Fast and Secure Root Finding for Code-based Cryptosystems

Falko Strenzke

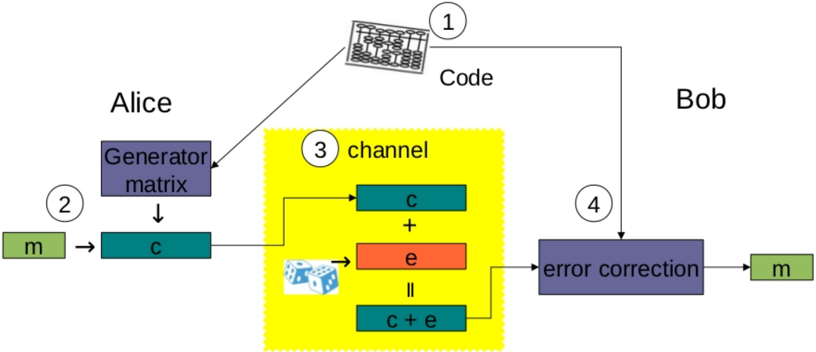
Cryptography and Computeralgebra, Department of Computer Science,
Technische Universität Darmstadt, Germany,
fstrenzke@crypto-source.de

April 13, 2015

Introduction

- Code-based Cryptography employs error corrections codes
- its security is based on the syndrome decoding problem
- secure in the presence of quantum computers
- Code-based Cryptosystems: McEliece and Niederreiter
- both use the Patterson Algorithm in decryption
- root-finding of polynomial over $\mathbb{F}_{2^m}$

Error Correcting Codes



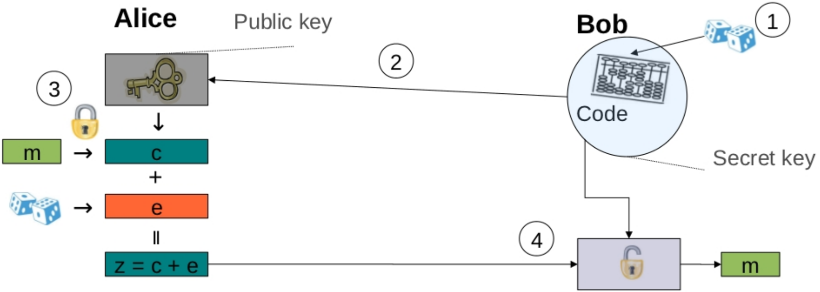
Goppa Codes

- 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

- key generation
  - choose the parameters n and t
  - generate randomly $g(Y)$ and Γ (determining the secret the code)
  - for this private code $\mathcal{C}_s$ one has a private generator matrix G_s
  - the public key is $G_p = [\mathbb{I}|G_p'] = TG_s$
- encryption: $\vec{z} = \vec{m}G_p + \vec{e}$, $\mathrm{wt}\left(\vec{e}\right) = t$
- decryption: knowing $g(Y)$ and Γ, $\vec{e}$ and thus also $\vec{m}$ can be recovered

The McEliece PKC



Syndrome Decoding

- secret key: $g(Y)$, $\Gamma = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$
- error vector $\vec{e} \in \mathbb{F}_{2^m}^n$, $\mathrm{wt}\,(\vec{e}) = t$ chosen during encryption

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

Previous Work

- Biswas, Sendrier, PQCrypto 2008: HyMES McEliece implementation
- Strenzke, Tews, Molter, Overbeck, Shoufan, PQCrypto 2008: message-aimed side-channel attack

Exhaustive Evaluation with and without Division

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

Require: the polynomial $\sigma(Y)$ over $\mathbb{F}_{2^m}$
Ensure: the set $\mathcal{E}$, where γ_i is a root of $\sigma(Y)$ if and only if $i \in \mathcal{E}$

$\mathcal{E} = \emptyset$
for $i = 0$ up to $i = n - 1$ **do**
    if $\sigma(\gamma_i) = 0$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{i\}$
        $\sigma(Y) \leftarrow \sigma(Y)/(Y \oplus \gamma_i)$
    end if
end for
return $\mathcal{E}$

$\rightarrow$ *eval-rf*, *eval-div-rf*

Berlekamp Trace Algorithm

- $\mathrm{Tr}(Y) = Y + Y^2 + Y^{2^2} + \ldots + Y^{2^{m-1}}$, and $\{\beta_1, \beta_2, \ldots, \beta_m\}$ is a standard basis of $\mathbb{F}_{2^m}$.
- initial call: BTA($\sigma(Y)$, 1)
- algorithm BTA($\Omega(Y)$, i) :

