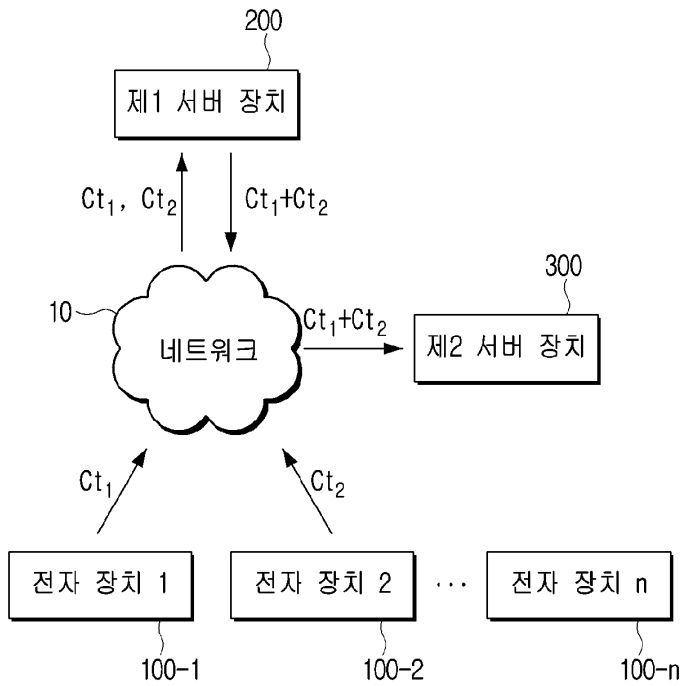
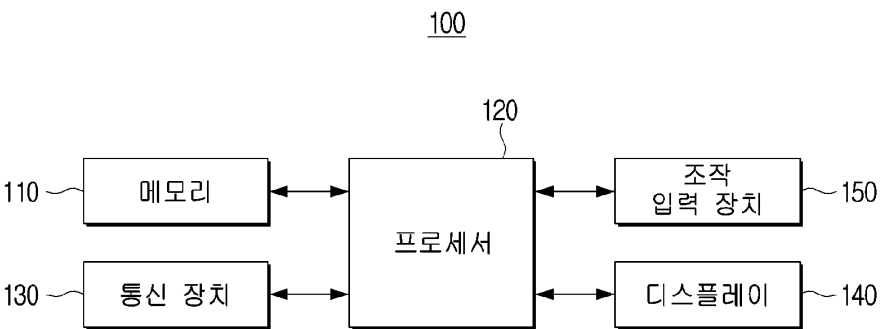


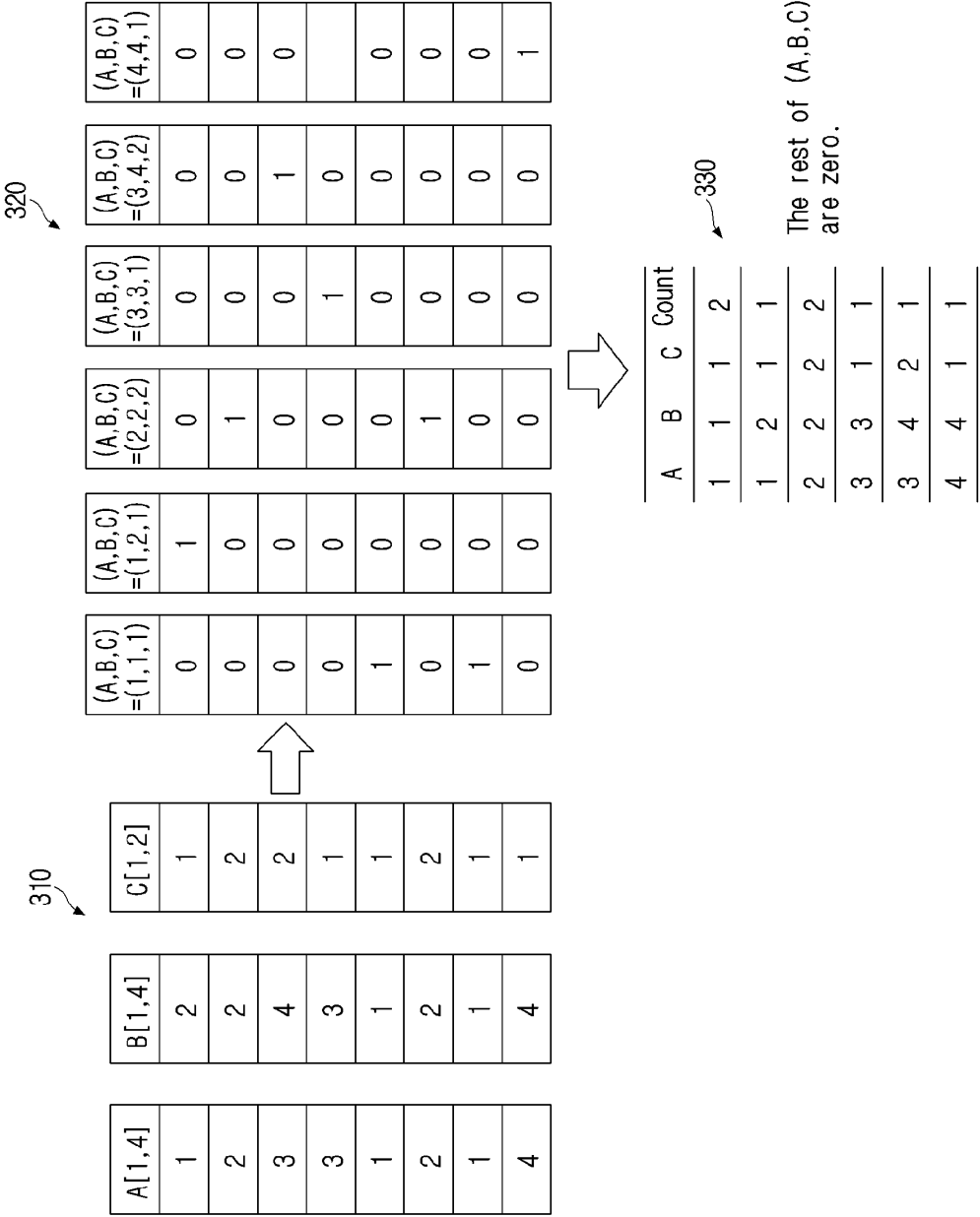
[도1]



