Side-Channel Attacks on the McEliece and Niederreiter Public-Key Cryptosystems

Simon Hoerder
Roberto M. Avanzi, Dan Page, Michael Tunstall

Crypto Group, University of Bristol

University of BRISTOL

25th of February, 2011

Outline

1) A short introduction to the McEliece PKC

Simon Hoerder

University of Bristol

Outline

1) A short introduction to the McEliece PKC
2) Timing related leakage

Simon Hoerder

University of Bristol

Outline

1) A short introduction to the McEliece PKC
2) Timing related leakage
3) Leaking the Goppa polynomial

Simon Hoerder

University of Bristol

Outline

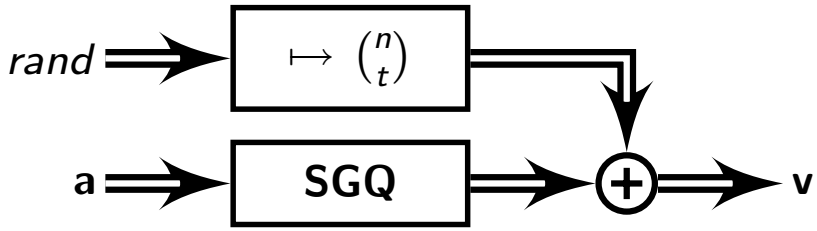
1) A short introduction to the McEliece PKC
2) Timing related leakage
3) Leaking the Goppa polynomial
4) Leakage from the constant weight encoder

Simon Hoerder

University of Bristol

The McEliece PKC I

Encryption:



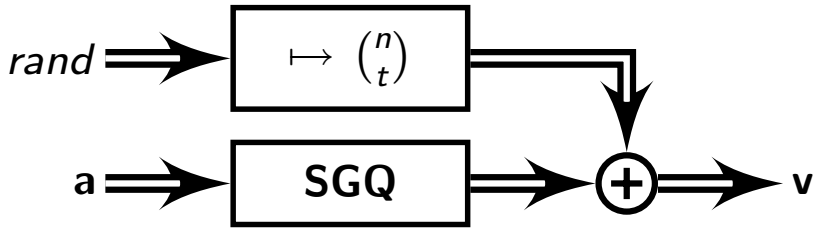
Simon Hoerder

University of Bristol

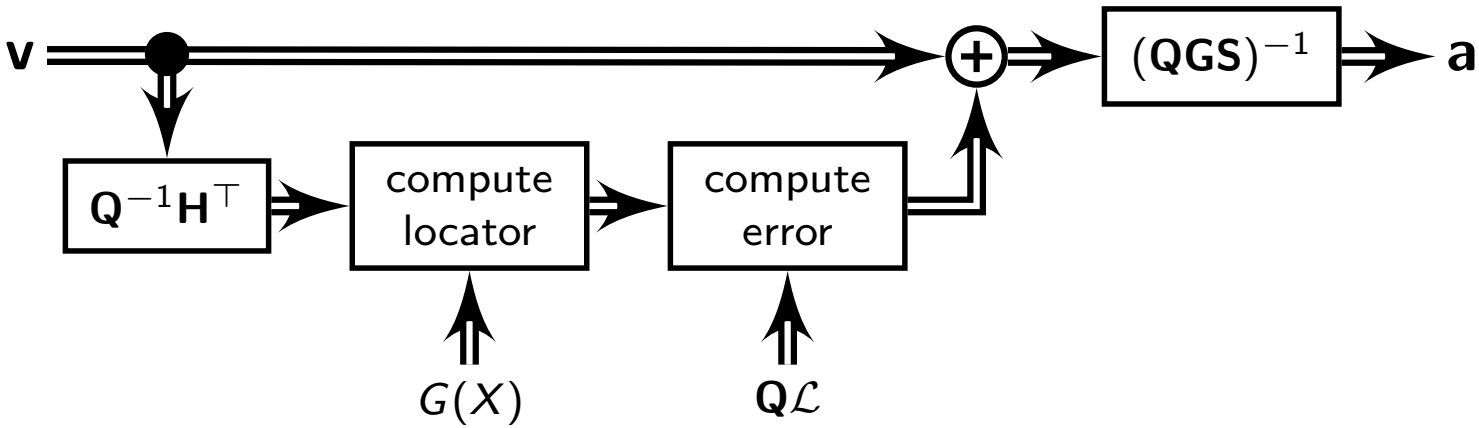
25/02/2011

The McEliece PKC I

Encryption:



Decryption:

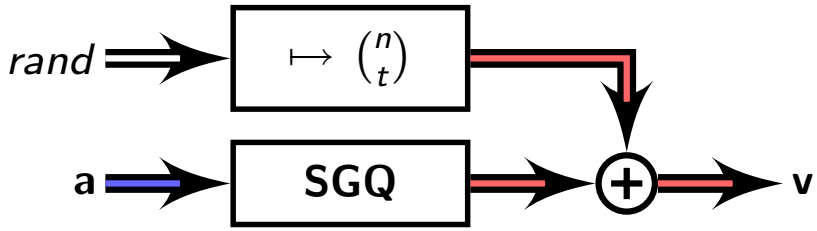


Simon Hoerder

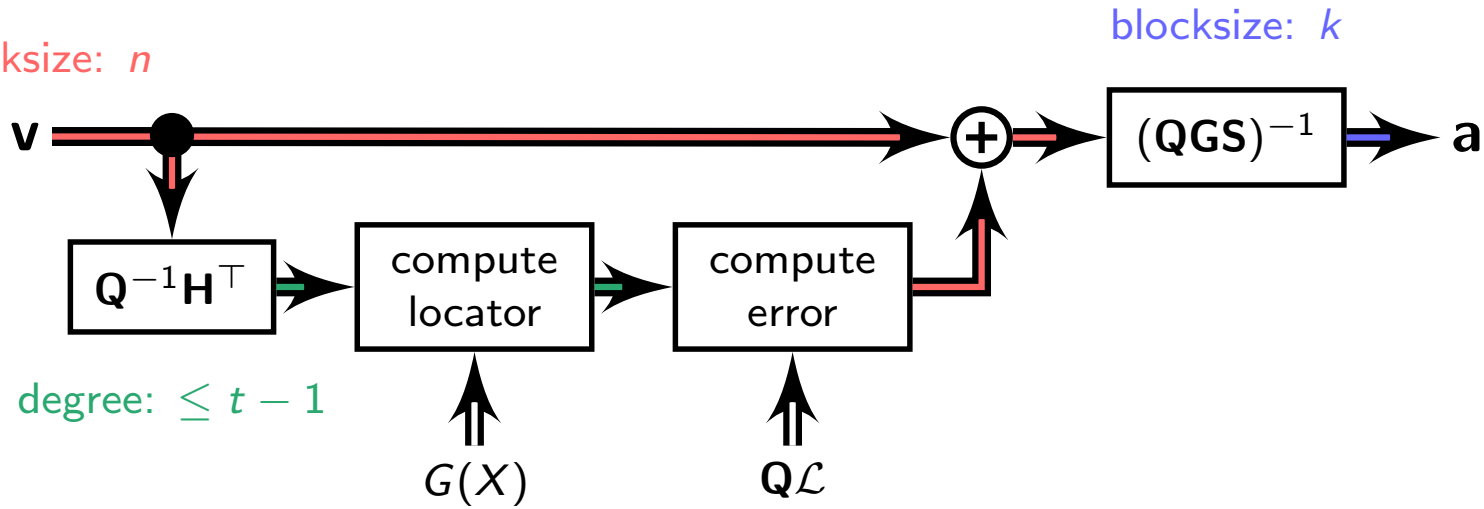
University of Bristol

The McEliece PKC I

Encryption:



Decryption:

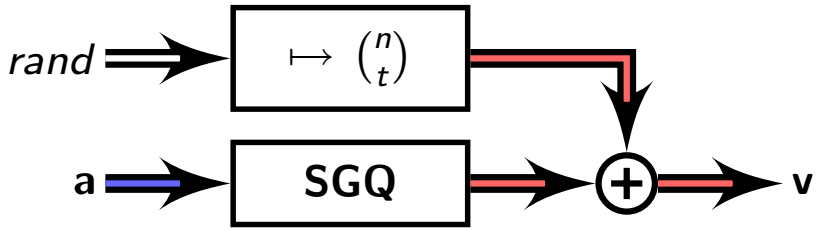


Simon Hoerder

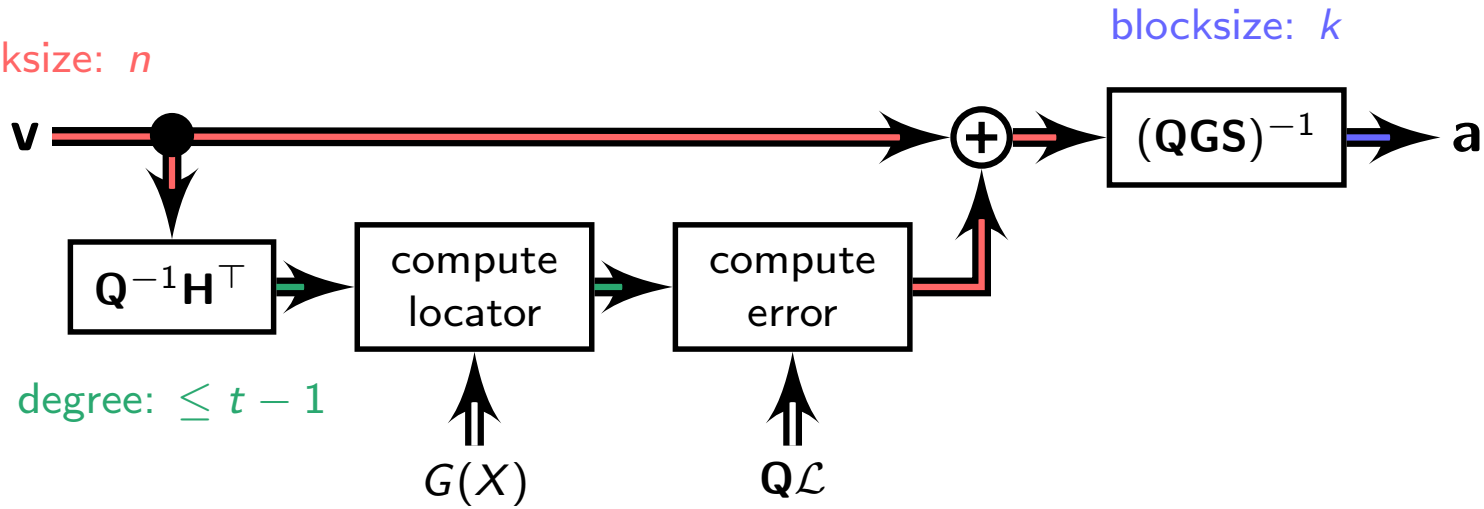
University of Bristol

The McEliece PKC I

Encryption: $\mathrm{PK}_{\mathrm{McE}} = \{n, k, t, \underbrace{(\mathbf{SGQ})}_{\mathbf{G}'} \in \mathbb{F}_2^{k \times n}\}$



Decryption: $\mathrm{SK}_{\mathrm{McE}} = \{\mathbf{Q}, \mathbf{G}, \mathbf{S}, \mathbf{H}^\top \in \mathbb{F}_{2^m}^{n \times t}, G(X), \mathcal{L}\}$



Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!

Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!
80-bit security	2048	1751	27	

Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!
80-bit security	2048	1751	27	
128-bit security	2960	2288	56	

Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!
80-bit security	2048	1751	27	
128-bit security	2960	2288	56	
256-bit security	6624	5129	115	

Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!
80-bit security	2048	1751	27	
128-bit security	2960	2288	56	
256-bit security	6624	5129	115	

Advantages:

- Suitable for post-quantum cryptography

Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!
80-bit security	2048	1751	27	
128-bit security	2960	2288	56	
256-bit security	6624	5129	115	

Advantages:

- Suitable for post-quantum cryptography
- Adds variety: not based on factorization or DL

Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!
80-bit security	2048	1751	27	
128-bit security	2960	2288	56	
256-bit security	6624	5129	115	

Advantages:

- Suitable for post-quantum cryptography
- Adds variety: not based on factorization or DL
- Relatively well researched theory

Simon Hoerder

University of Bristol

The McEliece PKC II

Parameters:

Security as in [1]	n	k	t	
original	1024	524	50	insecure!
80-bit security	2048	1751	27	
128-bit security	2960	2288	56	
256-bit security	6624	5129	115	

Advantages:

- Suitable for post-quantum cryptography
- Adds variety: not based on factorization or DL
- Relatively well researched theory
- Relatively efficient (compared to other PQ PKCs)

Simon Hoerder

University of Bristol

Timing Side-Channels

Basic idea:

- Attacker can toggle bits in the cipher text and measure the timing.

Simon Hoerder

University of Bristol

Timing Side-Channels

Basic idea:

- Attacker can toggle bits in the cipher text and measure the timing.
- Bit was in an error position: $w_H(\mathbf{e}) \rightarrow w_H(\mathbf{e}) - 1$
  - $\Rightarrow$ correctable
  - $\Rightarrow$ decryption is somewhat faster

Simon Hoerder

University of Bristol

Timing Side-Channels

Basic idea:

- Attacker can toggle bits in the cipher text and measure the timing.
- Bit was in an error position: $w_H(\mathbf{e}) \rightarrow w_H(\mathbf{e}) - 1$
  - $\Rightarrow$ correctable
  - $\Rightarrow$ decryption is somewhat faster
- Otherwise: $w_H(\mathbf{e}) \rightarrow w_H(\mathbf{e}) + 1$
  - $\Rightarrow$ uncorrectable: $\mathbf{c}' \oplus \mathbf{e}'$ instead of $\mathbf{c} \oplus \mathbf{e}$
  - $\Rightarrow$ $w_H(\mathbf{e}') = t$ instead of $w_H(\mathbf{e}) = t + 1$
  - $\Rightarrow$ decryption time remains the same

Simon Hoerder

University of Bristol

Timing Side-Channels

Basic idea:

- Attacker can toggle bits in the cipher text and measure the timing.
- Bit was in an error position: $w_H(\mathbf{e}) \rightarrow w_H(\mathbf{e}) - 1$
  - $\Rightarrow$ correctable
  - $\Rightarrow$ decryption is somewhat faster
- Otherwise: $w_H(\mathbf{e}) \rightarrow w_H(\mathbf{e}) + 1$
  - $\Rightarrow$ uncorrectable: $\mathbf{c}' \oplus \mathbf{e}'$ instead of $\mathbf{c} \oplus \mathbf{e}$
  - $\Rightarrow$ $w_H(\mathbf{e}') = t$ instead of $w_H(\mathbf{e}) = t + 1$
  - $\Rightarrow$ decryption time remains the same
- Testing all bits, attacker learns $\mathbf{e}$ using n time measurements

Simon Hoerder

University of Bristol

Timing Side-Channels

Basic idea:

- Attacker can toggle bits in the cipher text and measure the timing.
- Bit was in an error position: $w_H(\mathbf{e}) \rightarrow w_H(\mathbf{e}) - 1$
  - $\Rightarrow$ correctable
  - $\Rightarrow$ decryption is somewhat faster
- Otherwise: $w_H(\mathbf{e}) \rightarrow w_H(\mathbf{e}) + 1$
  - $\Rightarrow$ uncorrectable: $\mathbf{c}' \oplus \mathbf{e}'$ instead of $\mathbf{c} \oplus \mathbf{e}$
  - $\Rightarrow$ $w_H(\mathbf{e}') = t$ instead of $w_H(\mathbf{e}) = t + 1$
  - $\Rightarrow$ decryption time remains the same
- Testing all bits, attacker learns $\mathbf{e}$ using n time measurements
- Attacker is able to remove error and recover data

Simon Hoerder

University of Bristol

Decryption Algorithm - Timing leaks

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.

OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1 [3, 4]]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2 [3, 4]]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$

Simon Hoerder

University of Bristol

Decryption Algorithm - Timing leaks

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.

OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1 [3, 4]]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2 [3, 4]]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [EXPLOITED leak [2]]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$ [reported leak 3]

Simon Hoerder

University of Bristol

Decryption Algorithm - Timing leaks

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.

OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1 [3, 4]]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2 [3, 4]]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [EXPLOITED leak [2]]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$ [reported leak 3]

Ignored (so far) timing leak: Cache – LUTs for $\mathbb{F}_{2^m}$.

Simon Hoerder

University of Bristol

Countermeasure I

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1 [3, 4]]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2 [3, 4]]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [EXPLOITED leak [2]]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$ [reported leak 3]

- Countermeasure: Assure $\deg(u(X)) = t$ in line 5.
- Problem I: Avoid reaction attack
- Problem II: Avoid inserting roots from $\mathcal{L}$

Simon Hoerder

University of Bristol

Countermeasure II

Our Solution: Add roots from the *non-support* to $u(X)$.

Simon Hoerder

University of Bristol

Countermeasure II

Our Solution: Add roots from the *non-support* to $u(X)$.

Definition (Non-Support)

The *non-support* $\overline{\mathcal{L}}$ is defined as

- $\overline{\mathcal{L}} = \mathbb{F}_{2^m} \setminus \mathcal{L}$ iff $\mathcal{L} \subsetneq \mathbb{F}_{2^m}$, as

Simon Hoerder

University of Bristol

Countermeasure II

Our Solution: Add roots from the *non-support* to $u(X)$.

Definition (Non-Support)

The *non-support* $\overline{\mathcal{L}}$ is defined as

- $\overline{\mathcal{L}} = \mathbb{F}_{2^m} \setminus \mathcal{L}$ iff $\mathcal{L} \subsetneq \mathbb{F}_{2^m}$, as
- $\overline{\mathcal{L}} = \mathbb{F}_{2^{m'}} \setminus \mathcal{L}$ with $m' \geq m+1$ iff $|\mathcal{L}| = 2^m$ and a free choice of m' is possible or as

Simon Hoerder

University of Bristol

Countermeasure II

Our Solution: Add roots from the *non-support* to $u(X)$.

Definition (Non-Support)

The *non-support* $\overline{\mathcal{L}}$ is defined as

- $\overline{\mathcal{L}} = \mathbb{F}_{2^m} \setminus \mathcal{L}$ iff $\mathcal{L} \subsetneq \mathbb{F}_{2^m}$, as
- $\overline{\mathcal{L}} = \mathbb{F}_{2^{m'}} \setminus \mathcal{L}$ with $m' \geq m+1$ iff $|\mathcal{L}| = 2^m$ and a free choice of m' is possible or as
- $\overline{\mathcal{L}} = \mathbb{F}_{2^{xm}} \setminus \mathbb{F}_{2^m}$ with $x \geq 2$ otherwise.

Simon Hoerder

University of Bristol

Countermeasure II

Our Solution: Add roots from the *non-support* to $u(X)$.

Definition (Non-Support)

The *non-support* $\overline{\mathcal{L}}$ is defined as

- $\overline{\mathcal{L}} = \mathbb{F}_{2^m} \setminus \mathcal{L}$ iff $\mathcal{L} \subsetneq \mathbb{F}_{2^m}$, as
- $\overline{\mathcal{L}} = \mathbb{F}_{2^{m'}} \setminus \mathcal{L}$ with $m' \geq m+1$ iff $|\mathcal{L}| = 2^m$ and a free choice of m' is possible or as
- $\overline{\mathcal{L}} = \mathbb{F}_{2^{xm}} \setminus \mathbb{F}_{2^m}$ with $x \geq 2$ otherwise.

In the last case, the error computation has to be done in the extension field $\mathbb{F}_{(2^m)^x}$.

Simon Hoerder

University of Bristol

Countermeasure III

Algorithm 2 Hardened McEliece decryption with Patterson's algorithm.

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.

OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. ... [Syndrome & locator computation as in alg. 1.]

Simon Hoerder

University of Bristol

Countermeasure III

Algorithm 2 Hardened McEliece decryption with Patterson's algorithm.

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. ... [Syndrome & locator computation as in alg. 1.]
2. $u_h(X) \leftarrow u(X), \quad \overline{u(X)} \leftarrow 1$
3. **for** $i = 0$ **to** $t - 1$
4. $\quad \overline{\gamma_i} \leftarrow$ **chooseElement** $(\overline{\mathcal{L}})$
5. $\quad$ **if** $\deg(u_h(X)) \leq i$ **then** $u_h(X) \leftarrow u_h(X)(X + \overline{\gamma_i})$
6. $\quad$ **else** $\overline{u(X)} \leftarrow \overline{u(X)}(X + \overline{\gamma_i})$

Simon Hoerder

University of Bristol

Countermeasure III

Algorithm 2 Hardened McEliece decryption with Patterson's algorithm.

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. ... [Syndrome & locator computation as in alg. 1.]
2. $u_h(X) \leftarrow u(X), \quad \overline{u(X)} \leftarrow 1$
3. **for** $i = 0$ **to** $t - 1$
4. $\quad \overline{\gamma_i} \leftarrow$ **chooseElement** $(\overline{\mathcal{L}})$
5. $\quad$ **if** $\deg(u_h(X)) \leq i$ **then** $u_h(X) \leftarrow u_h(X)(X + \overline{\gamma_i})$
6. $\quad$ **else** $\overline{u(X)} \leftarrow \overline{u(X)}(X + \overline{\gamma_i})$
7. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
8. $\quad$ **if** $(u_h(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$
9. $\quad$ **else** $e_i \leftarrow 0$
10. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$

Simon Hoerder

University of Bristol

Countermeasure III

Algorithm 2 Hardened McEliece decryption with Patterson's algorithm.

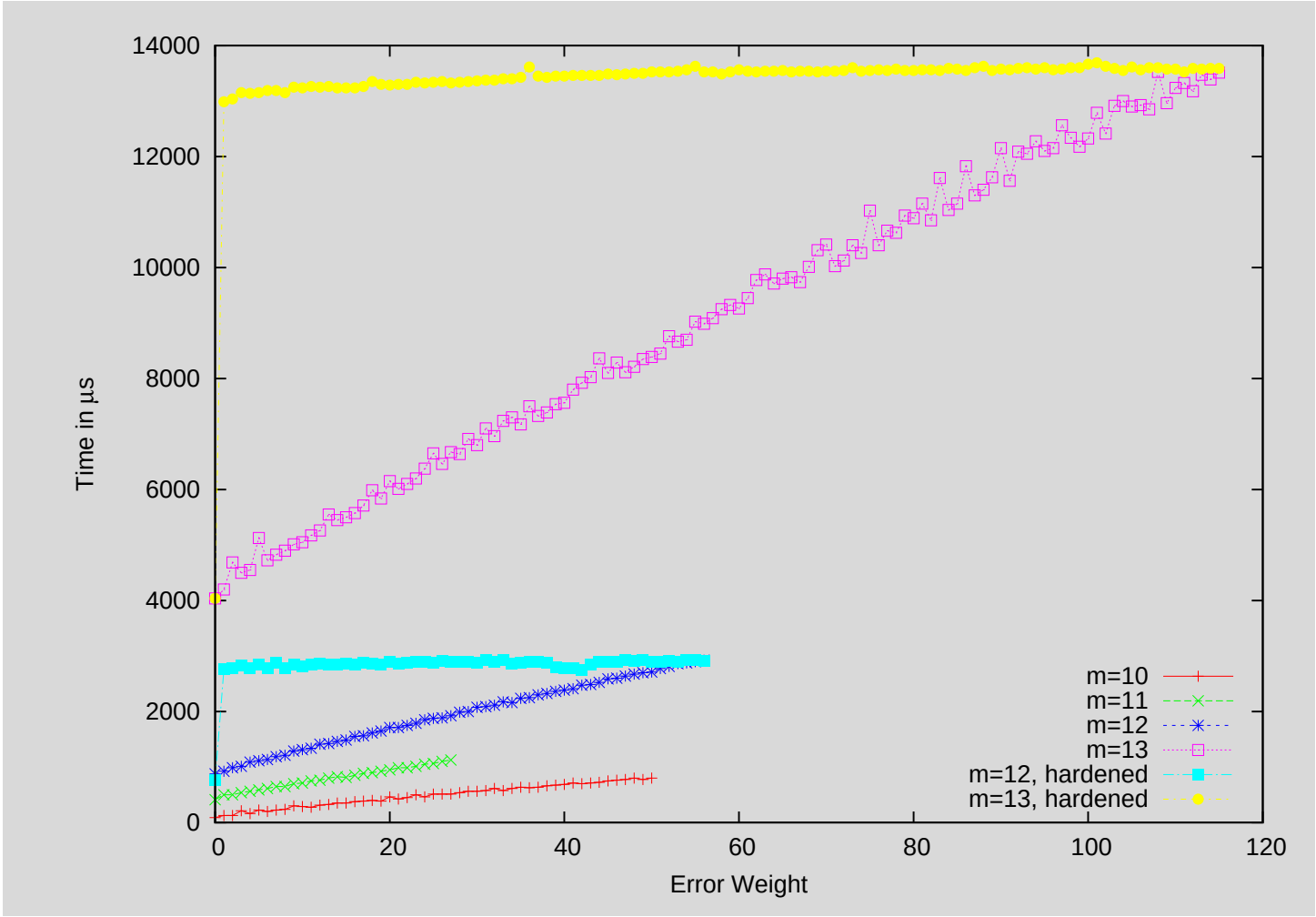
INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. ... [Syndrome & locator computation as in alg. 1.]
2. $u_h(X) \leftarrow u(X), \quad \overline{u(X)} \leftarrow 1$
3. **for** $i = 0$ **to** $t - 1$
4. $\quad \overline{\gamma_i} \leftarrow$ **chooseElement** $(\overline{\mathcal{L}})$
5. $\quad$ **if** $\deg(u_h(X)) \leq i$ **then** $u_h(X) \leftarrow u_h(X)(X + \overline{\gamma_i})$
6. $\quad$ **else** $\overline{u(X)} \leftarrow \overline{u(X)}(X + \overline{\gamma_i})$
7. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
8. $\quad$ **if** $(u_h(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$
9. $\quad$ **else** $e_i \leftarrow 0$
10. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$

