gemalto
security to be free

Toward Secure Implementation of McEliece Decryption

Mariya Georgieva &
Frédéric de Portzamparc

Gemalto & LIP6 , 13/04/2015

Code-based Cryptography

Introduced in 1978 by McEliece

Advantages

- Very fast encryption and fast decryption, faster than RSA
- No need for crypto coprocessors
- Based on NP-hard problem (Syndrome Decoding Problem)
- Post-quantum security

Code-based Cryptography

Introduced in 1978 by McEliece

Advantages

- Very fast encryption and fast decryption, faster than RSA
- No need for crypto coprocessors
- Based on NP-hard problem (Syndrome Decoding Problem)
- Post-quantum security

Disadvantages

- Big public keys ($\approx$ 100 Kbits)

Few side-channel analysis for secure implementation ...

Code based Cryptography

Syndrome Decoding Problem

$$\underset{\text{plaintext } \mathbf{m} \in \mathbb{F}_q^k}{\begin{pmatrix} 0,\ldots,1 \end{pmatrix}} \underset{}{\overset{\text{public key } \mathbf{G}_{pk} \in \mathbb{F}_q^{k \times n}}{\begin{pmatrix} & \mathbf{G}_{pk} & \end{pmatrix}}} + \underset{\text{error } \mathbf{e} \in \mathbb{F}_q^n}{\begin{pmatrix} 1,0,\ldots,0,1 \end{pmatrix}} = \underset{\text{ciphertext } \mathbf{c} \in \mathbb{F}_q^n}{\begin{pmatrix} 1,0,\ldots,1,1 \end{pmatrix}}$$

Find $\mathbf{m}$, $\mathbf{e}$ knowing $\mathbf{G}_{pk}$, $\mathbf{c}$: NP-complete for $\mathbf{G}_{pk}$ random.

Code based Cryptography

Syndrome Decoding Problem

plaintext $\mathbf{m} \in \mathbb{F}_q^k$ · public key $\mathbf{G}_{pk} \in \mathbb{F}_q^{k \times n}$ · error $\mathbf{e} \in \mathbb{F}_q^n$ · ciphertext $\mathbf{c} \in \mathbb{F}_q^n$

$$\begin{pmatrix} 0,\ldots,1 \end{pmatrix} \begin{pmatrix} \mathbf{G}_{pk} \end{pmatrix} + \begin{pmatrix} 1,0,\ldots,0,1 \end{pmatrix} = \begin{pmatrix} 1,0,\ldots,1,1 \end{pmatrix}$$

Find **m**, **e** knowing $\mathbf{G}_{pk}$, **c**: NP-complete for $\mathbf{G}_{pk}$ random.

Definitions

- A *support*: $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{F}_{q^m}^n$, with $x_i \neq x_j$
- A polynomial $g(x) \in \mathbb{F}_{q^m}[x]$ of degree t with $g(x_i) \neq 0$.
- A Goppa code $\mathscr{G}(\mathbf{x}, g)$ is described by the secret elements $\mathbf{x}$ and $g(z)$
- T_t a t-decoder for $\mathscr{G}(\mathbf{x}, g)$, using the secret elements $\mathbf{x}$ and $g(z)$
- $\mathbf{G}$ a generator matrix of $\mathscr{G}(\mathbf{x}, g)$

McEliece Public-Key Encryption

PARAMETERS : Field size $q = 2$
PUBLIC KEY : $\mathbf{G}_{pk} = \mathbf{SGP}$ with

- $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ random matrix
- $\mathbf{P} \in \mathbb{F}_q^{n\times n}$ a random permutation matrix.

PRIVATE KEY : the t-decoder T_t , $\mathbf{S}$ and $\mathbf{P}$

Algorithm 1 McEliece Cryptosystem

ENCRYPT

1: Input $\mathbf{m} \in \mathbb{F}_q^k$.
2: Generate random $\mathbf{e} \in \mathbb{F}_q^n$ with $w_H(\mathbf{e}) = t$.
3: Output $\mathbf{c} = \mathbf{m}\mathbf{G}_{pk} + \mathbf{e}$.

DECRYPT

1: Input $\mathbf{c} \in \mathbb{F}_q^n$.
2: Compute $\bar{\mathbf{m}} = T_t(\mathbf{c}\mathbf{P}^{-1})$).
3: If decoding succeeds, output $\mathbf{S}^{-1}\bar{\mathbf{m}}$, else output $\perp$.

The Decoder

The main steps are :

- Compute the **polynomial syndrome** $S(z)$, a polynomial deduced from **c**, but depending only on **e**.
- Use the Extended Euclidean Algorithm (EEA) to compute the **error locator polynomial** $\sigma(z)$,

  roots of $\sigma(z)$ are related to the support elements x_{i_j} in the error positions i_j.
- Find the roots of $\sigma(z)$. Here $\mathbf{e} \in \mathbb{F}_2^n$, so $e_{i_j} \neq 0$ implies that $e_{i_j} = 1$.

