Efficiency and Implementation Security of Code-based Cryptosystems

PhD Thesis by Falko Strenzke

Falko Strenzke

Cryptography and Computeralgebra, Department of Computer Science,
Technische Universität Darmstadt, Germany,
fstrenzke@cryptosource.de

November 11, 2013

Public Key Encryption

Alice

Bob

secret key (s)

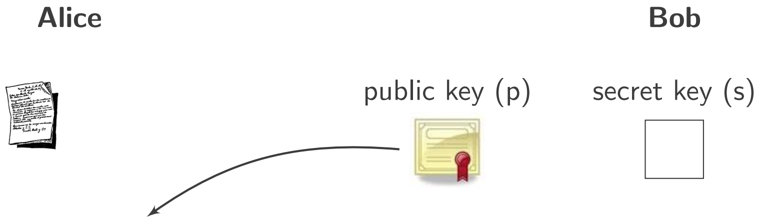
Public Key Encryption

Alice **Bob**

public key (p)

secret key (s)

Public Key Encryption



Public Key Encryption

Alice

Bob

public key (p)

secret key (s)

$$c = \mathcal{E}_{\mathrm{p}}(m)$$

Public Key Encryption

Alice

Bob

public key (p)

secret key (s)

$$c = \mathcal{E}_{\mathrm{p}}(m)$$

Public Key Encryption

Alice

Bob

public key (p)

secret key (s)

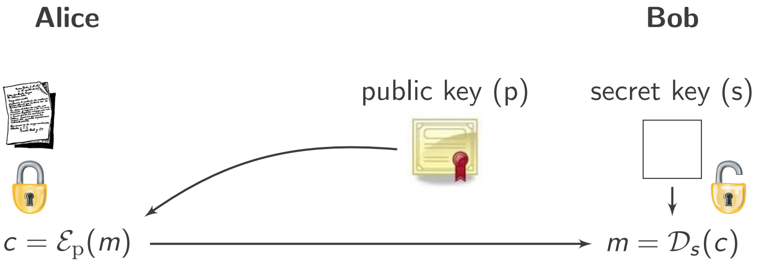
$$c = \mathcal{E}_{\mathrm{p}}(m)$$

Alice **Bob**

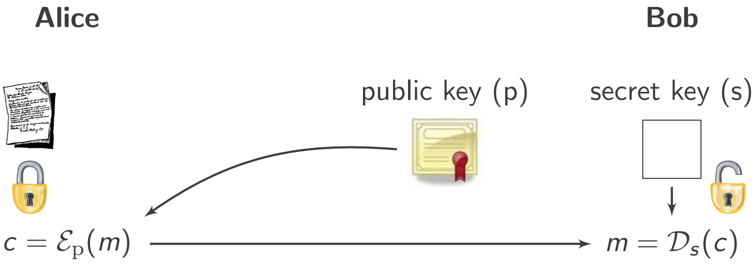

public key (p) secret key (s)

$c = \mathcal{E}_{\mathrm{p}}(m) \longrightarrow m = \mathcal{D}_s(c)$

Public Key Encryption



Public Key Encryption



RSA, ElGamal, etc.

Public Key Encryption

Alice **Bob**

public key (p)

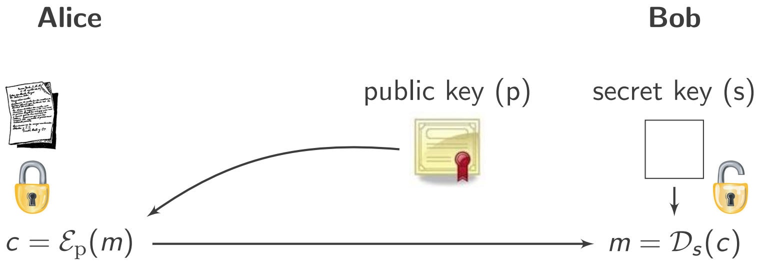
secret key (s)