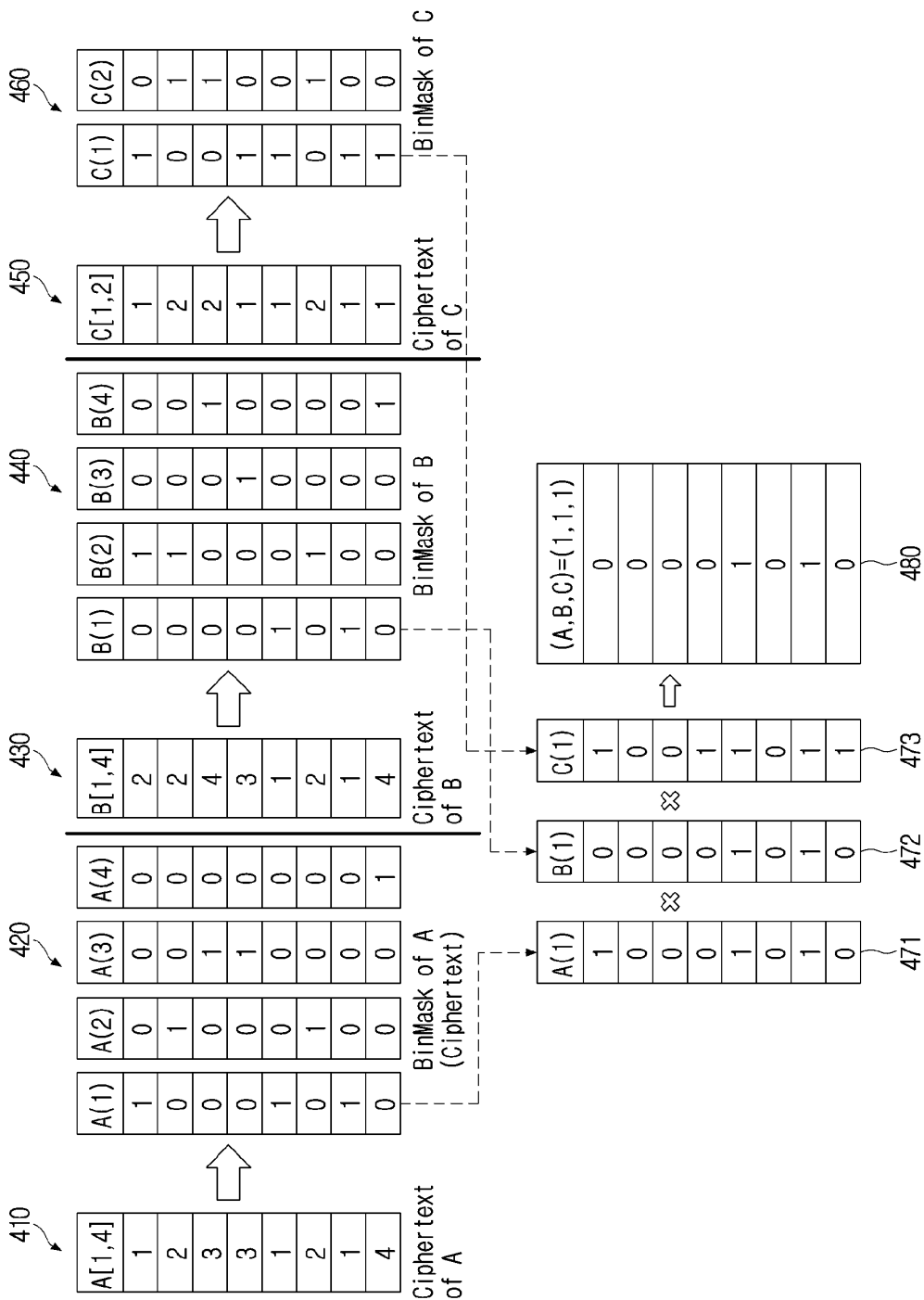
[도2]



