

THE CASCADE: Implementation Results

Julien FRANCO
Airbus CyberSecurity

01/06/2017

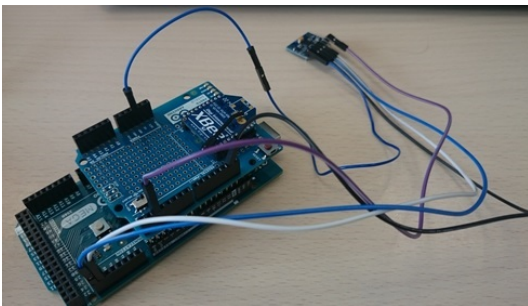
- 1 Proof-Of-Concepts (PoCs)
- 2 Hardware Implementations
- 3 Fault Attacks
- 4 Conclusion and Future Works

PoCs Platforms

- ▶ 2 **Arduinos Mega2560** (256kB Flash, 8kB SRAM, 4kB EEPROM)
- ▶ Each one integrates 1 **XBee module** for radio communications
- ▶ (optional: sensing environmental data – temperature, acceleration, etc.)
- ▶ PoCs implementing LPV-SSSC with dimension = 7
 - ▶ Also available in dimension 40

PoCs Platforms

- ▶ 2 **Arduinos Mega2560** (256kB Flash, 8kB SRAM, 4kB EEPROM)
- ▶ Each one integrates 1 **XBee module** for radio communications
- ▶ (optional: sensing environmental data – temperature, acceleration, etc.)
- ▶ PoCs implementing LPV-SSSC with dimension = 7
 - ▶ Also available in dimension 40



Encryption/Decryption Validation

```

COM32 (Arduino/Genuino Mega or Mega 2560)
- SH[- SLW
Set configuration
- CH _
Command successful
- ID=fx
Command successful
- DH 3
Command successful
- EE 1
Command successful
- KY R
Command successful
- WRM
Command successful
SETUP FINISHED!

Sending data...
Raw Data to Encrypt: Simple Demo!
True Data to Encrypt (HEX values with padding): 8310053696D706C65204465D6F210093
Data encrypted: 990CC4CF41D559C7A1D64983BAC64BA638
- ' '
Data sent
Data received

[ ] Défilement automatique Pas de fin de ligne 9600 baud
  
```

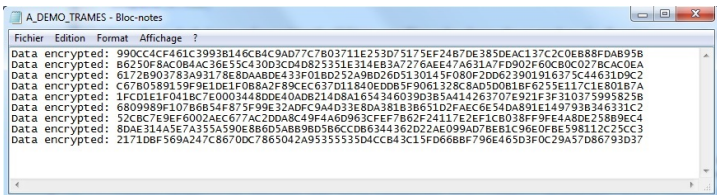
```

COM35 (Arduino/Genuino Mega or Mega 2560)
- SH[- SLW
Set configuration
- CH _
Command successful
- ID=fx
Command successful
- DH 3
Command successful
- EE 1
Command successful
- KY R
Command successful
- WRM
Command successful
SETUP FINISHED!

Packet received
Data received: 990CC4CF41D559C7A1D64983BAC64BA638
Data decrypted: 995CD5D053696D706C65204465D6F2100
Raw Data decrypted: Simple Demo!

[ ] Défilement automatique Pas de fin de ligne 9600 baud
  
```


Traffic Analysis (2/2)



```
A_DEMO_TRAMES - Bloc-notes
Fichier  Edition  Format  Affichage  ?
Data encrypted: 990CC4CF461C3993B146CB4C9AD77C7B03711E253D75175EF24B7DE385DEAC137C2C0E888FDAB958
Data encrypted: B6250F8AC0B4AC36E55C430D3CD4D825351E314EB3A7276AEE47A631A7FD902F60CB0C027BCAC0EA
Data encrypted: 6172B903783A93178E8DAABDE433F01BD252A9BD26D5130145F080F2DD623901916375C44631D9C2
Data encrypted: C67B0589159F9E1DE1F0B8A2F89CEC637D11840EDDB5F9061328C8AD5D0B1BF6255E117C1E801B7A
Data encrypted: 1FCD1E1F0418C7E0003448DDE40ADB214D8A1654346039D3B5A414263707E921F3F3103759958258
Data encrypted: 6809989F107B6B54F875F99E32ADFC9A4D33E8DA381B38651D2FAEC6E54DA891E149793B346331C2
Data encrypted: 52CBC7E9EF6002AEC677AC2DDA8C49F4A6D963CFEF7B62F24117E2EF1CB038FF9FE4A8DE258B9EC4
Data encrypted: 8DAE314A5E7A355A90E8B6D5ABB9BD5B6CCDB6344362D22AE099AD7BE81C96E0FBE598112C25CC3
Data encrypted: 2171D8F569A247C8670DC7865042A9535535D4CCB43C15FD6688F796E465D3F0C29A57D86793D37
```


Summary of the PoC Features

© 2017 Airbus. Confidential. All rights reserved. The reproduction, distribution and use of this document is made in the context of a specific contract and is subject to the terms and conditions of this contract. All rights reserved in the event of the good or a patent, utility model or design.

