# A New Analysis of the McEliece Cryptosystem based on QC-LDPC Codes

Marco Baldi[1] Marco Bodrato[2] Franco Chiaraluce[1]

DEIT, Università Politecnica delle Marche, Ancona, Italy

CIVV, Università di Roma "Tor Vergata", Roma, Italy

SCN 2008 - September 11$^{\text{th}}$

see appendices

# The McEliece cryptosystem

- Public-key cryptosystem based on algebraic coding theory [McEliece1978].
- It adopts generator matrices as private and public keys.
- Security lies in the difficulty of decoding a large linear code with no visible structure, that is an NP complete problem [Berlekamp1978].

**Advantages**

The system is faster than competing solutions, like RSA.

**Drawbacks**

It has large public keys and low transmission rate.


- McEliece, R.J., "A public-key cryptosystem based on algebraic coding theory." DSN Progress Report (1978) 114–116
- Berlekamp, E., McEliece, R., van Tilborg, H., "On the inherent intractability of certain coding problems." IEEE Trans. Inform. Theory **24** (May 1978) 384–386

# The McEliece cryptosystem (2)

- The original version adopts Goppa codes with length $n = 1024$, dimension $k = 524$, and minimum distance $d_{\min}$ of at least 101.
- The key size is $n \times k$ bits $= 67072$ bytes.
- The transmission rate is $k/n \approx 0.5$.
- Several attempts have been made for adopting other codes, able to overcome the system's drawbacks...
- ...but they always compromised the system security [Niederreiter1986], [Gaborit2005].


- Niederreiter, H., "Knapsack-type cryptosystems and algebraic coding theory." Probl. Contr. and Inform. Theory **15** (1986) 159–166
- Gaborit, P., "Shorter keys for code based cryptography." Proc. WCC 2005, Bergen, Norway (March 2005) 81–90

# Low-Density Parity-Check Codes

- LDPC codes are state-of-art forward error correcting codes.
- They approach the theoretical Shannon limit [Richardson2001].
- Each code is defined as the kernel of a sparse $(n-k) \times n$ binary matrix $\mathbf{H}$.
- Belief Propagation decoding exploits the sparse nature of their matrices to implement very efficient and low-complexity decoding.
- Quasi-cyclic (QC) LDPC codes are a particular class of LDPC codes, characterized by structured $\mathbf{H}$ matrices that allow low-complexity encoding too.

Richardson, T., Urbanke, R., "The capacity of low-density parity-check codes under message-passing decoding." IEEE Trans. Inform. Theory **47** (February 2001) 599–618

# QC-LDPC Codes

We consider a particular class of QC-LDPC codes, for which

## Matrix $\mathbf{H}$

is formed by a row $\{\mathbf{H}_0, \ldots, \mathbf{H}_{n_0-1}\}$ of $n_0$ binary circulant blocks with size $p$ and row/column weight $d_v$.

## The generator matrix $\mathbf{G}$

is formed by a $k \times k$ identity matrix $\mathbf{I}$ ($k = k_0 \cdot p$), followed by a column of $k_0$ binary circulant blocks with size $p$. If $\mathbf{H}_{n_0-1}$ is non-singular,

$$\mathbf{G} = \begin{bmatrix} & \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_0\right)^T \\ & \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_1\right)^T \\ \mathbf{I} & \vdots \\ & \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_{n_0-2}\right)^T \end{bmatrix}.$$

# McEliece cryptosystem based on QC-LDPC Codes

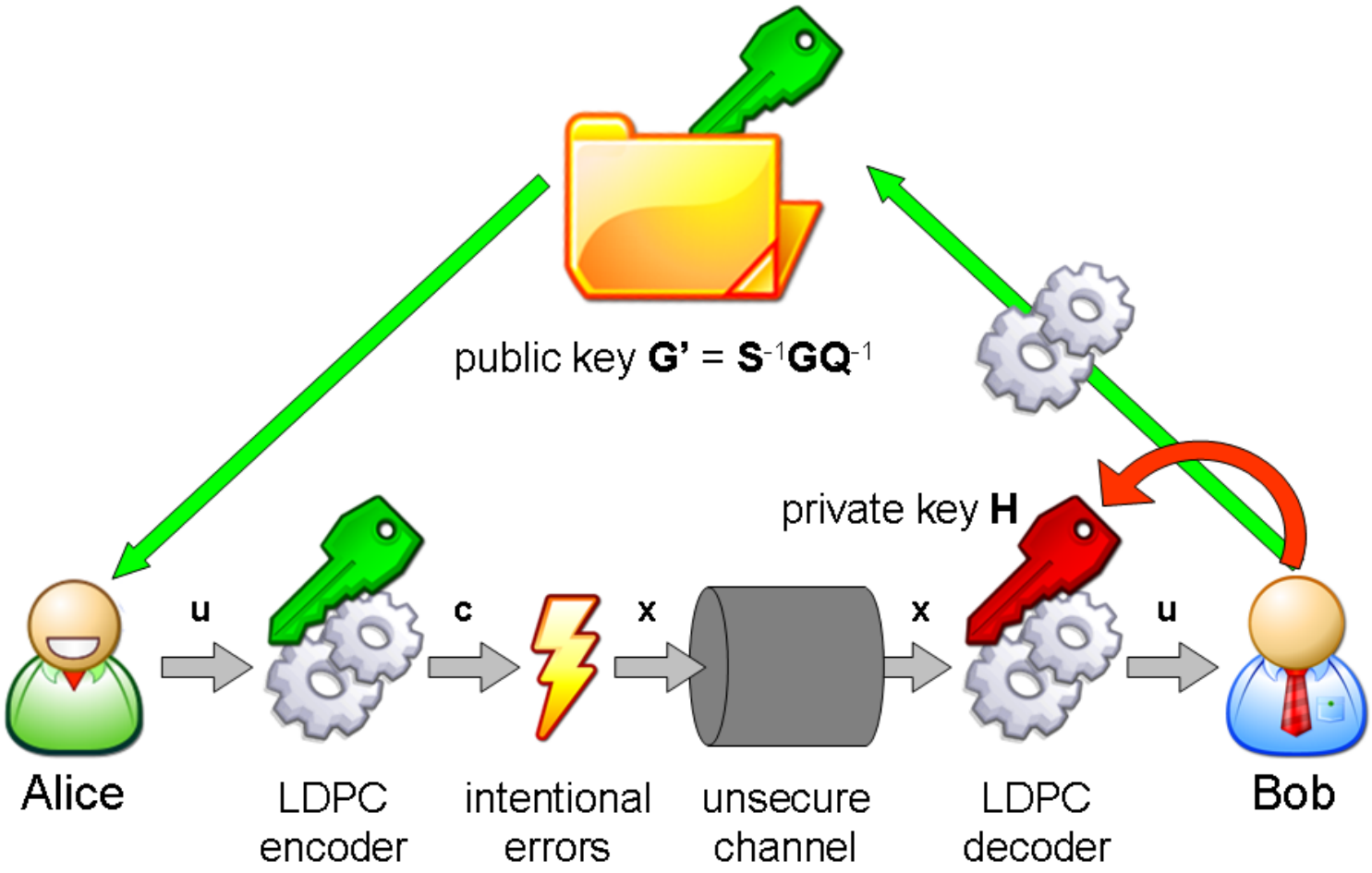
- In the original system (adopting Goppa codes) it suffices to hide the chosen code through a permutation.
- When adopting LDPC codes, the sparse nature of the **H** matrix must be hidden to avoid attacks based on it.
- We have recently proposed a QC-LDPC based variant that adopts a "dense" transformation [Baldi2007].
- This causes an "error spreading" phenomenon during decryption...
- ...but it is compensated by the high correction capability of LDPC codes.
- This version is able to counter all the classic attacks.

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

# McEliece cryptosystem based on QC-LDPC Codes (2)

- Bob randomly chooses a code in a family of $(n_0, d_v, p)$ QC-LDPC codes by selecting its parity-check matrix $\mathbf{H}$.
- Bob produces a generator matrix $\mathbf{G}$ in reduced echelon form.
- Bob randomly chooses a $k \times k$ non-singular matrix $\mathbf{S}$ and a sparse $n \times n$ non-singular matrix $\mathbf{Q}$ with row/column weight $m$.
- Bob computes the public key as $\mathbf{G}' = \mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{Q}^{-1}$.
- $\mathbf{G}'$ can be seen as a $k_0 \times n_0$ matrix with entries in the ring of polynomials $\mathbb{R} = \mathrm{GF}(2)[x]/(x^p + 1)$, so it can be simply described by the set of polynomial coefficients.
- Alice uses $\mathbf{G}'$ for encoding her message, before adding $t'$ intentional errors: $\mathbf{x} = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e} = \mathbf{c} + \mathbf{e}$.
- Bob uses $\mathbf{H}$ (the private key) for LDPC decoding.

# McEliece cryptosystem based on QC-LDPC Codes (3)



# System Parameters

- In the previous version of the cryptosystem we fixed $n_0 = 4$, $d_v = 13$, $p = 4032$, $m = 7$ and $t' = 27$.
- Such choice allows to resist all the standard attacks.
- Both $\mathbf{S}$ and $\mathbf{Q}$ were chosen sparse, with non-null blocks having row/column weight $m$, and

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{n_0-1} \end{bmatrix}.$$

- This, together with its low density, gave raise to a new attack formulated by Otmani et al. (OTD) [Otmani2008].

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

# Rationale of the OTD attacks

- By selecting the first $k$ columns of $\mathbf{G}'$, an eavesdropper can obtain

$$\mathbf{G}'_{\leq k} = \mathbf{S}^{-1} \cdot \begin{bmatrix} \mathbf{Q}_0^{-1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1^{-1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{Q}_{n_0-2}^{-1} \end{bmatrix}.$$

- By inverting $\mathbf{G}'_{\leq k}$ and considering its block at position $(i, j)$, he can obtain $\bar{\mathbf{Q}}_i \mathbf{S}_{i,j}$, that corresponds to the polynomial

$$g_{i,j}(x) = q_i(x) \cdot s_{i,j}(x) \bmod (x^p + 1).$$

- Both $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$ are sparse, so it is highly probable that $g_{i,j}(x)$ has exactly $m^2$ non-null coefficients and its support contains at least one shift $x^{l_a} \cdot q_i(x)$, $0 \leq l_a \leq p-1$.

# OTD attack strategies

1. OTD1: requires on average $2^{50}$ binary operations
2. OTD2: requires on average $2^{36}$ binary operations
3. OTD3: requires on average $2^{32}$ binary operations

## Countermeasure

- The OTD attack strategies rely on the fact that both **S** and **Q** are sparse and that **Q** has block-diagonal form.
- They can be countered by adopting dense **S** matrices, without altering the remaining system parameters.
- For example, **S** could have density about 0.5, with odd weight blocks along the main diagonal, and even weight blocks elsewhere, to ensure non-singularity.

# Rationale of the new proposals

- With dense $\mathbf{S}$ matrices the eavesdropper cannot obtain $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$, even knowing $\mathbf{Q}_i\mathbf{S}_{i,j}$.
- To preserve the ability of correcting all the intentional errors, $\mathbf{Q}$ is kept sparse (with row/column weight $m$).
- The choice of a dense $\mathbf{S}$ influences complexity of the decoding stage, that, however, can be reduced by resorting to efficient computation algorithms for circulant matrices.
- The OTD attacks demonstrate that the choice of $\mathbf{Q}$ in block-diagonal form is weak, so we avoid it in the new versions of the cryptosystem.

# First new variant

The first new variant adopts almost the same parameters of the previous one:

- $p = 4096$
- $n_0 = 4 \Rightarrow n = 16384$
- $k_0 = 3 \Rightarrow k = 12288 \Rightarrow R = 0.75$
- $d_v = 13$, $m = 7$ and $t' = 27$

- $\mathbf{Q}$ is obtained by randomly permuting the block rows and columns of a matrix of $4 \times 4$ circulant blocks with weight 2, except those along the main diagonal, that have weight 1.
- The absence of the block-diagonal structure in $\mathbf{Q}$ prevents from attacking each single block, and attacking a whole row or column would require $p\binom{p}{2}^3 \approx 2^{81}$ attempts.
- $\mathbf{S}$ is dense, with row/column weight $\approx k/2$.

# Second new variant

The second new variant adopts a new choice of the parameters that ensures increased security:

- $p = 8192$
- $n_0 = 3 \Rightarrow n = 24576$
- $k_0 = 2 \Rightarrow k = 16384 \Rightarrow R = 0.67$
- $d_v = 13$, $m = 11$ and $t' = 40$

- It is obtained at the cost of a slightly decreased transmission rate.
- $\mathbf{Q}$ is obtained by randomly permuting the block rows and columns of a matrix of $3 \times 3$ circulant blocks with weight 4, except those along the main diagonal, that have weight 3.
- Attacking a whole row or column of $\mathbf{Q}$ would require $\binom{p}{4}^2 \binom{p}{3} \approx 2^{131}$ attempts.
- $\mathbf{S}$ is dense, with row/column weight $\approx k/2$.

# Vector-matrix: naïve evaluations

Both encryption and decryption require a vector-matrix product. With the usual naïve algorithm, the product of a vector with $k$ entries by a $k \times n$ matrix requires:

- $k \cdot n$ products,
- $k \cdot n$ additions.

Since the matrix is fixed, we can *hard-code* the products and compute only the needed additions, those corresponding with a 1 entry. With dense matrices, about half of them.
Then we assume $\frac{k \cdot n}{2}$ operations.

# Circulant matrices: definition

## Toeplitz square matrices — entries in $\mathbb{F} = \mathrm{GF}(m)$

$$\mathbf{A} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{p-2} & a_{p-1} \\ a_{-1} & a_0 & a_1 & \cdots & a_{p-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{2-p} & \cdots & a_{-1} & a_0 & a_1 \\ a_{1-p} & a_{2-p} & \cdots & a_{-1} & a_0 \end{bmatrix}$$

is circulant iff $\forall i, a_i = a_{i-p} = a_{i+p}$.

Let $\mathbf{X}$ be the circulant matrix with $a_i = \delta_1(i)$.

## Isomorphism between circulants and $\mathbb{R} = \mathbb{F}[x]/(x^p + 1)$

$$\text{Circulants} \ni \sum_{i=0}^{p-1} \alpha_i \mathbf{X}^i \quad \longleftrightarrow \quad \sum_{i=0}^{p-1} \alpha_i x^i \in \mathbb{F}[x]/(x^p + 1)$$

# Vector-circulant multiplication is a polynomial product

Multiplication of a $p$-sized vector by a $p \times p$ circulant matrix is equivalent to multiplication of two polynomials modulo $x^p + 1$. This means we can use fast product.

**Many algorithms are known for polynomial multiplication.**

- Naïve $\mathrm{O}(p^2)$

Each one has a different complexity, and a different range where it is the fastest.

see thresholds

# Vector-circulant multiplication is a polynomial product

Multiplication of a $p$-sized vector by a $p \times p$ circulant matrix is equivalent to multiplication of two polynomials modulo $x^p + 1$. This means we can use fast product.

## Many algorithms are known for polynomial multiplication.

- Naïve $\mathrm{O}(p^2)$
- Karatsuba (Toom-2) $\mathrm{O}(p^{\log_2 3})$

## Karatsuba or Winograd can be used

Splitting in two can be easily implemented.

# Vector-circulant multiplication is a polynomial product

Multiplication of a $p$-sized vector by a $p \times p$ circulant matrix is equivalent to multiplication of two polynomials modulo $x^p + 1$. This means we can use fast product.

## Many algorithms are known for polynomial multiplication.

- Naïve $\mathrm{O}(p^2)$
- Karatsuba (Toom-2) $\mathrm{O}(p^{\log_2 3})$
- Toom-Cook-$k$ $\mathrm{O}(p^{\log_k 2k-1})$

## Toom-Cook strategy has been extended to binary polynomials

Splitting in three or four parts is trickier, but faster in our range.

# Vector-circulant multiplication is a polynomial product

Multiplication of a $p$-sized vector by a $p \times p$ circulant matrix is equivalent to multiplication of two polynomials modulo $x^p + 1$. This means we can use fast product.

**Many algorithms are known for polynomial multiplication.**

- Naïve — $\mathrm{O}(p^2)$
- Karatsuba (Toom-2) — $\mathrm{O}(p^{\log_2 3})$
- Toom-Cook-$k$ — $\mathrm{O}(p^{\log_k 2k-1})$
- Schönhage-FFT/Cantor — $\mathrm{O}(p \log p \log\log p)$

**Schönhage-FFT or Cantor are inefficient for our sizes**

We are not interested in asymptotic complexity...

# Fast product is very effective on matrices

**All the fast polynomial products use three phases:**

1. Evaluations (both operands) — Overhead
2. Point-wise products
3. Interpolation — Overhead

Every product requires all the three phases.
12 products $\rightsquigarrow$ 24 evaluations, 12 mid-products, 12 interpolations.

# Fast product is very effective on matrices

**All the fast polynomial products use three phases:**

1. Evaluations (both operands) — Overhead
2. Point-wise products
3. Interpolation — Overhead

Every product requires all the three phases.
12 products $\rightsquigarrow$ 24 evaluations, 12 mid-products, 12 interpolations.

$$\begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} \begin{pmatrix} M_{1,1} & M_{1,2} & M_{1,3} & M_{1,4} \\ M_{2,1} & M_{2,2} & M_{2,3} & M_{2,4} \\ M_{3,1} & M_{3,2} & M_{3,3} & M_{3,4} \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 & r_4 \end{pmatrix}$$

The matrix is fixed and pre-evaluated, interpolation is linear:
12 products $\rightsquigarrow$ 3 evaluations, 12 mid-products, 4 interpolations.

# Winograd's: a simpler approach

## Use the fact that circulant matrices are Toeplitz

Circulant matrices are Toeplitz's; can be decomposed as shown:

$$\begin{pmatrix} \mathbf{T}_0 & \mathbf{T}_1 \\ \mathbf{T}_2 & \mathbf{T}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{T}_1 - \mathbf{T}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_2 - \mathbf{T}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_0 \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{I} \end{pmatrix}$$

### Advantages

- Very simple to implement
- Asymptotically equivalent to Toom-2, with a smaller overhead

### Drawbacks

- Does not work with odd-sized matrices
- Cannot be mixed with Toom-Cook

# Winograd and Toom-Cook: complexity comparisons

## Number of bit operations

| p | dim | Naïve | Winograd | Toom-Cook |
|---|---|---|---|---|
| 8192 | $2 \times 3$ | 201 326 592 | 14 106 224 | 12 684 343 |
| 4096 | $3 \times 4$ | 100 663 296 | 8 103 706 | 8 074 444 |
| 5555 | $3 \times 4$ | 92 574 075 | N.A. | 7 310 809 |

The Toom-Cook approach is the fastest (in a wide range), and can be used for all sizes.

We can suggest Winograd for test implementations, it is fast enough, and it is far simpler.

# Encryption and decryption complexity

- Encryption complexity is due to encoding of the cleartext and to addition of intentional errors:

$$C_{enc} = C_{mul}\left(\mathbf{u} \cdot \mathbf{G}'\right) + n.$$

- The decryption complexity can expressed as:

$$C_{dec} = C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right) + C_{SPA} + C_{mul}\left(\mathbf{u}' \cdot \mathbf{S}\right).$$

- $C_{SPA}$ is the number of operations required for LDPC decoding through the sum-product algorithm [Hu2001]:

$$C_{SPA} = I_{ave} \cdot n\left[q\left(8d_v + 12R - 11\right) + d_v\right]$$

with $I_{ave}$ average number of decoding iterations and $q$ number of quantization bits used inside the decoder.

Hu, X.-Y., Eleftheriou, E., Arnold, D.-M., Dholakia, A., "Efficient implementations of the sum-product algorithm for decoding LDPC codes." Proc. IEEE GLOBECOM '01, San Antonio, TX (Nov. 2001) 1036–1036E

| | McEliece (original) | Niederreiter | RSA | QC-LDPC McEliece 1 | QC-LDPC McEliece 2 |
|---|---|---|---|---|---|
| Key Size (bytes) | 67072 | 32750 | 256 | 6144 | 6144 |
| Information Bits | 524 | 276 | 1024 | 12288 | 16384 |
| Transmission Rate | 0.5117 | 0.5681 | 1 | 0.75 | 0.6667 |
| Enc Ops per bit | 514 | 50 | 2402 | 658 | 776 |
| Dec Ops per bit | 5140 | 7863 | 738112 | 4678 | 8901 |

- The new versions are secure against the known attacks.
- The lowest work factor is achieved by local deduction attacks known as "information set decoding".
- Such attacks require more than $2^{70}$ and $2^{80}$ operations for the two new variants, respectively.
- The McEliece and Niederreiter cryptosystems with their standard parameters reach lower security levels.

# Conclusions

- We have shown that the adoption of QC-LDPC codes in the McEliece cryptosystem can help overcoming its drawbacks.
- Our previous proposal, however, was exposed to newly developed total break attacks.
- We have proposed two new variants of the cryptosystem secure against such attacks.
- We have reported complexity estimates based on the Toom-Cook method for polynomials in $\mathrm{GF}(2)[x]$, that permits to reduce the encryption and decryption complexity of the proposed cryptosystems.
- They can be seen as a trade-off between the original McEliece cryptosystem and other widespread solutions, like RSA.

# Open problems

- There is still no supporting proof of security for coding based cryptographic techniques.
- The error correction capability of LDPC codes over the "McEliece" channel has been assessed through numerical simulations, but total error correction is not guaranteed as for Goppa codes.

Thank you very much for your kind attention

Presentation will be available on the web:
http://bodrato.it/papers/#SCN2008,

4 Polynomial product
- Exact division
- Timings

back to index

# Exact division

## detailed only for $D = x^n + 1 \in \mathrm{GF}(2)[x]$

We start from an element $\mathrm{GF}(2)[x] \ni a = qD$, whose degree is $\deg(a) = d + n$. We want the quotient $q$. Compute with $2^k n \leqslant d$.

$$q \equiv a \cdot (1 + x^n) \cdot (1 + x^{2n}) \cdots (1 + x^{2^k n}) \pmod{x^{d+1}}$$

Division can be performed limb by limb starting from less significant one, obtaining linear complexity.

### Division limb by limb obtain linear complexity

for $i = 0 \ldots d/w$

$a_i \leftarrow a_i \cdot D^{-1} \pmod{x^w}$

$a_{i+1} \leftarrow a_{i+1} - \frac{a_i \cdot D}{x^w} = a_{i+1} - a_i >> (w - n)$

Thanks to Jörg Arndt for suggesting a clean description

# Thresholds for polynomial product

## NTL-based plain implementations

### Range where each algorithm is the fastest

| Algorithm | operand degree (bits) | | | asymptotic |
|---|---|---|---|---|
| Naïve | | $<$ | 190 | $\mathrm{O}(d^2)$ |
| Karatsuba | 190 | ... | 360 | $\mathrm{O}(d^{\log_2 3})$ |
| Toom-3 | 360 | ... | 8,000 | $\mathrm{O}(d^{\log_3 5})$ |
| Toom-4 | 8,000 | ... | 15,000 | $\mathrm{O}(d^{\log_4 7})$ |
| Schönhage-FFT | 15,000 | $<$ | | $\mathrm{O}(d \log d \log\log d)$ |

Those values highly depend on implementation, architecture...

Algorithms in blue where implemented by Paul Zimmermann, the others by Marco Bodrato