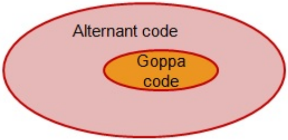
Alternant Decoder: generic for Alternant codes

1. $\mathrm{EEA}(z^{2t}, S_{Alt,\mathbf{e}}(z), t)$

Patterson Decoder: specific for binary Goppa codes

1. $\mathrm{EEA}(g(z), S_{Gop,\mathbf{e}}(z), 0)$
2. $\mathrm{EEA}(g(z), \tau, \lfloor t/2 \rfloor)$

with $\tau = \sqrt{S_{Gop,\mathbf{e}}(z)^{-1} + 1} \mod g(z)$



Extended Euclidean Algorithm

Algorithm 2 Extended Euclidean Algorithm (EEA)

Input: $a(z), b(z), \deg(a) \geqslant \deg(b), d_{fin}$
Output: $u(z), r(z)$ with $b(z)u(z) = r(z) \mod a(z)$ and $\deg(r) \leqslant d_{fin}$

$r_{-1}(z) \leftarrow a(z), r_0(z) \leftarrow b(z), u_{-1}(z) \leftarrow 1, u_0(z) \leftarrow 0,$
$i \leftarrow 0$
while $\deg(r_i(z)) > d_{fin}$ **do**
  $i \leftarrow i + 1$
  $q_i \leftarrow r_{i-2}(z)/r_{i-1}(z)$
  $r_i \leftarrow r_{i-2}(z) - q_i(z)r_{i-1}(z)$
  $u_i \leftarrow u_{i-2}(z) - q_i(z)u_{i-1}(z)$
end while
$N \leftarrow i$
return $u_N(z), r_N(z)$

The number of steps in the "while" depends on inputs $a(z)$ and $b(z)$.

Complexity is in $O(\deg(a)^2)$ fields multiplications.

Motivation and attacks

Difficulties for a secure implementation

- The operation flow of the decryption is strongly influenced by the error vector
- No information is known about the error vector before determining σ_e
- The observed or manipulated device may leak information before any detection of the attack

Motivation and attacks

Difficulties for a secure implementation

- The operation flow of the decryption is strongly influenced by the error vector
- No information is known about the error vector before determining σ_e
- The observed or manipulated device may leak information before any detection of the attack

Various attacks when using an unprotected decryption:

- on the messages (R. Avanzi et al., A. Shoufan et al.)
- on the secret key (F. Strenzke)

Motivation and attacks

Difficulties for a secure implementation

- The operation flow of the decryption is strongly influenced by the error vector
- No information is known about the error vector before determining σ_e
- The observed or manipulated device may leak information before any detection of the attack

Various attacks when using an unprotected decryption:

- on the messages (R. Avanzi et al., A. Shoufan et al.)
- on the secret key (F. Strenzke)

No satisfactory countermeasure.

Motivation and attacks

Difficulties for a secure implementation

- The operation flow of the decryption is strongly influenced by the error vector
- No information is known about the error vector before determining σ_e
- The observed or manipulated device may leak information before any detection of the attack

Various attacks when using an unprotected decryption:

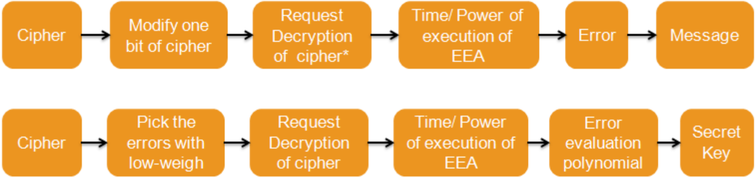
- on the messages (R. Avanzi et al., A. Shoufan et al.)
- on the secret key (F. Strenzke)

No satisfactory countermeasure.

This work

- Shows the need for an efficient countermeasure.
- Proposes such countermeasure.

Decryption oracle timing attacks



Algorithm 3 Framework for key-recovery attacks on a decryption device. (Strenzke)

INPUT: A decryption device $\mathcal{D}$, public encryption key $\mathbf{G}_{pub}$.
OUTPUT: The secret support $\mathbf{x}$.

Choose w well-chosen error weights
for $(i_1, \ldots, i_w)$ subset of $\{0, \ldots, n-1\}$ **do**
  Pick $\mathbf{e} = (0, \ldots, e_{i_1}, \ldots, e_{i_w}, \ldots, 0)$ with $w_H(\mathbf{e}) = w$.
  Request decryption $\mathcal{D}(\mathbf{e})$.
  Perform timing or power consumption analysis of $\mathcal{D}(\mathbf{e})$.
  If EEA is faster than average, deduce a polynomial condition on $x_{i_1}, \ldots, x_{i_w}$
end for
Solve the non-linear system of all the collected equations.
return Secret support $\mathbf{x} = (x_0, \ldots, x_{n-1})$.