Simon Hoerder

University of Bristol

Results



Simon Hoerder

University of Bristol

Outlook on Timing Leaks

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [FIXED leak]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$ [reported leak 3]

Simon Hoerder

University of Bristol

Outlook on Timing Leaks

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [FIXED leak]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$ [reported leak 3]

- Reported leaks 1 and 2: could be fixed using $\overline{\mathbf{H}}$ and

$$w_H(\mathbf{e}) \leftarrow \mathit{simple\_but\_unlikely\_function}(\mathbf{v})$$

Simon Hoerder

University of Bristol

Outlook on Timing Leaks

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [FIXED leak]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$ [reported leak 3]

- Reported leaks 1 and 2: could be fixed using $\overline{\mathbf{H}}$ and
$$w_H(\mathbf{e}) \leftarrow \mathit{simple\_but\_unlikely\_function}(\mathbf{v})$$
- But: they are a lot more difficult to exploit.

Simon Hoerder
University of Bristol

Outlook on Timing Leaks

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.
OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [reported leak 1]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [reported leak 2]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [FIXED leak]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$ [reported leak 3]

- Reported leaks 1 and 2: could be fixed using $\overline{\mathbf{H}}$ and
$$w_H(\mathbf{e}) \leftarrow \mathit{simple\_but\_unlikely\_function}(\mathbf{v})$$
- But: they are a lot more difficult to exploit.
- Opinion: Cache leaks more serious (than remaining leaks).

Simon Hoerder

University of Bristol

Leaking the Goppa Polynomial

- Possible side channels: DPA, Cache Attacks

Simon Hoerder

University of Bristol

Leaking the Goppa Polynomial

- Possible side channels: DPA, Cache Attacks
- Weak point: Finite field arithmetic
- Finite field arithmetic implemented using lookup tables
  - *"constant"* time multiplications and exponentiations
  - no branches required but:
  - tables need a lot of cache:

$$2.5\text{kB for } m = 10, \ \ldots, \ 26\text{kB for } m = 13$$

Simon Hoerder

University of Bristol

Leaking the Goppa Polynomial

- Possible side channels: DPA, Cache Attacks
- Weak point: Finite field arithmetic
- Finite field arithmetic implemented using lookup tables
  - *"constant"* time multiplications and exponentiations
  - no branches required but:
  - tables need a lot of cache:

    $$\text{2.5kB for } m = 10, \ \ldots, \ \text{26kB for } m = 13$$

- Steps using $G(X)$:
  - modular inversion (Line 1 of Alg. 1)
  - modular square root computation (Line 3 of Alg. 1)
  - $\mathtt{EEA}_{\mathtt{DEC}}$ (Line 4 of Alg. 1)

Simon Hoerder

University of Bristol

Modular Square-Root Computation

$$\sqrt{s(X)+X} \bmod G(X) \equiv \left(\sqrt{\tilde{s}_1(X)} + \underbrace{\sqrt{X} \bmod G(X)}_{\text{constant}} \sqrt{\tilde{s}_2(X)} \right) \bmod G(X)$$

Simon Hoerder

University of Bristol

Modular Square-Root Computation

$$\sqrt{s(X)+X}^{\bmod G(X)} \equiv \left(\sqrt{\tilde{s}_1(X)} + \underbrace{\sqrt{X}^{\bmod G(X)}}_{\text{constant}} \sqrt{\tilde{s}_2(X)}\right) \bmod G(X)$$

1) Learn $\sqrt{X}^{\bmod G(X)}$

Simon Hoerder

University of Bristol

Modular Square-Root Computation

$$\sqrt{s(X)+X}^{\bmod G(X)} \equiv \left(\sqrt{\tilde{s}_1(X)} + \underbrace{\sqrt{X}^{\bmod G(X)}}_{\text{constant}} \sqrt{\tilde{s}_2(X)}\right) \bmod G(X)$$

1) Learn $\sqrt{X}^{\bmod G(X)}$
2) Compute

$$(\sqrt{X}^{\bmod G(X)})^2 + X = z(X)G(X)$$

Simon Hoerder

University of Bristol

Modular Square-Root Computation

$$\sqrt{s(X)+X}^{\bmod G(X)} \equiv \left(\sqrt{\tilde{s}_1(X)} + \underbrace{\sqrt{X}^{\bmod G(X)}}_{\text{constant}} \sqrt{\tilde{s}_2(X)}\right) \bmod G(X)$$

1) Learn $\sqrt{X}^{\bmod G(X)}$
2) Compute
$$(\sqrt{X}^{\bmod G(X)})^2 + X = z(X)G(X)$$
3) Use
   - $\deg(z(X)G(X)) < 2t$, $\deg(G(X)) = t$
   - t relatively small

   $\Rightarrow$ factorization of $z(X)G(X)$ possible.

Simon Hoerder

University of Bristol

Modular Square-Root Computation

$$\sqrt{s(X)+X}^{\bmod G(X)} \equiv \left(\sqrt{\tilde{s}_1(X)} + \underbrace{\sqrt{X}^{\bmod G(X)}}_{\text{constant}} \sqrt{\tilde{s}_2(X)}\right) \bmod G(X)$$

1) Learn $\sqrt{X}^{\bmod G(X)}$
2) Compute
$$(\sqrt{X}^{\bmod G(X)})^2 + X = z(X)G(X)$$
3) Use
   - $\deg(z(X)G(X)) < 2t$, $\deg(G(X)) = t$
   - t relatively small

   $\Rightarrow$ factorization of $z(X)G(X)$ possible.

$\Rightarrow$ Obtain $G(X)$.

Simon Hoerder

University of Bristol

SPA against the Constant Weight Encoder

- Required for both: McEliece and Niederreiter

Simon Hoerder

University of Bristol

SPA against the Constant Weight Encoder

- Required for both: McEliece and Niederreiter
- Sendrier's CWE ([5]) is good choice for
  - encoding speed
  - almost optimal but input dependend code rate

SPA against the Constant Weight Encoder

- Required for both: McEliece and Niederreiter
- Sendrier's CWE ([5]) is good choice for
  - encoding speed
  - almost optimal but input dependend code rate
- No constant secret involved $\Rightarrow$ DPA not applicable

Simon Hoerder

University of Bristol

SPA against the Constant Weight Encoder

- Required for both: McEliece and Niederreiter
- Sendrier's CWE ([5]) is good choice for
  - encoding speed
  - almost optimal but input dependend code rate
- No constant secret involved $\Rightarrow$ DPA not applicable
- Leaks some input bits to SPA (data dependend branch!):
  - Average: $\approx 26.6\%$ input bits leaked (10^7 random samples)
  - Minimum (for our samples): $\approx 13.9\%$
  - Maximum (for our samples): $\approx 53.0\%$, worst case: 100%

Simon Hoerder

University of Bristol

SPA against the Constant Weight Encoder

- Required for both: McEliece and Niederreiter
- Sendrier's CWE ([5]) is good choice for
  - encoding speed
  - almost optimal but input dependend code rate
- No constant secret involved $\Rightarrow$ DPA not applicable
- Leaks some input bits to SPA (data dependend branch!):
  - Average: $\approx 26.6\%$ input bits leaked (10^7 random samples)
  - Minimum (for our samples): $\approx 13.9\%$
  - Maximum (for our samples): $\approx 53.0\%$, worst case: 100%

$\Rightarrow$ **Input could not be reconstructed:**

- too many unknown bits (e.g. $m = 11$: $(1 - 0.266)202$)
- positions of known bits in bit stream unknown

Simon Hoerder

University of Bristol

References

[1] D.J. Bernstein, T. Lange and C. Peters. *Attacking and defending the McEliece cryptosystem*. In: Proceedings of PQCrypto 2008, LNCS 5299, pages 31–46, 2008. See also: Cryptology ePrint Archive, Report 2008/318, 2008. URL: `http://eprint.iacr.org/2008/318.pdf`

[2] F. Strenzke, E. Tews, H.G. Molter, R. Overbeck and A. Shoufan. *Side Channels in the McEliece PKC*. In Proceedings of PQCrypto 2008, LNCS 5299, pages 216–229, Springer-Verlag Berlin Heidelberg, October 2008.

[3] A. Shoufan, F. Strenzke, H.G. Molter and M. Stöttinger. *A Timing Attack Against Patterson Algorithm in the McEliece PKC*. In Information, Security and Cryptology -– ICISC 2009, LNCS 5984, pages 161–175, Springer-Verlag Berlin Heidelberg, 2010.

[4] F. Strenzke. *A Timing Attack against the secret Permutation in the McEliece PKC*. In Proceedings of PQCrypto 2010, LNCS 6061, pages 95–107, Springer-Verlag Berlin Heidelberg, 2010.

[5] N. Sendrier. *Encoding Information into Constant Weight Words*. In Proceedings of the 2005 IEEE International Symposium on Information Theory, Adelaide, pages 435-438, Springer-Verlag Berlin Heidelberg, September 2005

Do you have questions?

?**?**?