$c = \mathcal{E}_{\mathrm{p}}(m)$ $\longrightarrow$ $m = \mathcal{D}_s(c)$

$\longrightarrow$ $\boxed{\text{RSA, ElGamal, etc.}}$

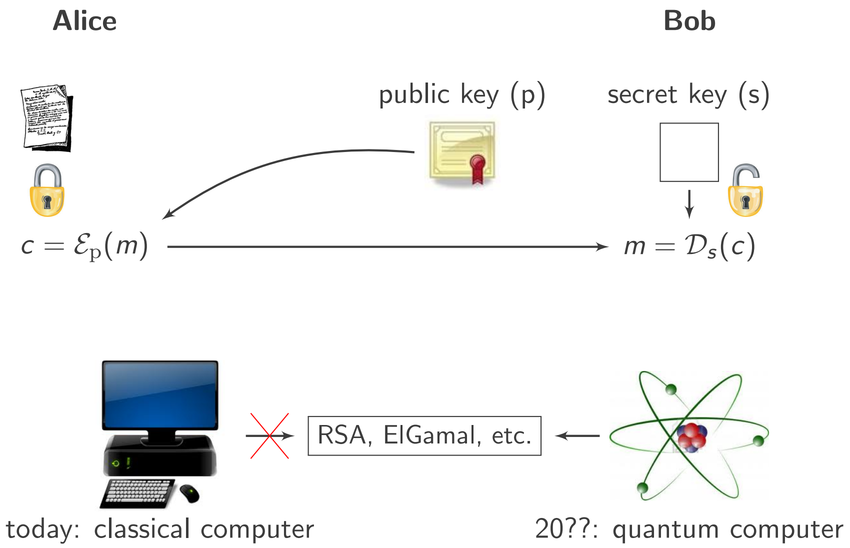
today: classical computer

Public Key Encryption

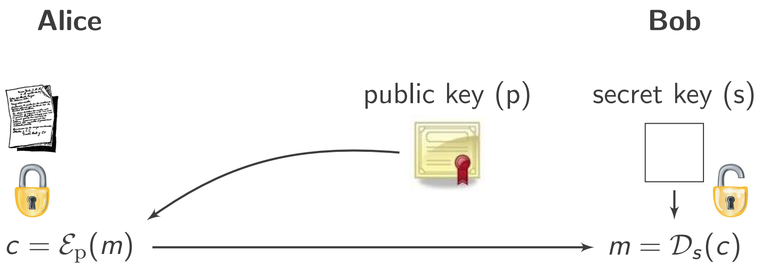


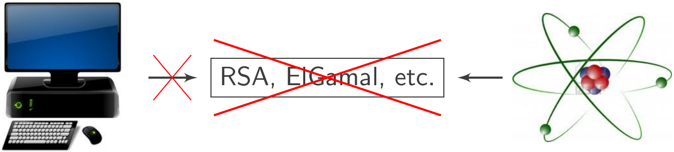
today: classical computer

Public Key Encryption



Public Key Encryption

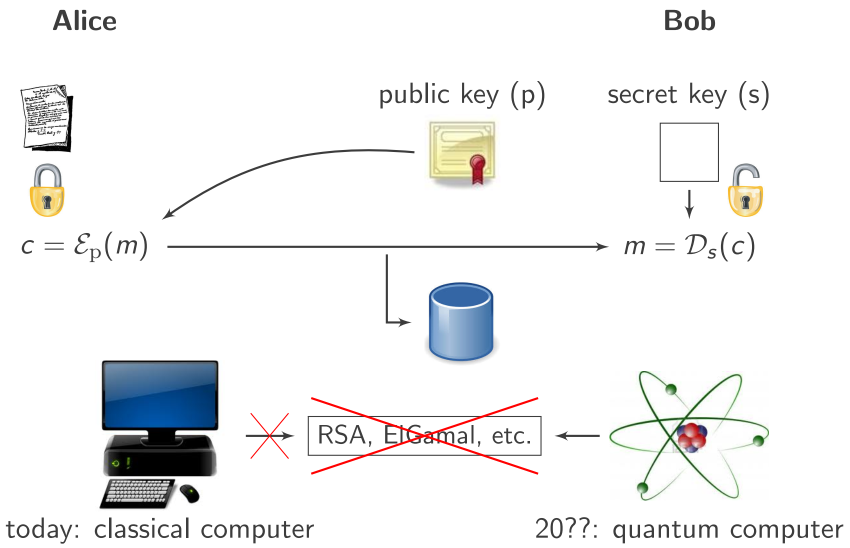




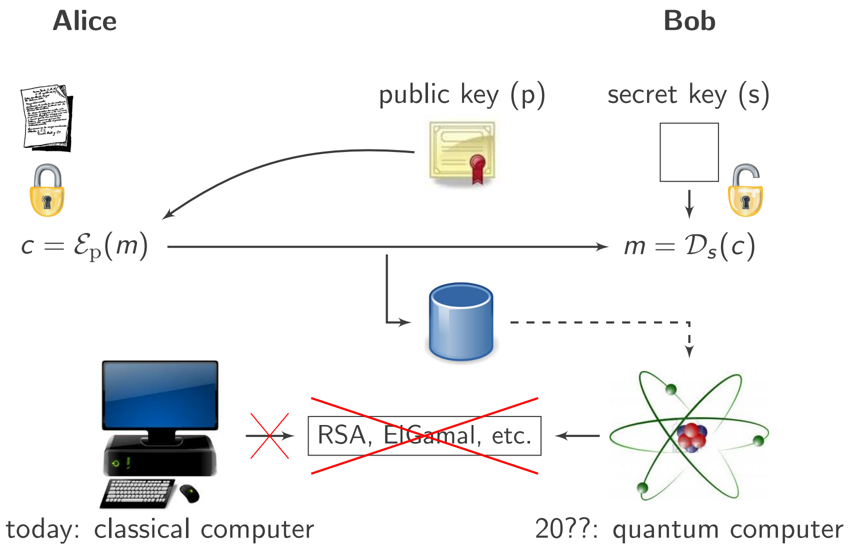
today: classical computer

20??: quantum computer

Public Key Encryption



Public Key Encryption



Code-based Cryptosystems

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

Code-based Cryptosystems

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

Code-based Cryptosystems

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

Code-based Cryptosystems

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

Code-based Cryptosystems

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

Code-based Cryptosystems

- need for cryptosystems in a post-quantum world
- lattice-based, multivariate, ...
- code-based cryptosystems
  - McEliece scheme proposed in 1976
  - still regarded secure
  - fast encryption and decryption
  - large public key
  - Niederreiter scheme very similar

Outline

- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions

Outline

- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions

Outline

- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions

Outline

- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions

Outline

- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions

Outline

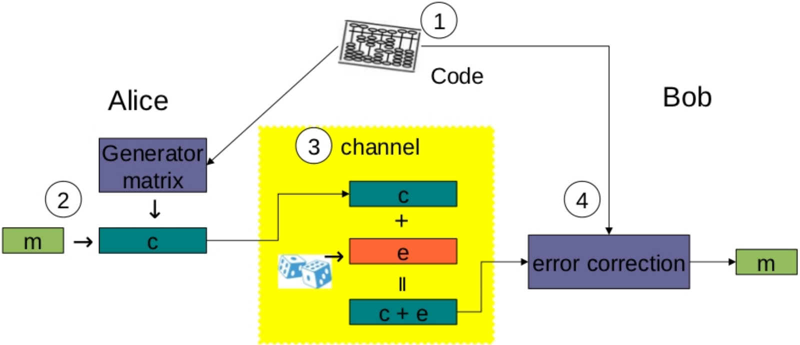
- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions

Outline

- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions

Outline

- Preliminaries
  - Error Correcting Codes
  - Goppa Codes
  - McEliece scheme
    - Encryption
    - Decryption (syndrome decoding)
  - Challenges of code-based cryptosystems
- Contributions


1
Code
Alice
Bob
3
channel
Generator matrix
2
m
c
c
+
e
=
c + e
4
error correction
m

Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in C$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in C$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^T = 0$ if $c \in C$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in \mathcal{C}$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in \mathcal{C}$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in \mathcal{C}$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in \mathcal{C}$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

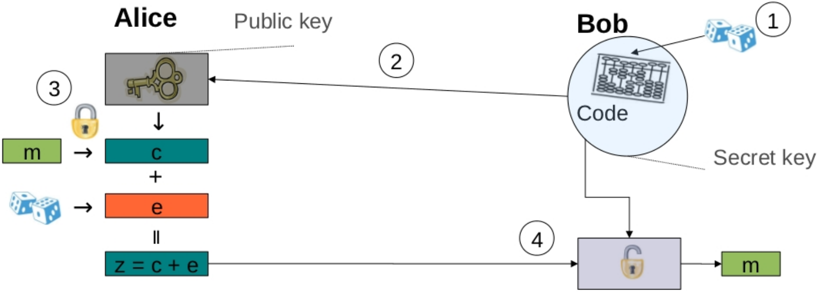
Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in \mathcal{C}$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

Goppa Codes

- Parameters of a Goppa Code
  - irreducible polynomial $g(Y) \in \mathbb{F}_{2^m}[Y]$ of degree t (the Goppa Polynomial)
  - support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, where α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$
- Properties of the Code
  - the code has length $n \leq 2^m$ (code word length) ,
  - dimension $k = n - mt$ (message length) and
  - can correct up to t errors.
  - a parity check matrix H, where $cH^\top = 0$ if $c \in \mathcal{C}$
  - example for secure parameters: $n = 2048$, $t = 50$ for 100 bit security

The McEliece PKC



- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} (Y^{t-1}, \cdots, Y, 1)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \dots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

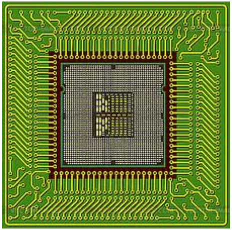
- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

Syndrome Decoding: Patterson Algorithm

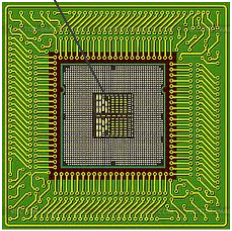
- secret key: $g(Y)$, $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
- input: distorted codeword $\vec{e} \oplus \vec{c}$
- output: error vector $\vec{e} \in \mathbb{F}_{2^m}^n$

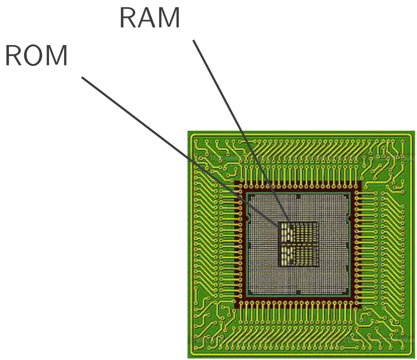
- $S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
- $U(Y) \leftarrow S^{-1} \bmod g(Y)$ // by EEA
- $\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
- $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$ $//\beta(Y)\tau(Y) \equiv \alpha(Y) \bmod g(Y)$
- $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$ // $\sigma(Y) = \prod_{i=0}^{t-1}(\alpha_{f_i} - Y)$
- $e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$ // root finding

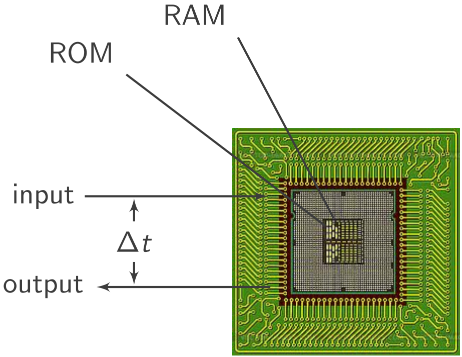
Implementation Aspects of Cryptograpic Algorithms



