsquigarrow \left(\mathcal{R}_{1,0} \;\; \mathcal{R}_{1,1} \;\; \cdots \;\; \mathcal{R}_{1,l-1} \right) \\ \vdots & \vdots \\ \mathcal{M}_{n-1} & \text{AES-128.Enc}(k, \mathcal{M}_{n-1}) \rightsquigarrow \left(\mathcal{R}_{n-1,0} \;\; \mathcal{R}_{n-1,1} \;\; \cdots \;\; \mathcal{R}_{n-1,l-1} \right) \end{array}$$

$$\begin{array}{ll} \mathcal{M}_0 & \text{HW}(\text{S-box}(\mathcal{M}_{0,t} \oplus \tilde{k}_t)) \rightsquigarrow \left(\mathcal{H}_{0,0} \;\; \mathcal{H}_{0,1} \;\; \cdots \;\; \mathcal{H}_{0,h-1} \right) \\ \mathcal{M}_1 & \text{HW}(\text{S-box}(\mathcal{M}_{1,t} \oplus \tilde{k}_t)) \rightsquigarrow \left(\mathcal{H}_{1,0} \;\; \mathcal{H}_{1,1} \;\; \cdots \;\; \mathcal{H}_{1,h-1} \right) \\ \vdots & \vdots \\ \mathcal{M}_{n-1} & \text{HW}(\text{S-box}(\mathcal{M}_{n-1,t} \oplus \tilde{k}_t)) \rightsquigarrow \left(\mathcal{H}_{n-1,0} \;\; \mathcal{H}_{n-1,1} \;\; \cdots \;\; \mathcal{H}_{n-1,h-1} \right) \end{array}$$

$$\underbrace{\tilde{k}_t = 0 \;\; \tilde{k}_t = 1 \;\; \cdots \;\; \tilde{k}_t = h-1}_{h = 2^8 = 256 \text{ hypothetical values for } k_t}$$

Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

$$\begin{array}{ll} \mathcal{M}_0 & \text{AES-128.Enc}(k, \mathcal{M}_0) \rightsquigarrow \left(\mathcal{R}_{0,0} \;\; \mathcal{R}_{0,1} \;\; \cdots \;\; \mathcal{R}_{0,l-1} \right) \\ \mathcal{M}_1 & \text{AES-128.Enc}(k, \mathcal{M}_1) \rightsquigarrow \left(\mathcal{R}_{1,0} \;\; \mathcal{R}_{1,1} \;\; \cdots \;\; \mathcal{R}_{1,l-1} \right) \\ \vdots & \vdots \\ \mathcal{M}_{n-1} & \text{AES-128.Enc}(k, \mathcal{M}_{n-1}) \rightsquigarrow \left(\mathcal{R}_{n-1,0} \;\; \mathcal{R}_{n-1,1} \;\; \cdots \;\; \mathcal{R}_{n-1,l-1} \right) \end{array}$$

$$\begin{array}{ll} \mathcal{M}_0 & \text{HW}(\text{S-box}(\mathcal{M}_{0,t} \oplus \tilde{k}_t)) \rightsquigarrow \left(\mathcal{H}_{0,0} \;\; \mathcal{H}_{0,1} \;\; \cdots \;\; \mathcal{H}_{0,h-1} \right) \\ \mathcal{M}_1 & \text{HW}(\text{S-box}(\mathcal{M}_{1,t} \oplus \tilde{k}_t)) \rightsquigarrow \left(\mathcal{H}_{1,0} \;\; \mathcal{H}_{1,1} \;\; \cdots \;\; \mathcal{H}_{1,h-1} \right) \\ \vdots & \vdots \\ \mathcal{M}_{n-1} & \text{HW}(\text{S-box}(\mathcal{M}_{n-1,t} \oplus \tilde{k}_t)) \rightsquigarrow \left(\mathcal{H}_{n-1,0} \;\; \mathcal{H}_{n-1,1} \;\; \cdots \;\; \mathcal{H}_{n-1,h-1} \right) \end{array}$$

$$\underbrace{\tilde{k}_t = 0 \;\; \tilde{k}_t = 1 \;\; \cdots \;\; \tilde{k}_t = h-1}_{h = 2^8 = 256 \text{ hypothetical values for } k_t}$$

some $\tilde{k}_t = k_t$, i.e., is correct

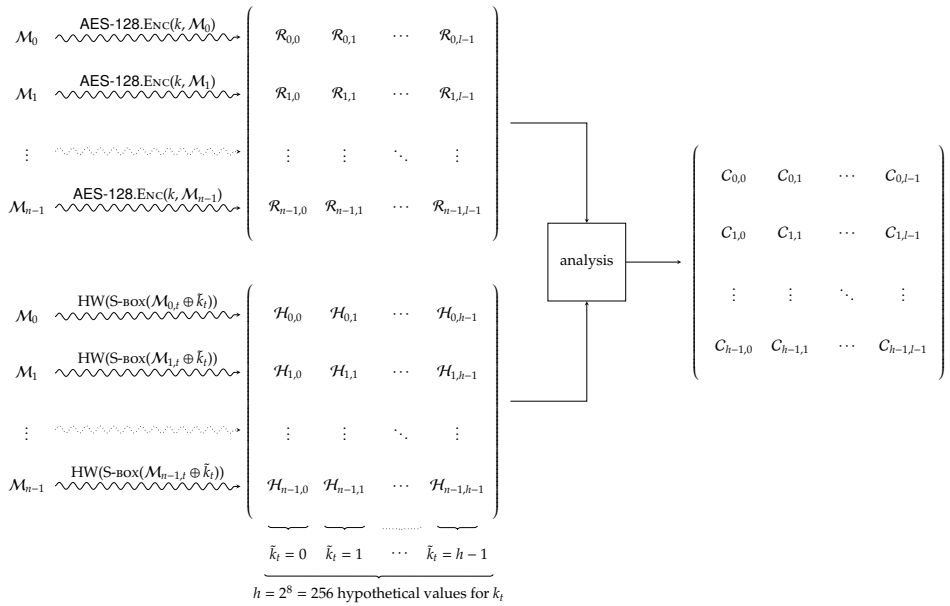
© Daniel Page ⟨...⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

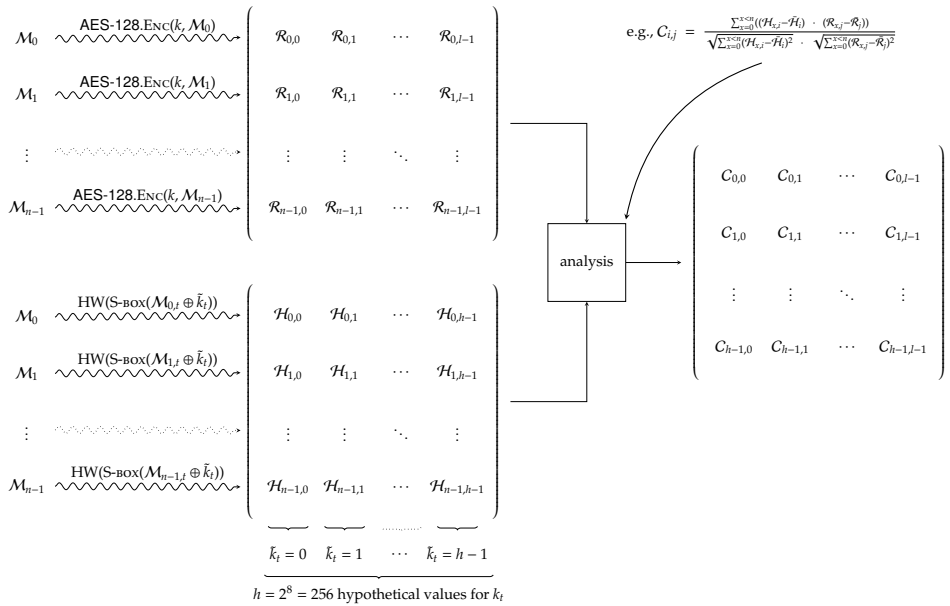


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:



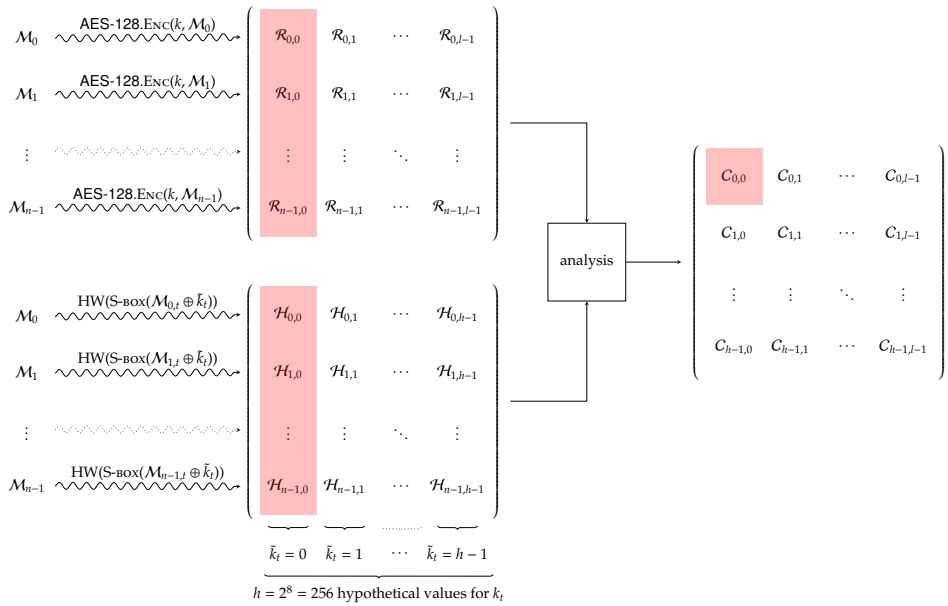
© Daniel Page ⟨dan@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

