RUHR-UNIVERSITÄT BOCHUM Universität Bremen

RUB

IND-CCA Secure Hybrid Encryption from QC-MDPC Niederreiter

7th International Conference on Post-Quantum Cryptography 2016

Ingo von Maurich[1], Lukas Heberle[1], Tim Güneysu[2]

[1]Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Germany

[2]University of Bremen & DFKI, Germany

2/24/2016

Motivation

- **Demand for secure embedded devices / Internet of Things (IoT)**
  - Cost-sensitive
  - Long life-time/security
  - User interaction/low latency
  - Limited resources/memory
- **Goal**
  - Alternative public-key cryptosystems resistant to quantum computing attacks
  - Evaluate if potentially could replace RSA/ECC
  - Efficient implementations for low-cost embedded devices

➔ **Code-based cryptography**

Overview

Motivation

Background

IND-CCA Hybrid Encryption from Niederreiter

Implementing QC-MDPC Niederreiter KEM/DEM

Results

Conclusion

Niederreiter Encryption Scheme [1986]

Key Generation

Given a t-error correcting linear $[n, k, d]$-code C with parity-check matrix H.
Private key: (S, H, P), where S is a scrambling and P is a permutation matrix
Public key: $H' = S \cdot H \cdot P$

Encryption

Error vector $e \in \mathbb{F}_2^n$, $\mathrm{wt}(e) \leq t$, representing message m.

$$s' \leftarrow H' e^T$$

Decryption

Let Ψ_H be a t-error-correcting decoding algorithm.

$$e \leftarrow P^{-1} \Psi_H (S^{-1} s')$$

Modern Niederreiter Encryption

Key Generation

Given a t-error correcting $[n, k, d]$-code C with parity-check matrix H.

Private key: H

Public key: $H' = [Q \mid I_{n-k}] \leftarrow H$ in systematic form

Encryption

Error vector $e \in \mathbb{F}_2^n$, $\mathrm{wt}(e) \leq t$, representing message m.

$$s' \leftarrow H' e^T$$

Decryption

Let Ψ_H be a t-error-correcting decoding algorithm.