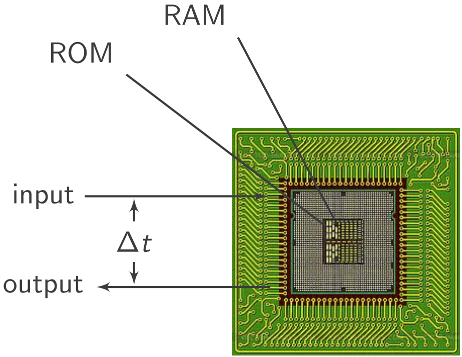
RAM




RAM
ROM

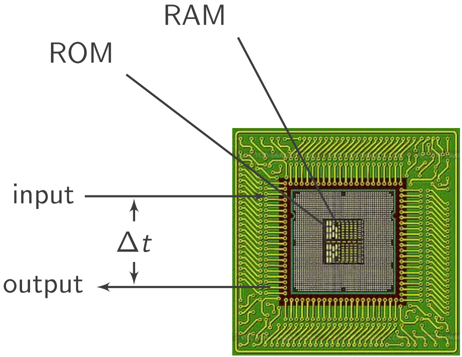

RAM
ROM
input
Δt
output

Efficiency

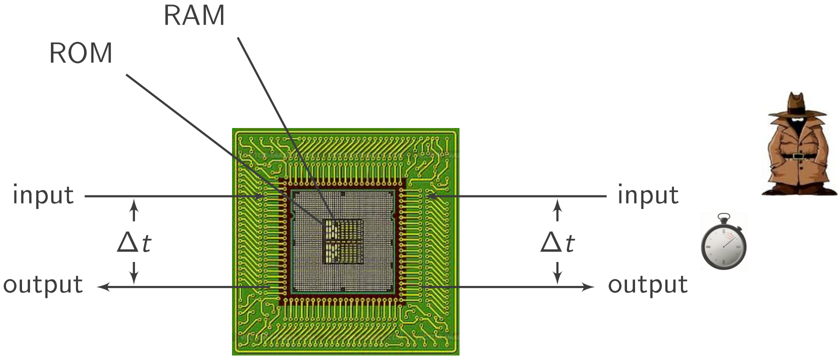


Implementation Aspects of Cryptograpic Algorithms

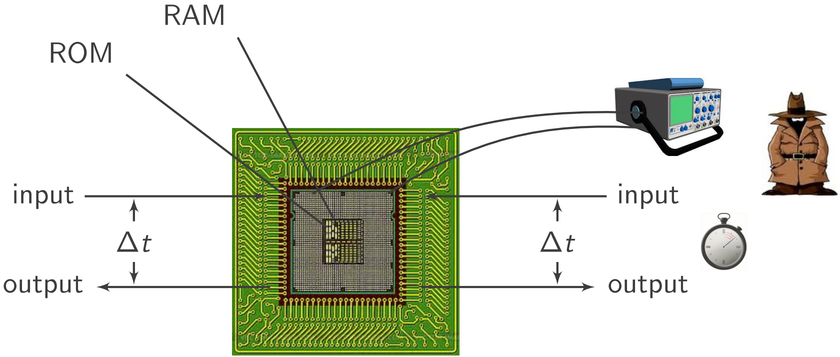
Efficiency

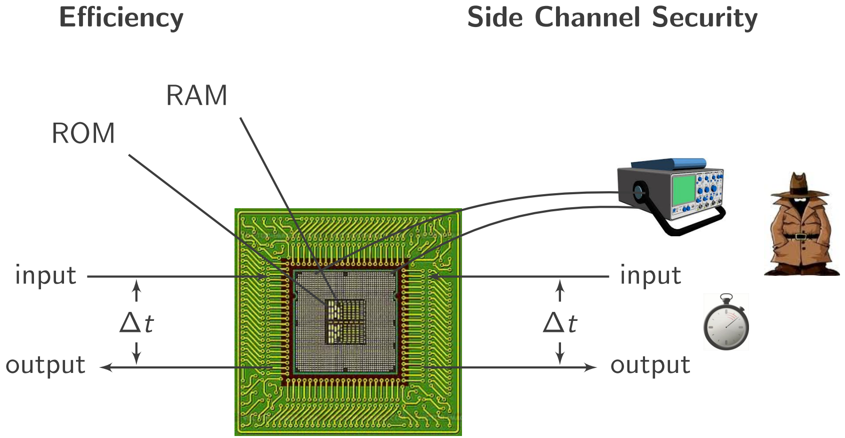


Efficiency



Efficiency





Efficiency
Side Channel Security
RAM
ROM
input
Δt
output
input
Δt
output

The Challenges of Code-based Encryption

- Code-based schemes known to be fast
  - fast enough on embedded systems (smart cards)?
  - time memory trade-offs?
- Large public-key size
  - what does this mean for embedded systems?
- Side Channel Security
  - no previous works

The Challenges of Code-based Encryption

- Code-based schemes known to be fast
  - fast enough on embedded systems (smart cards)?
  - time memory trade-offs?
- Large public-key size
  - what does this mean for embedded systems?
- Side Channel Security
  - no previous works

The Challenges of Code-based Encryption

- Code-based schemes known to be fast
  - fast enough on embedded systems (smart cards)?
  - time memory trade-offs?
- Large public-key size
  - what does this mean for embedded systems?
- Side Channel Security
  - no previous works

The Challenges of Code-based Encryption

- Code-based schemes known to be fast
  - fast enough on embedded systems (smart cards)?
  - time memory trade-offs?
- Large public-key size
  - what does this mean for embedded systems?
- Side Channel Security
  - no previous works

The Challenges of Code-based Encryption

- Code-based schemes known to be fast
  - fast enough on embedded systems (smart cards)?
  - time memory trade-offs?
- Large public-key size
  - what does this mean for embedded systems?
- Side Channel Security
  - no previous works

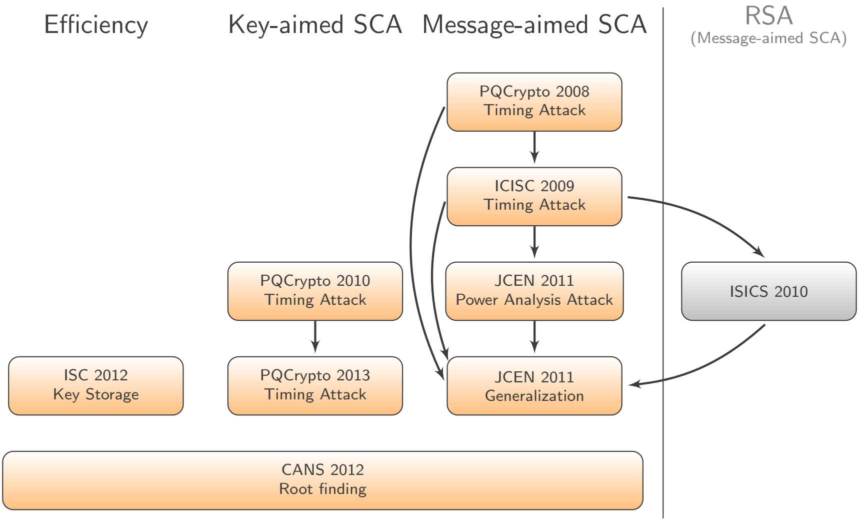
The Challenges of Code-based Encryption

- Code-based schemes known to be fast
  - fast enough on embedded systems (smart cards)?
  - time memory trade-offs?
- Large public-key size
  - what does this mean for embedded systems?
- Side Channel Security
  - no previous works

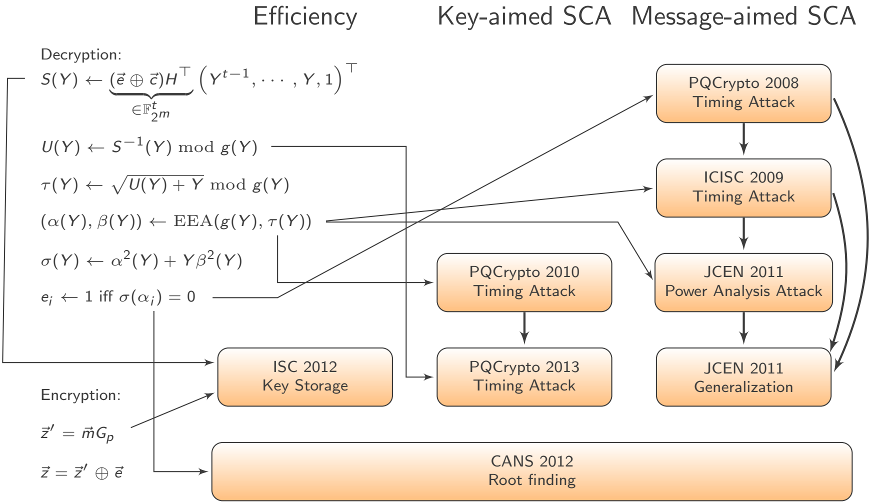
The Challenges of Code-based Encryption

- Code-based schemes known to be fast
  - fast enough on embedded systems (smart cards)?
  - time memory trade-offs?
- Large public-key size
  - what does this mean for embedded systems?
- Side Channel Security
  - no previous works

Overview

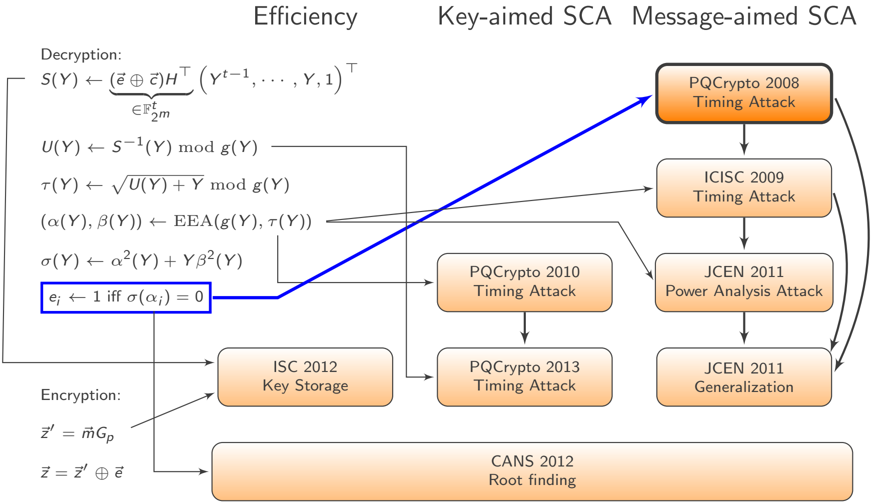

Efficiency
Key-aimed SCA
Message-aimed SCA
RSA
(Message-aimed SCA)
PQCrypto 2008
Timing Attack
ICISC 2009
Timing Attack
PQCrypto 2010
Timing Attack
JCEN 2011
Power Analysis Attack
ISICS 2010
ISC 2012
Key Storage
PQCrypto 2013
Timing Attack
JCEN 2011
Generalization
CANS 2012
Root finding

