Applied Cryptology

Daniel Page

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

September 5, 2025

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**!

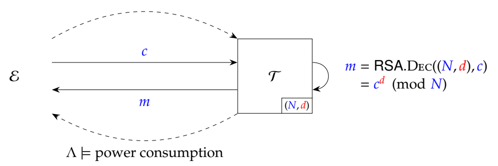
$$\sim 2 \text{ hours} \quad \Rightarrow \quad \text{introductory, and (very) selective (versus definitive) coverage.}$$

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** $\mathcal{T}$




```
algorithm RSA.Dec((N, d), c) begin
  Pre-compute Π = (N, ρ, ω) from N
  ĉ ← MontMul(Π, c, ρ² mod N)
  r̂ ← MontMul(Π, c, ρ² mod N)
  for i = |d| − 2 downto 0 do
    r̂ ← MontSqr(Π, r̂)
    if d_i = 1 then
      r̂ ← MontMul(Π, r̂, ĉ)
    end
  end
  return MontMul(Π, r̂, 1)
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

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

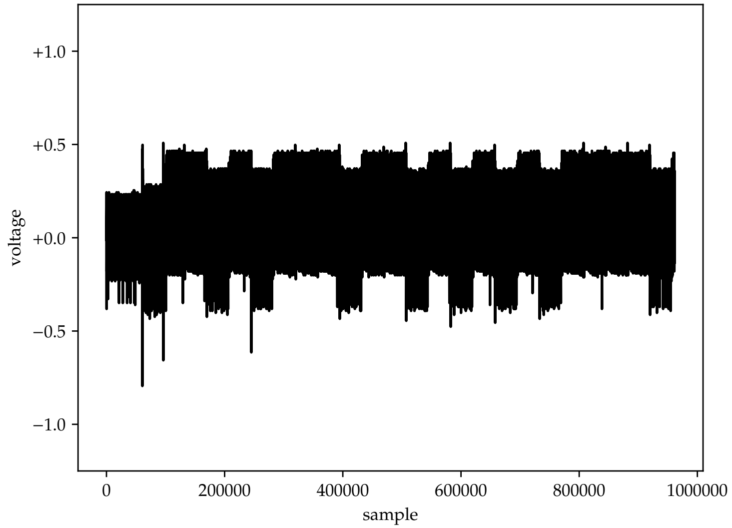
      the former is more efficient than the latter,
    - the Montgomery squaring and Montgomery multiplication implementations are FIOS-based [12],
  - how can $\mathcal{E}$ mount a successful attack, i.e., recover d ?

Notes:

Part 2.1: in practice (2)

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

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

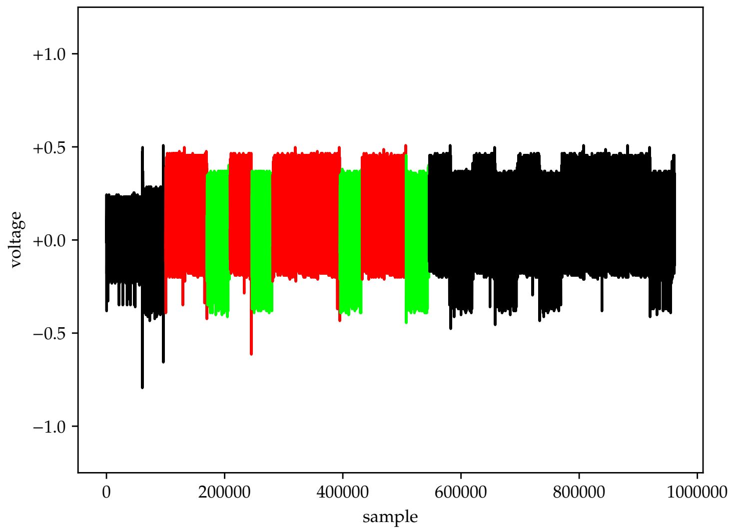


Notes:

Part 2.1: in practice (2)

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

- Attack [14]:

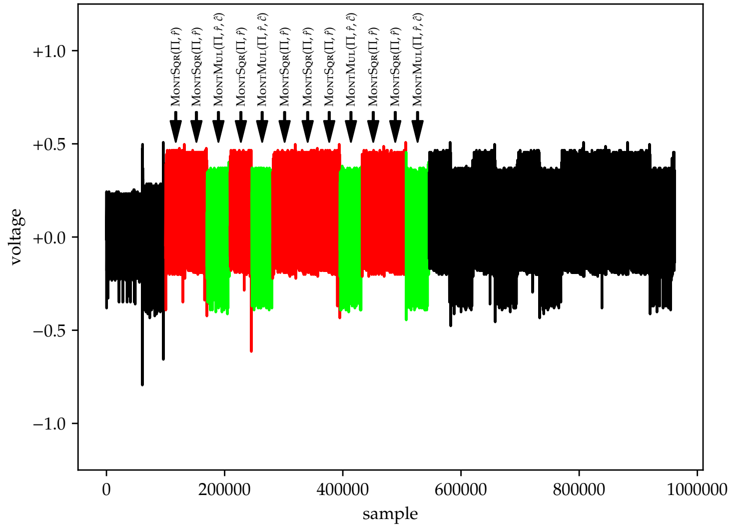


Notes:

Part 2.1: in practice (2)

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

- Attack [14]:

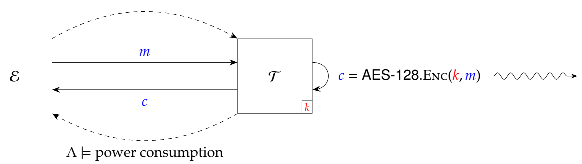


Notes:

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** $\mathcal{T}$




```
algorithm AES-128.Enc(k, m) begin
  s ← m
  s ← AddRoundKey(s, k = rk^(0))
  for r = 1 upto 9 step +1 do
    k ← EvolveRoundKey(k, rc^(r))
    s ← SubBytes(s)
    s ← ShiftRows(s)
    s ← MixColumns(s)
    s ← AddRoundKey(s, k = rk^(r))
  end
  k ← EvolveRoundKey(k, rc^(10))
  s ← SubBytes(s)
  s ← ShiftRows(s)
  s ← AddRoundKey(s, k = rk^(10))
  c ← s
  return c
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]:

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

Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

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

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

Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

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

$$
\begin{array}{r}
\mathcal{M}_0 \xrightarrow{\text{HW}(\textsc{S-box}(\mathcal{M}_{0,t} \oplus \tilde{k}_t))} \\
\mathcal{M}_1 \xrightarrow{\text{HW}(\textsc{S-box}(\mathcal{M}_{1,t} \oplus \tilde{k}_t))} \\
\vdots \\
\mathcal{M}_{n-1} \xrightarrow{\text{HW}(\textsc{S-box}(\mathcal{M}_{n-1,t} \oplus \tilde{k}_t))}
\end{array}
\begin{pmatrix}
\mathcal{H}_{0,0} & \mathcal{H}_{0,1} & \cdots & \mathcal{H}_{0,h-1} \\
\mathcal{H}_{1,0} & \mathcal{H}_{1,1} & \cdots & \mathcal{H}_{1,h-1} \\
\vdots & \vdots & \ddots & \vdots \\
\mathcal{H}_{n-1,0} & \mathcal{H}_{n-1,1} & \cdots & \mathcal{H}_{n-1,h-1}
\end{pmatrix}
$$

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

Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

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

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

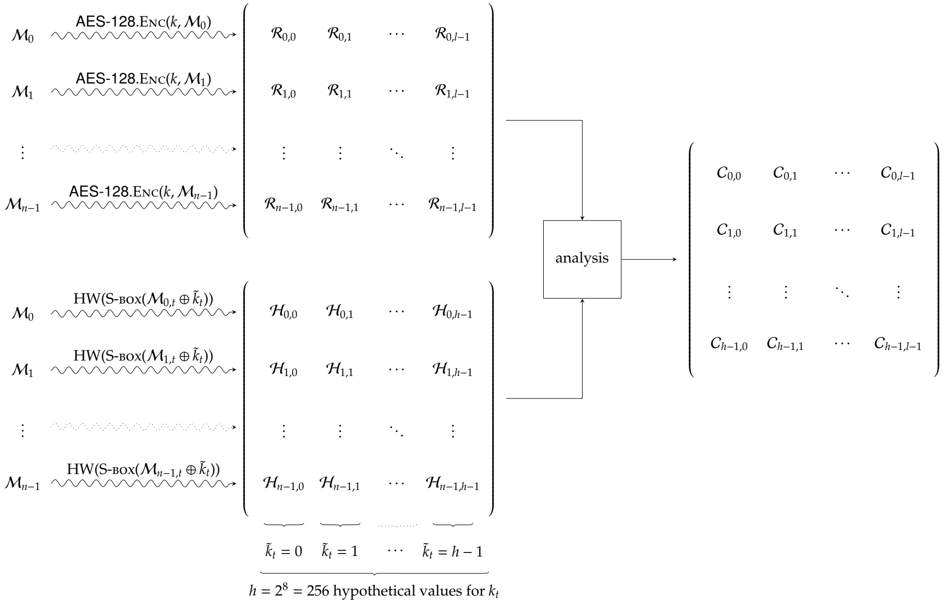
$$
\begin{array}{r}
\mathcal{M}_0 \xrightarrow{\text{HW}(\textsc{S-box}(\mathcal{M}_{0,t} \oplus \tilde{k}_t))} \\
\mathcal{M}_1 \xrightarrow{\text{HW}(\textsc{S-box}(\mathcal{M}_{1,t} \oplus \tilde{k}_t))} \\
\vdots \\
\mathcal{M}_{n-1} \xrightarrow{\text{HW}(\textsc{S-box}(\mathcal{M}_{n-1,t} \oplus \tilde{k}_t))}
\end{array}
\begin{pmatrix}
\mathcal{H}_{0,0} & \mathcal{H}_{0,1} & \cdots & \mathcal{H}_{0,h-1} \\
\mathcal{H}_{1,0} & \mathcal{H}_{1,1} & \cdots & \mathcal{H}_{1,h-1} \\
\vdots & \vdots & \ddots & \vdots \\
\mathcal{H}_{n-1,0} & \mathcal{H}_{n-1,1} & \cdots & \mathcal{H}_{n-1,h-1}
\end{pmatrix}
$$