Secret decryption key recovery attacks

Lemma (Patterson decoder)

Let $\mathscr{G}(\mathbf{x}, g(z))$ be a binary Goppa code and $S_{\mathbf{e}}(z)$ the pol. syndrome associated to an error $\mathbf{e}$ with $w_H(\mathbf{e}) \leqslant \deg(g)/2 - 1$. Write $S_{\mathbf{e}}(z) = \frac{\omega_{\mathbf{e}}(z)}{\sigma_{\mathbf{e}}(z)} \mod g(z)$. The number of iterations of the **while** *loop* $(\mathrm{EEA}(g(z), S_{\mathbf{e}}(z), 0), \mathrm{EEA}(g(z), \tau(z), \lfloor t/2 \rfloor)) = (N_I, N_K)$.

$$N_I \leqslant \deg(\omega_{\mathbf{e}}(z)) + w_H(\mathbf{e}) \text{ and } N_K \leqslant \deg(\omega_{\mathbf{e}}(z))/2. \quad (1)$$

Secret decryption key recovery attacks

Lemma (Patterson decoder)

Let $\mathscr{G}(\mathbf{x}, g(z))$ *be a binary Goppa code and* $S_{\mathbf{e}}(z)$ *the pol. syndrome associated to an error* $\mathbf{e}$ *with* $w_H(\mathbf{e}) \leqslant \deg(g)/2 - 1$*. Write* $S_{\mathbf{e}}(z) = \frac{\omega_{\mathbf{e}}(z)}{\sigma_{\mathbf{e}}(z)} \mod g(z)$*. The number of iterations of the* **while** *loop* $(\mathrm{EEA}(g(z), S_{\mathbf{e}}(z), 0), \mathrm{EEA}(g(z), \tau(z), \lfloor t/2 \rfloor)) = (N_I, N_K)$.

$$N_I \leqslant \deg(\omega_{\mathbf{e}}(z)) + w_H(\mathbf{e}) \text{ and } N_K \leqslant \deg(\omega_{\mathbf{e}}(z))/2. \tag{1}$$

Strenzke's attacks in brief

- 2010 : Observe N_K for error weights $w = 4$.

$$\omega_{\mathbf{e}}(z) = \underbrace{(x_{i_1} + x_{i_2} + x_{i_3} + x_{i_4})}_{\omega_1(\mathbf{e})} z^2 + \underbrace{x_{i_1}x_{i_2}x_{i_3} + x_{i_1}x_{i_2}x_{i_4} + x_{i_1}x_{i_3}x_{i_4} + x_{i_2}x_{i_3}x_{i_4}}_{\omega_3(\mathbf{e})}.$$

  If N_K is smaller than average $\Rightarrow x_{i_1} + x_{i_2} + x_{i_3} + x_{i_4} = 0$
  No practical attack, countermeasure proposed.
- 2011 : Observe N_I for $w = 6 \implies$ **practical attack**. Countermeasure proposed.

Secret decryption key recovery attacks

Lemma (Patterson decoder)

Let $\mathscr{G}(\mathbf{x}, g(z))$ *be a binary Goppa code and* $S_{\mathbf{e}}(z)$ *the pol. syndrome associated to an error* $\mathbf{e}$ *with* $w_H(\mathbf{e}) \leqslant \deg(g)/2 - 1$. *Write* $S_{\mathbf{e}}(z) = \frac{\omega_{\mathbf{e}}(z)}{\sigma_{\mathbf{e}}(z)} \mod g(z)$. *The number of iterations of the* **while** *loop* $(\mathrm{EEA}(g(z), S_{\mathbf{e}}(z), 0), \mathrm{EEA}(g(z), \tau(z), \lfloor t/2 \rfloor)) = (N_I, N_K)$.

$$N_I \leqslant \deg(\omega_{\mathbf{e}}(z)) + w_H(\mathbf{e}) \text{ and } N_K \leqslant \deg(\omega_{\mathbf{e}}(z))/2. \qquad (1)$$

Strenzke's attacks in brief

- 2010 : Observe N_K for error weights $w = 4$.

$$\omega_{\mathbf{e}}(z) = \underbrace{(x_{i_1} + x_{i_2} + x_{i_3} + x_{i_4})}_{\omega_1(\mathbf{e})} z^2 + \underbrace{x_{i_1}x_{i_2}x_{i_3} + x_{i_1}x_{i_2}x_{i_4} + x_{i_1}x_{i_3}x_{i_4} + x_{i_2}x_{i_3}x_{i_4}}_{\omega_3(\mathbf{e})}.$$

  If N_K is smaller than average $\Rightarrow x_{i_1} + x_{i_2} + x_{i_3} + x_{i_4} = 0$
  No practical attack, countermeasure proposed.
- 2011 : Observe N_I for $w = 6 \Longrightarrow$ **practical attack**. Countermeasure proposed.