Overview



Efficiency
Key-aimed SCA
Message-aimed SCA
Decryption:
$S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^{\top}}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^{\top}$
$U(Y) \leftarrow S^{-1}(Y) \bmod g(Y)$
$\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
$(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$
$\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$
$e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$
Encryption:
$\vec{z}' = \vec{m}G_p$
$\vec{z} = \vec{z}' \oplus \vec{e}$
PQCrypto 2008
Timing Attack
ICISC 2009
Timing Attack
JCEN 2011
Power Analysis Attack
JCEN 2011
Generalization
PQCrypto 2010
Timing Attack
PQCrypto 2013
Timing Attack
ISC 2012
Key Storage
CANS 2012
Root finding

Message-aimed Timing Attack



Message-aimed Timing Attack (I)

- let $w = \mathrm{wt}(\vec{e})$
- $\deg(\sigma(Y)) = w$ for $w \leq t$
- basically any root-finding variant:
- (at least) linear dependency of root-finding time on $\deg(\sigma(Y))$

Message-aimed Timing Attack (I)

- let $w = \mathrm{wt}(\vec{e})$
- $\deg(\sigma(Y)) = w$ for $w \leq t$
- basically any root-finding variant:
- (at least) linear dependency of root-finding time on $\deg(\sigma(Y))$

Message-aimed Timing Attack (I)

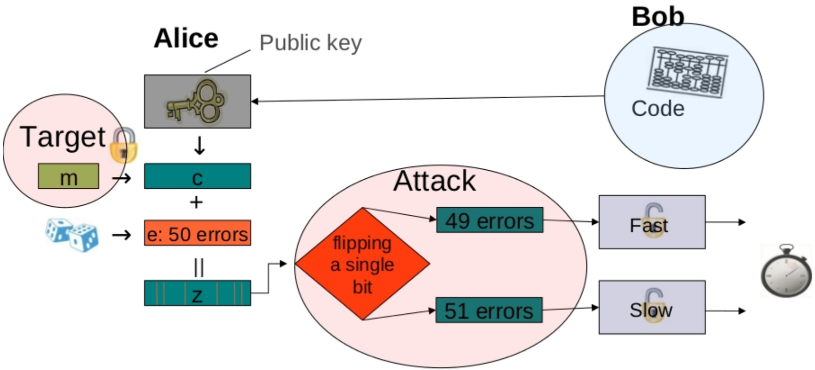
- let $w = \mathrm{wt}\left(\vec{e}\right)$
- $\deg\left(\sigma(Y)\right) = w$ for $w \leq t$
- basically any root-finding variant:
- (at least) linear dependency of root-finding time on $\deg\left(\sigma(Y)\right)$

Message-aimed Timing Attack (I)

- let $w = \mathrm{wt}(\vec{e})$
- $\deg(\sigma(Y)) = w$ for $w \leq t$
- basically any root-finding variant:
- (at least) linear dependency of root-finding time on $\deg(\sigma(Y))$

Message-aimed Timing Attack (II)

$t = 50$



Overview

Efficiency | Key-aimed SCA | Message-aimed SCA

Decryption:

$S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$

$U(Y) \leftarrow S^{-1}(Y) \bmod g(Y)$

$\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$

$(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$

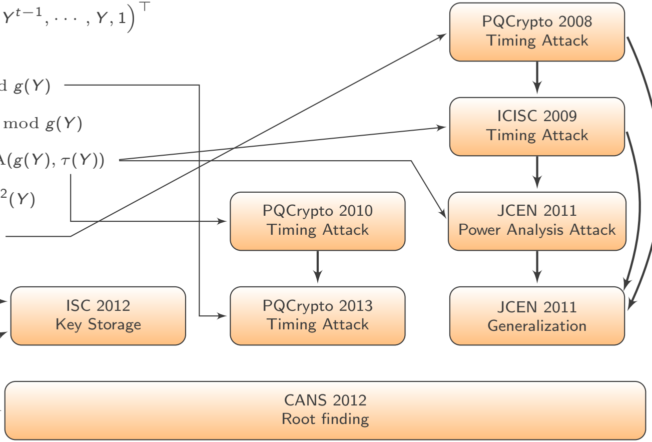
$\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$

$e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$

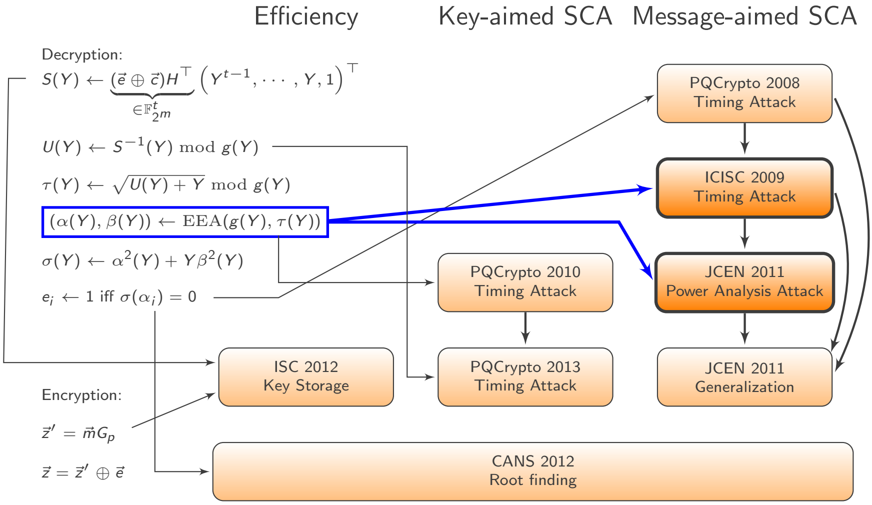
Encryption:

$\vec{z}' = \vec{m}G_p$

$\vec{z} = \vec{z}' \oplus \vec{e}$



Refinements of the Message-aimed Attack



Refinements of the Message-aimed Attack (outline)

- Number of iterations in the EEA already dependent on w
  - smaller timing differences, allowing same attack
  - countermeasure: avoid "premature" abortion of the EEA
- Related simple power analysis attack on the number of iterations in EEA
  - similar countermeasure

Refinements of the Message-aimed Attack (outline)

- Number of iterations in the EEA already dependent on w
  - smaller timing differences, allowing same attack
  - countermeasure: avoid "premature" abortion of the EEA
- Related simple power analysis attack on the number of iterations in EEA
  - similar countermeasure

Refinements of the Message-aimed Attack (outline)

- Number of iterations in the EEA already dependent on w
  - smaller timing differences, allowing same attack
  - countermeasure: avoid "premature" abortion of the EEA
- Related simple power analysis attack on the number of iterations in EEA
  - similar countermeasure

Refinements of the Message-aimed Attack (outline)

- Number of iterations in the EEA already dependent on w
  - smaller timing differences, allowing same attack
  - countermeasure: avoid "premature" abortion of the EEA
- Related simple power analysis attack on the number of iterations in EEA
  - similar countermeasure

Refinements of the Message-aimed Attack (outline)

- Number of iterations in the EEA already dependent on w
  - smaller timing differences, allowing same attack
  - countermeasure: avoid "premature" abortion of the EEA
- Related simple power analysis attack on the number of iterations in EEA
  - similar countermeasure

Overview

Efficiency | Key-aimed SCA | Message-aimed SCA

Decryption:

$$S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^{\top}}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^{\top}$$

$$U(Y) \leftarrow S^{-1}(Y) \bmod g(Y)$$

$$\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$$

$$(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$$

$$\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$$

$$e_i \leftarrow 1 \text{ iff } \sigma(\alpha_i) = 0$$

Encryption:

$$\vec{z}' = \vec{m}G_p$$

$$\vec{z} = \vec{z}' \oplus \vec{e}$$

PQCrypto 2008
Timing Attack

ICISC 2009
Timing Attack

PQCrypto 2010
Timing Attack

JCEN 2011
Power Analysis Attack

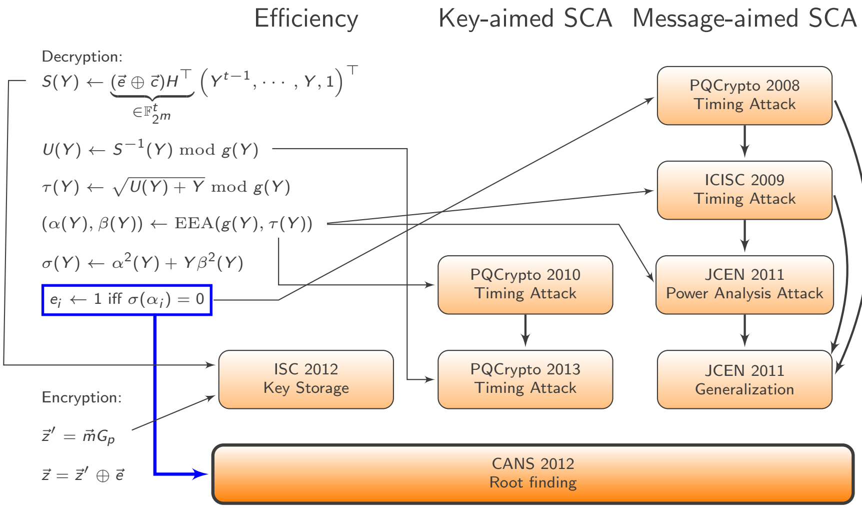
ISC 2012
Key Storage

PQCrypto 2013
Timing Attack

JCEN 2011
Generalization

CANS 2012
Root finding

Analysis of Root-Finding Variants



Analysis of Root-Finding Variants

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation				

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms			

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte		

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division				

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms			

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte		

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2				

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms			

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte		

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte	unsafe	

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte	unsafe	probably unsafe

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte	unsafe	probably unsafe
linearized polynomials				