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

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

Notes:

- Attack [7]:

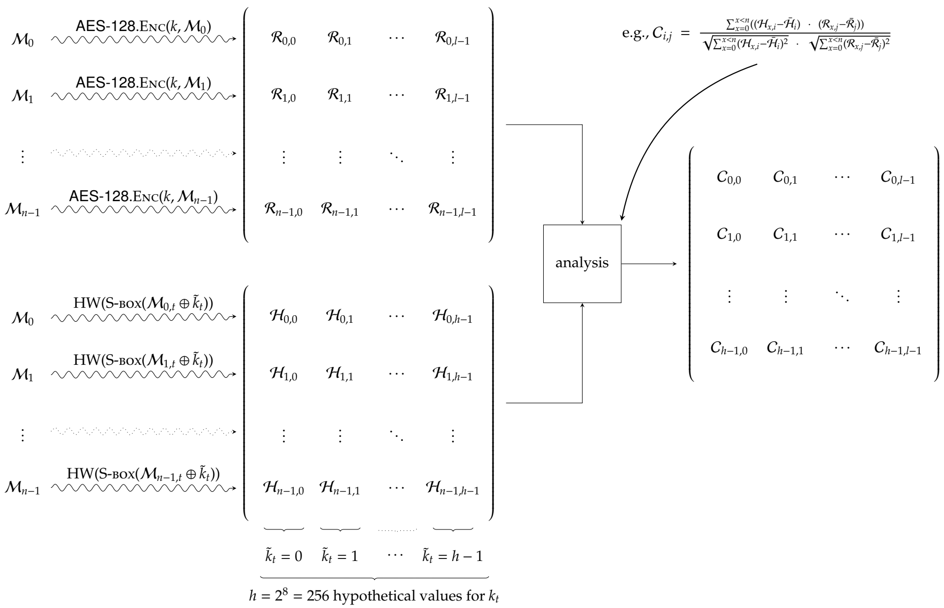


Notes:

Part 2.1: in practice (4)
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Attack [7]:

e.g., $C_{i,j} = \frac{\sum_{x=0}^{x<n}((\mathcal{H}_{x,i} - \bar{\mathcal{H}}_i) \cdot (\mathcal{R}_{x,j} - \bar{\mathcal{R}}_j))}{\sqrt{\sum_{x=0}^{x<n}(\mathcal{H}_{x,i} - \bar{\mathcal{H}}_i)^2} \cdot \sqrt{\sum_{x=0}^{x<n}(\mathcal{R}_{x,j} - \bar{\mathcal{R}}_j)^2}}$

