A Generic Fault Countermeasure Providing Data and Program Flow Security

Marcel Medwed, Jörn-Marc Schmidt

IAIK – Graz University of Technology
marcel.medwed@iaik.tugraz.at

joern-marc.schmidt@iaik.tugraz.at

www.iaik.tugraz.at

Outline

Motivation – Attacks vs. Countermeasures

Arithmetic codes

Security

Case Study: RSA

Conclusions

Motivation 1/2

CRT-RSA – random fault (Boneh, Demillo and Lipton)

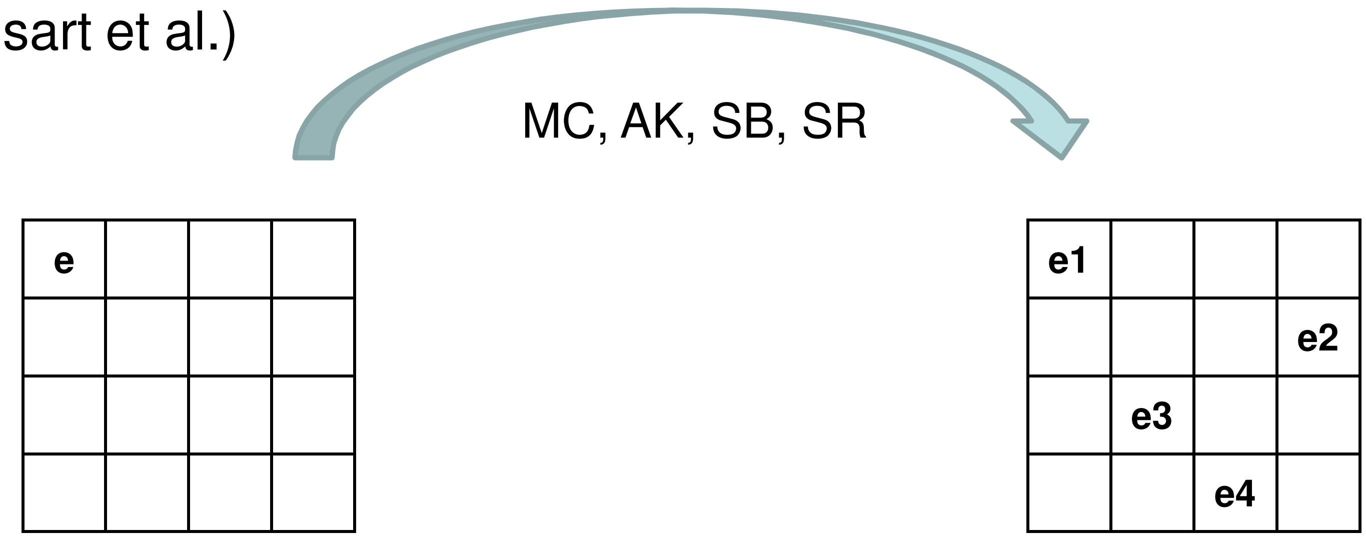
$$\left.\begin{aligned} C &= ac_1 + bc_2 \\ \hat{C} &= ac_1 + b\hat{c}_2 \end{aligned}\right\} b(c_2 - \hat{c}_2) \rightarrow p = \gcd(C - \hat{C}, N)$$

RSA – skip squaring (Schmidt, Herbst)

Motivation 2/2

Safe-Error Attacks (Bao et al.)

Random word errors (e.g. AES 9th round before MC) (Dusart et al.)



Previous Work on Countermeasures

General

- Time redundancy
- Space redundancy
- Memory encryption / protection / scrambling

Asymmetric

- Compound Algebras
- Error Diffusion

Symmetric

- Small error detection rate
- Only parts – restricted parities

Goals

Generic redundant representation

Program flow security and encode addresses

No assumptions about adversary

Detection and diffusion possibility

Adding Redundancy

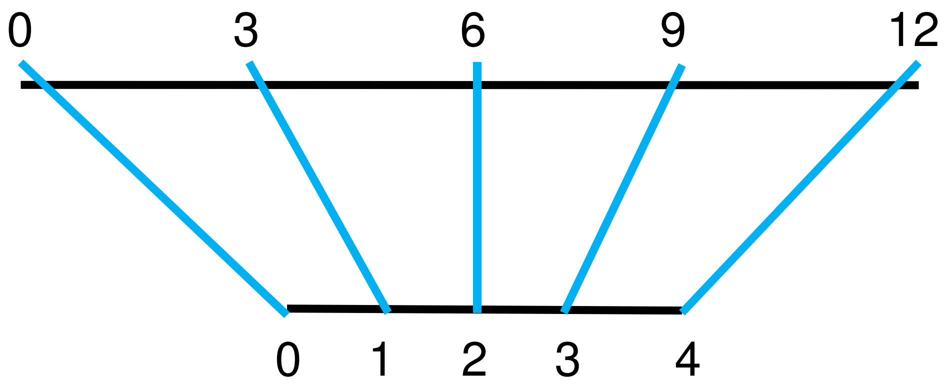
Coding Theory

- Error detection / correction
- Linear / Non-linear

G*H = 0	…	gen. and check matrices
y = x*G	…	encoding
S = y*H	…	syndrome calculation

Arithmetic Codes – AN – Code

(Proulder, 1989)



Z/mZ → aZ/amZ

→ ipZ/amZ

ax+ ay = a(x+y)

ax * ay = a^2*x*y

a = a^2 → idempotent

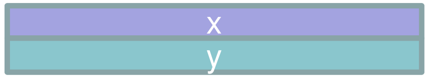
Excursus: CRT

$X := Z / mZ$

$Y := Z / aZ$

$X \times Y \cong CRT(X, Y)$

$x * a * (a^{-1} \bmod m) + y * m * (m^{-1} \bmod a) \bmod am$



Idempotent elements

$$\begin{pmatrix} X \\ Y \end{pmatrix} = X \times Y \cong U$$

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} * \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 * x_2 \\ y_1 * y_2 \end{pmatrix}$$

$$u_1 * u_2 = CRT(x_1 * x_2, y_1 * y_2)$$

Introducing an Error

$$x * a * (a^{-1} \bmod m) + y * m * (m^{-1} \bmod a) + \varepsilon$$

$$\Rightarrow (x + \varepsilon_x) * a * (a^{-1} \bmod m) + (y + \varepsilon_y) * m * (m^{-1} \bmod a)$$

$$\begin{pmatrix} x + \varepsilon_x \\ y + \varepsilon_y \end{pmatrix} \begin{pmatrix} \bmod m \\ \bmod a \end{pmatrix}$$

This holds for instruction skipping as well

AN – Codes – Error Detection

✓ $(ip * x + \varepsilon) \rightarrow 1 - 1/a$

✗ $ip * (x + \varepsilon)$

$$\begin{pmatrix} x \\ 0 \end{pmatrix}$$

✓ $(ip * x) + (ip * z) + \varepsilon$
$= ip * (x + z) + \varepsilon$

$$\begin{pmatrix} x \\ 0 \end{pmatrix} + \begin{pmatrix} z + \varepsilon_x \\ \varepsilon_y \end{pmatrix} = \begin{pmatrix} x + z + \varepsilon_x \\ \varepsilon_y \end{pmatrix}$$

✗ $(ip * x) * (ip * z + \varepsilon)$
$= ip * (xz + x\varepsilon)$

$$\begin{pmatrix} x \\ 0 \end{pmatrix} * \begin{pmatrix} z + \varepsilon_x \\ \varepsilon_y \end{pmatrix} = \begin{pmatrix} (z + \varepsilon_x) * x \\ 0 \end{pmatrix}$$

AN + B Codes

(Proulder, 1989)

$$\begin{pmatrix} x \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ 1 \end{pmatrix} * \begin{pmatrix} z+\varepsilon_x \\ \varepsilon_y \end{pmatrix} = \begin{pmatrix} (z+\varepsilon_x)*y \\ \varepsilon_y \end{pmatrix}$$

$$(ip_1 * x + ip_2 + \varepsilon) * (ip_1 * y + ip_2) =$$
$$ip_1 * x * y + ip_2 + \varepsilon * (ip_1 * y + ip_2)$$

Extended AN + B Codes

$$\begin{pmatrix} x \\ s \end{pmatrix}$$

$$\begin{pmatrix} x \\ s \end{pmatrix}^t = \begin{pmatrix} x^t \\ s^t \end{pmatrix}$$

$$\begin{pmatrix} a \\ s_a \end{pmatrix} + \begin{pmatrix} b \\ s_b \end{pmatrix} \neq \begin{pmatrix} a \\ s_a \end{pmatrix} + \begin{pmatrix} c \\ s_c \end{pmatrix}$$

Protects program flow as well

Instruction sequence must be known

- AES
- RSA

Security

$$(x+\varepsilon_x)*a*(a^{-1} \bmod m)+(y+\varepsilon_y)*m*(m^{-1} \bmod a)$$

$$\Rightarrow P = 1/a$$

$$\varepsilon_{y1}+\varepsilon_{y2} \equiv 0 \bmod a$$

$$\Rightarrow \varepsilon_{y1} \equiv -\varepsilon_{y2} \bmod a$$

$$\Rightarrow P = 1/a$$

$$y_2\varepsilon_{y1} + y_1\varepsilon_{y2} + \varepsilon_{y1}\varepsilon_{y2} \equiv 0 \bmod z$$

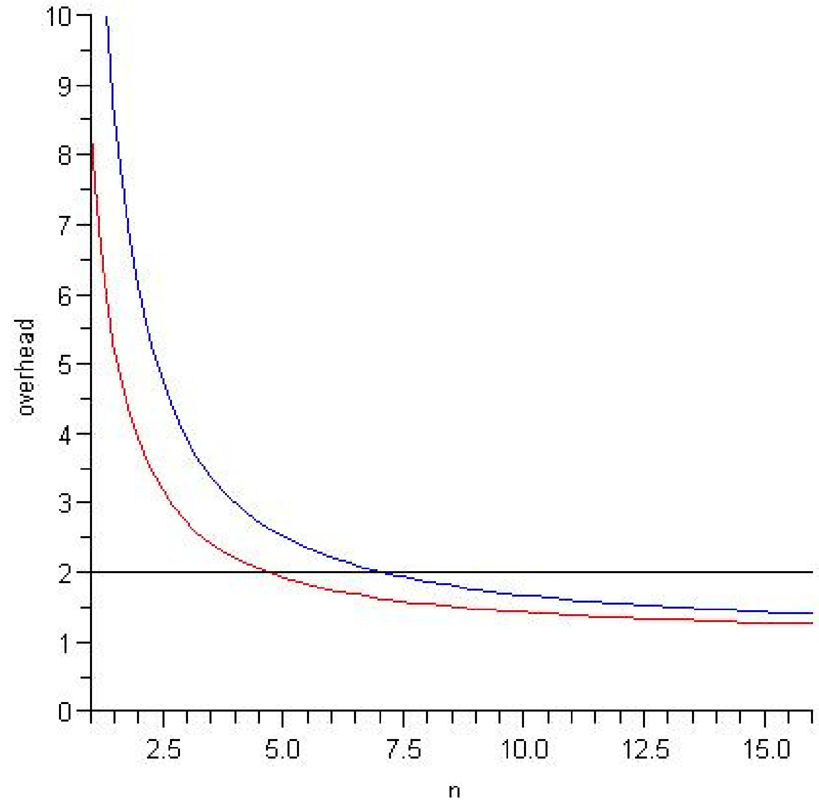
$$\Rightarrow \varepsilon_{y2} \equiv -(y_1+\varepsilon_{y1})^{-1}(y_2\varepsilon_{y1})$$

$$\Rightarrow P = \Phi(a)/a^2$$

Case Study: RSA

Error detection rates

Bit error	1
Word error*	1
Multiword error	$1-\frac{1}{2^{96}}$



* For a > 2^wordsize

Conclusions and Ongoing Work

EAN+B codes

- provide high amount of security
- are generic
- achieve reasonable performance
- can even secure the program flow
- atomic approach

Outlook: EAN+B + AES

A Generic Fault Countermeasure Providing Data and Program Flow Security

Marcel Medwed, Jörn-Marc Schmidt

IAIK – Graz University of Technology
marcel.medwed@iaik.tugraz.at

joern-marc.schmidt@iaik.tugraz.at

www.iaik.tugraz.at