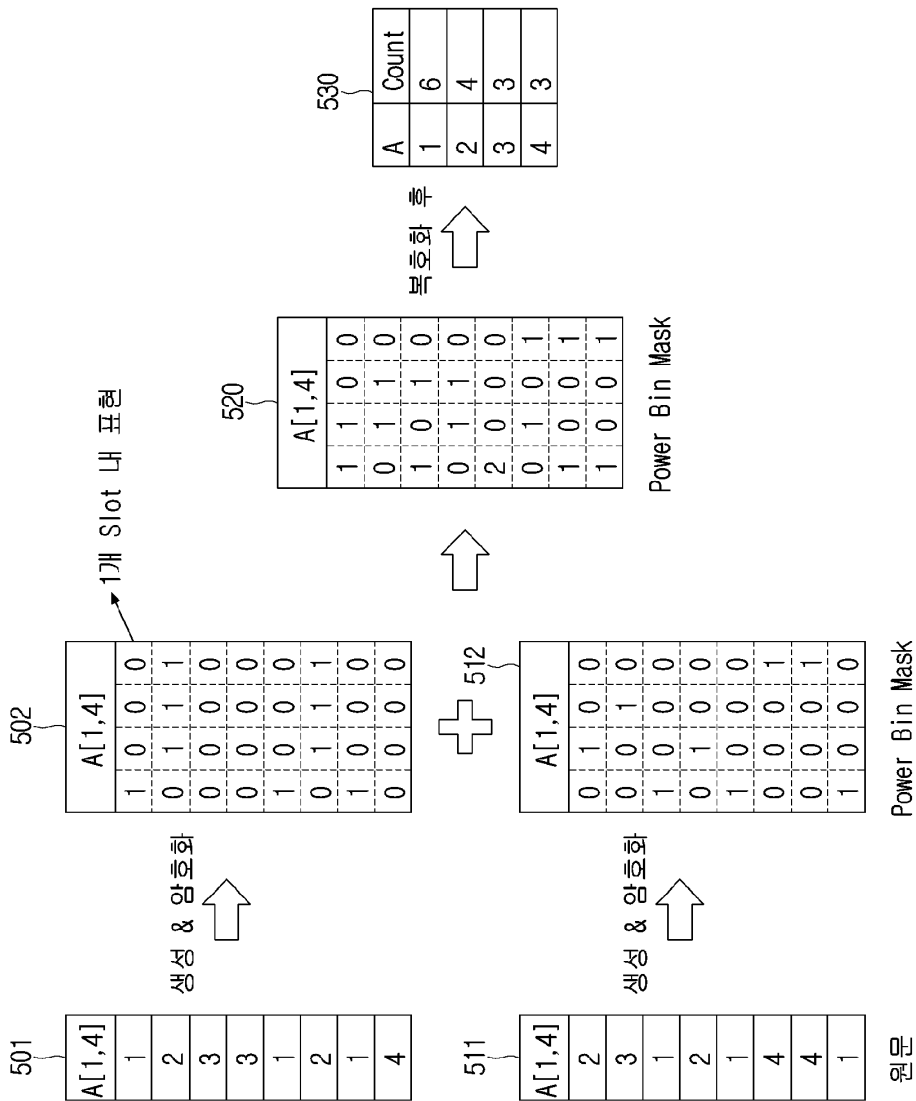
[도3]



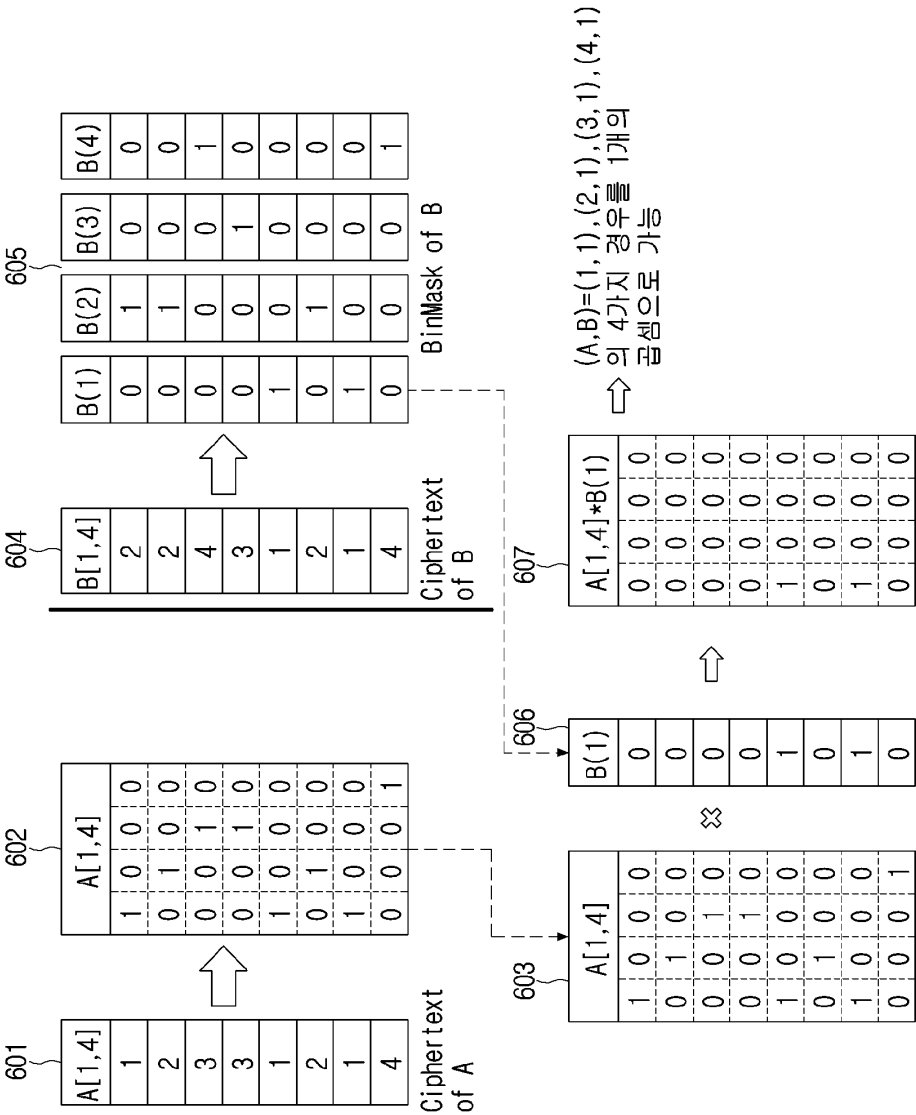
[도4]