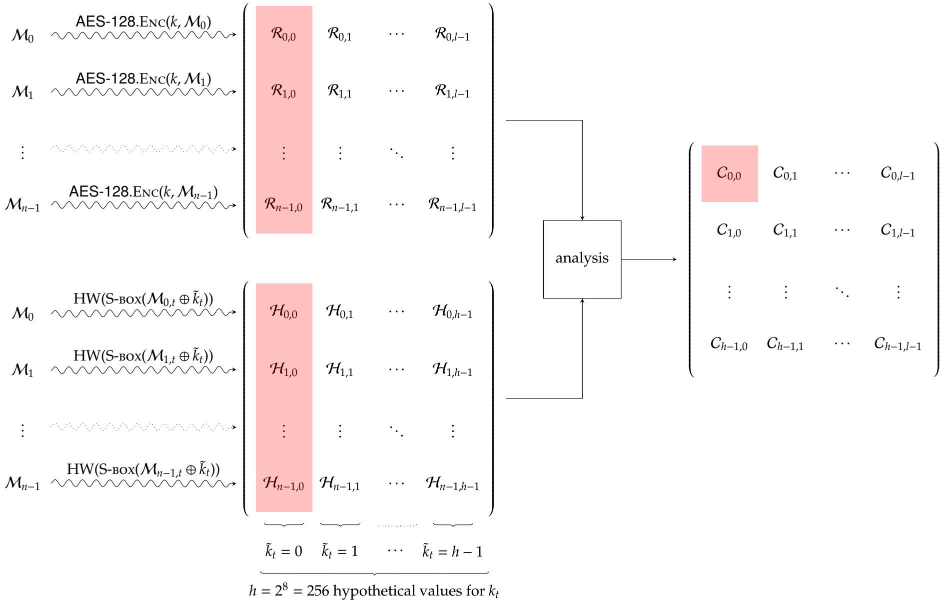


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

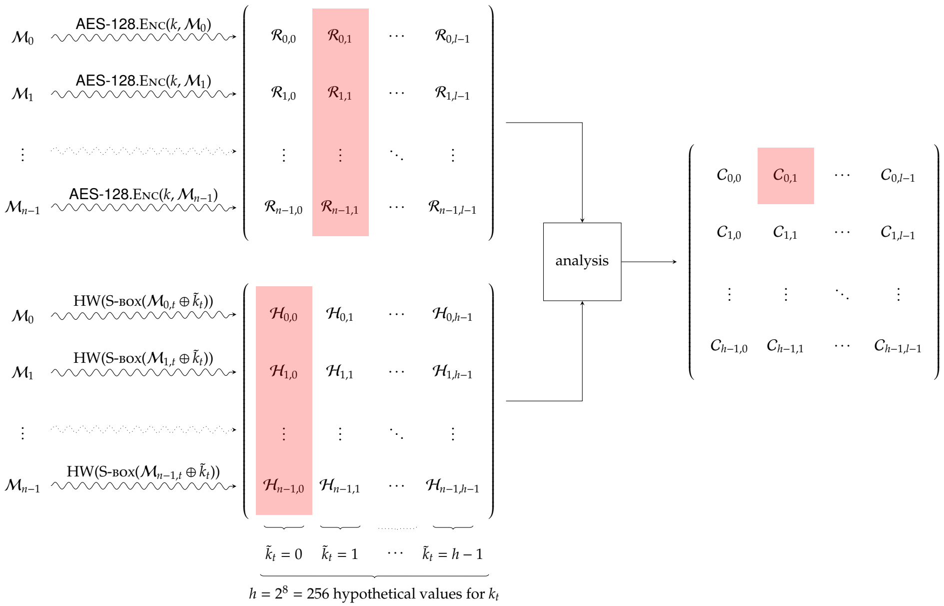


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

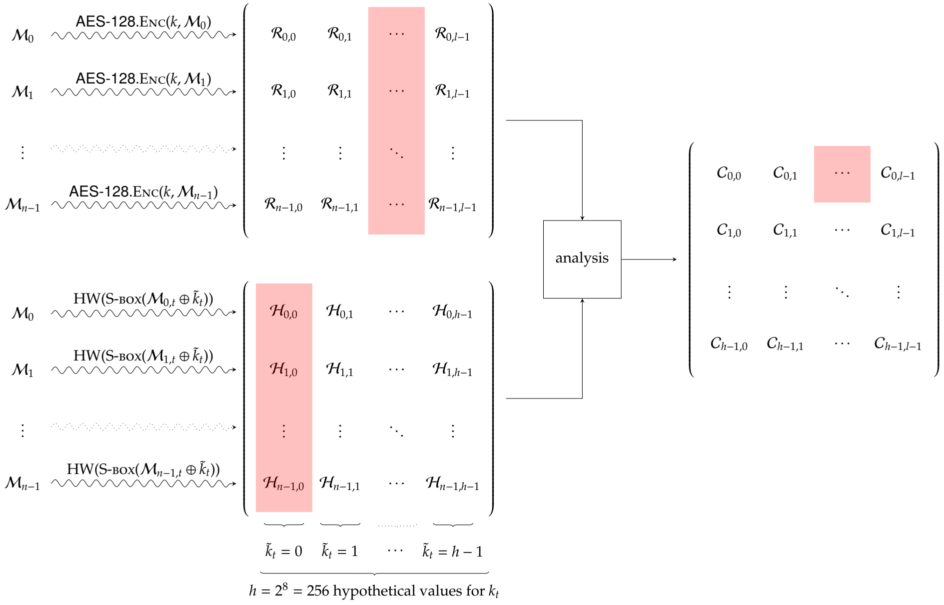


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

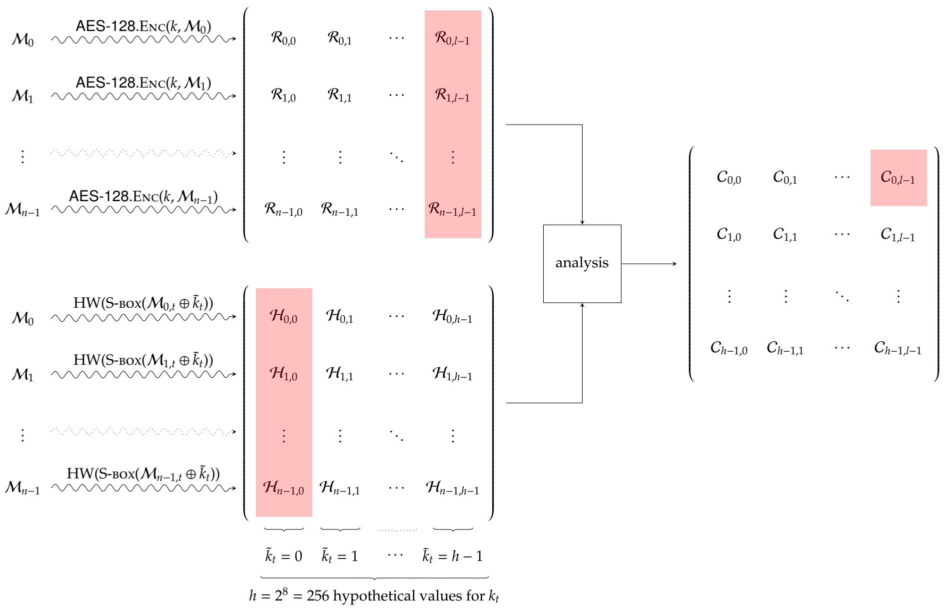


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

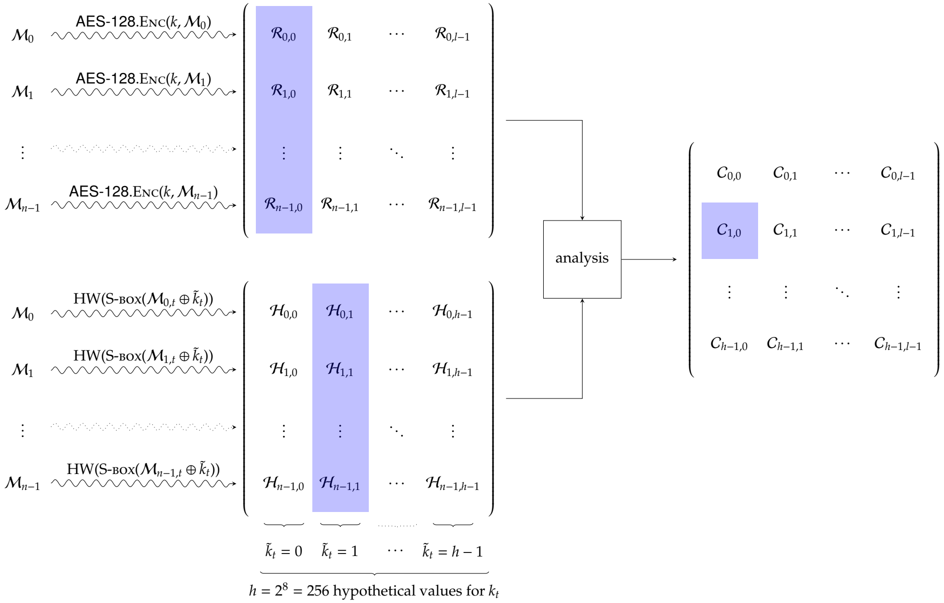


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

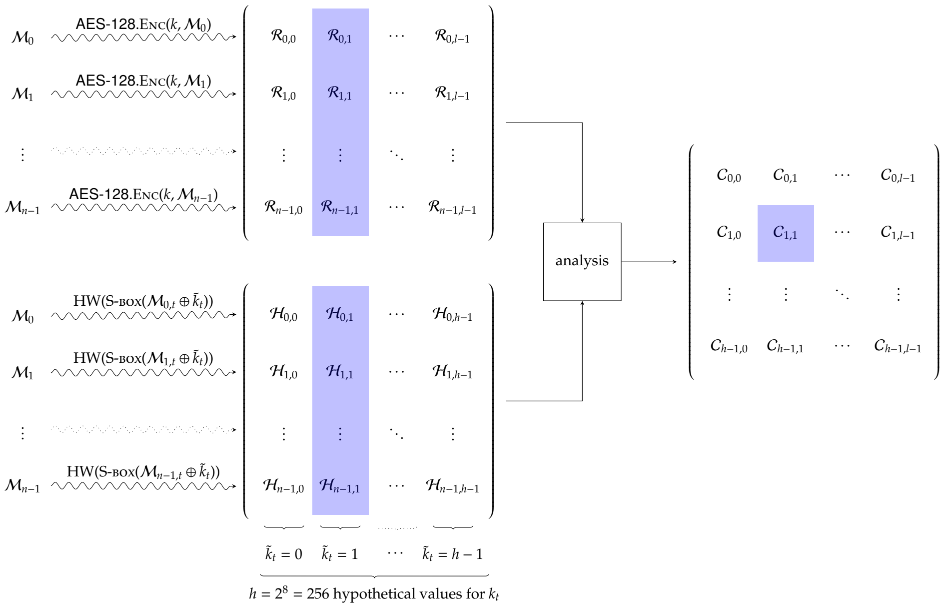


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

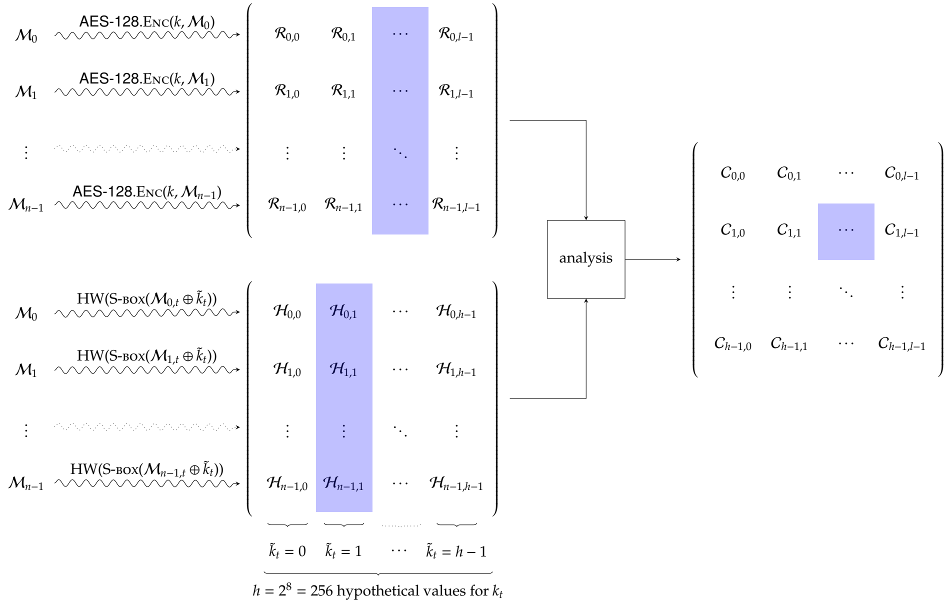


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

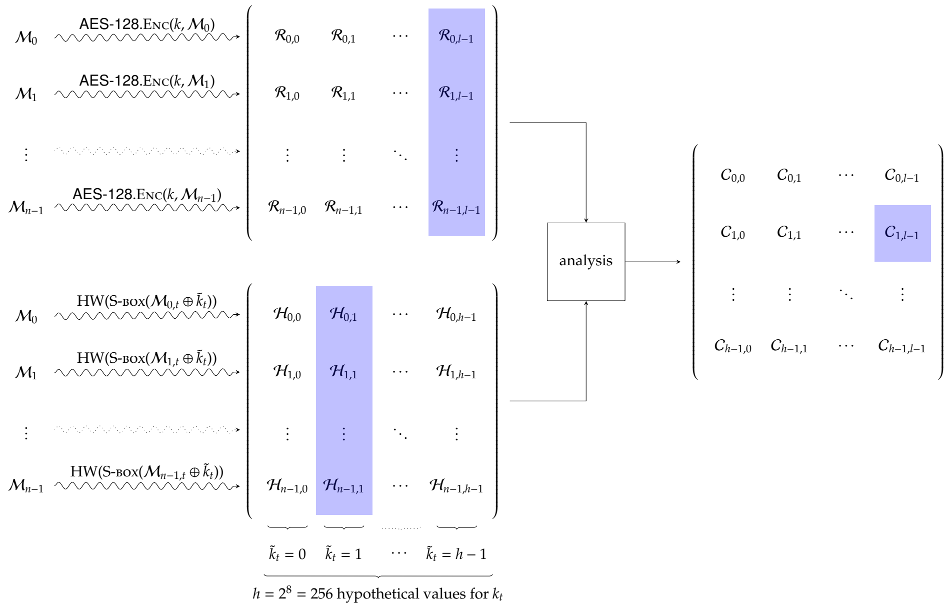


Notes:

Part 2.1: in practice (4)

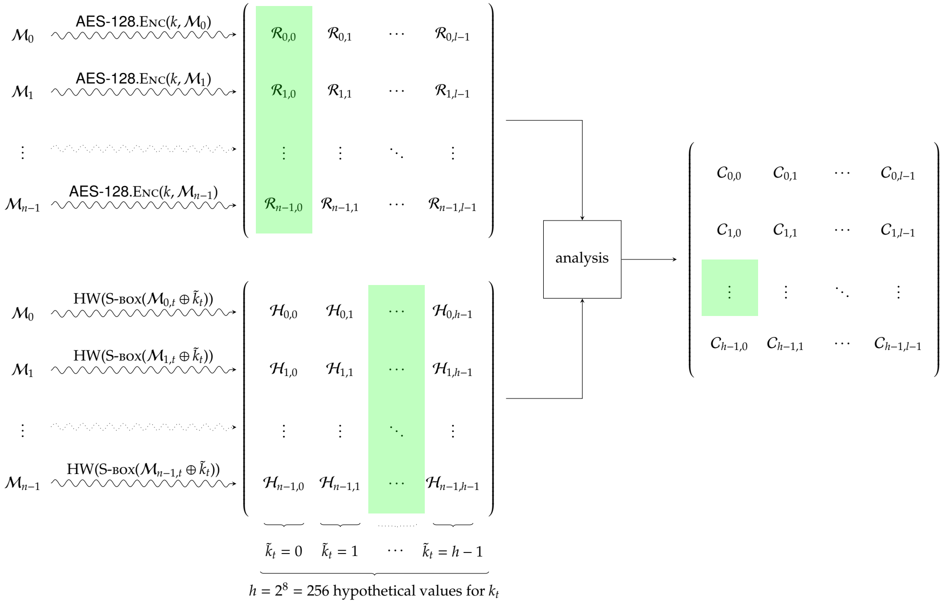
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Attack [7]:



Notes:

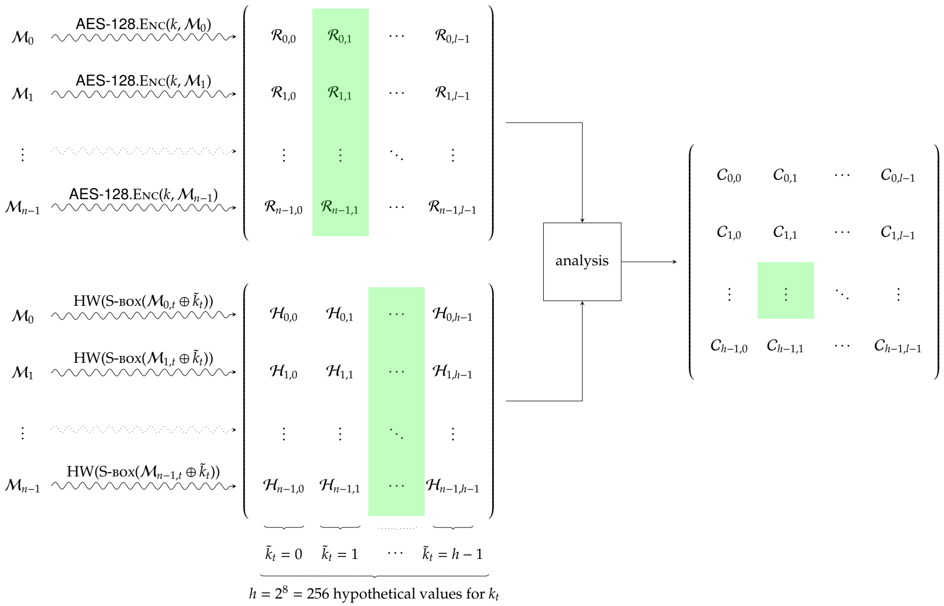
- Attack [7]:



Notes:

Part 2.1: in practice (4)
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

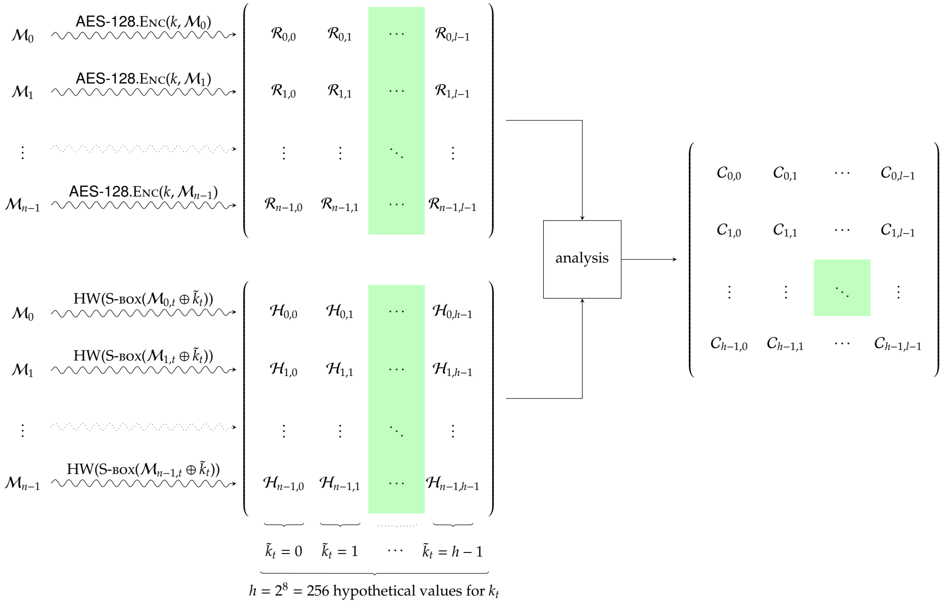
- Attack [7]:



© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

Notes:

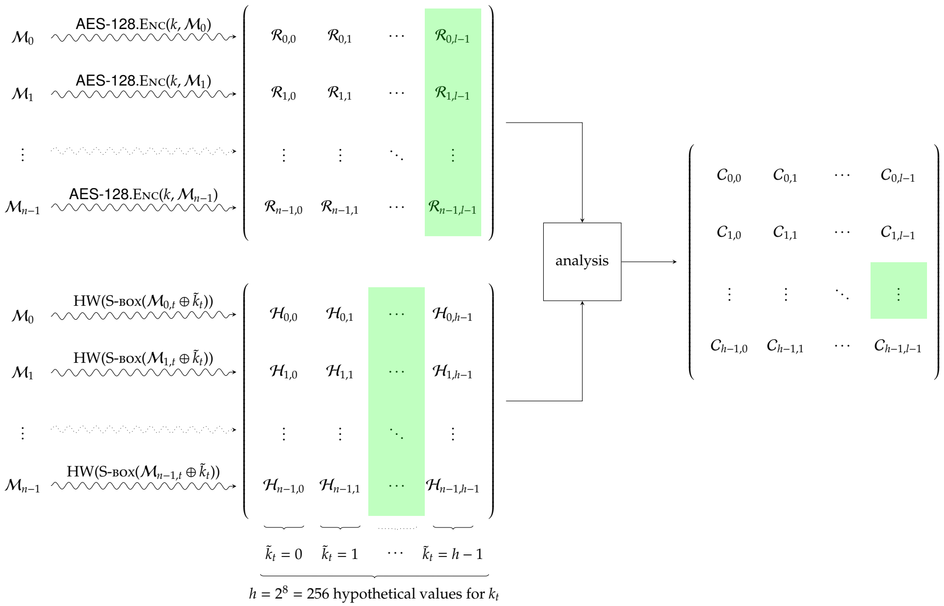
- **Attack** [7]:



Notes:

Part 2.1: in practice (4)
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- **Attack** [7]:

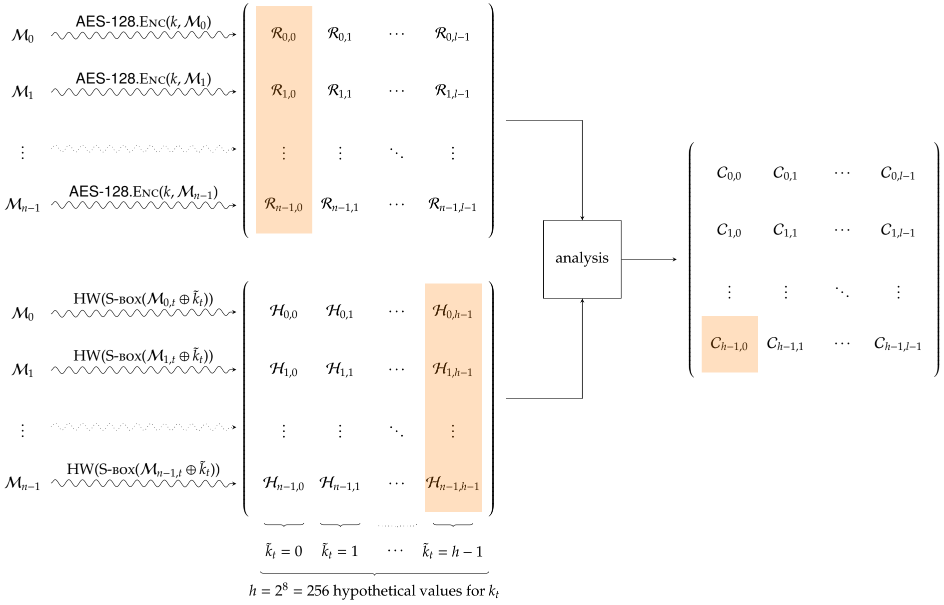


Notes:

Part 2.1: in practice (4)

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

- Attack [7]:

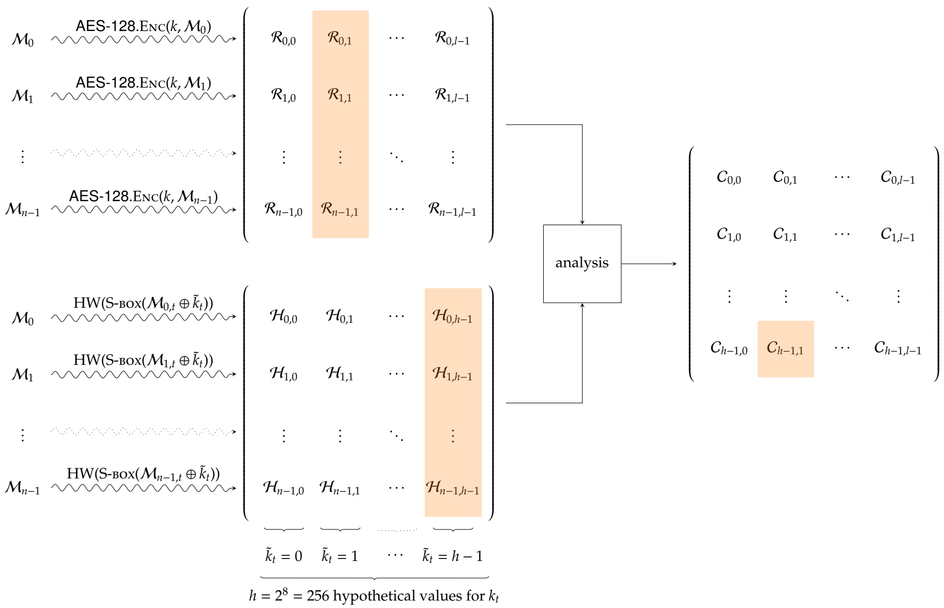


Notes:

Part 2.1: in practice (4)

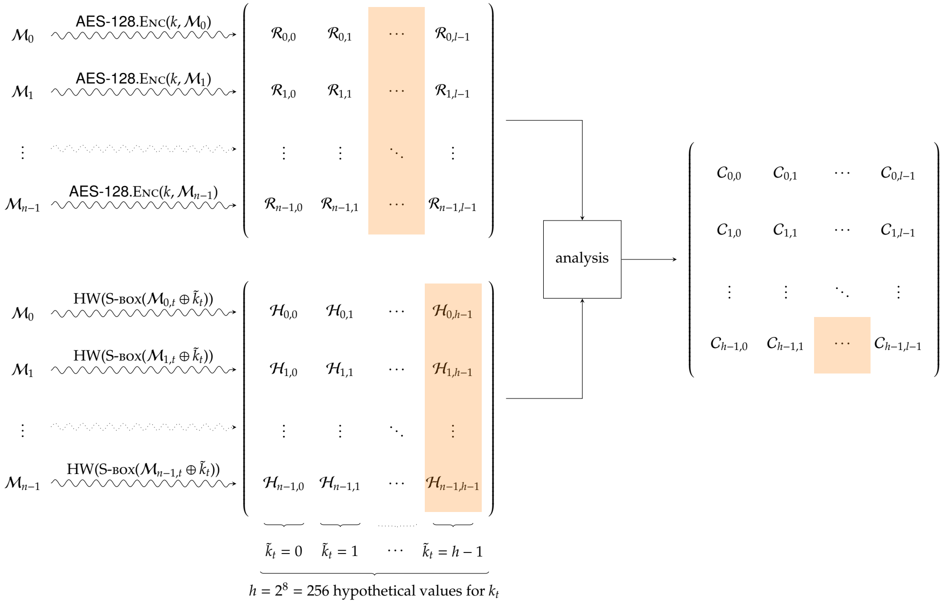
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- Attack [7]:



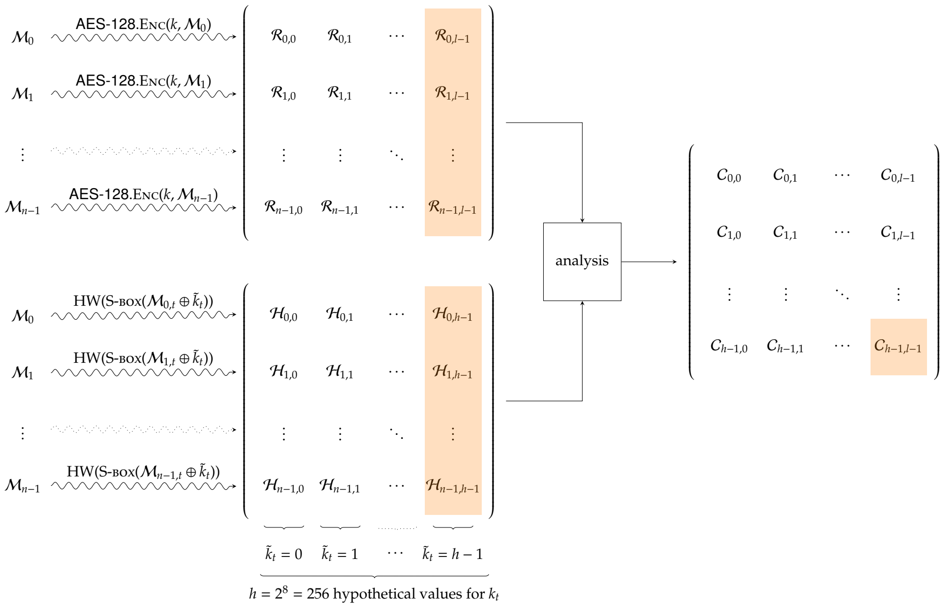
Notes:

- Attack [7]:



Notes:

- Attack [7]:



Notes:

Part 2.1: in practice (4)
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

- **Attack** [7]:

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

$$
\begin{array}{rcl}
\mathcal{M}_0 & \xrightarrow{\text{HW}(\text{S-box}(\mathcal{M}_{0,t} \oplus \tilde{k}_t))} & \\
\mathcal{M}_1 & \xrightarrow{\text{HW}(\text{S-box}(\mathcal{M}_{1,t} \oplus \tilde{k}_t))} & \\
\vdots & & \\
\mathcal{M}_{n-1} & \xrightarrow{\text{HW}(\text{S-box}(\mathcal{M}_{n-1,t} \oplus \tilde{k}_t))} &
\end{array}
\underbrace{\left(\begin{array}{cccc}
\mathcal{H}_{0,0} & \mathcal{H}_{0,1} & \cdots & \mathcal{H}_{0,h-1} \\
\mathcal{H}_{1,0} & \mathcal{H}_{1,1} & \cdots & \mathcal{H}_{1,h-1} \\
\vdots & \vdots & \ddots & \vdots \\
\mathcal{H}_{n-1,0} & \mathcal{H}_{n-1,1} & \cdots & \mathcal{H}_{n-1,h-1} \\
\tilde{k}_t = 0 & \tilde{k}_t = 1 & \cdots & \tilde{k}_t = h-1
\end{array}\right)}_{h = 2^8 = 256 \text{ hypothetical values for } k_t}
$$

$$
\xrightarrow{\text{analysis}}
\left(\begin{array}{cccc}
C_{0,0} & C_{0,1} & \cdots & C_{0,l-1} \\
C_{1,0} & C_{1,1} & \cdots & C_{1,l-1} \\
\vdots & \vdots & & \vdots \\
C_{h-1,0} & C_{h-1,1} & \cdots & C_{h-1,l-1}
\end{array}\right)
$$

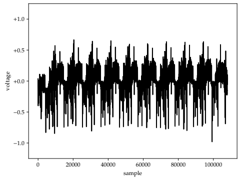
at $C_{\tilde{k}_t, \tilde{j}}$ we expect a "peak" due to high(er) correlation;
identifying said "peak" recovers k_t

Notes:

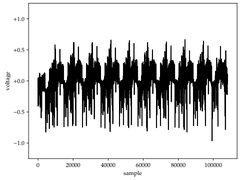
Part 2.1: in practice (5)
Attacks: $\mathcal{T} \simeq$ AES, Λ = power consumption

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

$$
\left.\begin{array}{rcl}
k & = & \langle 2B_{(16)}, 7E_{(16)}, \ldots, 3C_{(16)} \rangle \\
\mathcal{M}_0 & = & \langle 1C_{(16)}, E5_{(16)}, \ldots, 4B_{(16)} \rangle
\end{array}\right\}
\xrightarrow{\text{AES-128.Enc}(k, \mathcal{M}_0)} \mathcal{R}_0 =
$$

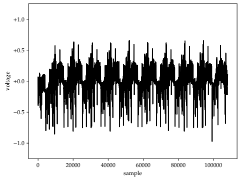


$$
\left.\begin{array}{rcl}
k & = & \langle 2B_{(16)}, 7E_{(16)}, \ldots, 3C_{(16)} \rangle \\
\mathcal{M}_1 & = & \langle 41_{(16)}, 62_{(16)}, \ldots, DF_{(16)} \rangle
\end{array}\right\}
\xrightarrow{\text{AES-128.Enc}(k, \mathcal{M}_1)} \mathcal{R}_1 =
$$



$\vdots$

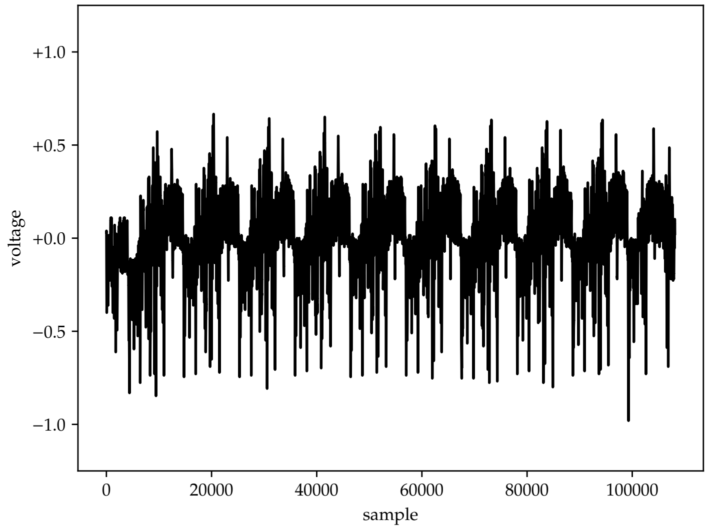
$$
\left.\begin{array}{rcl}
k & = & \langle 2B_{(16)}, 7E_{(16)}, \ldots, 3C_{(16)} \rangle \\
\mathcal{M}_{n-1} & = & \langle E1_{(16)}, CF_{(16)}, \ldots, BE_{(16)} \rangle
\end{array}\right\}
\xrightarrow{\text{AES-128.Enc}(k, \mathcal{M}_{n-1})} \mathcal{R}_{n-1} =
$$



Notes:

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

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

$t = 0$, $C =$  $\rightsquigarrow$ $\tilde{k}_0 = k_0 = 2B_{(16)}$

$t = 1$, $C =$  $\rightsquigarrow$ $\tilde{k}_1 = k_1 = 7E_{(16)}$

$\vdots$

$t = 15$, $C =$  $\rightsquigarrow$ $\tilde{k}_{15} = k_{15} = 3C_{(16)}$

Notes:

Part 2.1: in practice (5)

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

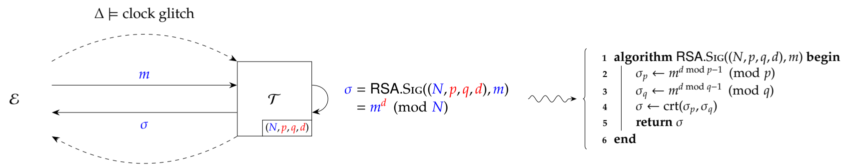
- Example: ARM Cortex-M3, $n = 1000$, $l = 108124$.