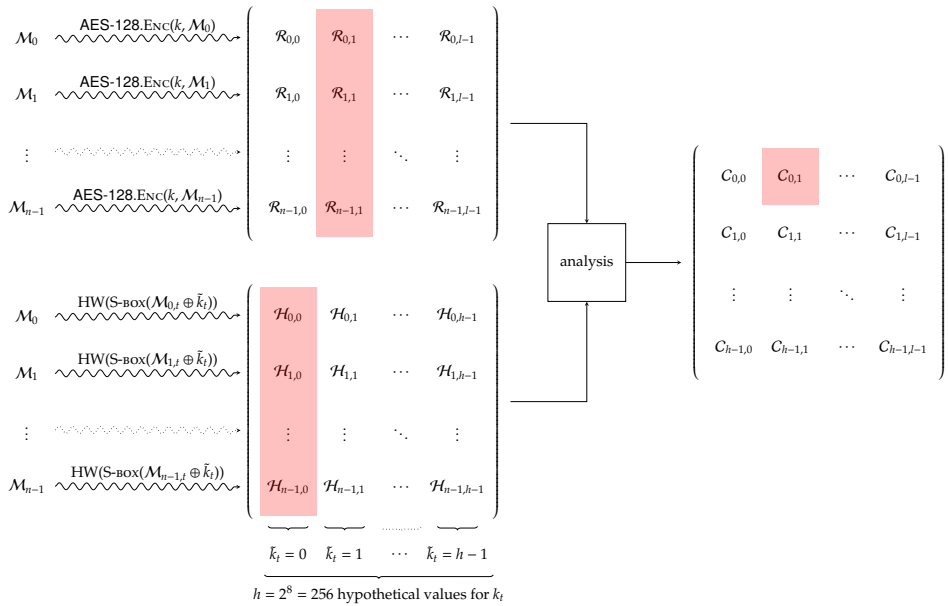


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

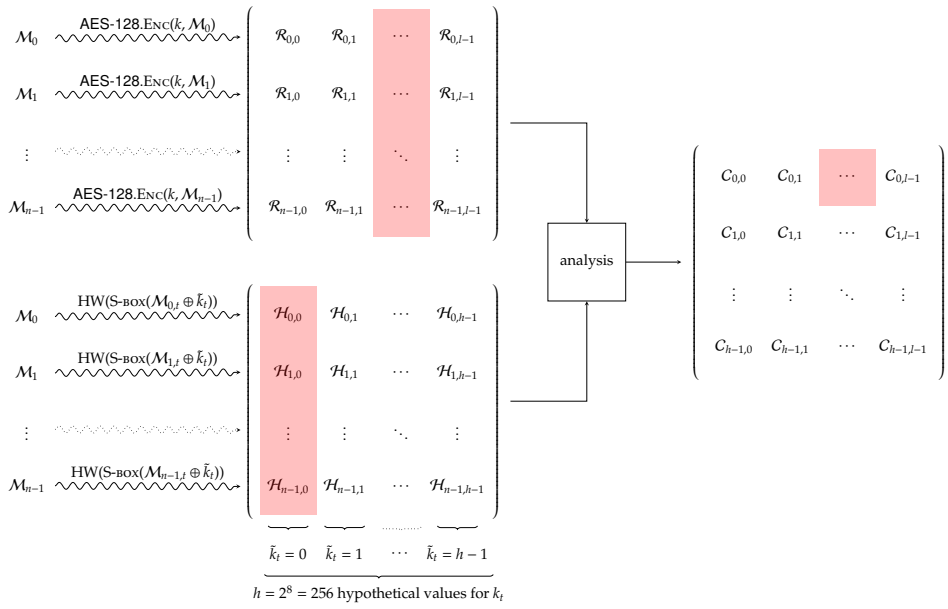


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

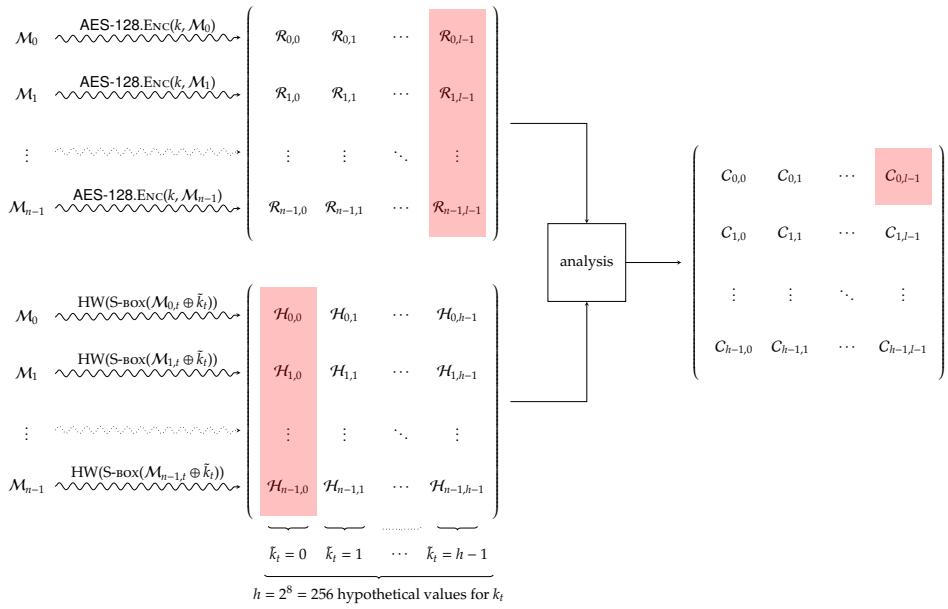


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

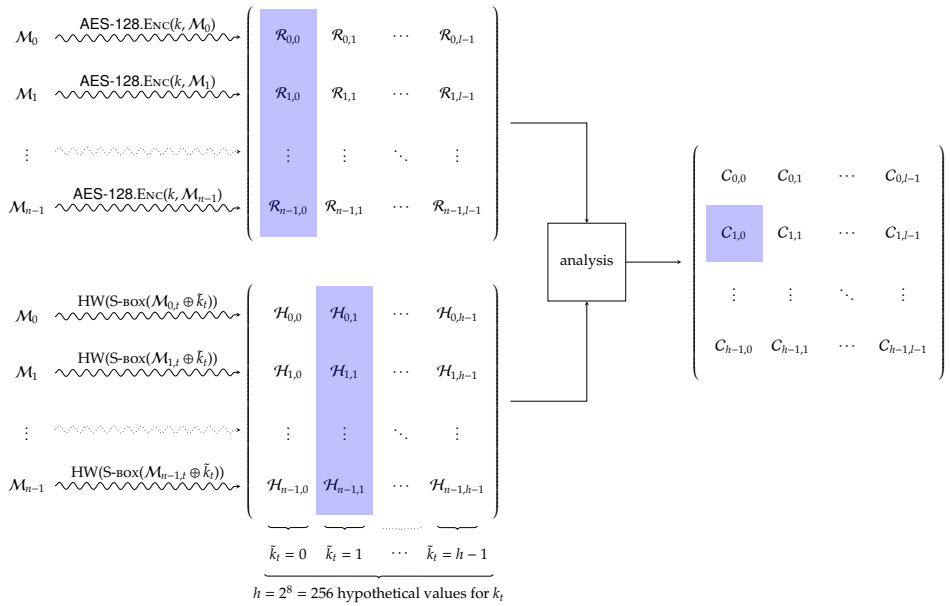


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

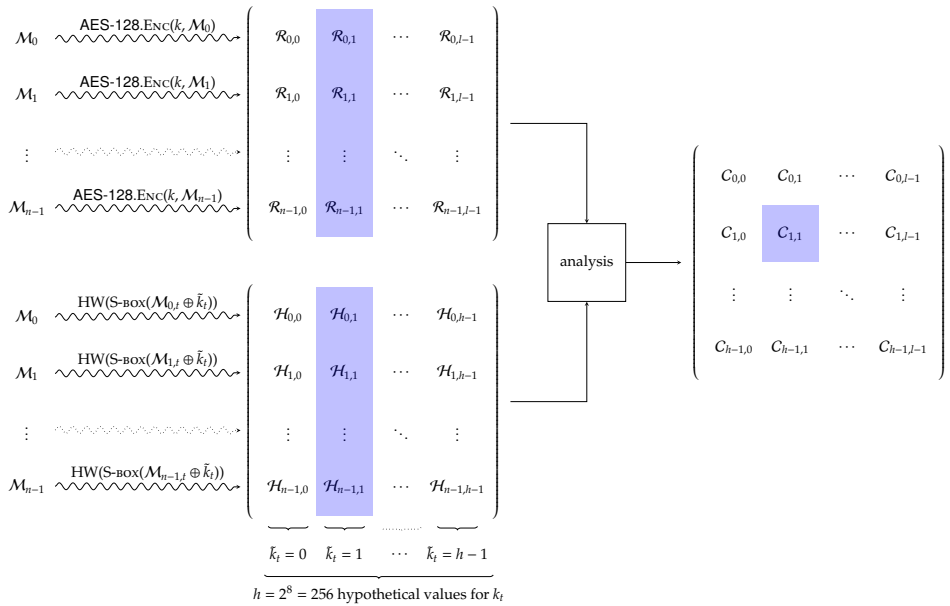


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

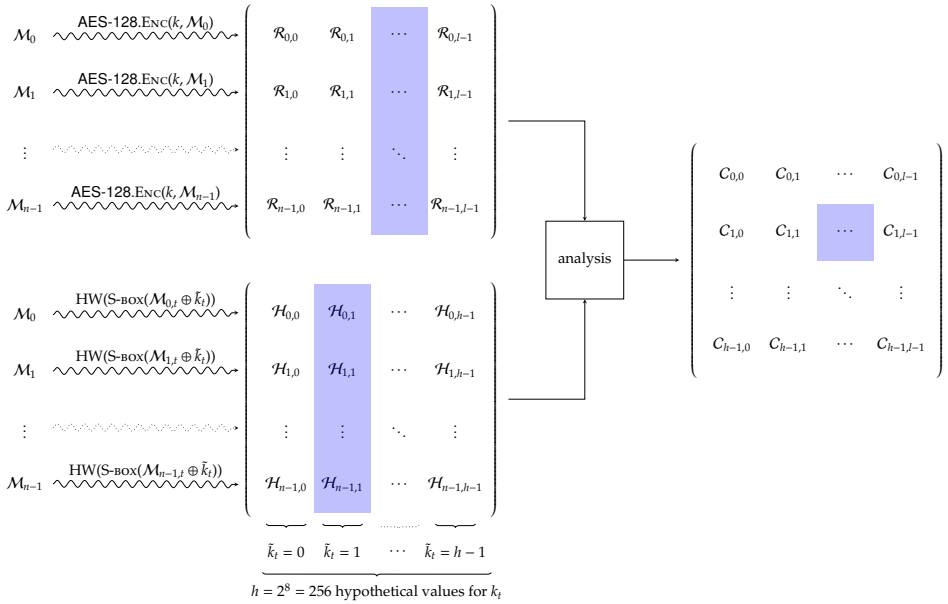


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

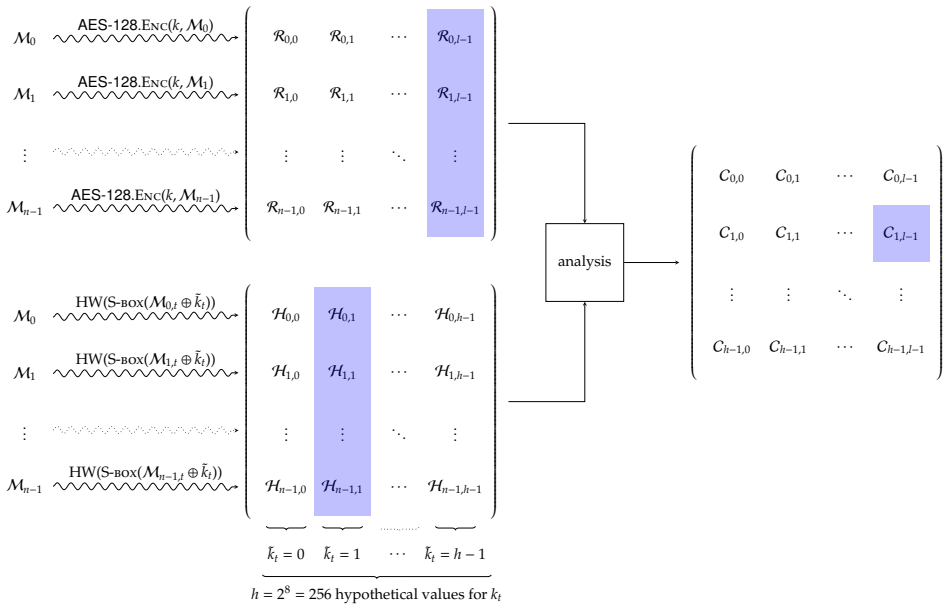


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

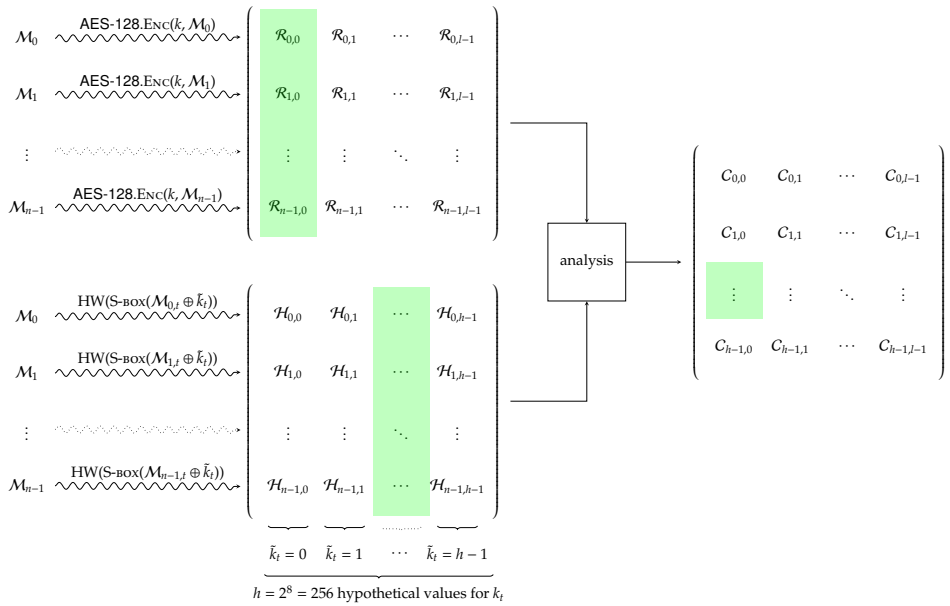


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

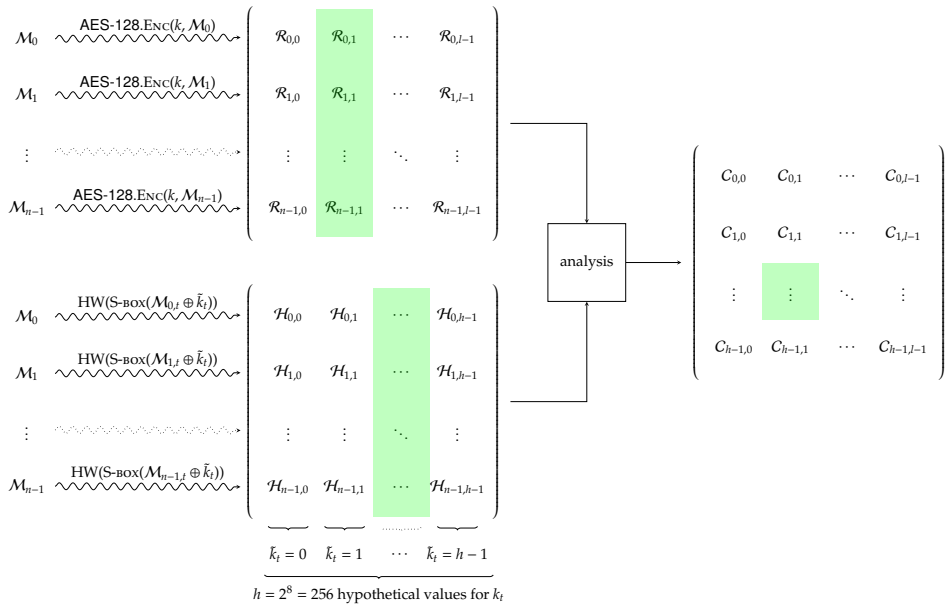


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

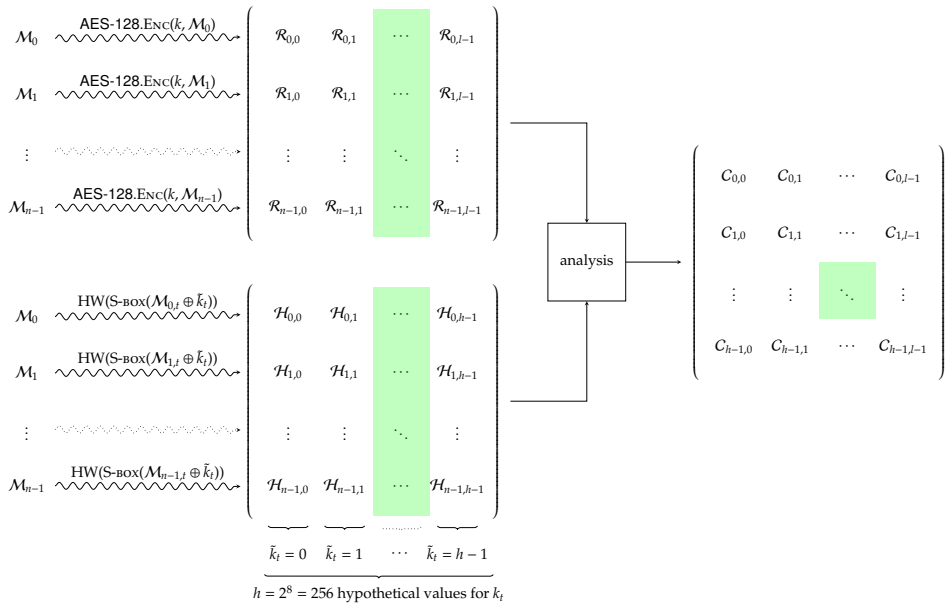


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

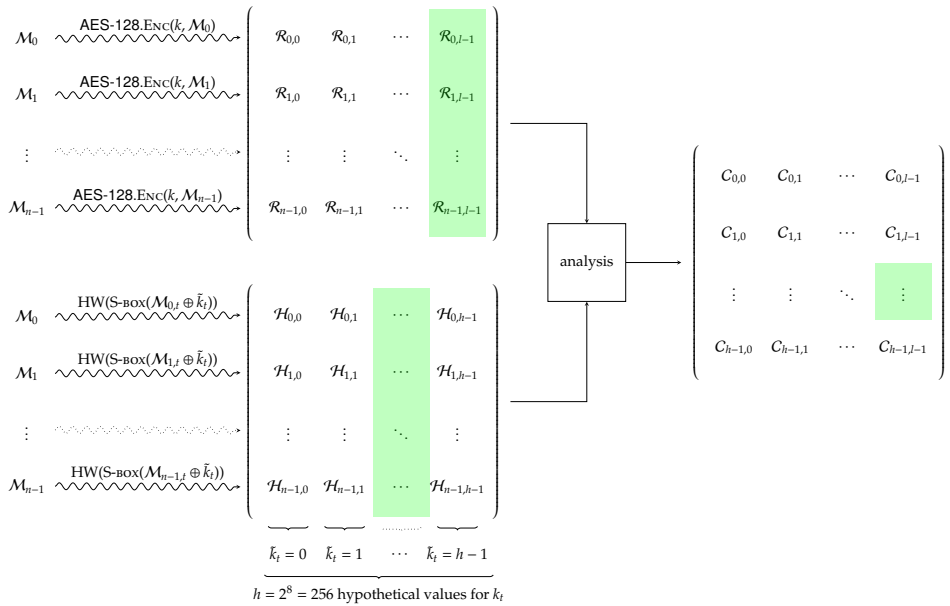


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

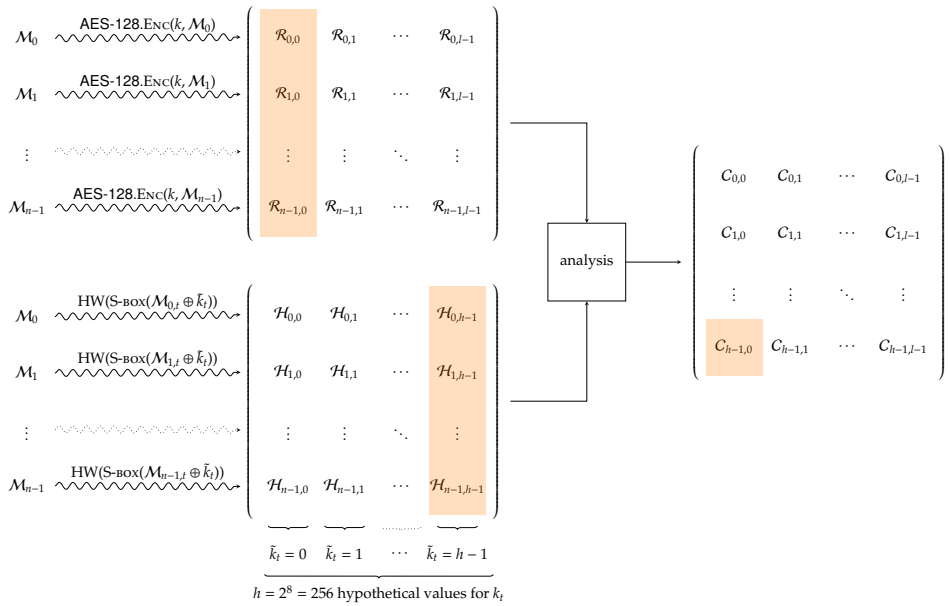


