Diffusion matrices from algebraic-geometry codes with efficient SIMD implementation

Daniel Augot[✧,⁜] Pierre-Alain Fouque[✿,⋆] **Pierre Karpman**[✧,✐,⁜]

[✧]Inria, France
[⁜]LIX — École polytechnique, France
[✿]Université Rennes 1, France
[⋆]Institut universitaire de France, France
[✐]Nanyang Technological University, Singapore

Selected Areas in Cryptography, Montréal
2014–08–15

Motivations: finding M with good diffusion

What is "good diffusion"?
$\Rightarrow$ branch number (Daemen & Rijmen, 2002)

Differential & linear branch number

If M is a matrix and $\mathrm{w}(\mathbf{x})$ the number of non-zero positions of the vector $\mathbf{x}$, the *differential branch number* of M is

$$\min_{\mathbf{x}\neq 0}(\mathrm{w}(\mathbf{x})+\mathrm{w}(M(\mathbf{x})))$$

The *linear branch number* of M is

$$\min_{\mathbf{x}\neq 0}(\mathrm{w}(\mathbf{x})+\mathrm{w}(M^t(\mathbf{x})))$$

$\Rightarrow$ Wide trail construction (Ibid.)

Motivations: using linear codes

Minimum distance & branch number

Let C be a $[2k, k, d]$ code and $(I_k \ M)$ a *systematic* generating matrix of C, then M has a differential branch number of d

$\Rightarrow$ The branch number is maximum if the code is MDS

Example: The AES *MixColumn* matrix (over $\mathbf{F}_{2^8}$) : $\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$

Objective: using long codes for better diffusion

Idea Multiplying the whole state by a (dense) matrix

⇒ Complete diffusion at every round
⇒ More active S-boxes on average

Example: SHARK (64-bit block, 8×8 MDS matrix) (Rijmen & al., 1996)

Goal

1. Finding codes with good parameters, *e.g.* $[32, 16, d]$ with d close to 17
2. Finding efficient encoders
3. ⇒ Working with a small field, *e.g.* $\mathbf{F}_{2^4}$

Objective: long codes over $\mathbf{F}_{2^4}$

- We want $[32, 16, d]_{\mathbf{F}_{2^4}}$ codes with d maximum
- From the MDS conjecture, we cannot have MDS codes longer than $2^4 + 1 = 17$
- $\Rightarrow$ MDS codes not possible
- $\Rightarrow$ Use *algebraic-geometry* (AG) codes instead!

A $[32, 16, 15]_{\mathbf{F}_{2^4}}$ hyperelliptic code

The generating matrix of an AG code is built by evaluating well-chosen bivariate polynomials on points of an algebraic curve

Example:

- Take the 16 polynomials $(1, x, x^2, y, x^3, xy, x^4, x^2y, x^5, x^3y, x^6, x^4y, x^7, x^5y, x^8, x^6y)$
- Evaluate each of them on the 32 points (α, β) of the curve of genus $g = 2$ defined on $\mathbf{F}_{2^4}$ by $\alpha^5 = \beta^2 + \beta$ in some order
- $\Rightarrow$ Forms a 16×32 generating matrix of a $[32, 16, 15]_{\mathbf{F}_{2^4}}$ code (where 15 = 32 - 16 + 1 - g)
- $\Rightarrow$ From $(I_{16}\ D)$, deduce a diffusion matrix D of branch number 15
- Bonus: $D \cdot D^t = I_{16}$

A $[32, 16, 15]_{\mathbf{F}_{2^4}}$ hyperelliptic code

The generating matrix of an AG code is built by evaluating well-chosen bivariate polynomials on points of an algebraic curve

Example:

- Take the 16 polynomials $(1, x, x^2, y, x^3, xy, x^4, x^2y, x^5, x^3y, x^6, x^4y, x^7, x^5y, x^8, x^6y)$
- Evaluate each of them on the 32 points (α, β) of the curve of genus $g = 2$ defined on $\mathbf{F}_{2^4}$ by $\alpha^5 = \beta^2 + \beta$ in some order
- $\Rightarrow$ Forms a 16×32 generating matrix of a $[32, 16, 15]_{\mathbf{F}_{2^4}}$ code (where 15 = 32 - 16 + 1 - g)
- $\Rightarrow$ From $(I_{16}\ D)$, deduce a diffusion matrix D of branch number 15
- Bonus: $D \cdot D^t = I_{16}$

A $[32, 16, 15]_{\mathbf{F}_{2^4}}$ hyperelliptic code

The generating matrix of an AG code is built by evaluating well-chosen bivariate polynomials on points of an algebraic curve

Example:

- Take the 16 polynomials $(1, x, x^2, y, x^3, xy, x^4, x^2y, x^5, x^3y, x^6, x^4y, x^7, x^5y, x^8, x^6y)$
- Evaluate each of them on the 32 points (α, β) of the curve of genus $g = 2$ defined on $\mathbf{F}_{2^4}$ by $\alpha^5 = \beta^2 + \beta$ in some order
- $\Rightarrow$ Forms a 16×32 generating matrix of a $[32, 16, 15]_{\mathbf{F}_{2^4}}$ code (where $15 = 32 - 16 + 1 - g$)
- $\Rightarrow$ From $(I_{16}\ D)$, deduce a diffusion matrix D of branch number 15
- Bonus: $D \cdot D^t = I_{16}$

A $[32, 16, 15]_{\mathbf{F}_{2^4}}$ hyperelliptic code

The generating matrix of an AG code is built by evaluating well-chosen bivariate polynomials on points of an algebraic curve

Example:

- Take the 16 polynomials $(1, x, x^2, y, x^3, xy, x^4, x^2y, x^5, x^3y, x^6, x^4y, x^7, x^5y, x^8, x^6y)$
- Evaluate each of them on the 32 points (α, β) of the curve of genus $g = 2$ defined on $\mathbf{F}_{2^4}$ by $\alpha^5 = \beta^2 + \beta$ in some order
- $\Rightarrow$ Forms a 16×32 generating matrix of a $[32, 16, 15]_{\mathbf{F}_{2^4}}$ code (where 15 = 32 - 16 + 1 - g)
- $\Rightarrow$ From $(I_{16}\ D)$, deduce a diffusion matrix D of branch number 15
- Bonus: $D \cdot D^t = I_{16}$

A $[32, 16, 15]_{\mathbf{F}_{2^4}}$ hyperelliptic code

The generating matrix of an AG code is built by evaluating well-chosen bivariate polynomials on points of an algebraic curve

Example:

- Take the 16 polynomials $(1, x, x^2, y, x^3, xy, x^4, x^2y, x^5, x^3y, x^6, x^4y, x^7, x^5y, x^8, x^6y)$
- Evaluate each of them on the 32 points (α, β) of the curve of genus $g = 2$ defined on $\mathbf{F}_{2^4}$ by $\alpha^5 = \beta^2 + \beta$ in some order
- $\Rightarrow$ Forms a 16×32 generating matrix of a $[32, 16, 15]_{\mathbf{F}_{2^4}}$ code (where $15 = 32 - 16 + 1 - g$)
- $\Rightarrow$ From $(I_{16}\ D)$, deduce a diffusion matrix D of branch number 15
- Bonus: $D \cdot D^t = I_{16}$

A $[32, 16, 15]_{\mathbf{F}_{2^4}}$ hyperelliptic code

The generating matrix of an AG code is built by evaluating well-chosen bivariate polynomials on points of an algebraic curve

Example:

- Take the 16 polynomials $(1, x, x^2, y, x^3, xy, x^4, x^2y, x^5, x^3y, x^6, x^4y, x^7, x^5y, x^8, x^6y)$
- Evaluate each of them on the 32 points (α, β) of the curve of genus $g = 2$ defined on $\mathbf{F}_{2^4}$ by $\alpha^5 = \beta^2 + \beta$ in some order
- $\Rightarrow$ Forms a 16×32 generating matrix of a $[32, 16, 15]_{\mathbf{F}_{2^4}}$ code (where 15 = 32 - 16 + 1 - g)
- $\Rightarrow$ From $(I_{16}\ D)$, deduce a diffusion matrix D of branch number 15
- Bonus: $D \cdot D^t = I_{16}$

Properties of AG codes

- The maximum length is the number $\#\mathscr{X}$ of points on the curve
- There are $\binom{\#\mathscr{X}}{n} \cdot n!$ equivalent codes of length n
- Curve with small genus $\Rightarrow$ code with high minimum distance
- $\Rightarrow$ Tradeoff length vs. minimum distance

How to implement matrix multiplication?

- Explicit field arithmetic
- Table implementation
- Bitsliced implementation
- SIMD “Vector” implementation

We propose two vector algorithms

1. A “generic” one
2. One that is more efficient for some matrices

Algorithm 2: Example 1

$$\begin{pmatrix} 1 & 0 & 2 & 2 \\ 3 & 1 & 2 & 3 \\ 2 & 3 & 3 & 2 \\ 0 & 2 & 3 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$=$$

$$\begin{pmatrix} x_0 \\ x_1 \\ 0 \\ x_3 \end{pmatrix} + 2 \cdot \begin{pmatrix} x_2 \\ x_2 \\ x_0 \\ x_1 \end{pmatrix} + 2 \cdot \begin{pmatrix} x_3 \\ 0 \\ x_3 \\ 0 \end{pmatrix} + 3 \cdot \begin{pmatrix} 0 \\ x_0 \\ x_1 \\ x_2 \end{pmatrix} + 3 \cdot \begin{pmatrix} 0 \\ x_3 \\ x_2 \\ 0 \end{pmatrix}$$

The shuffles and constant multiplications can be computed with a single `pshufb` instruction

Algorithm 2: Example 2

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$= \begin{cases} a = e = i = \alpha \\ b = d = g = h = \beta \\ c = \gamma \\ f = \delta \end{cases}$$

$$\alpha \cdot \begin{pmatrix} x_0 \\ 0 \\ 0 \end{pmatrix} + \beta \cdot \begin{pmatrix} x_1 \\ 0 \\ 0 \end{pmatrix} + \gamma \cdot \begin{pmatrix} x_2 \\ 0 \\ 0 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ x_0 \\ 0 \end{pmatrix} + \alpha \cdot \begin{pmatrix} 0 \\ x_1 \\ 0 \end{pmatrix} + \delta \cdot \begin{pmatrix} 0 \\ x_2 \\ 0 \end{pmatrix} +$$

$$\beta \cdot \begin{pmatrix} 0 \\ 0 \\ x_0 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ 0 \\ x_1 \end{pmatrix} + \alpha \cdot \begin{pmatrix} 0 \\ 0 \\ x_2 \end{pmatrix}$$

Algorithm 2: Example 2

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$= \begin{cases} a = e = i = \alpha \\ b = d = g = h = \beta \\ c = \gamma \\ f = \delta \end{cases}$$

$$\alpha \cdot \begin{pmatrix} x_0 \\ 0 \\ 0 \end{pmatrix} + \alpha \cdot \begin{pmatrix} 0 \\ x_1 \\ 0 \end{pmatrix} + \alpha \cdot \begin{pmatrix} 0 \\ 0 \\ x_2 \end{pmatrix} + \beta \cdot \begin{pmatrix} x_1 \\ 0 \\ 0 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ x_0 \\ 0 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ 0 \\ x_0 \end{pmatrix} +$$

$$\beta \cdot \begin{pmatrix} 0 \\ 0 \\ x_1 \end{pmatrix} + \gamma \cdot \begin{pmatrix} x_2 \\ 0 \\ 0 \end{pmatrix} + \delta \cdot \begin{pmatrix} 0 \\ x_2 \\ 0 \end{pmatrix}$$

Algorithm 2: Example 2

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$= \begin{cases} a = e = i = \alpha \\ b = d = g = h = \beta \\ c = \gamma \\ f = \delta \end{cases}$$

$$\alpha \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} + \beta \cdot \begin{pmatrix} x_1 \\ x_0 \\ x_0 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ 0 \\ x_1 \end{pmatrix} + \gamma \cdot \begin{pmatrix} x_2 \\ 0 \\ 0 \end{pmatrix} + \delta \cdot \begin{pmatrix} 0 \\ x_2 \\ 0 \end{pmatrix}$$

$\longleftrightarrow$

Algorithm 2: cost function for a matrix

The number of `pshufb` instructions depends on:

1. The number of constants > 1
2. The number of shuffles

This is easy to compute by:

1. $\Rightarrow$ Taking a look at the coefficients
2. $\Rightarrow$ For each constant > 1, this is the max. occurrence of the constant per line

Cost function: back to example 2

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$= \begin{cases} a = e = i = \alpha \\ b = d = g = h = \beta \\ c = \gamma \\ f = \delta \end{cases}$$

$$\alpha \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} + \beta \cdot \begin{pmatrix} x_1 \\ x_0 \\ x_0 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ 0 \\ x_1 \end{pmatrix} + \gamma \cdot \begin{pmatrix} x_2 \\ 0 \\ 0 \end{pmatrix} + \delta \cdot \begin{pmatrix} 0 \\ x_2 \\ 0 \end{pmatrix}$$

$\Rightarrow$ cost of $9 = (1+1) + (1+2) + (1+1) + (1+1)$

Low-cost matrices

Observation

$D = (\alpha, \pi_0(\alpha), \pi_1(\alpha), \ldots, \pi_p(\alpha))^t$ with π_i permutations $\Rightarrow$ criterion #2 is minimized for D

- Particular case: circulant matrices
  - Cost $\approx$ 30 for dense matrices of dim. 16
- Low cost if all the rows derive from a few ones

How to find permutations?

- We want $M = (I \ D)$ s.t. D is circulant (or close enough)
- $\Rightarrow$ Use automorphisms of the code

Group of automorphisms *Aut* of a code

Aut($\mathscr{C}$) with $\mathscr{C}$ of length n is a subgroup of $\mathfrak{S}_n$ s.t.
$\pi \in \mathrm{Aut}(\mathscr{C}) \Rightarrow (c \in \mathscr{C} \Rightarrow \pi(c) \in \mathscr{C})$

$\Rightarrow$ For AG codes, can be deduced from automorphisms of the curve

Back to the hyperelliptic code: automorphisms

- Automorphisms of the curve $\alpha^5 = \beta^2 + \beta$ have two generators (Duursma, 1999)
  - $\pi_0 : \mathbf{F}_{2^4}^2 \to \mathbf{F}_{2^4}^2,\ (x, y) \mapsto (\zeta x, y)$ with $\zeta^5 = 1$
  - $\pi_{1_{(\alpha,\beta)}} : \mathbf{F}_{2^4}^2 \to \mathbf{F}_{2^4}^2,\ (x, y) \mapsto (x + \alpha, y + \alpha^8 x^2 + \alpha^4 x + \beta^4)$ with (α, β) an affine point of the curve
  - $\Rightarrow$ span a group of order 160
  - $\Rightarrow$ Automorphisms of the code
- Can add the Frobenius mapping $F : \mathbf{F}_{2^4}^2 \to \mathbf{F}_{2^4}^2,\ (x, y) \mapsto (x^2, y^2)$
  - $\Rightarrow$ Not an automorphism of the code

Automorphisms: example

- $\sigma = F \circ \sigma_2 \circ \sigma_1$ with $\begin{cases} \sigma_1 : (x,y) \mapsto (x+1, y+x^2+x+7) \\ \sigma_2 : (x,y) \mapsto (12x, y) \end{cases}$
- Only σ^4 is an automorphism of the code
- Can be used to define a matrix $(I_{16}\ D)$ with D of the form

$$(\mathbf{a}^0, \mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3, \sigma^4(\mathbf{a}^0), \sigma^4(\mathbf{a}^1), \sigma^4(\mathbf{a}^2), \sigma^4(\mathbf{a}^3), \mathbf{a}^8, \mathbf{a}^9, \mathbf{a}^{10}, \mathbf{a}^{11}, \sigma^4(\mathbf{a}^8), \sigma^4(\mathbf{a}^9), \sigma^4(\mathbf{a}^{10}), \sigma^4(\mathbf{a}^{11}))^t$$

- $\Rightarrow$ The matrix can be compressed in 8 rows
- Cost of 52

Complete coloured compressed matrix

$$
\begin{pmatrix}
5 & 2 & 1 & 3 & 8 & 5 & 1 & 5 & 12 & 10 & 14 & 6 & 7 & 11 & 4 & 11 \\
2 & 2 & 4 & 1 & 5 & 12 & 2 & 1 & 9 & 15 & 8 & 11 & 7 & 6 & 9 & 3 \\
1 & 4 & 4 & 3 & 1 & 2 & 15 & 4 & 5 & 13 & 10 & 12 & 9 & 6 & 7 & 13 \\
3 & 1 & 3 & 3 & 5 & 1 & 4 & 10 & 14 & 2 & 14 & 8 & 15 & 13 & 7 & 6 \\
8 & 5 & 1 & 5 & 5 & 2 & 1 & 3 & 7 & 11 & 4 & 11 & 12 & 10 & 14 & 6 \\
5 & 12 & 2 & 1 & 2 & 2 & 4 & 1 & 7 & 6 & 9 & 3 & 9 & 15 & 8 & 11 \\
1 & 2 & 15 & 4 & 1 & 4 & 4 & 3 & 9 & 6 & 7 & 13 & 5 & 13 & 10 & 12 \\
5 & 1 & 4 & 10 & 3 & 1 & 3 & 3 & 15 & 13 & 7 & 6 & 14 & 2 & 14 & 8 \\
12 & 9 & 5 & 14 & 7 & 7 & 9 & 15 & 7 & 6 & 11 & 3 & 15 & 5 & 13 & 7 \\
10 & 15 & 13 & 2 & 11 & 6 & 6 & 13 & 6 & 6 & 7 & 9 & 5 & 10 & 2 & 14 \\
14 & 8 & 10 & 14 & 4 & 9 & 7 & 7 & 11 & 7 & 7 & 6 & 13 & 2 & 8 & 4 \\
6 & 11 & 12 & 8 & 11 & 3 & 13 & 6 & 3 & 9 & 6 & 6 & 7 & 14 & 4 & 12 \\
7 & 7 & 9 & 15 & 12 & 9 & 5 & 14 & 15 & 5 & 13 & 7 & 7 & 6 & 11 & 3 \\
11 & 6 & 6 & 13 & 10 & 15 & 13 & 2 & 5 & 10 & 2 & 14 & 6 & 6 & 7 & 9 \\
4 & 9 & 7 & 7 & 14 & 8 & 10 & 14 & 13 & 2 & 8 & 4 & 11 & 7 & 7 & 6 \\
11 & 3 & 13 & 6 & 6 & 11 & 12 & 8 & 7 & 14 & 4 & 12 & 3 & 9 & 6 & 6
\end{pmatrix}
$$

Sorry, colour-blind folks...

Hyperelliptic code: random generating matrices

- Search for point orders giving low-cost matrices
- Search space of size $32! \approx 2^{117{,}7}$
- Many matrices of cost 43 found

cost	#matrices	cumulative #matrices	cumulative proportion of the search space
43	146 482	146 482	0.00000053
44	73 220	219 702	0.00000080
45	218 542	438 244	0.0000016
46	879 557	1 317 801	0.0000048
47	1 978 159	3 295 960	0.000012
48	5 559 814	8 855 774	0.000032
49	21 512 707	30 368 481	0.00011
50	93 289 020	123 657 501	0.00045
51	356 848 829	480 506 330	0.0017
52	1 282 233 658	1 762 739 988	0.0064
53	3 534 412 567	5 297 152 555	0.019
54	8 141 274 412	13 438 426 967	0.049
55	15 433 896 914	28 872 323 881	0.11
56	24 837 735 898	53 710 059 779	0.20
57	33 794 051 687	87 504 111 466	0.32
58	38 971 338 149	126 475 449 615	0.46
59	38 629 339 524	165 104 789 139	0.60

Block ciphers with a SHARK structure

- We have matrices of $\mathscr{M}_{16}(\mathbf{F}_{2^4})$ of branch number 15
- Can be defined as well over $\mathbf{F}_{2^8} \cong \mathbf{F}_{2^4}[t]/p(t)$ (same b.n.)
  - (All computations done in $\mathbf{F}_{2^4}$: $\alpha \cdot (at+b) = (\alpha at + \alpha b)$)
- How many rounds to do?

Max. d.p./l.b. of a single path

	2 rd.	4 rd.	6 rd.	8 rd.
64 bits (best 4-bit Sbox)	2^{-30}	2^{-60}	2^{-90}	2^{-120}
128 bits (best 8-bit Sbox)	2^{-90}	2^{-180}	2^{-270}	2^{-360}
128 bits (faster 8-bit Sbox)	2^{-75}	2^{-150}	2^{-225}	2^{-300}

Performance

Performance of software implementations of 64 and 128-bit (best S-box) SHARK structures, in cycles per byte for one-block messages

		64-bit Block		128-bit Block	
Processor architecture	# rounds	Alg. 1	Alg. 2	Alg. 1	Alg. 2
S. Bridge (E5-2650)	6	50 (45.5)	33 (24.2)	58 (52.3)	32.7 (26.5)
	8	66.5 (60.2)	44.5 (31.9)	76.8 (69.6)	43.8 (35.7)
S. Bridge (E5-2609)	6	72.3 (63.7)	45.3 (33.2)	79.8 (75.6)	47.1 (36.8)
	8	95.3 (84.7)	63.3 (45.6)	106.6 (97.1)	62.1 (50.3)
Westmere (E5649)	6	84.7	46	84.5	47
	8	111.3	59.8	111	61.9

Further applications

- Conversion to stream-cipher with a LEX leak (Biryukov, 2007)
  - 2× speedup from the 8 rd. version with 4-word leak
  - 3× with 6-word leak! ⇝ 12 cpb on E5-2650
- Good "random" matrices for ASASA schemes (Biryukov & al., 2014)?