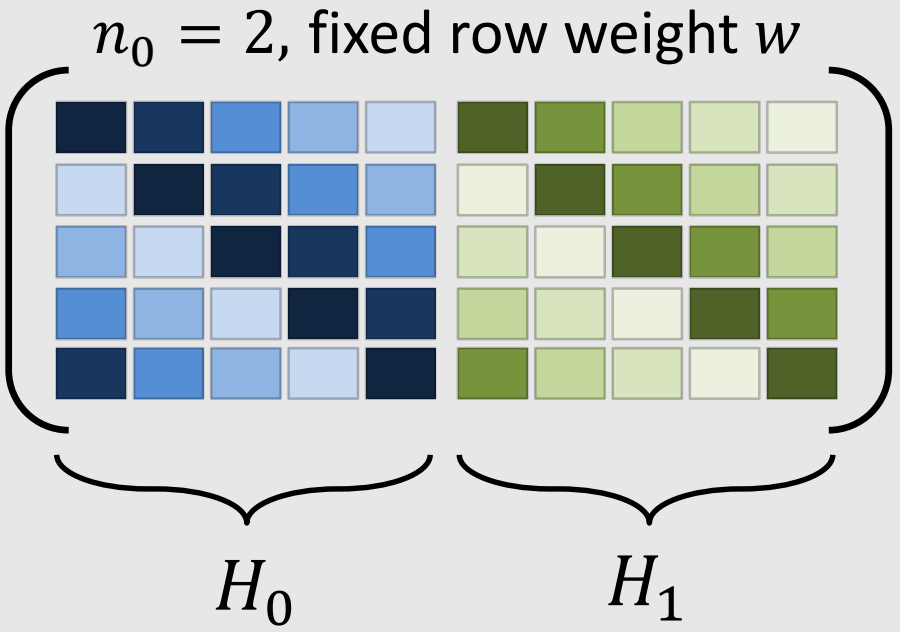
$$e \leftarrow \Psi_H(s')$$

Code-based Cryptography for Embedded Devices

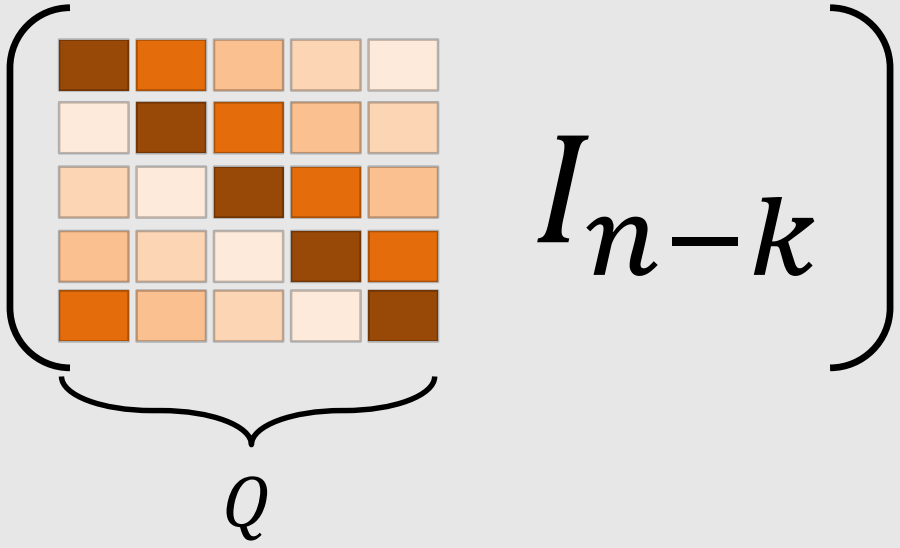
- **Selection of the employed code is critical**
  - Properties of code determine key size, **short keys essential**
  - Structures in codes reduce key size, but can enable attacks
  - Encoding is a fast operation (matrix-vector multiplication)
  - **Efficient decoding** required in terms of time and memory
- **Quasi-cyclic moderate density parity-check codes [MTSB13]**
  - Short keys due to quasi-cyclic structure
  - Efficient decoding in software and hardware

Introducing QC-MDPC Codes

Private parity-check matrix H



Public parity-check matrix H'



QC-MDPC Codes for Niederreiter

Quasi-cyclic moderate density parity-check codes for Niederreiter

- t-error correcting (n, r, w)-QC-MDPC code of length $n = n_0 r, r = n - k$
- Parity-check matrix H consists of n_0 blocks with fixed row weight w

Code/Key Generation

1. Generate n_0 first rows of parity-check matrix blocks H_i
   $h_i \in_R \mathbb{F}_2^r$ of weight $d_v = w/n_0$
2. Obtain remaining rows by $r - 1$ quasi-cyclic shifts of h_i
3. If H_{n_0-1} is not invertible, start over
4. Private key
$$H = [H_0 \mid H_1 \mid \ldots \mid H_{n_0-1}]$$
5. Systematic public key
$$H' = [H_{n_0-1}^{-1} \cdot H] = [H_{n_0-1}^{-1} \cdot H_0 \mid \ldots \mid H_{n_0-1}^{-1} \cdot H_{n_0-2} \mid I_r]$$

(QC-)MDPC Niederreiter [BBMR14]

Encryption

Error vector $e \in F_2^n, wt(e) \leq t$. Compute public syndrome

$$s' \leftarrow H' e^T$$

Decryption

Let Ψ_H be a t-error-correcting (QC-)MDPC decoding algorithm.

$$e \leftarrow \Psi_H(s')$$

Parameters for 80-/128-/256-bit equivalent symmetric security [MTSB13]