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

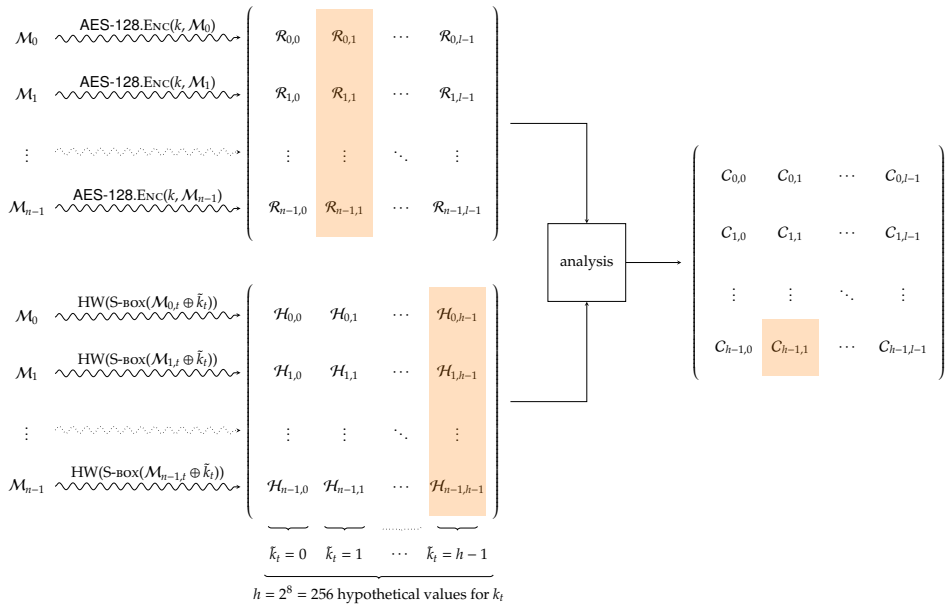


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

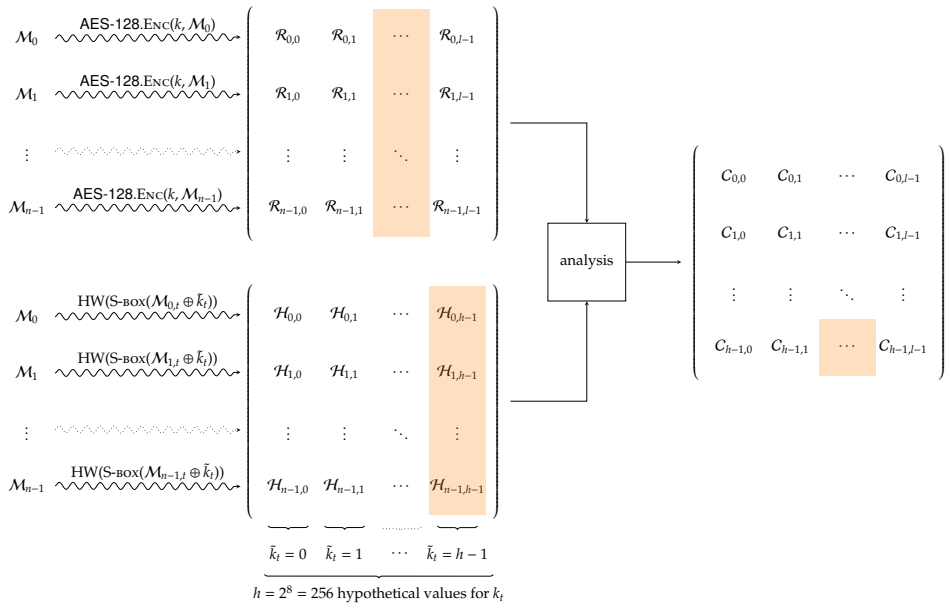


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

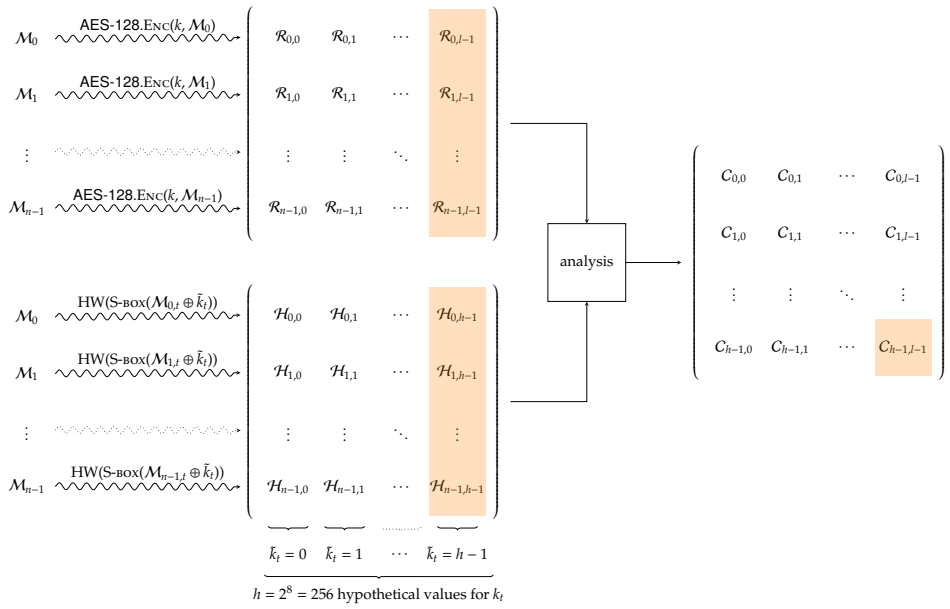


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

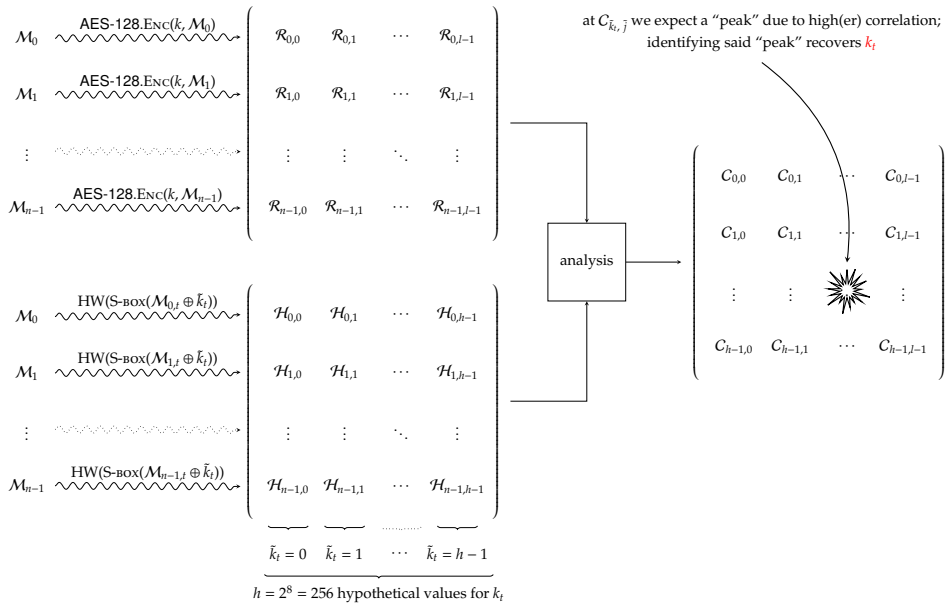


Notes:

Part 2.1: in practice (4)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Attack [7]:

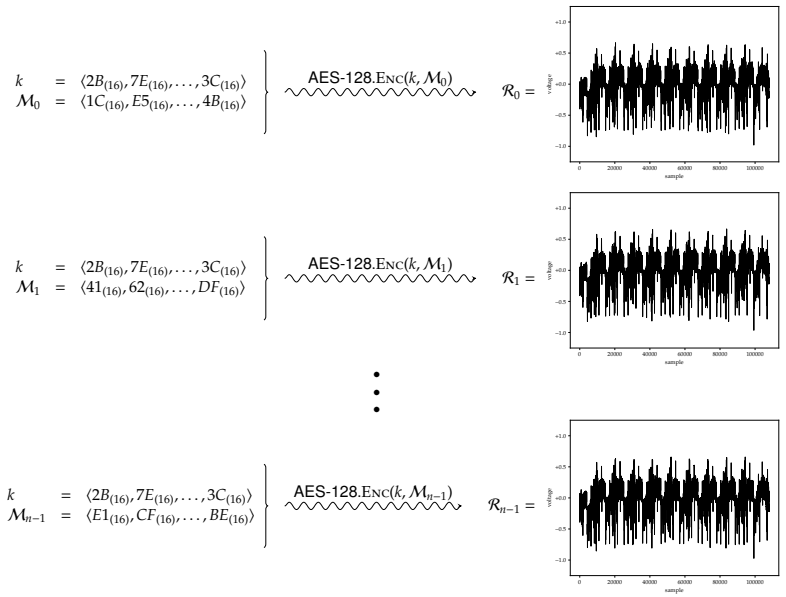


Notes:

Part 2.1: in practice (5)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Example: ARM Cortex-M3, $n = 1000$, $l = 108124$.

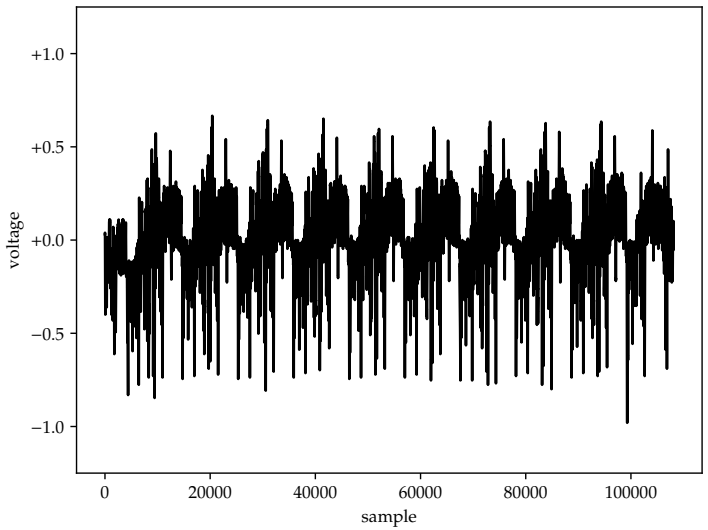


Notes:

Part 2.1: in practice (5)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Example: ARM Cortex-M3, $n = 1000$, $l = 108124$.

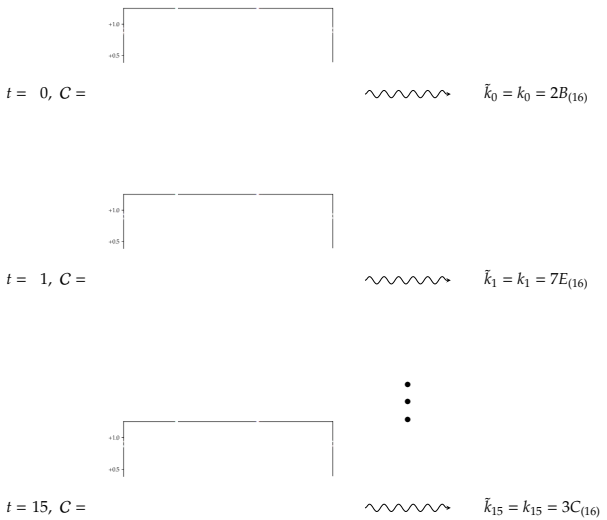


Notes:

Part 2.1: in practice (5)

Attacks: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Example: ARM Cortex-M3, $n = 1000$, $l = 108124$.



Notes:

© Daniel Page ⟨...⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