In this paper : Extended attack bypassing previous countermeasure

Combination of first and second EEA: observe **couples** (N_I, N_K) for errors with $w = 8$

Extended Euclidean Algorithm

EEA with a flow of operations independent of the error vector

- Discards previous **message**-recovery attacks
- Discards previous **key**-recovery attacks

Extended Euclidean Algorithm

EEA with a flow of operations independent of the error vector

- Discards previous **message**-recovery attacks
- Discards previous **key**-recovery attacks

Inspired by a work of Berlekamp (VLSI)

- No clear completeness proofs found in the literature
- Never proposed for McEliece
- Fully efficient only for the Alternant decoder

Unrolling Euclidean division

Step 1: Decomposition of each euclidean division into a number of polynomial subtractions depending only on $\delta_i = \deg(q_i(z)) = \deg(r_{i-2}) - \deg(r_{i-1})$.

```
while deg(r_i(z)) > d_fin do
    i ← i + 1
    q_i ← r_{i-2}(z)/r_{i-1}(z)
    r_i ← r_{i-2}(z) − q_i(z) r_{i-1}(z)
end while
```

$$z^4 \quad \Big| \quad \alpha^{11}z^2 + \alpha^7 z + \alpha^{11}$$

$$\alpha^{11}(z^4) - z^2(\alpha^{11}z^2 + \alpha^7 z + \alpha^{11})$$

$$\alpha^7 z^3 + \alpha^{11} z^2$$

$$\alpha^{11}(\alpha^7 z^3 + \alpha^{11} z^2) - \alpha^7 z(\alpha^{11}z^2 + \alpha^7 z + \alpha^{11})$$

$$\alpha z^2 + \alpha^3 z$$

$$\alpha^{11}(\alpha z^2 + \alpha^3 z) - \alpha(\alpha^{11}z^2 + \alpha^7 z + \alpha^{11})$$

$$\alpha^6 z + \alpha^{12}$$

$$z^4 = (\alpha^4 z^2 + z + \alpha^{13})(\alpha^{11}z^2 + \alpha^7 z + \alpha^{11}) + (\alpha^3 z + \alpha^9)$$

Unrolling Euclidean division

Step 1: Decomposition of each euclidean division into a number of polynomial subtractions depending only on $\delta_i = \deg(q_i(z)) = \deg(r_{i-2}) - \deg(r_{i-1})$.

while $\deg(r_i(z)) > d_{fin}$ **do**
  $i \leftarrow i + 1$
  $q_i \leftarrow r_{i-2}(z)/r_{i-1}(z)$
  $r_i \leftarrow r_{i-2}(z) - q_i(z)r_{i-1}(z)$
end while

while $\deg(R_i(z)) > d_{fin}$ **do**
  $i \leftarrow i + 1$
  $R^{(0)}_{i-2}(z) \leftarrow R_{i-2}(z),\ \beta_i \leftarrow \mathrm{LC}(R_{i-1}(z))$
  $\Delta_i \leftarrow \deg(R_{i-2}) - \deg(R_{i-1})$
  for $j = 0, \ldots, \Delta_i$ **do**
    $\alpha_{i,j} \leftarrow R^{(j)}_{i,d_{i-2}-j}$,
    $R^{(j+1)}_{i-2}(z) \leftarrow \beta_i R^{(j)}_{i-2}(z) - \alpha_{i,j} z^{\Delta_i - j} R_{i-1}(z)$
  end for
  $R_i(z) \leftarrow R^{(\Delta_i+1)}_{i-2}(z)$,
end while

Unrolling Euclidean division

Step 1: Decomposition of each euclidean division into a number of polynomial subtractions depending only on $\delta_i = \deg(q_i(z)) = \deg(r_{i-2}) - \deg(r_{i-1})$.