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte	unsafe	probably unsafe
linearized polynomials	415ms			

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte	unsafe	probably unsafe
linearized polynomials	415ms	2344 byte		

Analysis of Root-Finding Variants

using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

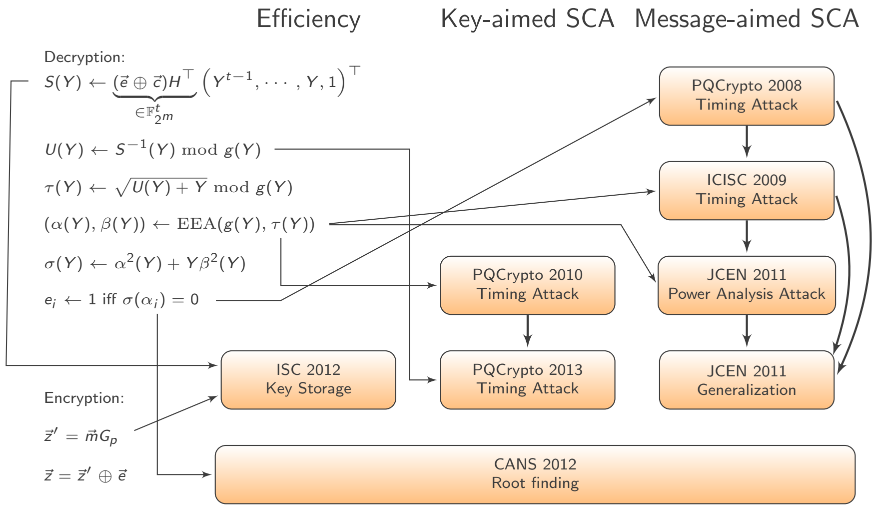
	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte	unsafe	probably unsafe
linearized polynomials	415ms	2344 byte	safe	

Analysis of Root-Finding Variants

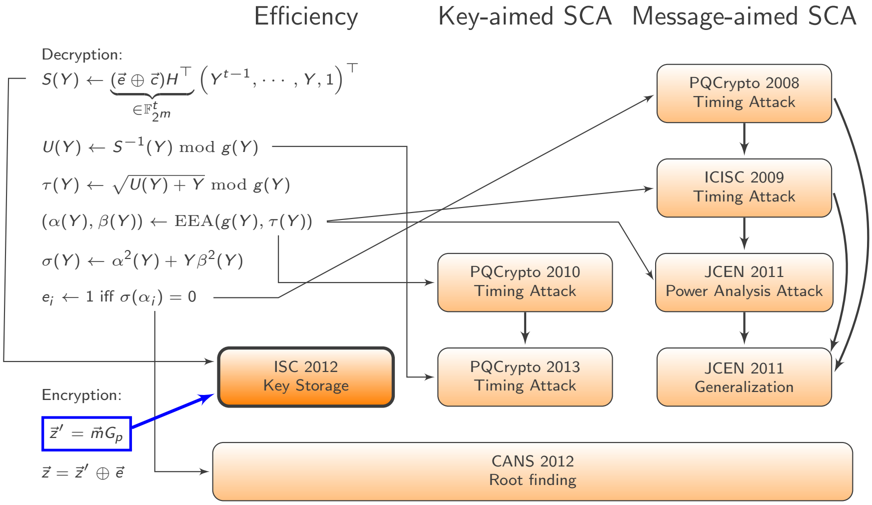
using parameters $n = 6624$, $t = 115$ (244 bit security); Atmel AP7000, 30 MHz

	Speed	RAM demands	Mess.-aim. TA	Key-aim. TA
exh. evaluation	1269ms	2344 byte	safe	safe
exh. evaluation w/ division	638ms	2344 byte	unsafe	safe with c.m.
BTZ_2	272ms	34886 byte	unsafe	probably unsafe
linearized polynomials	415ms	2344 byte	safe	safe with c.m.

Overview

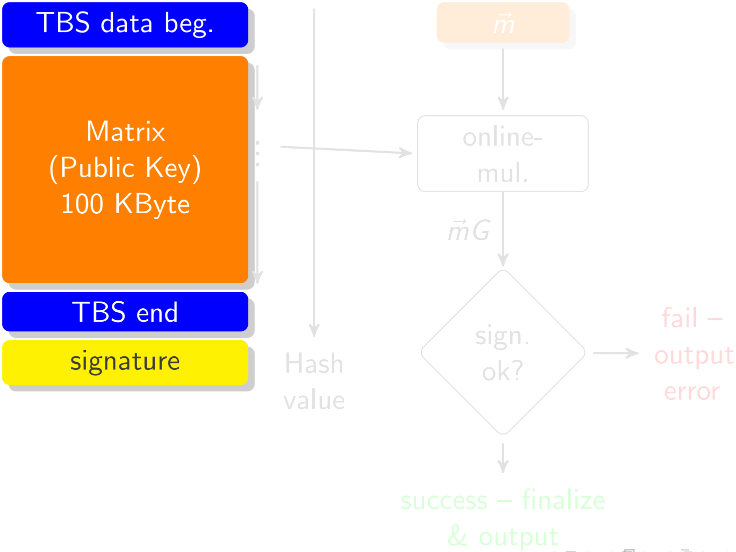

Efficiency
Key-aimed SCA
Message-aimed SCA
Decryption:
$S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
$U(Y) \leftarrow S^{-1}(Y) \bmod g(Y)$
$\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
$(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$
$\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$
$e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$
Encryption:
$\vec{z}' = \vec{m}G_p$
$\vec{z} = \vec{z}' \oplus \vec{e}$
PQCrypto 2008
Timing Attack
ICISC 2009
Timing Attack
JCEN 2011
Power Analysis Attack
JCEN 2011
Generalization
PQCrypto 2010
Timing Attack
PQCrypto 2013
Timing Attack
ISC 2012
Key Storage
CANS 2012
Root finding

Encryption in PKI



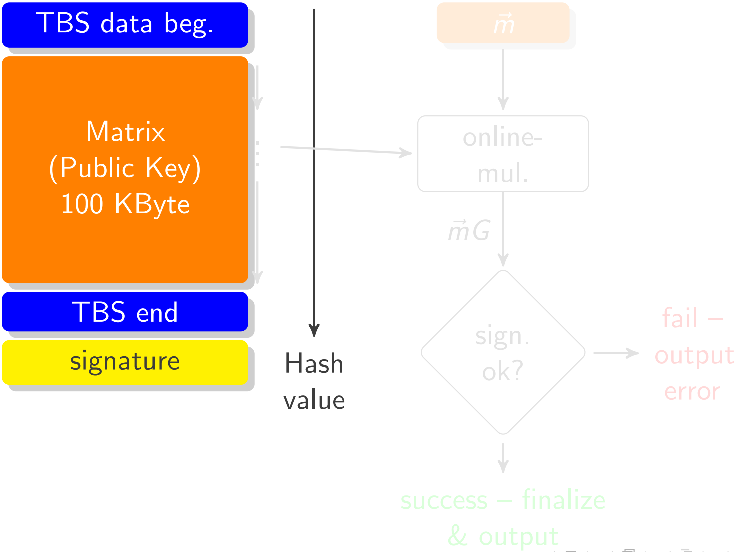
Solution for Memory-constrained Platforms

Process the certificate during receipt:



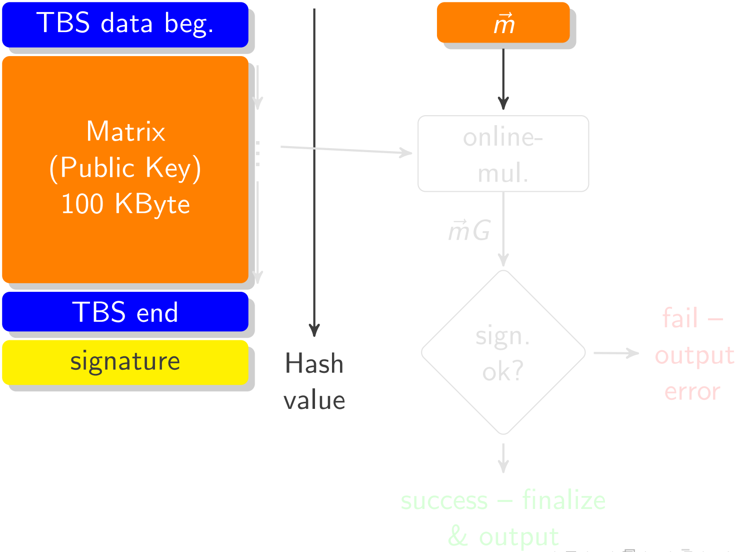
Solution for Memory-constrained Platforms