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** $\mathcal{T}$



  - and noting that
    - there are no countermeasures implemented,
    - to improve efficiency, a CRT-based [16] implementation is used,
    - based on pre-computation of

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

      the Gauss-based recombination is such that

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

    - 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

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

    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

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

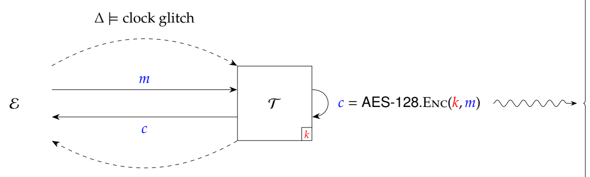
    i.e., factor N, then compute d.

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** $\mathcal{T}$




```
algorithm AES-128.Enc(k, m) begin
  s ← m
  s ← AddRoundKey(s, k = rk^(0))
  for r = 1 upto 9 step +1 do
    k ← EvolveRoundKey(k, rc^(r))
    s ← SubBytes(s)
    s ← ShiftRows(s)
    s ← MixColumns(s)
    s ← AddRoundKey(s, k = rk^(r))
  end
  k ← EvolveRoundKey(k, rc^(10))
  s ← SubBytes(s)
  s ← ShiftRows(s)
  s ← AddRoundKey(s, k = rk^(10))
  c ← s
  return c
end
```

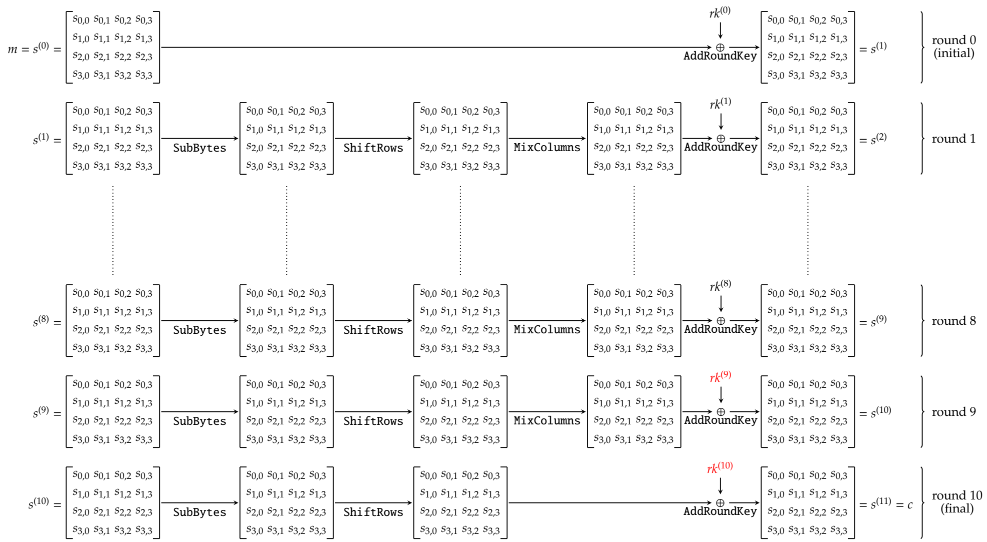
  - and noting that
    - there are no countermeasures implemented,
    - the fault induced randomises $s_{i,j}^{(8)}$, i.e., a chosen element of the state matrix used as input to the 8-th round,
  - how can $\mathcal{E}$ mount a successful attack, i.e., recover k ?

Notes:

Part 2.1: in practice (9)

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

- Attack [17]:

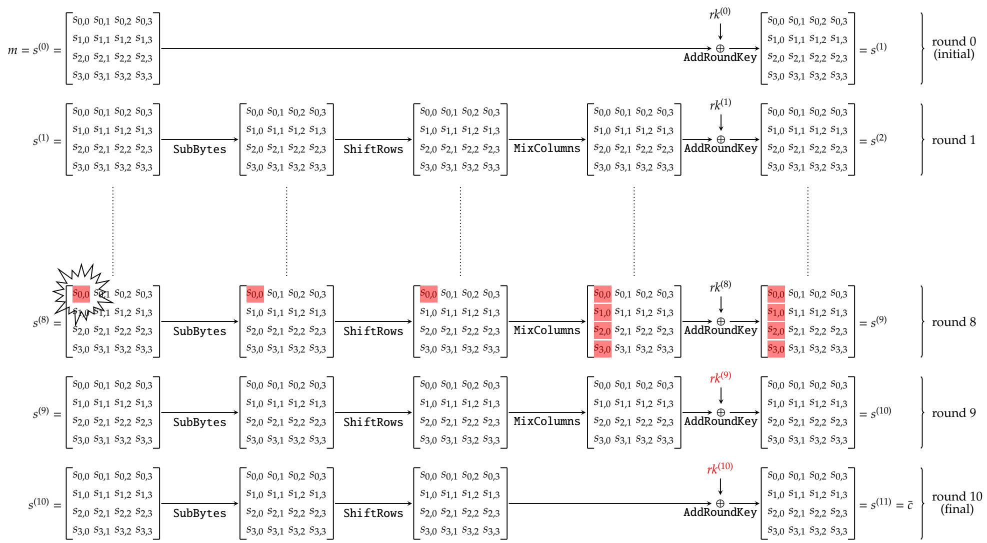


Notes:

Part 2.1: in practice (9)

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

- Attack [17]:

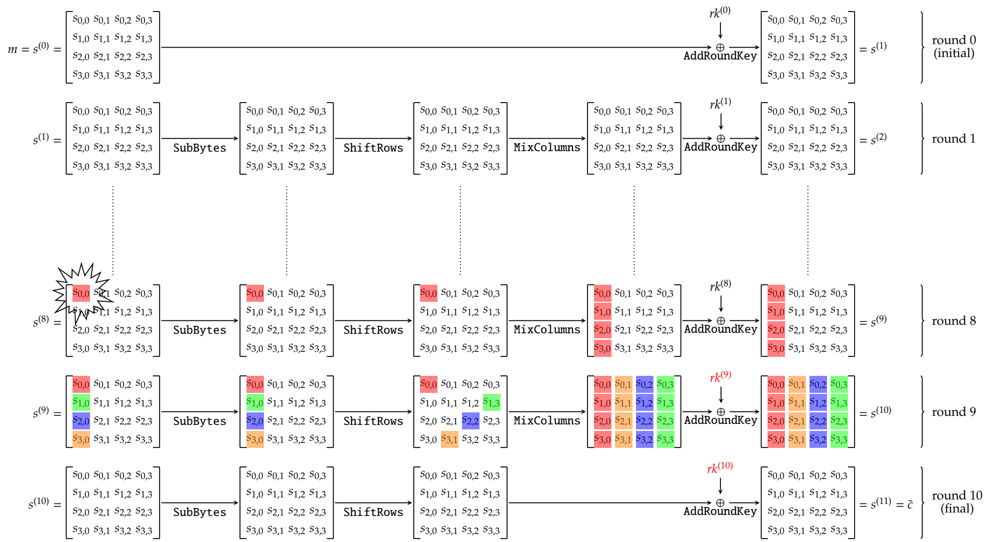


Notes:

Part 2.1: in practice (9)

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

- Attack [17]:

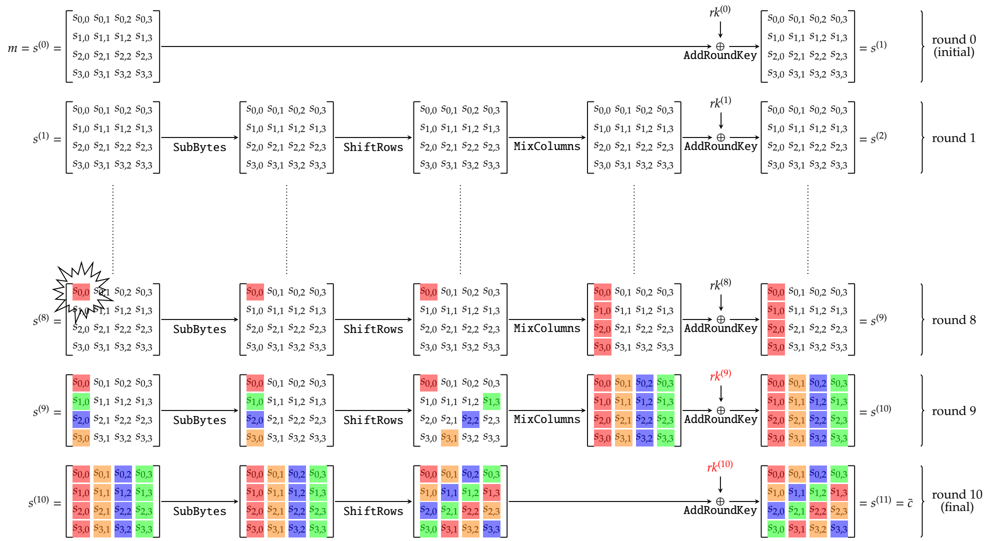


Notes:

Part 2.1: in practice (9)

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

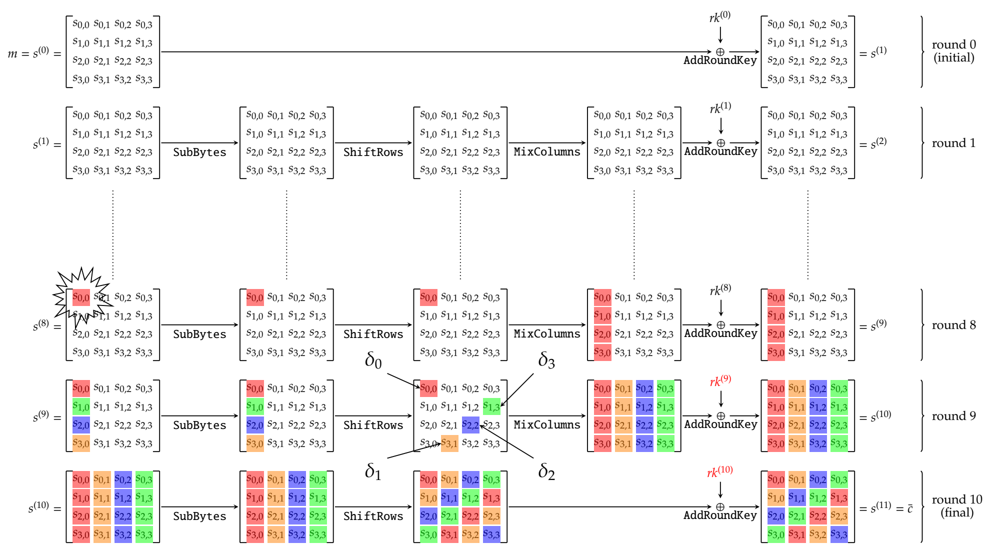
- Attack [17]:



© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

Notes:

- Attack [17]:



Notes:

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

- Attack [17]:
  - Step #1:
    - consider a fault induced in the input of the 8-th round, such that

$$\begin{array}{rcl} \tilde{c} = \text{AES-128.Enc}(k, m) & \Leftarrow & \text{fault induced} \\ c = \text{AES-128.Enc}(k, m) & \Leftarrow & \text{no fault induced} \end{array}$$

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

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

    - 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, $\Delta =$ clock glitch

- Attack [17]:
  - Step #1:
    - consider a fault induced in the input of the 8-th round, such that

$$\begin{array}{rcl} \tilde{c} = \text{AES-128.Enc}(k, m) & \Leftarrow & \text{fault induced} \\ c = \text{AES-128.Enc}(k, m) & \Leftarrow & \text{no fault induced} \end{array}$$

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

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

    - 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, $\Delta =$ clock glitch

- Attack [17]:
  - Step #1:
    - consider a fault induced in the input of the 8-th round, such that

$$\begin{array}{rcl} \tilde{c} = \text{AES-128.Enc}(k, m) & \Leftarrow & \text{fault induced} \\ c = \text{AES-128.Enc}(k, m) & \Leftarrow & \text{no fault induced} \end{array}$$

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

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

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

Notes:

- Attack [17]:
  - Step #1:
    - consider a fault induced in the input of the 8-th round, such that

$$\begin{array}{rcl} \tilde{c} = \text{AES-128.}\textsc{Enc}(k, m) & \Leftarrow & \text{fault induced} \\ c = \text{AES-128.}\textsc{Enc}(k, m) & \Leftarrow & \text{no fault induced} \end{array}$$

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

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

    - 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 [17]:
  - Step #1:
    - consider a fault induced in the input of the 8-th round, such that

$$\begin{array}{rcl} \tilde{c} = \text{AES-128.}\textsc{Enc}(k, m) & \Leftarrow & \text{fault induced} \\ c = \text{AES-128.}\textsc{Enc}(k, m) & \Leftarrow & \text{no fault induced} \end{array}$$

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

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

    - 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.

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

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):

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

```
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):

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

```
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, 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. balanced (via dummy XOR):

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

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

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

  if( c ) {
    x = x ^ 0x1B;
  }
  else {
    x = x ^ 0x00;
  }

  return x;
}
```

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

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
  2. straight-line (via multiplexer):

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

```
uint8_t aes_gf28_mulx( uint8_t x ) {
  uint8_t c, t_0, t_1;

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

  t_0 = x ^ 0x00;
  t_1 = x ^ 0x1B;

  x = ( -c & ( t_0 ^ t_1 ) ) ^ t_0;

  return x;
}
```


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):

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

```
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):

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

```
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:

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

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
  1. original (i.e., no countermeasure):

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

```
void aes_enc_rnd_sub( uint8_t* s ) {
  for( int i = 0; i < 16; i++ ) {
    s[ i ] = aes_enc_sbox( s[ i ] );
  }
}
```

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
  2. temporal padding $\Rightarrow$ random delay:

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

```
void aes_enc_rnd_sub( uint8_t* s ) {
  delay( prng() );

  for( int i = 0; i < 16; i++ ) {
    s[ i ] = aes_enc_sbox( s[ i ] );
  }
}
```

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

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

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

```c
void aes_enc_rnd_sub( uint8_t* s ) {
  int r = prng();

  for( int i = 0; i < 16; i++ ) {
    int j = ( i + r ) % 16;

    s[ j ] = aes_enc_sbox( s[ j ] );
  }
}
```

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

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

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

```c
void aes_enc_rnd_sub( uint8_t* s ) {
  int r = prng();

  for( int i = 0; i < 16; i++ ) {
    int j = ( i ^ r ) % 16;

    s[ j ] = aes_enc_sbox( s[ j ] );
  }
}
```

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:

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

```c
void aes_enc_rnd_sub( uint8_t* s ) {
  int T[ 16 ];

  for( int i = 15; i >= 0; i-- ) {
    T[ i ] = i;
  }

  for( int i = 15; i >= 1; i-- ) {
    int j = prng() % ( i + 1 );

    uint8_t t      = T[ j ];
            T[ j ] = T[ i ];
            T[ i ] = t     ;
  }

  for( int i = 0; i < 16; i++ ) {
    int j = T[ i ];