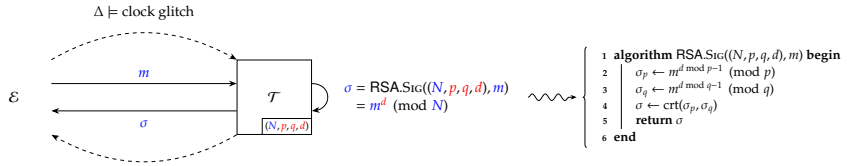
Part 2.1: in practice (5)
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Example: ARM Cortex-M3, $n = 1000$, $l = 108124$.

Notes:

Part 2.1: in practice (6)
Attacks: $\mathcal{T} \simeq$ RSA, Δ = clock glitch

- Scenario:
  - given the following interaction between an **attacker** $\mathcal{E}$ and a **target** $\mathcal{T}$




```
1 algorithm RSA.Sig((N, p, q, d), m) begin
2   σ_p ← m^(d mod p−1) (mod p)
3   σ_q ← m^(d mod q−1) (mod q)
4   σ ← crt(σ_p, σ_q)
5   return σ
6 end
```

  - and noting that
    - there are no countermeasures implemented,
    - to improve efficiency, a CRT-based [13] implementation is used,
    - based on pre-computation of

$$\begin{aligned} t_0 &= q^{p-1} \pmod N \\ t_1 &= p^{q-1} \pmod N \end{aligned}$$

      the Gauss-based recombination is such that

$$\sigma = \text{crt}(\sigma_p, \sigma_q) = \sigma_p \times t_0 + \sigma_q \times t_1 \pmod N,$$

    - the fault induced randomises computation of σ_p, i.e., the first "small" exponentiation,
  - how can $\mathcal{E}$ mount a successful attack, i.e., recover d ?

