Counting Coprime Polynomials over Finite Fields with Formal Languages and Compositions of Natural Numbers

Luca Mariot
Joint work with Enrico Formenti

l.mariot@utwente.nl

AI@Bicocca Seminars – November 16, 2023

Coprime Polynomials

Object: pairs of binary polynomials of degree $n \in \mathbb{N}$:

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n \ ,$$
$$g(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1} + x^n \ ,$$

where $a_i, b_i \in GF(2) = \mathbb{F}_2 = \{0, 1\}$

$$f, g \in \mathbb{F}_2[x] \text{ are } \textbf{coprime} \Leftrightarrow \gcd(f, g) = 1$$

Applications cryptography and coding theory:

- *Discrete logarithms* in finite fields [C84]
- Decoding *alternant codes* [F95]

Euclid’s Algorithm

Check if $\gcd(f, g) = 1 \Rightarrow$ **Euclid’s algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

Euclid's Algorithm

Check if $\gcd(f, g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3$, $f(x) = x^3 + x^2 + x + 1$, $g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$
$x^3 + 1 = (x + 1) \cdot (x^2 + x) + (x + 1)$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$
$x^3 + 1 = (x + 1) \cdot (x^2 + x) + (x + 1)$
$x^2 + x = x \cdot (x + 1) + 0$

Euclid's Algorithm

Check if $\gcd(f,g) = 1 \Rightarrow$ **Euclid's algorithm**

Example: $n = 3, \quad f(x) = x^3 + x^2 + x + 1, \quad g(x) = x^3 + 1$

$$f(x) = q(x) \cdot g(x) + r(x)$$

$$x^3 + x^2 + x + 1 = 1 \cdot (x^3 + 1) + (x^2 + x)$$
$$x^3 + 1 = (x + 1) \cdot (x^2 + x) + (x + 1)$$
$$x^2 + x = x \cdot (x + 1) + 0$$

Compact notation:

$$(x^3 + x^2 + x + 1, x^3 + 1) \xrightarrow{1} (x^3 + 1, x^2 + x) \xrightarrow{x+1} (x^2 + x, x + 1) \xrightarrow{x} (x + 1, 0)$$

Euclid's Algorithm

Check if $\gcd(f,g)=1 \Rightarrow$ **Euclid's algorithm**

Example: $n=3, \quad f(x)=x^3+x^2+x+1, \quad g(x)=x^3+1$

$$f(x)=q(x)\cdot g(x)+r(x)$$

$$\begin{aligned}
&x^3+x^2+x+1=1\cdot(x^3+1)+(x^2+x)\\
&x^3+1=(x+1)\cdot(x^2+x)+(x+1)\\
&x^2+x=x\cdot(x+1)+0
\end{aligned}$$

Compact notation:

$(x^3+x^2+x+1,x^3+1)\xrightarrow{1}(x^3+1,x^2+x)\xrightarrow{x+1}(x^2+x,x+1)\xrightarrow{x}(x+1,0)$

$$gcd(f,g)=x+1 \Rightarrow (f,g)\ \textbf{not}\ \text{coprime}$$

DilcuE's Algorithm

- **Remark:** (f,g) can be recovered from $(x+1,0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$(x+1,0) \xrightarrow{x}$

DilcuE's Algorithm

- **Remark:** (f, g) can be recovered from $(x+1, 0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1, 0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1}$$

DilcuE's Algorithm

- **Remark:** (f, g) can be recovered from $(x+1, 0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$(x+1,0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1} (x^3+1, x^2+x) \xrightarrow{1}$
$(x^3+x^2+x+1, x^3+1) = (f, g)$

DilcuE's Algorithm

- **Remark:** (f,g) can be recovered from $(x+1,0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$(x+1,0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1} (x^3+1, x^2+x) \xrightarrow{1}$
$(x^3+x^2+x+1, x^3+1) = (f,g)$

- Suppose we change the **last** remainder from 0 to 1:

$(x+1,\mathbf{1}) \xrightarrow{x}$

DilcuE's Algorithm

- **Remark:** (f,g) can be recovered from $(x+1,0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1,0) \xrightarrow{x} (x^2+x, x+1) \xrightarrow{x+1} (x^3+1, x^2+x) \xrightarrow{1} (x^3+x^2+x+1, x^3+1) = (f,g)$$