while $\deg(r_i(z)) > d_{fin}$ **do**
  $i \leftarrow i + 1$
  $q_i \leftarrow r_{i-2}(z)/r_{i-1}(z)$
  $r_i \leftarrow r_{i-2}(z) - q_i(z)r_{i-1}(z)$
end while

while $\deg(R_i(z)) > d_{fin}$ **do**
  $i \leftarrow i + 1$
  $R^{(0)}_{i-2}(z) \leftarrow R_{i-2}(z)$, $\beta_i \leftarrow \mathrm{LC}(R_{i-1}(z))$
  $\Delta_i \leftarrow \deg(R_{i-2}) - \deg(R_{i-1})$
  for $j = 0, \ldots, \Delta_i$ **do**
    $\alpha_{i,j} \leftarrow R^{(j)}_{i,d_{i-2}-j}$,
    $R^{(j+1)}_{i-2}(z) \leftarrow \beta_i R^{(j)}_{i-2}(z) - \alpha_{i,j} z^{\Delta_i - j} R_{i-1}(z)$
  end for
  $R_i(z) \leftarrow R^{(\Delta_i+1)}_{i-2}(z)$,
end while

Lemma

For all $i = -1, \ldots, N$, there exists $\lambda_i \in \mathbb{F}^*_{q^m}$ such that: $R_i(z) = \lambda_i r_i(z)$,
As a consequence, $\Delta_i = \deg(R_{i-2}) - \deg(R_{i-1}) = \deg(r_{i-2}) - \deg(r_{i-1}) = \delta_i$.

Unrolling Euclidean division

Step 1: Decomposition of each euclidean division into a number of polynomial subtractions depending only on $\delta_i = \deg(q_i(z)) = \deg(r_{i-2}) - \deg(r_{i-1})$.

while $\deg(r_i(z)) > d_{fin}$ **do**
 $i \leftarrow i + 1$
 $q_i \leftarrow r_{i-2}(z)/r_{i-1}(z)$
 $r_i \leftarrow r_{i-2}(z) - q_i(z)r_{i-1}(z)$
end while

while $\deg(R_i(z)) > d_{fin}$ **do**
 $i \leftarrow i + 1$
 $R^{(0)}_{i-2}(z) \leftarrow R_{i-2}(z)$, $\beta_i \leftarrow \mathrm{LC}(R_{i-1}(z))$
 $\Delta_i \leftarrow \deg(R_{i-2}) - \deg(R_{i-1})$
 for $j = 0, \ldots, \Delta_i$ **do**
  $\alpha_{i,j} \leftarrow R^{(j)}_{i,d_{i-2}-j}$,
  $R^{(j+1)}_{i-2}(z) \leftarrow \beta_i R^{(j)}_{i-2}(z) - \alpha_{i,j} z^{\Delta_i - j} R_{i-1}(z)$
 end for
 $R_i(z) \leftarrow R^{(\Delta_i+1)}_{i-2}(z)$,
end while

Lemma

For all $i = -1, \ldots, N$*, there exists* $\lambda_i \in \mathbb{F}^*_{q^m}$ *such that:* $R_i(z) = \lambda_i r_i(z)$,
As a consequence, $\Delta_i = \deg(R_{i-2}) - \deg(R_{i-1}) = \deg(r_{i-2}) - \deg(r_{i-1}) = \delta_i$.

Problems:

- Still a while loop.
- Polynomial shift changes.

Regular polynomial shift pattern

Step 2: multiply the operand by z at each **for** iteration ("re-aligning").

Regular polynomial shift pattern

Step 2: multiply the operand by z at each **for** iteration ("re-aligning").

$$z^4 = (\alpha^4 z^2 + z + \alpha^{13})(\alpha^{11} z^2 + \alpha^7 z + \alpha^{11}) + (\alpha^3 z + \alpha^9)$$

$$\begin{array}{l|l}
z^4 & \alpha^{11} z^2 \quad +\alpha^7 z \quad +\alpha^{11} \\
\multicolumn{2}{c}{\alpha^{11}(z^4) - {\color{red}z^2}(\alpha^{11} z^2 + \alpha^7 z + \alpha^{11})} \\
\hline
\multicolumn{2}{c}{\alpha^7 z^3 \quad +\alpha^{11} z^2} \\
\multicolumn{2}{c}{\alpha^{11}(\alpha^7 z^3 + \alpha^{11} z^2) - \alpha^7 {\color{red}z}(\alpha^{11} z^2 + \alpha^7 z + \alpha^{11})} \\
\hline
\multicolumn{2}{c}{\alpha z^2 \quad +\alpha^3 z} \\
\multicolumn{2}{c}{\alpha^{11}(\alpha z^2 + \alpha^3 z) - \alpha(\alpha^{11} z^2 + \alpha^7 z + \alpha^{11})} \\
\hline
\multicolumn{2}{c}{\alpha^6 z \quad +\alpha^{12}}
\end{array}$$