Notes:

Part 2.1: in practice (7)
Attacks: $\mathcal{T} \simeq$ RSA, Δ = clock glitch

- Attack [6, Section 2.2]:
  - generate

$$\begin{array}{lll} \bar{\sigma} = \bar{\sigma}_p \times t_0 + \sigma_q \times t_1 \pmod{N} & \Leftarrow & \text{fault injected} \\ \sigma = \sigma_p \times t_0 + \sigma_q \times t_1 \pmod{N} & \Leftarrow & \text{no fault injected} \end{array}$$

    such that $\bar{\sigma} \neq \sigma$ due to the fault induced in the former,
  - observe that the CRT pre-computation means

$$t_0 \equiv 0 \pmod{q},$$

    i.e., t_0 is divisible by q, and so, likewise,

$$(\sigma_p - \bar{\sigma}_p) \times t_0 \equiv 0 \pmod{q},$$

  - compute

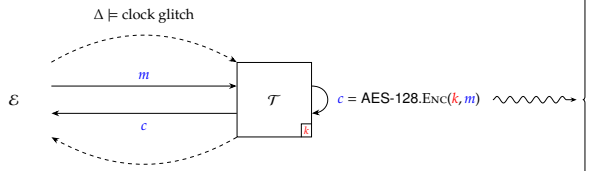
$$\begin{array}{rcl} \gcd(\sigma - \bar{\sigma}, N) & = & \gcd((\sigma_p \times t_0 + \sigma_q \times t_1) - (\bar{\sigma}_p \times t_0 + \sigma_q \times t_1), N) \\ & = & \gcd((\sigma_p \times t_0) - (\bar{\sigma}_p \times t_0), N) \\ & = & \gcd((\sigma_p - \bar{\sigma}_p) \times t_0, N) \\ & = & q \end{array}$$

    i.e., factor N, then compute d.

Notes:

Part 2.1: in practice (8)
Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Scenario:
  - given the following interaction between an **attacker** $\mathcal{E}$ and a **target** $\mathcal{T}$




```
1  algorithm AES-128.Enc(k, m) begin
2      s ← m
3      s ← AddRoundKey(s, k = rk^(0))
4      for r = 1 upto 9 step +1 do
5          k ← EvolveRoundKey(k, rc^(r))
6          s ← SubBytes(s)
7          s ← ShiftRows(s)
8          s ← MixColumns(s)
9          s ← AddRoundKey(s, k = rk^(r))
10     end
11     k ← EvolveRoundKey(k, rc^(10))
12     s ← SubBytes(s)
13     s ← ShiftRows(s)
14     s ← AddRoundKey(s, k = rk^(10))
15     c ← s
16     return c
17 end
```

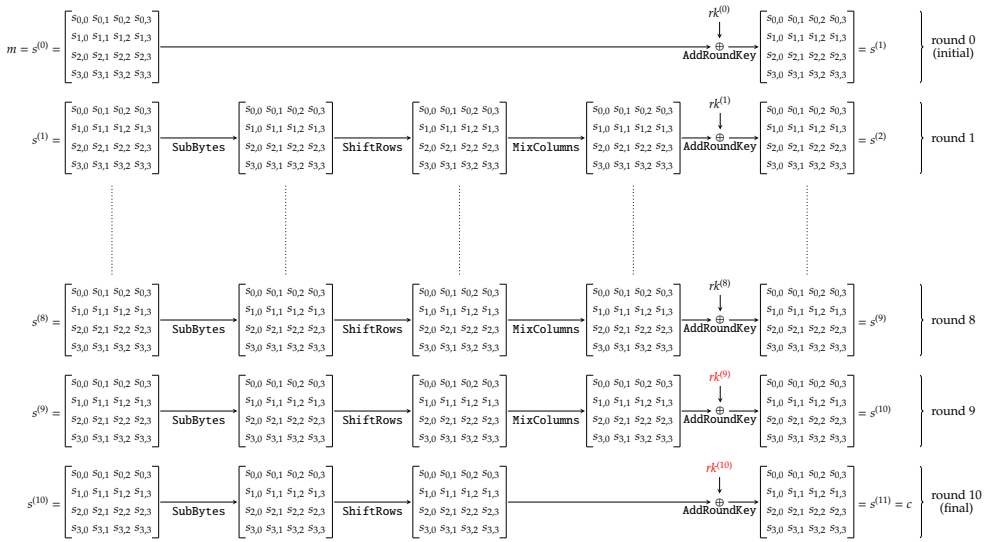
  - and noting that
    - there are no countermeasures implemented,
    - the fault injected randomises $s_{i,j}^{(8)}$, i.e., a chosen element of the state matrix used as input to the 8-th round,
  - how can $\mathcal{E}$ mount a successful attack, i.e., recover k ?

Notes:

Part 2.1: in practice (9)

Attacks: $\mathcal{T} \simeq$ AES, $\Delta =$ clock glitch

- Attack [14]:

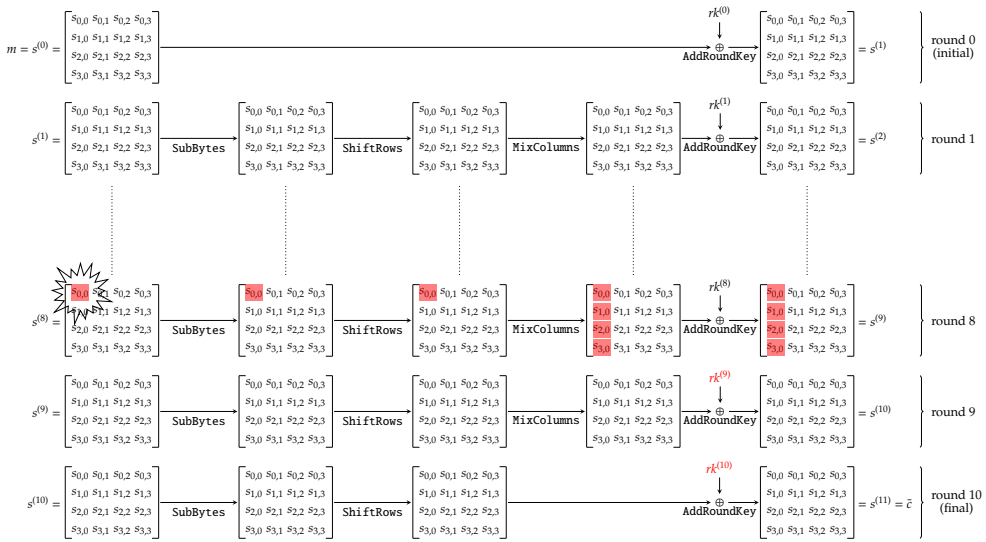


Notes:

Part 2.1: in practice (9)

Attacks: $\mathcal{T} \simeq$ AES, $\Delta =$ clock glitch

- Attack [14]:

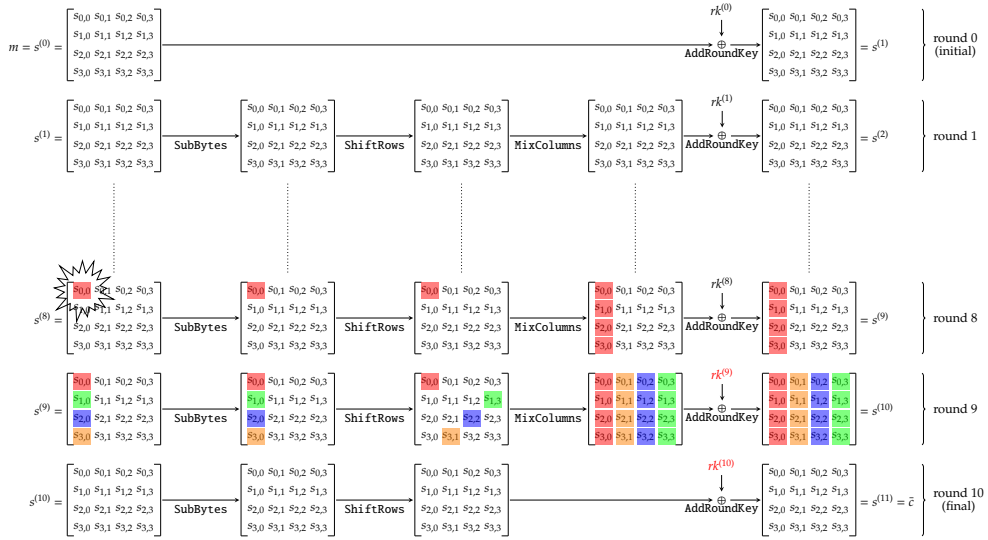


Notes:

Part 2.1: in practice (9)

Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Attack [14]:

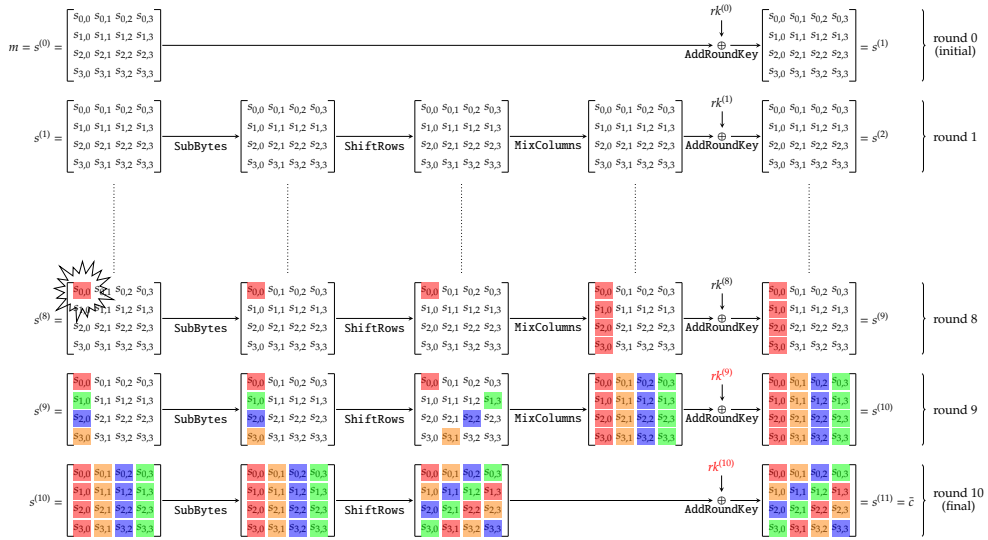


Notes:

Part 2.1: in practice (9)

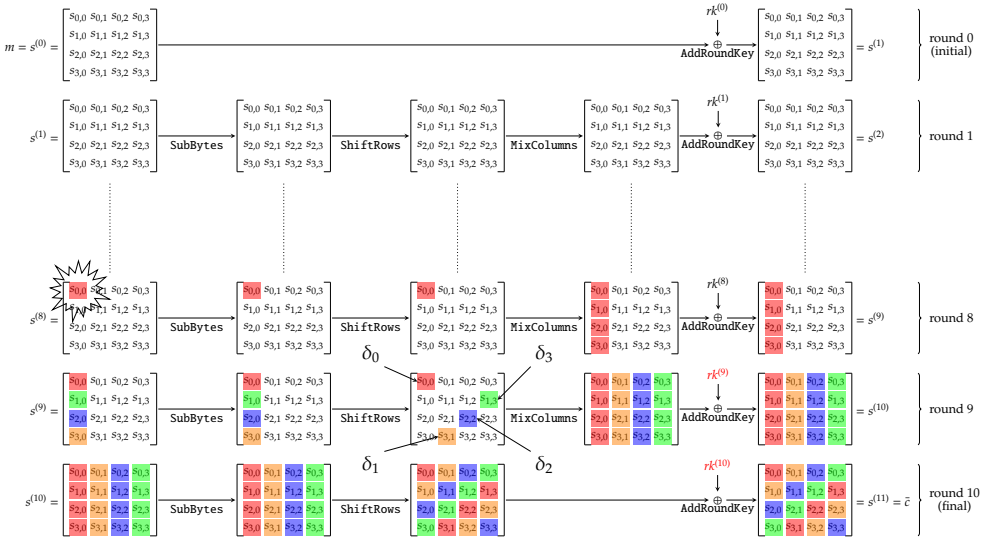
Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Attack [14]:



Notes:

- Attack [14]:



Notes:

Part 2.1: in practice (9)
Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Attack [14]:
  - Step #1:
    - consider a fault injected into the input of the 8-th round, such that

$$\begin{aligned} \bar{c} &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{fault injected} \\ c &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{no fault injected} \end{aligned}$$

      and focus on the state after `ShiftRows` in the 9-th round,
    - formulate a set of constraints, e.g.,

$$\begin{aligned} \texttt{02} \otimes_{\mathbb{F}_{2^8}} \delta_0 &= \text{S-box}^{-1}(c_{0,0} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{0,0}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{0,0} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{0,0}) \\ \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_0 &= \text{S-box}^{-1}(c_{1,3} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{1,3}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{1,3} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{1,3}) \\ \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_0 &= \text{S-box}^{-1}(c_{2,2} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{2,2}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{2,2} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{2,2}) \\ \texttt{03} \otimes_{\mathbb{F}_{2^8}} \delta_0 &= \text{S-box}^{-1}(c_{3,1} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{3,1}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{3,1} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{3,1}) \end{aligned}$$

    - noting that each group involves 32 bits of $rk^{(10)}$, these constraints yield a set of *initial* hypotheses for $rk^{(10)}$.

