DIMACS Workshop on The Mathematics of Post-Quantum Cryptography
Rutgers University
January 12 - 16, 2015

Constructive aspects of code-based cryptography

Marco Baldi

Università Politecnica delle Marche
Ancona, Italy

m.baldi@univpm.it

Code-based cryptography

- Cryptographic primitives based on the decoding problem
- Main challenge: put the adversary in the condition of decoding a random-like code
- Everything started with the McEliece (1978) and Niederreiter (1986) public-key cryptosystems
- A large number of variants originated from them
- Some private-key cryptosystems were also derived
- The extension to digital signatures is still challenging (most concrete proposals: Courtois-Finiasz-Sendrier (CFS) and Kabatianskii-Krouk-Smeets (KKS) schemes)

Main ingredients (McEliece)

- Private key:

$$\{\mathbf{G}, \mathbf{S}, \mathbf{P}\}$$

  - **G**: generator matrix of a *t*-error correcting (n, k) Goppa code
  - **S**: k x k non-singular dense matrix
  - **P**: n x n permutation matrix

- Public key:

$$\mathbf{G'} = \mathbf{S} \cdot \mathbf{G} \cdot \boxed{\mathbf{P}}$$

The private and public codes are permutation equivalent!

Main ingredients (McEliece)

- Encryption map:

$$\mathbf{x} = \mathbf{u} \cdot \mathbf{G'} + \mathbf{e}$$

- Decryption map:

$$\mathbf{x'} = \mathbf{x} \cdot \mathbf{P}^{-1} = \mathbf{u} \cdot \mathbf{S} \cdot \mathbf{G} + \boxed{\mathbf{e} \cdot \mathbf{P}^{-1}}$$

all errors are corrected, so we have:

$\mathbf{u'} = \mathbf{u} \cdot \mathbf{S}$ at the decoder output

$$\mathbf{u} = \mathbf{u'} \cdot \mathbf{S}^{-1}$$

Main ingredients (McEliece)

- Goppa codes are classically used as secret codes
- Any degree-t (irreducible) polynomial generates a different Goppa code (very large families of codes with the same parameters and correction capability)
- Their matrices are non-structured, thus their storage requires kn bits, which are reduced to rk bits with a CCA2 secure conversion
- The public key size grows quadratically with the code length

Niederreiter cryptosystem

- Exploits the same principle, but uses the code parity-check matrix (**H**) in the place of the generator matrix (**G**)

- Secret key: {**H**, **S**} → Public key: $\mathbf{H}' = \mathbf{SH}$

- Message mapped into a weight-*t* error vector (**e**)
- Encryption: $\mathbf{x} = \mathbf{H}'\mathbf{e}^T$
- Decryption: $\mathbf{s} = \mathbf{S}^{-1}\mathbf{x} = \mathbf{He}^T$ → syndrome decoding (**e**)

- In this case there is no permutation (identity), since passing from **G** to **H** suffices to hide the Goppa code (indeed the permutation could be avoided also in McEliece)

Permutation equivalence

- Using permutation equivalent private and public codes works for the original system based on Goppa codes
- Many attempts of using other families of codes (RS, GRS, convolutional, RM, QC, QD, LDPC) have been made, aimed at reducing the public key size
- In most cases, they failed due to permutation equivalence between the private and the public code
- In fact, permutation equivalence was exploited to recover the secret key from the public key

Permutation equivalence (2)

- Can we remove permutation equivalence?
- We need to replace $\mathbf{P}$ with a more general matrix $\mathbf{Q}$
- This way, $\mathbf{G'} = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{Q}$ and the two codes are no longer permutation equivalent
- Encryption is unaffected
- Decryption: $\mathbf{x'} = \mathbf{x} \cdot \mathbf{Q}^{-1} = \mathbf{u} \cdot \mathbf{S} \cdot \mathbf{G} + \boxed{\mathbf{e} \cdot \mathbf{Q}^{-1}}$

Permutation equivalence (3)

- How can we guarantee that $\mathbf{e'} = \mathbf{e} \cdot \mathbf{Q}^{-1}$ is still correctable by the private code?
- We shall guarantee that $\mathbf{e'}$ has a low weight
- This is generally impossible with a randomly designed matrix $\mathbf{Q}$
- But it becomes possible through some special choices of $\mathbf{Q}$

Design of **Q**: first approach

- Design $\mathbf{Q}^{-1}$ as an $n \times n$ sparse matrix, with average row and column weight equal to m:

$$1 < m \ll n$$

- This way, $w(\mathbf{e'}) \leq m \cdot w(\mathbf{e})$ and $w(\mathbf{e'}) \approx m \cdot w(\mathbf{e})$ due to the matrix sparse nature

- $w(\mathbf{e'})$ is always $\leq m \cdot w(\mathbf{e})$ with regular matrices (m integer)

- The same can be achieved with irregular matrices (m fractional), with some trick in the design of **Q**

Design of **Q**: second approach

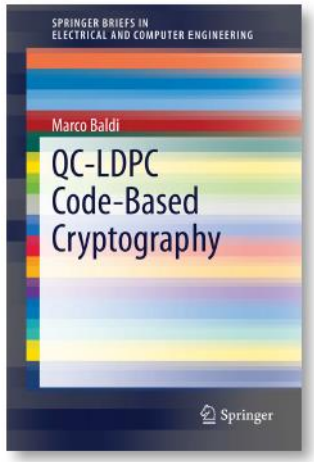
- Design $\mathbf{Q}^{-1}$ as an $n \times n$ sparse matrix **T**, with average row and column weight equal to m, summed to a low rank matrix **R**, such that:

$$\mathbf{e} \cdot \mathbf{Q}^{-1} = \mathbf{e} \cdot \mathbf{T} + \mathbf{e} \cdot \mathbf{R}$$

- Then:
  – Use only intentional error vectors **e** such that $\mathbf{e} \cdot \mathbf{R} = \mathbf{0}$
  …or…
  – Make Bob informed of the value of $\mathbf{e} \cdot \mathbf{R}$

LDPC-code based cryptosystems

(example of use of the first approach)


SPRINGER BRIEFS IN ELECTRICAL AND COMPUTER ENGINEERING
Marco Baldi
QC-LDPC Code-Based Cryptography
Springer

2014, XVI, 120 p. 15 illus.