    s[ j ] = aes_enc_sbox( s[ j ] );
  }
}
```


Notes:

Part 2.2: in practice (3)

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

- **Problem**: data-dependent computation within

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

  is observable via Λ.

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

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

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

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

  - 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 [10, 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., $\text{S-}\textsc{box}_{\mu_4}(x \oplus \mu_4) = \text{S-}\textsc{box}(x) \oplus \nu_4$.

Notes:

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

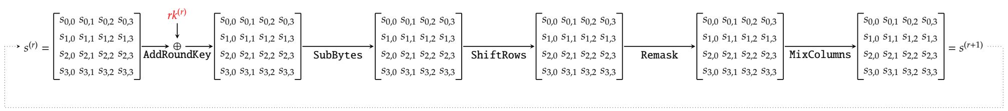
- Problem: data-dependent computation within

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

is observable via Λ.

- Idea [10, Section 3.1] (or see [2, Chapter 9]):

  - Step #2: construct a masked round implementation



Notes:

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

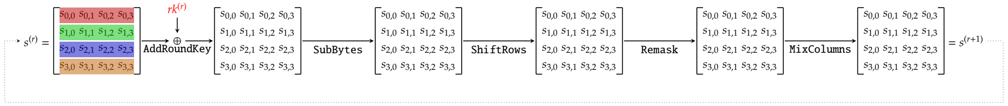
Part 2.2: in practice (4)

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 [10, Section 3.1] (or see [2, Chapter 9]):
  - Step #2: construct a masked round implementation



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

Notes:

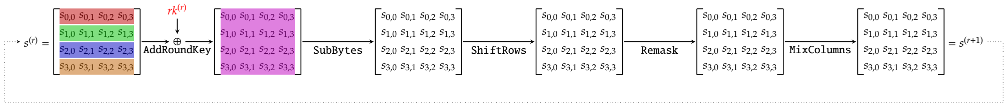
Part 2.2: in practice (4)

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 [10, Section 3.1] (or see [2, Chapter 9]):
  - Step #2: construct a masked round implementation



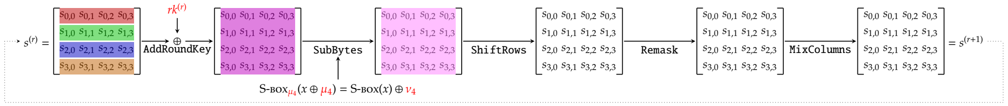
  such that $s_{i,j}^{(r)}$ is masked with μ_4 due to alteration of key schedule (and thus $rk^{(r)}$).

Notes:

- Problem: data-dependent computation within

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

  is observable via Λ.
- Idea [10, Section 3.1] (or see [2, Chapter 9]):
  - Step #2: construct a masked round implementation



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

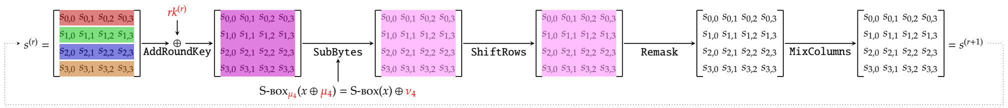
Notes:

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

- Problem: data-dependent computation within

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

  is observable via Λ.
- Idea [10, Section 3.1] (or see [2, Chapter 9]):
  - Step #2: construct a masked round implementation



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

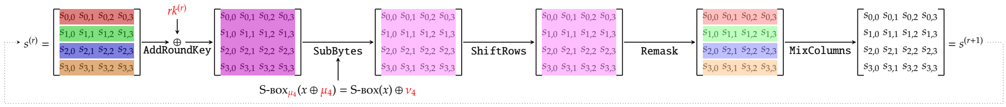
© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

Notes:

- **Problem**: data-dependent computation within

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

  is observable via Λ.
- **Idea** [10, Section 3.1] (or see [2, Chapter 9]):
  - **Step #2**: construct a masked round implementation



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

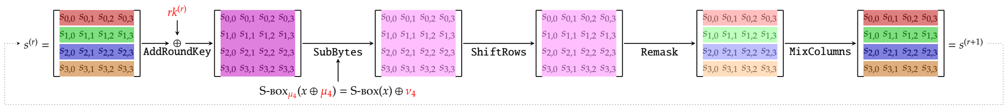
Notes:

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

- **Problem**: data-dependent computation within

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

  is observable via Λ.
- **Idea** [10, Section 3.1] (or see [2, Chapter 9]):
  - **Step #2**: construct a masked round implementation

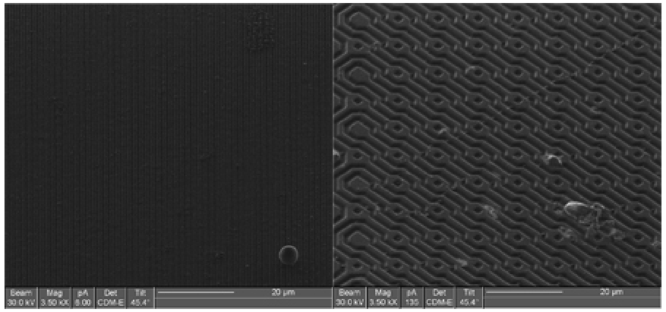


  such that $s^{(r)}_{i,j}$ is masked with ν_i due to alteration of `MixColumns`: this makes iteration possible.

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Applied Cryptology
University of BRISTOL
git # b282dbb9 @ 2025-09-03

Notes:

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



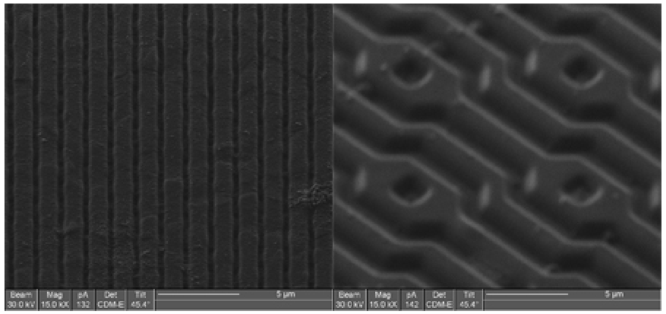
i.e., a top-layer of metal which can be classed as

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

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

Notes:

- 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).

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

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]:
  - 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

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

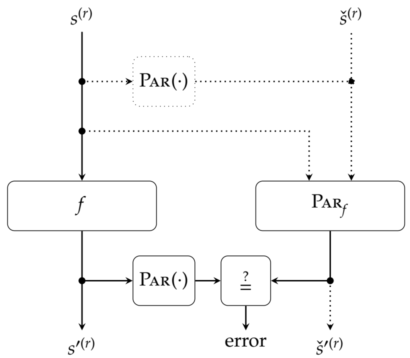
    and similarly for each round key $rk^{(r)}$.

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:

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]:
  - for $x, y \in \mathbb{F}_{2^8}$, note that we have

$$
\begin{array}{rcl}
\textsc{Par}(x \oplus_{\mathbb{F}_{2^8}} y) & = & \textsc{Par}(x) \oplus \textsc{Par}(y) \\
\textsc{Par}(\texttt{01} \otimes_{\mathbb{F}_{2^8}} x) & = & \textsc{Par}(x) \\
\textsc{Par}(\texttt{02} \otimes_{\mathbb{F}_{2^8}} x) & = & \mathrm{MSB}(x) \oplus \textsc{Par}(x) \\
\textsc{Par}(\texttt{03} \otimes_{\mathbb{F}_{2^8}} x) & = & \mathrm{MSB}(x)
\end{array}
$$

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

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

where

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

Notes:

- Problem: data-dependent (conditional) subtraction, i.e., **if** $r \geq N$ **then** $r \leftarrow r - N$.
- Solution [18, 19, 8, 9]:
  - Montgomery multiplication demands input operands in the range
  $$0 \leq \hat{x} < N,$$
  - using a redundant representation, we can relax this to
  $$0 \leq \hat{x} < N \cdot \epsilon$$
  and thereby avoid the conditional subtraction,
  - doing so requires changing the selection of
  $$\rho = b^k > N \cdot \epsilon^2,$$
  with $\epsilon = 2$ enough to do the trick.

Notes:

- Problem: data-dependent sequence of $m \leftarrow m^2 \pmod{N}$ and $m \leftarrow m \cdot c \pmod{N}$.
- Solution:
  - *blind*, i.e., randomise, the exponent [13, Section 10]:
    1. select random $m \in \mathbb{Z}$ and compute
    $$d' = d + m \cdot \Phi(N),$$
    2. compute
    $$\begin{array}{lcll} c^{d'} & \equiv & c^{d+m \cdot \Phi(N)} & \pmod{N} \\ & \equiv & c^d \cdot c^{m \cdot \Phi(N)} & \pmod{N} \\ & \equiv & c^d \cdot (c^m)^{\Phi(N)} & \pmod{N} \\ & \equiv & c^d \cdot 1 & \pmod{N} \\ & \equiv & c^d & \pmod{N} \end{array}$$
  and/or
  - *blind*, i.e., randomise, the base [13, Section 10]:
    1. select random $m, m' \in \mathbb{Z}_N^*$ st.
    $$\frac{1}{m'} \equiv m^d \pmod{N},$$
    2. compute
    $$\begin{array}{lcll} m' \cdot ((m \cdot c)^d) & \equiv & m' \cdot m^d \cdot c^d & \pmod{N} \\ & \equiv & m' \cdot \frac{1}{m'} \cdot c^d & \pmod{N} \\ & \equiv & c^d & \pmod{N} \end{array}$$

Notes:

Part 2.2: in practice (9)

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

- **Problem**: Δ can be used to influence, e.g., corrupt

$$\sigma_p = m^{d \bmod p-1} \pmod{p}$$

  or

$$\sigma_q = m^{d \bmod q-1} \pmod{q}$$

  i.e., "small" exponentiations during CRT.

- **Solution** [21]:
  1. select a random $r \in \mathbb{Z}$,
  2. compute the exponentiation step of the CRT as

$$\begin{array}{rcll} \sigma_p & = & m^{d \bmod \Phi(p \cdot r)} & \pmod{p \cdot r} \\ \sigma_q & = & m^{d \bmod \Phi(q \cdot r)} & \pmod{q \cdot r} \end{array}$$

  3. if

$$\sigma_p \not\equiv \sigma_q \pmod{r}$$

     then abort,
  4. otherwise apply the recombination step to

$$\begin{array}{ll} \sigma_p & \pmod{p} \\ \sigma_q & \pmod{q} \end{array}$$

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. 157).

[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, 157).

[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. 157).

[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. 157).

[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] J.-J. Quisquater G. Hachez. "Montgomery Exponentiation with no Final Subtractions: Improved Results". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 1965. Springer-Verlag, 2000, pp. 293–301 (see p. 149).

[9] S. Gueron. "Enhanced Montgomery Multiplication". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 2523. Springer-Verlag, 2002, pp. 46–56 (see p. 149).

[10] 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).

[11] 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. 157).

Notes:

References

[12] 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).

[13] P.C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology (CRYPTO)*. LNCS 1109. https://doi.org/10.1007/3-540-68697-5_9. Springer-Verlag, 1996, pp. 104–113 (see p. 151).

[14] P.C. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis". In: *Advances in Cryptology (CRYPTO)*. LNCS 1666. https://doi.org/10.1007/0-387-23483-7_110. Springer-Verlag, 1999, pp. 388–397 (see pp. 11, 13).

[15] P.C. Kocher et al. "Introduction to differential power analysis". In: *Journal of Cryptographic Engineering (JCEN)* 1.1 (2011), pp. 5–27 (see p. 157).

[16] 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).

[17] 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).

[18] C.D. Walter. "Montgomery Exponentiation Needs No Final Subtractions". In: *IEE Electronics Letters* 35.21 (1999), pp. 1831–1832 (see p. 149).

[19] C.D. Walter. "Montgomery's Multiplication Technique: How to Make It Smaller and Faster". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 1717. Springer-Verlag, 1999, pp. 80–93 (see p. 149).

[20] 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. 157).

[21] A. Shamir. *Method and Apparatus for protecting public key schemes from timing and fault attacks*. U.S. Patent 5,991,415. URL: http://www.google.com/patents/US5991415 (see p. 153).

Notes: