Improved Heuristics for Short Linear Programs

Thomas Peyrin    Quan Quan Tan

Nanyang Technological University

CHES 2020
Contributions of this paper:

A new algorithm that finds good implementations of linear systems, to reduce the number of XOR gates/operations.

Our algorithm performs better than the state-of-the-art (Paar and Boyar-Peralta algorithms), we tested on existing and also random matrices.
Diffusion Matrices

Figure 1: Figure inspired from [Jea16]

\[
\begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{bmatrix}
\begin{bmatrix}
w_0 \\
w_1 \\
w_2 \\
w_3
\end{bmatrix}
= \begin{bmatrix}
2 \cdot w_0 \oplus 3 \cdot w_1 \oplus w_2 \oplus w_3 \\
w_0 \oplus 2 \cdot w_1 \oplus 3 \cdot w_2 \oplus w_3 \\
w_0 \oplus w_1 \oplus 2 \cdot w_2 \oplus 3 \cdot w_3 \\
3 \cdot w_0 \oplus w_1 \oplus w_2 \oplus 2 \cdot w_3
\end{bmatrix},
\] 
\(w_i \in GF(2^8)\)
Diffusion Matrices

Figure 1: Figure inspired from [Jea16]

\[
\begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{bmatrix}
\begin{bmatrix}
w_0 \\
w_1 \\
w_2 \\
w_3
\end{bmatrix}
= \begin{bmatrix}
2 \cdot w_0 \oplus 3 \cdot w_1 \oplus w_2 \oplus w_3 \\
w_0 \oplus 2 \cdot w_1 \oplus 3 \cdot w_2 \oplus w_3 \\
w_0 \oplus w_1 \oplus 2 \cdot w_2 \oplus 3 \cdot w_3 \\
3 \cdot w_0 \oplus w_1 \oplus w_2 \oplus 2 \cdot w_3
\end{bmatrix},
w_i \in GF(2^8)
\]
From $GF(2^n)$ to $GF(2)$

Multiplication by a fixed element in $GF(2^n)$ can be replaced by a $n \times n$ binary matrix multiplication.

$$w_0 = x_7x_6x_5x_4x_3x_2x_1x_0$$
irreducible polynomial $= p^8 + p^4 + p^3 + p + 1$

$$3 \times w_0 = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
x_7 \\
x_6 \\
x_5 \\
x_4 \\
x_3 \\
x_2 \\
x_1 \\
x_0
\end{bmatrix}$$
From $GF(2^n)$ to $GF(2)$

Multiplication by a fixed element in $GF(2^n)$ can be replaced by a $n \times n$ binary matrix multiplication.

\[ w_0 = x_7x_6x_5x_4x_3x_2x_1x_0 \]
irreducible polynomial = $p^8 + p^4 + p^3 + p + 1$

\[
3 \times w_0 = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_7 \\
x_6 \\
x_5 \\
x_4 \\
x_3 \\
x_2 \\
x_1 \\
x_0
\end{bmatrix}
\]
Problem
For any given fixed matrix M, how can we minimize the number of ‘⊕’ operations required to compute it?

- Naive counting (d-XOR). Compute each row individually.
- Sequential counting (g-XOR). Count the actual number of sequential XORs required for all the rows.

Example
\begin{align*}
y_0 &= x_0 \oplus x_1 \oplus x_2 \\
y_1 &= x_1 \oplus x_2 \oplus x_3 \\
t_0 &= x_1 \oplus x_2 \\
y_0 &= x_0 \oplus t_0 \\
y_1 &= t_0 \oplus x_3 \\
d-XOR &= 4 \\
g-XOR &= 3
\end{align*}
Idea: identify most frequent \((x_i, x_j)\) pairs and use an XOR to compute \(x_i \oplus x_j\). Repeat until done.

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1
\end{pmatrix}
\rightarrow
\begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}
\]

In the case of a tie,
- Choose the first one in lexicographical order (Paar1)
- Exhaust all equally frequent options (Paar2)
Past Works: Boyar-Peralta’s algorithm [BP10]

\[ e_1, e_2, ..., e_n, s_1, s_2, ..., s_k \]

\[ s_{k+1} = a \oplus b, \ a, b \in S \]

1. Choose \( s_{k+1} \) such that \( d_0 + d_1 + ... + d_n \) is minimized
2. L2-norm is used in an event of a tie
Past Works: Masoleh, Taha and Ashmawy’s algorithms [RTA18]

An alternative criteria: Shortest-Dist-First
Instead of using the L1-norm as the criteria, the criteria selects the pair that is able to reduce as many “nearest” targets as possible.

Suppose the current distance vector to the targets is [3, 4, 2, 2, 4, 5]

<table>
<thead>
<tr>
<th>Candidate’s distance</th>
<th>[2, 3, 2, 2, 3, 4]</th>
<th>[3, 4, 1, 1, 4, 5]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP criteria [BP10]</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SDF criteria [RTA18]</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Past Works: Masoleh, Taha and Ashmawy’s algorithms [RTA18]

An alternative criteria: Shortest-Dist-First
Instead of using the L1-norm as the criteria, the criteria selects the pair that is able to reduce as many “nearest” targets as possible.

Suppose the current distance vector to the targets is [3, 4, 2, 2, 4, 5]

<table>
<thead>
<tr>
<th>Candidate’s distance</th>
<th>[2,3,2,2,3,4]</th>
<th>[3,4,1,1,4,5]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP criteria [BP10]</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SDF criteria [RTA18]</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Limitations

- BP algorithm’s implementation follows a **lexicographical order** which did not consider all other pairs that are equally good.
- Paar1 suffers from the same issue as BP
- Paar2 exhaustively searches through all the possible pairs, which is costly for matrices that are relatively large

Solution

1. When we have more than one equally good pairs, randomly pick one of them.
2. Repeat the algorithm $k$ times and pick the best circuit.
Randomized Algorithms

Limitations
- BP algorithm’s implementation follows a **lexicographical order** which did not consider all other pairs that are equally good.
- Paar1 suffers from the same issue as BP
- Paar2 exhaustively searches through all the possible pairs, which is costly for matrices that are relatively large

Solution
1. When we have more than one equally good pairs, randomly pick one of them.
2. Repeat the algorithm $k$ times and pick the best circuit.
Our Criteria

Relaxing the criteria of having to reduce as many nearest targets as possible + maintaining the “main path” using L1-norm.

1. Shortlist all pairs such that at least one of the “nearest” targets is reduced
2. Apply L1-norm criteria to the remaining pairs. (A1)
3. If there is a tie, apply L2-norm criteria. (A2)

Suppose the current distance vector to the targets is \([3, 4, 2, 2, 4, 5]\)

<table>
<thead>
<tr>
<th>Candidate’s distance</th>
<th>([2, 3, 2, 2, 3, 5])</th>
<th>([3, 4, 1, 1, 4, 5])</th>
<th>([3, 3, 1, 2, 4, 4])</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP criteria [BP10]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDF criteria [RTA18]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Our criteria</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Our Criteria

Relaxing the criteria of having to reduce as many nearest targets as possible + maintaining the “main path” using L1-norm.

1. Shortlist all pairs such that at least one of the “nearest” targets is reduced
2. Apply L1-norm criteria to the remaining pairs. (A1)
3. If there is a tie, apply L2-norm criteria. (A2)

Suppose the current distance vector to the targets is [3, 4, 2, 2, 4, 5]

<table>
<thead>
<tr>
<th>Candidate's distance</th>
<th>[2,3,2,2,3,5]</th>
<th>[3,4,1,1,4,5]</th>
<th>[3,3,1,2,4,4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP criteria [BP10]</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>SDF criteria [RTA18]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Our criteria</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Rationale of our Criteria

Our guess: targets with high distance often cluster together
- High distance targets dominate the path from the start
- Targets with a lower distance can play a part in the path towards targets with a higher distance value.
Local Optimization

Given a circuit, find some ways to reduce the number of XORs.

Yosys [Wol]
Verilog RTL synthesis tool that does some optimization

Our local optimization techniques

:t1 = x0 ⊕ x1:
:t2 = x0 ⊕ x2:
:t3 = x2 ⊕ t1:
:t4 = x3 ⊕ t2:
:...
Results (Random Matrices [VSP18])

Figure 2: Average XOR count difference (A1 vs BP)

Figure 3: Average XOR count difference (A2 vs BP)

Our algorithms outperform BP for random matrices. The improvement is more obvious with the increase in size.
### Table 1: Percentage of best circuits obtained

<table>
<thead>
<tr>
<th>Matrix Size</th>
<th>BP</th>
<th>Paar1</th>
<th>RPaar1</th>
<th>SDF</th>
<th>RNBP</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 × 15</td>
<td>25.56</td>
<td>14.44</td>
<td>14.44</td>
<td>70.00</td>
<td>38.89</td>
<td>58.89</td>
<td>66.67</td>
</tr>
<tr>
<td>16 × 16</td>
<td>21.11</td>
<td>8.89</td>
<td>10.00</td>
<td>61.11</td>
<td>28.89</td>
<td>53.33</td>
<td>73.33</td>
</tr>
<tr>
<td>17 × 17</td>
<td>17.78</td>
<td>11.11</td>
<td>11.11</td>
<td>62.22</td>
<td>26.67</td>
<td>53.33</td>
<td>72.22</td>
</tr>
<tr>
<td>18 × 18</td>
<td>15.56</td>
<td>8.89</td>
<td>11.11</td>
<td>41.11</td>
<td>31.11</td>
<td>52.22</td>
<td>85.56</td>
</tr>
<tr>
<td>19 × 19</td>
<td>14.44</td>
<td>11.11</td>
<td>11.11</td>
<td>32.22</td>
<td>26.67</td>
<td>54.44</td>
<td>74.44</td>
</tr>
<tr>
<td>20 × 20</td>
<td>12.22</td>
<td>11.11</td>
<td>11.11</td>
<td>25.56</td>
<td>23.33</td>
<td>58.89</td>
<td>87.78</td>
</tr>
</tbody>
</table>
### Table 2: XOR count of $16 \times 16$ matrices

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{4,5}^{9,3}$</td>
<td>$(A_4, -, -)$</td>
<td>35</td>
<td>38</td>
<td>45</td>
<td>36</td>
<td>37</td>
<td>39</td>
<td>37</td>
</tr>
<tr>
<td>$M_{4,5}^{9,3}$</td>
<td>$(A_4^{-1})$</td>
<td>36</td>
<td>40</td>
<td>46</td>
<td>38</td>
<td>39</td>
<td>38</td>
<td>35</td>
</tr>
<tr>
<td>$M_{4,6}^{8,3}$</td>
<td>$(A_4, -, -)$</td>
<td>35</td>
<td>38</td>
<td>45</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>38</td>
</tr>
<tr>
<td>$M_{4,6}^{8,3}$</td>
<td>$(A_4^{-1})$</td>
<td>35</td>
<td>40</td>
<td>46</td>
<td>36</td>
<td>38</td>
<td>38</td>
<td>35</td>
</tr>
<tr>
<td>$M_{4,5}^{8,3}$</td>
<td>$(A_4^{-1}, A_4, A_4^{-2})$</td>
<td>36</td>
<td>40</td>
<td>47</td>
<td>40</td>
<td>39</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>$M_{4,4}^{9,4}$</td>
<td>$(A_4, -, -)$</td>
<td>39</td>
<td>41</td>
<td>47</td>
<td>41</td>
<td>40</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>$M_{4,4}^{9,3}$</td>
<td>$(A_4^{-1}, A_4, A_4^{-2})$</td>
<td>40</td>
<td>40</td>
<td>43</td>
<td>40</td>
<td>39</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>$M_{4,4}^{8,4}$</td>
<td>$(A_4, -, -)$</td>
<td>38</td>
<td>40</td>
<td>43</td>
<td>41</td>
<td>39</td>
<td>40</td>
<td>39</td>
</tr>
<tr>
<td>$M_{4,4}^{8,4}$</td>
<td>$(A_4, -, -)$</td>
<td>38</td>
<td>43</td>
<td>41</td>
<td>38</td>
<td>41</td>
<td>39</td>
<td>38</td>
</tr>
<tr>
<td>$M_{4,4}^{8,4''}$</td>
<td>$(A_4, -, -)$</td>
<td>37</td>
<td>40</td>
<td>43</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>39</td>
</tr>
<tr>
<td>$M_{4,3}^{9,5}$</td>
<td>$(A_4, -, -)$</td>
<td>41</td>
<td>40</td>
<td>43</td>
<td>41</td>
<td>40</td>
<td>41</td>
<td>40</td>
</tr>
<tr>
<td>$M_{4,4}^{9,5}$</td>
<td>$(A_4^{-1}, -, -)$</td>
<td>41</td>
<td>43</td>
<td>44</td>
<td>44</td>
<td>41</td>
<td>41</td>
<td>40</td>
</tr>
</tbody>
</table>
Table 3: XOR count of $32 \times 32$ matrices

<table>
<thead>
<tr>
<th>Matrix $M_{n,m}$</th>
<th>Instantiation $\alpha, \beta, \gamma$</th>
<th>Const. $\alpha, \beta, \gamma$</th>
<th>BP $\alpha, \beta, \gamma$</th>
<th>Paar2 $\alpha, \beta, \gamma$</th>
<th>RSDF $\alpha, \beta, \gamma$</th>
<th>RNBP $\alpha, \beta, \gamma$</th>
<th>A1 $\alpha, \beta, \gamma$</th>
<th>A2 $\alpha, \beta, \gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{4,3}^{9,3}$</td>
<td>$(A_8, - , -)$</td>
<td>67</td>
<td>74</td>
<td>88</td>
<td>74</td>
<td>67</td>
<td>77</td>
<td>69</td>
</tr>
<tr>
<td>$M_{4,5}^{9,3}$</td>
<td>$(A_8^{-1}, - , -)$</td>
<td>67</td>
<td>71</td>
<td>89</td>
<td>79</td>
<td>69</td>
<td>78</td>
<td>68</td>
</tr>
<tr>
<td>$M_{4,3}^{8,3}$</td>
<td>$(A_8, - , -)$</td>
<td>67</td>
<td>74</td>
<td>88</td>
<td>71</td>
<td>67</td>
<td>76</td>
<td>69</td>
</tr>
<tr>
<td>$M_{4,6}^{8,3}$</td>
<td>$(A_8^{-1}, - , -)$</td>
<td>67</td>
<td>71</td>
<td>89</td>
<td>78</td>
<td>69</td>
<td>78</td>
<td>68</td>
</tr>
<tr>
<td>$M_{4,5}^{8,3}$</td>
<td>$(A_8^{-1}, A_8, A_8^{-2})$</td>
<td>68</td>
<td>75</td>
<td>77</td>
<td>81</td>
<td>68</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>$M_{4,4}^{9,4}$</td>
<td>$(A_8, - , -)$</td>
<td>76</td>
<td>77</td>
<td>92</td>
<td>84</td>
<td>76</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td>$M_{4,4}^{9,3}$</td>
<td>$(A_8^{-1}, A_8, A_8^2)$</td>
<td>76</td>
<td>76</td>
<td>83</td>
<td>79</td>
<td>75</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td>$M_{4,4}^{8,4}$</td>
<td>$(A_8, - , -)$</td>
<td>70</td>
<td>72</td>
<td>74</td>
<td>77</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>$M_{4,4}^{8,4'}$</td>
<td>$(A_8, - , -)$</td>
<td>70</td>
<td>81</td>
<td>79</td>
<td>76</td>
<td>76</td>
<td>72</td>
<td>71</td>
</tr>
<tr>
<td>$M_{4,4}^{8,4''}$</td>
<td>$(A_8, - , -)$</td>
<td>69</td>
<td>72</td>
<td>85</td>
<td>77</td>
<td>69</td>
<td>76</td>
<td>70</td>
</tr>
<tr>
<td>$M_{4,3}^{9,5}$</td>
<td>$(A_8, - , -)$</td>
<td>77</td>
<td>76</td>
<td>86</td>
<td>82</td>
<td>76</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td>$M_{4,3}^{9,5}$</td>
<td>$(A_8^{-1}, - , -)$</td>
<td>77</td>
<td>79</td>
<td>86</td>
<td>85</td>
<td>77</td>
<td>77</td>
<td>77</td>
</tr>
</tbody>
</table>
## Results (AES)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AES MixCol</td>
<td>97 [KLSW17]</td>
<td>102</td>
<td>95</td>
<td>95</td>
<td>94</td>
</tr>
<tr>
<td>AES InvMixCol</td>
<td>155</td>
<td>162</td>
<td>153</td>
<td>153</td>
<td>152</td>
</tr>
</tbody>
</table>

Very recently, [Max19, XZL+20] further improved our result for AES matrix to 92 XORs
A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5.

However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large.

More techniques in local optimization may lead to even lower XOR count.

The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.
A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5.

However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large.

More techniques in local optimization may lead to even lower XOR count.

The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.
A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5.

However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large.

More techniques in local optimization may lead to even lower XOR count.

The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.
A1 and A2 criteria perform the best when the densities of the matrices are about 0.4-0.5. However, our algorithm is BP-like (like [RTA18]) which makes it too costly if the matrix grows very large. More techniques in local optimization may lead to even lower XOR count. The average (XOR) cost of implementing a matrix with density 0.9 is actually less than one with a density of 0.2.
<table>
<thead>
<tr>
<th><strong>References</strong></th>
</tr>
</thead>
</table>
| **Joan Boyar and René Peralta.**  
A New Combinational Logic Minimization Technique with Applications to Cryptology.  
| **Sébastien Duval and Gaëtan Leurent.**  
MDS Matrices with Lightweight Circuits.  
| **Jérémy Jean.**  
TikZ for Cryptographers.  
| **Thorsten Kranz, Gregor Leander, Ko Stoffelen, and Friedrich Wiemer.**  
Shorter Linear Straight-Line Programs for MDS Matrices.  
| **Alexander Maximov.**  
AES MixColumn with 92 XOR gates.  
Christof Paar and Martin Rosner. 
Comparison of arithmetic architectures for Reed-Solomon decoders in reconfigurable hardware. 

Arash Reyhani-Masoleh, Mostafa M. I. Taha, and Doaa Ashmawy. 
Smashing the Implementation Records of AES S-box. 

Andrea Visconti, Chiara Valentina Schiavo, and René Peralta. 
Improved upper bounds for the expected circuit complexity of dense systems of linear equations over GF(2). 

Clifford Wolf. 
Yosys open synthesis suite. 
http://www.clifford.at/yosys/.

Optimizing implementations of linear layers. 