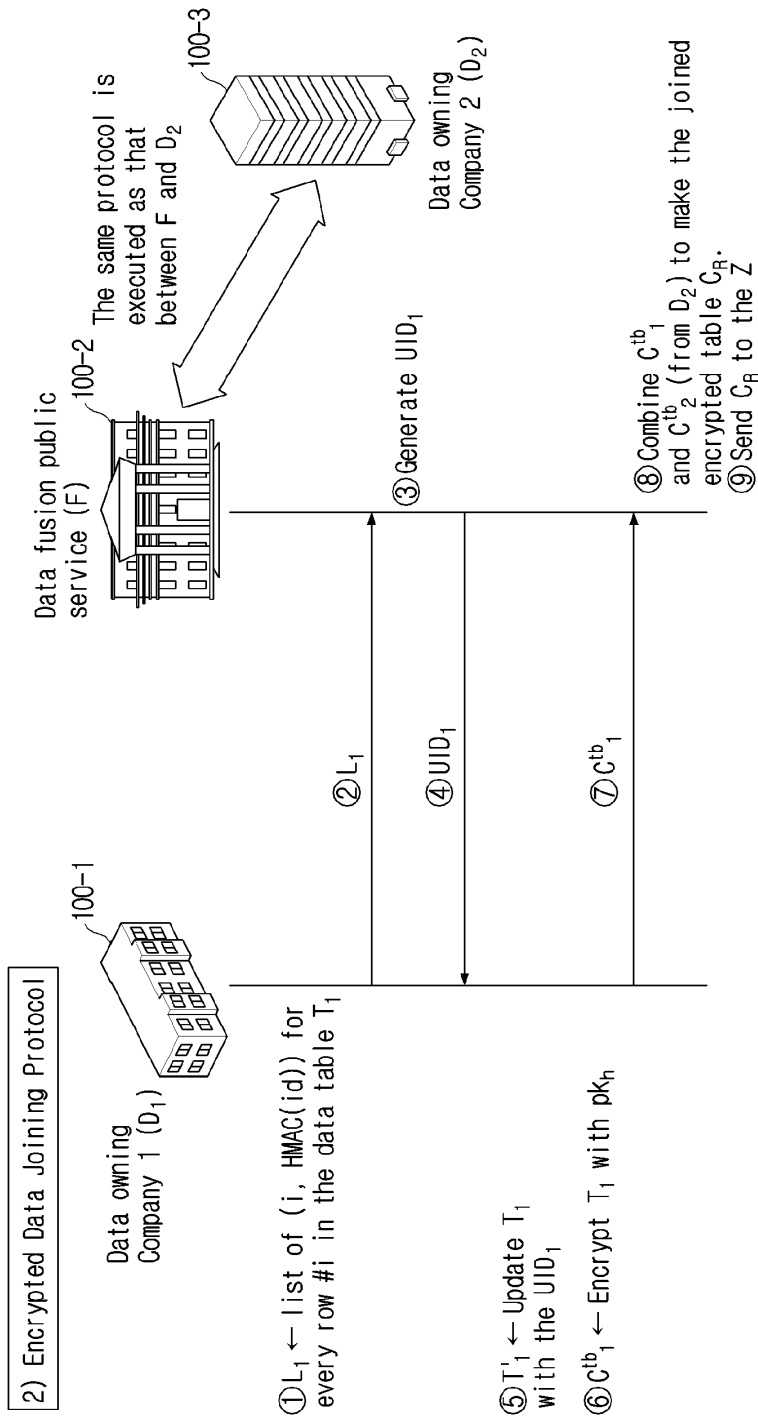
[도5]



[도6]



[도7]



[도 8]

나이	지역	신용등급		나이	지역	신용등급
20대 이하	서울	1	Encoding →	1	1	1
30대~50대	서울	2		2	1	2
60대 이상	광역시	2		3	2	2
30대~50대	도	3		2	3	3
20대 이하	도	2		1	3	2
60대 이상	광역시	3		3	2	3
30대~50대	서울	2		2	1	2



60대 이상+광역시 인 사람 수→2(Bin count)
 서울 사람의 신용등급 평균→1.66(Bin average)
 30~50대의 신용등급 분산→1.89(Bin variance)

[도 9]

900

```

1: procedure GENBINMASK(x, v,  $\tilde{f}$ ,  $\tilde{s}$ )
2:    $n_b \leftarrow \lceil n/M \rceil$ 
3:    $m \leftarrow \text{length}(\tilde{f})$ 
4:   for  $i \leftarrow 0, m-1$  do
5:     for  $j \leftarrow 0, n_b-1$  do
6:       for  $k \leftarrow 1, s_i$  do
7:          $\text{tmp} \leftarrow 0^M$ 
8:         for  $l \leftarrow 0, M-1$  do
9:           if  $\tilde{x}_j^{(fi)}(l) == k$  &  $\tilde{v}_j^{(fi)}(l) == 1$  then
10:             $\text{tmp}[l] \leftarrow 1$ 
11:          $b_{k,j}^{(fi)} \leftarrow \text{Enc}_{pk}(\text{tmp})$ 
12:   return b

```

[도 10]

1000

```

1: procedure GENBINMASK(x,  $\bar{f}$ ,  $\bar{s}$ )
2:    $n_b \leftarrow \lceil n/M \rceil$ 
3:    $m \leftarrow \text{length}(\bar{f})$ 
4:   for  $i \leftarrow 0, m-1$  do
5:     for  $j \leftarrow 0, n_b-1$  do
6:       for  $k \leftarrow 1, s_j$  do
7:          $\text{tmp} \leftarrow 0^M$ 
8:         for  $l \leftarrow 0, M-1$  do
9:           if  $\bar{x}_j^{(fi)}(l) = k$  then
10:             $\text{tmp}[l] \leftarrow 1$ 
11:             $b_{k,j}^{(fi)} \leftarrow \text{Enc}_{pk}(\text{tmp})$ 
12:   return b

```

[도 11]