Sec. level	n_0	n	r	w	t
80-bit	2	9602	4801	90	84
128-bit	2	19714	9857	142	134
256-bit	2	65542	32771	274	264

Efficient Decoding of (QC-)MDPC Codes

Bit-flipping decoder

Private key $H = [H_0 \mid H_1 \mid \ldots \mid H_{n_0-1}]$, initialize error candidate $e_{cand} = \mathbf{0}$

1. Compute private syndrome $s = H_{n_0-1}s' = H_{n_0-1}H_{n_0-1}^{-1}He^T$
2. Count unsatisfied parity-checks $\#_{upc} =$ shared set bits of columns $H_i[j]$ and s
3. If $\#_{upc} \geq b$, flip corresponding error candidate bit $e_{cand}[\mathrm{i} \cdot \mathrm{r} + \mathrm{j}]$ and update $s = s \oplus H_i[j]$
4. Repeat 2.+3. until $s = 0$ or reaching max. iterations (decoding failure)
5. In case of decoding failure increment thresholds and repeat

How to determine threshold b?

- Precompute b_i for each iteration based on code parameters [Gal62]

Overview

Motivation

Background

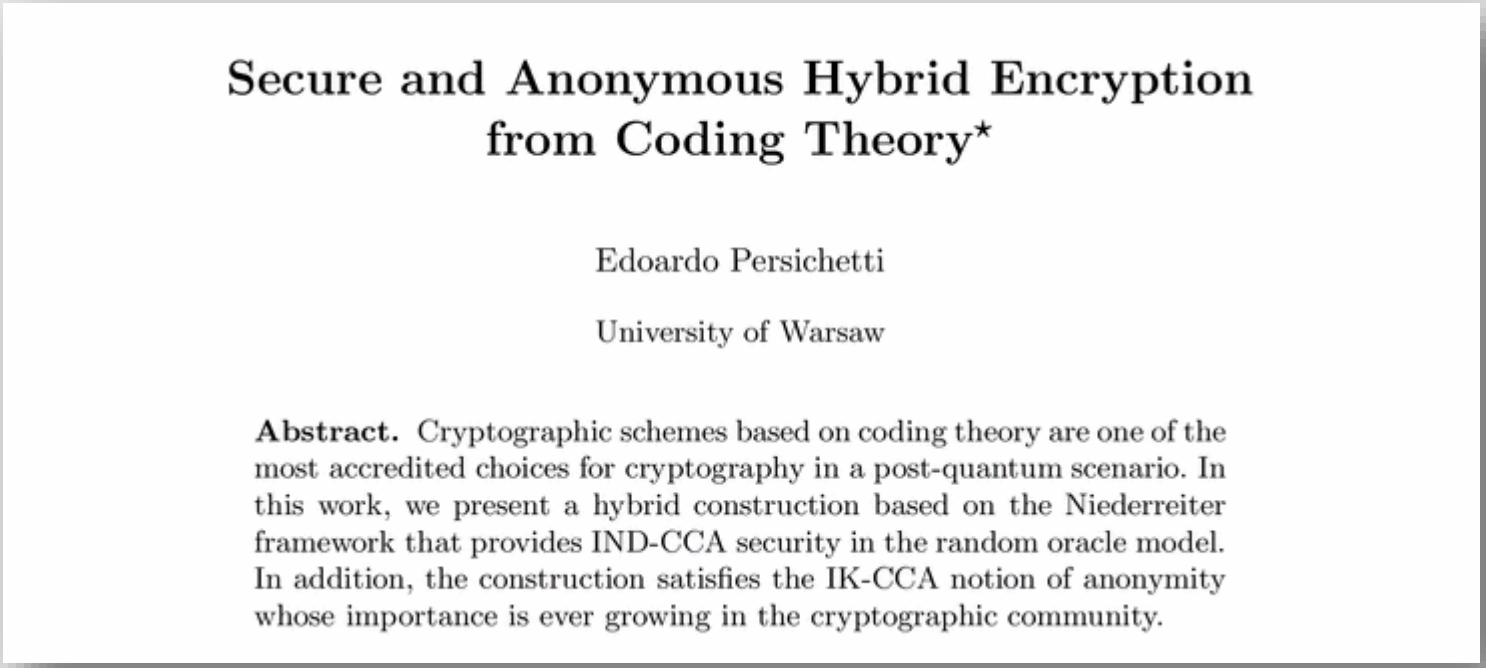
IND-CCA Hybrid Encryption from Niederreiter