Summary of the PoC Features

- ▶ **Encryption and Decryption** validated (of course!)

Summary of the PoC Features

- ▶ **Encryption and Decryption** validated (of course!)
- ▶ **Fast synchronization** in case of injected errors
 - ▶ AES-CFB (Cipher Feedback): $128 \times 10 = 1280$ cc
 - ▶ MOUSTIQUE: 96 cc
 - ▶ LPV-SSSC: 40 cc

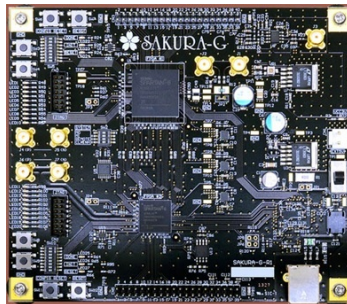
Summary of the PoC Features

- ▶ **Encryption and Decryption** validated (of course!)
- ▶ **Fast synchronization** in case of injected errors
 - ▶ AES-CFB (Cipher Feedback): $128 \times 10 = 1280$ cc
 - ▶ MOUSTIQUE: 96 cc
 - ▶ **LPV-SSSC: 40 cc**
- ▶ **Traffic analysis** hard to handle for attackers:
 - ▶ No synchronization value (**I**nitialization **V**ector – **IV**) sent during the communications
 - ▶ 2 **identical** plaintexts give **different** ciphertexts if the initial state changes between two encryptions
 - ▶ **Good randomness** (confirmed by additional cryptographic analysis)

Summary of the PoC Features

- ▶ **Encryption and Decryption** validated (of course!)
- ▶ **Fast synchronization** in case of injected errors
 - ▶ AES-CFB (Cipher Feedback): $128 \times 10 = 1280$ cc
 - ▶ MOUSTIQUE: 96 cc
 - ▶ **LPV-SSSC: 40 cc**
- ▶ **Traffic analysis** hard to handle for attackers:
 - ▶ No synchronization value (**Initialization Vector – IV**) sent during the communications
 - ▶ 2 **identical** plaintexts give **different** ciphertexts if the initial state changes between two encryptions
 - ▶ **Good randomness** (confirmed by additional cryptographic analysis)
- ▶ It is a kind of **“magic”** when it is synchronizing...

Hardware Platform



SAKURA-G

- ▶ (Japanese) platform dedicated to **side-channel evaluation**
- ▶ Spartan-6 LX75: **11662 Slices** (4 6-input LUTs + 8 1-bit register), **BRAM**: 172 × 18 Kb, **DSP Blocks**: 132

Implemented Versions

- ▶ **“Enc. Full/Dec. Full”**: Full encryption/decryption version in dimension 40 with **781 SBoxes**
- ▶ **“Enc.+Dec. Full”**: **Combination** of previous encryption and decryption components
- ▶ **“Enc./Dec. r1”**: Reduced version with one specific **80 SBoxes** instantiation
- ▶ **“Enc./Dec. r2”**: Reduced version with another specific **80 SBoxes** instantiation
- ▶ **“Folded Version”**: only **1 line** implemented
- ▶ **GRAIN128, Trivium**: **eSTREAM** candidates (synchronous stream ciphers)

Implementation Results

Version	Slices	LUTs	Freq. (MHz)	TP (Mbps)
Enc. Full	3254	10443	97	388
Dec. Full	3174	10280	96	384
Enc.+Dec. Full	4322	11792	68	272
Enc. r1	834	1952	100	400
Dec. r1	880	2100	93	372
Enc. r2	826	2006	108	432
Dec. r2	893	2194	98	392
Folded Version	904	2186	55	5
GRAIN128	22	54	232	232
Trivium	348	1200	141	141

Implementation Details

- ▶ Spartan-6 LX75, **Post-P&R**, **Area** Optim., **TP** (Throughput)

Main Lessons

© 2017 Confidant, Confidentiality A45. All rights reserved. The reproduction, distribution and use of this document is made in the context of a specific contract without express authorization is prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

Main Lessons

- ▶ Full LPV-SSSC version **very costly**

Main Lessons

- ▶ Full LPV-SSSC version **very costly**
- ▶ Usually, **decryption** is a little bit **more costly** than encryption

Main Lessons

- ▶ Full LPV-SSSC version **very costly**
- ▶ Usually, **decryption** is a little bit **more costly** than encryption
- ▶ Both reduced versions have equivalent performances

Main Lessons

- ▶ Full LPV-SSSC version **very costly**
- ▶ Usually, **decryption** is a little bit **more costly** than encryption
- ▶ Both reduced versions have equivalent performances
- ▶ **Folded version** of full version gives **bad results**

Main Lessons

- ▶ Full LPV-SSSC version **very costly**
- ▶ Usually, **decryption** is a little bit **more costly** than encryption
- ▶ Both reduced versions have equivalent performances
- ▶ **Folded version** of full version gives **bad results**
- ▶ Optimized LPV-SSSC is still **2.5× bigger** than Trivium

Main Lessons

- ▶ Full LPV-SSSC version **very costly**
- ▶ Usually, **decryption** is a little bit **more costly** than encryption
- ▶ Both reduced versions have equivalent performances
- ▶ **Folded version** of full version gives **bad results**
- ▶ Optimized LPV-SSSC is still **2.5× bigger** than Trivium
- ▶ Mean occupation per slice: **1 register** ($\approx 12\%$) + **3 LUTs** ($\approx 75\%$)

Main Lessons

- ▶ Full LPV-SSSC version **very costly**
- ▶ Usually, **decryption** is a little bit **more costly** than encryption
- ▶ Both reduced versions have equivalent performances
- ▶ **Folded version** of full version gives **bad results**
- ▶ Optimized LPV-SSSC is still **2.5× bigger** than Trivium
- ▶ Mean occupation per slice: **1 register** ($\approx 12\%$) + **3 LUTs** ($\approx 75\%$)
- ▶ Only **75% reuse** rate when combining encryption and decryption
 - ▶ Variant exists that would increase this reuse rate, but decrease encryption performance

Main Lessons

- ▶ Full LPV-SSSC version **very costly**
- ▶ Usually, **decryption** is a little bit **more costly** than encryption
- ▶ Both reduced versions have equivalent performances
- ▶ **Folded version** of full version gives **bad results**
- ▶ Optimized LPV-SSSC is still **2.5× bigger** than Trivium
- ▶ Mean occupation per slice: **1 register** ($\approx 12\%$) + **3 LUTs** ($\approx 75\%$)
- ▶ Only **75% reuse** rate when combining encryption and decryption
 - ▶ Variant exists that would increase this reuse rate, but decrease encryption performance
- ▶ Relatively **slow** compared to competitors due to longer datapath
 - ▶ Due to XOR sum of the line components of the internal state

One Slide Summary

© 2017 Airbus. Confidential. All rights reserved. The reproduction, distribution and use of this document is made in the context of the company's internal communication and is not intended for external use. All rights reserved in the event of a patent, utility model or design.

One Slide Summary

- ▶ More details on **Jean-Max Dutertre** presentation (thanks Jean-Max!)

One Slide Summary

- ▶ More details on **Jean-Max Dutertre** presentation (thanks Jean-Max!)
- ▶ Passive (e.g., Side-Channel Attacks) vs. Active (e.g., Fault) Attacks

One Slide Summary

- ▶ More details on **Jean-Max Dutertre** presentation (thanks Jean-Max!)
- ▶ Passive (e.g., Side-Channel Attacks) vs. Active (e.g., Fault) Attacks
- ▶ Perturbate the cryptosystem \Rightarrow Faulty Results \Rightarrow **Cryptographic key**

One Slide Summary

- ▶ More details on **Jean-Max Dutertre** presentation (thanks Jean-Max!)
- ▶ Passive (e.g., Side-Channel Attacks) vs. Active (e.g., Fault) Attacks
- ▶ Perturbate the cryptosystem \Rightarrow Faulty Results \Rightarrow **Cryptographic key**
- ▶ Different **ways** to generate a fault
 - ▶ Temperature, Glitches on external clock or supply voltage, Magnetic Attacks, Light Attacks

One Slide Summary

- ▶ More details on **Jean-Max Dutertre** presentation (thanks Jean-Max!)
- ▶ Passive (e.g., Side-Channel Attacks) vs. Active (e.g., Fault) Attacks
- ▶ Perturbate the cryptosystem \Rightarrow Faulty Results \Rightarrow **Cryptographic key**
- ▶ Different **ways** to generate a fault
 - ▶ Temperature, Glitches on external clock or supply voltage, Magnetic Attacks, Light Attacks
- ▶ Permanent vs. Transient Faults
 - ▶ After each fault, the attacker can reset the component and induce the same initial state value

One Slide Summary

- ▶ More details on **Jean-Max Dutertre** presentation (thanks Jean-Max!)
- ▶ Passive (e.g., Side-Channel Attacks) vs. Active (e.g., Fault) Attacks
- ▶ Perturbate the cryptosystem \Rightarrow Faulty Results \Rightarrow **Cryptographic key**
- ▶ Different **ways** to generate a fault
 - ▶ Temperature, Glitches on external clock or supply voltage, Magnetic Attacks, Light Attacks
- ▶ Permanent vs. Transient Faults
 - ▶ After each fault, the attacker can reset the component and induce the same initial state value
- ▶ From random to precise bit errors
 - ▶ Random value faults on a chosen register
 - ▶ Simultaneous 4-bit stuck-at 0 value faults

One Slide Summary

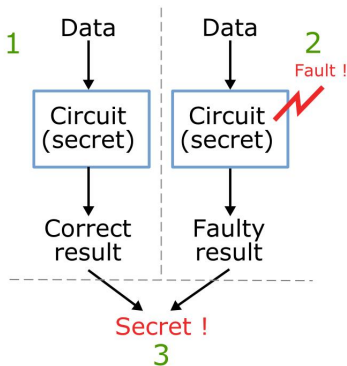
- ▶ More details on **Jean-Max Dutertre** presentation (thanks Jean-Max!)
- ▶ Passive (e.g., Side-Channel Attacks) vs. Active (e.g., Fault) Attacks
- ▶ Perturbate the cryptosystem \Rightarrow Faulty Results \Rightarrow **Cryptographic key**
- ▶ Different **ways** to generate a fault
 - ▶ Temperature, Glitches on external clock or supply voltage, Magnetic Attacks, Light Attacks
- ▶ Permanent vs. Transient Faults
 - ▶ After each fault, the attacker can reset the component and induce the same initial state value
- ▶ From random to precise bit errors
 - ▶ Random value faults on a chosen register
 - ▶ Simultaneous 4-bit stuck-at 0 value faults
- ▶ Single fault (1st order) vs. Multiple faults (2nd order+)

DFA

Definition (DFA)

DFA = Differential Fault Analysis

- pairs of faulty/correct ciphertexts are needed



LPV SSSC Properties

Notations

- ▶ Register R_x containing n nibbles $x[i]$ initialized with an internal IV
- ▶ Key length $\frac{4 \times (n+s+r)}{2}$ bits, with:
 - ▶ s : number of encrypted symbols used in the update function
 - ▶ r : relative degree
- ▶ $x'[i]$: faulted value of $x[i]$ and $\Delta x[i]$ the difference $x[i] + x'[i]$
- ▶ δ_i : initial difference (i.e. $x'_t[i] = x_t[i] + \delta_i$)
- ▶ c'_t : faulted encrypted symbol

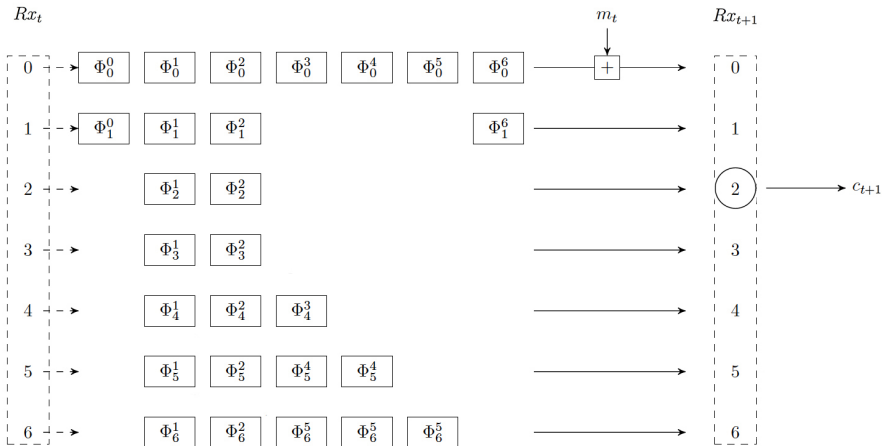
Parameters

- ▶ $n = 7, r = 3, s = 1$, bijective S-Box S
- ▶ Master Key length: 22 bits, derivated into 22 4-bit subkeys SK_i
 - ▶ Toy example with further extensions for $n = 40$ in mind

© 2017 Airbus. Confidential. All rights reserved. The reproduction, distribution and/or use of this document is only allowed in the context of the project in which it was created. All rights reserved in the event of a patent, utility model or design.



Encryption Routine

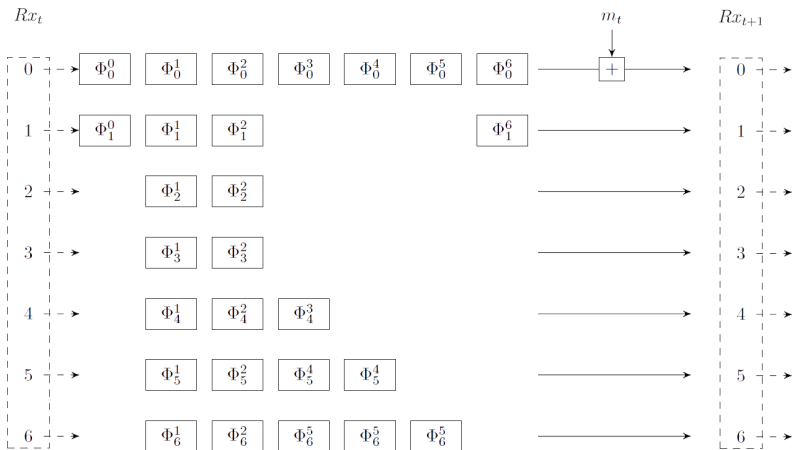


One Round Encryption by LPV-SSSC

DFA Principle

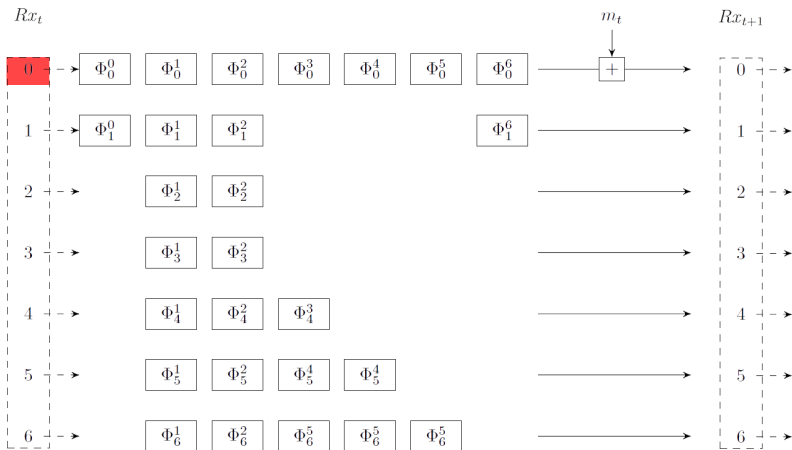
- ▶ Study the differences on the encrypted symbols between a normal and faulted execution, with a fault injected on $x[i]$
 - ▶ System of equations obtained
 - ▶ Extraction of subkey bits by solving the system
- ▶ More precisely, obtain linear equations with the SK_i s
 - ▶ Eliminate some unknown values with the available linear equations
 - ▶ Obtain additional linear equations by injecting the variables computed before in quadratic equations

Fault Diffusion Example



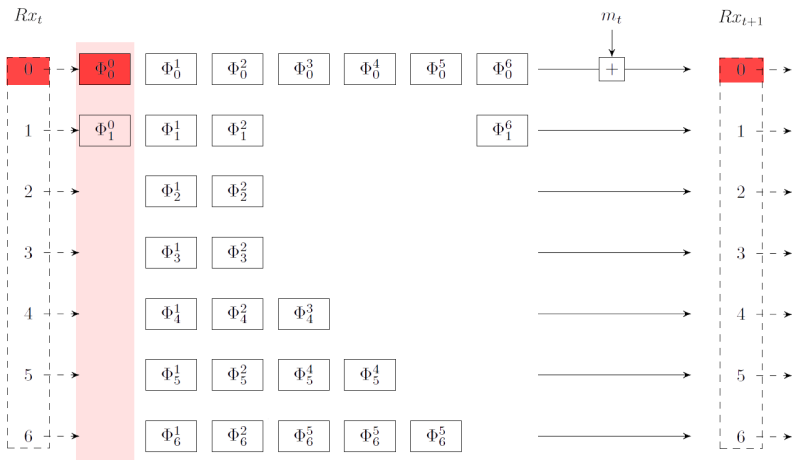
Fault Diffusion on $x_t[0]$

Fault Diffusion Example



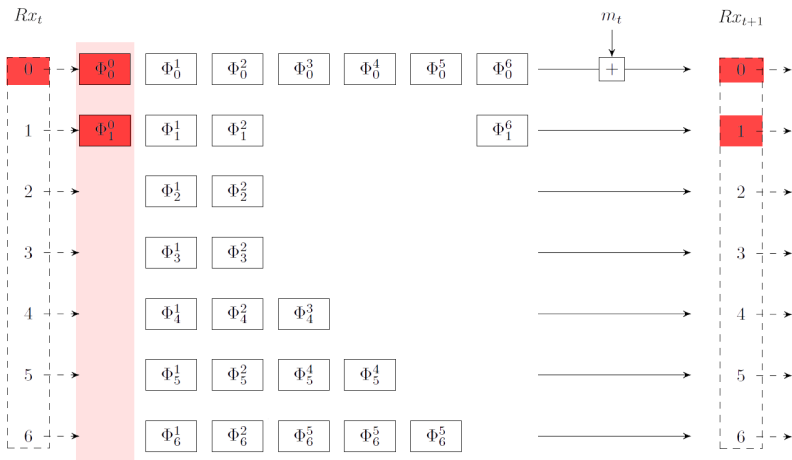
Fault Diffusion on $x_t[0]$

Fault Diffusion Example



Fault Diffusion on $x_t[0]$

Fault Diffusion Example



Fault Diffusion on $x_t[0]$

Example: Fault Injected on $x_t[0]$ (i.e. $\Delta x_t[0] = \delta_0$)

$$x_{t+1}[0] = \sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j]$$

Example: Fault Injected on $x_t[0]$ (i.e. $\Delta x_t[0] = \delta_0$)

$$x_{t+1}[0] = \sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j]$$

$$\Rightarrow \Delta x_{t+1}[0] = \Delta \left(\sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j] \right)$$

Example: Fault Injected on $x_t[0]$ (i.e. $\Delta x_t[0] = \delta_0$)

$$\begin{aligned}x_{t+1}[0] &= \sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j] \\ \Rightarrow \Delta x_{t+1}[0] &= \Delta \left(\sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j] \right) \\ &= \sum_{j=0}^6 S(c_t + SK_j) \cdot \Delta x_t[j]\end{aligned}$$

Example: Fault Injected on $x_t[0]$ (i.e. $\Delta x_t[0] = \delta_0$)

$$\begin{aligned}x_{t+1}[0] &= \sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j] \\ \Rightarrow \Delta x_{t+1}[0] &= \Delta \left(\sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j] \right) \\ &= \sum_{j=0}^6 S(c_t + SK_j) \cdot \Delta x_t[j] \\ &= \delta_0 \cdot S(c_t + SK_0) + \sum_{j=1}^6 S(c_t + SK_j) \cdot 0\end{aligned}$$

Example: Fault Injected on $x_t[0]$ (i.e. $\Delta x_t[0] = \delta_0$)

$$\begin{aligned}x_{t+1}[0] &= \sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j] \\ \Rightarrow \Delta x_{t+1}[0] &= \Delta \left(\sum_{j=0}^6 S(c_t + SK_j) \cdot x_t[j] \right) \\ &= \sum_{j=0}^6 S(c_t + SK_j) \cdot \Delta x_t[j] \\ &= \delta_0 \cdot S(c_t + SK_0) + \sum_{j=1}^6 S(c_t + SK_j) \cdot 0 \\ &= \delta_0 \cdot S(c_t + SK_0)\end{aligned}$$

Example: Fault Injected on $x_t[0]$ (i.e. $x'_t[0] = x_t[0] + \delta_0$)

Cycle t :

$$\Delta x_{t+1}[0] = \delta_0 \cdot S(c_t + SK_0)$$

$$\Delta x_{t+1}[1] = \delta_0$$

$$\Delta x_{t+1}[2] = 0$$

$$\Delta x_{t+1}[3] = 0$$

$$\Delta x_{t+1}[4] = 0$$

$$\Delta x_{t+1}[5] = 0$$

$$\Delta x_{t+1}[6] = 0$$

Fault on $x_t[0]$ (i.e. $x'_t[0] = x_t[0] + \delta_0$)

Cycle $t + 1$:

$$\Delta x_{t+2}[0] = \delta_0 \cdot [S(c_{t+1} + SK_0) \cdot S(c_t + SK_0) + S(c_{t+1} + SK_1)]$$

$$\Delta x_{t+2}[1] = \delta_0 \cdot [S(c_t + SK_0) + 1]$$

$$\Delta x_{t+2}[2] = \delta_0$$

$$\Delta x_{t+2}[3] = \delta_0 \cdot S(c_{t+1} + SK_8)$$

$$\Delta x_{t+2}[4] = \delta_0 \cdot S(c_{t+1} + SK_{10})$$

$$\Delta x_{t+2}[5] = \delta_0 \cdot S(c_{t+1} + SK_{13})$$

$$\Delta x_{t+2}[6] = \delta_0 \cdot S(c_{t+1} + SK_{17})$$

Remarks

© 2017 Confidant Cybersecurity SAS. All rights reserved. The reproduction, distribution and use of this document is made in the context of a specific contract without express authorization is prohibited. Offenses will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

Remarks

1. We don't want to reconstruct the internal state

Remarks

1. We don't want to reconstruct the internal state
2. Subkeys are involved always at the same positions, however the attacker can only access to the 3rd line

Remarks

1. We don't want to reconstruct the internal state
2. Subkeys are involved always at the same positions, however the attacker can only access to the 3rd line

→ Observability problem

Remarks

1. We don't want to reconstruct the internal state
2. Subkeys are involved always at the same positions, however the attacker can only access to the 3rd line

→ Observability problem
3. Once the produced encrypted symbol is faulted, we get $x[i]$ terms in equations Δ after two iterations

Remarks

1. We don't want to reconstruct the internal state
2. Subkeys are involved always at the same positions, however the attacker can only access to the 3rd line
→ Observability problem
3. Once the produced encrypted symbol is faulted, we get $x[i]$ terms in equations Δ after two iterations
→ Limited number of exploitable equations for one fault

SK₀, SK₆, SK₇ Recovery

- ▶ Random fault on $x_t[0]$:

$$\begin{cases} \Delta c_{t+2} = \delta_0 \\ \Delta c_{t+3} = \delta_0 \cdot S(c_t + SK_0) \end{cases}$$

SK₀, SK₆, SK₇ Recovery

- ▶ Random fault on $x_t[0]$:

$$\begin{cases} \Delta c_{t+2} = \delta_0 \\ \Delta c_{t+3} = \delta_0 \cdot S(c_t + SK_0) \end{cases}$$

$$\rightarrow SK_0 = S^{-1}(\Delta c_{t+3} \cdot (\Delta c_{t+2})^{-1}) + c_t$$

SK₀, SK₆, SK₇ Recovery

- ▶ Random fault on $x_t[0]$:

$$\begin{cases} \Delta c_{t+2} = \delta_0 \\ \Delta c_{t+3} = \delta_0 \cdot S(c_t + SK_0) \end{cases}$$

$$\rightarrow SK_0 = S^{-1}(\Delta c_{t+3} \cdot (\Delta c_{t+2})^{-1}) + c_t$$

- ▶ Same principle for SK_6 and SK_7

(SK₁, SK₁₇) and (SK₂, SK₁₈) Recovery

- ▶ For a random fault on $x_t[0]$, at cycle $t + 4$:

(SK₁, SK₁₇) and (SK₂, SK₁₈) Recovery

- ▶ For a random fault on $x_t[0]$, at cycle $t + 4$:

$$S(c_{t+1} + SK_1) + S(c_{t+1} + SK_{17}) = f_0(\delta_0, SK_0, SK_7, c_i, c'_i)$$

(SK₁, SK₁₇) and (SK₂, SK₁₈) Recovery

- ▶ For a random fault on $x_t[0]$, at cycle $t + 4$:

$$S(c_{t+1} + SK_1) + S(c_{t+1} + SK_{17}) = f_0(\delta_0, SK_0, SK_7, c_i, c'_i)$$

→ 16 possible candidates for (SK₁, SK₁₇), because of the Sbox bijectivity

- ▶ Random fault on $x_t[1]$ or on $x_t[6]$: new set of 16 possible couples for (SK₁, SK₁₇).

(SK₁, SK₁₇) and (SK₂, SK₁₈) Recovery

- ▶ For a random fault on $x_t[0]$, at cycle $t + 4$:

$$S(c_{t+1} + SK_1) + S(c_{t+1} + SK_{17}) = f_0(\delta_0, SK_0, SK_7, c_i, c'_i)$$

→ 16 possible candidates for (SK₁, SK₁₇), because of the Sbox bijectivity

- ▶ Random fault on $x_t[1]$ or on $x_t[6]$: new set of 16 possible couples for (SK₁, SK₁₇).

→ By intersecting the sets, two candidates for the couple (SK₁, SK₁₇).

(SK₁, SK₁₇) and (SK₂, SK₁₈) Recovery

- ▶ For a random fault on $x_t[0]$, at cycle $t + 4$:

$$S(c_{t+1} + SK_1) + S(c_{t+1} + SK_{17}) = f_0(\delta_0, SK_0, SK_7, c_i, c'_i)$$

→ 16 possible candidates for (SK₁, SK₁₇), because of the Sbox bijectivity

- ▶ Random fault on $x_t[1]$ or on $x_t[6]$: new set of 16 possible couples for (SK₁, SK₁₇).

→ By intersecting the sets, two candidates for the couple (SK₁, SK₁₇).

- ▶ Same principle to recover (SK₂, SK₁₈)

(SK5, SK21), (SK4, SK16, SK20), (SK3, SK12, SK15, SK19) Recovery

- ▶ Random fault on $x_t[5]$:

$$\begin{cases} \Delta c_{t+3} = \delta_5 \cdot [S(c_t + \text{SK}_5) + S(c_t + \text{SK}_{21})] \\ \Delta c_{t+4} = \delta_5 \cdot [S(c_{t+1} + \text{SK}_0) \cdot S(c_t + \text{SK}_5) + S(c_{t+1} + \text{SK}_6) \cdot S(c_t + \text{SK}_{21})] \end{cases}$$

(SK5, SK21), (SK4, SK16, SK20), (SK3, SK12, SK15, SK19) Recovery

- ▶ Random fault on $x_t[5]$:

$$\begin{cases} \Delta c_{t+3} = \delta_5 \cdot [S(c_t + \text{SK}_5) + S(c_t + \text{SK}_{21})] \\ \Delta c_{t+4} = \delta_5 \cdot [S(c_{t+1} + \text{SK}_0) \cdot S(c_t + \text{SK}_5) + S(c_{t+1} + \text{SK}_6) \cdot S(c_t + \text{SK}_{21})] \end{cases}$$

→ 15 possibilities for δ_5

(SK5, SK21), (SK4, SK16, SK20), (SK3, SK12, SK15, SK19) Recovery

- ▶ Random fault on $x_t[5]$:

$$\begin{cases} \Delta c_{t+3} = \delta_5 \cdot [S(c_t + SK_5) + S(c_t + SK_{21})] \\ \Delta c_{t+4} = \delta_5 \cdot [S(c_{t+1} + SK_0) \cdot S(c_t + SK_5) + S(c_{t+1} + SK_6) \cdot S(c_t + SK_{21})] \end{cases}$$

→ 15 possibilities for δ_5

- ▶ For each δ_5 , 1 solution for (SK5, SK21)
- ▶ → 15 possible candidates for $(\delta_5, SK_5, SK_{21})$

(SK5, SK21), (SK4, SK16, SK20), (SK3, SK12, SK15, SK19) Recovery

- ▶ Random fault on $x_t[5]$:

$$\begin{cases} \Delta c_{t+3} = \delta_5 \cdot [S(c_t + \text{SK}_5) + S(c_t + \text{SK}_{21})] \\ \Delta c_{t+4} = \delta_5 \cdot [S(c_{t+1} + \text{SK}_0) \cdot S(c_t + \text{SK}_5) + S(c_{t+1} + \text{SK}_6) \cdot S(c_t + \text{SK}_{21})] \end{cases}$$

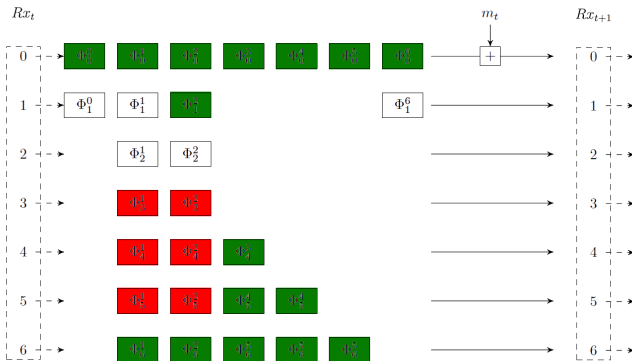
→ 15 possibilities for δ_5

- ▶ For each δ_5 , 1 solution for (SK₅, SK₂₁)
 - ▶ → 15 possible candidates for (δ_5 , SK₅, SK₂₁)
- ▶ Same principle for (SK₄, SK₁₆, SK₂₀) and (SK₃, SK₁₂, SK₁₅, SK₁₉) recovery, except that we got more hypothesis to test since we face more unknown values

Summary Table

Subkeys	# Required Faults
SK ₀	1 on x[0]
SK ₇	1 on x[2]
SK ₆	1 on x[6]
(SK ₁ , SK ₁₇)	1 on x[1]
(SK ₂ , SK ₁₈)	2 on x[2]
(SK ₅ , SK ₂₁)	3 on x[5]
(SK ₄ , SK ₂₀)	3 on x[4]
(SK ₃ , SK ₁₂ , SK ₁₅ , SK ₁₉)	5 on x[3]
Total	17

Summary of the Obtained Subkeys



- ▶ Remaining red subkeys can be obtained thanks to a **stronger attack model** (Simultaneous 4-bit stuck-at 0 value faults + more precise location)
- ▶ Around **20 faults** to recover all the subkeys

Summary of the Results

- ▶ (Fast) synchronization is a very nice property in crypto but it can come with a **high cost**
 - ▶ 2.5 bigger than Trivium
- ▶ Limited overhead to combine both encryption and decryption
 - ▶ Variants are also possible to have a near 100% reuse
- ▶ **Intrinsic protection against side-channel attacks** when initial state is unknown by the attacker
 - ▶ (see Brandon Dravie's presentation)
- ▶ DFAs in dimension 7 are possible, seems very much more difficult in dimension 40
 - ▶ (due to very fast fault diffusion)

Future Works

Hardware

- ▶ Side-channel protection with Threshold Implementation (**costly**) and (1st-2nd order) analysis on FPGA/SAKURA-G
- ▶ **ASIC** (CMOS 90nm) implementations for a deeper comparison

Software

- ▶ Side-channel protection and (1st-2nd order) analysis on ATMega smart cards
- ▶ Give **throughput benchmarks** on different platforms

Fault Attacks

- ▶ Extend the analysis done in dimension **7** to dimension **40** (**not easy!**)

© 2017 Airbus. Confidential. All rights reserved. The reproduction, distribution and use of this document is subject to the terms and conditions of the Airbus Confidentiality Policy. Airbus will be held liable for the payment of damages. All rights reserved in the event of a patent, utility model or design.