- Suppose we change the **last** remainder from 0 to 1:

$$(x+1,\mathbf{1}) \xrightarrow{x} (x^2+x+1, x+1) \xrightarrow{x+1}$$

DilcuE's Algorithm

- **Remark:** (f,g) can be recovered from $(x+1,0)$ with the same quotients in reverse order
- Called **DilcuE's algorithm** by Benjamin and Bennett [BB07]

$$(x+1,0) \xrightarrow{x} (x^2+x,x+1) \xrightarrow{x+1} (x^3+1,x^2+x) \xrightarrow{1} (x^3+x^2+x+1,x^3+1) = (f,g)$$

- Suppose we change the **last** remainder from 0 to 1:

$$(x+1,\mathbf{1}) \xrightarrow{x} (x^2+x+1,x+1) \xrightarrow{x+1} (x^3+x^2,x^2+x+1) \xrightarrow{1} (x^3+x+1,x^3+x^2) = (f',g')$$

- By construction, (f',g') **are coprime**

Counting by Bijection

In essence: bijection for coprime/non-coprime pairs over $\mathbb{F}_2$:

1. Apply Euclid to (f, g)
2. If the last remainder is 0, change it to 1. Otherwise, set it to the second-last remainder
3. Apply DilcuE's algorithm to the reversed quotients

Theorem ([BB07, R00])

Let $f, g \in \mathbb{F}_2[x]$ of degree n be randomly chosen. Then, the probability that $\gcd(f, g) = 1$ is $\frac{1}{2}$.

In other words: the number of coprime pairs is 2^{2n-1}

Enter the complication

We require now that both f and g have a **nonzero constant term**:

$$f(x) = \mathbf{1} + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n \ ,$$
$$g(x) = \mathbf{1} + b_1 x + \cdots + b_{n-1} x^{n-1} + x^n \ .$$

Problems:

1. *Count* all such pairs
2. *Enumeration algorithm*

Remark: the trick above does not work!

non-coprime $\leftrightarrow$ **coprime**

$$(x^3 + x^2 + x + \mathbf{1}, x^3 + \mathbf{1}) \leftrightarrow (x^3 + x + \mathbf{1}, x^3 + x^2 \ (+\mathbf{0}))$$

... Why do we want to do that?

n-MOLS by Cellular Automata

- Bipermutive Linear rule: $f(x) = x_1 \oplus a_1 x_2 \oplus \cdots \oplus a_{n-1} x_{n-1} \oplus x_n$
- Associated Polynomial: $P_f(X) = 1 + a_1 X + \cdots + a_{n-1} X^{n-1} + X^n$

Theorem ([MGFL20])

n bipermutive linear CA generates a set of n-MOLS if and only if their associated polynomials are pairwise coprime

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(a) Rule 150

1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

(b) Rule 90

1, 1	4, 2	3, 3	2, 4
2, 2	3, 1	4, 4	1, 3
4, 3	1, 4	2, 1	3, 2
3, 4	2, 3	1, 2	4, 1

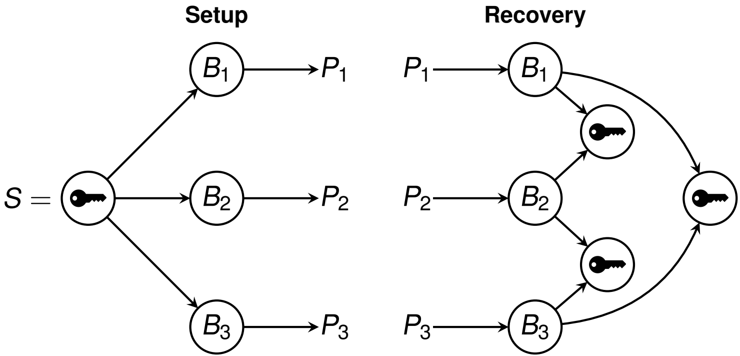
(c) Superposition

Figure: $P_{150}(X) = 1 + X + X^2$, $P_{90}(X) = 1 + X^2$ (coprime)

Secret Sharing Schemes (SSS)

- (k, n)–Secret sharing scheme share a secret S among n players, so that at least k of them are required to recover S
- $(2, n)$–schemes $\Leftrightarrow$ families of n-MOLS

Example: $(2,3)$–scheme



Asking for clues (7 years ago...)

Asking for clues (7 years ago...)

Dear Arthur, what do you think of this complication?

Luca

Asking for clues (7 years ago...)

Dear Arthur, what do you think of this complication?

Luca

Arthur Benjamin

Dear Luca, off the top of my head, there are $q^2 - 1$ equivalence classes, all of which are co-prime except one? But I may be wrong.

Asking for clues (7 years ago...)

Dear Arthur, what do you think of this complication?

Luca

Arthur Benjamin

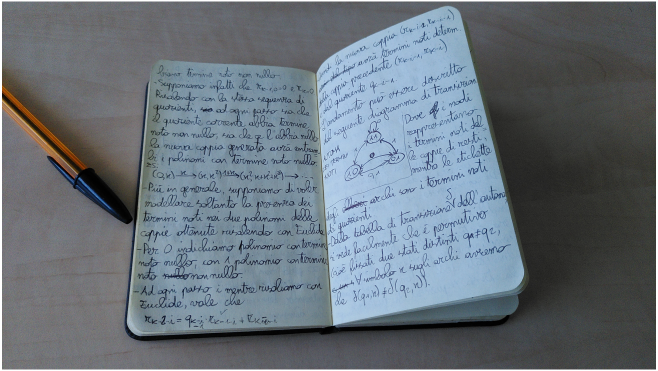
Dear Luca, off the top of my head, there are $q^2 - 1$ equivalence classes, all of which are co-prime except one? But I may be wrong.

?

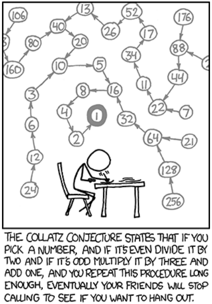
One MONTH later...

One MONTH later...

... He was indeed right! But took me several weeks to prove it



Sadly, the clue was not enough to solve the counting problem



S `https://xkcd.com/710/`

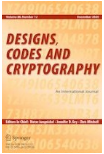
- Number of coprime polynomial pairs of degree n and nonzero constant term:

$$a(n) = 4^{n-1} + a(n-1) = \frac{4^{n-1}-1}{3}$$
$$= 0, 1, 5, 21, 85, \ldots$$

- Corresponds to OEIS A002450

- Generalized for any finite field $\mathbb{F}_q$ in [MGFL20] (but enumeration not addressed)

L. Mariot, M. Gadouleau, E. Formenti, and A. Leporati. Mutually orthogonal latin squares based on cellular automata. Des. Codes Cryptogr. 88(2):391–411 (2020)



Problem Structure

Strategy: characterize the *sequences* of quotients that gives only $(1,1)$ coprime pairs when starting from the remainders $(1,0)$

Three parts of the problem:

Problem Structure

Strategy: characterize the *sequences* of quotients that gives only $(1,1)$ coprime pairs when starting from the remainders $(1,0)$

Three parts of the problem:

$$
\begin{array}{llllll}
q_1 \rightarrow & \overbrace{x^{d_1}}^{\textit{degrees}} & + \overbrace{q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x}^{\text{middle terms}} & + & \overbrace{s_1}^{\text{constant terms}} \\
q_2 \rightarrow & x^{d_2} & + q_{2,d_2-1}x^{d_2-1} + \cdots + q_{2,1}x & + & s_2 \\
\vdots \rightarrow & \vdots & + \quad \vdots \quad + \cdots + \quad \vdots & + & \vdots \\
q_k \rightarrow & x^{d_k} & + q_{k,d_k-1}x^{d_k-1} + \cdots + q_{k,1}x & + & s_k
\end{array}
$$

Problem Structure

Strategy: characterize the *sequences* of quotients that gives only $(1,1)$ coprime pairs when starting from the remainders $(1,0)$

Three parts of the problem:

$$
\begin{array}{lcl}
q_1 \rightarrow & \overbrace{x^{d_1}}^{\textit{degrees}} & + \overbrace{q_{1,d_1-1}x^{d_1-1} + \cdots + q_{1,1}x}^{\text{middle terms}} + \overbrace{s_1}^{\text{constant terms}} \\
q_2 \rightarrow & x^{d_2} & + q_{2,d_2-1}x^{d_2-1} + \cdots + q_{2,1}x + s_2 \\
\vdots \rightarrow & \vdots & + \vdots + \cdots + \vdots + \vdots \\
q_k \rightarrow & x^{d_k} & + q_{k,d_k-1}x^{d_k-1} + \cdots + q_{k,1}x + s_k
\end{array}
$$

Notation: $r_i, r_{i+1} \rightarrow$ consecutive remainders produced by Euclid's algorithm at step i. Step $i+1$:

$$r_i(x) = q_{i+1}(x)r_{i+1}(x) + r_{i+2}(x)$$

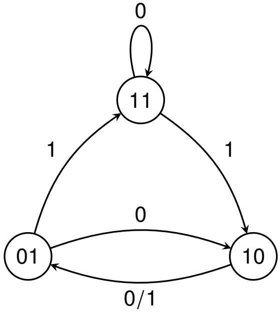
Finite State Automaton of Remainders

$$r_i(x) = q_{i+1}(x) r_{i+1}(x) + r_{i+2}(x)$$

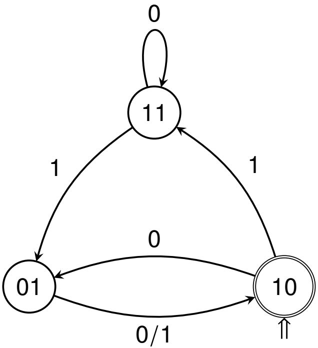
- $(c_i, c_{i+1}) \to$ constant terms of r_i and r_{i+1}
- $s_{i+1} \to$ constant term of q_{i+1}
- $\delta((c_i, c_{i+1}), s_{i+1}) \to$ *next* pair (c_{i+1}, c_{i+2})

(c_i, c_{i+1})	s_{i+1}	$\delta((c_i, c_{i+1}), s_{i+1})$
$(1,1)$	0	$(1,1)$
$(1,1)$	1	$(1,0)$
$(1,0)$	0	$(0,1)$
$(1,0)$	1	$(0,1)$
$(0,1)$	0	$(1,0)$
$(0,1)$	1	$(1,1)$

Remark: the pair $(0,0)$ *never* occurs



The Regular Language of Constant Terms Sequences



Inverse FSA

- The FSA is *permutative*: for DilcuE's, simply reverse the arrows
- **Initial state**: 10
- **Final state**: 11 (but we can use 10)

Regular Expression of the Language:

$$L = (0(0+1) + (10^*1(0+1)))^*$$

Enumeration/counting of Constant Terms Sequences

- **Enumeration**: generate all words of length k [M97]
- **Counting**: exploit *algebraic language theory*

Transform $L = (0(0+1)+(10^*1(0+1)))^*$ in a FPS as follows:

- $0,1 \Rightarrow X$
- $+,\cdot \Rightarrow +,\cdot$
- $^* \Rightarrow \frac{1}{1-X}$

Generating Function:

$$\sum_{k=0}^{\infty} a_k \cdot X^k = \frac{1-X}{1-X-2X^2} ,$$

Closed Form:

$$a_k = \frac{2^k + 2\cdot(-1)^k}{3}$$

Sequences of quotients' degrees

Second part: Characterize the *degrees* of the quotients

Example: $n = 4$, $\{1, x, x^2, x, 1\}$

$$(1,0) \xrightarrow{1} (1,1) \xrightarrow{x} (x+1,1) \xrightarrow{x^2} (x^3+x^2+1, x+1) \xrightarrow{x}$$
$$(x^4+x^3+1, x^3+x^2+1) \xrightarrow{1} (x^4+x^2+1, x^4+x^3+1)$$

Sum of degrees: $1+2+1 = 4$, $k = 3$

Question: what are the combinations of *ordered sums* of n?

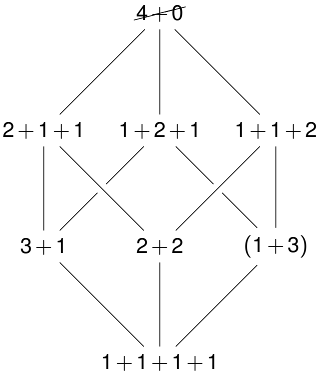
$\Rightarrow$ **compositions** of $n \in \mathbb{N}$

Quotients' degrees as compositions of n

- **Representation:** $n-1$ *boxes* that can be either "+" or ","

$$1\overbrace{\square 1 \square \ldots \square 1 \square}^{n-1}1$$

- **Example:** $1,1+1,1 \to 1+2+1 \quad (n=4,\, k=3)$



- We remove the top of the poset
- **Enumeration**: generate all binary strings of length n with k 1s
- **Counting**: $\binom{n-1}{k-1}$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n}$$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1,0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n} \underbrace{2^{n-k}}_{\text{middle}}$$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1, 0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n} \underbrace{2^{n-k}}_{\text{middle}} \cdot \underbrace{\binom{n-1}{k-1}}_{degrees}$$

Enumeration Algorithm

- **Third part**: middle terms are *free*
- once k is fixed, all three parts are *independent*

So for **enumeration**, given $n \in \mathbb{N}$:

For each composition *comp* of n of length k (except $k = 0$) do:

- Generate all quotients' sequences of *comp* (2^{n-k})
- For each quotients' sequence *seq* do:
  - For each constant term sequence of length k do:
    - Add the constant terms to the quotients
    - Apply DilcuE's from $(1,0)$ by applying *seq*

And for **counting**, we reobtain the formula $\frac{4^{n-1}-1}{3}$ from:

$$\sum_{k=2}^{n} \underbrace{2^{n-k}}_{\text{middle}} \cdot \underbrace{\binom{n-1}{k-1}}_{\textit{degrees}} \cdot \underbrace{\frac{2^k + 2\cdot(-1)^k}{3}}_{\textit{constant}}$$

Conclusions and Future Work

Summing up:

- Enumeration more complicated with nonzero constant terms
- We divided the problem in three tasks:
  1. sequences of constant terms ($\Rightarrow$ regular language)
  2. sequences of degrees ($\Rightarrow$ compositions)
  3. sequences of middle terms ($\Rightarrow$ free)
- Results informally published in [FM22]

Future directions:

- Generalize to any finite field $\mathbb{F}_q$ and to *m*-tuples of polynomials
- Applications to cryptography [GM20, M21, GMP22]

Thank you!

References

[BB07] Benjamin, A.T., Bennett, C.D.: The probability of relatively prime polynomials. Mathematics Magazine 80(3): 196-202 (2007)

[C84] Coppersmith, D.: Fast evaluation of logarithms in fields of characteristic two. IEEE Trans. Inf. Theory 30(4): 587-593 (1984)

[F95] Fitzpatrick, P.: On the key equation. IEEE Trans. Inf. Theory 41(5): 1290-1302 (1995)

[FM22] Formenti, E., Mariot, L.: An Enumeration Algorithm for Binary Coprime Polynomials with Nonzero Constant Term. CoRR abs/2207.00406 (2022)

[GMP22] Gadouleau, M., Mariot, L., Picek, S.: Bent functions in the partial spread class generated by linear recurring sequences. Des. Codes Cryptogr. In press. DOI: `https://doi.org/10.1007/s10623-022-01097-1` (2022)

[GM20] Gadouleau, M., Mariot, L.: Latin Hypercubes and Cellular Automata. Proceedings of Automata 2020, pp. 139-151 (2020)

[GR11] Ghorpade, S. R., Ram, S.: Block companion Singer cycles, primitive recursive vector sequences, and coprime polynomial pairs over finite fields. Finite Fields Their Appl. 17(5): 461-472 (2011)

[M97] E. Mäkinen: On Lexicographic Enumeration of Regular and Context-Free Languages. Acta Cybern. 13(1): 55-61 (1997)

[M21] Mariot, L.: Hip to Be (Latin) Square: Maximal Period Sequences from Orthogonal Cellular Automata. In: Proceedings of CANDAR 2021, pp. 29-37 (2021)

[MGFL20] Mariot, L., Gadouleau, M. Formenti, E., Leporati A.: Mutually orthogonal latin squares based on cellular automata. Des. Codes Cryptogr. 88(2):391–411 (2020)

[R00] Reifegerste, A.: On an involution concerning pairs of polynomials in $\mathbb{F}_2$. J. Combin. Theory Ser. A 90, 216-220 (2000)