Simon Hoerder

University of Bristol

25/02/2011

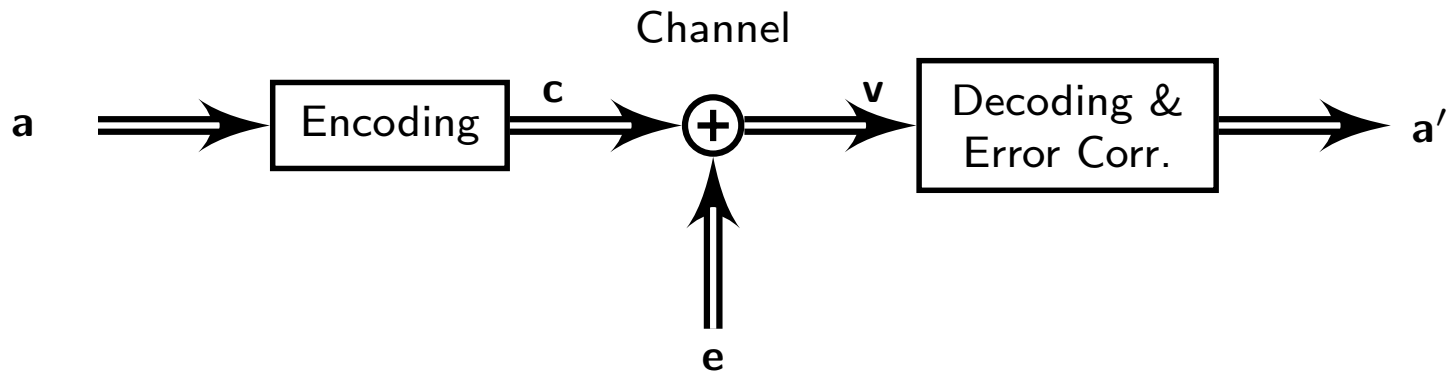
Backup Slides

Backup Slides

Simon Hoerder

University of Bristol

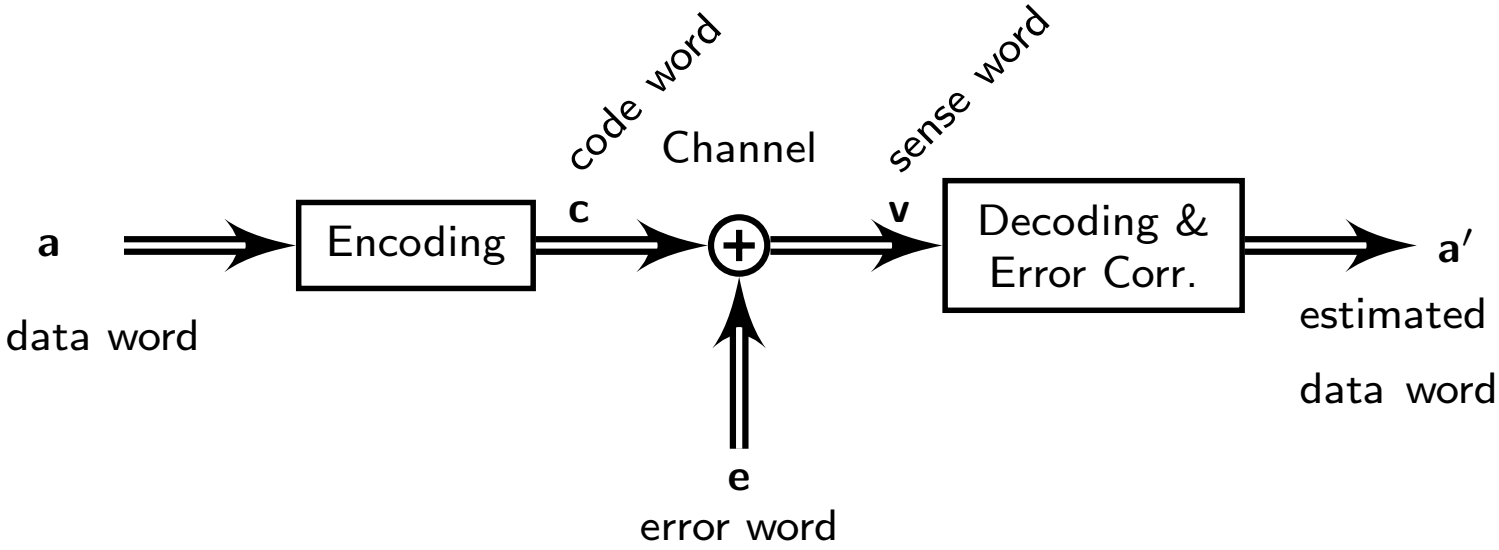
Using Error Correction Codes as PKC



Simon Hoerder

University of Bristol

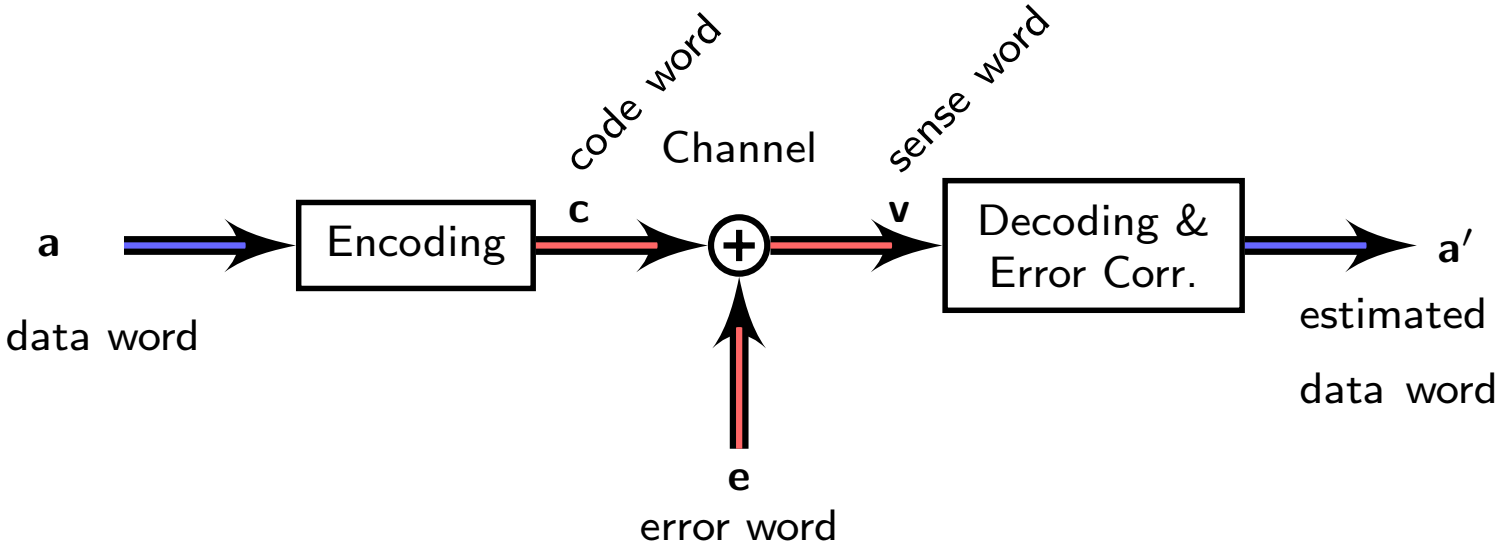
Using Error Correction Codes as PKC



Simon Hoerder

University of Bristol

Using Error Correction Codes as PKC



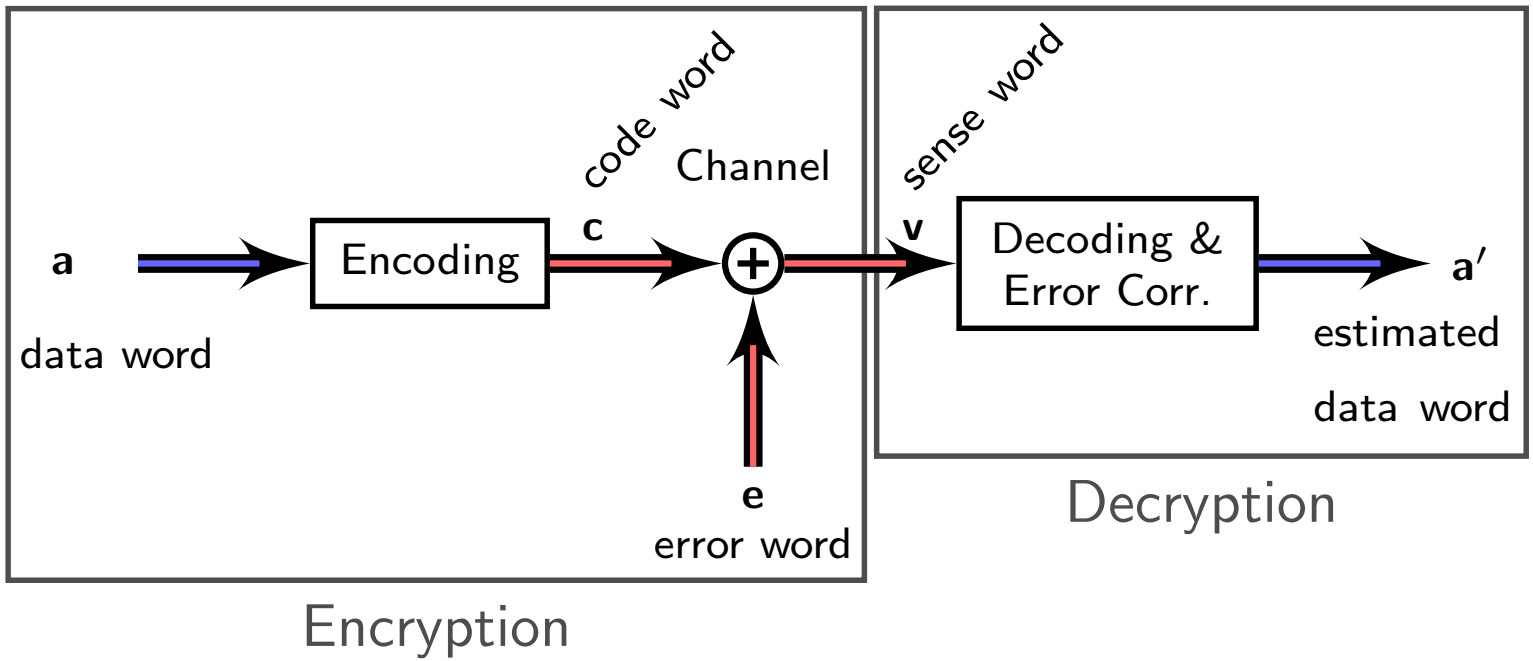
blocksize: k blocksize: n blocksize: k

Simon Hoerder

University of Bristol

Using Error Correction Codes as PKC

Idea: Error correction for general $(n, k, 2t+1)$ codes is NP-hard



blocksize: k blocksize: n blocksize: k

Simon Hoerder

University of Bristol

Decoding & Error Correction for Linear Codes

Classical Goppa Codes (CGCs): subfield codes defined by

Simon Hoerder

University of Bristol

Decoding & Error Correction for Linear Codes

Classical Goppa Codes (CGCs): subfield codes defined by irreducible *"Goppa polynomial"* $G(X)$ with $\deg(G(X)) = t$ and code support $\mathcal{L} \subseteq \mathbb{F}_{2^m}[X]$

Simon Hoerder

University of Bristol

Decoding & Error Correction for Linear Codes

Classical Goppa Codes (CGCs): subfield codes defined by irreducible *"Goppa polynomial"* $G(X)$ with $\deg(G(X)) = t$ and code support $\mathcal{L} \subseteq \mathbb{F}_{2^m}[X]$

can correct up to t errors efficiently.

Simon Hoerder

University of Bristol

Decoding & Error Correction for Linear Codes

Classical Goppa Codes (CGCs): subfield codes defined by irreducible *"Goppa polynomial"* $G(X)$ with $\deg(G(X)) = t$ and code support $\mathcal{L} \subseteq \mathbb{F}_{2^m}[X]$ can correct up to t errors efficiently.

Syndrome Computation with $\mathbf{H} \in \mathbb{F}_{2^m}^{t \times n}$, derived from $G(X)$:

$$\sum_{j=0}^{n-1} \frac{c_j}{X - \gamma_j} \equiv 0 \bmod G(X), \qquad c_j \in \mathbb{F}_2,\ \gamma_j \in \mathcal{L} \subseteq \mathbb{F}_{2^m}$$

Simon Hoerder

University of Bristol

Decoding & Error Correction for Linear Codes