1100

```

1: procedure DISCRETEEQUALZERO(x,  $\mu$ , K)
2:    $d \leftarrow \lceil \log_2 \mu \rceil$ 
3:    $y \leftarrow \text{Mult}_{evk}(x, 1/2^d)$ 
4:
5:   declare pows[K+1]                                ▷ array of ciphertexts
6:    $\text{pows}[0] \leftarrow \text{Enc}_{pk}((1^M)^T)$ 
7:    $\text{pows}[1] \leftarrow y$ 
8:   for  $i \leftarrow 2, K$  do
9:      $\text{pows}[i] \leftarrow \text{Mult}_{evk}(\text{pows}[\lceil i/2 \rceil], \text{pows}[\lfloor i/2 \rfloor])$ 
10:    $s \leftarrow \text{Enc}_{pk}((0^M)^T)$ 
11:    $c \leftarrow \text{Enc}_{pk}((0^M)^T)$ 
12:   for  $i \leftarrow 0, K$  do
13:      $s \leftarrow \text{Add}(s, \text{mult}_{evk}(a_i, \text{pow}[i]))$ 
14:      $c \leftarrow \text{Add}(s, \text{mult}_{evk}(b_i, \text{pow}[i]))$ 
15:   ▷  $s \approx \text{sinc}(x/2^d)$ ,  $c \approx \cos(\pi x/2^d)$ 
16:
17:    $s \leftarrow \text{Mult}_{evk}(s, c)$ 
18:   for  $i \leftarrow 0, d$  do
19:      $c \leftarrow \text{Mult}_{evk}(c, c)$ 
20:      $c \leftarrow \text{Mult}_{evk}(c, 2)$ 
21:      $c \leftarrow \text{Sub}(c, 1)$ 
22:      $s \leftarrow \text{Mult}_{evk}(s, c)$ 
23:   return s

```

[도 12]

1200

```

1: procedure GENBINMASK(X, V,  $\vec{f}$ ,  $\vec{s}$ )
2:    $n_b \leftarrow \lceil n/M \rceil$ 
3:    $m \leftarrow \text{length}(\vec{f})$ 
4:   for  $i \leftarrow 0, m - 1$  do
5:     for  $j \leftarrow 0, n_b - 1$  do
6:       for  $k \leftarrow 1, s_i$  do
7:          $\text{tmp} \leftarrow \text{Sub}(X_j^{(fi)}, k)$ 
8:          $\text{tmp} \leftarrow \text{DiscreteEqualZero}(\text{tmp}, s_j, K)$ 
9:         if v-bit table exists, then
10:           $\text{tmp} \leftarrow \text{Mult}_{\text{evk}}(\text{tmp}, V_j^{(fi)})$ 
11:           $b_{k,j}^{(fi)} \leftarrow \text{tmp}$ 
12:   return b

```

[도 13]

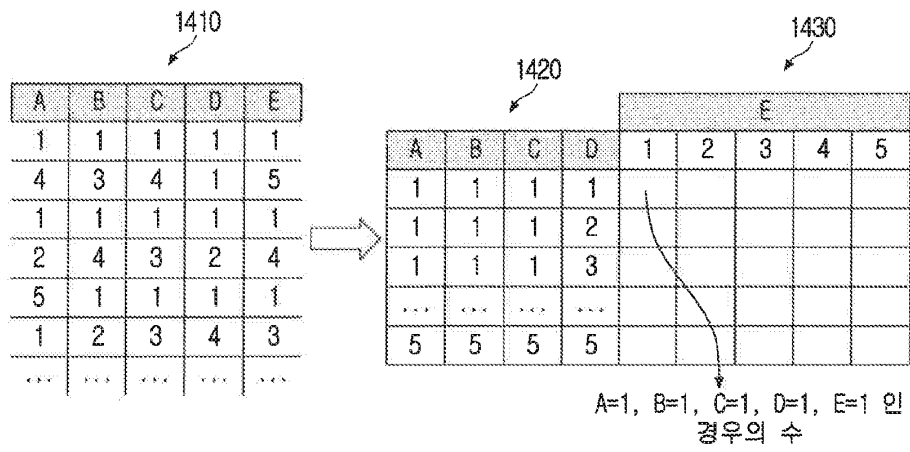
1300

```

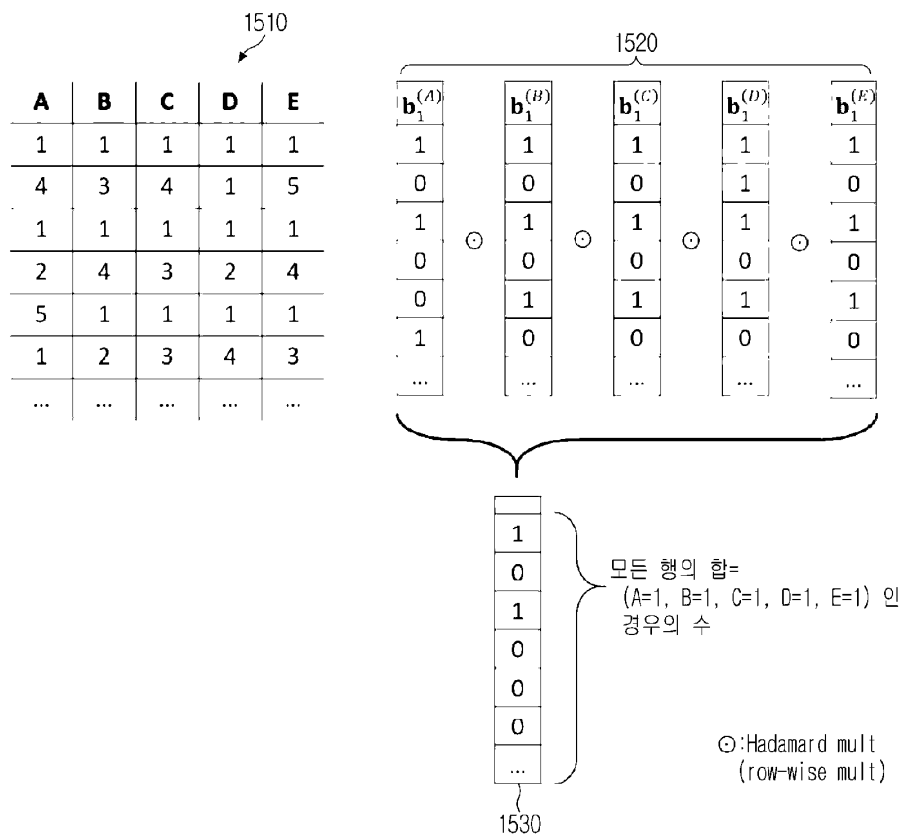
1: procedure BINCOUNT(b,  $\vec{f}$ ,  $\vec{c}$ )
2:    $n_b \leftarrow \lceil n/M \rceil$ 
3:    $m \leftarrow \text{length}(\vec{f})$ 
4:    $\text{sum} \leftarrow \text{Enc}_{\text{pk}}((O^M)^T)$ 
5:   for  $i \leftarrow 0, n_b - 1$  do
6:      $\text{tmp} \leftarrow b_{\text{co},i}^{(fo)}$ 
7:     for  $j \leftarrow 1, m - 1$  do
8:        $\text{tmp} \leftarrow \text{Mult}_{\text{evk}}(\text{tmp}, b_{c,j,1}^{(fi)})$ 
9:        $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{tmp})$ 
10:   for  $i \leftarrow 0, \log_2(M) - 1$  do
11:      $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{Rot}_{\text{evk}}(\text{sum}, 2^i))$ 
12:   return sum