1: **if** $\deg(\Omega(Y) \leq 1)$ **then**
2:     **return** root of $\Omega(Y)$
3: **end if**
4: $\Omega_0(Y) \leftarrow \gcd(\Omega(Y), \mathrm{Tr}(\beta_i \cdot Y))$
5: $\Omega_1(Y) \leftarrow \gcd(\Omega(Y), 1 + \mathrm{Tr}(\beta_i \cdot Y))$
6: **return** $\mathrm{BTA}(\Omega_0(Y), i+1) \cup \mathrm{BTA}(\Omega_1(Y), i+1)$

$\rightarrow$ *BTA-rf*

Berlekamp Trace Algorithm - Hybrid Algorithms

- Biswas, Herbert 2009: improvement of BTA with root-finding algorithms for low degrees
- efficient root-finding for degree 2 with lookup tables
- $\rightarrow$ *BTZ_2-rf*

Root Finding with Linearized Polynomials

Definition

linearized polynomial: $L(Y) = \sum_i L_i Y^{2^i}$, where $L_i \in \mathbb{F}_{2^m}$.

Definition

affine polynomial: $A(Y) = L(Y) + \beta$ with $\beta \in \mathbb{F}_{2^m}$

- Federenko, Trifonov 2002:
- $A(x_i) = A(x_{i-1}) + L(\Delta_i), \Delta_i = x_i - x_{i-1} = \alpha^{\delta(x_i, x_{i-1})}$,
- where $\{\alpha^0, \alpha^1, \dots, \alpha^{m-1}\}$ is a standard basis of $\mathbb{F}_{2^m}$ and $\mathrm{wt}\,(x_i \oplus x_{i-1}) = 1$

Root Finding with Linearized Polynomials

$$f(Y) = f_3 Y^3 + \sum_{i=0}^{\lceil (t-4)/5 \rceil} Y^{5i} A_i(Y), \tag{1}$$

where

$$A_i(Y) = f_{5i} + \sum_{j=0}^{3} f_{5i+2^j} Y^{2^j}. \tag{2}$$

$\rightarrow$ *dcmp-rf*

Root Finding with Linearized Polynomials – Hybrid Variant

- *dcmp-div-rf* : perform divisions by found roots (after each 5 roots)

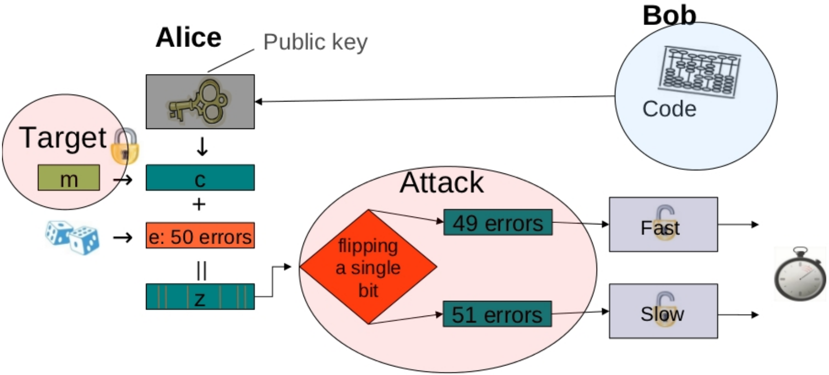
Side Channel Security Aspects of Root Finding

- Only timing attacks
- Message-aimed attacks: observe decryption and recover message
- Key-aimed attacks: observe decryption and recover key

Previously Known Message-aimed Attacks

- $\deg(\sigma(Y)) = \mathrm{wt}(\vec{e})$ when $\mathrm{wt}(\vec{e}) \leq t$
- $\rightarrow$ known TA against *eval-rf*:
- decryption time $\sim \mathrm{wt}(\vec{e})$