$$\begin{array}{l}
z^4 \qquad \qquad \alpha^{11} z^2 + \alpha^7 z + \alpha^{11} \\
{\color{red}z}(0 \times (z^4) - 1 \times (\alpha^{11} z^2 + \alpha^7 z + \alpha^{11}) \\
\hline
\alpha^{11} z^3 + \alpha^7 z^2 + \alpha^{11} z^1 \\
{\color{red}z}(0 \times (z^4) - 1 \times (\alpha^{11} z^3 + \alpha^7 z^2 + \alpha^{11} z^1)) \\
\hline
\alpha^{11} z^4 + \alpha^7 z^3 + \alpha^{11} z^2 \\
{\color{red}z}(\alpha^{11}(z^4) - 1 \times (\alpha^{11} z^4 + \alpha^7 z^3 + \alpha^{11} z^2)) \\
\hline
\alpha^7 z^4 + \alpha^{11} z^3 \\
{\color{red}z}(\alpha^7(\alpha^{11} z^4 + \alpha^7 z^3 + \alpha^{11} z^2) - \alpha^{11}(\alpha^7 z^4 + \alpha^{11} z^3)) \\
\hline
\alpha z^4 + \alpha^3 z^3 \\
{\color{red}z}(\alpha(\alpha^{11} z^4 + \alpha^7 z^3 + \alpha^{11} z^2) - \alpha^{11}(\alpha z^4 + \alpha^3 z^3)) \\
\hline
\alpha^6 z^4 + \alpha^{12} z^3
\end{array}$$

Regular polynomial shift pattern

while $\deg(R_i(z)) > d_{fin}$ **do**
$\quad i \leftarrow i + 1$
$\quad R^{(0)}_{i-2}(z) \leftarrow R_{i-2}(z)$,
$\quad \beta_i \leftarrow \mathrm{LC}(R_{i-1}(z))$
$\quad \Delta_i \leftarrow \deg(R_{i-2}) - \deg(R_{i-1})$
$\quad$ **for** $j = 0, \ldots, \Delta_i$ **do**
$\quad\quad \alpha_{i,j} \leftarrow R^{(j)}_{i,d_{i-2}-j}$,
$\quad\quad R^{(j+1)}_{i-2}(z) \leftarrow$
$\quad\quad \beta_i R^{(j)}_{i-2}(z) - \alpha_{i,j} z^{\Delta_i - j} R_{i-1}(z)$
$\quad$ **end for**
$\quad R_i(z) \leftarrow R^{(\Delta_i+1)}_{i-2}(z)$,
end while

for $i = 1, \ldots, N$ **do**
$\quad \tilde{R}^{(0)}_{i-2}(z) \leftarrow \tilde{R}_{i-2}(z)$,
$\quad$ **for** $j = 1, \ldots, \Delta_i - 1$ **do**
$\quad\quad \tilde{R}_{i-1}(z) \leftarrow z\tilde{R}_{i-1}(z)$
$\quad$ **end for**
$\quad$ (lines above: L_1)
$\quad$ **for** $j = 0, \ldots, \Delta_i$ **do**
$\quad\quad \tilde{\alpha}_{i,j} \leftarrow \tilde{R}^{(j)}_{i,d}$, $\tilde{\beta}_i \leftarrow \tilde{R}_{i-1,d}$.
$\quad\quad \tilde{R}^{(j+1)}_{i-2}(z) \leftarrow$
$\quad\quad z\left(\tilde{\beta}_i \tilde{R}^{(j)}_{i-2}(z) - \tilde{\alpha}_{i,j} \tilde{R}_{i-1}(z)\right)$
$\quad$ **end for**
$\quad \tilde{R}_i(z) \leftarrow \tilde{R}^{(\Delta_i+1)}_{i-2}(z)$,
$\quad$ (lines from the second **for** j loop to here: L_2)
end for

Regular polynomial shift pattern

while $\deg(R_i(z)) > d_{fin}$ **do**
  $i \leftarrow i + 1$
  $R^{(0)}_{i-2}(z) \leftarrow R_{i-2}(z)$,
  $\beta_i \leftarrow \mathrm{LC}(R_{i-1}(z))$
  $\Delta_i \leftarrow \deg(R_{i-2}) - \deg(R_{i-1})$
  for $j = 0, \ldots, \Delta_i$ **do**
    $\alpha_{i,j} \leftarrow R^{(j)}_{i,d_{i-2}-j}$,
    $R^{(j+1)}_{i-2}(z) \leftarrow$
    $\beta_i R^{(j)}_{i-2}(z) - \alpha_{i,j} z^{\Delta_i - j} R_{i-1}(z)$
  end for
  $R_i(z) \leftarrow R^{(\Delta_i+1)}_{i-2}(z)$,
end while

