Linear codes for SSS-based polynomial masking

The SSS-based masking is a special case of GCM (Generalized Code-based masking)

$n = 3$ shares, $t = 1$, $\ell = 8$ bits: (3,1)-SSS (Shamir's Secret Sharing)

Parameters:

- $Z = (X + \alpha_1 Y_1, X + \alpha_2 Y_1, X + \alpha_3 Y_1) = X\mathbf{G} + Y\mathbf{H}$ where $X, Y = (Y_1)$ and Z are the sensitive variable, a mask and the protected variable, respectively, where α_i for $1 \le i \le 3$ are three public points in SSS-scheme
- $\mathbf{G} = [1, 1, 1]$ and $\mathbf{H} = [\alpha_1, \alpha_2, \alpha_3]$ are two generator matrices of codes $\mathcal{C}$ and $\mathcal{D}$, resp.
- $\alpha_i \in \mathbb{F}_{2^\ell} \backslash \{0\}$ and $\alpha_1 \neq \alpha_2 \neq \alpha_3$, thus there are $\binom{255}{3}$ = 2731135 linear codes for (3,1)-SSS
- Each nonzero element over $\mathbb{F}_{2^\ell}$ can be denoted as α^i where $i \in \{0, 1, \ldots, 254\}$, the corresponding irreducible polynomial is $g(\alpha) = \alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
- Due to equivalence of linear codes, we simplify the enumeration by choosing $(\alpha_1, \alpha_2, \alpha_3) = (\alpha^i, \alpha^j, \alpha^k)$ where $i = 0$ and $0 < j < k$. Therefore, we get 32131 linear codes

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import pandas as pd # Pandas for tables
from IPython.display import Latex
from IPython.display import HTML
```

In [2]:
```
def read_log(file_name):

    pow_ind = []
    dim_all = []
    d_orig_w = []
    d_dual_w = []
    d_dual_b = []
    with open(file_name, 'r') as fp:
        wd = fp.read().split("\n")[:]
        len_all = 0
        for i in range(len(wd)):
            if wd[i].startswith("j, k ="):
                pow_ind.append([int(i) for i in re.findall(r"\d+", wd[i])])
                len_all = len_all + 1
            elif wd[i].startswith("Dimension:"):
                dim_all.append([int(i) for i in re.findall(r"\d+", wd[i])])
            elif wd[i].startswith("WD orig D (word):"):
                d_orig_w.append([int(i) for i in re.findall(r"\d+", wd[i])])
            elif wd[i].startswith("WD dual D (word):"):
                d_dual_w.append([int(i) for i in re.findall(r"\d+", wd[i])])
            elif wd[i].startswith("WD dual D  (bit):"):
                d_dual_b.append([int(i) for i in re.findall(r"\d+", wd[i]+wd[i+1])])
            else:
                continue

    return pow_ind, d_dual_b
```

1. Loading all weight enumerators

In [3]:
```
pow_ind, d_dual_b = read_log("./magma_paper/gen_codes_sss_3_1_8b.log") # indices and Weight distributions

print(len(pow_ind)) # 91 entries: 91 for (3,1)-SSS
#print(len(d_dual_b))
```

```
32131
```

1.1 Generating values

In [4]:
```
alpha_all = np.array(['$\\alpha^{%d}$'%i for i in np.arange(255)])
d_all = np.zeros(len(pow_ind))
B_all = np.zeros(len(pow_ind))
alpha_2 = np.zeros(len(pow_ind), dtype=int)
alpha_3 = np.zeros(len(pow_ind), dtype=int)
for i in range(len(pow_ind)):
    d_all[i] = d_dual_b[i][2]
    B_all[i] = d_dual_b[i][3]
    alpha_2[i] = pow_ind[i][0]
    alpha_3[i] = pow_ind[i][1]
```

1.2 Defining styles of dataframe

See more setting of dataframe from https://mode.com/example-gallery/python_dataframe_styling/

In [5]:
```
# Set properties for th, td and caption elements in dataframe
th_props = [('font-size', '14px'), ('text-align', 'left'), ('font-weight', 'bold'), ('background-color', '#E0E0E0')]
td_props = [('font-size', '13px'), ('text-align', 'left'), ('min-width', '80px')]
cp_props = [('font-size', '16px'), ('text-align', 'center')]
# Set table styles
styles = [dict(selector="th", props=th_props), dict(selector="td", props=td_props), dict(selector="caption", props=cp_props)]
cm_1 = sns.light_palette("red", as_cmap=True)
cm_2 = sns.light_palette("purple", as_cmap=True, reverse=True)
```

In [6]:
```
#df = pd.DataFrame({'$\\alpha_2$': alpha_all[alpha_2[:]], '$\\alpha_3$': alpha_all[alpha_3[:]], '$d_{\mathcal{D}}^\perp$': d_all[:],
#                   '$B_{d_{\mathcal{D}}^\perp}$': B_all[:], 'Weight Enumerators': d_dual_b[:]})

pd.set_option('display.max_colwidth', 1000)
pd.set_option('display.width', 800)
```

We omit the complete Tab. I here to avoid wasting of spaces. Hereafter, we only show all codes with maximized $d_{\mathcal{D}}^{\perp} = 4$.

2. Optimal linear codes for (3,1)-SSS-based masking

2.1 Linear codes with $d_{\mathcal{D}}^{\perp} = 4$

We focus on the the linear codes with greater $d_{\mathcal{D}}^{\perp}$, which are better in the sense of side-channel resistance (from our paper).

In [7]:
```
# Finding the indices of d_C=6
d_index = []
d_index_alpha_2 = []
d_index_alpha_3 = []
d_D_perp = 4
for i in range(len(d_dual_b)):
    if d_dual_b[i][2] == d_D_perp:
        d_index.append(i)
        d_index_alpha_2.append(pow_ind[i][0])
        d_index_alpha_3.append(pow_ind[i][1])

d_index = np.array(d_index)
d_index_alpha_2 = np.array(d_index_alpha_2)
d_index_alpha_3 = np.array(d_index_alpha_3)
```

In [8]:
```
print(len(d_index))
#print(d_index)
```

```
768
```

In [9]:
```
def highlight(s, threshold, column):
    is_min = pd.Series(data=False, index=s.index)
    is_min[column] = (s.loc[column] <= threshold)
    return ['background-color: gold' if is_min.any() else '' for v in is_min]
```

In [10]:
```
df_4 = pd.DataFrame({'$\\alpha_2$': np.array(alpha_all)[d_index_alpha_2], '$\\alpha_3$': np.array(alpha_all)[d_index_alpha_3], '$d_{\mathcal{D}}^
                   d_all[d_index], '$B_{d_{\mathcal{D}}^\perp}$': B_all[d_index], 'Weight Enumerators* (the first 22 values)': np.array(d_dual_b
df_4 = df_4.sort_values(by=['$B_{d_{\mathcal{D}}^\perp}$'], ascending=True)

(df_4.style
    .apply(highlight, threshold=12, column=['$B_{d_{\mathcal{D}}^\perp}$'], axis=1)
    .background_gradient(cmap=cm_2, subset=['$B_{d_{\mathcal{D}}^\perp}$'])
    .set_caption('Tab. II Linear codes for (3,1)-SSS-based masking with $d_{\mathcal{D}}^\perp=3$.')
    .set_table_styles(styles))
```

Out[10]:

Tab. II Linear codes for (3,1)-SSS-based masking with $d_{\mathcal{D}}^{\perp} = 3$.

	α_2	α_3	$d_{\mathcal{D}}^{\perp}$	$B_{d_{\mathcal{D}}^{\perp}}$	Weight Enumerators* (the first 22 values)
707	α^{175}	α^{247}	4	36	[0, 1, 4, 36, 5, 200, 6, 538, 7, 1312, 8, 2854, 9, 5088, 10, 7712, 11, 9824, 12, 10476, 175, 248]
311	α^{72}	α^{80}	4	36	[0, 1, 4, 36, 5, 200, 6, 538, 7, 1312, 8, 2854, 9, 5088, 10, 7712, 11, 9824, 12, 10476, 72, 81]
24	α^{8}	α^{183}	4	36	[0, 1, 4, 36, 5, 200, 6, 538, 7, 1312, 8, 2854, 9, 5088, 10, 7712, 11, 9824, 12, 10476, 8, 184]
492	α^{87}	α^{214}	4	37	[0, 1, 4, 37, 5, 198, 6, 537, 7, 1320, 8, 2842, 9, 5080, 10, 7748, 11, 9816, 12, 10442, 87, 215]
485	α^{86}	α^{214}	4	37	[0, 1, 4, 37, 5, 200, 6, 534, 7, 1308, 8, 2862, 9, 5108, 10, 7692, 11, 9788, 12, 10526, 86, 215]
552	α^{127}	α^{168}	4	37	[0, 1, 4, 37, 5, 198, 6, 537, 7, 1320, 8, 2842, 9, 5080, 10, 7748, 11, 9816, 12, 10442, 127, 169]
214	α^{41}	α^{128}	4	37	[0, 1, 4, 37, 5, 198, 6, 537, 7, 1320, 8, 2842, 9, 5080, 10, 7748, 11, 9816, 12, 10442, 41, 129]
494	α^{88}	α^{104}	4	37	[0, 1, 4, 37, 5, 197, 6, 542, 7, 1308, 8, 2838, 9, 5144, 10, 7700, 11, 9708, 12, 10566, 88, 105]
213	α^{41}	α^{127}	4	37	[0, 1, 4, 37, 5, 200, 6, 534, 7, 1308, 8, 2862, 9, 5108, 10, 7692, 11, 9788, 12, 10526, 41, 128]
484	α^{86}	α^{213}	4	37	[0, 1, 4, 37, 5, 200, 6, 536, 7, 1306, 8, 2848, 9, 5122, 10, 7734, 11, 9746, 12, 10456, 86, 214]
229	α^{42}	α^{128}	4	37	[0, 1, 4, 37, 5, 200, 6, 536, 7, 1306, 8, 2848, 9, 5122, 10, 7734, 11, 9746, 12, 10456, 42, 129]
325	α^{72}	α^{245}	4	37	[0, 1, 4, 37, 5, 201, 6, 526, 7, 1312, 8, 2902, 9, 5068, 10, 7620, 11, 9888, 12, 10566, 72, 246]
575	α^{151}	α^{239}	4	37	[0, 1, 4, 37, 5, 197, 6, 542, 7, 1308, 8, 2838, 9, 5144, 10, 7700, 11, 9708, 12, 10566, 151, 240]
50	α^{10}	α^{82}	4	37	[0, 1, 4, 37, 5, 201, 6, 526, 7, 1312, 8, 2902, 9, 5068, 10, 7620, 11, 9888, 12, 10566, 10, 83]
167	α^{16}	α^{167}	4	37	[0, 1, 4, 37, 5, 197, 6, 542, 7, 1308, 8, 2838, 9, 5144, 10, 7700, 11, 9708, 12, 10566, 16, 168]
698	α^{173}	α^{183}	4	37	[0, 1, 4, 37, 5, 201, 6, 526, 7, 1312, 8, 2902, 9, 5068, 10, 7620, 11, 9888, 12, 10566, 173, 184]
558	α^{128}	α^{169}	4	37	[0, 1, 4, 37, 5, 200, 6, 534, 7, 1308, 8, 2862, 9, 5108, 10, 7692, 11, 9788, 12, 10526, 128, 170]
553	α^{127}	α^{169}	4	37	[0, 1, 4, 37, 5, 200, 6, 536, 7, 1306, 8, 2848, 9, 5122, 10, 7734, 11, 9746, 12, 10456, 127, 170]
551	α^{127}	α^{167}	4	38	[0, 1, 4, 38, 5, 197, 6, 531, 7, 1326, 8, 2856, 9, 5066, 10, 7734, 11, 9830, 12, 10442, 127, 168]
491	α^{87}	α^{213}	4	38	[0, 1, 4, 38, 5, 199, 6, 529, 7, 1312, 8, 2870, 9, 5108, 10, 7692, 11, 9760, 12, 10512, 87, 214]
696	α^{173}	α^{181}	4	38	[0, 1, 4, 38, 5, 208, 6, 509, 7, 1276, 8, 2978, 9, 5140, 10, 7464, 11, 9820, 12, 10732, 173, 182]
493	α^{87}	α^{215}	4	38	[0, 1, 4, 38, 5, 201, 6, 527, 7, 1306, 8, 2876, 9, 5102, 10, 7698, 11, 9810, 12, 10462, 87, 216]
705	α^{174}	α^{247}	4	38	[0, 1, 4, 38, 5, 200, 6, 525, 7, 1312, 8, 2890, 9, 5088, 10, 7656, 11, 9824, 12, 10532, 174, 248]
697	α^{173}	α^{182}	4	38	[0, 1, 4, 38, 5, 208, 6, 512, 7, 1276, 8, 2958, 9, 5140, 10, 7520, 11, 9820, 12, 10648, 173, 183]
702	α^{174}	α^{182}	4	38	[0, 1, 4, 38, 5, 207, 6, 510, 7, 1282, 8, 2972, 9, 5126, 10, 7478, 11, 9834, 12, 10718, 174, 183]
499	α^{88}	α^{215}	4	38	[0, 1, 4, 38, 5, 197, 6, 531, 7, 1326, 8, 2856, 9, 5066, 10, 7734, 11, 9830, 12, 10442, 88, 216]
500	α^{89}	α^{104}	4	38	[0, 1, 4, 38, 5, 194, 6, 548, 7, 1312, 8, 2802, 9, 5152, 10, 7764, 11, 9712, 12, 10508, 89, 105]
576	α^{151}	α^{240}	4	38	[0, 1, 4, 38, 5, 194, 6, 548, 7, 1312, 8, 2802, 9, 5152, 10, 7764, 11, 9712, 12, 10508, 151, 241]
199	α^{40}	α^{127}	4	38	[0, 1, 4, 38, 5, 201, 6, 527, 7, 1306, 8, 2876, 9, 5102, 10, 7698, 11, 9810, 12, 10462, 40, 128]
145	α^{15}	α^{166}	4	38	[0, 1, 4, 38, 5, 194, 6, 548, 7, 1312, 8, 2802, 9, 5152, 10, 7764, 11, 9712, 12, 10508, 15, 167]
343	α^{73}	α^{247}	4	38	[0, 1, 4, 38, 5, 207, 6, 510, 7, 1282, 8, 2972, 9, 5126, 10, 7478, 11, 9834, 12, 10718, 73, 248]
547	α^{126}	α^{168}	4	38	[0, 1, 4, 38, 5, 199, 6, 529, 7, 1312, 8, 2870, 9, 5108, 10, 7692, 11, 9760, 12, 10512, 126, 169]
1	α^{8}	α^{81}	4	38	[0, 1, 4, 38, 5, 207, 6, 510, 7, 1282, 8, 2972, 9, 5126, 10, 7478, 11, 9834, 12, 10718, 8, 82]
26	α^{9}	α^{82}	4	38	[0, 1, 4, 38, 5, 208, 6, 512, 7, 1276, 8, 2958, 9, 5140, 10, 7520, 11, 9820, 12, 10648, 9, 83]
23	α^{8}	α^{182}	4	38	[0, 1, 4, 38, 5, 200, 6, 525, 7, 1312, 8, 2890, 9, 5088, 10, 7656, 11, 9824, 12, 10532, 8, 183]
688	α^{172}	α^{180}	4	38	[0, 1, 4, 38, 5, 207, 6, 512, 7, 1278, 8, 2964, 9, 5146, 10, 7486, 11, 9798, 12, 10726, 172, 181]
375	α^{75}	α^{247}	4	38	[0, 1, 4, 38, 5, 207, 6, 512, 7, 1278, 8, 2964, 9, 5146, 10, 7486, 11, 9798, 12, 10726, 75, 248]
2	α^{8}	α^{82}	4	38	[0, 1, 4, 38, 5, 208, 6, 509, 7, 1276, 8, 2978, 9, 5140, 10, 7464, 11, 9820, 12, 10732, 8, 83]
230	α^{42}	α^{129}	4	38	[0, 1, 4, 38, 5, 199, 6, 529, 7, 1312, 8, 2870, 9, 5108, 10, 7692, 11, 9760, 12, 10512, 42, 130]
359	α^{74}	α^{247}	4	38	[0, 1, 4, 38, 5, 208, 6, 509, 7, 1276, 8, 2978, 9, 5140, 10, 7464, 11, 9820, 12, 10732, 74, 248]
200	α^{40}	α^{128}	4	38	[0, 1, 4, 38, 5, 197, 6, 531, 7, 1326, 8, 2856, 9, 5066, 10, 7734, 11, 9830, 12, 10442, 40, 129]
342	α^{73}	α^{246}	4	38	[0, 1, 4, 38, 5, 208, 6, 512, 7, 1276, 8, 2958, 9, 5140, 10, 7520, 11, 9820, 12, 10648, 73, 247]
557	α^{128}	α^{168}	4	38	[0, 1, 4, 38, 5, 201, 6, 527, 7, 1306, 8, 2876, 9, 5102, 10, 7698, 11, 9810, 12, 10462, 128, 169]
328	α^{73}	α^{81}	4	38	[0, 1, 4, 38, 5, 200, 6, 525, 7, 1312, 8, 2890, 9, 5088, 10, 7656, 11, 9824, 12, 10532, 73, 82]
3	α^{8}	α^{83}	4	38	[0, 1, 4, 38, 5, 207, 6, 512, 7, 1278, 8, 2964, 9, 5146, 10, 7486, 11, 9798, 12, 10726, 8, 84]
574	α^{151}	α^{238}	4	39	[0, 1, 4, 39, 5, 189, 6, 555, 7, 1302, 8, 2820, 9, 5186, 10, 7646, 11, 9742, 12, 10640, 151, 239]
546	α^{126}	α^{167}	4	39	[0, 1, 4, 39, 5, 195, 6, 527, 7, 1334, 8, 2864, 9, 5058, 10, 7714, 11, 9822, 12, 10492, 126, 168]
507	α^{90}	α^{194}	4	39	[0, 1, 4, 39, 5, 185, 6, 545, 7, 1372, 8, 2786, 9, 5008, 10, 7836, 11, 9852, 12, 10422, 90, 195]
512	α^{104}	α^{165}	4	39	[0, 1, 4, 39, 5, 185, 6, 545, 7, 1372, 8, 2786, 9, 5008, 10, 7836, 11, 9852, 12, 10422, 104, 166]
294	α^{61}	α^{243}	4	64	[0, 1, 4, 64, 5, 122, 6, 563, 7, 1364, 8, 2854, 9, 5228, 10, 7380, 11, 9796, 12, 10836, 61, 244]
267	α^{59}	α^{240}	4	64	[0, 1, 4, 64, 5, 129, 6, 555, 7, 1366, 8, 2824, 9, 5202, 10, 7534, 11, 9774, 12, 10670, 59, 241]
268	α^{59}	α^{241}	4	64	[0, 1, 4, 64, 5, 130, 6, 554, 7, 1354, 8, 2824, 9, 5250, 10, 7554, 11, 9682, 12, 10606, 59, 242]
94	α^{12}	α^{73}	4	64	[0, 1, 4, 64, 5, 122, 6, 563, 7, 1364, 8, 2854, 9, 5228, 10, 7380, 11, 9796, 12, 10836, 12, 74]
737	α^{182}	α^{195}	4	65	[0, 1, 4, 65, 5, 121, 6, 569, 7, 1352, 8, 2818, 9, 5292, 10, 7444, 11, 9688, 12, 10778, 182, 196]
280	α^{60}	α^{242}	4	65	[0, 1, 4, 65, 5, 121, 6, 569, 7, 1352, 8, 2818, 9, 5292, 10, 7444, 11, 9688, 12, 10778, 60, 243]
108	α^{13}	α^{73}	4	65	[0, 1, 4, 65, 5, 121, 6, 569, 7, 1352, 8, 2818, 9, 5292, 10, 7444, 11, 9688, 12, 10778, 13, 74]
729	α^{181}	α^{195}	4	66	[0, 1, 4, 66, 5, 127, 6, 559, 7, 1338, 8, 2836, 9, 5278, 10, 7458, 11, 9762, 12, 10718, 181, 196]
728	α^{181}	α^{194}	4	66	[0, 1, 4, 66, 5, 120, 6, 561, 7, 1360, 8, 2846, 9, 5264, 10, 7388, 11, 9744, 12, 10848, 181, 195]
293	α^{61}	α^{242}	4	66	[0, 1, 4, 66, 5, 120, 6, 561, 7, 1360, 8, 2846, 9, 5264, 10, 7388, 11, 9744, 12, 10848, 61, 243]
279	α^{60}	α^{241}	4	66	[0, 1, 4, 66, 5, 127, 6, 559, 7, 1338, 8, 2836, 9, 5278, 10, 7458, 11, 9762, 12, 10718, 60, 242]
122	α^{14}	α^{74}	4	66	[0, 1, 4, 66, 5, 127, 6, 559, 7, 1338, 8, 2836, 9, 5278, 10, 7458, 11, 9762, 12, 10718, 14, 75]
109	α^{13}	α^{74}	4	66	[0, 1, 4, 66, 5, 120, 6, 561, 7, 1360, 8, 2846, 9, 5264, 10, 7388, 11, 9744, 12, 10848, 13, 75]
759	α^{198}	α^{238}	4	67	[0, 1, 4, 67, 5, 132, 6, 510, 7, 1402, 8, 2920, 9, 5082, 10, 7510, 11, 9858, 12, 10588, 198, 239]
228	α^{42}	α^{59}	4	67	[0, 1, 4, 67, 5, 142, 6, 510, 7, 1355, 8, 2920, 9, 5162, 10, 7510, 11, 9806, 12, 10588, 42, 60]
193	α^{17}	α^{213}	4	67	[0, 1, 4, 67, 5, 142, 6, 510, 7, 1355, 8, 2920, 9, 5162, 10, 7510, 11, 9806, 12, 10588, 17, 214]
751	α^{196}	α^{238}	4	67	[0, 1, 4, 67, 5, 142, 6, 510, 7, 1355, 8, 2920, 9, 5162, 10, 7510, 11, 9806, 12, 10588, 196, 239]
197	α^{40}	α^{57}	4	67	[0, 1, 4, 67, 5, 132, 6, 510, 7, 1402, 8, 2920, 9, 5082, 10, 7510, 11, 9858, 12, 10588, 40, 58]
195	α^{17}	α^{215}	4	67	[0, 1, 4, 67, 5, 132, 6, 510, 7, 1402, 8, 2920, 9, 5082, 10, 7510, 11, 9858, 12, 10588, 17, 216]
438	α^{80}	α^{142}	4	68	[0, 1, 4, 68, 5, 130, 6, 530, 7, 1363, 8, 2880, 9, 5202, 10, 7498, 11, 9782, 12, 10606, 80, 143]
301	α^{62}	α^{175}	4	68	[0, 1, 4, 68, 5, 130, 6, 530, 7, 1363, 8, 2880, 9, 5202, 10, 7498, 11, 9782, 12, 10606, 62, 176]
525	α^{113}	α^{193}	4	68	[0, 1, 4, 68, 5, 130, 6, 530, 7, 1363, 8, 2880, 9, 5202, 10, 7498, 11, 9782, 12, 10606, 113, 194]
288	α^{61}	α^{175}	4	71	[0, 1, 4, 71, 5, 120, 6, 550, 7, 1360, 8, 2814, 9, 5264, 10, 7532, 11, 9744, 12, 10642, 61, 176]
252	α^{57}	α^{171}	4	71	[0, 1, 4, 71, 5, 131, 6, 504, 7, 1393, 8, 2906, 9, 5132, 10, 7528, 11, 9772, 12, 10638, 57, 172]
473	α^{84}	α^{141}	4	71	[0, 1, 4, 71, 5, 131, 6, 504, 7, 1393, 8, 2906, 9, 5132, 10, 7528, 11, 9772, 12, 10638, 84, 142]
437	α^{80}	α^{141}	4	71	[0, 1, 4, 71, 5, 120, 6, 550, 7, 1360, 8, 2814, 9, 5264, 10, 7532, 11, 9744, 12, 10642, 80, 142]
536	α^{114}	α^{194}	4	71	[0, 1, 4, 71, 5, 120, 6, 550, 7, 1360, 8, 2814, 9, 5264, 10, 7532, 11, 9744, 12, 10642, 114, 195]
540	α^{114}	α^{198}	4	71	[0, 1, 4, 71, 5, 131, 6, 504, 7, 1393, 8, 2906, 9, 5132, 10, 7528, 11, 9772, 12, 10638, 114, 199]
450	α^{81}	α^{142}	4	72	[0, 1, 4, 72, 5, 121, 6, 526, 7, 1390, 8, 2852, 9, 5194, 10, 7558, 11, 9734, 12, 10586, 81, 143]
539	α^{114}	α^{197}	4	72	[0, 1, 4, 72, 5, 127, 6, 494, 7, 1444, 8, 2826, 9, 5184, 10, 7620, 11, 9540, 12, 10760, 114, 198]
287	α^{61}	α^{174}	4	72	[0, 1, 4, 72, 5, 121, 6, 526, 7, 1390, 8, 2852, 9, 5194, 10, 7558, 11, 9734, 12, 10586, 61, 175]
526	α^{113}	α^{194}	4	72	[0, 1, 4, 72, 5, 121, 6, 526, 7, 1390, 8, 2852, 9, 5194, 10, 7558, 11, 9734, 12, 10586, 113, 195]
259	α^{58}	α^{172}	4	72	[0, 1, 4, 72, 5, 127, 6, 494, 7, 1444, 8, 2826, 9, 5184, 10, 7620, 11, 9540, 12, 10760, 58, 173]
467	α^{83}	α^{141}	4	72	[0, 1, 4, 72, 5, 127, 6, 494, 7, 1444, 8, 2826, 9, 5184, 10, 7620, 11, 9540, 12, 10760, 83, 142]
258	α^{58}	α^{171}	4	73	[0, 1, 4, 73, 5, 129, 6, 471, 7, 1474, 8, 2864, 9, 5094, 10, 7626, 11, 9594, 12, 10768, 58, 172]
251	α^{57}	α^{170}	4	73	[0, 1, 4, 73, 5, 131, 6, 498, 7, 1399, 8, 2900, 9, 5098, 10, 7578, 11, 9850, 12, 10548, 57, 171]
530	α^{113}	α^{198}	4	73	[0, 1, 4, 73, 5, 131, 6, 498, 7, 1399, 8, 2900, 9, 5098, 10, 7578, 11, 9850, 12, 10548, 113, 199]
529	α^{113}	α^{197}	4	73	[0, 1, 4, 73, 5, 129, 6, 471, 7, 1474, 8, 2864, 9, 5094, 10, 7626, 11, 9594, 12, 10768, 113, 198]
479	α^{85}	α^{142}	4	73	[0, 1, 4, 73, 5, 131, 6, 498, 7, 1399, 8, 2900, 9, 5098, 10, 7578, 11, 9850, 12, 10548, 85, 143]
474	α^{84}	α^{142}	4	73	[0, 1, 4, 73, 5, 129, 6, 471, 7, 1474, 8, 2864, 9, 5094, 10, 7626, 11, 9594, 12, 10768, 84, 143]
275	α^{60}	α^{174}	4	74	[0, 1, 4, 74, 5, 116, 6, 538, 7, 1380, 8, 2822, 9, 5228, 10, 7568, 11, 9764, 12, 10552, 60, 175]
274	α^{60}	α^{173}	4	74	[0, 1, 4, 74, 5, 118, 6, 518, 7, 1402, 8, 2860, 9, 5186, 10, 7566, 11, 9698, 12, 10566, 60, 174]
449	α^{81}	α^{141}	4	74	[0, 1, 4, 74, 5, 116, 6, 538, 7, 1380, 8, 2822, 9, 5228, 10, 7568, 11, 9764, 12, 10552, 81, 142]
460	α^{82}	α^{142}	4	74	[0, 1, 4, 74, 5, 118, 6, 518, 7, 1402, 8, 2860, 9, 5186, 10, 7566, 11, 9698, 12, 10566, 82, 143]
537	α^{114}	α^{195}	4	74	[0, 1, 4, 74, 5, 116, 6, 538, 7, 1380, 8, 2822, 9, 5228, 10, 7568, 11, 9764, 12, 10552, 114, 196]
538	α^{114}	α^{196}	4	74	[0, 1, 4, 74, 5, 118, 6, 518, 7, 1402, 8, 2860, 9, 5186, 10, 7566, 11, 9698, 12, 10566, 114, 197]
527	α^{113}	α^{195}	4	74	[0, 1, 4, 74, 5, 118, 6, 518, 7, 1402, 8, 2860, 9, 5186, 10, 7566, 11, 9698, 12, 10566, 113, 196]
264	α^{59}	α^{173}	4	74	[0, 1, 4, 74, 5, 118, 6, 518, 7, 1402, 8, 2860, 9, 5186, 10, 7566, 11, 9698, 12, 10566, 59, 174]
459	α^{82}	α^{141}	4	74	[0, 1, 4, 74, 5, 118, 6, 518, 7, 1402, 8, 2860, 9, 5186, 10, 7566, 11, 9698, 12, 10566, 82, 142]
528	α^{113}	α^{196}	4	76	[0, 1, 4, 76, 5, 118, 6, 500, 7, 1429, 8, 2874, 9, 5132, 10, 7588, 11, 9680, 12, 10608, 113, 197]
263	α^{59}	α^{172}	4	76	[0, 1, 4, 76, 5, 118, 6, 500, 7, 1429, 8, 2874, 9, 5132, 10, 7588, 11, 9680, 12, 10608, 59, 173]
468	α^{83}	α^{142}	4	76	[0, 1, 4, 76, 5, 118, 6, 500, 7, 1429, 8, 2874, 9, 5132, 10, 7588, 11, 9680, 12, 10608, 83, 143]

2.2 Optimal codes for (3,1)-SSS-based masking

As shown in our paper, the codes satifying two conditions are optimal:

- Maximizing $d_{\mathcal{D}}^{\perp}$, here $\max\{d_{\mathcal{D}}^{\perp}\} = 4$
- Minimizing $B_{d_{\mathcal{D}}^{\perp}}$, here $\min\{B_{d_{\mathcal{D}}^{\perp}}\} = 36$

Note that we use two complementary metrics **SNR** (signal-to-noise ratio) and **MI** (mutual information) to assess the side-channel resistance of SSS-based masking with different codes.

As a result of Tab. II, we conclude that the optimal codes for (3,1)-SSS based masking are generated by $\mathbf{H} = [\alpha_1, \alpha_2, \alpha_3]$ where $(\alpha_1, \alpha_2, \alpha_3) \in \{(\alpha^0, \alpha^{175}, \alpha^{247}),\ (\alpha^0, \alpha^{72}, \alpha^{80}),\ (\alpha^0, \alpha^8, \alpha^{183})\}$. Note that permutation on three public points does not change the codes due to equivalence.

The generator matrices of the three optimal codes are shown below.

- $(\alpha_1, \alpha_2, \alpha_3) = (\alpha^0, \alpha^{175}, \alpha^{247})$

$$\mathbf{H}_{optimal} = \begin{pmatrix} \alpha^0 & \alpha^{175} & \alpha^{247} \end{pmatrix} \in \mathbb{F}_{2^8}^{1\times 3} = \begin{pmatrix} 1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0&0&1&1&0&0&0&1&1&1&1&1&0&1&1&0&0&0 \\ 0&0&1&0&0&0&0&0&1&1&0&1&1&0&1&1&0&1&1&0&1&1&0&0 \\ 0&0&0&1&0&0&0&0&1&1&0&1&0&1&0&1&0&0&1&1&0&1&1&0 \\ 0&0&0&0&1&0&0&0&1&1&0&1&0&0&1&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&0&1&0&0&0&1&1&0&1&0&0&1&1&0&1&1&0&1&0&1 \\ 0&0&0&0&0&0&1&0&1&0&0&0&1&1&0&0&1&1&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&1&0&0&0&1&1&0&0&1&1&1&0&0&0&1 \end{pmatrix} \in \mathbb{F}_2^{8\times 24}$$

- $(\alpha_1, \alpha_2, \alpha_3) = (\alpha^0, \alpha^{72}, \alpha^{80})$

$$\mathbf{H}_{optimal} = \begin{pmatrix} \alpha^0 & \alpha^{72} & \alpha^{80} \end{pmatrix} \in \mathbb{F}_{2^8}^{1\times 3} = \begin{pmatrix} 1&0&0&0&0&0&0&0&1&0&1&0&0&1&1&0&1&0&1&1&1&1&1&1 \\ 0&1&0&0&0&0&0&0&0&1&0&1&0&0&1&1&1&1&1&0&0&1&1&1 \\ 0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&1&1&1&0&0&1&0&1&1 \\ 0&0&0&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&0&1&1&1&0&1 \\ 0&0&0&0&1&0&0&0&0&1&1&1&1&0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&1&0&0&0&0&1&1&1&1&0&0&0&1&1&0&1&0&1&1 \\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&1&0&1&0&0&0&1&1&0&1 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&0 \end{pmatrix} \in \mathbb{F}_2^{8\times 24}$$

- $(\alpha_1, \alpha_2, \alpha_3) = (\alpha^0, \alpha^8, \alpha^{183})$

$$\mathbf{H}_{optimal} = \begin{pmatrix} \alpha^0 & \alpha^{8} & \alpha^{183} \end{pmatrix} \in \mathbb{F}_{2^8}^{1\times 3} = \begin{pmatrix} 1&0&0&0&0&0&0&0&1&0&1&1&1&0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&0&0&0&1&0&1&1&1&0&0&1&0&1&0&1&0&0&1 \\ 0&0&1&0&0&0&0&0&0&0&1&0&1&1&1&0&1&1&1&0&1&1&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&0&1&1&1&0&1&1&1&0&1&1&0 \\ 0&0&0&0&1&0&0&0&1&0&1&1&0&0&1&1&0&0&1&1&1&0&1&1 \\ 0&0&0&0&0&1&0&0&1&1&1&0&0&0&0&1&1&0&1&0&0&1&0&1 \\ 0&0&0&0&0&0&1&0&1&1&0&0&1&0&0&0&1&1&1&0&1&0&1&0 \\ 0&0&0&0&0&0&0&1&0&1&1&0&0&1&0&0&0&1&1&1&0&1&0&1 \end{pmatrix} \in \mathbb{F}_2^{8\times 24}$$

In [ ]:

This website does not host notebooks, it only renders notebooks available on other websites.
Delivered by Fastly, Rendered by OVHCloud
nbviewer GitHub repository.
nbviewer version: e83752b
nbconvert version: 5.6.1
Rendered a minute ago