Implementing QC-MDPC Niederreiter KEM/DEM

Results

Conclusion

Secure Hybrid Encryption from Coding Theory



Secure and Anonymous Hybrid Encryption from Coding Theory*

Edoardo Persichetti

University of Warsaw

Abstract. Cryptographic schemes based on coding theory are one of the most accredited choices for cryptography in a post-quantum scenario. In this work, we present a hybrid construction based on the Niederreiter framework that provides IND-CCA security in the random oracle model. In addition, the construction satisfies the IK-CCA notion of anonymity whose importance is ever growing in the cryptographic community.

- Proposed by E. Persichetti at PQCrypto‘13
- Hybrid construction based on the Niederreiter framework
- Provides IND-CCA security and IK-CCA anonymity in the ROM
- Plain McEliece/Niederreiter are not IND-CCA

Hybrid Encryption

Key Encapsulation Mechanism (KEM)
A public-key encryption scheme that encrypts a randomly generated symmetric session key under the public key of the intended receiver.

Data Encapsulation Mechanism (DEM)
A symmetric encryption scheme that encrypts the plaintext under the symmetric session key generated by the KEM.

Niederreiter KEM [Per13]

Niederreiter Key Encapsulation π_{NR_KEM}

Let $\mathcal{F}$ be the family of t-error correcting $[n, k]$-linear codes over $\mathbb{F}_q$ and let n, k, q, t be fixed system parameters.

***Gen*$_{NR\_KEM}$**

Pick a code $C \in_R \mathcal{F}$ with public parity-check matrix $H' = (Q \mid I_{n-k})$.
Output H' and private code description Δ.

***Enc*$_{NR\_KEM}$**

Generate $e \in_R \mathbb{F}_q^n, wt(e) = t$. Compute public syndrome $s' = H'e^T$.
Derive symmetric session key k of length l_k as
$k = (k_1, k_2) = KDF(e, l_k)$. Output (k, s').

***Dec*$_{NR\_KEM}$**

Decode $e \leftarrow \Psi_\Delta(s')$ and derive $k = (k_1, k_2) = KDF(e, l_k)$.
If decoding fails set $k = KDF(s', l_k)$.

Standard DEM [Per13]

Standard Data Encapsulation π_{DEM}

Let $Enc_{k_1}^{SE}(\cdot), Dec_{k_1}^{SE}(\cdot)$ be symmetric en-/decryption under k_1 and let $Ev_{k_2}(\cdot)$ be the evaluation of a keyed MAC with fixed length output τ.

***Enc*$_{DEM}$**

Given $k = (k_1, k_2)$, encrypt plaintext m to $T = Enc_{k_1}^{SE}(m)$.

Compute $\tau = Ev_{k_2}(T)$, output $c^* = (T, \tau)$.

***Dec*$_{DEM}$**

Given k, T, τ. Verify correctness of $Ev_{k_2}(T) = \tau$.

If the MAC is correct, decrypt $Dec_{k_1}^{SE}(T) \rightarrow m$, else return $\perp$.

QC-MDPC Niederreiter Hybrid Encryption

The QC-MDPC Niederreiter Hybrid Encryption Scheme

π_{NR_KEM}
→ QC-MDPC Niederreiter

π_{DEM}

Message en-/decryption:	AES-128 (AES-256) in CBC-mode
Message authentication:	AES-128 (AES-256) in CMAC-mode
Key derivation:	SHA-256 (SHA-512)

- AES-CBC with random IVs ⇒ IND-CPA
- AES-CMAC ⇒ integrity of ciphertexts (INT-CTXT)
- INT-CTXT and IND-CPA ⇒ IND-CCA [Bellare, Namprempre ’00]

Overview

Motivation

Background

IND-CCA Hybrid Encryption from Niederreiter

Implementing QC-MDPC Niederreiter KEM/DEM

Results

Conclusion

Implementation Platform

32-bit microcontroller ARM Cortex-M4

- STM32F417
- 1 Mbyte flash
- 192 kbyte SRAM
- 168 MHz clock frequency
- Crypto co-processors: AES, 3DES, SHA-1, TRNG
- Cost USD ~10

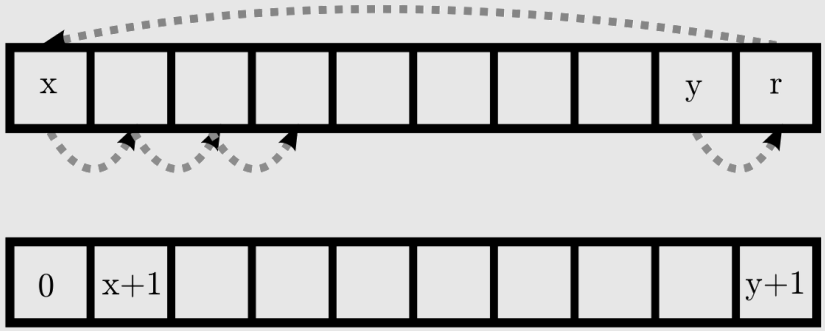
Polynomial Representations

`poly_t`

- Naïve approach, simply stores each bit after another

`sparse_t`

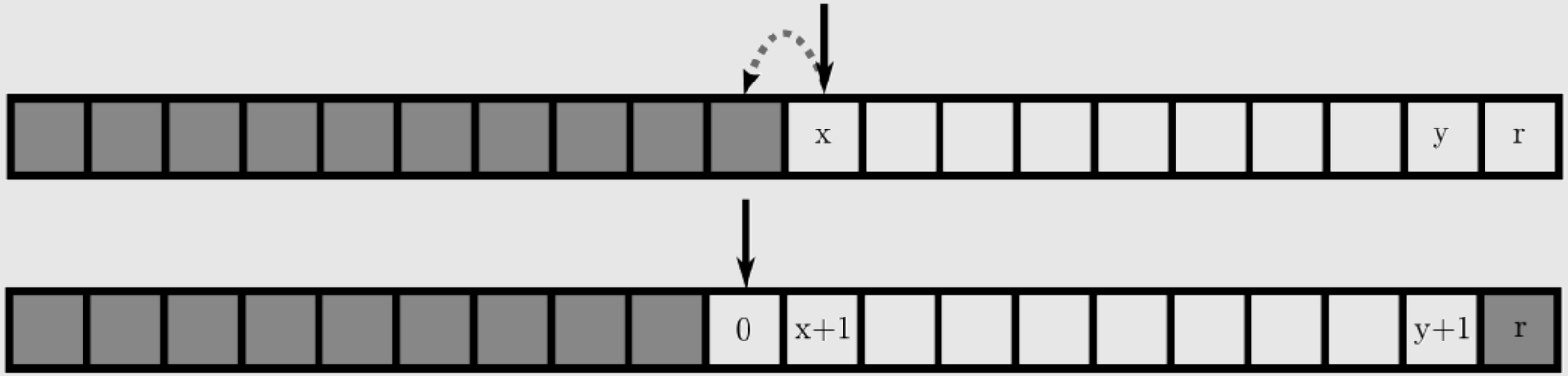
- Stores positions of set bits in ascending order
- Saves memory if only few bits are set
- Allows fast iteration of set bits
- Drawback: efficiency of counter overflows during rotation



Sparse Polynomial Rotations

`sparse_double_t`

- Similar to *sparse_t*
- Allocates twice the required memory
- Has a pointer to the start of the polynomial



Implementing Key Generation

Generate h_{n_0-1} of weight d_v

- Sample d_v positions from TRNG, rejection sampling since r is prime
- Compute inverse using EEA, if not existent, start over

Generate remaining h_i

- Similar to h_{n_0-1}, skip checking for inverse
- First columns as *sparse_double_t* ($r \gg d_v$, fast rotation)

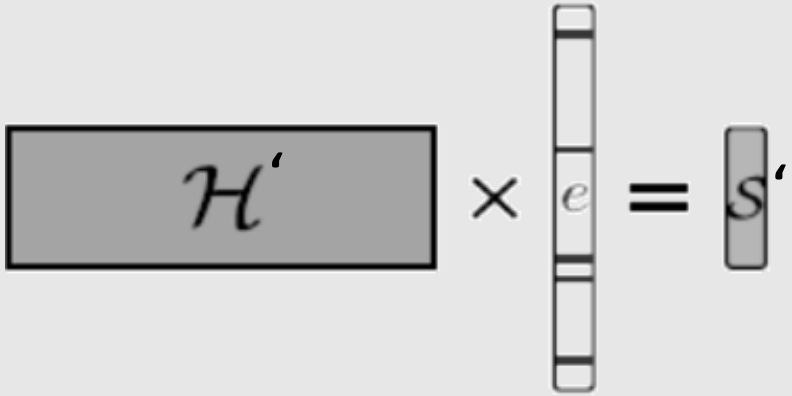
Compute public key H'

- $H' = H_{n_0-1}^{-1} \cdot H = [H_{n_0-1}^{-1} \cdot H_0 \,|\ldots|\, I_r]$
- Since $n_0 = 2$, compute $H_1^{-1} \cdot H_0$ and append I_r
- Store first column as *poly_t*

Implementing Encryption

Implementing QC-MDPC Niederreiter encryption

- Recall $s' = H'e^T$
- Set bits in error e select columns of the public key H'
- e is stored as *sparse_t,* allows fast iteration of set bits
- XOR corresponding H' columns
- Resulting s' is dense and stored as *poly_t*



Implementing Decryption

Computing the private syndrome

- Recall $s = H_{n_0-1} s'$
- Similar to encryption, s' selects columns of H_{n_0-1}
- Fast iteration of columns of H_{n_0-1} due to *sparse_double_t*

Decoding the error vector

- Count $\#_{upc}$ that are set in the s and column $H_i[j]$, invert ciphertext bit at pos $i \cdot r + j$ if $\#_{upc}$ above threshold b

$\rightarrow wt(s \,\&\, H_i[j]) \geq b$?

  - $e_{cand}[i \cdot r + j] = e_{cand}[i \cdot r + j] \oplus 1$
  - Update syndrome to $s = s \oplus H_i[j]$

- Generate next column: increment counters (+handle overflow)

Implementing Hybrid Encryption

Hybrid Encryption

- QC-MDPC Niederreiter encryption
- TRNG generates random $e, wt(e) = t$ and random CBC IV
- Software implementation of SHA-256 derives k_1, k_2 from e
- AES-CBC and AES-CMAC accelerated using AES co-processor

Hybrid Decryption

- QC-MDPC Niederreiter decryption
- SHA-256 derives k_1, k_2 from e, same as in encryption
- AES-CBC and AES-CMAC accelerated using AES co-processor

Overview

Motivation

Background

IND-CCA Hybrid Encryption from Niederreiter

Implementing QC-MDPC Niederreiter KEM/DEM

Results

Conclusion

Implementation Results

Scheme	Cycles/Op 80-bit	Time 80-bit	Cycles/Op 128-bit	Time 128-bit
QC-MDPC NR (enc)	2,623,000	16 ms	13,726,000	82 ms
QC-MDPC NR (dec)	18,416,000	110 ms	80,261,000	478 ms
QC-MDPC NR (keygen)	63,185,000	376 ms	251,289,000	1,496 ms
Hybrid QC-MDPC NR (enc)	2,688,000	16 ms	13,944,000	83 ms
Hybrid QC-MDPC NR (dec)	18,648,000	111 ms	80,304,000	478 ms

- Hybrid scheme requires 25KiB flash (2.4%), 4KiB SRAM (2.0%), QC-MDPC Niederreiter, AES-CBC, AES-CMAC, SHA-256, **keys**

Results Comparison

Scheme	Platform	SRAM [byte]	Flash [byte]	Cycles/Op	Time/Op [ms]
QC-MDPC $NR_{80\text{-bit,enc}}$	STM32F417	2,048	3,064	2,623,432	16
QC-MDPC $NR_{80\text{-bit,dec}}$	STM32F417	2,048	8,621	18,416,012	110
QC-MDPC $NR_{80\text{-bit,keygen}}$	STM32F417	3,136	8,784	63,185,108	376
QC-MDPC $NR_{80\text{-bit,combined}}$	STM32F417	3,136	16,124	-	-
QC-MDPC $NR_{128\text{-bit,enc}}$	STM32F417	2,048	4,272	13,725,688	82
QC-MDPC $NR_{128\text{-bit,dec}}$	STM32F417	2,048	8,962	80,260,696	478
QC-MDPC $NR_{128\text{-bit,keygen}}$	STM32F417	3,136	12,096	251,288,544	1496
QC-MDPC $NR_{128\text{-bit,combined}}$	STM32F417	3,136	20,416	-	-
QC-MDPC $McE_{80\text{-bit,enc}}$	STM32F407	2,700[1]	5,700[1]	7,018,493	42
QC-MDPC $McE_{80\text{-bit,dec}}$	STM32F407	2,700[1]	5,700[1]	42,129,589	251
QC-MDPC $McE_{80\text{-bit,keygen}}$	STM32F407	2,700[1]	5,700[1]	148,576,008	884
QC-MDPC $McE_{80\text{-bit,enc}}$ [HvMG13]	ATxmega256	606	5,500	26,767,463	836
QC-MDPC $McE_{80\text{-bit,dec}}$ [HvMG13]	ATxmega256	198	2,200	86,874,388	2,710
Goppa McE_{enc} [EGHP09]	ATxmega256	512	438,000	14,406,080	450
Goppa McE_{dec} [EGHP09]	ATxmega256	12,000	130,400	19,751,094	617
Goppa McE_{enc} [Hey11]	ATxmega256	3,500	11,000	6,358,400	199
Goppa McE_{dec} [Hey11]	ATxmega256	8,600	156,000	33,536,000	1,100
Srivastava McE_{enc} [CHP12]	ATxmega256	-	-	4,171,734	130
Srivastava McE_{dec} [CHP12]	ATxmega256	-	-	14,497,587	453

Conclusions and Outlook

QC-MDPC Niederreiter for ARM Cortex-M Microcontrollers

- Practical key sizes compared to binary Goppa codes
- IND-CCA secure hybrid encryption scheme with small computational overhead

Open Questions

- Post-quantum security of current QC-MDPC parameters?
- Countermeasures for SPA, DPA & fault-injection attacks
- Constant-time implementation

➔ See ‘QcBits’ talk by Tung Chou in Hot Topics session!

RUHR-UNIVERSITÄT BOCHUM Universität Bremen RUB

IND-CCA Secure Hybrid Encryption from QC-MDPC Niederreiter

7th International Conference on Post-Quantum Cryptography 2016

Ingo von Maurich[1], Lukas Heberle[1], Tim Güneysu[2]

[1]Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Germany

[2]University of Bremen & DFKI, Germany

2/24/2016