```

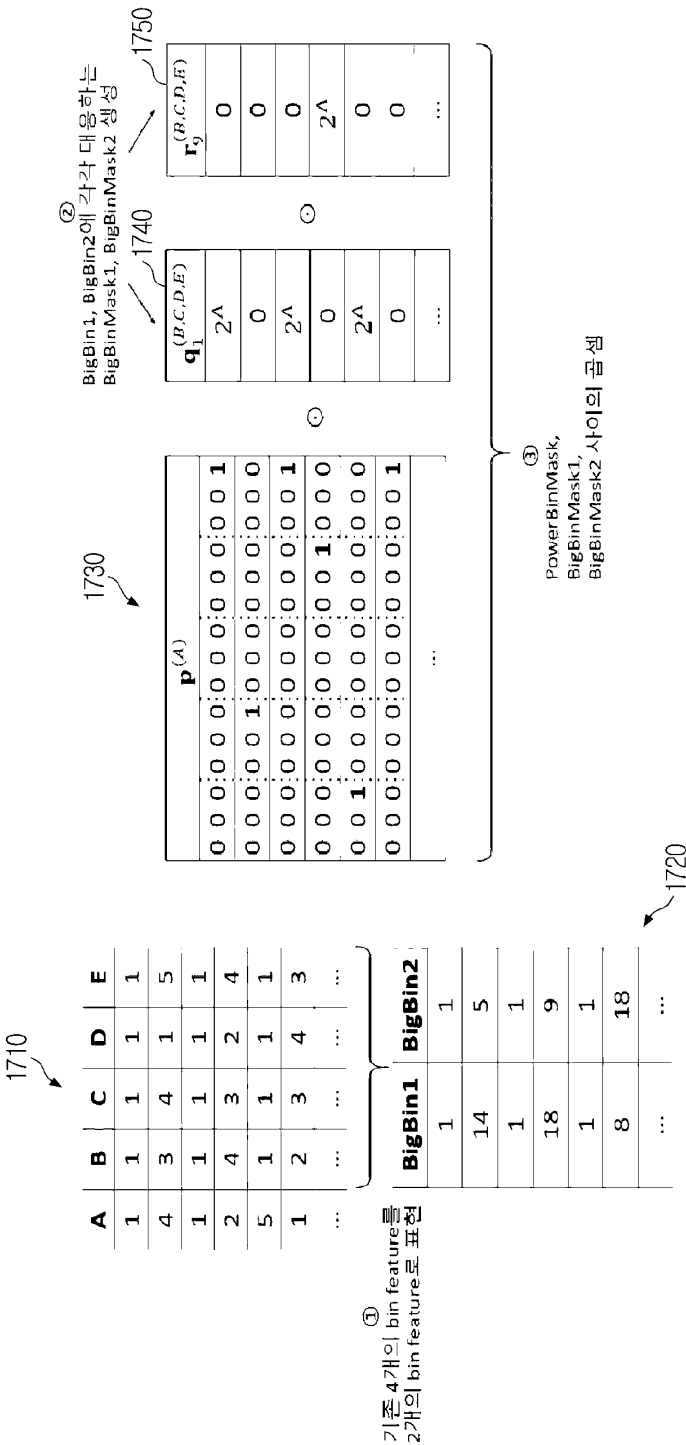
[도 14]



[도15]



[도17]



[도 18]

1800

```

1: procedure GENPOWERBINMASK(x, v, fo)
2:    $n_b \leftarrow \lceil n/M \rceil$ 
3:   for  $i \leftarrow 0, n_b - 1$  do
4:      $\bar{p} \leftarrow (0^M)^T$ 
5:     for  $j \leftarrow 0, M - 1$  do
6:       if  $\bar{v}^{(fo)}(i * M + j) == 1$  then
7:          $\bar{p}(j) \leftarrow \text{Pow}(2, \Delta \cdot (\bar{x}^{(fo)}(i * M + j) - 1) + \delta)$ 
8:        $p_i \leftarrow \text{Enc}_{sk}(\bar{p})$ 
9:   return p

```

[도 19]

