CODE-BASED
CRYPTOSYSTEMS
WITH SHORT KEYS

Marco Baldi
Università Politecnica delle Marche
Ancona, Italy
goo.gl/qTJWIk

m.baldi@univpm.it
Asymmetric cryptography primitives

Most used

- ECC
- RSA

Post-quantum

- Lattice-based (NTRU, LWE)
- Code-based (McEliece, Niederreiter)
Do we need Post-quantum crypto?

- On a quantum computer, Shor’s algorithm breaks RSA, ECC, and similar systems in polynomial time.
- October 2011:
  - University of Southern California, Lockheed Martin and D-Wave Systems develop D-Wave One.
- August 2012:
  - Harvard Researchers Use D-Wave quantum computer to fold proteins.
- May 2013:
  - NASA and Google jointly order a 512-qubit D-Wave Two.
Do we need Post-quantum crypto?

- According to Edward Snowden:
  - The National Security Agency has a $79.7 million research program (Penetrating Hard Targets) to build a “cryptologically useful quantum computer”

- RSA and ECC could become practically insecure in about 10-15 years or less
The (public) key size issue

- One of the main advantages of RSA is the short length of the public keys
  - RSA-2048 (recommended): $2 \times 2048$ bits = 512 bytes

- In NTRU, the key is a polynomial with maximum degree $N - 1$ over a polynomial ring, hence a vector of size $N$ over $\mathbb{Z}_q$

- The key size is $N \cdot \lceil \log q \rceil$ bits
  - $N=1171$ and $q=2048$ (recommended): 1611 bytes
Code-based cryptography

- Cryptographic primitives based on the decoding problem (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)
McEliece cryptosystem

- **Private key:**
  \[ \{G, S, P\} \]
  - \( G \): generator matrix of a \( t \)-error correcting \((n, k)\) Goppa code
  - \( S \): \( k \times k \) non-singular dense matrix
  - \( P \): \( n \times n \) permutation matrix

- **Public key:**
  \[ G' = S \cdot G \cdot P \]

The private and public codes are permutation equivalent!
McEliece cryptosystem

- Encryption map:
  \[ x = u \cdot G' + e \]

- Decryption map:
  \[ x' = x \cdot P^{-1} = u \cdot S \cdot G + e \cdot P^{-1} \]

All errors are corrected, so we have:

\[ u' = u \cdot S \] at the decoder output
\[ u = u' \cdot S^{-1} \]
McEliece cryptosystem

- 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: $\{H, S\} \rightarrow$ Public key: $H' = SH$

- Message mapped into a weight-$t$ error vector ($e$)

- Encryption: $x = H'e^T$

- Decryption: $s = S^{-1}x = He^T \rightarrow$ 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)
Public key size

- Goppa code-based Niederreiter system
  - $n = 1632$
  - $k = 1269$
  - $t = 33$

- 80-bit security

- Key size = 57581 bytes
Permutation equivalence

- 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.

- 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 $P$ with a more general matrix $Q$

- This way, $G' = S \cdot G \cdot Q$ and the two codes are no longer permutation equivalent

- Encryption is unaffected

- Decryption: $x' = x \cdot Q^{-1} = u \cdot S \cdot G + e \cdot Q^{-1}$
How can we guarantee that $e' = e \cdot Q^{-1}$ is still correctable by the private code?

We shall guarantee that $e'$ has a low weight.

This is generally impossible with a randomly designed matrix $Q$.

But it becomes possible with a careful design of $Q$ (and $Q^{-1}$).
Design of \( Q \): first approach

- Design \( Q^{-1} \) as an \( n \times n \) sparse matrix, with average row and column weight equal to \( m \):
  
  \[
  1 < m \ll n
  \]

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

- \( w(e') \) is always \( \leq m \cdot w(e) \) 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 $Q^{-1}$ 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:

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

- Then:
  - Use only intentional error vectors $e$ such that $e \cdot R = 0$
    ...or...
  - Make Bob informed of the value of $e \cdot R$
LDPC-CODE BASED CRYPTOSYSTEMS

(example of use of the first approach)

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

2014, XVI, 120 p. 15 illus.
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


LDPC codes (2)

- 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

\[
H = \begin{bmatrix}
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}
\]
Bit flipping decoding

- Hard-decision decoding of LDPC codes is known as bit-flipping (BF) decoding.