Classical Goppa Codes (CGCs): subfield codes defined by irreducible *"Goppa polynomial"* $G(X)$ with $\deg(G(X)) = t$ and code support $\mathcal{L} \subseteq \mathbb{F}_{2^m}[X]$ can correct up to t errors efficiently.

Syndrome Computation with $\mathbf{H} \in \mathbb{F}_{2^m}^{t \times n}$, derived from $G(X)$:

$$\sum_{j=0}^{n-1} \frac{c_j}{X - \gamma_j} \equiv 0 \bmod G(X), \qquad c_j \in \mathbb{F}_2,\ \gamma_j \in \mathcal{L} \subseteq \mathbb{F}_{2^m}$$

Error Computation find $\gamma_j \in \mathcal{L}$ s.t. $u(\gamma_j) = 0$ for $u(X) \in \mathbb{F}_{2^m}[X]$.

Simon Hoerder

University of Bristol

Decryption Algorithm

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.

OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [Syndrome computation.]

Simon Hoerder

University of Bristol

Decryption Algorithm

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.

OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [Syndrome computation.]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$ [Locator computation.]
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [Break off iff:]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$ [$\deg(\mathrm{x}(X)) < t$]

Simon Hoerder

University of Bristol

Decryption Algorithm

Algorithm 1 McEliece Decryption based on Patterson's Algorithm

INPUT: A binary ciphertext word $\mathbf{v} \in \mathbb{F}_2^n$.

OUTPUT: The closest cleartext $\mathbf{a}' \in \mathbb{F}_2^k$.

1. $s(X) \leftarrow (\mathbf{v}\mathbf{Q}^{-1}\mathbf{H}^\top)^{-1} \bmod G(X)$ [Syndrome computation.]
2. **if** $s(X) = X$ **then** $u(X) \leftarrow s(X)$ [Locator computation.]
3. **else** $s'(X) \leftarrow \sqrt{s(X) + X} \bmod G(X)$
4. $[\mathrm{x}(X), \mathrm{y}(X)] \leftarrow \mathrm{EEA}_{\mathrm{DEC}}(G(X), s'(X), t)$ [Break off iff:]
5. $u(X) \leftarrow \mathrm{x}^2(X) + X\mathrm{y}^2(X)$ [$\deg(\mathrm{x}(X)) < t$]
6. **for** $\gamma_i \in (\mathbf{Q}\mathcal{L})$ [Error computation.]
7. **if** $(u(\gamma_i) = 0)$ **then** $e_i \leftarrow 1$ [Horner Scheme]
8. **else** $e_i \leftarrow 0$
9. **return** $\mathbf{a}' \leftarrow (\mathbf{v} \oplus \mathbf{e})(\mathbf{G}')^{-1}$

Simon Hoerder

University of Bristol