Process the certificate during receipt:



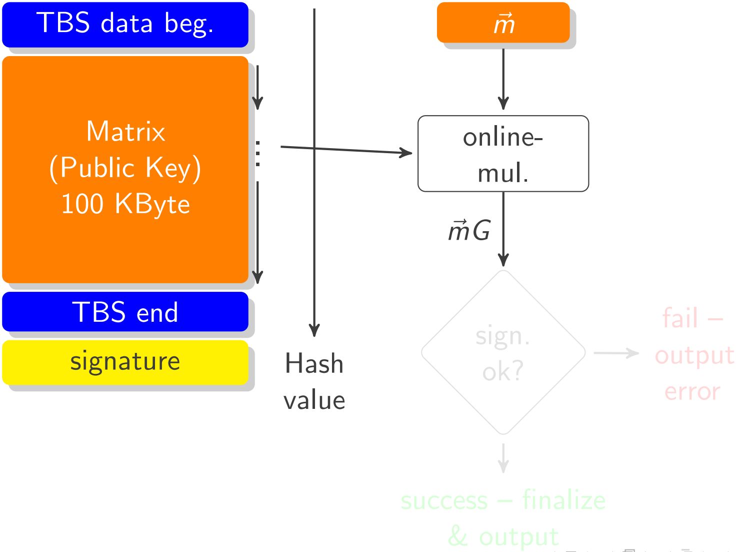
Solution for Memory-constrained Platforms

Process the certificate during receipt:



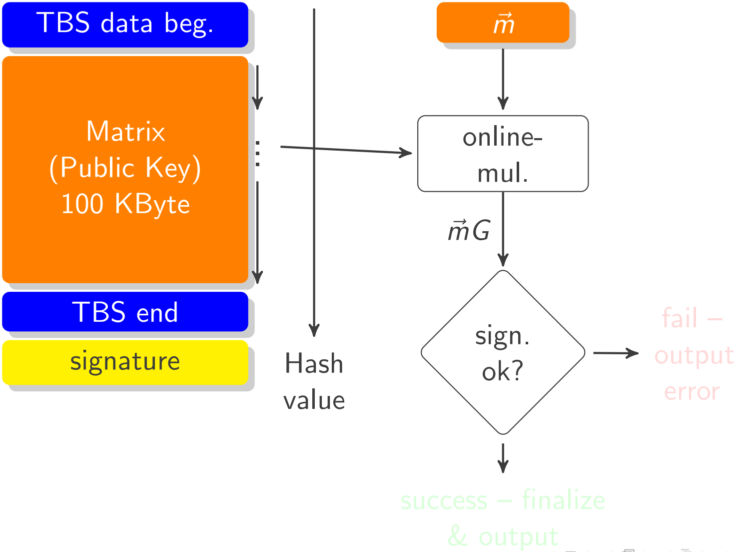
Solution for Memory-constrained Platforms

Process the certificate during receipt:



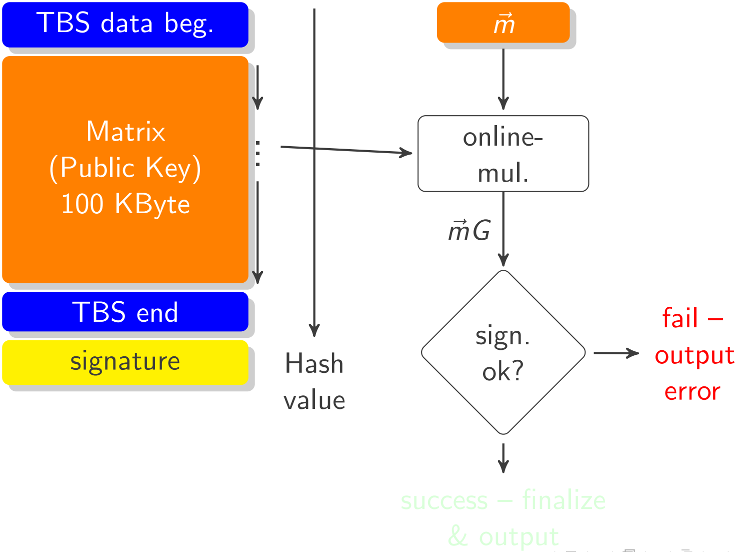
Solution for Memory-constrained Platforms

Process the certificate during receipt:



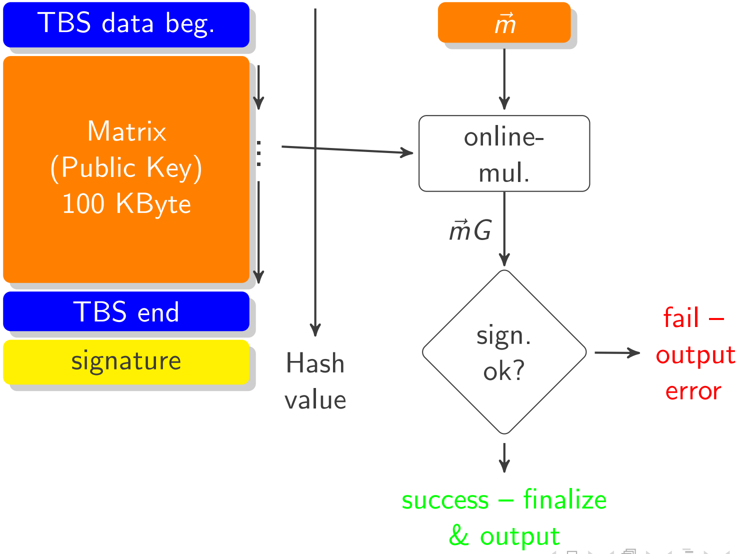
Solution for Memory-constrained Platforms

Process the certificate during receipt:



Solution for Memory-constrained Platforms

Process the certificate during receipt:



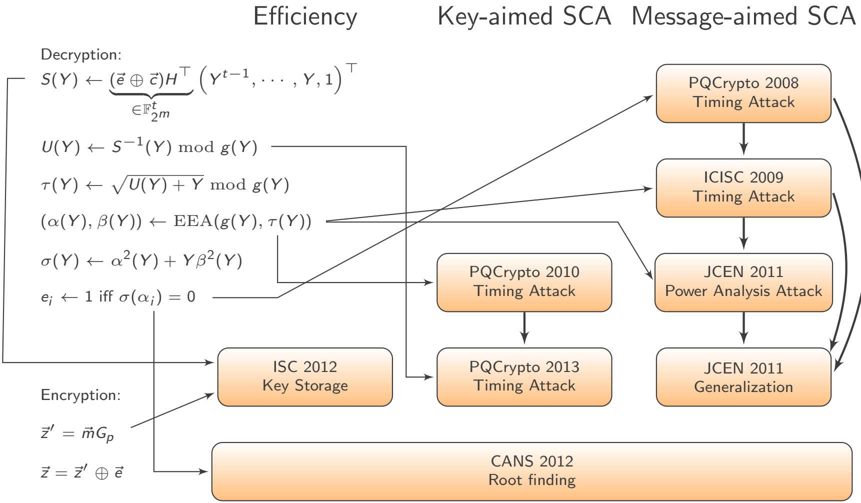
Results

- experiments: transmission rate is the limiting factor
- for a key with security level 244 bit: $t > 13s$

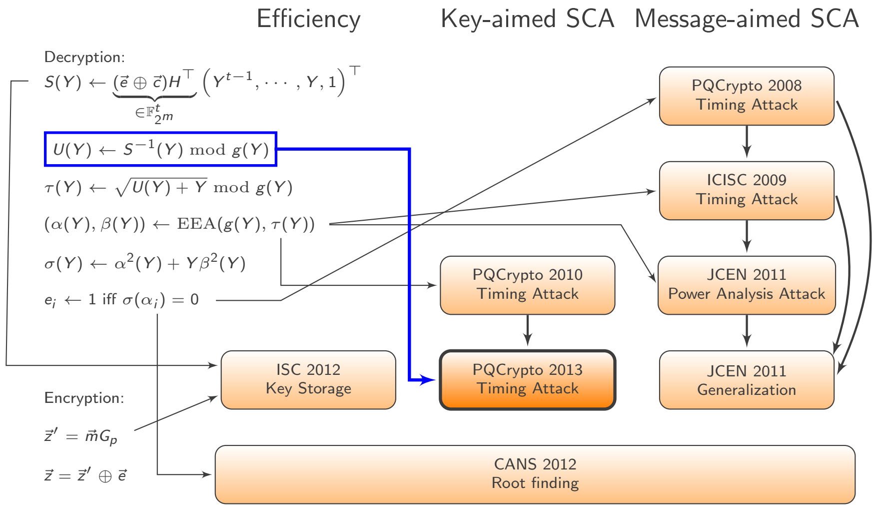
Results

- experiments: transmission rate is the limiting factor
- for a key with security level 244 bit: $t > 13s$