- 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.

<table>
<thead>
<tr>
<th>$n$ [bits]</th>
<th>12288</th>
<th>15360</th>
<th>18432</th>
<th>21504</th>
<th>24576</th>
<th>27648</th>
<th>30720</th>
<th>33792</th>
<th>36864</th>
<th>39936</th>
<th>43008</th>
<th>46080</th>
<th>49152</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 2/3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 13$</td>
<td>190</td>
<td>237</td>
<td>285</td>
<td>333</td>
<td>380</td>
<td>428</td>
<td>476</td>
<td>523</td>
<td>571</td>
<td>619</td>
<td>666</td>
<td>714</td>
<td>762</td>
</tr>
<tr>
<td>$d_v = 15$</td>
<td>192</td>
<td>240</td>
<td>288</td>
<td>336</td>
<td>384</td>
<td>432</td>
<td>479</td>
<td>527</td>
<td>575</td>
<td>622</td>
<td>670</td>
<td>718</td>
<td>766</td>
</tr>
<tr>
<td>$R = 3/4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 13$</td>
<td>181</td>
<td>225</td>
<td>270</td>
<td>315</td>
<td>360</td>
<td>405</td>
<td>450</td>
<td>495</td>
<td>540</td>
<td>585</td>
<td>630</td>
<td>675</td>
<td>720</td>
</tr>
<tr>
<td>$d_v = 15$</td>
<td>187</td>
<td>233</td>
<td>280</td>
<td>327</td>
<td>374</td>
<td>421</td>
<td>468</td>
<td>515</td>
<td>561</td>
<td>608</td>
<td>655</td>
<td>702</td>
<td>749</td>
</tr>
</tbody>
</table>
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 \( p \) \((k = k_0p \text{ and } n = n_0p)\)
  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
Rate \((n_0 - 1)/n_0\) random QC-LDPC codes

- A **Random Difference Family** (RDF) is a list 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:

\[
H = \begin{bmatrix}
H_0^c & H_1^c & \cdots & H_{n_0-1}^c
\end{bmatrix}
\]

- The codes in a family share the characteristics that mostly influence LDPC decoding, thus they have equivalent error correction performance.

- Each of them is represented by single a row of \(H\) (**short keys**).
An example

- RDF over $\mathbb{Z}_{13}$:
  - $\{1, 3, 8\}$ (differences: $2, 11, 7, 6, 5, 8$)
  - $\{5, 6, 9\}$ (differences: $1, 12, 4, 9, 3, 10$)

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

\[
H = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
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 $Q^{-1}$ with row and column weight $m \ll n$

- $Q$ and $Q^{-1}$ are formed by $n_0 \times n_0$ circulant blocks with size $p$ to preserve the QC nature in the public code

- The public code has parity-check matrix $H' = H(Q^{-1})^T$

- The row weight of $H'$ is about $m$ times that of $H$
## Security level and Key Size

### Minimum attack WF for $m = 7$:

<table>
<thead>
<tr>
<th>$p$ [bits]</th>
<th>4096</th>
<th>5120</th>
<th>6144</th>
<th>7168</th>
<th>8192</th>
<th>9216</th>
<th>10240</th>
<th>11264</th>
<th>12288</th>
<th>13312</th>
<th>14336</th>
<th>15360</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_0 = 3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 13$</td>
<td>$2^{254}$</td>
<td>$2^{263}$</td>
<td>$2^{273}$</td>
<td>$2^{284}$</td>
<td>$2^{294}$</td>
<td>$2^{305}$</td>
<td>$2^{316}$</td>
<td>$2^{325}$</td>
<td>$2^{336}$</td>
<td>$2^{347}$</td>
<td>$2^{358}$</td>
<td>$2^{369}$</td>
<td>$2^{380}$</td>
</tr>
<tr>
<td>$d_v = 15$</td>
<td>$2^{254}$</td>
<td>$2^{264}$</td>
<td>$2^{275}$</td>
<td>$2^{285}$</td>
<td>$2^{295}$</td>
<td>$2^{306}$</td>
<td>$2^{317}$</td>
<td>$2^{328}$</td>
<td>$2^{339}$</td>
<td>$2^{350}$</td>
<td>$2^{361}$</td>
<td>$2^{372}$</td>
<td>$2^{383}$</td>
</tr>
</tbody>
</table>

| $n_0 = 4$  |      |      |      |      |      |      |       |       |       |       |       |       |       |
| $d_v = 13$ | $2^{260}$ | $2^{273}$ | $2^{285}$ | $2^{298}$ | $2^{310}$ | $2^{322}$ | $2^{334}$ | $2^{346}$ | $2^{358}$ | $2^{370}$ | $2^{382}$ | $2^{394}$ | $2^{406}$ |
| $d_v = 15$ | $2^{262}$ | $2^{275}$ | $2^{288}$ | $2^{300}$ | $2^{312}$ | $2^{324}$ | $2^{336}$ | $2^{348}$ | $2^{360}$ | $2^{372}$ | $2^{384}$ | $2^{396}$ | $2^{408}$ |

### Key size (bytes):

<table>
<thead>
<tr>
<th>$p$ [bits]</th>
<th>4096</th>
<th>5120</th>
<th>6144</th>
<th>7168</th>
<th>8192</th>
<th>9216</th>
<th>10240</th>
<th>11264</th>
<th>12288</th>
<th>13312</th>
<th>14336</th>
<th>15360</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_0 = 3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$1024$</td>
<td>$1280$</td>
<td>$1536$</td>
<td>$1792$</td>
<td>$2048$</td>
<td>$2304$</td>
<td>$2560$</td>
<td>$2816$</td>
<td>$3072$</td>
<td>$3328$</td>
<td>$3584$</td>
<td>$3840$</td>
<td>$4096$</td>
</tr>
</tbody>
</table>

| $n_0 = 4$  |      |      |      |      |      |      |       |       |       |       |       |       |       |
|            | $1536$ | $1920$ | $2304$ | $2688$ | $3072$ | $3456$ | $3840$ | $4224$ | $4608$ | $4992$ | $5376$ | $5760$ | $6144$ |

---


Marco Baldi - Code-based cryptosystems with short keys  October 7, 2015
Comparison with Goppa codes

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

<table>
<thead>
<tr>
<th>Solution</th>
<th>n</th>
<th>k</th>
<th>t</th>
<th>Key size [bytes]</th>
<th>Enc. compl.</th>
<th>Dec. compl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goppa based</td>
<td>1632</td>
<td>1269</td>
<td>33</td>
<td>57581</td>
<td>48</td>
<td>7890</td>
</tr>
<tr>
<td>QC-LDPC based</td>
<td>24576</td>
<td>18432</td>
<td>38</td>
<td>2304</td>
<td>1206</td>
<td>1790 (BF)</td>
</tr>
</tbody>
</table>

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 $Q^{-1}$ 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

### MDPC-LDPC comparison

<table>
<thead>
<tr>
<th></th>
<th>QC-MDPC</th>
<th>QC-LDPC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security against known attacks</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Key size</strong></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Complexity</strong></td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Security reduction(^1)</strong></td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Security decrease with even-sized circulants</strong></td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

\(^1\)to the random linear code decoding problem
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

<table>
<thead>
<tr>
<th>QC-LDPC code type</th>
<th>$n_0$</th>
<th>$d'_v$</th>
<th>$t$</th>
<th>$d_v$</th>
<th>$n$</th>
<th>Key size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>regular</td>
<td>4</td>
<td>97</td>
<td>79</td>
<td>13</td>
<td>54616</td>
<td>5121</td>
</tr>
<tr>
<td>irregular</td>
<td>4</td>
<td>97</td>
<td>79</td>
<td>13</td>
<td>46448</td>
<td>4355</td>
</tr>
</tbody>
</table>

160-bit security

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):
  \[ H' = S^{-1} \cdot H \cdot Q^{-1} \]

- \( Q^{-1} = R + T \)
- \( R \): 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 \( GF(q) \)

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

Encryption:
- Alice maps the message into an error vector $e$ with weight $[t/m]$
- Alice computes the ciphertext as $x = H' \cdot e^T$
Avoiding permutation equivalence (3)

Decryption:

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

- How can Bob be informed of the value of $\gamma = a \cdot e^T$?

- Two possibilities:
  - Alice knows $a$ (which is made public), computes $\gamma$ and sends it along with the ciphertext (or select only error vectors such that $\gamma$ is known (all-zero)).
  - Alice does not know $a$ and Bob has to guess the value of $\gamma$

- Both them have pros and cons
A History of proposals and attacks

- ...
Subcode vulnerability

- When $a$ is public, an attacker can look at $H_S = \begin{bmatrix} H' \\ a \end{bmatrix}$.

- For any codeword $c$ in this subcode: $S^{-1} H T c^T = 0$.

- Hence, the effect of the dense matrix $R$ is removed.

- When $T$ is a permutation matrix, the subcode defined by $H_S$ is permutation-equivalent to a subcode of the secret code.

- The dimension of the subcode is $n - \text{rank}\{H_S\}$.
Distinguishing attacks

- When \( a \) is private, Bob has to guess the value of \( y \)

- 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 \( H_S \) is a square invertible matrix

- The attacker could consider the system
  \[
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix} = H_S \cdot e^T
  \]
This further attacks is avoided if:

- we introduce another secret matrix $X$ and change the definition of $R$ into $R = b^T X a$
- we design $b$ such that it has rank $z' < z$ and make a basis of the kernel of $b^T$ public (through a $z' \times z$ matrix $B$)
- rather than sending $Y$ along with the ciphertext, Alice computes and sends $Y' = Y + v$, where $v$ is a $z \times 1$ vector in the kernel of $b^T$ (that is, $b^T v = 0$)
- $v$ is obtained as a non-trivial random linear combination of the basis vectors

This way, when Bob computes $b^T Y'$ he still obtains $b^T Y$, but the attack is avoided since $Y$ is hidden.
Avoiding attacks (3)

- An attacker could exploit the matrix $H'_S = \begin{bmatrix} H' \\ B^\perp a \end{bmatrix}$

- Without $X$, $H'_S$ has the same kernel as $H_S$, and it can be successfully exploited by the attacker.

- With a random (non-singular) $X$, this is no longer possible (already verified, paper in preparation).
## Security level and Key Size

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

<table>
<thead>
<tr>
<th></th>
<th>( n = 4096 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>3004 2884 2764 2644 2524 2404 2284 2164 2044 1924</td>
</tr>
<tr>
<td>( t )</td>
<td>91 101 111 121 131 141 151 161 171 181</td>
</tr>
<tr>
<td>WF</td>
<td>180 184 187 189 189 187 184 180 176</td>
</tr>
<tr>
<td>KS</td>
<td>400.4 426.7 449.4 468.6 484.3 496.5 505.2 510.4 512.0 510.1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>( n = 520 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>348 340 332 324 316 308 300 292 284 276</td>
</tr>
<tr>
<td>( t )</td>
<td>86 90 94 98 102 106 110 114 118 122</td>
</tr>
<tr>
<td>WF</td>
<td>180 181 182 183 183 183 183 182 181</td>
</tr>
<tr>
<td>KS</td>
<td>367.9 361.1 354.2 347.3 340.2 333.1 325.9 318.7 311.3 303.9</td>
</tr>
</tbody>
</table>

Marco Baldi - Code-based cryptosystems with short keys   October 7, 2015
Security level and Key Size

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

<table>
<thead>
<tr>
<th>$k$</th>
<th>$t$</th>
<th>WF</th>
<th>KS</th>
<th>$n = 8192$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6957</td>
<td>95</td>
<td>261</td>
<td>1048.8</td>
<td>log₂ KiB</td>
</tr>
<tr>
<td>6892</td>
<td>100</td>
<td>267</td>
<td>1093.7</td>
<td></td>
</tr>
<tr>
<td>6827</td>
<td>105</td>
<td>273</td>
<td>1137.6</td>
<td></td>
</tr>
<tr>
<td>6762</td>
<td>110</td>
<td>279</td>
<td>1180.4</td>
<td></td>
</tr>
<tr>
<td>6697</td>
<td>115</td>
<td>285</td>
<td>1222.2</td>
<td></td>
</tr>
<tr>
<td>6632</td>
<td>120</td>
<td>290</td>
<td>1262.9</td>
<td></td>
</tr>
<tr>
<td>6567</td>
<td>125</td>
<td>295</td>
<td>1302.7</td>
<td></td>
</tr>
<tr>
<td>6502</td>
<td>130</td>
<td>299</td>
<td>1341.4</td>
<td></td>
</tr>
<tr>
<td>6437</td>
<td>135</td>
<td>303</td>
<td>1379.0</td>
<td></td>
</tr>
<tr>
<td>6372</td>
<td>140</td>
<td>307</td>
<td>1415.7</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$k$</th>
<th>$t$</th>
<th>WF</th>
<th>KS</th>
<th>$n = 796$</th>
</tr>
</thead>
<tbody>
<tr>
<td>564</td>
<td>116</td>
<td>260</td>
<td>901.9</td>
<td>log₂ KiB</td>
</tr>
<tr>
<td>556</td>
<td>120</td>
<td>262</td>
<td>891.8</td>
<td></td>
</tr>
<tr>
<td>548</td>
<td>124</td>
<td>264</td>
<td>881.7</td>
<td></td>
</tr>
<tr>
<td>540</td>
<td>128</td>
<td>266</td>
<td>871.4</td>
<td></td>
</tr>
<tr>
<td>532</td>
<td>132</td>
<td>267</td>
<td>861.1</td>
<td></td>
</tr>
<tr>
<td>524</td>
<td>136</td>
<td>269</td>
<td>850.7</td>
<td></td>
</tr>
<tr>
<td>516</td>
<td>140</td>
<td>270</td>
<td>840.3</td>
<td></td>
</tr>
<tr>
<td>508</td>
<td>144</td>
<td>271</td>
<td>829.7</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>148</td>
<td>271</td>
<td>819.1</td>
<td></td>
</tr>
<tr>
<td>492</td>
<td>152</td>
<td>272</td>
<td>808.4</td>
<td></td>
</tr>
</tbody>
</table>
DIGITAL SIGNATURE SCHEMES BASED ON SPARSE SYNDROMES

(another example of use of the second approach)
From PKC to Digital Signatures

RSA

encryption map

McEliece

Marco Baldi - Code-based cryptosystems with short keys  October 7, 2015
Courtois-Finiasz-Sendrier (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 vector \( s \) such that \( s' = S^{-1} s \) is 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 $s = F(H(D))$ and $s' = S^{-1} s$
  - He decodes the syndrome $s'$ through the secret code
  - The error vector $e$ is the signature

- **Verification**:
  - The verifier computes $s = F(H(D))$
  - He checks that $H' e^T = S H e^T = S S^{-1} s = s$
The main issue is to find an efficient function $F(h)$.

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 are 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 operation

---

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: $H = [X | I]$, with $I$ an $r \times r$ identity matrix
  - $s$ is an $r \times 1$ sparse syndrome vector
  - the error vector $e = [0 | s^T]$ is sparse and verifies $H e^T = s$
Key generation

- Private key: \{Q, H, S\}, with
  - \(H\): \(r \times n\) parity-check matrix of the secret code \(C(n, k)\)
  - \(Q = R + T\)
  - \(R = a^T b\), 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: \(H' = Q^{-1} H S^{-1}\)
Signature generation

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

- He checks that the weight of $e'$ is $\leq (m_Tw + w_c)m_S$, otherwise the signature is discarded

- He computes $s^* = F(H(M))$ and checks that it has weight $w$, otherwise the signature is discarded

- He computes $H' e'^T = Q^{-1} H S^{-1} S (e^T + c^T) = Q^{-1} H (e^T + c^T) = Q^{-1} H e^T = Q^{-1} s' = s$

- If $s = s^*$, 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
Examples

- 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)
Attacks to regular $S$

- If the matrix $S$ is (sparse and) regular, statistical arguments could be used to analyze large number of intercepted signatures (pointed out by J. P. Tillich, analysis in progress)

- 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)
Attacks to regular S
Attacks to regular $S$

- An attacker has probability of success equal to the inverse of the security level ($SL$) when he collects at least $L_{\text{min}}$ signatures.

<table>
<thead>
<tr>
<th>$SL$</th>
<th>$L_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>3565</td>
</tr>
<tr>
<td>120</td>
<td>14296</td>
</tr>
<tr>
<td>160</td>
<td>37947</td>
</tr>
</tbody>
</table>

- This can be avoided by using an irregular matrix $S$ with the same average weight (already verified, paper in preparation).