SpringerBriefs in Electrical and Computer Engineering
(preprint available on ResearchGate)

LDPC codes

- Low-Density Parity-Check (LDPC) codes are capacity-achieving codes under Belief Propagation (BP) decoding
- They allow a random-based design, which results in large families of codes with similar characteristics
- The low density of their matrices could be used to reduce the key size, but this exposes the system to key recovery attacks
- Hence, the public code cannot be an LDPC code, and permutation equivalence to the private code must be avoided

[1] C. Monico, J. Rosenthal, and A. Shokrollahi, "Using low density parity check codes in the McEliece cryptosystem," in *Proc. IEEE ISIT 2000*, Sorrento, Italy, Jun. 2000, p. 215.

[2] M. Baldi, F. Chiaraluce, "Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes," Proc. IEEE ISIT 2007, Nice, France (June 2007) 2591–2595

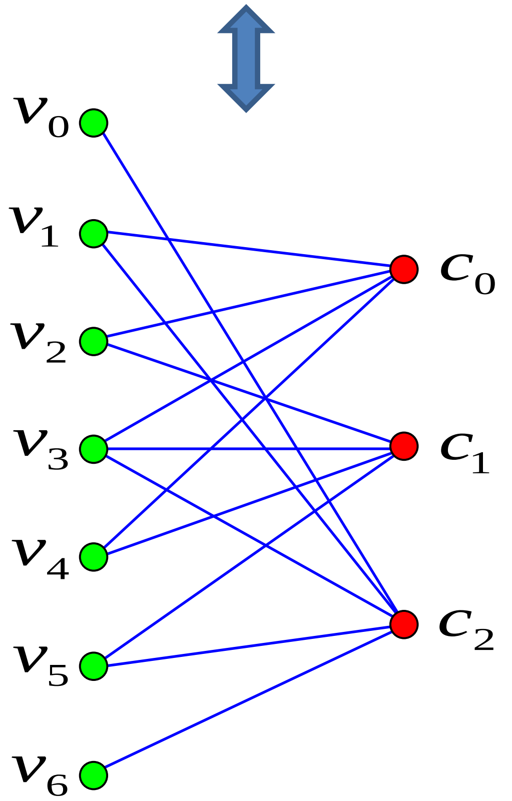
[3] A. Otmani, J.P. Tillich, L. Dallot, "Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes," Proc. SCC 2008, Beijing, China (April 2008)

LDPC codes (2)

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- LDPC codes are linear block codes
  - n: code length
  - k: code dimension
  - $r = n - k$: code redundancy
  - **G**: $k \times n$ generator matrix
  - **H**: $r \times n$ parity-check matrix
  - d_v: average **H** column weight
  - d_c: average **H** row weight

- LDPC codes have parity-check matrices with:
  - Low density of ones ($d_v \ll r$, $d_c \ll n$)
  - No more than one overlapping symbol 1 between any two rows/columns
  - No short cycles in the associated Tanner graph

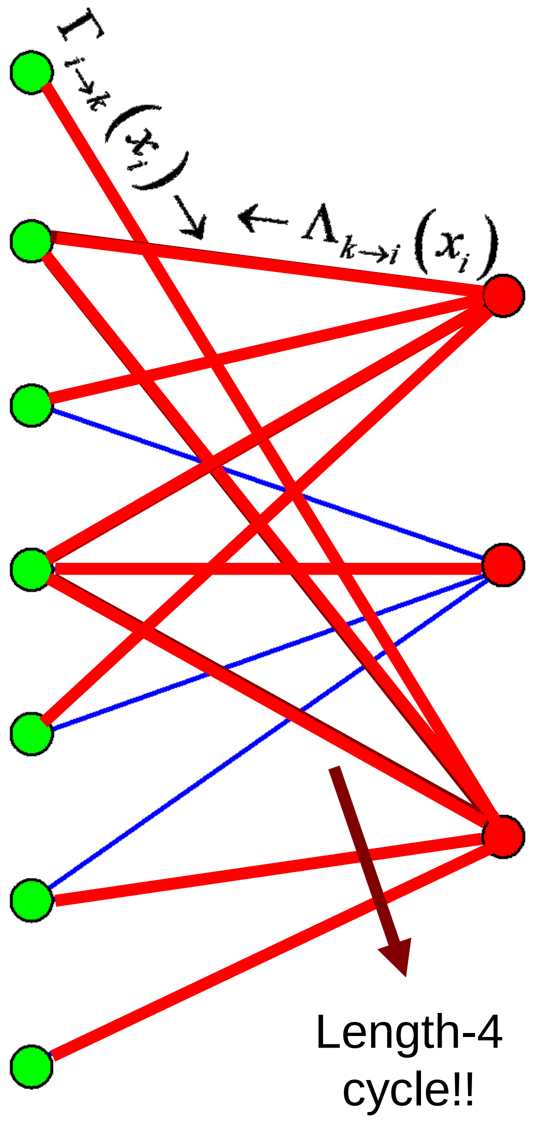


LDPC decoding

- LDPC decoding can be accomplished through the Sum-Product Algorithm (SPA) with Log-Likelihood Ratios (LLR)
- For a random variable U:

$$LLR(U) = \ln\left[\frac{\Pr(U=0)}{\Pr(U=1)}\right]$$

- The initial LLRs are derived from the channel
- They are then updated by exchanging messages on the Tanner graph


$\Gamma_{i \to k}(x_i) \searrow$
$\leftarrow \Lambda_{k \to i}(x_i)$
Length-4 cycle!!

LDPC decoding for the McEliece PKC

- The McEliece encryption map is equivalent to transmission over a special Binary Symmetric Channel with error probability *p* = *t/n*
- LLR of *a priori* probabilities associated with the codeword bit at position *i*:

$$LLR(x_i) = \ln\left[\frac{P\left(x_i = 0 \mid y_i = y\right)}{P\left(x_i = 1 \mid y_i = y\right)}\right]$$

- Applying the Bayes theorem:

$$LLR(x_i \mid y_i = 0) = \ln\left(\frac{1-p}{p}\right) = \ln\left(\frac{n-t}{t}\right)$$

$$LLR(x_i \mid y_i = 1) = \ln\left(\frac{p}{1-p}\right) = \ln\left(\frac{t}{n-t}\right)$$

Bit flipping decoding

- LDPC decoding can also be accomplished through hard-decision iterative algorithms known as bit-flipping (BF)

- During an iteration, every check node sends each neighboring variable node the binary sum of all its neighboring variable nodes, excluding that node

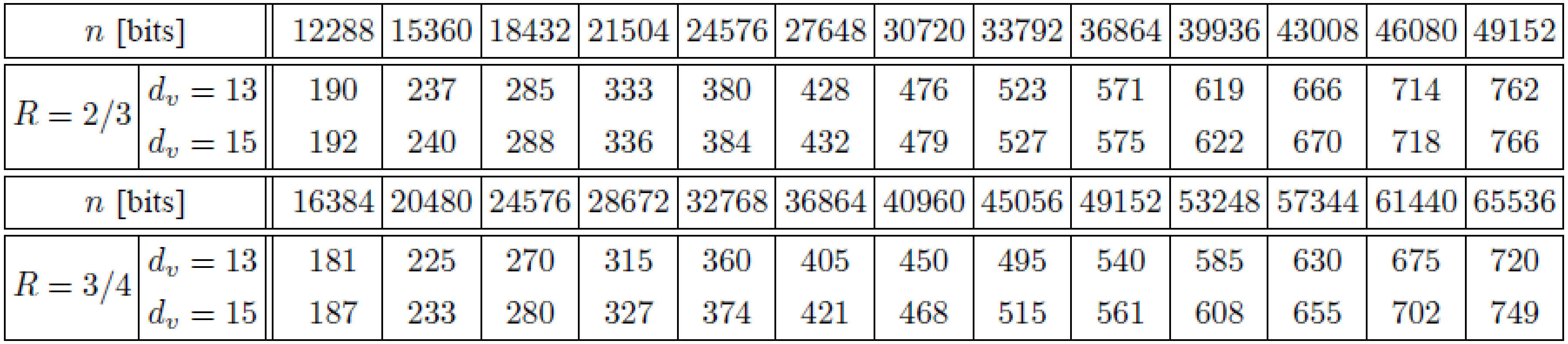
- In order to send a message back to each neighboring check node, a variable node counts the number of unsatisfied parity-check sums from the other check nodes

- If this number overcomes some threshold, the variable node flips its value and sends it back, otherwise, it sends its initial value unchanged

- BF is well suited when soft information from the channel is not available (as in the McEliece cryptosystem)

Decoding threshold

- Differently from algebraic codes, the **decoding radius** of LDPC codes is not easy to estimate
- Their error correction capability is statistical (with a high mean)
- For iterative decoders, the **decoding threshold** of large ensembles of codes can be estimated through density evolution techniques
- The decoding threshold of BF decoders can be found by iterating simple closed-form expressions



n [bits]		12288	15360	18432	21504	24576	27648	30720	33792	36864	39936	43008	46080	49152
$R = 2/3$	$d_v = 13$	190	237	285	333	380	428	476	523	571	619	666	714	762
	$d_v = 15$	192	240	288	336	384	432	479	527	575	622	670	718	766
n [bits]		16384	20480	24576	28672	32768	36864	40960	45056	49152	53248	57344	61440	65536
$R = 3/4$	$d_v = 13$	181	225	270	315	360	405	450	495	540	585	630	675	720
	$d_v = 15$	187	233	280	327	374	421	468	515	561	608	655	702	749

Quasi-Cyclic codes

- A linear block code is a Quasi-Cyclic (QC) code if:
  1. Its dimension and length are both multiple of an integer p ($k = k_0 p$ and $n = n_0 p$)
  2. Every cyclic shift of a codeword by n_0 positions yields another codeword

- The generator and parity-check matrices of a QC code can assume two alternative forms:
  - Circulant of blocks
  - Block of circulants

QC-LDPC codes with rate $(n_0 - 1)/n_0$

- For $r_0 = 1$, we obtain a particular family of codes with length $n = n_0p$, dimension $k = k_0p$ and rate $(n_0 - 1)/n_0$

- **H** has the form of a single row of circulants:

$$\mathbf{H} = \left[\mathbf{H}_0^c \quad \mathbf{H}_1^c \quad \cdots \quad \mathbf{H}_{n_0-1}^c \right] \longleftarrow$$

completely described by its first row

- In order to be non-singular, **H** must have at least one non-singular block (suppose the last)

- In this case, **G** (in systematic form) is easily derived:

$$\mathbf{G} = \left[\begin{array}{cc} & \left[\left(\mathbf{H}_{n_0-1}^c \right)^{-1} \cdot \mathbf{H}_0^c \right]^T \\ \mathbf{I} & \left[\left(\mathbf{H}_{n_0-1}^c \right)^{-1} \cdot \mathbf{H}_1^c \right]^T \\ & \\ & \left[\left(\mathbf{H}_{n_0-1}^c \right)^{-1} \cdot \mathbf{H}_{n_0-2}^c \right]^T \end{array} \right] \longleftarrow$$

completely described by its $(k + 1)$-th column

Random-based design

- A Random Difference Family (RDF) is a set of subsets of a finite group *G* such that every non-zero element of *G* appears no more than once as a difference of two elements in a subset
- An RDF can be used to obtain a QC-LDPC matrix free of length-4 cycles in the form:

$$\mathbf{H} = \left[\mathbf{H}_0^c \quad \mathbf{H}_1^c \quad \cdots \quad \mathbf{H}_{n_0-1}^c \right]$$

- The random-based approach allows to design large families of codes with fixed parameters
- The codes in a family share the characteristics that mostly influence LDPC decoding, thus they have equivalent error correction performance

An example

- RDF over Z_{13}:
  - {1, 3, 8} (differences: 2, 11, 7, 6, 5, 8)
  - {5, 6, 9} (differences: 1, 12, 4, 9, 3, 10)

- Parity-check matrix (n_0 = 2, p = 13):

$$\mathbf{H} = \left[\begin{array}{ccccccccccccc|ccccccccccccc}
0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0&0\\
0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0\\
0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0\\
0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&1\\
0&0&0&0&0&1&0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&1&1&0&0\\
1&0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0\\
0&1&0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1\\
0&0&1&0&0&0&0&0&1&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&0&0&1&0&1&0&1&1&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&1&0&1&1&0&0&1&0&0&0&0&0&0&0\\
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&1&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&1&0&0&1&0&0&0&0&0\\
1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0&0&0
\end{array}\right]$$

Attacks

- In addition to classical attacks against McEliece, some specific attacks exist against QC-LDPC codes
- Dual-code attacks: search for low weight codewords in the dual of the public code in order to recover the secret (and sparse) H
- QC code weakness: exploit the QC nature to facilitate information set decoding (decode one out of many) and low weight codeword searches
- Their work factor depends on the complexity of information set decoding (ISD)

Dual code attacks

- Avoiding permutation equivalence is fundamental to counter these attacks
- We use $\mathbf{Q}^{-1}$ with row and column weight $m \ll n$
- $\mathbf{Q}$ and $\mathbf{Q}^{-1}$ are formed by n_0 x n_0 circulant blocks with size p to preserve the QC nature in the public code
- The public code has parity-check matrix $\mathbf{H'} = \mathbf{H}(\mathbf{Q}^{-1})^T$
- The row weight of $\mathbf{H'}$ is about m times that of $\mathbf{H}$

Security level and Key Size

- Minimum attack WF for *m* = 7:

p [bits]		4096	5120	6144	7168	8192	9216	10240	11264	12288	13312	14336	15360	16384
$n_0 = 3$	$d_v = 13$	2^{54}	2^{63}	2^{73}	2^{84}	2^{94}	2^{105}	2^{116}	2^{125}	2^{135}	2^{146}	2^{157}	2^{161}	2^{161}
	$d_v = 15$	2^{54}	2^{64}	2^{75}	2^{85}	2^{94}	2^{105}	2^{116}	2^{126}	2^{137}	2^{146}	2^{157}	2^{168}	2^{179}
$n_0 = 4$	$d_v = 13$	2^{60}	2^{73}	2^{85}	2^{98}	2^{109}	2^{121}	2^{134}	2^{146}	2^{153}	2^{154}	2^{154}	2^{154}	2^{154}
	$d_v = 15$	2^{62}	2^{75}	2^{88}	2^{100}	2^{113}	2^{127}	2^{138}	2^{152}	2^{165}	2^{176}	2^{176}	2^{176}	2^{176}

- Key size (bytes):

p [bits]	4096	5120	6144	7168	8192	9216	10240	11264	12288	13312	14336	15360	16384
$n_0 = 3$	1024	1280	1536	1792	2048	2304	2560	2816	3072	3328	3584	3840	4096
$n_0 = 4$	1536	1920	2304	2688	3072	3456	3840	4224	4608	4992	5376	5760	6144

[4] M. Baldi, M. Bianchi, F. Chiaraluce, ““Security and complexity of the McEliece cryptosystem based on QC-LDPC codes”, IET Information Security, Vol. 7, No. 3, pp. 212-220, Sep. 2013.

Comparison with Goppa codes

- Comparison considering the Niederreiter version with 80-bit security (CCA2 secure conversion)

Solution	n	k	t	Key size [bytes]	Enc. compl.	Dec. compl.
Goppa based	1632	1269	33	57581	48	7890
QC-LDPC based	24576	18432	38	2304	1206	1790 (BF)

1/25 !

- For the **QC-LDPC** code-based system, the key size **grows linearly** with the code length, due to the **quasi-cyclic** nature of the codes, while with Goppa codes it grows **quadratically**

MDPC code-based variants

- An alternative is to use Moderate-Density Parity-Check (MDPC) codes in the place of LDPC codes

- This means to incorporate the density of $\mathbf{Q}^{-1}$ into the private code, which is no longer an LDPC code

- Then the public code can still be permutation equivalent to the private code

- QC-MDPC code based variants can be designed too

[5] R. Misoczki, J.-P. Tillich, N. Sendrier, P. S. L. M. Barreto, “MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes”, Proc. IEEE ISIT 2013, Istanbul, Turkey, pp 2069–2073.

MDPC code-based variants (2)

- It appears that the short cycles in the Tanner graph are no longer a problem with MDPC codes
- Therefore, their matrices can be designed completely at random
- This has permitted to obtain the first **security reduction** (to the random linear code decoding problem) for these schemes
- On the other hand, decoding MDPC codes is more complex than for LDPC codes (due to denser graphs)

Irregular codes

- Irregular LDPC codes achieve higher error correction capability than regular ones
- This can be exploited to increase the system efficiency by reducing the code length…
- …although the QC structure and the need to avoid enumeration impose some constraints

160-bit security

QC-LDPC code type	n_0	d_v'	t	d_v	n	Key size (bytes)
regular	4	97	79	13	54616	5121
irregular	4	97	79	13	46448	4355

-15%

[6] M. Baldi, M. Bianchi, N. Maturo, F. Chiaraluce, “Improving the efficiency of the LDPC code-based McEliece cryptosystem through irregular codes”, Proc. IEEE ISCC 2013, Split, Croatia, July 2013.

Symmetric variants

- The same principles can also be exploited to build a symmetric cryptosystem inspired to the Barbero-Ytrehus system
- Also in this case, QC-LDPC codes allow to achieve considerable reductions in the key size
- A QC-LDPC matrix is used as a part of the private key
- The sparse nature of the circulant matrices is also exploited by using run-length coding and Huffman coding to achieve a very compact representation of the private key

[7] A. Sobhi Afshar, T. Eghlidos, M. Aref, "Efficient secure channel coding based on quasi-cyclic low-density parity-check codes", IET Communications, Vol. 3, No. 2, pp. 279–292.

GRS-code based cryptosystems

(example of use of the second approach)

Replacing Goppa with GRS codes

- GRS codes are maximum distance separable codes, thus have optimum error correction capability
- This would allow to reduce the public key size
- GRS codes are widespread, and already implemented in many practical systems
- On the other hand, they are more structured than Goppa codes (and wild Goppa codes)

Weakness of GRS codes

- When the public code is permutation equivalent to the private code, the latter can be recovered
- This was first shown by the Sidelnikov-Shestakov attack against the GRS code-based Niederreiter cryptosystem

Avoiding permutation equivalence

- Public parity-check matrix (Niederreiter):

$$\mathbf{H'} = \mathbf{S}^{-1} \cdot \mathbf{H} \cdot \mathbf{Q}^{-1}$$

- $\mathbf{Q}^{-1} = \mathbf{R} + \mathbf{T}$
- **R**: dense $n \times n$ matrix with rank $z \ll n$
- **T**: sparse $n \times n$ matrix with average row and column weight $m \ll n$
- All matrices are over $GF(q)$

[8] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, "Enhanced public key security for the McEliece cryptosystem", Journal of Cryptology, Aug. 2014 (Online First).

Avoiding permutation equivalence (2)

- Example of construction of **R**:
  - take two matrices **a** and **b** defined over GF(q), having size $z \times n$ and rank z
  - Compute $\mathbf{R} = \mathbf{b}^T \cdot \mathbf{a}$

- Encryption:
  - Alice maps the message into an error vector **e** with weight $[t/m]$
  - Alice computes the ciphertext as $\mathbf{x} = \mathbf{H}' \cdot \mathbf{e}^T$

Avoiding permutation equivalence (3)

- Decryption:
  - Bob computes $\mathbf{x'} = \mathbf{S} \cdot \mathbf{x} = \mathbf{H} \cdot \mathbf{Q}^{-1} \cdot \mathbf{e}^T = \mathbf{H} \cdot (\mathbf{b}^T\mathbf{a} + \mathbf{T}) \cdot \mathbf{e}^T = \mathbf{H} \cdot \mathbf{b}^T \cdot \boldsymbol{\gamma} + \mathbf{H} \cdot \mathbf{T} \cdot \mathbf{e}^T$, where $\boldsymbol{\gamma} = \mathbf{a} \cdot \mathbf{e}^T$
  - We suppose that Bob knows $\boldsymbol{\gamma}$, then he computes $\mathbf{x''} = \mathbf{x'} - \mathbf{H} \cdot \mathbf{b}^T \cdot \boldsymbol{\gamma} = \mathbf{H} \cdot \mathbf{T} \cdot \mathbf{e}^T$
  - $\mathbf{e'} = \mathbf{T} \cdot \mathbf{e}^T$ has weight $\leq t$, thus $\mathbf{x''}$ is a correctable syndrome
  - Bob recovers $\mathbf{e'}$ by syndrome decoding through the private code
  - He multiplies the result by $\mathbf{T}^{-1}$ and demaps $\mathbf{e}$ into the secret message

Main issue

- How can Bob be informed of the value of $\boldsymbol{\gamma} = \mathbf{a} \cdot \mathbf{e}^T$?
- Two possibilities:
  - Alice knows **a** (which is made public), computes **γ** and sends it along with the ciphertext (or select only error vectors such that **γ** is known (all-zero)).
  - Alice does not know **a** and Bob has to guess the value of **γ**
- Both them have pros and cons

A History of proposals and attacks

- M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, “A variant of the McEliece cryptosystem with increased public key security”, Proc. WCC 2011, Paris, France, 11-15 Apr. 2011.
- J.-P. Tillich and A. Otmani, “Subcode vulnerability”, private communication, 2011.
- M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, “Enhanced public key security for the McEliece cryptosystem”, arXiv:1108.2462v2
- A. Couvreur, P. Gaborit, V. Gauthier, A. Otmani, J.-P. Tillich, “Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes”, Designs, Codes and Cryptography, Vol. 73, No. 2, pp 641-666, Nov. 2014.
- M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, “Enhanced public key security for the McEliece cryptosystem”, Journal of Cryptology, Aug. 2014 (Online First).
- A. Couvreur, A. Otmani, J.-P. Tillich, V. Gauthier, “A Polynomial-Time Attack on the BBCRS Scheme”, to be presented at PKC 2015.
- M. Baldi, F. Chiaraluce, J. Rosenthal, D. Schipani, “An improved variant of McEliece cryptosystem based on Generalized Reed-Solomon codes”, submitted to MEGA 2015.

Subcode vulnerability

- When **a** is public, an attacker can look at $\mathbf{H}_S = \begin{bmatrix} \mathbf{H}' \\ \mathbf{a} \end{bmatrix}$
- For any codeword **c** in this subcode: $\mathbf{S}^{-1}\ \mathbf{H}\ \mathbf{T}\ \mathbf{c}^T = \mathbf{0}$
- Hence, the effect of the dense matrix **R** is removed
- When **T** is a permutation matrix, the subcode defined by $\mathbf{H}_S$ is permutation-equivalent to a subcode of the secret code
- The dimension of the subcode is $n - \text{rank}\{\mathbf{H}_S\}$

Distinguishing attacks

- When **a** is private, Bob has to guess the value of **γ**
- The number of attempts he needs increases as q^z
- Therefore only very small values of z ($z = 1$) are feasible
- When $z = 1$ and m is small, the system can be attacked by exploiting distinguishers
- These attacks, recently improved, force us to use very large values of m ($m \approx 2$) when $z = 1$

Avoiding attacks

- Publish **a** such that *z* can be increased, but avoid subcode attacks
- This could be achieved by reducing the dimension of the subcode to zero, which occurs for $z \geq k$
- Let us consider $z = k$ (can be extended to $z \geq k$): in this case $\mathbf{H}_S$ is a square invertible matrix
- The attacker could consider the system $\begin{bmatrix} \mathbf{x} \\ \boldsymbol{\gamma} \end{bmatrix} = \mathbf{H}_S \cdot \mathbf{e}^T$ and solve for **e**

Avoiding attacks (2)

- This further attacks is avoided if:
  - we design **b** such that it has rank $z' < z$ and make a basis of the kernel of $\mathbf{b}^T$ public (through a $z' \times z$ matrix **B**)
  - rather than sending $\boldsymbol{\gamma}$ along with the ciphertext, Alice computes and sends $\boldsymbol{\gamma}' = \boldsymbol{\gamma} + \mathbf{v}$, where **v** is a $z \times 1$ vector in the kernel of $\mathbf{b}^T$ (that is, $\mathbf{b}^T \mathbf{v} = \mathbf{0}$)
  - **v** is obtained as a non-trivial random linear combination of the basis vectors

- This way, when Bob computes $\mathbf{b}^T \boldsymbol{\gamma}'$ he still obtains $\mathbf{b}^T \boldsymbol{\gamma}$, but the attack is avoided since $\boldsymbol{\gamma}$ is hidden

ISD WF and Key Size

- Goppa code-based (PK: **H'** over GF(2))

n	4096										
k	3004	2884	2764	2644	2524	2404	2284	2164	2044	1924	
t	91	101	111	121	131	141	151	161	171	181	
WF	180.1	184.4	187.3	188.9	189.3	188.5	186.7	183.9	180.2	175.7	$\log_2$
KS	400.4	426.7	449.4	468.6	484.3	496.5	505.2	510.4	512.0	510.1	KiB

- GRS code-based (PK: {**H'**, **a**, **B**} over GF(512))

n	511										
k	311	307	303	299	295	291	287	283	279	275	
t	100	102	104	106	108	110	112	114	116	118	
WF	180.1	180.2	180.2	180.1	180.0	179.8	179.5	179.2	178.8	178.4	$\log_2$
KS	295.9	292.8	289.6	286.4	283.3	280.1	276.8	273.6	270.3	267.1	KiB

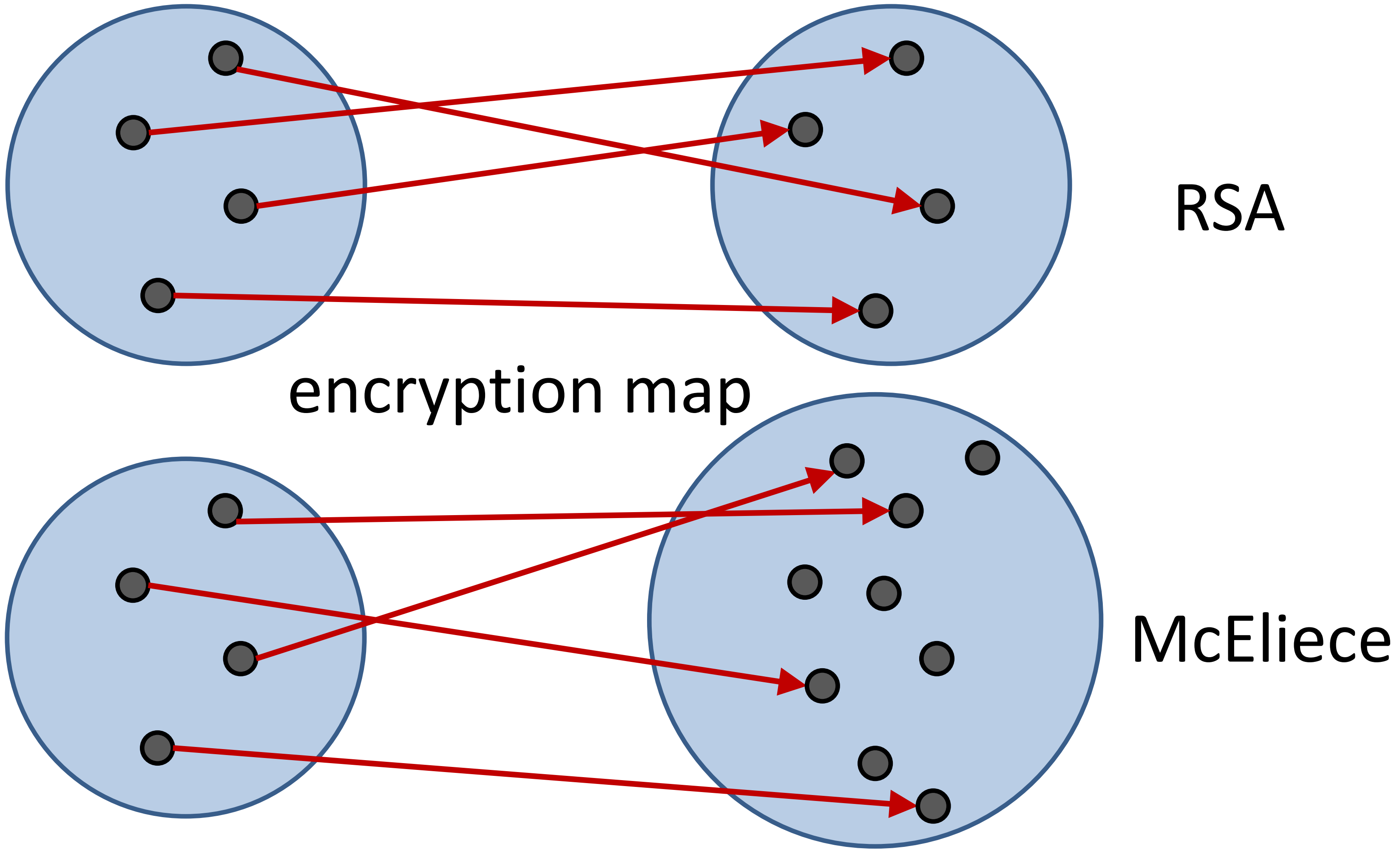
Comparison

- Consider the instances of both systems with highest code rate able to reach WF ≥ 2^{180}
- By using the GRS code-based system, we achieve a public key size reduction in the order of 26% over the classical one
- The gap is even larger by considering lower code rates

Digital signature schemes based on sparse syndromes

(another example of use of the second approach)

From PKC to Digital Signatures



Code-based signature schemes

- Simply inverting decryption with encryption does not work with code-based PKCs
- Some specific solution must be designed
- Two main code-based digital signature schemes:
  - Kabatianskii-Krouk-Smeets (KKS)
  - Courtois-Finiasz-Sendrier (CFS)
- CFS appears to be more robust than KKS

CFS

- Close to the original McEliece Cryptosystem
- Based on Goppa codes

- Public:
  - A hash function $\mathcal{H}(\cdot)$
  - A function $\mathcal{F}(h)$ able to transform any hash digest *h* into a correctable syndrome through the code *C*

- Key generation:
  - The signer chooses a Goppa code able to correct *t* errors, having parity-check matrix **H**
  - He chooses a scrambling matrix **S** and publishes **H’** = **SH**

CFS (2)

- Signing the document D:
  - The signer computes $\mathbf{s} = \mathcal{F}(\mathcal{H}(D))$ and $\mathbf{s'} = \mathbf{S}^{-1}\,\mathbf{s}$
  - He decodes the syndrome $\mathbf{s'}$ through the secret code
  - The error vector $\mathbf{e}$ is the signature

- Verification:
  - The verifier computes $\mathbf{s} = \mathcal{F}(\mathcal{H}(D))$
  - He checks that $\mathbf{H'}\,\mathbf{e}^T = \mathbf{S}\,\mathbf{H}\,\mathbf{e}^T = \mathbf{S}\,\mathbf{S}^{-1}\,\mathbf{s} = \mathbf{s}$

CFS (3)

- The main issue is to find an efficient function $\mathcal{F}(h)$
- In the original CFS there are two solutions:
  - Appending a counter to $h = \mathcal{H}(D)$ until a valid signature is generated
  - Performing complete decoding
- Both these methods require codes with very special parameters:
  - very high rate
  - very small error correction capability

Weaknesses

- Codes with small *t* and high rate could be decoded, with good probability, through the Generalized Birthday Paradox Algorithm (GBA)
- High rate Goppa codes have been discovered to produce public codes which are distinguishable from random codes
- The public key size and decoding complexity can be very large

A CFS variant

- Main differences:
  - Only a subset of sparse syndromes is considered
  - Goppa codes are replaced with low-density generator-matrix (LDGM) codes
- Main advantages:
  - Significant reductions in the public key size are achieved
  - Classical attacks against the CFS scheme are inapplicable
  - Decoding is replaced by a straightforward vector manipulation

[9] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, “Using LDGM Codes and Sparse Syndromes to Achieve Digital Signatures”, Proc. PQCrypto 2013, Limoges, France, June 2013.

Rationale

- If we use a secret code in systematic form and sparse syndromes, we can obtain sparse signatures
- An attacker instead can only forge dense signatures
- Example:
  - secret code: $\mathbf{H} = [\mathbf{X}|\mathbf{I}]$, with $\mathbf{I}$ an $r \times r$ identity matrix
  - $\mathbf{s}$ is an $r \times 1$ sparse syndrome vector
  - the error vector $\mathbf{e} = [\mathbf{0}|\mathbf{s}^T]$ is sparse and verifies $\mathbf{H}\,\mathbf{e}^T = \mathbf{s}$

Issues

- The map **s** $\leftrightarrow$ **e** is trivial (and also linear!)
- The public syndrome should undergo (at least) a secret permutation before obtaining **e**
- Also **e** should be disguised before being made public
- Sparsity is used to distinguish **e** from other (forged) vectors in the same coset, but it should not endanger the system security

Key generation

- Private key: {**Q**, **H**, **S**}, with
  - **H**: $r \times n$ parity-check matrix of the secret code C(n, k)
  - $\mathbf{Q} = \mathbf{R} + \mathbf{T}$
  - $\mathbf{R} = \mathbf{a}^T \mathbf{b}$, having rank $z \ll n$
  - **T** : sparse random matrix with row and column weight m_T , such that **Q** is full rank
  - **S**: sparse non-singular $n \times n$ matrix with average row and column weight $m_S \ll n$

- Public key: $\mathbf{H}' = \mathbf{Q}^{-1} \mathbf{H} \mathbf{S}^{-1}$

Signature generation

- Given the document M
- The signer computes $h = \mathcal{H}(M)$
- The signer finds $\mathbf{s} = \mathcal{F}(h)$, with weight w, such that $\mathbf{b}\ \mathbf{s} = \mathbf{0}$ (this requires 2^z attempts, on average)
- The signer computes the private syndrome $\mathbf{s}' = \mathbf{Q}\ \mathbf{s}$, with weight $\leq m_T w$
- The signer computes the private error vector $\mathbf{e} = [\mathbf{0} | \mathbf{s}'^T]$
- The signer selects a random codeword $\mathbf{c} \in C$ with small weight w_c
- The signer computes the public signature of M as

$$\mathbf{e}' = (\mathbf{e} + \mathbf{c})\ \mathbf{S}^T$$

Signature generation issues

- Without any random codeword **c**, the signing map becomes linear, and signatures can be easily forged
- With **c** having weight $w_c \ll n$, the map becomes affine, and summing two signatures does not result in a valid signature
- The signature should not change each time a document is signed, to avoid attacks exploiting many signatures of the same document
- It suffices to choose **c** as a deterministic function of M

Signature verification

- The verifier receives the message M, its signature $\mathbf{e'}$ and the parameters to use in $\mathcal{F}$
- He checks that the weight of $\mathbf{e'}$ is $\leq (m_T w + w_c) m_S$, otherwise the signature is discarded
- He computes $\mathbf{s^*} = \mathcal{F}(\mathcal{H}(M))$ and checks that it has weight w, otherwise the signature is discarded
- He computes $\mathbf{H'}\,\mathbf{e'}^T = \mathbf{Q}^{-1}\,\mathbf{H}\,\mathbf{S}^{-1}\,\mathbf{S}\,(\mathbf{e}^T + \mathbf{c}^T) = \mathbf{Q}^{-1}\,\mathbf{H}\,(\mathbf{e}^T + \mathbf{c}^T) = \mathbf{Q}^{-1}\,\mathbf{H}\,\mathbf{e}^T = \mathbf{Q}^{-1}\,\mathbf{s'} = \mathbf{s}$
- If $\mathbf{s} = \mathbf{s^*}$, the signature is accepted, otherwise it is discarded

LDGM codes

- LDGM codes are codes with a low density generator matrix **G**
- The row weight of **G** is $w_g \ll n$
- They are useful in this cryptosystem because:
  - Large random-based families of codes can be designed
  - Finding low weight codewords is very easy
  - Structured codes (e.g. QC) can be designed

Attacks

- The signature **e’** is an error vector corresponding to the public syndrome **s** through the public code parity-check matrix **H’**
- If **e’** has a low weight it is difficult to find, otherwise signatures could be forged
- If **e’** has a too low weight the supports of **e** and **c** could be almost disjoint, and the link between the support of **s** and that of **e’** could be discovered
- Hence, the density of **e’** must be:
  - sufficiently low to avoid forgeries
  - sufficiently high to avoid support decompositions

Attacks (2)

- If the matrix **S** is (sparse and) regular, statistical arguments could be used to analyze large number of intercepted signatures (thanks to *J. P. Tillich* for pointing this out)
- This way, an attacker could discover which columns of **S** have a symbol 1 in the same row
- By iterating the procedure, the structure of the matrix **S** could be recovered (except for a permutation)
- This can be avoided by using an irregular matrix **S** with the same average weight

[10] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, "Proposal and Cryptanalysis of a Digital Signature Scheme Based on Sparse Syndromes", in preparation.

Examples

SL (bits)	n	k	p	w	w_g	w_c	z	m_T	m_S	A_{w_c}	N_s	S_k (KiB)
80	9800	4900	50	18	20	160	2	1	9	$2^{82.76}$	$2^{166.10}$	117
120	24960	10000	80	23	25	325	2	1	14	$2^{140.19}$	$2^{242.51}$	570
160	46000	16000	100	29	31	465	2	1	20	$2^{169.23}$	$2^{326.49}$	1685

- For 80-bit security, the original CFS system needs a Goppa code with $n = 2^{21}$ and $r = 2^{10}$, which gives a key size of 52.5 MiB

- By using the parallel CFS, the same security level is obtained with key sizes between 1.25 MiB and 20 MiB

- The proposed system requires a public key of only 117 KiB to achieve 80-bit security (by using QC-LDGM codes)

Comments

- Permutation equivalence between private and public codes can be avoided
- This opens the way to the use of families of codes other than Goppa codes
- Both public-key encryption and digital signature schemes can take advantage of this
- This results in strong reductions in the size of the public keys