Overview


Efficiency
Key-aimed SCA
Message-aimed SCA
Decryption:
$S(Y) \leftarrow \underbrace{(\vec{e} \oplus \vec{c})H^\top}_{\in \mathbb{F}_{2^m}^t} \left(Y^{t-1}, \dots, Y, 1\right)^\top$
$U(Y) \leftarrow S^{-1}(Y) \bmod g(Y)$
$\tau(Y) \leftarrow \sqrt{U(Y) + Y} \bmod g(Y)$
$(\alpha(Y), \beta(Y)) \leftarrow \mathrm{EEA}(g(Y), \tau(Y))$
$\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$
$e_i \leftarrow 1$ iff $\sigma(\alpha_i) = 0$
Encryption:
$\vec{z}' = \vec{m}G_p$
$\vec{z} = \vec{z}' \oplus \vec{e}$
PQCrypto 2008
Timing Attack
ICISC 2009
Timing Attack
JCEN 2011
Power Analysis Attack
JCEN 2011
Generalization
PQCrypto 2010
Timing Attack
PQCrypto 2013
Timing Attack
ISC 2012
Key Storage
CANS 2012
Root finding

Timing Attack against the secret Support



Timing Attack against the secret Support

secret key:

$$g(Y) \qquad \Gamma = (\alpha_0, \alpha_1, \ldots \alpha_{n-1})$$

$$\begin{array}{lcccccccccccc}
\vec{e} = & (& 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots \;) \\
\text{indexes:} & & 0 & 1 & \ldots & & f_1 & & & & f_2 & & \\
\hline
 & & & & & & \alpha_{f_1} & & & & \alpha_{f_2} & &
\end{array}$$

$\sigma(Y) = \prod_{i=0}^{w-1}(\alpha_{f_i} - Y)$

Timing Attack against the secret Support

secret key:

$$g(Y) \qquad \Gamma = (\alpha_0, \alpha_1, \dots \alpha_{n-1})$$

$\vec{e} =$	(	0	0	...	0	1	0	...	0	1	0	...	)
indexes:		0	1	...		f_1				f_2			
						α_{f_1}				α_{f_2}			

$$\sigma(Y) = \prod_{i=0}^{w-1} (\alpha_{f_i} - Y)$$

Timing Attack against the secret Support

secret key:

$$g(Y) \qquad \Gamma = (\alpha_0, \alpha_1, \ldots \alpha_{n-1})$$

$$\begin{array}{lcccccccccccc} \vec{e} = & (& 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 1 & 0 & \ldots \;) \\ \text{indexes:} & & 0 & 1 & \ldots & & f_1 & & & & f_2 & & \\ \hline & & & & & & \alpha_{f_1} & & & & \alpha_{f_2} & & \end{array}$$

$$\sigma(Y) = \prod_{i=0}^{w-1} (\alpha_{f_i} - Y)$$

Timing Attack against the secret Support

secret key:

$$g(Y) \qquad \Gamma = (\alpha_0, \alpha_1, \dots \alpha_{n-1})$$

$$\begin{array}{lcccccccccccc}
\vec{e} = & (& 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & 0 & \dots \;) \\
\text{indexes:} & & 0 & 1 & \dots & & f_1 & & & & f_2 & & \\
\hline
 & & & & & & \alpha_{f_1} & & & & \alpha_{f_2} & &
\end{array}$$

$$\sigma(Y) = \prod_{i=0}^{w-1} (\alpha_{f_i} - Y)$$

Overview of the Attack

- Timing vulnerabilities:
  - for $w = 4$: linear equations
  - for $w = 1$: zero element
  - for $w = 6$: cubic equations

- Timing vulnerabilities:
  - for $w = 4$: linear equations
  - for $w = 1$: zero element
  - for $w = 6$: cubic equations

Overview of the Attack

- Timing vulnerabilities:
  - for $w = 4$: linear equations
  - for $w = 1$: zero element
  - for $w = 6$: cubic equations

Overview of the Attack

- Timing vulnerabilities:
  - for $w = 4$: linear equations
  - for $w = 1$: zero element
  - for $w = 6$: cubic equations

Source of timing differences for weight 4 error vectors

- Syndrome

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \bmod g(Y)$$

- If $w \leq t/2$
- then $\sigma(Y)$ can be found be EEA
- (break once $\deg(r_i(Y)) \leq (t/2) - 1$)
- $\rightarrow$ information about an intermediate iteration where coefficient $= \sigma(Y)$

Source of timing differences for weight 4 error vectors

- Syndrome

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \mod g(Y)$$

- If $w \leq t/2$
- then $\sigma(Y)$ can be found be EEA
- (break once $\deg(r_i(Y)) \leq (t/2) - 1$)
- $\rightarrow$ information about an intermediate iteration where coefficient $= \sigma(Y)$

Source of timing differences for weight 4 error vectors

- Syndrome

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \mod g(Y)$$

- If $w \leq t/2$
- then $\sigma(Y)$ can be found be EEA
- (break once $\deg(r_i(Y)) \leq (t/2) - 1$)
- $\rightarrow$ information about an intermediate iteration where coefficient $= \sigma(Y)$

Source of timing differences for weight 4 error vectors

- Syndrome

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \mod g(Y)$$

- If $w \leq t/2$
- then $\sigma(Y)$ can be found be EEA
- (break once $\deg(r_i(Y)) \leq (t/2) - 1$)
- $\rightarrow$ information about an intermediate iteration where coefficient $= \sigma(Y)$

Source of timing differences for weight 4 error vectors

- Syndrome

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \bmod g(Y)$$

- If $w \leq t/2$
- then $\sigma(Y)$ can be found be EEA
- (break once $\deg(r_i(Y)) \leq (t/2) - 1$)
- $\rightarrow$ information about an intermediate iteration where coefficient $= \sigma(Y)$

The Syndrome Inversion EEA for $w = 4$

$$S(Y) \equiv \sum_{i=1}^{4} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \equiv \frac{\sigma_3 Y^2 \oplus \sigma_1}{Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y \oplus \sigma_0} \mod g(Y)$$

- maximal number of iterations $M = \deg(\Omega(Y)) + \deg(\sigma(Y))$
- if $\sigma_3 = 0$, then M smaller than otherwise
- $\rightarrow$ fewer iterations, smaller timing
- $\sigma_3 = \alpha_{f_1} \oplus \alpha_{f_2} \oplus \alpha_{f_3} \oplus \alpha_{f_4} = 0$

The Syndrome Inversion EEA for $w = 4$

$$S(Y) \equiv \sum_{i=1}^{4} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \equiv \frac{\sigma_3 Y^2 \oplus \sigma_1}{Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y \oplus \sigma_0} \bmod g(Y)$$

- maximal number of iterations $M = \deg(\Omega(Y)) + \deg(\sigma(Y))$
- if $\sigma_3 = 0$, then M smaller than otherwise
- $\rightarrow$ fewer iterations, smaller timing
- $\sigma_3 = \alpha_{f_1} \oplus \alpha_{f_2} \oplus \alpha_{f_3} \oplus \alpha_{f_4} = 0$

The Syndrome Inversion EEA for $w = 4$

$$S(Y) \equiv \sum_{i=1}^{4} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \equiv \frac{\sigma_3 Y^2 \oplus \sigma_1}{Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y \oplus \sigma_0} \mod g(Y)$$

- maximal number of iterations $M = \deg(\Omega(Y)) + \deg(\sigma(Y))$
- if $\sigma_3 = 0$, then M smaller than otherwise
- $\rightarrow$ fewer iterations, smaller timing
- $\sigma_3 = \alpha_{f_1} \oplus \alpha_{f_2} \oplus \alpha_{f_3} \oplus \alpha_{f_4} = 0$

$$S(Y) \equiv \sum_{i=1}^{4} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \equiv \frac{\sigma_3 Y^2 \oplus \sigma_1}{Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y \oplus \sigma_0} \mod g(Y)$$

- maximal number of iterations $M = \deg(\Omega(Y)) + \deg(\sigma(Y))$
- if $\sigma_3 = 0$, then M smaller than otherwise
- $\rightarrow$ fewer iterations, smaller timing
- $\sigma_3 = \alpha_{f_1} \oplus \alpha_{f_2} \oplus \alpha_{f_3} \oplus \alpha_{f_4} = 0$

The Syndrome Inversion EEA for $w = 4$

$$S(Y) \equiv \sum_{i=1}^{4} \frac{1}{Y \oplus \alpha_{f_i}} \equiv \frac{\Omega(Y)}{\sigma(Y)} \equiv \frac{\sigma_3 Y^2 \oplus \sigma_1}{Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y \oplus \sigma_0} \mod g(Y)$$

- maximal number of iterations $M = \deg(\Omega(Y)) + \deg(\sigma(Y))$
- if $\sigma_3 = 0$, then M smaller than otherwise
- $\rightarrow$ fewer iterations, smaller timing
- $\sigma_3 = \alpha_{f_1} \oplus \alpha_{f_2} \oplus \alpha_{f_3} \oplus \alpha_{f_4} = 0$