Notes:

© Daniel Page ⟨dsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Part 2.1: in practice (9)

Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Attack [14]:
  - Step #1:
    - consider a fault injected into the input of the 8-th round, such that

$$\begin{aligned} \bar{c} &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{fault injected} \\ c &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{no fault injected} \end{aligned}$$

    - and focus on the state after `ShiftRows` in the 9-th round,
    - formulate a set of constraints, e.g.,

$$\begin{aligned} \texttt{03} \otimes_{\mathbb{F}_{2^8}} \delta_1 &= \text{S-box}^{-1}(c_{0,1} \oplus_{\mathbb{F}_{2^8}} rk_{0,1}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{0,1} \oplus_{\mathbb{F}_{2^8}} rk_{0,1}^{(10)}) \\ \texttt{02} \otimes_{\mathbb{F}_{2^8}} \delta_1 &= \text{S-box}^{-1}(c_{1,0} \oplus_{\mathbb{F}_{2^8}} rk_{1,0}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{1,0} \oplus_{\mathbb{F}_{2^8}} rk_{1,0}^{(10)}) \\ \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_1 &= \text{S-box}^{-1}(c_{2,3} \oplus_{\mathbb{F}_{2^8}} rk_{2,3}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{2,3} \oplus_{\mathbb{F}_{2^8}} rk_{2,3}^{(10)}) \\ \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_1 &= \text{S-box}^{-1}(c_{3,2} \oplus_{\mathbb{F}_{2^8}} rk_{3,2}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{3,2} \oplus_{\mathbb{F}_{2^8}} rk_{3,2}^{(10)}) \end{aligned}$$

    - noting that each group involves 32 bits of $rk^{(10)}$, these constraints yield a set of *initial* hypotheses for $rk^{(10)}$.

Notes:

Part 2.1: in practice (9)

Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Attack [14]:
  - Step #1:
    - consider a fault injected into the input of the 8-th round, such that

$$\begin{aligned} \bar{c} &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{fault injected} \\ c &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{no fault injected} \end{aligned}$$

    - and focus on the state after `ShiftRows` in the 9-th round,
    - formulate a set of constraints, e.g.,

$$\begin{aligned} \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_2 &= \text{S-box}^{-1}(c_{0,2} \oplus_{\mathbb{F}_{2^8}} rk_{0,2}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{0,2} \oplus_{\mathbb{F}_{2^8}} rk_{0,2}^{(10)}) \\ \texttt{03} \otimes_{\mathbb{F}_{2^8}} \delta_2 &= \text{S-box}^{-1}(c_{1,1} \oplus_{\mathbb{F}_{2^8}} rk_{1,1}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{1,1} \oplus_{\mathbb{F}_{2^8}} rk_{1,1}^{(10)}) \\ \texttt{02} \otimes_{\mathbb{F}_{2^8}} \delta_2 &= \text{S-box}^{-1}(c_{2,0} \oplus_{\mathbb{F}_{2^8}} rk_{2,0}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{2,0} \oplus_{\mathbb{F}_{2^8}} rk_{2,0}^{(10)}) \\ \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_2 &= \text{S-box}^{-1}(c_{3,3} \oplus_{\mathbb{F}_{2^8}} rk_{3,3}^{(10)}) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\bar{c}_{3,3} \oplus_{\mathbb{F}_{2^8}} rk_{3,3}^{(10)}) \end{aligned}$$

    - noting that each group involves 32 bits of $rk^{(10)}$, these constraints yield a set of *initial* hypotheses for $rk^{(10)}$.

Notes:

© Daniel Page ⟨dan@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Part 2.1: in practice (9)

Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Attack [14]:
  - Step #1:
    - consider a fault injected into the input of the 8-th round, such that

$$\begin{aligned} \bar{c} &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{fault injected} \\ c &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{no fault injected} \end{aligned}$$

    - and focus on the state after `ShiftRows` in the 9-th round,
    - formulate a set of constraints, e.g.,

$$\begin{aligned} \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{0,3} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{0,3}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{0,3} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{0,3}\,) \\ \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{1,2} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{1,2}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{1,2} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{1,2}\,) \\ \texttt{03} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{2,1} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{2,1}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{2,1} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{2,1}\,) \\ \texttt{02} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{3,0} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{3,0}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{3,0} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{3,0}\,) \end{aligned}$$

    - noting that each group involves 32 bits of $rk^{(10)}$, these constraints yield a set of *initial* hypotheses for $rk^{(10)}$.

Notes:

Part 2.1: in practice (9)

Attacks: $\mathcal{T} \simeq$ AES, Δ = clock glitch

- Attack [14]:
  - Step #1:
    - consider a fault injected into the input of the 8-th round, such that

$$\begin{aligned} \bar{c} &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{fault injected} \\ c &= \text{AES-128.Enc}(k, m) \quad \Leftarrow \quad \text{no fault injected} \end{aligned}$$

    - and focus on the state after `ShiftRows` in the 9-th round,
    - formulate a set of constraints, e.g.,

$$\begin{aligned} \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{0,3} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{0,3}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{0,3} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{0,3}\,) \\ \texttt{01} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{1,2} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{1,2}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{1,2} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{1,2}\,) \\ \texttt{03} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{2,1} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{2,1}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{2,1} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{2,1}\,) \\ \texttt{02} \otimes_{\mathbb{F}_{2^8}} \delta_3 &= \text{S-box}^{-1}(\, c_{3,0} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{3,0}\,) \oplus_{\mathbb{F}_{2^8}} \text{S-box}^{-1}(\, \bar{c}_{3,0} \oplus_{\mathbb{F}_{2^8}} rk^{(10)}_{3,0}\,) \end{aligned}$$

    - noting that each group involves 32 bits of $rk^{(10)}$, these constraints yield a set of *initial* hypotheses for $rk^{(10)}$.
  - Step #2: *either*
    1. repeat step #1: successive faults, and so constraints, act to reduce the set of *initial* hypotheses,
    2. perform brute-force search of $\sim 2^{32}$ *initial* hypotheses,
    3. perform brute-force search of $\sim 2^{8}$ *filtered* hypotheses, produced by considering the relationship between $rk^{(9)}$ and $rk^{(10)}$ that stems from the key schedule.

Notes:

- Problem: data-dependent computation within

$$\texttt{xtime}(x) = \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x$$

  is observable via Λ.
- Idea: **hiding**, e.g., via
  1. original (i.e., no countermeasure):

$$\texttt{xtime}(x) \quad \mapsto$$

```
uint8_t aes_gf28_mulx( uint8_t x ) {
  if( ( x & 0x80 ) == 0x80 ) {
    return 0x1B ^ ( x << 1 );
  }
  else {
    return        ( x << 1 );
  }
}
```


Notes:

Part 2.2: in practice (1)
Countermeasures: $\mathcal{T} \simeq$ AES, Λ = execution time

- Problem: data-dependent computation within

$$\texttt{xtime}(x) = \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x$$

  is observable via Λ.
- Idea: **hiding**, e.g., via
  1. original (i.e., no countermeasure):

$$\texttt{xtime}(x) \quad \mapsto$$

```
uint8_t aes_gf28_mulx( uint8_t x ) {
  uint8_t c;

  c = x >> 7;
  x = x << 1;

  if( c ) {
    x = x ^ 0x1B;
  }

  return x;
}
```


Notes:

Part 2.2: in practice (1)

Countermeasures: $\mathcal{T} \simeq$ AES, $\Lambda =$ execution time

- Problem: data-dependent computation within

$$\texttt{xtime}(x) = \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x$$

is observable via Λ.

- Idea: **hiding**, e.g., via

1. balanced (via dummy XOR):

$$\texttt{xtime}(x) \quad \mapsto$$

```
1  uint8_t aes_gf28_mulx( uint8_t x ) {
2    uint8_t c;
3
4    c = x >> 7;
5    x = x << 1;
6
7    if( c ) {
8      x = x ^ 0x1B;
9    }
10   else {
11     x = x ^ 0x00;
12   }
13
14   return x;
15 }
```

although this assumes no, e.g., compiler-based strength reduction.

Notes:

Part 2.2: in practice (1)

Countermeasures: $\mathcal{T} \simeq$ AES, $\Lambda =$ execution time

- Problem: data-dependent computation within

$$\texttt{xtime}(x) = \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x$$

is observable via Λ.

- Idea: **hiding**, e.g., via

2. straight-line (via multiplexer):

$$\texttt{xtime}(x) \quad \mapsto$$

```
1  uint8_t aes_gf28_mulx( uint8_t x ) {
2    uint8_t c, t_0, t_1;
3
4    c = x >> 7;
5    x = x << 1;
6
7    t_0 = x ^ 0x00;
8    t_1 = x ^ 0x1B;
9
10   x = ( -c & ( t_0 ^ t_1 ) ) ^ t_0;
11
12   return x;
13 }
```


Notes:

Part 2.2: in practice (1)

Countermeasures: $\mathcal{T} \simeq$ AES, Λ = execution time

- Problem: data-dependent computation within

$$\texttt{xtime}(x) = \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x$$

is observable via Λ.

- Idea: **hiding**, e.g., via

3. straight-line (via multiplexer):

$$\texttt{xtime}(x) \quad \mapsto$$

```
uint8_t aes_gf28_mulx( uint8_t x ) {
  uint8_t c;

    c = x >> 7;
    x = x << 1;

    x = x ^ ( c * 0x1B );

  return x;
}
```

although this assumes data-oblivious, i.e., constant-latency, multiplication.

Notes:

Part 2.2: in practice (1)

Countermeasures: $\mathcal{T} \simeq$ AES, Λ = execution time

- Problem: data-dependent computation within

$$\texttt{xtime}(x) = \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x$$

is observable via Λ.

- Idea: **hiding**, e.g., via

4. straight-line (via look-up):

$$\texttt{xtime}(x) \quad \mapsto$$

```
uint8_t aes_gf28_mulx( uint8_t x ) {
  uint8_t c, T[ 2 ];

  c = x >> 7;
  x = x << 1;

  T[ 0 ] = x ^ 0x00;
  T[ 1 ] = x ^ 0x1B;

  x = T[ c ];

  return x;
}
```

although this assumes data-oblivious, i.e., constant-latency, memory access.

Notes:

Part 2.2: in practice (2)

Countermeasures: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Problem: data-dependent computation within

$$\texttt{SubBytes}(s^{(r)})$$

is observable via Λ.