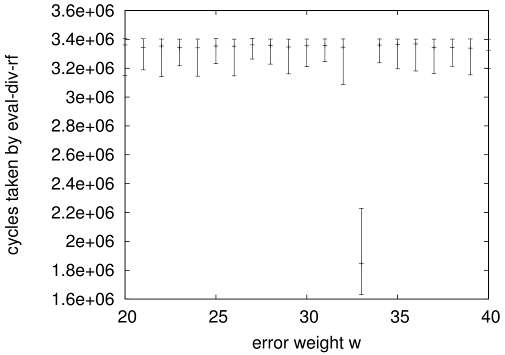
Previously Known Message-aimed Attacks



Vulnerability of *eval-div-rf*

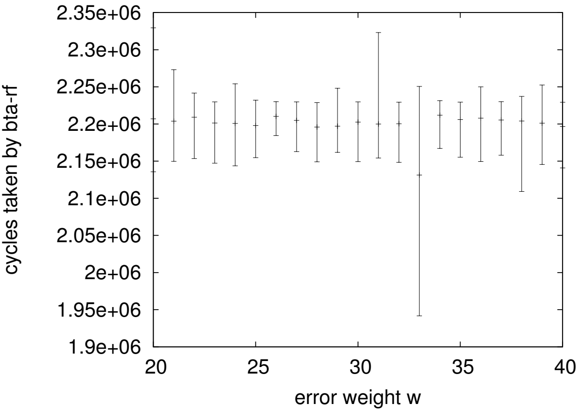
- countermeasure against this vulnerability:
- ensure $\deg(\sigma(Y)) = t$
- number of roots very small when $\mathrm{wt}(\vec{e}) > t$
- also for $\mathrm{wt}(\vec{e}) < t$ due to countermeasure
- $\rightarrow$ number of roots very small when $\mathrm{wt}(\vec{e}) \neq t$

remaining vulnerability of *eval-div-rf* ($t = 33$):



- number of roots very small when $\mathrm{wt}\left(\vec{e}\right) \neq t$
- $\rightarrow$ two-bit-flip attack is still successful:
- attacker learns when he flipped one error and one non-error position

Vulnerability of *BTA-rf*



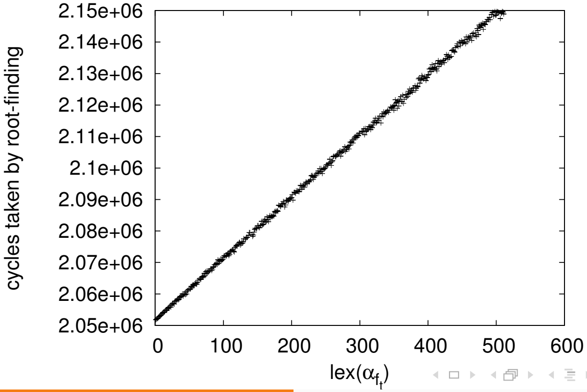
Error Positions and Support Elements

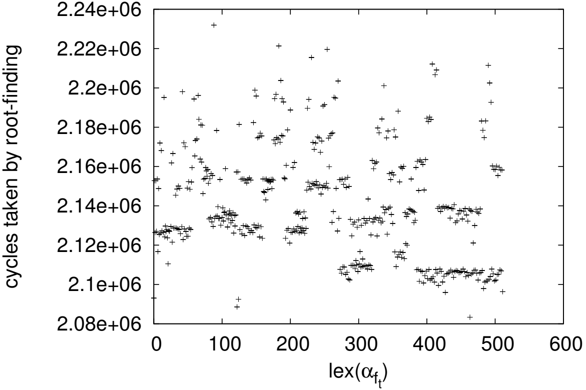
$$
\begin{array}{llcccccccccccl}
\vec{e} = & (& 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & 0 & \dots &) \\
\text{indexes:} & & 0 & 1 & \dots & & f_1 & & & & f_2 & & & \\
\hline
 & & & & & & \alpha_{f_1} & & & & \alpha_{f_2} & & &
\end{array}
$$

- $\sigma(Y) = \prod_{i=0}^{w-1}(\alpha_{f_i} - Y)$
- $\Gamma = \{\alpha_0, \alpha_1, \dots \alpha_{n-1}\}$

Vulnerability of *eval-div-rf*

- implementation evaluates $\sigma(Y)$ in order $0, 1, x, x+1, \ldots$ (lexicographical ordering)
- "support-scan": $t-1$ error positions fixed and the $t-th$ position varies (same order)



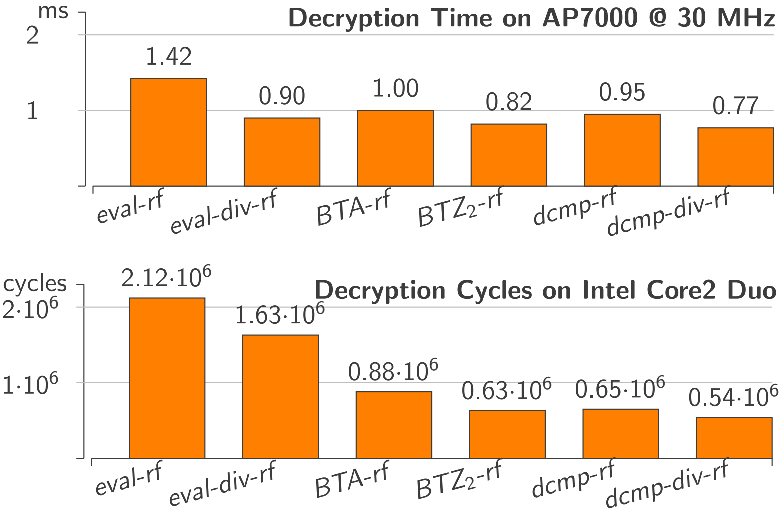


2.24e+06
2.22e+06
2.2e+06
2.18e+06
2.16e+06
2.14e+06
2.12e+06
2.1e+06
2.08e+06
cycles taken by root-finding
0
100
200
300
400
500
600
lex(α_{f_t})

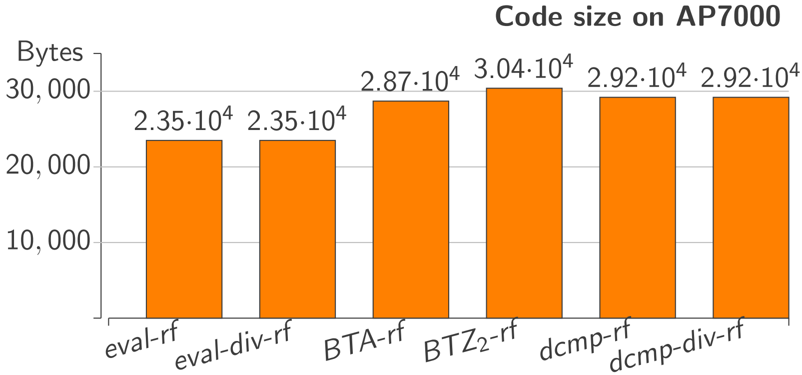
Parameters and Platforms

- $n = 2960$, $t = 56$ with more than 122 bit security
- Atmel AVR32 AP7000

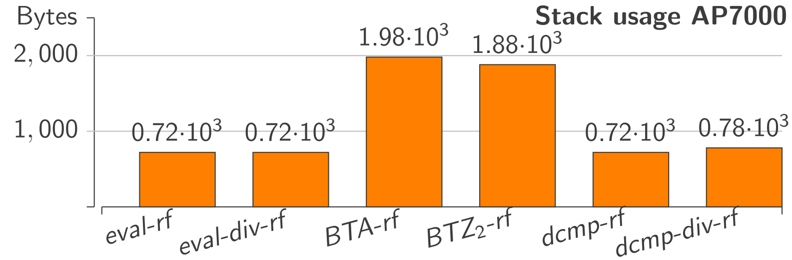
Performance – Decryption Time

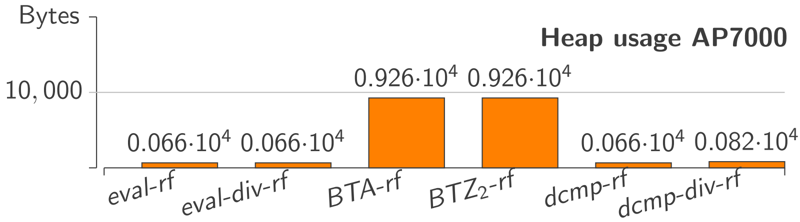


Performance – Code Size



Performance – RAM Usage





Conclusion

- many side-channel security issues in root-finding algorithms
- performance result: high RAM demands of *BTA-rf*
- *dcmp-rf* offers both side-channel security and good performance
- hardware implementation: parallelization issues

Thank you!

download the McEliece implementation and these slides:
`http://crypto-source.de`