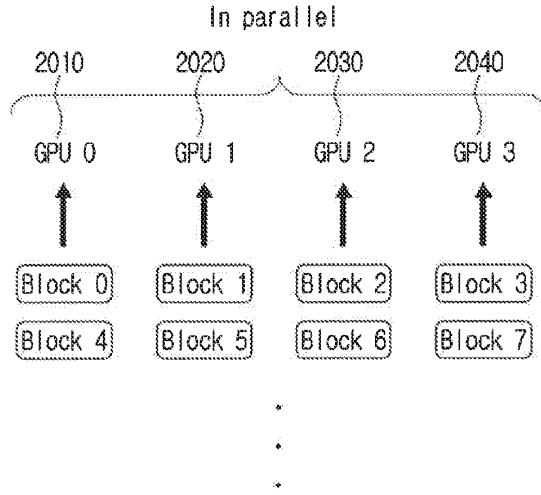
1900

```

1: procedure GENBIGBINMASK(x, v, (f1, ..., fm-1), (s1, ..., sm-1))
2:    $n_b \leftarrow \lceil n/M \rceil$ 
3:    $Q \leftarrow \lceil \sqrt{\prod_{i=1}^{m-1} (s_i)} \rceil$ 
4:   for  $i \leftarrow 0, n_b - 1$  do
5:      $\vec{q}^{(1)}, \dots, \vec{q}^{(Q)} \leftarrow (0^M)^T$ 
6:      $\vec{r}^{(1)}, \dots, \vec{r}^{(Q)} \leftarrow (0^M)^T$ 
7:     for  $j \leftarrow 0, M - 1$  do
8:       if  $\bar{v}_i^{(f_l)}(j) == 1$  for  $l = 0, 1, \dots, m - 1$  then
9:         ▷ if every valid bits are one,
10:          tmp ← 0
11:          for  $l \leftarrow 0, m - 1$  do
12:            tmp ←  $s_l * \text{tmp} + (\bar{x}_i^{(f_l)}(j) - 1)$ 
13:           $q' \leftarrow \lfloor \text{tmp}/Q \rfloor + 1$ 
14:           $r' \leftarrow \text{tmp} \% Q + 1$ 
15:           $\vec{q}^{(q')}(j) \leftarrow 2^j$ 
16:           $\vec{r}^{(r')}(j) \leftarrow 2^j$ 
17:       for  $j \leftarrow 1, Q$  do
18:          $q_i^{(j)} \leftarrow \text{Enc}_{pk}(\vec{q}^{(j)})$ 
19:          $r_i^{(j)} \leftarrow \text{Enc}_{pk}(\vec{r}^{(j)})$ 
20:   return  $q^{(i)}, r^{(i)}$  ( $i = 1, 2 \dots Q$ )

```

[도 20]



[도 21]

2100

```

1: procedure MULTPERGPU(p, q, r, curr_gpu_idx, Ng)
2:   l ← current_gpu_idx
3:   while l < ⌈n/M⌉ do
4:     a ← 0
5:     while a < Q do
6:       for i ← 1, c do
7:         ql(a+i) ← Multevk(pl, ql(a+i))
8:         b ← 0
9:         while b < Q do
10:          for j ← 1, c do
11:            tmp ← Multevk(ql(a+i), rl(b+j))
12:            if l == current_gpu_idx then
13:              rescurrent_gpu_idx(a+i, b+j) ← tmp
14:            else
15:              rescurrent_gpu_idx(a+i, b+j) ← Add(tmp, rescurrent_gpu_idx(a+i, b+j))
16:          b ← b + c
17:        a ← a + c
18:      l ← l + Ng
19:    return {rescurrent_gpu_idx(i,j) | i, j = 1, 2, ..., Q}
20:
21: procedure LARGE BIN COUNT MULT(p, q, r, Ng)
22:   mres ← {}
23:   for curr_gpu_idx ← 0, Ng - 1 do
24:     mres ← mres ∪ MULTPERGPU(p, q, r, curr_gpu_idx, Ng)
25:   return mres = {resk(i,j) | k = 0, 1, ..., Ng - 1, i, j = 1, 2, ..., Q}

```

[도 22]