for $i = 1, \ldots, N$ **do**
  $\tilde{R}^{(0)}_{i-2}(z) \leftarrow \tilde{R}_{i-2}(z)$,
  for $j = 1, \ldots, \Delta_i - 1$ **do** (L_1)
    $\tilde{R}_{i-1}(z) \leftarrow z\tilde{R}_{i-1}(z)$
  end for
  for $j = 0, \ldots, \Delta_i$ **do** (L_2)
    $\tilde{\alpha}_{i,j} \leftarrow \tilde{R}^{(j)}_{i,d}$, $\tilde{\beta}_i \leftarrow \tilde{R}_{i-1,d}$.
    $\tilde{R}^{(j+1)}_{i-2}(z) \leftarrow$
    $z\left(\tilde{\beta}_i \tilde{R}^{(j)}_{i-2}(z) - \tilde{\alpha}_{i,j}\tilde{R}_{i-1}(z)\right)$
  end for
  $\tilde{R}_i(z) \leftarrow \tilde{R}^{(\Delta_i+1)}_{i-2}(z)$,
end for

Lemma

For all $i = 1, \ldots, N$, $(\tilde{R}_{i-1}(z), \tilde{R}_i(z)) = (z^{d-d_{i-1}} R_{i-1}(z), z^{d-d_{i-1}+1} R_i(z))$.

Regular polynomial shift pattern

while $\deg(R_i(z)) > d_{fin}$ **do**
  $i \leftarrow i + 1$
  $R^{(0)}_{i-2}(z) \leftarrow R_{i-2}(z)$,
  $\beta_i \leftarrow \mathrm{LC}(R_{i-1}(z))$
  $\Delta_i \leftarrow \deg(R_{i-2}) - \deg(R_{i-1})$
  for $j = 0, \ldots, \Delta_i$ **do**
    $\alpha_{i,j} \leftarrow R^{(j)}_{i,d_{i-2}-j}$,
    $R^{(j+1)}_{i-2}(z) \leftarrow$
    $\beta_i R^{(j)}_{i-2}(z) - \alpha_{i,j} z^{\Delta_i - j} R_{i-1}(z)$
  end for
  $R_i(z) \leftarrow R^{(\Delta_i+1)}_{i-2}(z)$,
end while

for $i = 1, \ldots, N$ **do**
  $\tilde{R}^{(0)}_{i-2}(z) \leftarrow \tilde{R}_{i-2}(z)$,
  for $j = 1, \ldots, \Delta_i - 1$ **do**
    $\tilde{R}_{i-1}(z) \leftarrow z\tilde{R}_{i-1}(z)$
  end for } L_1
  for $j = 0, \ldots, \Delta_i$ **do**
    $\tilde{\alpha}_{i,j} \leftarrow \tilde{R}^{(j)}_{i,d}$, $\tilde{\beta}_i \leftarrow \tilde{R}_{i-1,d}$.
    $\tilde{R}^{(j+1)}_{i-2}(z) \leftarrow$
    $z\left(\tilde{\beta}_i \tilde{R}^{(j)}_{i-2}(z) - \tilde{\alpha}_{i,j}\tilde{R}_{i-1}(z)\right)$
  end for
  $\tilde{R}_i(z) \leftarrow \tilde{R}^{(\Delta_i+1)}_{i-2}(z)$, } L_2
end for

Lemma

For all $i = 1, \ldots, N$, $(\tilde{R}_{i-1}(z), \tilde{R}_i(z)) = (z^{d-d_{i-1}} R_{i-1}(z), z^{d-d_{i-1}+1} R_i(z))$.

Problems (pedagogical algorithm):

- Find N
- Find the Δ_i during the execution

Complete regular flow EEA

- For $\mathrm{EEA}(z^{2t}, S_{\mathbf{e}}(z), t)$:

$$\sum_{i=1}^{N} \delta_i = w_H(\mathbf{e}) - 1.$$

  $\Rightarrow N = 2t$
- δ is a counter for the number of shifts to re-align the operands: $\Rightarrow \Delta_i$
- Merge the loops L_1 and L_2 in a common pattern.

Complete regular flow EEA

- For $\mathrm{EEA}(z^{2t}, S_{\mathbf{e}}(z), t)$:

$$\sum_{i=1}^{N} \delta_i = w_H(\mathbf{e}) - 1.$$

$\Rightarrow N = 2t$

- δ is a counter for the number of shifts to re-align the operands: $\Rightarrow \Delta_i$
- Merge the loops L_1 and L_2 in a common pattern.

