Optimizing Implementations of Linear Layers

Zejun Xiang, Xiangyong Zeng, Da Lin, Zhenzhen Bao, Shasha Zhang

Nov. 09, 2020

Lightweight Cryptography

-Requirements

- Circuit size
- Latency
- Throughput
- ……

-Lightweight Components:

- Lightweight MDS or near-MDS matrix
- Sbox?

-Primitives:

- SIMON，SPECK
- PRESENT, RECTANGLE
- LED，MIDORI
- ……

-Optimization (sbox)

- Gladman (Serpent)
- SAT based method (Stoffelen, FSE 2016)
- LIGHTER (Jean, FSE 2017)
- ……

Optimization---Linear layer

Local optimization

- optimize the multiplication (finite field or matrix-vector multiplication) of each entry of a matrix.
- **Method**:using different basis
  reuse intermediate values
  LIGHTER
- **Cost** = fixed cost + multiplication
- **AES** = 96 + ()

Global optimization

- treat the linear layer as a binary matrix and find the minimal number of xor operations to implement the matrix
- **Method**: Paar1 and Paar2
  BP (Boyar and Peralta)
  LIGHTER
  variants……
- **Cost** = returned by the algorithms
- **AES** = 92

Metrics

Consider a matrix to be implemented, $M = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$

$$GF(2): \begin{cases} y_1 = x_1 \oplus x_2 \oplus x_3 \\ y_2 = x_1 \oplus x_2 \oplus x_4 \\ y_3 = x_1 \oplus x_3 \\ y_4 = x_2 \oplus x_3 \end{cases}$$

D-xor

Counting the number of 1's within a binary matrix.

D-xor(M) = 6

Metrics

G-xor → Counting the minimal number of operations $x_i = x_m \oplus x_n$ implementing the matrix.

Shortest Linear Straight-Line Program, NP-hard

$$
\begin{aligned}
t_1 &= x_1 \oplus x_3 \quad (y_3)\\
t_2 &= x_2 \oplus x_3 \quad (y_4)\\
t_3 &= x_1 \oplus x_2\\
t_4 &= t_3 \oplus x_3 \quad (y_1)\\
t_5 &= t_3 \oplus x_4 \quad (y_2)
\end{aligned}
$$

D-xor = 6 → G-xor = 5

Metrics

S-xor

Counting the nominal number of operations $x_i = x_i \oplus x_j$ implementing the matrix.

Optimal pivoting in Gauss-Jordan elimination

$$
\begin{array}{ll}
x_4 = x_4 \oplus x_2 & x_4 \oplus x_2 \\
x_4 = x_4 \oplus x_1 & x_4 \oplus x_2 \oplus x_1 = y_2 \\
x_1 = x_1 \oplus x_3 & x_1 \oplus x_3 = y_3 \\
x_3 = x_3 \oplus x_2 & x_3 \oplus x_2 = y_4 \\
x_2 = x_2 \oplus x_1 & x_2 \oplus x_1 \oplus x_3 = y_1
\end{array}
$$

S-xor = 5

Extra advantage of S-xor

$$t_1 = x_1 \oplus x_3 \quad (y_3)$$
$$t_2 = x_2 \oplus x_3 \quad (y_4)$$
$$t_3 = x_1 \oplus x_2$$
$$t_4 = t_3 \oplus x_3 \quad (y_1)$$
$$t_5 = t_3 \oplus x_4 \quad (y_2)$$

Bit-sliced software implementation

$$t_1 = x_1 \hat{} x_3 \quad (y_3)$$
$$t_2 = x_2 \hat{} x_3 \quad (y_4)$$
$$t_3 = x_1 \hat{} x_2$$
$$t_4 = t_3 \hat{} x_3 \quad (y_1)$$
$$t_5 = t_3 \hat{} x_4 \quad (y_2)$$

Xor destructive

$t_1 = x_1 \hat{} x_3$

```
movl  x1, t1
xorl  x3, t1
```

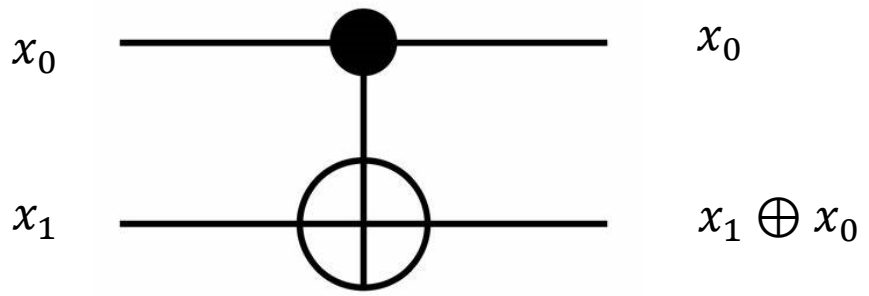
$$t_1 = x_1 \hat{} x_3 \quad (y_3)$$
$$t_2 = x_2 \hat{} x_3 \quad (y_4)$$
$$x_1 = x_1 \hat{} x_2$$
$$x_3 = x_1 \hat{} x_3 \quad (y_1)$$
$$x_4 = x_1 \hat{} x_4 \quad (y_2)$$

Extra advantage of S-xor

$x_4 = x_4 \hat{} x_2$	$x_4 \oplus x_2$
$x_4 = x_4 \hat{} x_1$	$x_4 \oplus x_2 \oplus x_1 = y_2$
$x_1 = x_1 \hat{} x_3$	$x_1 \oplus x_3 = y_3$
$x_3 = x_3 \hat{} x_2$	$x_3 \oplus x_2 = y_4$
$x_2 = x_2 \hat{} x_1$	$x_2 \oplus x_1 \oplus x_3 = y_1$

```
xorl    x2, x4
xorl    x1, x4
xorl    x3, x1
xorl    x2, x3
xorl    x1, x2
```

Quantum Implementation



CNOT gate

Elementary operation and elementary matrix

Type-1 Interchange two rows (columns)

$$E(1 \leftrightarrow 2) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Type-2 Multiply a row (column) with a nonzero number

$k \in F_2$

$$E(1 + 2 * k) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Type-3 Add a row (column) to another one multiplied by a nonzero number

$k \in F_2$

$$E(1 + 2 * k) = \begin{pmatrix} 1 & k & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Cost of elementary matrix

Type-1 $E(1 \leftrightarrow 2) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Type-3 $E(1 + 2) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 = x_2 \\ y_2 = x_1 \\ y_3 = x_3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 = x_1 \oplus x_2 \\ y_2 = x_2 \\ y_3 = x_3 \end{pmatrix}$$

Cost = 0

Cost = 1 (S-xor)

Matrix Decomposition

Theorem

Any invertible matrix can be transformed into an identity matrix using elementary row and/or column operations. Thus, any invertible matrix can ben decomposed as a product of elementary matrices.

F_2

Corollary

Any matrix in $GL(2, F_2)$ can be transformed into an identity matrix by applying a series of type-1 and type-3 elementary row and/or column operations. Thus, any matrix in $GL(2, F_2)$ can ben decomposed as a product of type-1 and type-3 elementary matrices.

Matrix decompositions

1. Elementary row operation based matrix decomposition

   Gaussian Elimination

2. Elementary column operation based matrix decomposition

3. Hybrid elementary operation based matrix decomposition

Pick the elementary operation which minimize the most number of 1's in the given matrix

- Tie break: the first one VS random
- Increase the number of 1's –> infinite loop -> 1 or 2

Matrix Decomposition

Property

$$E(i+j)E(k \leftrightarrow l) = E(k \leftrightarrow l)E\left(f_{k,l}(i) + f_{k,l}(j)\right),$$
$$E(k \leftrightarrow l)E(i+j) = E\left(f_{k,l}(i) + f_{k,l}(j)\right)E(k \leftrightarrow l),$$

where

$$f_{k,l}(x) = \begin{cases} k, & \text{if } x = l, \\ l, & \text{if } x = k, \\ x, & \text{else.} \end{cases}$$

Theorem

Any matrix M in $GL(2, F_2)$ can ben decomposed as:

$$M = E(i_t + j_t) \cdots E(i_1 + j_1)E(i'_s \leftrightarrow j'_s) \cdots E(i'_1 \leftrightarrow j'_s)$$

↳ Cost(M) = t

Properties of matrix multiplication

Property

Let $E(i \leftrightarrow j)$ and E(i+j) denote a type-1 and type-3 elementary matrices in $GL(2, F_2)$ respectively, then the following equations hold.

R1 $E(k+i)E(k+j)E(i+j) = E(i+j)E(k+i)$

R2 $E(i+k)E(k+j)E(i+j) = E(k+j)E(i+k)$

R3 $E(i+k)E(j+k)E(i+j) = E(i+j)E(j+k)$

R4 $E(j+k)E(i+k)E(i+j) = E(i+j)E(j+k)$

R5 $E(k+j)E(k+i)E(i+j) = E(i+j)E(k+i)$

R6 $E(k+j)E(i+k)E(i+j) = E(i+j)E(k+j)$

R7 $E(j+i)E(i+j) = E(i \leftrightarrow j)E(j+i)$

Properties of matrix multiplication

$$E(k+i)E(k+j)E(i+j) = E(i+j)E(k+i)$$

Consider $E(k+i)E(k+j)E(i+j)x$, where $x = (x_1, x_2, \dots, x_n)$.
Only x_i, x_j and x_k are involved in the computation.

x_i	x_j	x_k
$x_i \oplus x_j$	x_j	x_k
$x_i \oplus x_j$	x_j	$x_k \oplus x_j$
$x_i \oplus x_j$	x_j	$x_k \oplus x_i$

equivalent

x_i	x_j	x_k
x_i	x_j	$x_k \oplus x_i$
$x_i \oplus x_j$	x_j	$x_k \oplus x_i$

Properties of matrix multiplication

matrix multiplication does NOT generally satisfy the **commutative law.**

special cases

Property

Let i, j, k, l be integers and $i \neq j \neq k \neq l$, then we have

1 $E(k+l)E(i+j) = E(i+j)E(k+l)$

2 $E(i+j)E(k+j) = E(k+j)E(i+j)$

3 $E(i+j)E(i+k) = E(i+k)E(i+j)$

Example

cost = 4

$$M = E(3+2)E(3+4)E(3+1)E(2+1)$$

commutative

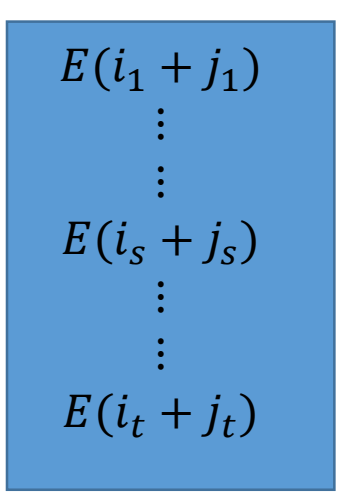
$$M = E(3+4)E(3+2)E(3+1)E(2+1)$$

reduction (R1)

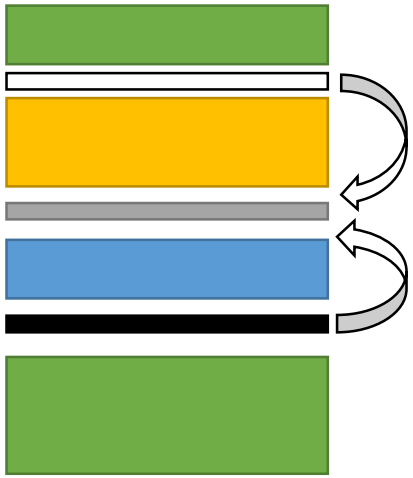
cost = 3

$$M = E(3+4)E(2+1)\,E(3+2)$$

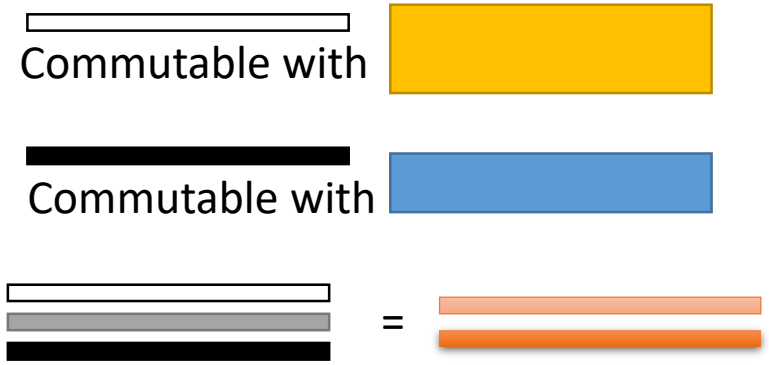
Reduction algorithm



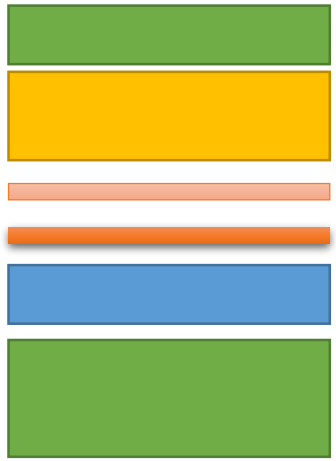
A given matrix decomposition



Identify possible reduction

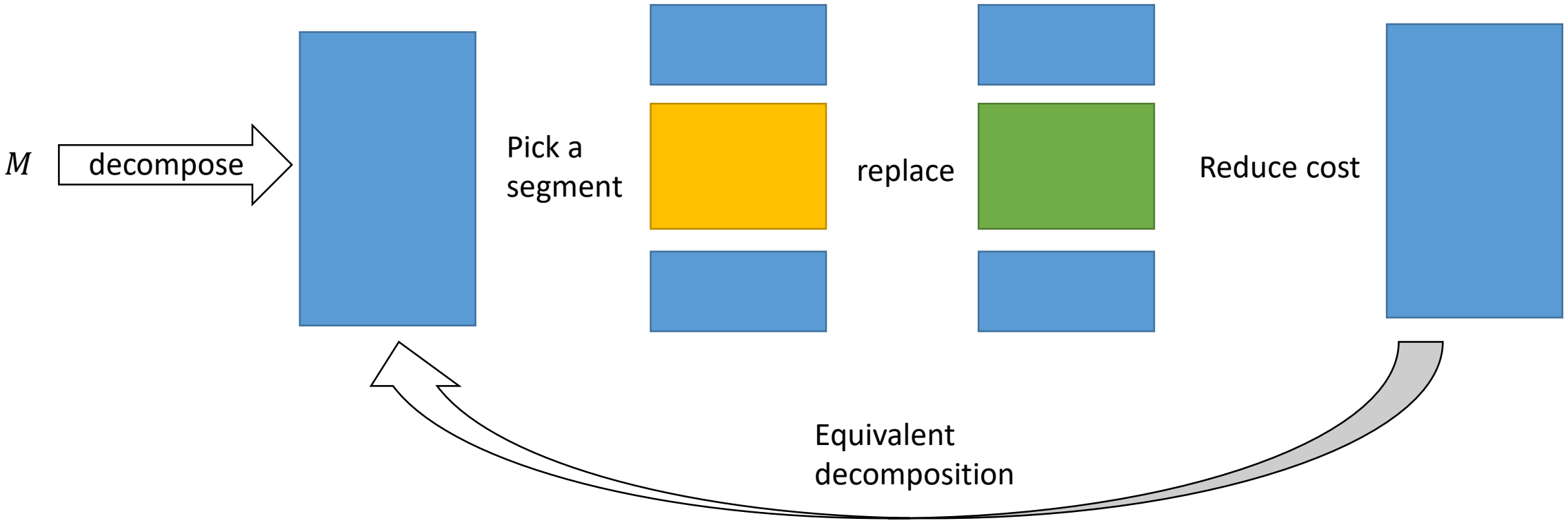


Conditions



Reduction

Search algorithm



Applications

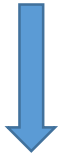
Table 1: Implementation cost of cipher matrices under different optimization heuristics.

Cipher	Size	[KLSW17][1]	[KLSW17][2]	[KLSW17][3]	[BFI19]	This paper
Fox mu8 [JV04]	64	611	-	594	592	-
Grostl [GKM+09]	64	493	-	475	460	-
Khazad [BR00b]	64	488	-	507	492	366
Whirlpool [BR00c]	64	481	-	465	464	-
Aes[4] [DR02]	32	108	108	97	95	92
Anubis [BR00a]	32	122	121	113	102	99
Clefia M_0 [SSA+07]	32	121	121	106	102	98
Clefia M_1 [SSA+07]	32	121	121	111	110	103
Fox mu4 [JV04]	32	144	143	137	131	136
Twofish [SKW+98]	32	151	149	129	125	111
Joltik [JNP14]	16	52	48	48	47	44
SmallScale Aes [CMR05]	16	54	54	47	45	43
Whirlwind M_0 [BNN+10]	32	218	218	212	210	183
Whirlwind M_1 [BNN+10]	32	246	244	235	234	190
Qarma128 [Ava17]	32	48	48	48	48	48
Midori [BBI+15]	16	24	24	24	24	24
Prince M_0, M_1 [BCG+12]	16	24	24	24	24	24
Pride$L_0 - L_3$ [ADK+14]	16	24	24	24	24	24
Qarma64 [Ava17]	16	24	24	24	24	24
Skinny64 [BJK+16]	16	12	12	12	12	12

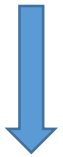
[1] This column presents the results in [KLSW17] obtained by Paar1's heuristic.
[2] This column presents the results in [KLSW17] obtained by Paar2's heuristic.
[3] This column presents the results in [KLSW17] obtained by Boyar-Peralta's heuristic.
[4] [Max19] presents a 92-Xor implementation of AES.

On Inverse Matrices

$$M = E_1 E_2 \cdots E_n \qquad M^{-1} = E_n^{-1} E_{n-1}^{-1} \cdots E_1^{-1}$$



the cost of the inverse of M = the cost of M



The inverse of AES MixColumns can ben implemented using 92 xor's.

Thanks!