Linear codes for SSS-based polynomial masking

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

$n = 3$ shares, $t = 1, \ell = 4$ bits: (3,1)-SSS (Shamir's Secret Sharing)

Parameters:

- $\mathbf{Z} = (X + \alpha_1 Y_0, X + \alpha_2 Y_1, X + \alpha_3 Y_2) = X\mathbf{G} + Y\mathbf{H}$ where $X, Y = (Y_0)$ 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.
- $\alpha_i \in \mathbb{F}_{2^\ell}(\mathbf{0})$ and $\alpha_1 \neq \alpha_2 \neq \alpha_3$, thus there are $\binom{15}{3}$=455 linear codes for (3,1)-SSS
- Each nonzero element over $\mathbb{F}_{2^\ell}$ can be denoted as ω^i where $i \in \{0, 1, \ldots, 14\}$, the corresponding irreducible polynomial is $g(\alpha) = \alpha^4 + \alpha + 1$
- Due to equivalence of linear codes, we simplify the enumeration by choosing $(\alpha_1, \alpha_2, \alpha_3) = (\omega^0, \omega^i, \omega^j)$ where $i = 0$ and $0 < j < k$. Therefore, we get 91 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 = []
    d_ab_all = []
    d_wrig_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("), k ="):
                pow_ind.append([int(i) for i in re.findall(r"\d+", wd[i])])
                len_all += len(wd[i])
            elif wd[i].startswith("Dimension:"):
                d_ab_all.append([int(i) for i in re.findall(r"\d+", wd[i])])
            elif wd[i].startswith("TD dual 2 (wrig):"):
                d_wrig_w.append([int(i) for i in re.findall(r"\d+", wd[i])])
            elif wd[i].startswith("TD dual 2 (word):"):
                d_dual_w.append([int(i) for i in re.findall(r"\d+", wd[i])])
            elif wd[i].startswith("TD dual 2 (bit):"):
                d_dual_b.append([int(i) for i in re.findall(r"\d+", 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/pms_codes_sss_3_1_4b.log") # indices and Weight distributions

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

91

1.1 Generating values

In [4]:
```
alpha_all = np.array(['$\\alpha ^%d$'%i for i in np.arange(15)])
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_ab_all[i][0]
    B_all[i] = d_dual_b[i][0]
    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://moode.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', '#EDEDED')]
td_props = [('font-size', '12px'), ('text-align', 'left'), ('min-width', '50px')]
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[:]], '$d_{\mathcal{D}}$': d_all[:],
                   '$B_{d_{\mathcal{D}}}$': B_all[:], 'Weight Enumerators': d_dual_b[:]})

pd.set_option('display.max_colwidth', 1000)
pd.set_option('display.width', 800)
(df.style
   .background_gradient(cmap=cm_1, subset=['$d_{\mathcal{D}}$','$B_{d_{\mathcal{D}}}$'])
   .background_gradient(cmap=cm_2, subset=['$B_{d_{\mathcal{D}}}$'])
   .set_caption('Tab. I All linear codes for (3,1)-SSS-based masking with $n=3$ shares over $\\mathbb{F}_{2^4}$.')
   .set_table_styles(styles))
```

Out[6]:

Tab. I All linear codes for (3,1)-SSS-based masking with $n = 3$ shares over $\mathbb{F}_{2^4}$.

	α_2	α_3	$d_{\mathcal{D}}$	$B_{d_{\mathcal{D}}}$	Weight Enumerators
0	α^0	α^2	2	1	[0, 1, 2, 8, 9, 10, 4, 28, 5, 50, 6, 50, 7, 62, 8, 35, 9, 6, 10, 6, 1, 3]
1	α^0	α^3	2	6	[0, 1, 2, 6, 3, 14, 4, 22, 5, 52, 6, 64, 7, 48, 8, 33, 9, 12, 10, 2, 11, 2, 1, 4]
2	α^0	α^4	2	4	[0, 1, 2, 4, 3, 16, 4, 23, 5, 48, 6, 72, 7, 48, 8, 23, 9, 16, 10, 4, 12, 1, 1, 5]
3	α^0	α^5	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 1, 6]
4	α^0	α^6	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 1, 7]
5	α^0	α^7	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 1, 8]
6	α^0	α^8	2	2	[0, 1, 2, 3, 3, 14, 4, 30, 5, 52, 6, 58, 7, 48, 8, 33, 9, 12, 10, 3, 11, 2, 1, 9]
7	α^0	α^9	2	3	[0, 1, 2, 3, 3, 13, 4, 30, 5, 56, 6, 58, 7, 48, 8, 35, 9, 14, 10, 5, 11, 1, 1, 10]
8	α^0	α^{10}	2	3	[0, 1, 2, 3, 3, 10, 4, 36, 5, 58, 6, 48, 7, 48, 8, 35, 9, 14, 10, 5, 11]
9	α^0	α^{11}	2	3	[0, 1, 2, 3, 3, 10, 4, 36, 5, 58, 6, 48, 7, 48, 8, 35, 9, 14, 10, 5, 1, 12]
10	α^1	α^{13}	2	4	[0, 1, 2, 4, 3, 12, 4, 28, 5, 58, 6, 58, 7, 42, 8, 35, 9, 14, 10, 2, 11, 2, 1, 13]
11	α^1	α^{14}	2	6	[0, 1, 2, 6, 3, 12, 4, 26, 5, 52, 6, 60, 7, 52, 8, 29, 9, 12, 10, 6, 11, 4]
12	α^1	α^{12}	2	6	[0, 1, 2, 6, 3, 10, 4, 28, 5, 50, 6, 60, 7, 62, 8, 35, 9, 6, 10, 6, 2, 3]
13	α^2	α^3	2	6	[0, 1, 2, 6, 3, 12, 4, 26, 5, 52, 6, 60, 7, 52, 8, 29, 9, 12, 10, 6, 2, 4]
14	α^2	α^4	2	4	[0, 1, 2, 4, 3, 14, 4, 26, 5, 52, 6, 64, 7, 48, 8, 29, 9, 12, 10, 4, 11, 2, 2, 5]
15	α^2	α^5	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 2, 6]
16	α^2	α^6	2	2	[0, 1, 2, 2, 3, 16, 4, 30, 5, 48, 6, 64, 7, 48, 8, 25, 9, 16, 10, 6, 12, 2, 2, 7]
17	α^2	α^7	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 2, 8]
18	α^2	α^8	2	2	[0, 1, 2, 2, 3, 16, 4, 31, 5, 48, 6, 60, 7, 48, 8, 31, 9, 16, 10, 2, 12, 1, 2, 9]
19	α^2	α^9	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 2, 10]
20	α^2	α^{10}	2	2	[0, 1, 2, 2, 3, 13, 4, 34, 5, 56, 6, 52, 7, 42, 8, 37, 9, 16, 10, 2, 11, 1, 2, 11]
21	α^2	α^{11}	2	2	[0, 1, 2, 2, 3, 13, 4, 34, 5, 56, 6, 52, 7, 42, 8, 37, 9, 16, 10, 2, 11, 1, 2, 12]
22	α^2	α^{12}	2	3	[0, 1, 2, 3, 3, 12, 4, 34, 5, 52, 6, 64, 7, 52, 8, 29, 9, 12, 10, 7, 2, 13]
23	α^2	α^{13}	2	6	[0, 1, 2, 4, 3, 14, 4, 28, 5, 52, 6, 64, 7, 48, 8, 29, 9, 12, 10, 4, 11, 2, 2, 14]
24	α^2	α^{14}	2	6	[0, 1, 2, 6, 3, 14, 4, 22, 5, 52, 6, 64, 7, 48, 8, 33, 9, 12, 10, 2, 11, 2, 3, 4]
25	α^3	α^4	2	4	[0, 1, 2, 4, 3, 12, 4, 28, 5, 58, 6, 58, 7, 42, 8, 35, 9, 14, 10, 2, 11, 2, 3, 5]
26	α^3	α^5	2	3	[0, 1, 2, 3, 3, 12, 4, 34, 5, 52, 6, 64, 7, 52, 8, 29, 9, 12, 10, 7, 3, 6]
27	α^3	α^6	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 3, 7]
28	α^3	α^7	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 3, 8]
29	α^3	α^8	2	1	[0, 1, 2, 1, 3, 16, 4, 36, 5, 46, 6, 52, 7, 54, 8, 35, 9, 10, 10, 3, 11, 2, 3, 9]
30	α^3	α^9	2	1	[0, 1, 2, 1, 3, 17, 4, 34, 5, 44, 6, 58, 7, 54, 8, 29, 9, 12, 10, 5, 11, 1, 3, 10]
31	α^3	α^{10}	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 3, 11]
32	α^3	α^{11}	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 3, 12]
33	α^3	α^{12}	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 3, 13]
34	α^3	α^{13}	2	2	[0, 1, 2, 2, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 3, 14]
35	α^3	α^{14}	2	4	[0, 1, 2, 4, 3, 16, 4, 23, 5, 48, 6, 72, 7, 48, 8, 23, 9, 16, 10, 4, 12, 1, 4, 5]
36	α^4	α^5	2	3	[0, 1, 2, 3, 3, 10, 4, 36, 5, 58, 6, 48, 7, 48, 8, 35, 9, 14, 10, 5, 4, 6]
37	α^4	α^6	2	2	[0, 1, 2, 2, 3, 13, 4, 34, 5, 56, 6, 52, 7, 42, 8, 37, 9, 16, 10, 2, 11, 1, 4, 7]
38	α^4	α^7	2	2	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 4, 8]
39	α^4	α^8	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 9]
40	α^4	α^9	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 10]
41	α^4	α^{10}	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 11]
42	α^4	α^{11}	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 12]
43	α^4	α^{12}	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 4, 13]
44	α^4	α^{13}	2	2	[0, 1, 2, 2, 3, 16, 4, 30, 5, 48, 6, 64, 7, 48, 8, 25, 9, 16, 10, 6, 4, 14]
45	α^4	α^{14}	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 4, 15]
46	α^5	α^6	2	2	[0, 1, 2, 2, 3, 13, 4, 34, 5, 56, 6, 52, 7, 42, 8, 37, 9, 16, 10, 2, 11, 1, 5, 6]
47	α^5	α^7	2	2	[0, 1, 2, 2, 3, 13, 4, 34, 5, 56, 6, 52, 7, 42, 8, 37, 9, 16, 10, 2, 11, 1, 5, 7]
48	α^5	α^8	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 5, 8]
49	α^5	α^9	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 5, 10]
50	α^5	α^{10}	3	10	[0, 1, 3, 16, 4, 39, 5, 48, 6, 48, 7, 48, 8, 39, 9, 16, 12, 1, 5, 11]
51	α^5	α^{11}	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 5, 12]
52	α^5	α^{12}	2	1	[0, 1, 2, 1, 3, 16, 4, 36, 5, 46, 6, 52, 7, 54, 8, 35, 9, 10, 10, 3, 11, 2, 5, 13]
53	α^5	α^{13}	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 5, 14]
54	α^5	α^{14}	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 5, 7]
55	α^6	α^7	2	3	[0, 1, 2, 3, 3, 13, 4, 30, 5, 56, 6, 58, 7, 48, 8, 33, 9, 12, 10, 3, 11, 1, 6, 8]
56	α^6	α^8	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 6, 9]
57	α^6	α^9	2	1	[0, 1, 2, 1, 3, 17, 4, 34, 5, 44, 6, 58, 7, 54, 8, 29, 9, 12, 10, 5, 11, 1, 6, 10]
58	α^6	α^{10}	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 6, 11]
59	α^6	α^{11}	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 6, 12]
60	α^6	α^{12}	2	1	[0, 1, 2, 1, 3, 17, 4, 34, 5, 44, 6, 58, 7, 54, 8, 29, 9, 12, 10, 5, 11, 1, 6, 13]
61	α^6	α^{13}	2	2	[0, 1, 2, 2, 3, 16, 4, 31, 5, 48, 6, 60, 7, 48, 8, 31, 9, 16, 10, 2, 12, 1, 6, 14]
62	α^6	α^{14}	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 7, 8]
63	α^7	α^8	2	2	[0, 1, 2, 2, 3, 14, 4, 30, 5, 52, 6, 58, 7, 48, 8, 33, 9, 12, 10, 3, 11, 2, 7, 9]
64	α^7	α^9	2	2	[0, 1, 2, 2, 3, 16, 4, 31, 5, 48, 6, 60, 7, 48, 8, 31, 9, 16, 10, 2, 12, 1, 7, 10]
65	α^7	α^{10}	2	2	[0, 1, 2, 2, 3, 16, 4, 36, 5, 46, 6, 52, 7, 54, 8, 35, 9, 10, 10, 3, 11, 2, 7, 11]
66	α^7	α^{11}	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 7, 12]
67	α^7	α^{12}	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 7, 13]
68	α^7	α^{13}	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 7, 14]
69	α^7	α^{14}	2	3	[0, 1, 2, 3, 3, 14, 4, 30, 5, 52, 6, 58, 7, 48, 8, 33, 9, 12, 10, 3, 11, 2, 8, 9]
70	α^8	α^9	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 8, 10]
71	α^8	α^{10}	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 8, 11]
72	α^8	α^{11}	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 8, 12]
73	α^8	α^{12}	2	1	[0, 1, 2, 1, 3, 16, 4, 34, 5, 48, 6, 58, 7, 48, 8, 29, 9, 16, 10, 5, 8, 13]
74	α^8	α^{13}	2	1	[0, 1, 2, 2, 3, 13, 4, 34, 5, 56, 6, 52, 7, 42, 8, 37, 9, 16, 10, 2, 11, 1, 8, 14]
75	α^8	α^{14}	2	3	[0, 1, 2, 3, 3, 13, 4, 30, 5, 56, 6, 58, 7, 42, 8, 33, 9, 16, 10, 3, 11, 1, 9, 10]
76	α^9	α^{10}	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 9, 11]
77	α^9	α^{11}	2	2	[0, 1, 2, 2, 3, 16, 4, 30, 5, 48, 6, 64, 7, 48, 8, 25, 9, 16, 10, 6, 9, 12]
78	α^9	α^{12}	2	2	[0, 1, 2, 2, 3, 15, 4, 32, 5, 50, 6, 58, 7, 48, 8, 31, 9, 14, 10, 4, 11, 1, 9, 13]
79	α^9	α^{13}	2	2	[0, 1, 2, 2, 3, 13, 4, 34, 5, 56, 6, 52, 7, 42, 8, 37, 9, 16, 10, 2, 11, 1, 9, 14]
80	α^9	α^{14}	2	3	[0, 1, 2, 3, 3, 10, 4, 36, 5, 58, 6, 48, 7, 48, 8, 35, 9, 14, 10, 5, 11]
81	α^{10}	α^{11}	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 10, 12]
82	α^{10}	α^{12}	2	3	[0, 1, 2, 3, 3, 15, 4, 28, 5, 50, 6, 64, 7, 48, 8, 27, 9, 14, 10, 5, 11, 1, 10, 13]
83	α^{10}	α^{13}	2	3	[0, 1, 2, 3, 3, 12, 4, 34, 5, 52, 6, 64, 7, 52, 8, 29, 9, 12, 10, 7, 10, 14]
84	α^{10}	α^{14}	2	3	[0, 1, 2, 3, 3, 10, 4, 36, 5, 58, 6, 48, 7, 48, 8, 35, 9, 14, 10, 5, 11, 12]
85	α^{11}	α^{12}	2	6	[0, 1, 2, 4, 3, 16, 4, 23, 5, 48, 6, 72, 7, 48, 8, 23, 9, 16, 10, 4, 12, 1, 11, 13]
86	α^{11}	α^{13}	2	4	[0, 1, 2, 4, 3, 14, 4, 26, 5, 52, 6, 64, 7, 48, 8, 29, 9, 12, 10, 4, 11, 2, 11, 14]
87	α^{11}	α^{14}	2	4	[0, 1, 2, 4, 3, 12, 4, 28, 5, 58, 6, 58, 7, 42, 8, 35, 9, 14, 10, 2, 11, 2, 12, 13]
88	α^{12}	α^{13}	2	6	[0, 1, 2, 6, 3, 14, 4, 22, 5, 52, 6, 64, 7, 48, 8, 33, 9, 12, 10, 2, 11, 2, 12, 14]
89	α^{12}	α^{14}	2	6	[0, 1, 2, 6, 3, 12, 4, 26, 5, 52, 6, 60, 7, 52, 8, 29, 9, 12, 10, 6, 13, 14]
90	α^{13}	α^{14}	2	4	[0, 1, 2, 8, 3, 10, 4, 28, 5, 50, 6, 50, 7, 62, 8, 35, 9, 6, 10, 6]

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

2.1 Linear codes with $d_{\mathcal{D}}^* = 3$

We focus on the the linear codes with *greater* $d_{\mathcal{D}}^*$, 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_prp = 3
for i in range(len(d_dual_b)):
    if d_dual_b[i][2] == d_D_prp:
        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)
```

10
[39 40 41 42 49 50 51 58 59 66]

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], '$B_{d_{\mathcal{D}}}$': B_all[d_index], 'Weight Enumerators': np.array(d_dual_b)[d_index]})
df_4 = df_4.sort_values(by=['$B_{d_{\mathcal{D}}}$'], ascending=True)

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

Out[10]:

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

	α_2	α_3	$d_{\mathcal{D}}$	$B_{d_{\mathcal{D}}}$	Weight Enumerators
5	α^0	α^{10}	3	10	[0, 1, 3, 16, 4, 39, 5, 48, 6, 48, 7, 48, 8, 39, 9, 16, 12, 1, 5, 11]
0	α^0	α^2	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 9]
1	α^0	α^3	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 10]
2	α^0	α^4	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 11]
3	α^0	α^5	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 12]
4	α^0	α^6	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 13]
6	α^0	α^7	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 4, 14]
7	α^0	α^8	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 5, 6]
8	α^0	α^9	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 5, 7]
9	α^0	α^{11}	3	17	[0, 1, 3, 17, 4, 38, 5, 44, 6, 52, 7, 54, 8, 33, 9, 12, 10, 4, 11, 1, 7, 12]

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

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

- Maximizing $d_{\mathcal{D}}^*$, here $\max\{d_{\mathcal{D}}^*\} = 3$
- Minimizing $B_{d_{\mathcal{D}}^*}$, here $\min\{B_{d_{\mathcal{D}}^*}\} = 16$

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 $(\alpha_1, \alpha_2, \alpha_3) \in \{(\alpha^0, \alpha^5, \alpha^{10})\}$. Note that permutation on three public points does not change the codes due to equivalence.

The generator matrix of the code is:

$$\mathbf{H}_{optimal} = (\alpha^0 \quad \alpha^5 \quad \alpha^{10}) \in \mathbb{F}_{2^4}^{1\times 3}, \mathbf{G} = \begin{pmatrix} 1&0&0&0&0&1&1&0&1&1&1&0 \\ 0&1&0&0&0&0&1&1&0&1&1&1 \\ 0&0&1&0&1&1&0&1&1&1&1&1 \\ 0&0&0&1&1&0&1&0&1&0&1&1 \end{pmatrix} \in \mathbb{F}_2^{4\times 12}$$

In [ ]:
```

```