$\delta \leftarrow -1.$
for $j = 1, \ldots, 2t$ **do**
    $\alpha_j \leftarrow \hat{R}_{j-1,2t}, \beta_j \leftarrow \hat{R}_{j-2,2t}.$
    $temp_R(z) \leftarrow z \left(\alpha_j \hat{R}_{j-2}(z) - \beta_j \hat{R}_{j-1}(z) \right).$
    if $\alpha_j = 0$ (ie $\deg(\hat{R}_{j-1}) < \deg(\hat{R}_{j-2})$) **then**
        $\delta \leftarrow \delta + 1.$
    else
        $\delta \leftarrow \delta - 1.$
    end if
    if $\delta < 0$ **then**
        $(\hat{R}_j(z), \hat{R}_{j-1}(z)) \leftarrow (\hat{R}_{j-1}(z), temp_R)$
        $\delta \leftarrow 0.$
    else
        $(\hat{R}_j(z), \hat{R}_{j-1}(z)) \leftarrow (temp_R, \hat{R}_{j-2}(z))$
        $\delta \leftarrow \delta.$
    end if
end for

Complete regular flow EEA

- For $\mathrm{EEA}(z^{2t}, S_{\mathbf{e}}(z), t)$:

$$\sum_{i=1}^{N} \delta_i = w_H(\mathbf{e}) - 1.$$

$\Rightarrow N = 2t$

- δ is a counter for the number of shifts to re-align the operands: $\Rightarrow \Delta_i$
- Merge the loops L_1 and L_2 in a common pattern.

```
δ ← −1.
for j = 1, . . . , 2t do
    α_j ← R̂_{j−1,2t}, β_j ← R̂_{j−2,2t}.
    temp_R(z) ← z (α_j R̂_{j−2}(z) − β_j R̂_{j−1}(z)).
    if α_j = 0 (ie deg(R̂_{j−1}) < deg(R̂_{j−2})) then
        δ ← δ + 1.
    else
        δ ← δ − 1.
    end if
    if δ < 0 then
        (R̂_j(z), R̂_{j−1}(z)) ← (R̂_{j−1}(z), temp_R)
        δ ← 0.
    else
        (R̂_j(z), R̂_{j−1}(z)) ← (temp_R, R̂_{j−2}(z))
        δ ← δ.
    end if
end for
```

Lemma

$$\hat{R}_d(z) = z^{d - w_H(\mathbf{e})+1} R_N(z) = \mu z^{d - w_H(\mathbf{e})+1} r(z)$$

Complete regular flow EEA

- For $\mathrm{EEA}(z^{2t}, S_{\mathbf{e}}(z), t)$:

$$\sum_{i=1}^{N} \delta_i = w_H(\mathbf{e}) - 1.$$

$\Rightarrow N = 2t$

- δ is a counter for the number of shifts to re-align the operands: $\Rightarrow \Delta_i$
- Merge the loops L_1 and L_2 in a common pattern.

$\delta \leftarrow -1$.
for $j = 1, \ldots, 2t$ **do**
  $\alpha_j \leftarrow \hat{R}_{j-1,2t}, \beta_j \leftarrow \hat{R}_{j-2,2t}$.
  $temp_R(z) \leftarrow z\left(\alpha_j \hat{R}_{j-2}(z) - \beta_j \hat{R}_{j-1}(z)\right)$.
  if $\alpha_j = 0$ (ie $\deg(\hat{R}_{j-1}) < \deg(\hat{R}_{j-2})$) **then**
    $\delta \leftarrow \delta + 1$.
  else
    $\delta \leftarrow \delta - 1$.
  end if
  if $\delta < 0$ **then**
    $(\hat{R}_j(z), \hat{R}_{j-1}(z)) \leftarrow (\hat{R}_{j-1}(z), temp_R)$
    $\delta \leftarrow 0$.
  else
    $(\hat{R}_j(z), \hat{R}_{j-1}(z)) \leftarrow (temp_R, \hat{R}_{j-2}(z))$
    $\delta \leftarrow \delta$.
  end if
end for

Lemma

$$\hat{R}_d(z) = z^{d - w_H(\mathbf{e}) + 1} R_N(z) = \mu z^{d - w_H(\mathbf{e}) + 1} r(z)$$

Therefore, provided 0 is not an element of $\mathbf{x}$, $\hat{R}_d(z)$ allows to recover the error positions without ambiguity. (EEA in Alternant decoder and EEA2 in Patterson decoder)

Conclusion

In this page

- Extend the attacks of Strenzke
- Propose a new EEA algorithm determining the error-locator polynomial
  - Costs always $16t^2$ field multiplications on any input (for Alternant decoder)
  - The test that depends on the secret data is followed by two balanced branches
- Provide completeness proofs

Conclusion

In this paper

- Extend the attacks of Strenzke
- Propose a new EEA algorithm determining the error-locator polynomial
  - Costs always $16t^2$ field multiplications on any input (for Alternant decoder)
  - The test that depends on the secret data is followed by two balanced branches
- Provide completeness proofs

Perspectives

- Hardware secure implementation and tests,
- other kinds of attacks (fault, memory, template...)

Thank you for your attention!