Local Weight Distribution of the (256, 93) Third-Order Binary Reed-Muller Code

Kenji Yasunaga (Osaka Univ.)
Toru Fujiwara (Osaka Univ.)
Tadao Kasami (NAIST)

HISC2006, Nara, Japan

Local Weight Distribution (LWD)

- Is the weight distribution of *minimal codewords* in a code.
  - Studies of minimal codewords are crucial for ML performance analysis of the code.
- Gives a tighter upper bound than the usual union bound.
  - The union bound uses the *(global) weight distribution*.
- Determines the complexity of gradient-like decoding.
  - Gradient-like decoding is one of the nearest codeword decoding.

Minimal Codeword

$\boldsymbol{v}$ is a minimal codeword in C.
$\Leftrightarrow$ C does not contain $\boldsymbol{v}_1, \boldsymbol{v}_2 \in C$ such that
$\boldsymbol{v} = \boldsymbol{v}_1 + \boldsymbol{v}_2$, $\mathrm{Supp}(\boldsymbol{v}_1) \cap \mathrm{Supp}(\boldsymbol{v}_2) = \phi$.

$\mathrm{Supp}(\boldsymbol{v}) := \{ i : v_i \neq 0$ for $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \}$

Ex.) If C contains $\boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{v}_2,$

$$\boldsymbol{v} = (1, 1, 1, 1)$$
$$\boldsymbol{v}_1 = (1, 1, 0, 0)$$
$$\boldsymbol{v}_2 = (0, 0, 1, 1)$$

$\Rightarrow$ $\boldsymbol{v}$ is not a minimal codeword in C.

Previous Results for LWD

- Codes completely determined:
  - Hamming codes [Ashikhmin and Barg, IEEE Trans. IT 1998]
  - 2nd-order Reed-Muller codes [Ashikhmin and Barg, IEEE IT 1998]
- Codes obtained by computation:
  - BCH codes of length 63 [Mohri et al., IEICE Trans. Fund. 2003]
  - (128, k) extended BCH codes of $k \leq 50$ [Yasunaga and Fujiwara, ISITA2004]
  - (128, 64) 3rd-order Reed-Muller code [Yasunaga and Fujiwara, IEICE Tech. Rep. 2004]

Our Results

- LWD of (256, 93) 3rd-order Reed-Muller code is obtained by computation.
  - By using a *modified* coset partitioning algorithm.
    - Coset partitioning algorithm is useful for codes closed under large automorphism group (e.g. extended BCH, Reed-Muller).
      → (128, k) extended BCH and (128, 64) Reed-Muller.
    - Modification is to use *binary shifts* and applicable to Reed-Muller codes.
    - Computation complexity is reduced to 1/256.

Coset Partitioning Algorithm for Computing LWD of C

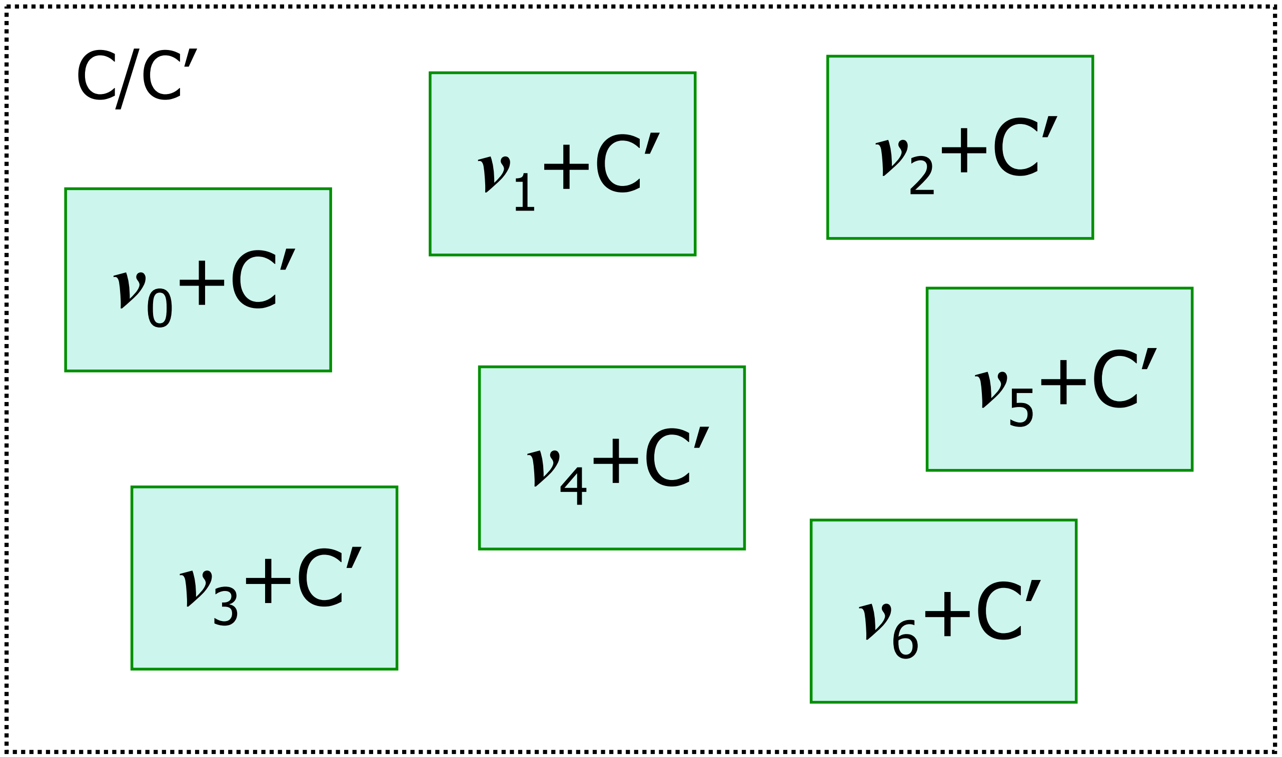
1. Select C′ as a subcode of C.
2. Partition C/C′ into equivalence classes.
3. Compute LWSDs* for representative cosets.

⇒ Let's see more details …

* LWSD (Local weight subdistribution) for a coset:
  The weight distribution of minimal codewords in the coset.

Coset Partitioning Algorithm:
1. Select C′ as a subcode of C

- C can be seen as the set of cosets of C′ (denoted by C/C′).



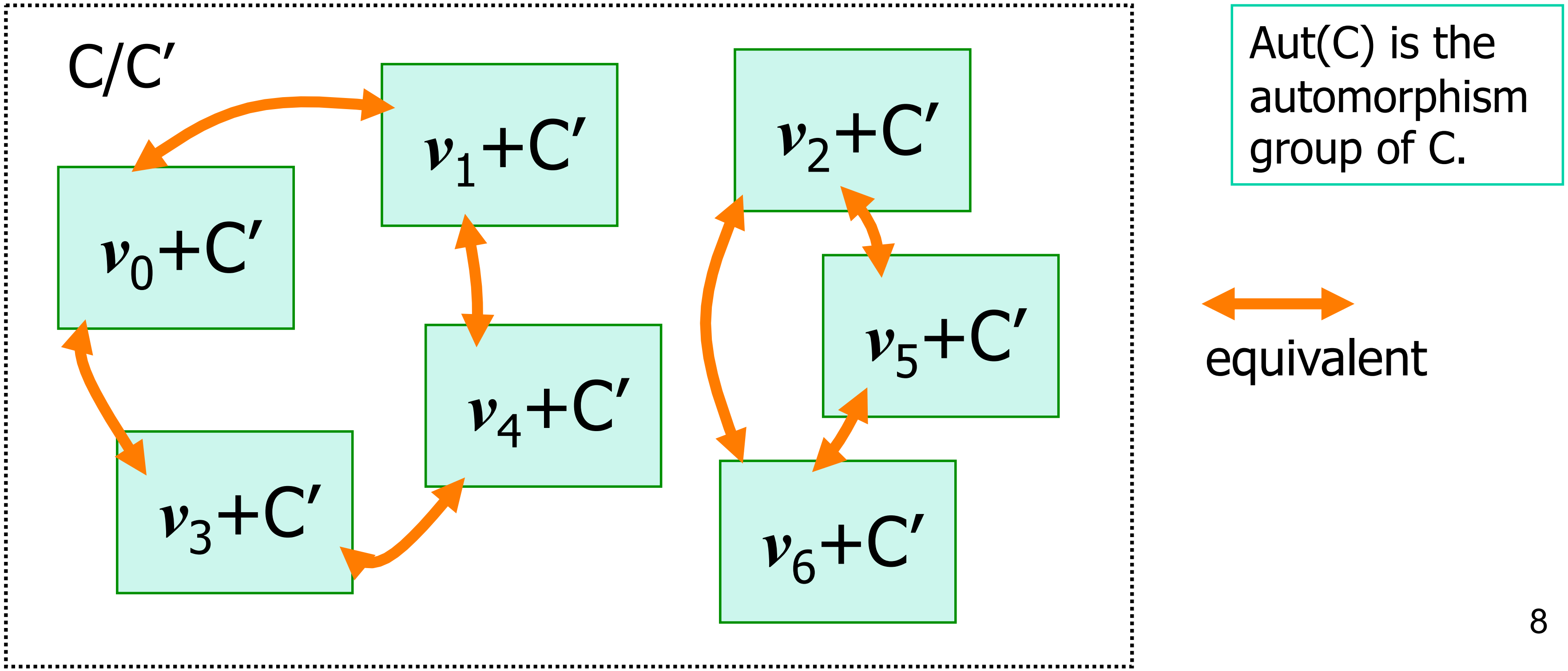
v_i+C′ : coset

v_i : coset leader

Coset Partitioning Algorithm:
2. Partition C/C′ into equivalence classes

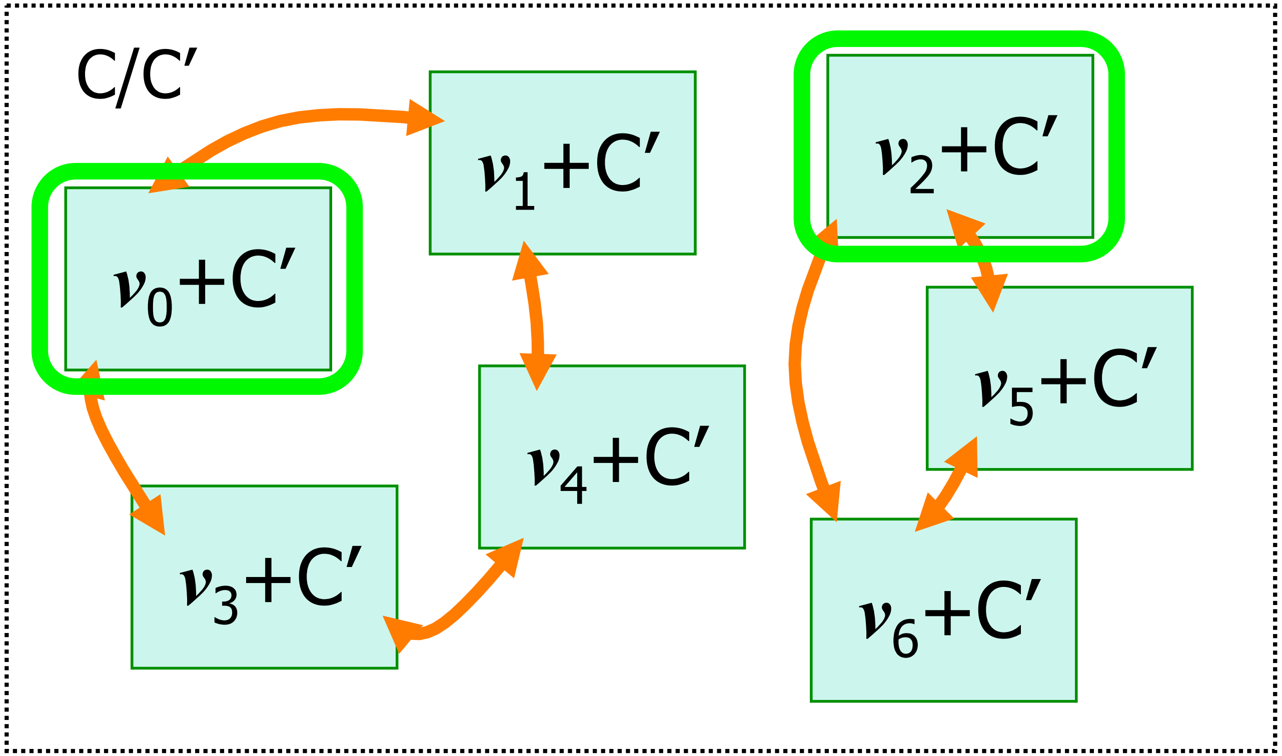
- v_1+C′ and v_2+C′ are equivalent.
  ⇔ There exists π such that $\pi v_1 \in v_2$+C′, π∈ Aut(C) ∩ Aut(C′).
  ⇔ LWSDs for v_1+C′ and v_2+C′ are the same.

- This algorithm works effectively if Aut(C) ∩ Aut(C′) is large.



Coset Partitioning Algorithm:
3. Compute LWSDs for representative cosets.

- Need to compute LWSDs only for representative cosets.
  → LWD of C is determined.



Computing LWSDs only for two cosets leads LWD of C.

Recursive Use of Coset Partitioning Algorithm

- Coset partitioning algorithm can be used for computing LWSDs for cosets (not only LWD of C).

To compute LWSD of v+C′ ∈ C/C′

1. Select C″ as a subcode of C′.
2. Partition (v+C′)/C″ into equivalence classes*.
3. Compute LWSDs for representative cosets.

* {π: πv ∈ v+C′, π∈ Aut(C) ∩ Aut(C′) ∩ Aut(C″) } is used for partitioning cosets into equivalence classes.
Not all the permutations in Aut(C) ∩ Aut(C′) ∩ Aut(C″).

In Computing LWD of (256, 93) Reed-Muller Code

- RM(r,m) : r-th order RM code of length 2^m
  - RM(3,8) = (256, 93) Reed-Muller

- C : RM(3,8), C′ : RM(2,8), C″ :RM(1,8)
  - RM(2,8) = (256, 37) Reed-Muller
  - RM(1,8) = (256, 9) Reed-Muller

- Result for partitioning RM(3,8)/RM(2,8) into equivalence classes is known [Hou, Discr. Math, 1996].
  ⇒ Partitioned into 32 equivalence classes.

⇒ Need to compute LWSDs for 32 representative cosets.

Computation time for each coset will be large (3000 hours with 2GHz Pentium4). → Recursive use of the algorithm.

In Computing LWSD for $\boldsymbol{v}$+RM(2,8) ∈ RM(3,8)/RM(2,8)

- We recursively use coset partitioning algorithm.

- To partition ($\boldsymbol{v}$+RM(2,8))/RM(1,8) into equivalence classes, we need a set of permutations {π: π$\boldsymbol{v}$ ∈ $\boldsymbol{v}$+RM(2,8), π∈ GA(8) }.
  - GA(m) is the general affine group, and the automorphism group of RM(r,m).

- We find a candidate for such permutations,
  ⇒ *binary shifts*.

Reed-Muller Code; RM(r,m)

- Any binary vector of length 2^m has one-to-one correspondence with Boolean polynomial of m variables $(x_1, x_2, \ldots, x_m)$.

m-variable Boolean polynomial $f(x_1, x_2, \ldots, x_m)$	⟺	Binary vector of length 2^m $\boldsymbol{v} = (v_1, v_2, \ldots, v_{2^m})$ $\boldsymbol{v}$ consists of all 2^m arguments' truth evaluation of $f()$ (the truth table of $f()$).

Ex.) $f \in$ RM(2,2)

$$f = x_1 + x_2 \quad \Leftrightarrow \quad \boldsymbol{v} = (0+0,\ 1+0,\ 0+1,\ 1+1) = (0,\ 1,\ 1,\ 0)$$

- r-th order Reed-Muller code of length 2^m :
  RM(r,m) = { m-variable Boolean polynomials with degree at most r}

General Affine Group; GA(m)

- GA(m) : The set of transformation T for m-variable polynomials $f(x_1, \ldots, x_m)$.

$$\text{T : replace} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \text{ by } A \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} + b$$

  A is an invertible m×m matrix, b is a binary m-tuple.

- Since T does not increase the degree of polynomials, GA(m) is the automorphism group of RM(r,m).

- When A is the identity matrix, GA(m) is called *binary shifts*, denoted by BS(m).
  - π∈BS(m) replaces each x_i by x_i+b_i, $b_i=\{0, 1\}$.

⇒ Return to our subject …

In Computing LWSD for $\boldsymbol{v}$+RM(2,8) $\in$ RM(3,8)/RM(2,8)

- We need a set of permutations {π: π$\boldsymbol{v}$ $\in$ $\boldsymbol{v}$+RM(2,8), π$\in$ GA(8) } in order to partition ($\boldsymbol{v}$+RM(2,8))/RM(1,8) into equivalence classes.

- BS(m) is a candidate for such permutations.
  - For any coset leader $\boldsymbol{v}$, the degree of $\boldsymbol{v}$ is 3.
  - For π$\in$BS(8), the degree 3 Boolean polynomials contained in π$\boldsymbol{v}$ is only $\boldsymbol{v}$.
    $\Rightarrow$ π$\boldsymbol{v}$ $\in$ $\boldsymbol{v}$+RM(2,8).

Ex.) $\boldsymbol{v} = x_1x_2x_3.$

$\pi\boldsymbol{v} = (x_1+b_1)(x_2+b_2)(x_3+b_3)$

$= x_1x_2x_3 +$ (Boolean polynomial with degree at most 2).

$\in$ $\boldsymbol{v}$+RM(2,8)

π$\in$BS(m) replaces x_i by x_i+b_i, $b_i=\{0,1\}$.

In Computing LWSD for v+RM(2,8) ∈ RM(3,8)/RM(2,8)

- Let $C_{BS}(v)$ = { πv : $\pi \in$ BS(m) }.

Theorem 4: Linearity of $C_{BS}(v)$.
Let f be an r-th order Boolean polynomial.
For a coset f+RM(r-1,m), $C_{BS}(f)$ is a linear subspace of f+RM(r-1,m).

Lemma 4: Bases of $C_{BS}(v)$.
Let $\pi_i \in$ BS(m) be the permutation that only replaces x_i by x_i+1.
For a coset f+RM(r-1,m), $\pi_i f$ for $1 \le i \le$ m are bases of $C_{BS}(f)$.

Lemma 5: Equivalence of LWSDs for $v + v_1$+$C_{BS}(v)$+RM(r-2,m).
For v+RM(r-1,m)∈RM(r,m)/RM(r-1,m),
let $v+v_1$+RM(r-2,m) ∈ (v+RM(r-1,m)/RM(r-2,m)).
LWSD of $v + v_1$+RM(r-2,m) and LWSD of $v + v_1 + u$ +RM(r-2,m) for any $u \in C_{BS}(v)$ are the same.

In Computing LWSD for v+RM(2,8) ∈ RM(3,8)/RM(2,8)

- From the last lemma, each coset in (v+RM(2,8))/RM(1,8) has $|C_{BS}(v)|$ equivalent cosets.

  ⇒ Computation complexity for computing LWSD for v+RM(2,8) will be reduced to $1/|C_{BS}(v)|$.

- $|C_{BS}(v)| = 2^{\dim(C_{BS}(v))}$.
  - Clearly, $\dim(C_{BS}(v)) \le 8$ for v+RM(2,8)∈RM(3,8)/RM(2,8).
  - $\dim(C_{BS}(v))$ is obtained by investigating the number of independent vectors in bases of $C_{BS}(v)$.

dim($C_{BS}(\boldsymbol{v})$) for 32 representative cosets $\boldsymbol{v}$ +RM(2,8)∈RM(3,8)/RM(2,8)

- For 32 representative cosets $\boldsymbol{v}_i$+RM(2,8)∈RM(3,8)/RM(2,8), $1 \le i \le 32$,

$$\dim(C_{BS}(\boldsymbol{v}_i)) = \begin{cases} 0 & \text{for } i = 1, \\ 3 & \text{for } i = 2, \\ 5 & \text{for } i = 3, \\ 6 & \text{for } i = 4, 5, 6, \\ 7 & \text{for } i = 7, 8, \ldots, 12, \\ 8 & \text{for } i = 13, 14, \ldots, 32. \end{cases}$$

- For most cosets, dim($C_{BS}(\boldsymbol{v}_i)$) is 7 or 8, and thus the complexity is reduced to 1/128 or 1/256.

- For $i = 1, 2, 3$, binary shift method is not effective.

  ⇒ We take another method.

  Investigate the minimality of codewords in the cosets from the coset leaders.

Minimal codewords in $\boldsymbol{v}_i$+RM(2,8) for $i = 1, 2, 3$

- $i = 1,\ \boldsymbol{v}_1 = 0$
  - Any codeword in $\boldsymbol{v}_1$+RM(2,8) is not minimal in RM(3,8).

- $i = 2,\ \boldsymbol{v}_2 = x_1x_2x_3$
  - All minimal codewords are of the form $(x_1+a_1)(x_2+a_2)(x_3+a_3)$, $a_i = \{0, 1\}$.
    ⇒ These codewords have the minimum weight.
    Then there is 8 minimal codewords in $\boldsymbol{v}_2$+RM(2,8).

- $i = 3,\ \boldsymbol{v}_3 = x_1x_2x_3+x_2x_4x_5$
  - All minimal codewords are of the form $x_2((x_1x_3+x_4x_5)+g)$ or $(x_2+1)(x_1x_3+x_4x_5)+g)$ where g is a 1st order Boolean polynomial.
    ⇒ Checking minimality for all 2^{m+1} patterns leads LWSD of $\boldsymbol{v}_3$+RM(2,8).

Determination of LWDs for 32 representative cosets in RM(3,8)/RM(2,8)

- For $\boldsymbol{v}_i$+RM(2,8) of $i = 1, 2, 3$, we determined LWDs by investigating minimality of codewords from the coset leaders.
  Note: [Borissov and Manev, Serdica, 2004] derived the same results as this.

- For the other cosets, we compute LWDs by using binary shift method.

LWD of (256,93) Reed-Muller Code

weight	#(*minimal codewords*)
32	777 240
48	2 698 577 280
56	304 296 714 240
64	74 957 481 580 800
68	707 415 842 488 320
72	28 055 013 884 190 720
76	764 244 915 168 215 040
80	20 661 780 862 988 697 600
84	414 411 510 493 363 568 640
88	6 266 129 424 660 312 883 200
92	71 773 299 826 457 585 909 760
96	627 671 368 441 418 233 282 560
100	4 208 996 769 021 096 823 357 440

weight	#(*minimal codewords*)
104	21 729 928 024 588 603 285 831 680
108	86 666 048 822 136 825 068 912 640
112	267 785 773 787 841 625 294 110 720
116	642 456 218 534 940 726 012 149 760
120	1 198 819 482 820 829 207 341 301 760
124	1 741 767 435 501 050 021 239 848 960
128	1 971 038 877 022 035 145 182 412 800
132	1 735 627 864 909 747 949 509 017 600
136	1 184 951 930 170 762 649 130 762 240
140	620 824 077 435 771 999 611 781 120
144	242 710 219 348 184 804 622 336 000
148	65 293 324 137 047 881 521 561 600
152	8 982 921 659 842 430 396 006 400

Conclusions

- We obtained LWD of the (256,93) 3rd-order Reed-Muller code.
  - Using a modified coset partitioning algorithm.
    - We recursively use coset partitioning algorithm for computing LWSD for representative cosets.
    - Modification is to use BS(m) (binary shifts) in GA(m), and applicable to Reed-Muller codes.
    - Computation complexity of LWSD is reduced to 1/256 for most representative cosets in RM(3,8)/RM(2,8).