- Idea: **hiding**, e.g., via
  1. original (i.e., no countermeasure):

$$\texttt{SubBytes}(s^{(r)}) \quad \mapsto \quad \left\{ \begin{array}{l} \texttt{void aes\_enc\_rnd\_sub(uint8\_t* s) \{} \\ \texttt{\ \ for(int i = 0; i < 16; i++) \{} \\ \texttt{\ \ \ \ s[i] = aes\_enc\_sbox(s[i]);} \\ \texttt{\ \ \}} \\ \texttt{\}} \end{array} \right.$$

Notes:

Part 2.2: in practice (2)

Countermeasures: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Problem: data-dependent computation within

$$\texttt{SubBytes}(s^{(r)})$$

is observable via Λ.

- Idea: **hiding**, e.g., via
  2. temporal padding $\Rightarrow$ random delay:

$$\texttt{SubBytes}(s^{(r)}) \quad \mapsto \quad \left\{ \begin{array}{l} \texttt{void aes\_enc\_rnd\_sub(uint8\_t* s) \{} \\ \texttt{\ \ delay(prng());} \\ \\ \texttt{\ \ for(int i = 0; i < 16; i++) \{} \\ \texttt{\ \ \ \ s[i] = aes\_enc\_sbox(s[i]);} \\ \texttt{\ \ \}} \\ \texttt{\}} \end{array} \right.$$

Notes:

Part 2.2: in practice (2)

Countermeasures: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Problem: data-dependent computation within

$$\texttt{SubBytes}(s^{(r)})$$

is observable via Λ.

- Idea: **hiding**, e.g., via

3. temporal reordering $\Rightarrow$ random start index:

$$\texttt{SubBytes}(s^{(r)}) \quad \mapsto \quad \left\{ \right.$$

```
1 void aes_enc_rnd_sub( uint8_t* s ) {
2   int r = prng();
3
4   for( int i = 0; i < 16; i++ ) {
5     int j = ( i + r ) % 16;
6
7     s[ j ] = aes_enc_sbox( s[ j ] );
8   }
9 }
```


Notes:

Part 2.2: in practice (2)

Countermeasures: $\mathcal{T} \simeq$ AES, $\Lambda =$ power consumption

- Problem: data-dependent computation within

$$\texttt{SubBytes}(s^{(r)})$$

is observable via Λ.

- Idea: **hiding**, e.g., via

4. temporal reordering $\Rightarrow$ random permutation $\Rightarrow$ lower-quality, lower-overhead:

$$\texttt{SubBytes}(s^{(r)}) \quad \mapsto \quad \left\{ \right.$$

```
1 void aes_enc_rnd_sub( uint8_t* s ) {
2   int r = prng();
3
4   for( int i = 0; i < 16; i++ ) {
5     int j = ( i ^ r ) % 16;
6
7     s[ j ] = aes_enc_sbox( s[ j ] );
8   }
9 }
```


Notes:

Part 2.2: in practice (2)
Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Problem: data-dependent computation within

$$\texttt{SubBytes}(s^{(r)})$$

is observable via Λ.

- Idea: **hiding**, e.g., via

5. temporal reordering $\Rightarrow$ random permutation $\Rightarrow$ higher-quality, higher-overhead:

$\texttt{SubBytes}(s^{(r)}) \quad \mapsto$

```
1  void aes_enc_rnd_sub( uint8_t* s ) {
2    int T[ 16 ];
3
4    for( int i = 15; i >= 0; i-- ) {
5      T[ i ] = i;
6    }
7
8    for( int i = 15; i >= 1; i-- ) {
9      int j = prng() % ( i + 1 );
10
11     uint8_t t      = T[ j ];
12             T[ j ] = T[ i ];
13             T[ i ] = t     ;
14   }
15
16   for( int i = 0; i < 16; i++ ) {
17     int j = T[ i ];
18
19     s[ j ] = aes_enc_sbox( s[ j ] );
20   }
21 }
```


Notes:

Part 2.2: in practice (3)
Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Problem: data-dependent computation within

$$\text{AES-128.}\textsc{Enc}(k, m)$$

is observable via Λ.

- Idea: (e.g., Boolean) **masking**.
  - use a randomised, redundant representation

$$x \quad \mapsto \quad \hat{x} = \langle \hat{x}[0], \hat{x}[1], \ldots, \hat{x}[d] \rangle,$$

i.e., as $d + 1$ statistically independent shares, such that

$$x = \bigoplus_{i=0}^{i \leq d} \hat{x}[i],$$

  - computation of some functionality

$$r = f(x)$$

can be described as three high-level steps:

1. x is masked to yield $\hat{x}$,
2. an alternative but compatible functionality $\hat{r} = \hat{f}(\hat{x})$ is executed, then
3. $\hat{r}$ is unmasked to yield r.

Notes:

- Problem: data-dependent computation within

$$\text{AES-128.}\textsc{Enc}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):
  - Step #1:
    1. generate random input (μ_0, μ_1, μ_2, μ_3, and μ_4) and output (ν_4) masks,
    2. pre-compute output masks

$$\begin{bmatrix} \nu_0 \\ \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \texttt{MixColumn}\left(\begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}\right),$$

    3. pre-compute a masked S-box, i.e., $\text{S-box}_{\mu_4}(x \oplus \mu_4) = \text{S-box}(x) \oplus \nu_4$.

Notes:

Part 2.2: in practice (4)
Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

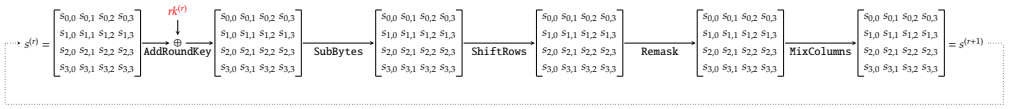
- Problem: data-dependent computation within

$$\text{AES-128.}\textsc{Enc}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):

  - Step #2: construct a masked round implementation



© Daniel Page ⟨...⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Notes:

Part 2.2: in practice (4)

Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

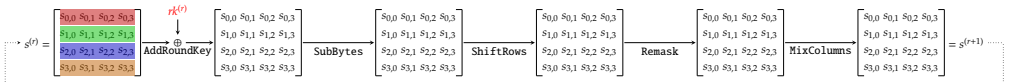
- Problem: data-dependent computation within

$$\text{AES-128.ENC}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):

  - Step #2: construct a masked round implementation



such that $s_{i,j}^{(r)}$ is masked with ν_i.

Notes:

Part 2.2: in practice (4)

Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

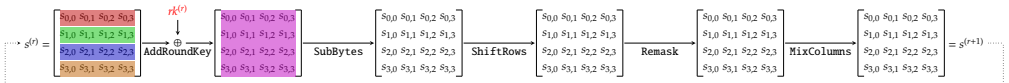
- Problem: data-dependent computation within

$$\text{AES-128.ENC}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):

  - Step #2: construct a masked round implementation



such that $s_{i,j}^{(r)}$ is masked with μ_4 due to alteration of key schedule (and thus $rk^{(r)}$).

Notes:

Part 2.2: in practice (4)

Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

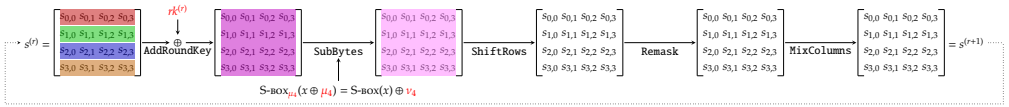
- Problem: data-dependent computation within

$$\text{AES-128.}\textsc{Enc}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):

  - Step #2: construct a masked round implementation



such that $s_{i,j}^{(r)}$ is masked with ν_4 due to alteration of S-box (i.e., use of S-box$_{\mu_4}$).

Notes:

Part 2.2: in practice (4)

Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

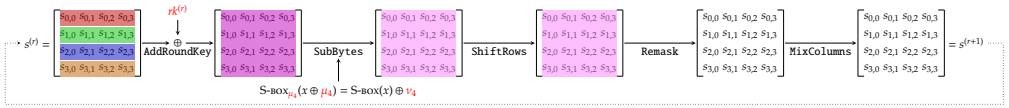
- Problem: data-dependent computation within

$$\text{AES-128.}\textsc{Enc}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):

  - Step #2: construct a masked round implementation



such that $s_{i,j}^{(r)}$ is (still) masked with ν_4.

Notes:

Part 2.2: in practice (4)
Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

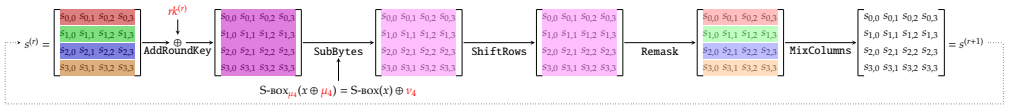
- Problem: data-dependent computation within

$$\text{AES-128.ENC}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):

- Step #2: construct a masked round implementation



such that $s_{i,j}^{(r)}$ is masked with μ_i due to addition of `Remask`.

Notes:

Part 2.2: in practice (4)
Countermeasures: $\mathcal{T} \simeq$ AES, Λ = power consumption

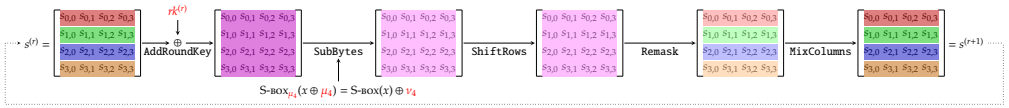
- Problem: data-dependent computation within

$$\text{AES-128.ENC}(k, m)$$

is observable via Λ.

- Idea [8, Section 3.1] (or see [2, Chapter 9]):

- Step #2: construct a masked round implementation



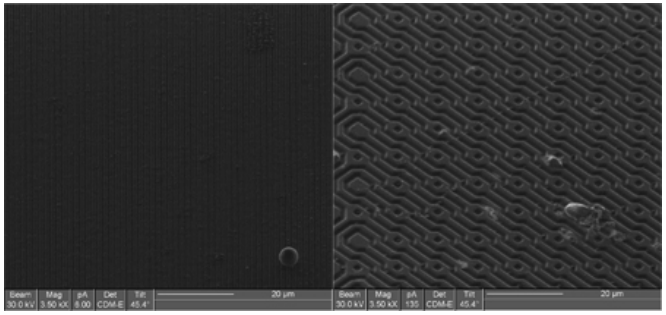
such that $s_{i,j}^{(r)}$ is masked with ν_i due to alteration of `MixColumns`: this makes iteration possible.

Notes:

Part 2.2: in practice (5)

Countermeasures: $\mathcal{T} \simeq$ AES, Δ = laser pulse

- Problem: Δ can be used to influence, e.g., corrupt $s^{(r)}$.
- Idea: prevent physical access to device via a **mesh** (or **shield**).



i.e., a top-layer of metal which can be classed as

1. passive (or analogue), or
2. active (or digital).

