Bricklayer Attack: A Side-Channel Analysis on the ChaCha Quarter Round

Alexandre Adomnicai [1,3] Jacques J.A. Fournier [2] Laurent Masson [1]

[1] Trusted Objects

[2] CEA-Leti

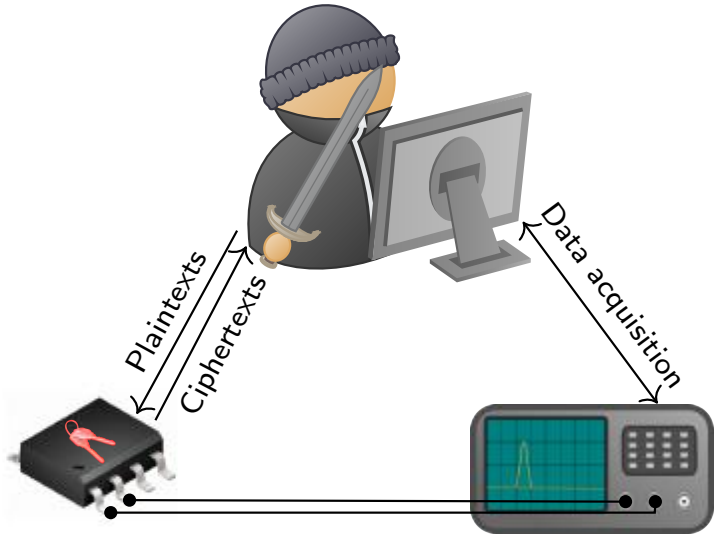
[3] EMSE

INDOCRYPT 2017

Chennai, December 13th

Side-Channel Attacks

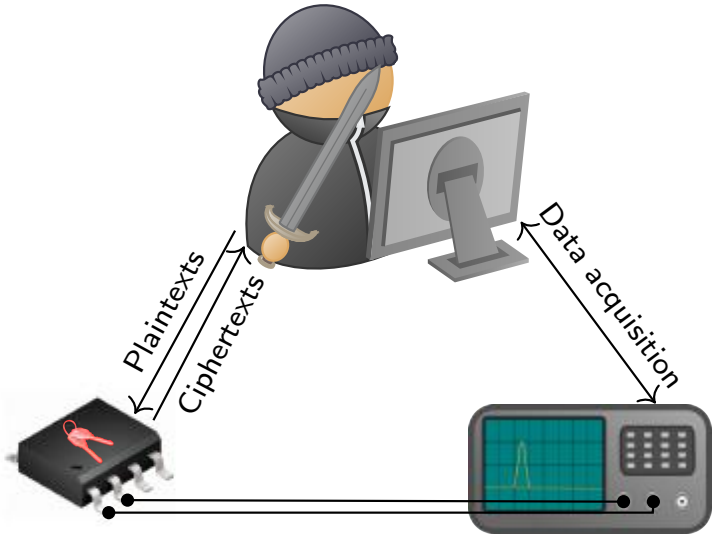
- Cryptographic primitives are designed to be finally executed on a **physical system**.
- **The physical characteristics** of the computing platform produce **side effects** depending on the processed data
  - Power consumption
  - Electromagnetic emanations
  - Time execution
  - Sound ...



TRUSTED OBJECTS

Side-Channel Attacks

- Cryptographic primitives are designed to be finally executed on a **physical system**.
- **The physical characteristics** of the computing platform produce **side effects** depending on the processed data
  - Power consumption
  - Electromagnetic emanations
  - Time execution
  - Sound ...
- One can **measure** these side effects to get information on the processed values during **sensitive operations**
- Using an appropriate **leakage model**, one can recover the **secrets** involved in calculations



TRUSTED OBJECTS

Selection Functions

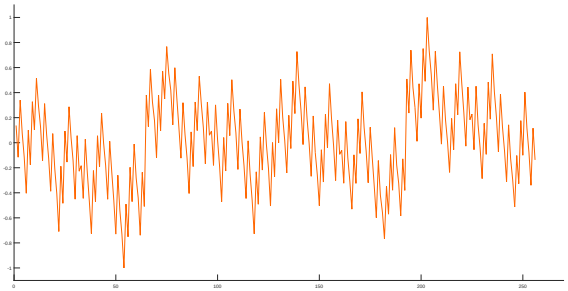
- Differential Power/Electromagnetic analyses target an **intermediate state** y which depends on a **known input** x and a **secret** k.

- This value is defined by a **selection function** $\varphi(x, k) = y$.

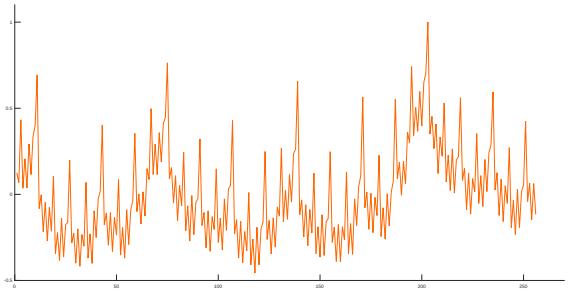
TRUSTED OBJECTS

Selection Functions

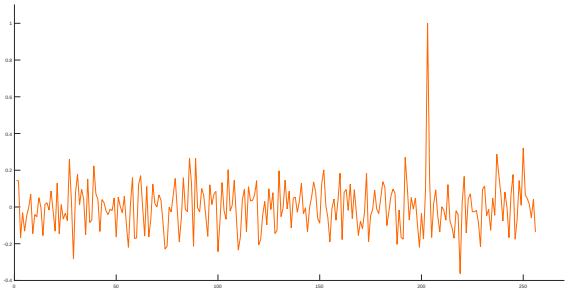
- Differential Power/Electromagnetic analyses target an **intermediate state** y which depends on a **known input** x and a **secret** k.
- This value is defined by a **selection function** $\varphi(x, k) = y$.
- **High non-linearity** is a valuable property as it ensures a good **distinguishability** between correct and incorrect key guesses.



$\varphi(x, k) = x \oplus k$



$\varphi(x, k) = x \boxplus k$



$\varphi(x, k) = \mathrm{AES}_{\mathrm{Sbox}}(x \oplus k)$

Simulation of **Correlation Power Analyses** (CPA) in the Hamming Weight model

TRUSTED OBJECTS

The ChaCha Family of Stream Ciphers

- **ChaCha** is a family of stream ciphers introduced by Daniel J. Bernstein in 2008.
- ChaCha is based on **Salsa20** (eSTREAM portfolio) while **improving diffusion without performance hit**.

TRUSTED OBJECTS

The ChaCha Family of Stream Ciphers

- **ChaCha** is a family of stream ciphers introduced by Daniel J. Bernstein in 2008.
- ChaCha is based on **Salsa20** (eSTREAM portfolio) while **improving diffusion without performance hit**.
- ChaCha20 has been **widely adopted** in practice
  - **Android phones** (ChaCha20-Poly1305 AEAD used in TLS with Chrome)
  - **Apple HomeKit for IoT devices** (ChaCha20-Poly1305 AEAD with HKDF-SHA-512 derived keys)
  - **Linux kernel 4.8+** (`/dev/urandom` based on ChaCha20)
  - **OpenBSD** (ChaCha20 now replaces RC4 for pseudo-random number generator)
  - Numerous security protocols (**TLS**, **SSH**, **IPsec**, ...)

TRUSTED OBJECTS

How to Dance the ChaCha

- Operate like an iterative **512-bit** block cipher using **CTR mode**

TRUSTED OBJECTS

How to Dance the ChaCha

- Operate like an iterative **512-bit** block cipher using **CTR mode**
- The internal state consists in a 4×4 **matrix of 32-bit elements**

‘expa’	‘nd 3’	‘2-by’	‘te k’
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
$\texttt{nonce}_0$	$\texttt{nonce}_1$	$\texttt{nonce}_2$	$\texttt{nonce}_3$

Initial State

TRUSTED OBJECTS

How to Dance the ChaCha

- Operate like an iterative **512-bit** block cipher using **CTR mode**
- The internal state consists in a 4×4 **matrix of 32-bit elements**
- Every round is divided in **quarter rounds** (QR)

‘expa’	‘nd 3’	‘2-by’	‘te k’
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
$\texttt{nonce}_0$	$\texttt{nonce}_1$	$\texttt{nonce}_2$	$\texttt{nonce}_3$

Initial State

TRUSTED OBJECTS

How to Dance the ChaCha

- Operate like an iterative **512-bit** block cipher using **CTR mode**
- The internal state consists in a 4×4 **matrix of 32-bit elements**
- Every round is divided in **quarter rounds** (QR)
- QRs only use **A**dditions, **R**otations and **X**ORs: **ARX**-based cipher

'expa'	'nd 3'	'2-by'	'te k'
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
nonce_0	nonce_1	nonce_2	nonce_3

Initial State

```
a ⊞= b;   d ⊕= a;   d ⋘= 16;
c ⊞= d;   b ⊕= c;   b ⋘= 12;
a ⊞= b;   d ⊕= a;   d ⋘= 8;
c ⊞= d;   b ⊕= c;   b ⋘= 7;
```

QR(a,b,c,d) pseudo code

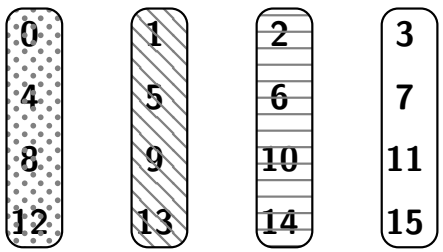
TRUSTED OBJECTS

How to Dance the ChaCha

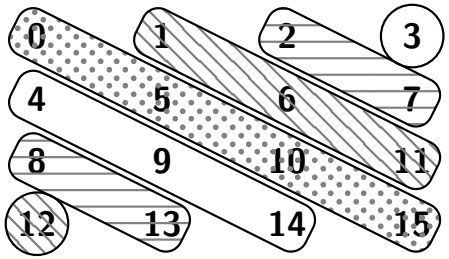
- Operate like an iterative **512-bit** block cipher using **CTR mode**
- The internal state consists in a 4×4 **matrix of 32-bit elements**
- Every round is divided in **quarter rounds** (QR)
- QRs only use **A**dditions, **R**otations and **X**ORs: **ARX**-based cipher
- If the round number is **odd/even** QRs are applied on **columns/diagonals**

'expa'	'nd 3'	'2-by'	'te k'
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
nonce_0	nonce_1	nonce_2	nonce_3

Initial State



(a) Even round



(b) Odd round

```
a ⊞= b;   d ⊕= a;   d ⋘= 16;
c ⊞= d;   b ⊕= c;   b ⋘= 12;
a ⊞= b;   d ⊕= a;   d ⋘= 8;
c ⊞= d;   b ⊕= c;   b ⋘= 7;
```

QR(a,b,c,d) pseudo code

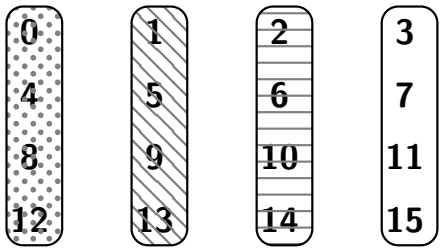
TRUSTED OBJECTS

How to Dance the ChaCha

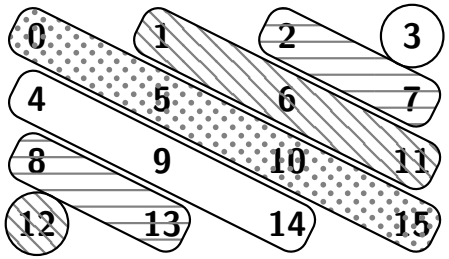
- Operate like an iterative **512-bit** block cipher using **CTR mode**
- The internal state consists in a 4×4 **matrix of 32-bit elements**
- Every round is divided in **quarter rounds** (QR)
- QRs only use **A**dditions, **R**otations and **X**ORs: **ARX**-based cipher
- If the round number is **odd/even** QRs are applied on **columns/diagonals**
- After the last round, the keystream is obtained **by adding** the current state with the initial one

'expa'	'nd 3'	'2-by'	'te k'
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
nonce_0	nonce_1	nonce_2	nonce_3

Initial State



(a) Even round



(b) Odd round

```
a ⊞= b;   d ⊕= a;   d ⋘= 16;
c ⊞= d;   b ⊕= c;   b ⋘= 12;
a ⊞= b;   d ⊕= a;   d ⋘= 8;
c ⊞= d;   b ⊕= c;   b ⋘= 7;
```

QR(a,b,c,d) pseudo code

Attack published at DATE 2017

- All key words are directly involved during the **first column round**.

‘expa’	‘nd 3’	‘2-by’	‘te k’
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
$\texttt{nonce}_0$	$\texttt{nonce}_1$	$\texttt{nonce}_2$	$\texttt{nonce}_3$

TRUSTED OBJECTS

Attack published at DATE 2017

- All key words are directly involved during the **first column round**.
- They interact with the only changing variable: **the nonce**.

‘expa’	‘nd 3’	‘2-by’	‘te k’
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
$\texttt{nonce}_0$	$\texttt{nonce}_1$	$\texttt{nonce}_2$	$\texttt{nonce}_3$

`quarter_round(‘expa’, `k_0`, `k_4`, `$\texttt{nonce}_0$`)`

1: $a \leftarrow$ ‘expa’ $\boxplus\ k_0$
2: $d \leftarrow a \oplus \texttt{nonce}_0$ ▷ k_0 recovery
3: $d \leftarrow d \lll 16$
4: $c \leftarrow d \boxplus k_4$ ▷ k_0 & k_4 recovery
5: $b \leftarrow c \oplus k_0$
6: $b \leftarrow b \lll 12$
7: $a \leftarrow a \boxplus b$
8: $d \leftarrow d \oplus a$
9: $d \leftarrow d \lll 8$
10: $c \leftarrow c \boxplus d$
11: $b \leftarrow b \oplus c$
12: $b \leftarrow b \lll 7$

TRUSTED OBJECTS

Attack published at DATE 2017

- All key words are directly involved during the **first column round**.
- They interact with the only changing variable: **the nonce**.
- The entire key can be recovered using **power/electromagnetic analyses** [2].

‘expa’	‘nd 3’	‘2-by’	‘te k’
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
$\texttt{nonce}_0$	$\texttt{nonce}_1$	$\texttt{nonce}_2$	$\texttt{nonce}_3$

`quarter_round(‘expa’, `k_0`, `k_4`, `$\texttt{nonce}_0$`)`

1: $a \leftarrow \text{‘expa’} \boxplus k_0$
2: $d \leftarrow a \oplus \texttt{nonce}_0$ ▷ k_0 recovery
3: $d \leftarrow d \lll 16$
4: $c \leftarrow d \boxplus k_4$ ▷ k_0 & k_4 recovery
5: $b \leftarrow c \oplus k_0$
6: $b \leftarrow b \lll 12$
7: $a \leftarrow a \boxplus b$
8: $d \leftarrow d \oplus a$
9: $d \leftarrow d \lll 8$
10: $c \leftarrow c \boxplus d$
11: $b \leftarrow b \oplus c$
12: $b \leftarrow b \lll 7$

TRUSTED OBJECTS

Attack published at DATE 2017

- All key words are directly involved during the **first column round**.
- They interact with the only changing variable: **the nonce**.
- The entire key can be recovered using **power/electromagnetic analyses** [2].
- $k_{0,1,2,3}$ are retrieved using $\varphi(x, k) = x \oplus k$

‘expa’	‘nd 3’	‘2-by’	‘te k’
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
$\texttt{nonce}_0$	$\texttt{nonce}_1$	$\texttt{nonce}_2$	$\texttt{nonce}_3$

`quarter_round(‘expa’,` k_0`,` k_4`,` $\texttt{nonce}_0$`)`

1: $a \leftarrow$ ‘expa’ $\boxplus k_0$
2: $d \leftarrow a \oplus \texttt{nonce}_0$ ▷ k_0 recovery
3: $d \leftarrow d \lll 16$
4: $c \leftarrow d \boxplus k_4$ ▷ k_0 & k_4 recovery
5: $b \leftarrow c \oplus k_0$
6: $b \leftarrow b \lll 12$
7: $a \leftarrow a \boxplus b$
8: $d \leftarrow d \oplus a$
9: $d \leftarrow d \lll 8$
10: $c \leftarrow c \boxplus d$
11: $b \leftarrow b \oplus c$
12: $b \leftarrow b \lll 7$

TRUSTED OBJECTS

Attack published at DATE 2017

- All key words are directly involved during the **first column round**.
- They interact with the only changing variable: **the nonce**.
- The entire key can be recovered using **power/electromagnetic analyses** [2].
- $k_{0,1,2,3}$ are retrieved using $\varphi(x, k) = x \oplus k$
- $k_{4,5,6,7}$ are retrieved using $\varphi(x, k) = x \boxplus k$

‘expa’	‘nd 3’	‘2-by’	‘te k’
k_0	k_1	k_2	k_3
k_4	k_5	k_6	k_7
$\texttt{nonce}_0$	$\texttt{nonce}_1$	$\texttt{nonce}_2$	$\texttt{nonce}_3$

`quarter_round(‘expa’, `k_0`, `k_4`, `$\texttt{nonce}_0$`)`

1: $a \leftarrow$ ‘expa’ $\boxplus\ k_0$
2: $d \leftarrow a \oplus \texttt{nonce}_0$ ▷ k_0 recovery
3: $d \leftarrow d \lll 16$
4: $c \leftarrow d \boxplus k_4$ ▷ k_0 & k_4 recovery
5: $b \leftarrow c \oplus k_0$
6: $b \leftarrow b \lll 12$
7: $a \leftarrow a \boxplus b$
8: $d \leftarrow d \oplus a$
9: $d \leftarrow d \lll 8$
10: $c \leftarrow c \boxplus d$
11: $b \leftarrow b \oplus c$
12: $b \leftarrow b \lll 7$

TRUSTED OBJECTS

Practical experiments

- All practical experiments were done on an **ARM Cortex-M3** clocked at 24MHz using
  - Langer HF-U 5 near-field probe (30 MHz - 3 GHz)
  - Langer PA 303 BNC preamplifier (+ 30dB)
  - LeCroy WaveSurfer 10 oscilloscope (10GS/s)

Device Under Test

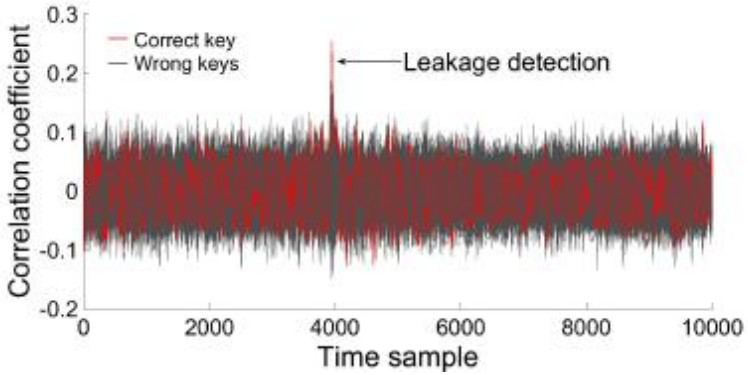
TRUSTED OBJECTS

Practical experiments

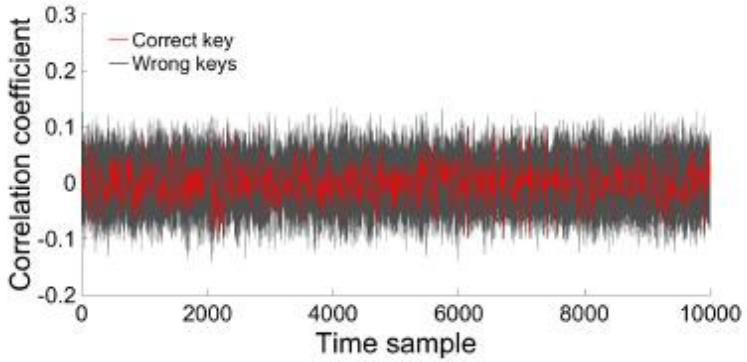
- All practical experiments were done on an **ARM Cortex-M3** clocked at 24MHz using
  - Langer HF-U 5 near-field probe (30 MHz - 3 GHz)
  - Langer PA 303 BNC preamplifier (+ 30dB)
  - LeCroy WaveSurfer 10 oscilloscope (10GS/s)

- Application of both attacks on two different ChaCha20 implementations
  - OpenSSL (1.0.1f) compiled using the GNU ARM C compiler (5.06)
  - Homemade ARM assembly

Device Under Test



(a) C compiled



(b) ARM Assembly

TRUSTED OBJECTS

Assembly VS C compiled

-O0 Compilation

```
LDR     r1,[sp,#0x10]
LDR     r0,[sp,#0x00]
ADD     r0,r0,r1
STR     r0,[sp,#0x00]
LDR     r1,[sp,#0x00]
LDR     r0,[sp,#0x30]
EORS    r0,r0,r1
LSLS    r1,r0,#16
LDR     r2,[sp,#0x00]
LDR     r0,[sp,#0x30]
EORS    r0,r0,r2
ORR     r0,r1,r0,LSR #16
STR     r0,[sp,#0x30]
LDR     r1,[sp,#0x30]
LDR     r0,[sp,#0x20]
ADD     r0,r0,r1
STR     r0,[sp,#0x20]
LDR     r1,[sp,#0x20]
LDR     r0,[sp,#0x10]
EORS    r0,r0,r1
LSLS    r1,r0,#12
LDR     r2,[sp,#0x20]
LDR     r0,[sp,#0x10]
EORS    r0,r0,r2
ORR     r0,r1,r0,LSR #20
STR     r0,[sp,#0x10]

        ...
```

-O3 Compilation

```
LDR     r1,[sp,#0x10]
LDR     r0,[sp,#0x00]
ADD     r0,r0,r1
STR     r0,[sp,#0x00]
LDR     r1,[sp,#0x00]
LDR     r0,[sp,#0x30]
EORS    r0,r0,r1
ROR     r0,r0,#16
STR     r0,[sp,#0x30]
LDR     r1,[sp,#0x30]
LDR     r0,[sp,#0x20]
ADD     r0,r0,r1
STR     r0,[sp,#0x20]
LDR     r1,[sp,#0x20]
LDR     r0,[sp,#0x10]
EORS    r0,r0,r1
ROR     r0,r0,#20
STR     r0,[sp,#0x10]

        ...
```

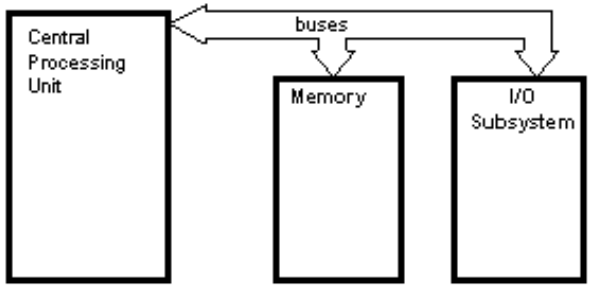
ARM Assembly

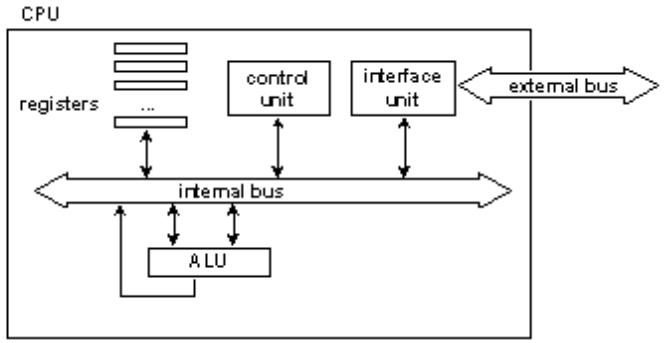
```
LDR     r1, [r0]
LDR     r2, [r0, #16]
LDR     r3, [r0, #32]
LDR     r4, [r0, #48]
ADD     r1, r1, r2
EOR     r4, r4, r1
ROR     r4, r4, #16
ADD     r3, r3, r4
EOR     r2, r2, r3
ROR     r2, r2, #20
ADD     r1, r1, r2
STR     r1, [r0]
EOR     r4, r4, r1
ROR     r4, r4, #24
STR     r4, [r0, #48]
ADD     r3, r3, r4
STR     r3, [r0, #32]
EOR     r2, r2, r3
ROR     r2, r2, #25
STR     r2, [r0, #16]
```

TRUSTED OBJECTS

Information Leakage & Implementation Aspects

- **Load/store** architectures divide instructions into 2 categories
  - **Memory accesses**
  - **Arithmetic Logic Unit** (ALU) operations



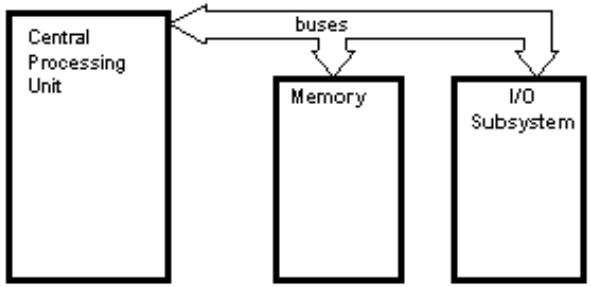


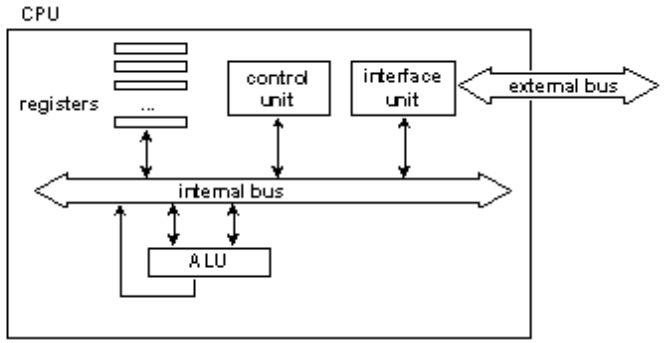
TRUSTED OBJECTS

Information Leakage & Implementation Aspects

- **Load/store** architectures divide instructions into 2 categories
  - **Memory accesses**
  - **Arithmetic Logic Unit** (ALU) operations

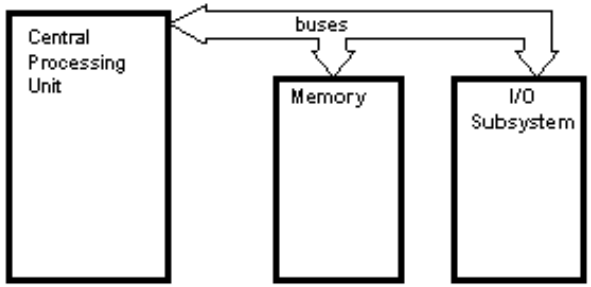
- When a CPU loads values from RAM to registers
  - The memory address is placed on the address bus
  - The data contained at the address is moved to the data bus
  - The data is transferred into a register



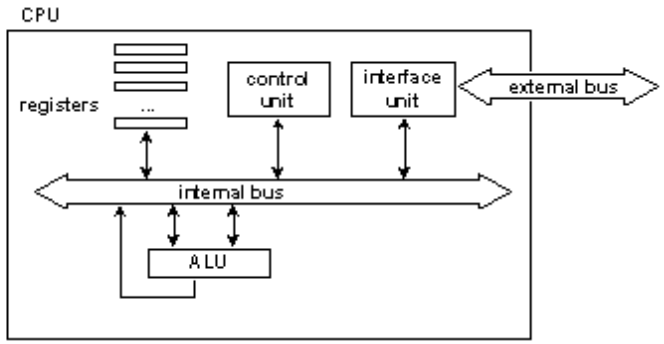


Information Leakage & Implementation Aspects

- **Load/store** architectures divide instructions into 2 categories
  - **Memory accesses**
  - **Arithmetic Logic Unit** (ALU) operations



- When a CPU loads values from RAM to registers
  - The memory address is placed on the address bus
  - The data contained at the address is moved to the data bus
  - The data is transferred into a register



- When a CPU performs ALU operations
  - The operand registers' content are transferred to the ALU
  - The ALU performs the calculation and places the result in the output register

Is it easier to exploit leakages in relation to memory instructions?

TRUSTED OBJECTS

Focusing on Memory Instructions

- Focusing on memory accesses imply to analyze the **whole QR**

TRUSTED OBJECTS

Focusing on Memory Instructions

- Focusing on memory accesses imply to analyze the **whole QR**

- The simplest selection function is defined by focusing the **first `STR` instruction**

$$\varphi_1\left(\mathtt{nonce}_i, k_i \parallel k_{i+4}\right) = \mathtt{nonce}_i \oplus \tilde{k}_i \lll 16 \boxplus k_{i+4} \oplus k_i \lll 12 \boxplus \tilde{k}_i$$

where $\tilde{k}_i = k_i \boxplus \mathtt{constant}_i$

TRUSTED OBJECTS

Focusing on Memory Instructions

- Focusing on memory accesses imply to analyze the **whole QR**

- The simplest selection function is defined by focusing the **first `STR` instruction**

$$\varphi_1\left(\mathtt{nonce}_i, k_i \parallel k_{i+4}\right) = \mathtt{nonce}_i \oplus \tilde{k}_i \lll 16 \boxplus k_{i+4} \oplus k_i \lll 12 \boxplus \tilde{k}_i$$

  where $\tilde{k}_i = k_i \boxplus \mathtt{constant}_i$

- φ_1 implies a side-channel attack on **2 key words at once** (*i.e.* $|\mathcal{K}| = 2^{64}$) $\Rightarrow$ undoable in practice!

Divide & Conquer

- It has been proved there is still a correlation when predicting a **subpart** of the word [4]

TRUSTED OBJECTS

Divide & Conquer

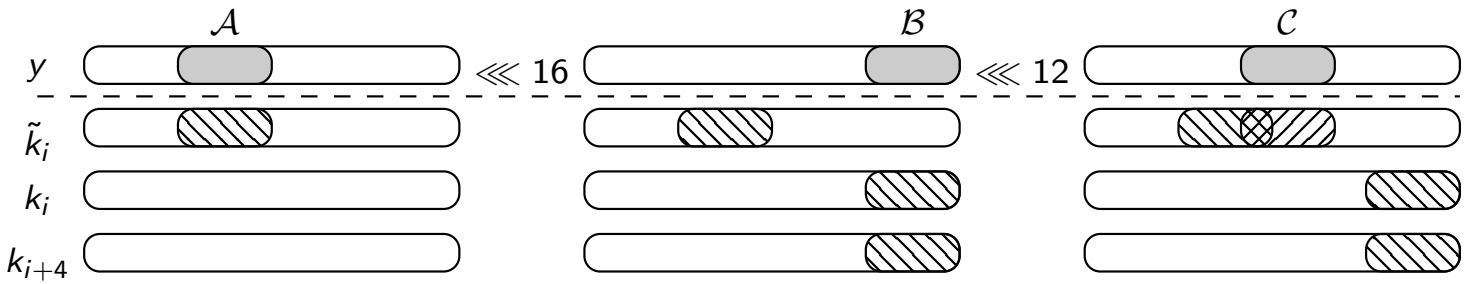
- It has been proved there is still a correlation when predicting a **subpart** of the word [4]
- Targeting n bits of $y = \varphi_1(\texttt{nonce}_i, k_i \parallel k_{i+4})$ does not lead to a complexity equal to 2^{2n}

TRUSTED OBJECTS

Divide & Conquer

- It has been proved there is still a correlation when predicting a **subpart** of the word [4]
- Targeting n bits of $y = \varphi_1(\texttt{nonce}_i, k_i \parallel k_{i+4})$ does not lead to a complexity equal to 2^{2n}
- The key search space depends on the **windows' size** n

$$|\mathcal{K}| = \begin{cases} 2^{4n}, & \text{if } n \leq 4 \\ 2^{3n+4}, & \text{if } 4 \leq n \leq 12 \\ 2^{2n+16}, & \text{if } 13 \leq n \leq 16 \\ 2^{n+32}, & \text{otherwise} \end{cases}$$



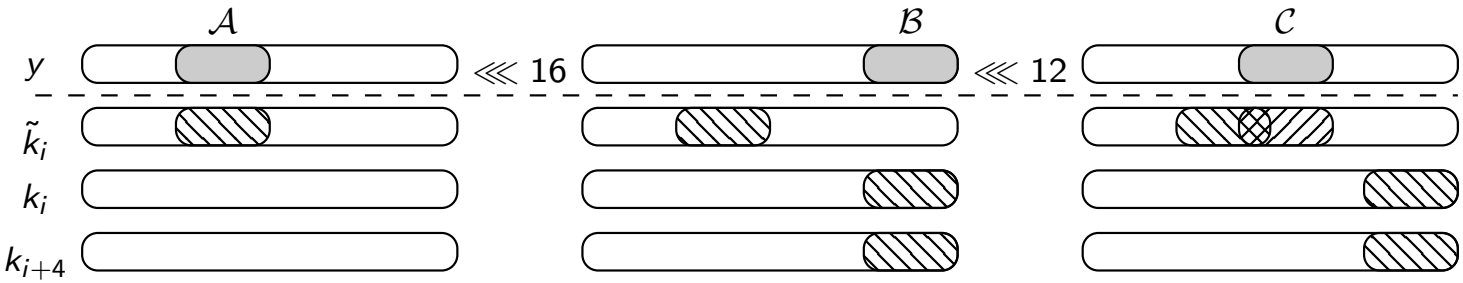
D&C approach on the ChaCha QR, $n = 8$

Divide & Conquer

- It has been proved there is still a correlation when predicting a **subpart** of the word [4]
- Targeting n bits of $y = \varphi_1(\mathtt{nonce}_i, k_i \parallel k_{i+4})$ does not lead to a complexity equal to 2^{2n}
- The key search space depends on the **windows' size** n

$$|\mathcal{K}| = \begin{cases} 2^{4n}, & \text{if } n \leq 4 \\ 2^{3n+4}, & \text{if } 4 \leq n \leq 12 \\ 2^{2n+16}, & \text{if } 13 \leq n \leq 16 \\ 2^{n+32}, & \text{otherwise} \end{cases}$$

- $\varphi_{2,n}\left(\mathtt{nonce}_i, \tilde{k}_i^{\mathcal{A}} \parallel k_i^{\mathcal{B}} \parallel k_{i+4}^{\mathcal{B}} \parallel \tilde{k}_i^{\mathcal{C}}\right) = \mathtt{nonce}_i^{\mathcal{A}} \oplus \tilde{k}_i^{\mathcal{A}} \boxplus_n k_{i+4}^{\mathcal{B}} \oplus k_i^{\mathcal{B}} \boxplus_n \tilde{k}_i^{\mathcal{C}}$



D&C approach on the ChaCha QR, $n = 8$

TRUSTED OBJECTS

Focusing on the QR

- We performed software simulations using the **Hamming Weight model** (without any additional noise) and random nonces

TRUSTED OBJECTS

Focusing on the QR

- We performed software simulations using the **Hamming Weight model** (without any additional noise) and random nonces
- As expected, the right key matches with the highest coefficient but others too $\Rightarrow$ **collisions**!

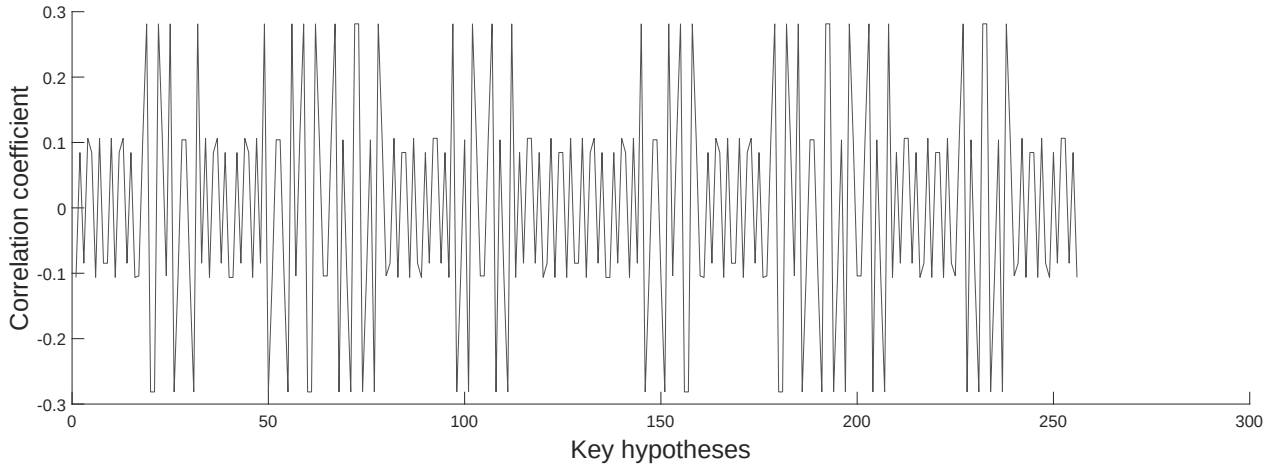
TRUSTED OBJECTS

Focusing on the QR

- We performed software simulations using the **Hamming Weight model** (without any additional noise) and random nonces
- As expected, the right key matches with the highest coefficient but others too $\Rightarrow$ **collisions**!

Proposition

An attack on $\varphi_{2,n}$ returns up to $n \cdot 2^{n+2}$ collisions.



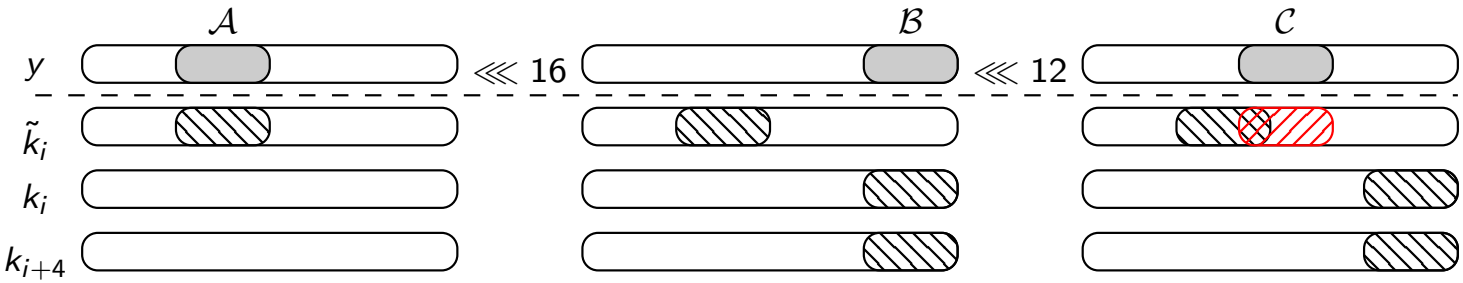
Attack simulation on $\varphi_{2,2}$

TRUSTED OBJECTS

Focusing on the QR

- On top of collisions, $\varphi_{2,n}$ is a victim of **carry propagations**

$$\varphi_{2,n}\left(\mathtt{nonce}_i, \tilde{k}_i^{\mathcal{A}} \parallel k_i^{\mathcal{B}} \parallel k_{i+4}^{\mathcal{B}} \parallel \tilde{k}_i^{\mathcal{C}}\right) = \mathtt{nonce}_i^{\mathcal{A}} \oplus \tilde{k}_i^{\mathcal{A}} \boxplus_n k_{i+4}^{\mathcal{B}} \oplus k_i^{\mathcal{B}} \boxplus_n \tilde{k}_i^{\mathcal{C}}$$

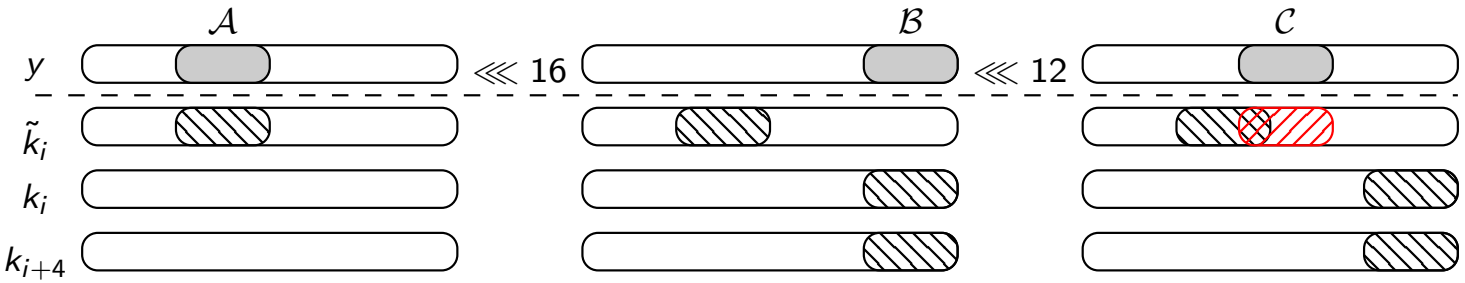


D&C approach on the ChaCha QR, $n = 8$

Focusing on the QR

- On top of collisions, $\varphi_{2,n}$ is a victim of **carry propagations**
- The attack should be run twice: **with** and **without** taking the carry into consideration

$$\varphi_{2,n}\left(\mathtt{nonce}_i, \tilde{k}_i^{\mathcal{A}} \parallel k_i^{\mathcal{B}} \parallel k_{i+4}^{\mathcal{B}} \parallel \tilde{k}_i^{\mathcal{C}}\right) = \mathtt{nonce}_i^{\mathcal{A}} \oplus \tilde{k}_i^{\mathcal{A}} \boxplus_n k_{i+4}^{\mathcal{B}} \oplus k_i^{\mathcal{B}} \boxplus_n \tilde{k}_i^{\mathcal{C}}$$



D&C approach on the ChaCha QR, $n = 8$

Benefits of the Inverse Quarter Round

```
a ⊞= b;   d ⊕= a;   d ⋘= 16;
c ⊞= d;   b ⊕= c;   b ⋘= 12;
a ⊞= b;   d ⊕= a;   d ⋘= 8;
c ⊞= d;   b ⊕= c;   b ⋘= 7;
```

QR(a,b,c,d) pseudo code

```
b ⋙= 7;   b ⊕= c;   c ⊟= d;
d ⋙= 8;   d ⊕= a;   a ⊟= b;
b ⋙= 12;  b ⊕= c;   c ⊟= d;
d ⋙= 16;  d ⊕= a;   a ⊟= b;
```

IQR(a,b,c,d) pseudo code

TRUSTED OBJECTS

Benefits of the Inverse Quarter Round

a ⊞= b;	d ⊕= a;	d ⋘= 16;
c ⊞= d;	b ⊕= c;	b ⋘= 12;
a ⊞= b;	d ⊕= a;	d ⋘= 8;
c ⊞= d;	b ⊕= c;	b ⋘= 7;

QR(a,b,c,d) pseudo code

b ⋙= 7;	b ⊕= c;	c ⊟= d;
d ⋙= 8;	d ⊕= a;	a ⊟= b;
b ⋙= 12;	b ⊕= c;	c ⊟= d;
d ⋙= 16;	d ⊕= a;	a ⊟= b;

IQR(a,b,c,d) pseudo code

- The simplest selection function is defined by

$$\varphi_3\left(b \parallel c \parallel \tilde{d}_i, k_b \parallel k_c\right) = \left(b \boxminus k_b \ggg 7\right) \oplus \left(c \boxminus k_c \ggg 12\right) \oplus \left(c \boxminus k_c \boxminus \tilde{d}_i\right)$$

where $\tilde{d}_i = d_i \boxminus \mathtt{nonce}_i$

Benefits of the Inverse Quarter Round

a ⊞= b;	d ⊕= a;	d ⋘= 16;
c ⊞= d;	b ⊕= c;	b ⋘= 12;
a ⊞= b;	d ⊕= a;	d ⋘= 8;
c ⊞= d;	b ⊕= c;	b ⋘= 7;

QR(a,b,c,d) pseudo code

b ⋙= 7;	b ⊕= c;	c ⊟= d;
d ⋙= 8;	d ⊕= a;	a ⊟= b;
b ⋙= 12;	b ⊕= c;	c ⊟= d;
d ⋙= 16;	d ⊕= a;	a ⊟= b;

IQR(a,b,c,d) pseudo code

- The simplest selection function is defined by

$$\varphi_3\left(b \parallel c \parallel \tilde{d}_i, k_b \parallel k_c\right) = \left(b \boxminus k_b \ggg 7\right) \oplus \left(c \boxminus k_c \ggg 12\right) \oplus \left(c \boxminus k_c \boxminus \tilde{d}_i\right)$$

  where $\tilde{d}_i = d_i \boxminus \mathtt{nonce}_i$

- *a* **does not impact** the update of *b*

TRUSTED OBJECTS

Benefits of the Inverse Quarter Round

a ⊞= b;	d ⊕= a;	d ⋘= 16;
c ⊞= d;	b ⊕= c;	b ⋘= 12;
a ⊞= b;	d ⊕= a;	d ⋘= 8;
c ⊞= d;	b ⊕= c;	b ⋘= 7;

QR(a,b,c,d) pseudo code

b ⋙= 7;	b ⊕= c;	c ⊟= d;
d ⋙= 8;	d ⊕= a;	a ⊟= b;
b ⋙= 12;	b ⊕= c;	c ⊟= d;
d ⋙= 16;	d ⊕= a;	a ⊟= b;

IQR(a,b,c,d) pseudo code

- The simplest selection function is defined by

$$\varphi_3\left(b \parallel c \parallel \tilde{d}_i, k_b \parallel k_c\right) = \left(b \boxminus k_b \ggg 7\right) \oplus \left(c \boxminus k_c \ggg 12\right) \oplus \left(c \boxminus k_c \boxminus \tilde{d}_i\right)$$

  where $\tilde{d}_i = d_i \boxminus \texttt{nonce}_i$

- *a* **does not impact** the update of *b*

- The probability *p* of a **carry propagation** can be estimated

$$p = \mathbb{P}\left(k_b^{[0,x[} > b^{[0,x[}\right) = \frac{2^x - \left(b^{[0,x[} + 1\right)}{2^x}$$

TRUSTED OBJECTS

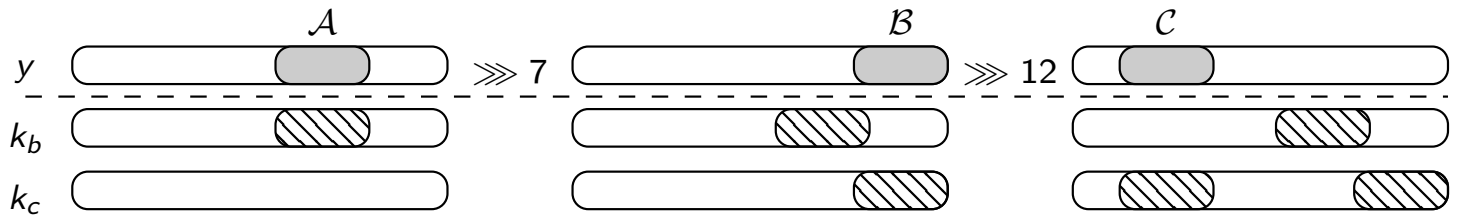
Benefits of the Inverse Quarter Round

- Discarding rotations results in

$$\varphi_{4,n}\left(b \parallel c \parallel \tilde{d}_i, k_b^{\mathcal{A}} \parallel k_c^{\mathcal{B}} \parallel k_c^{\mathcal{C}}\right) = \left(b^{\mathcal{A}} \boxplus_n k_b^{\mathcal{A}}\right) \oplus \left(c^{\mathcal{B}} \boxplus_n k_c^{\mathcal{B}}\right) \oplus \left(c^{\mathcal{C}} \boxplus_n k_c^{\mathcal{C}} \boxplus_n \tilde{d}_i^{\mathcal{C}}\right)$$

- **Smaller** key search space than $\varphi_{2,n}$

$$|\mathcal{K}| = \begin{cases} 2^{3n}, & \text{if } n \leq 12 \\ 2^{2n+12}, & \text{if } 12 \leq n \leq 20 \\ 2^{n+32}, & \text{otherwise} \end{cases}$$



D&C approach on the ChaCha IQR, $n = 8$

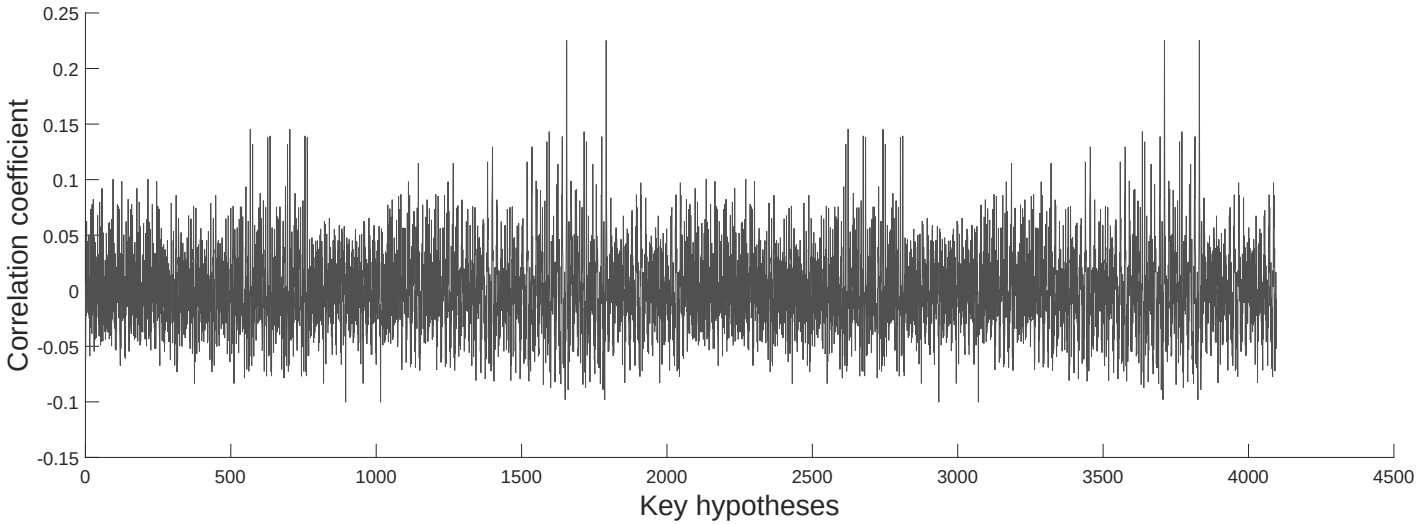
TRUSTED OBJECTS

Benefits of the Inverse Quarter Round

- Carries were taken into account if $p \geq \frac{3}{4}$
- Much **less collisions** than $\varphi_{2,n}$

Proposition

An attack on $\varphi_{4,n}$ returns 4 collisions.



Attack simulation on $\varphi_{4,4}$

TRUSTED OBJECTS

Bricklayer Attack

- **Sequential** approach
- Taking advantage of **windows previously recovered** instead of executing attacks in parallel

TRUSTED OBJECTS

Bricklayer Attack

- **Sequential** approach
- Taking advantage of **windows previously recovered** instead of executing attacks in parallel
- The **carry estimation is only necessary during the first attack** $\Rightarrow$ especially interesting for $\varphi_{2,n}$

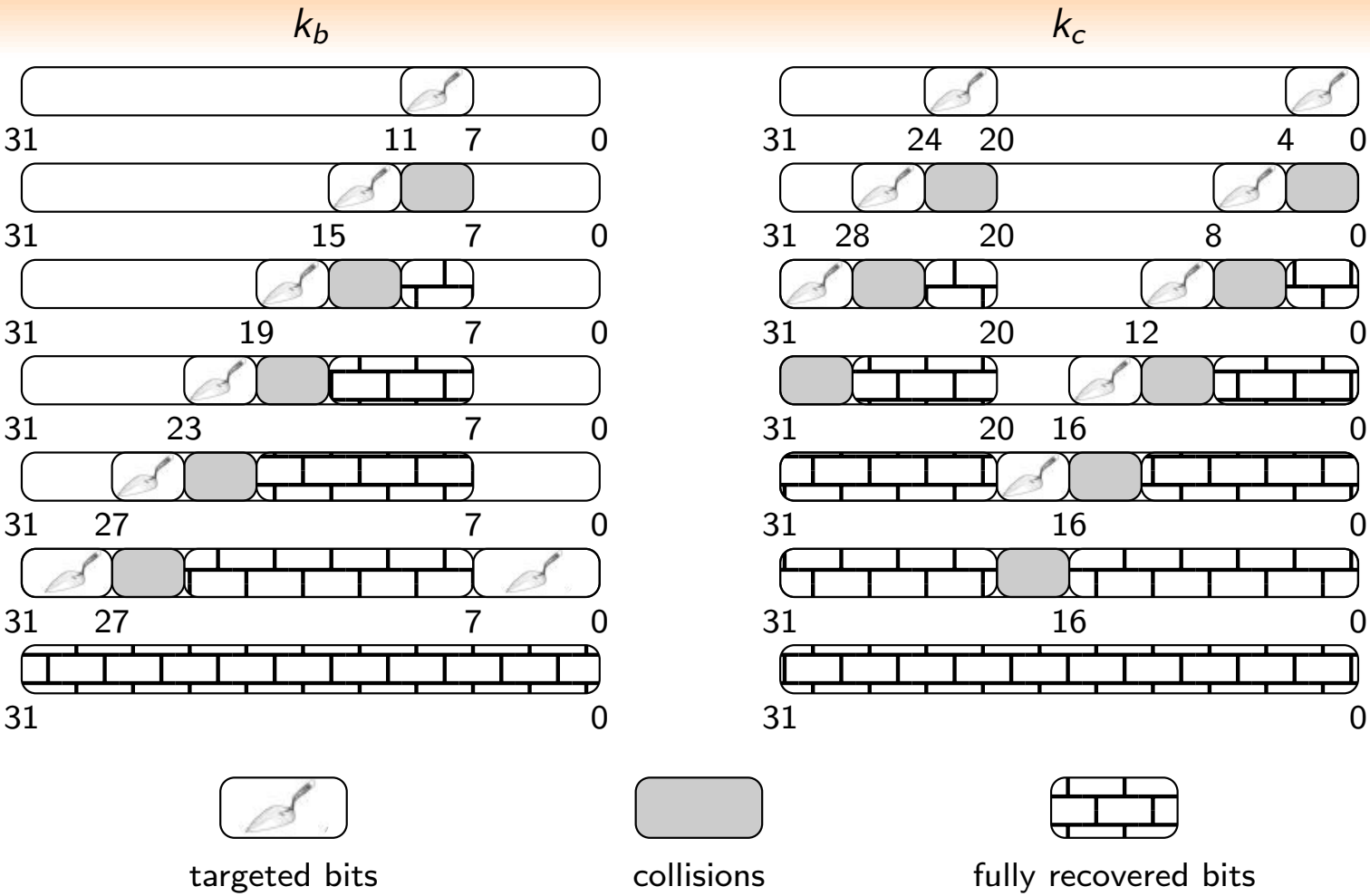
TRUSTED OBJECTS

Bricklayer Attack

- **Sequential** approach
- Taking advantage of **windows previously recovered** instead of executing attacks in parallel
- The **carry estimation is only necessary during the first attack** $\Rightarrow$ especially interesting for $\varphi_{2,n}$
- Collision bits' positions are changed at each attack $\Rightarrow$ some collisions cancelled
- In the case of $\varphi_{4,n}$, collisions only depends on MSBs $\Rightarrow$ **the bricklayer approach allows the correct collision to stand out**

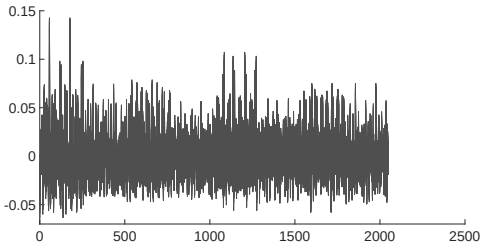
TRUSTED OBJECTS

Bricklayer Attack Overview

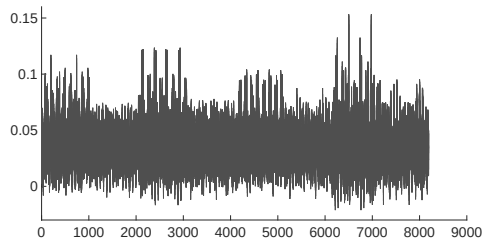


Bricklayer attack example on IQR

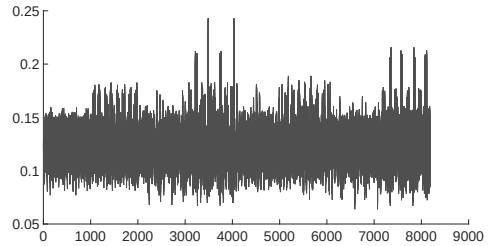
Practical Experiments



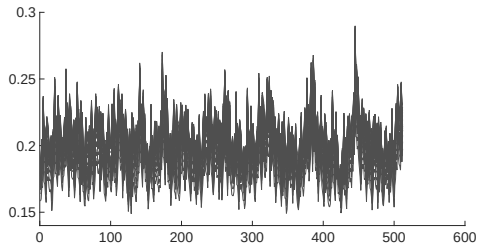
$\kappa = k_7^{23\ldots20} \parallel k_7^{3\ldots0} \parallel k_2^{10\ldots7}$



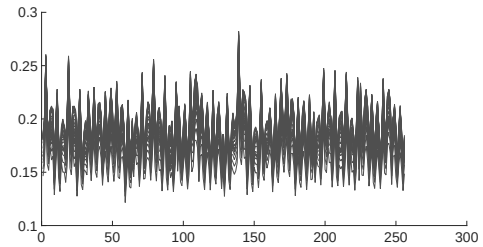
$\kappa = k_7^{27\ldots24} \parallel k_7^{7\ldots4} \parallel k_2^{14\ldots11}$



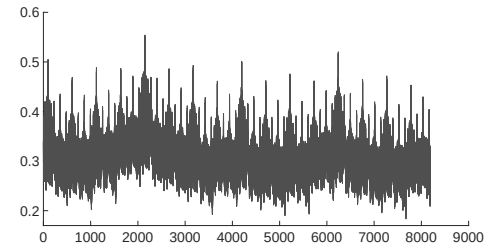
$\kappa = k_7^{31\ldots28} \parallel k_7^{11\ldots8} \parallel k_2^{18\ldots15}$



$\kappa = k_7^{15\ldots12} \parallel k_2^{22\ldots19}$



$\kappa = k_7^{19\ldots16} \parallel k_2^{26\ldots23}$



$\kappa = k_2^{31\ldots27} \parallel k_2^{6\ldots0}$

Figure: Bricklayer attack against k_2 and k_7

Application on Existing Protocols

- φ_1 requires the knowledge of **nonces**
- φ_3 requires the knowledge of **plaintexts + ciphertexts + nonces**

TRUSTED OBJECTS

Application on Existing Protocols

- φ_1 requires the knowledge of **nonces**
- φ_3 requires the knowledge of **plaintexts + ciphertexts + nonces**
- $\texttt{nonce}_{0,\ldots,4} = \text{counter} \parallel \text{IV}$

TRUSTED OBJECTS

Application on Existing Protocols

- φ_1 requires the knowledge of **nonces**
- φ_3 requires the knowledge of **plaintexts + ciphertexts + nonces**
- $\texttt{nonce}_{0,\ldots,4} = \text{counter} \parallel \text{IV}$
- About **TLS**
  - 96-bit IV is picked **randomly** for each session
  - 32-bit counter is the only predictable part $\Rightarrow$ 64 key bits can be recovered at most
  - Protocol-level **countermeasure**

TRUSTED OBJECTS

Application on Existing Protocols

- φ_1 requires the knowledge of **nonces**
- φ_3 requires the knowledge of **plaintexts + ciphertexts + nonces**
- $\texttt{nonce}_{0,\ldots,4} = \text{counter} \parallel \text{IV}$
- About **TLS**
  - 96-bit IV is picked **randomly** for each session
  - 32-bit counter is the only predictable part $\Rightarrow$ 64 key bits can be recovered at most
  - Protocol-level **countermeasure**
- About **SSH**
  - 64-bit IV defined by the **packet sequence number**
  - 64-bit counter reset for each packet
  - Possible to predict the **entire** nonce! $\Rightarrow$ Need of dedicated countermeasures

TRUSTED OBJECTS

Application on Existing Protocols

- φ_1 requires the knowledge of **nonces**
- φ_3 requires the knowledge of **plaintexts + ciphertexts + nonces**
- $\texttt{nonce}_{0,\ldots,4} = \text{counter} \parallel \text{IV}$
- About **TLS**
  - 96-bit IV is picked **randomly** for each session
  - 32-bit counter is the only predictable part $\Rightarrow$ 64 key bits can be recovered at most
  - Protocol-level **countermeasure**
- About **SSH**
  - 64-bit IV defined by the **packet sequence number**
  - 64-bit counter reset for each packet
  - Possible to predict the **entire** nonce! $\Rightarrow$ Need of dedicated countermeasures
- XChaCha construction
  -
  - Implemented in **Libsodium** ($\geq$ `1.0.12`)
  - Extend the nonce size to pick it **at random**
  - The nonce is public and must be **sent with the cryptogram**

TRUSTED OBJECTS

Masking ARX Designs

- Blinding processed values x using **random masks** $r \Rightarrow$ impossible to predict intermediate values

TRUSTED OBJECTS

Masking ARX Designs

- Blinding processed values x using **random masks** $r \Rightarrow$ impossible to predict intermediate values
- ARX designs need both **boolean** ($x' = x \oplus r$) and **arithmetic** ($x' = x \boxplus r$) masking
- Two approaches
  - **Switch** from one masking scheme to the other
  - Perform additions on the **masked values**

TRUSTED OBJECTS

Masking ARX Designs

- Blinding processed values x using **random masks** $r \Rightarrow$ impossible to predict intermediate values
- ARX designs need both **boolean** ($x' = x \oplus r$) and **arithmetic** ($x' = x \boxplus r$) masking
- Two approaches
  - **Switch** from one masking scheme to the other
  - Perform additions on the **masked values**
- **Boolean-to-arithmetic** conversions are cheap while **arithmetic-to-boolean** are very heavy
- **Secure adders** usually rely on arithmetic to boolean conversions $\Rightarrow$ same complexity

	Time	Penalty factor
ChaCha20 unmasked	4 380	1
ChaCha20 with Karroumi *et al.* SecAdd [3]	121 618	28
ChaCha20 with Coron *et al.* SecAdd [1]	93 993	22

Running time in clock cycles to encrypt a 512-bit block using ChaCha20 on an ARM Cortex-M3

TRUSTED OBJECTS

Conclusion & Perspectives

Conlusions

- ARX designs remain vulnerable to **power/electromagnetic** side-channel
- Our practical setup was able to exploit **memory accesses only**
- Introduced the **Bricklayer attack** with simulated & practical measurements
- Harder to attack the QR than its **reverse function**

TRUSTED OBJECTS

Conclusion & Perspectives

Conlusions

- ARX designs remain vulnerable to **power/electromagnetic** side-channel
- Our practical setup was able to exploit **memory accesses only**
- Introduced the **Bricklayer attack** with simulated & practical measurements
- Harder to attack the QR than its **reverse function**

Open Questions

- How could we exploit **ALU operations**? Is **decapping** necessary?
- Can we use these properties to mask a **subset of instructions**?
- Is it possible to implement ChaCha20 in a secure way with **reasonable performances**?

TRUSTED OBJECTS

References

Jean-Sébastien Coron, Johann Großschädl, Mehdi Tibouchi, and Praveen Kumar Vadnala.
Conversion from Arithmetic to Boolean Masking with Logarithmic Complexity, pages 130–149.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

B. Jungk and S. Bhasin.
Don't fall into a trap: Physical side-channel analysis of ChaCha20-Poly1305.
In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*.

Mohamed Karroumi, Benjamin Richard, and Marc Joye.
Addition with Blinded Operands, pages 41–55.
Springer International Publishing, Cham, 2014.

M. Tunstall, N. Hanley, R. McEvoy, C. Whelan, C. Murphy, and W. Marnane.
Correlation Power Analysis of Large Word Sizes.

TRUSTED OBJECTS

Thank you for your attention!

Questions?

TRUSTED OBJECTS