Weight 6 Vulnerability

$$S(Y) \equiv \frac{\sigma_5 Y^4 \oplus \sigma_3 Y^2 \oplus \sigma_1}{Y^6 \oplus \sigma_5 Y^5 \oplus \sigma_4 Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y + \sigma_0} \mod g(Y),$$

- $\sigma_5 = \sum_{i=1}^{6} \alpha_{f_i}$
- $\sigma_3 = \sum_{j=3}^{6} \sum_{k=2}^{j-1} \sum_{l=1}^{k-1} \alpha_{f_j} \alpha_{f_k} \alpha_{f_l} = 0$

Weight 6 Vulnerability

$$S(Y) \equiv \frac{\sigma_5 Y^4 \oplus \sigma_3 Y^2 \oplus \sigma_1}{Y^6 \oplus \sigma_5 Y^5 \oplus \sigma_4 Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y + \sigma_0} \bmod g(Y),$$

- $\sigma_5 = \sum_{i=1}^{6} \alpha_{f_i}$
- $\sigma_3 = \sum_{j=3}^{6} \sum_{k=2}^{j-1} \sum_{l=1}^{k-1} \alpha_{f_j} \alpha_{f_k} \alpha_{f_l} = 0$

Weight 6 Vulnerability

$$S(Y) \equiv \frac{\sigma_5 Y^4 \oplus \sigma_3 Y^2 \oplus \sigma_1}{Y^6 \oplus \sigma_5 Y^5 \oplus \sigma_4 Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y + \sigma_0} \mod g(Y),$$

- $\sigma_5 = \sum_{i=1}^{6} \alpha_{f_i}$
- $\sigma_3 = \sum_{j=3}^{6} \sum_{k=2}^{j-1} \sum_{l=1}^{k-1} \alpha_{f_j} \alpha_{f_k} \alpha_{f_l} = 0$

Building the Attack

- from the linear equations:

α_0	α_1	$\dots$	α_i	$\dots$	α_{n-m-3}	α_{n-m-2}	β_0	$\dots$	β_{m-1}
1	0	$\dots$	0	$\dots$	0	0	X	$\dots$	X
$\vdots$									
0	0	$\dots$	1	$\dots$	0	0	X	$\dots$	X
$\vdots$									
0	0	$\dots$	0	$\dots$	0	1	X	$\dots$	X

- $\alpha_i = \sum_{j \in B_i} \beta_j$
- $\rightarrow$ collect cubic equations s.th. system can be solved

Building the Attack

- from the linear equations:

$$
\begin{array}{cccccccc|ccc}
\alpha_0 & \alpha_1 & \dots & \alpha_i & \dots & \alpha_{n-m-3} & \alpha_{n-m-2} & \beta_0 & \dots & \beta_{m-1} \\
\hline
1 & 0 & \dots & 0 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 1 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 0 & \dots & 0 & 1 & X & \dots & X
\end{array}
$$

- $\alpha_i = \sum_{j \in B_i} \beta_j$
- $\rightarrow$ collect cubic equations s.th. system can be solved

Building the Attack

- from the linear equations:

$$
\begin{array}{ccccccc|ccc}
\alpha_0 & \alpha_1 & \dots & \alpha_i & \dots & \alpha_{n-m-3} & \alpha_{n-m-2} & \beta_0 & \dots & \beta_{m-1} \\
\hline
1 & 0 & \dots & 0 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 1 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 0 & \dots & 0 & 1 & X & \dots & X
\end{array}
$$

- $\alpha_i = \sum_{j \in B_i} \beta_j$
- $\rightarrow$ collect cubic equations s.th. system can be solved

Building the Attack

- from the linear equations:

$$
\begin{array}{ccccccc|ccc}
\alpha_0 & \alpha_1 & \dots & \alpha_i & \dots & \alpha_{n-m-3} & \alpha_{n-m-2} & \beta_0 & \dots & \beta_{m-1} \\
\hline
1 & 0 & \dots & 0 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 1 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 0 & \dots & 0 & 1 & X & \dots & X
\end{array}
$$

- $\alpha_i = \sum_{j \in B_i} \beta_j$
- $\rightarrow$ collect cubic equations s.th. system can be solved

Collecting cubic Equations

$$\Omega(Y) = \sigma_5 Y^4 \oplus \sigma_3 Y^2 \oplus \sigma_1$$

$$\begin{array}{llllllll}
C_1: & \beta_3 & \leftarrow & \beta_0, & \beta_1, & \beta_2 & \\
C_2: & \beta_4 & \leftarrow & \beta_0, & \beta_1, & \beta_2, & \beta_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \\
C_{m-3}: & \beta_{m-1} & \leftarrow & \beta_0, & \beta_1, & \dots & \beta_{m-2}
\end{array}$$

- practical timing attack on Intel Core2 Duo CPU
- number of queries ≈ millions

Collecting cubic Equations

$$\Omega(Y) = \sigma_5 Y^4 \oplus \sigma_3 Y^2 \oplus \sigma_1$$

$$\begin{array}{llllllll}
C_1: & \beta_3 & \leftarrow & \beta_0, & \beta_1, & \beta_2 & \\
C_2: & \beta_4 & \leftarrow & \beta_0, & \beta_1, & \beta_2, & \beta_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \\
C_{m-3}: & \beta_{m-1} & \leftarrow & \beta_0, & \beta_1, & \dots & \beta_{m-2}
\end{array}$$

- practical timing attack on Intel Core2 Duo CPU
- number of queries $\approx$ millions

Collecting cubic Equations

$$\Omega(Y) = \sigma_5 Y^4 \oplus \sigma_3 Y^2 \oplus \sigma_1$$

$$\begin{array}{cccccccc}
C_1: & \beta_3 & \leftarrow & \beta_0, & \beta_1, & \beta_2 & \\
C_2: & \beta_4 & \leftarrow & \beta_0, & \beta_1, & \beta_2, & \beta_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \\
C_{m-3}: & \beta_{m-1} & \leftarrow & \beta_0, & \beta_1, & \dots & \beta_{m-2}
\end{array}$$

- practical timing attack on Intel Core2 Duo CPU
- number of queries $\approx$ millions

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Conclusion

- Efficiency issues
  - handling of public key keys on embedded devices
  - investigation of a number of time-memory tradeoffs
- Implementation Security
  - message-aimed side-channel issues
  - key-aimed side-channel issues
- choice of root-finding algorithm is crucial for performance and security
- security against timing attacks is achievable
- the decryption operation can be implemented on embedded systems without hardware support
- the encryption on embedded systems remains as a problem

Contributions

Strenzke, F., Tews, E., Molter, H., Overbeck, R., Shoufan, A.: Side Channels in the McEliece PKC. In: The third international Workshop on Post-Quantum Cryptography, PQC 2008. Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2008)

Shoufan, A., **Strenzke, F.**, Molter, H., Stöttinger, M.: A Timing Attack against Patterson Algorithm in the McEliece PKC. In: Information, Security and Cryptology, ICISC 2009. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2009)

Strenzke, F.: A Timing Attack against the secret Permutation in the McEliece PKC. In: The third international Workshop on Post-Quantum Cryptography, PQC 2010. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2010)

Strenzke, F.: A Smart Card Implementation of the McEliece PKC. In: Workshop in Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices, WISTP 2010. Lecture Notes in Computer Science,
Springer Berlin / Heidelberg (2010)

Strenzke, F.: Message-aimed Side Channel and Fault Attacks against Public Key Cryptosystems with homomorphic Properties. In: Journal of Cryptographic Engineering (2011)

Molter, H.G., Stötinger, M., Shoufan, A., **Strenzke, F.**: A Simple Power Analysis Attack on a McEliece Cryptoprocessor. In: Journal of Cryptographic Engineering (2011)

Strenzke, F.: Fast and Secure Root-Finding for Code-based Cryptosystems. In: The 11th International Conference on Cryptology and Network Security, CANS 2012. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2012)

Strenzke, F.: Solutions for the Storage Problem of McEliece Public and Private Keys on Memory-constrained Platforms. In: Proceedings of the 15th international conference on Information Security, ISC 2012. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2012)

Strenzke, F.: Timing Attacks against the Syndrome Inversion in Code-based Cryptosystems. In: The fifth international Workshop on Post-Quantum Cryptography, PQC 2013. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2013)

McEliece and Niederreiter

- McEliece
  - $G_p = [\mathbb{I}|G_2] = GT \in \mathbb{F}_2^{n \times k}$
  - $G_2 \in \mathbb{F}_2^{mt \times k}$
  - $T \in \mathbb{F}_2^{k \times k}$
- Niederreiter
  - $H_p = [\mathbb{I}|H_2] = TH \in \mathbb{F}_2^{mt \times n}$
  - $H_2 \in \mathbb{F}_2^{mt \times k}$
  - secret key contains $T \in \mathbb{F}_2^{mt \times mt}$