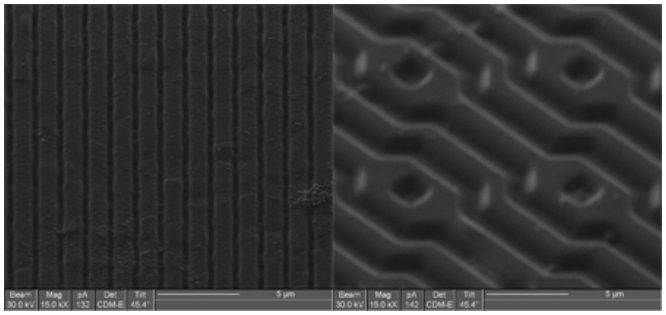
http://www.flylogic.net/blog/?p=86

Notes:

Part 2.2: in practice (5)

Countermeasures: $\mathcal{T} \simeq$ AES, Δ = laser pulse

- Problem: Δ can be used to influence, e.g., corrupt $s^{(r)}$.
- Idea: prevent physical access to device via a **mesh** (or **shield**).



i.e., a top-layer of metal which can be classed as

1. passive (or analogue), or
2. active (or digital).

http://www.flylogic.net/blog/?p=86

© Daniel Page (<redacted>)
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Notes:

- Problem: Δ can be used to influence, e.g., corrupt $s^{(r)}$.
- Idea [5]:
  - for $x \in \mathbb{F}_{2^8}$, let

$$\check{x} \;=\; \textsc{Par}(x) \;=\; \bigoplus_{i=0}^{i<8} x_i$$

  denote the (even) **parity bit** for x,
  - let

$$\check{s}^{(r)} = \begin{bmatrix} \check{s}^{(r)}_{0,0} & \check{s}^{(r)}_{0,1} & \check{s}^{(r)}_{0,2} & \check{s}^{(r)}_{0,3} \\ \check{s}^{(r)}_{1,0} & \check{s}^{(r)}_{1,1} & \check{s}^{(r)}_{1,2} & \check{s}^{(r)}_{1,3} \\ \check{s}^{(r)}_{2,0} & \check{s}^{(r)}_{2,1} & \check{s}^{(r)}_{2,2} & \check{s}^{(r)}_{2,3} \\ \check{s}^{(r)}_{3,0} & \check{s}^{(r)}_{3,1} & \check{s}^{(r)}_{3,2} & \check{s}^{(r)}_{3,3} \end{bmatrix}$$

  be the **parity matrix** associated with $s^{(r)}$, such that

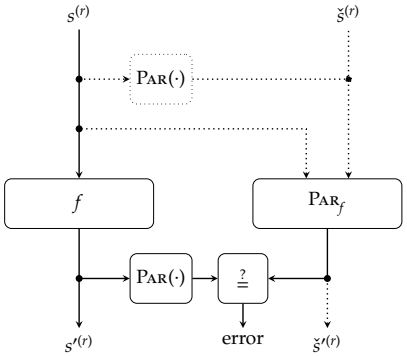
$$\check{s}^{(r)}_{i,j} \;=\; \textsc{Par}\left(s^{(r)}_{i,j} \right)$$

  and similarly for each round key $rk^{(r)}$.

Notes:

Part 2.2: in practice (6)
Countermeasures: $\mathcal{T} \simeq$ AES, Δ = laser pulse

- Problem: Δ can be used to influence, e.g., corrupt $s^{(r)}$.
- Idea [5]:
  - we can predict and check parity matrix per



  at say a
  1. round function,
  2. round, or
  3. encryption/decryption

  granularity: we just need a $\textsc{Par}_f$ for each f ...

Notes:

- Problem: Δ can be used to influence, e.g., corrupt $s^{(r)}$.
- Idea [5]:
  - for $x, y \in \mathbb{F}_{2^8}$, note that we have

$$\begin{aligned}
\textsc{Par}(x \oplus_{\mathbb{F}_{2^8}} y) &= \textsc{Par}(x) \oplus \textsc{Par}(y) \\
\textsc{Par}(\mathtt{01} \otimes_{\mathbb{F}_{2^8}} x) &= \textsc{Par}(x) \\
\textsc{Par}(\mathtt{02} \otimes_{\mathbb{F}_{2^8}} x) &= \text{MSB}(x) \oplus \textsc{Par}(x) \\
\textsc{Par}(\mathtt{03} \otimes_{\mathbb{F}_{2^8}} x) &= \text{MSB}(x)
\end{aligned}$$

Notes:

Part 2.2: in practice (6)
Countermeasures: $\mathcal{T} \simeq$ AES, Δ = laser pulse

- Problem: Δ can be used to influence, e.g., corrupt $s^{(r)}$.
- Idea [5]:
  - putting everything together, we construct and use

$$\begin{aligned}
\textsc{Par}_{\textsc{Key-Addition}} &: \text{compute } \check{s}_{i,j}^{(r)} \oplus \check{rk}_{i,j}^{(r)} \\
\textsc{Par}_{\textsc{Shift-Rows}} &: \text{rotate each row of } \check{s}_{i,j}^{(r)} \\
\textsc{Par}_{\textsc{Sub-Bytes}} &: \text{compute } \textsc{Par}\left(\text{S-box}(s_{i,j}^{(r)})\right) \\
\textsc{Par}_{\textsc{Mix-Columns}} &: \text{apply } \textsc{Par}_{\textsc{Mix-Column}} \text{ to each column of } \check{s}_{i,j}^{(r)}
\end{aligned}$$

where

$$\begin{bmatrix} \check{s}'^{(r)}_{0,j} \\ \check{s}'^{(r)}_{1,j} \\ \check{s}'^{(r)}_{2,j} \\ \check{s}'^{(r)}_{3,j} \end{bmatrix} = \textsc{Par}_{\textsc{Mix-Column}}\left(\begin{bmatrix} \check{s}^{(r)}_{0,j} \\ \check{s}^{(r)}_{1,j} \\ \check{s}^{(r)}_{2,j} \\ \check{s}^{(r)}_{3,j} \end{bmatrix}\right) = \begin{bmatrix} \left(\text{MSB}\left(s^{(r)}_{0,j}\right) \oplus \check{s}^{(r)}_{0,j}\right) \oplus \text{MSB}\left(s^{(r)}_{1,j}\right) \oplus \check{s}^{(r)}_{2,j} \oplus \check{s}^{(r)}_{3,j} \\ \check{s}^{(r)}_{0,j} \oplus \left(\text{MSB}\left(s^{(r)}_{1,j}\right) \oplus \check{s}^{(r)}_{1,j}\right) \oplus \text{MSB}\left(s^{(r)}_{2,j}\right) \oplus \check{s}^{(r)}_{3,j} \\ \check{s}^{(r)}_{0,j} \oplus \check{s}^{(r)}_{1,j} \oplus \left(\text{MSB}\left(s^{(r)}_{2,j}\right) \oplus \check{s}^{(r)}_{2,j}\right) \oplus \text{MSB}\left(s^{(r)}_{3,j}\right) \\ \text{MSB}\left(s^{(r)}_{0,j}\right) \oplus \check{s}^{(r)}_{1,j} \oplus \check{s}^{(r)}_{2,j} \oplus \left(\text{MSB}\left(s^{(r)}_{3,j}\right) \oplus \check{s}^{(r)}_{3,j}\right) \end{bmatrix}$$

© Daniel Page ⟨...⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Notes:

Conclusions

- Take away points:
  - For $\mathcal{E}$, this is fun!
    - can ignore the rules (cf. do whatever possible, versus what is modelled),
    - less and less "low hanging fruit", but still lots of opportunity,
    - more and more applicability at scale,
    - ...
  - For $\mathcal{T}$, this can be *really* difficult!
    - implications go beyond technical, into, e.g., reputational,
    - many general principles apply, but often specific details matter,
    - need to consider multiple layers of abstraction,
    - satisfactory trade-offs (e.g., efficiency versus security) are challenging,
    - raise problematic questions re. development practice, supply-chain, etc.
    - ...

Notes:

Additional Reading

- S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
- P.C. Kocher et al. "Introduction to differential power analysis". In: *Journal of Cryptographic Engineering (JCEN)* 1.1 (2011), pp. 5–27.
- M. Joye and M. Tunstall, eds. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012.
- H. Bar-El et al. "The Sorcerer's Apprentice Guide to Fault Attacks". In: *Proceedings of the IEEE* 94.2 (2006), pp. 370–382.
- A. Barenghi et al. "Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures". In: *Proceedings of the IEEE* 100.11 (2012), pp. 3056–3076.
- D. Karaklajić, J.-M. Schmidt, and I. Verbauwhede. "Hardware Designer's Guide to Fault Attacks". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21.12 (2013), pp. 2295–2306.
- B. Yuce, P. Schaumont, and M. Witteman. "Fault Attacks on Secure Embedded Software: Threats, Design, and Evaluation". In: *Journal of Hardware and Systems Security* 2.2 (2018), pp. 111–130.

Notes:

References

[1] M. Joye and M. Tunstall, eds. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012 (see p. 151).

[2] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007 (see pp. 121, 123, 125, 127, 129, 131, 133, 135, 151).

[3] H. Bar-El et al. "The Sorcerer's Apprentice Guide to Fault Attacks". In: *Proceedings of the IEEE* 94.2 (2006), pp. 370–382 (see p. 151).

[4] A. Barenghi et al. "Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures". In: *Proceedings of the IEEE* 100.11 (2012), pp. 3056–3076 (see p. 151).

[5] G. Bertoni et al. "A parity code based fault detection for an implementation of the Advanced Encryption Standard". In: *Defect and Fault Tolerance in VLSI Systems (DFT)*. 2002, pp. 51–59 (see pp. 141, 143, 145, 147).

[6] D. Boneh, R. DeMillo, and R. Lipton. "On the importance of checking cryptographic protocols for faults". In: *Journal of Cryptology* 14.2 (2001), pp. 101–119 (see p. 73).

[7] E. Brier, C. Clavier, and F. Olivier. "Correlation Power Analysis with a Leakage Model". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 3156. Springer-Verlag, 2004, pp. 16–29 (see pp. 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61).

[8] C. Herbst, E. Oswald, and S. Mangard. "An AES Smart Card Implementation Resistant to Power Analysis Attacks". In: *Applied Cryptography and Network Security (ACNS)*. LNCS 3989. Springer-Verlag, 2006, pp. 239–252 (see pp. 121, 123, 125, 127, 129, 131, 133, 135).

[9] D. Karaklajić, J.-M. Schmidt, and I. Verbauwhede. "Hardware Designer's Guide to Fault Attacks". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21.12 (2013), pp. 2295–2306 (see p. 151).

[10] Ç.K. Koç, T. Acar, and B.S. Kaliski. "Analyzing and comparing Montgomery multiplication algorithms". In: *IEEE Micro* 16.3 (1996), pp. 26–33 (see p. 7).

[11] P.C. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis". In: *Advances in Cryptology (CRYPTO)*. LNCS 1666. Springer-Verlag, 1999, pp. 388–397 (see pp. 11, 13).

Notes:

References

[12] P.C. Kocher et al. "Introduction to differential power analysis". In: *Journal of Cryptographic Engineering (JCEN)* 1.1 (2011), pp. 5–27 (see p. 151).

[13] J-J. Quisquater and C. Couvreur. "Fast decipherment algorithm for RSA public-key cryptosystem". In: *IEE Electronics Letters* 18.21 (1982), pp. 905–907 (see p. 71).

[14] M. Tunstall, D. Mukhopadhyay, and S. Ali. "Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault". In: *Workshop on Information Security Theory and Practice (WISTP)*. LNCS 6633. Springer-Verlag, 2011, pp. 224–233 (see pp. 77, 79, 81, 83, 85, 87, 89, 91, 93, 95).

[15] B. Yuce, P. Schaumont, and M. Witteman. "Fault Attacks on Secure Embedded Software: Threats, Design, and Evaluation". In: *Journal of Hardware and Systems Security* 2.2 (2018), pp. 111–130 (see p. 151).

© Daniel Page ⟨[redacted]⟩
Applied Cryptology
University of BRISTOL
git # c8178615 @ 2024-04-24

Notes: