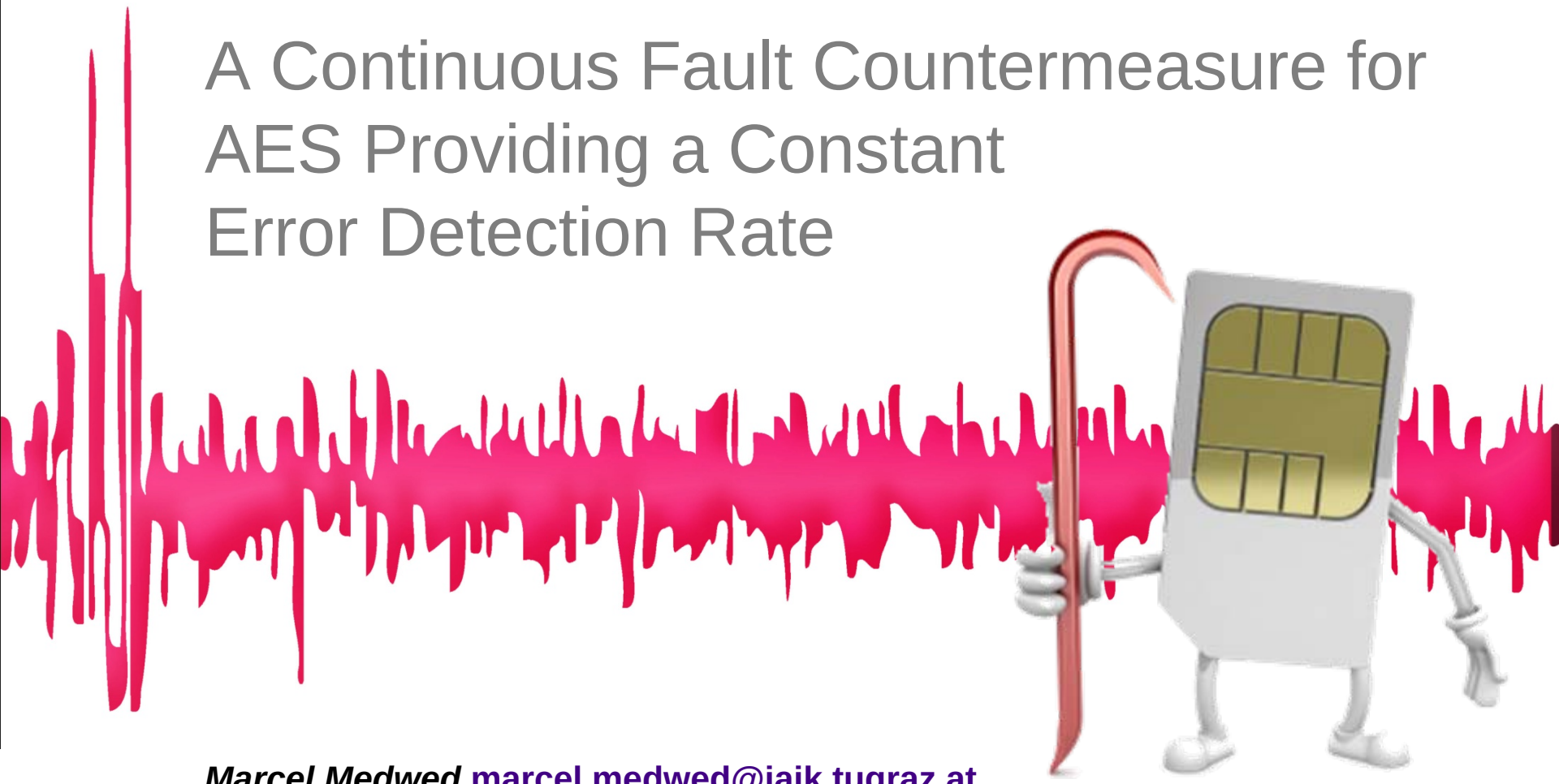
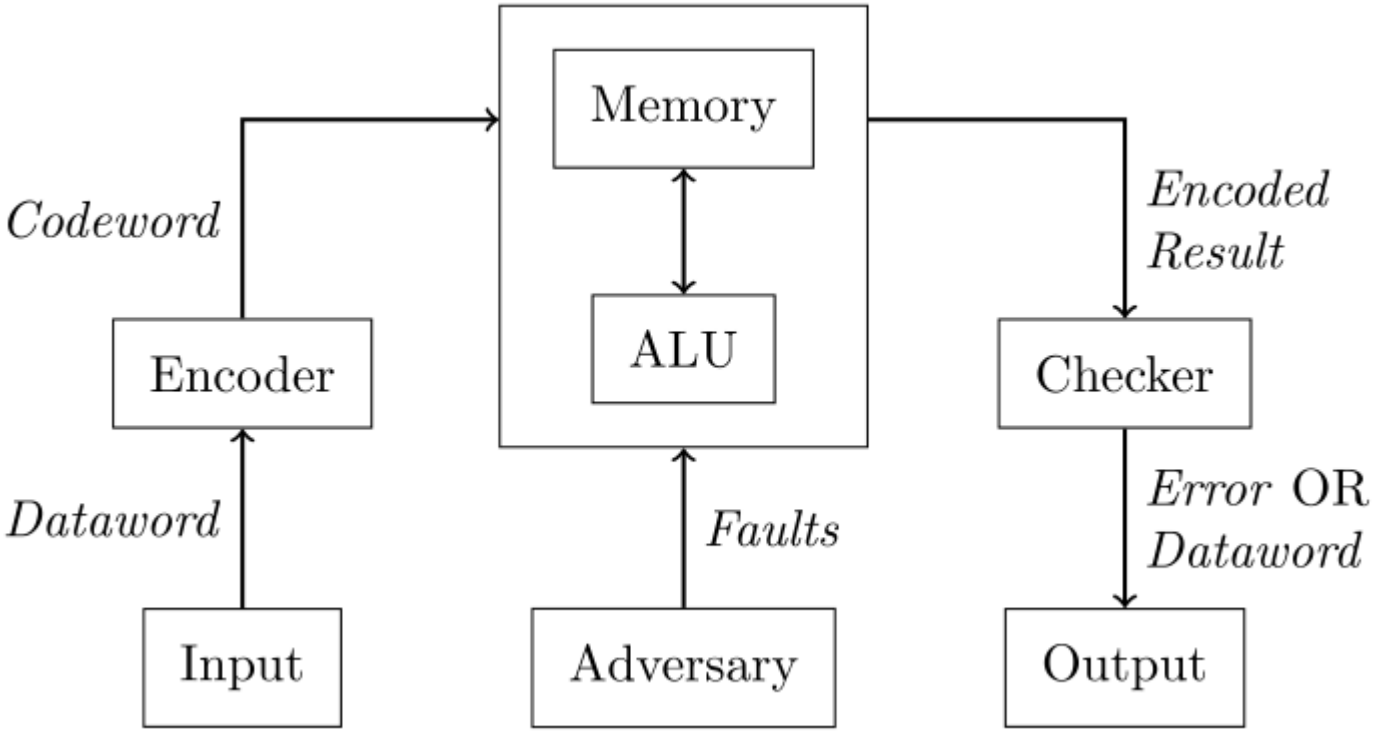
A Continuous Fault Countermeasure for AES Providing a Constant Error Detection Rate



Marcel Medwed **marcel.medwed@iaik.tugraz.at**

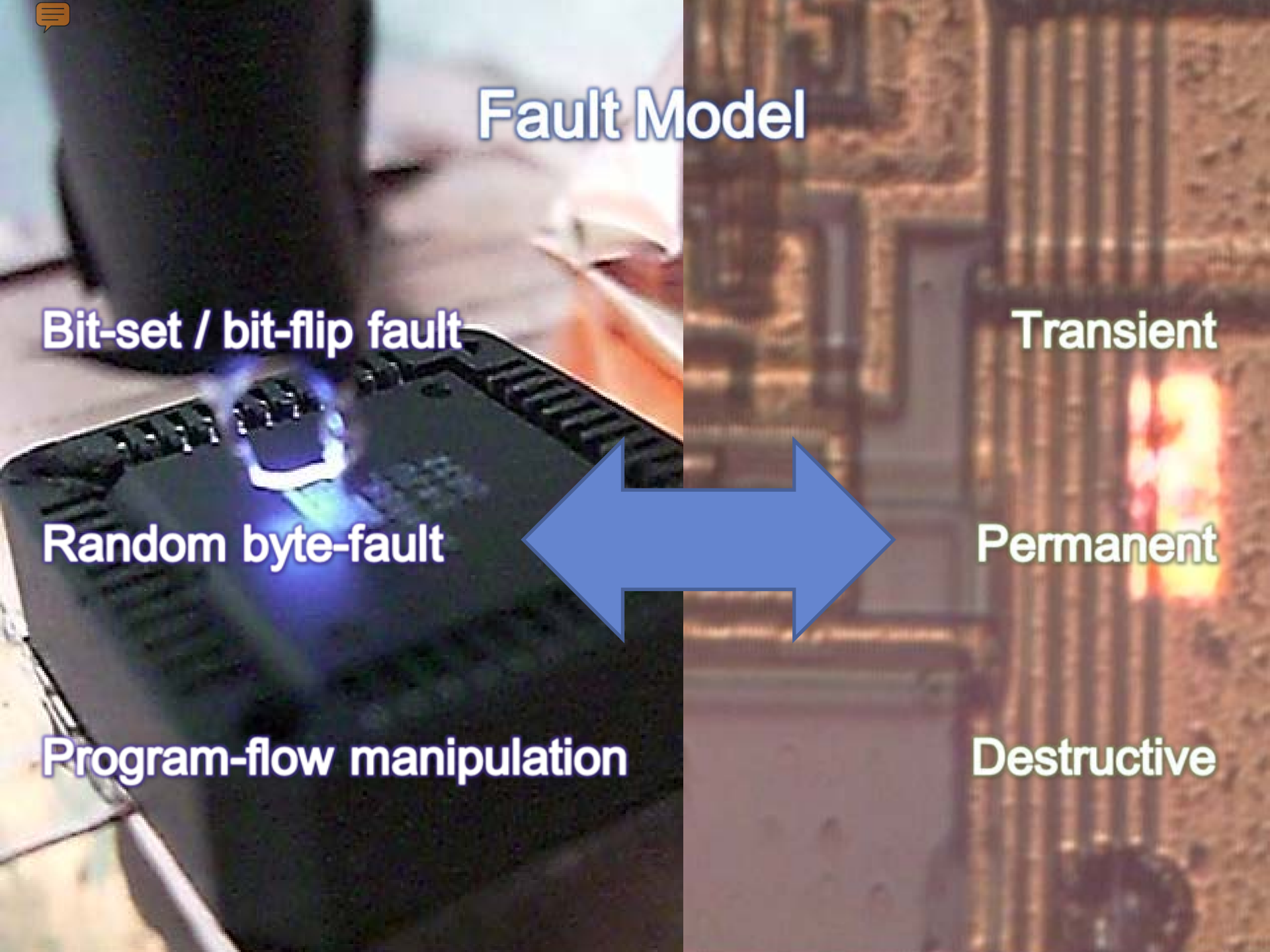
Jörn-Marc Schmidt **joern-marc.schmidt@iaik.tugraz.at**

Motivation


Memory
ALU
Encoder
Checker
Input
Adversary
Output
Codeword
Dataword
Faults
Encoded
Result
Error OR
Dataword

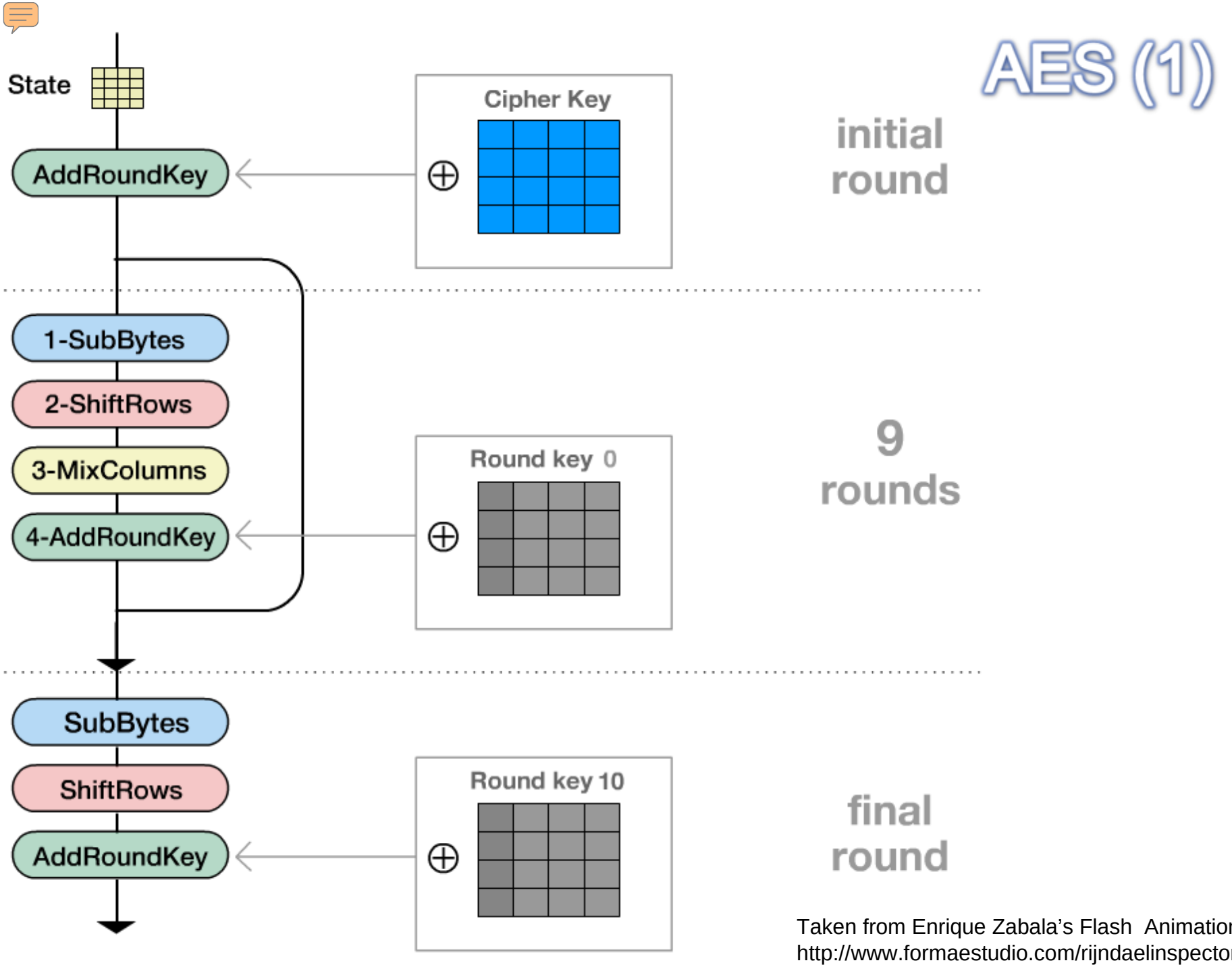
Outline

- Fault Model
- AES & AN+B codes
- Problems & Construction
- Results
- Conclusions


Fault Model
Bit-set / bit-flip fault
Random byte-fault
Program-flow manipulation
Transient
Permanent
Destructive

- Fault Model
- AES & AN+B codes
- Problems & Construction
- Results
- Conclusions

AES (1)



Taken from Enrique Zabala's Flash Animation
http://www.formaestudio.com/rijndaelinspector/

AES – Round Functions

S_i			

- AddRoundKey

  $S_i = S_i + K_{i,k}$

- SubBytes

  $S_i = A * (S_i^{254} \pmod{x^8+x^4+x^3+x+1}) + d$

- ShiftRows

  $R_m = R_m * y^m \pmod{y^4+1}$ for $m=0..3$

- MixColumns

  $C_l = C_l * (3y^3+y^2+y+2) \pmod{y^4+1}$ for $l=0..3$

Standard Embeddings

- Data algebra $\mathbf{F}_D$ mod n (e.g. RSA ring)
- Check algebra $\mathbf{F}_C$ mod r (32+ bits)
- Operate on ($\mathbf{F}_D$ x $\mathbf{F}_C$)

- x ← CRT(x_D, x_C) mod n*r
- y ← f(x) mod n*r
- y' ← f(x_C) mod r
- Check y' = y mod r

Embedding AES

- Pre-computations only needed once
- Detect data manipulations
- Detect a change of the algorithm
- Detect interchange of variables

- Fault Model
- AES & AN+B codes
- Problems & Construction
- Results
- Conclusions

Problems

- Polynomials of degree smaller than 16
- Multiplications, log tables?
- Non-linear functions, S-box table?

Construction of a Suitable Code

$$\left.\begin{array}{l} x_D : GF(2^8) \rightarrow GF(2^8)[y] \,/\, y+a_1 \\ x_C : GF(2^8) \rightarrow GF(2^8)[y] \,/\, y+a_2 \end{array}\right\} GF(2^8)[y] \,/\, y^2+c_1y+c_0 : x$$

$$x = (x_D i_{11} + x_C i_{21})*y + (x_D i_{12} + x_C i_{22})$$

- Multiplications → $GF(2^8)$ log tables
- SubBytes: Both coefficients contain info about x_D

Redundant S-box Lookup

- Normalize check bytes
- Lookup result + correction term
- Re-apply check bytes

$$t = (x_C i_{21})*y + (x_C i_{22})$$

$$d = (x_C i_{21})*y + (x_C i_{22}) + (x_{in} i_{21})*y + (x_{in} i_{22})$$

$$x = \frac{(SB(x_D)i_{11} + (x_C+x_{in})i_{21})*y + (SB(x_D)i_{12} + (x_C+x_{in})i_{22})}{(SB(x_D)i_{12} + (x_C+x_{in}+x_{out})i_{22})}$$

$E(x_D)$

$$x = (SB(x_D)i_{11} + x_{out}i_{21})*y + (SB(x_D)i_{12} + x_{out}i_{22}) + E(x_D)$$

Implementation

- Fix 32 input check bytes
- Perform dummy encryption
- Store check bytes of result

- Combine every new plaintext with check bytes
- Port check bytes if key changes

- Fault Model
- AES & AN+B codes
- Problems & Construction
- Results
- Conclusions

Security

- Bit fault: Codewords have a D_H of 4
- Byte fault: 2 bytes must be changed
- Program flow: Every operation alters x_C

Order	1st	2nd
Bit-fault	100%	100%
Byte-fault	100%	99.6%
Skip instruction	100%	99.6%*

Performance

- ATMega128 C implementation
- Multiplications in Assembly

Operation	# Cycles
AddRoundKey	305
SubBytes	4 235
ShiftRows+MixColumns	5 717
Encryption	98 322
Plaintext transformation	9 852
Ciphertext inverse transformation	7 933
Redundant key schedule (precomp.)	120 657
Redundant S-box generation (precomp.)	345 648

Comparison

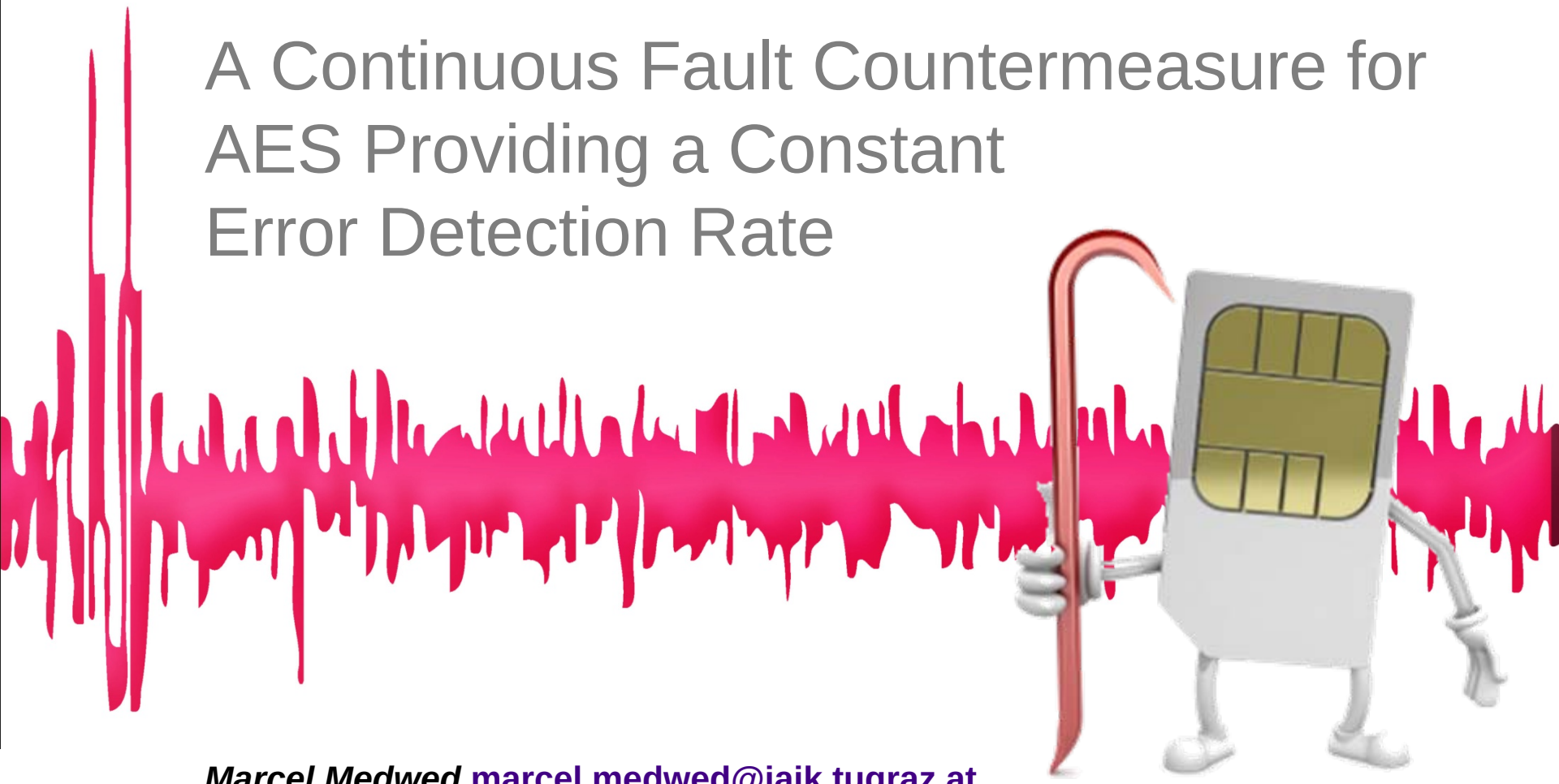
- Genelle et. al: Usage of digest values
- 10^6 cycles with on the fly key schedule, but less RAM
- 1st order: 100%, 2nd order: 2^8 vs. ~2^{12}
- No program flow protection
- Both schemes are extendable towards higher orders

- Fault Model
- AES & AN+B codes
- Problems & Construction
- Results
- Conclusions

Conclusions

- Countermeasure based on AN+B codes
- Redundant table lookups for SubBytes
- Provides data and program flow integrity
- Assures a constant error detection rate against a strong adversary

A Continuous Fault Countermeasure for AES Providing a Constant Error Detection Rate



Marcel Medwed marcel.medwed@iaik.tugraz.at

Jörn-Marc Schmidt joern-marc.schmidt@iaik.tugraz.at