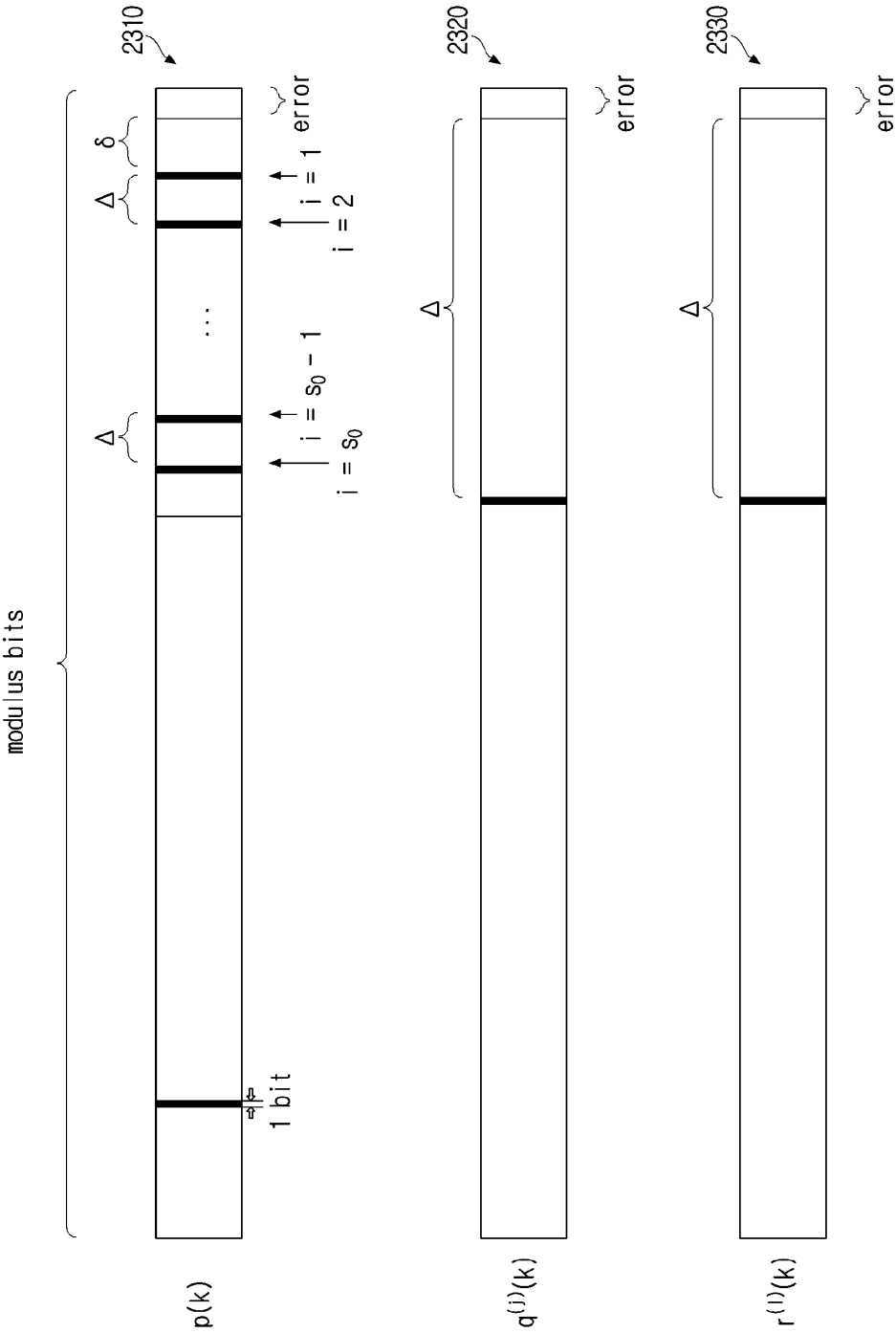
2200

```

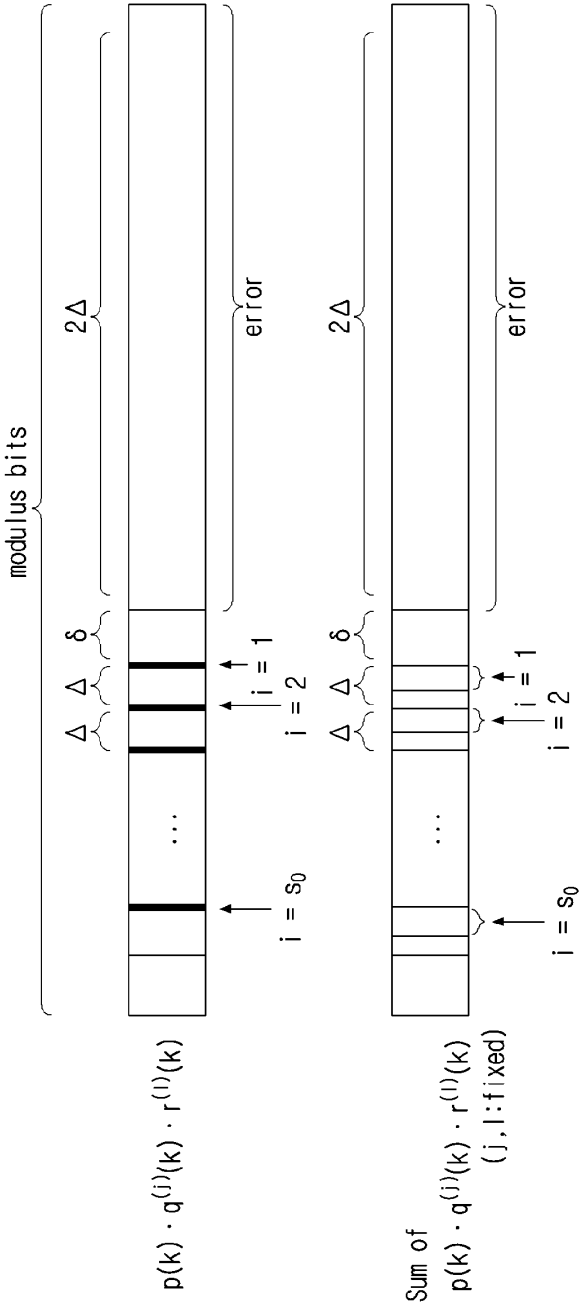
1: procedure LARGEBINCOUNTDEC(mres)
2:   arr ← zeros((Q2, s0))    ▷ an (Q2 * s0)-size array filled with zeros
3:   for i ← 1, Q do
4:     for j ← 1, Q do
5:       for k ← 0, Ng - 1 do
6:          $\vec{res} \leftarrow Dec_{sk}(res_k^{(i,j)})$ 
7:         for s ← 0, M - 1 do
8:           tmp ←  $\vec{res}(s)$ 
9:           tmp ← tmp / (2 ·  $\wedge$  +  $\delta$ )
10:          for l ← 1, s0 do
11:            cnt ← tmp % (1 <<  $\Delta$ )
12:            arr[Q * (i-1) + (j-1)][l-1] += cnt
13:            tmp ← tmp / (1 <<  $\Delta$ )
14:   return arr

```

[도23]



[도24]



[도25]

2500

Algorithm 1 COMP($\mathbf{x}_1, \mathbf{x}_2, iter$)

Require: $\mathbf{x}_1(k), \mathbf{x}_2(k) \in [0, 1]$ for every $k \in [0, M - 1]$
Ensure: $\mathbf{y}$ where $\mathbf{y}(k)$ is an approximation of 1 if $\mathbf{x}_1(k) > \mathbf{x}_2(k)$,
0 if $\mathbf{x}_1(k) < \mathbf{x}_2(k)$, and $1/2$ otherwise ($k \in [0, M - 1]$).
0: **procedure** COMP($\mathbf{x}_1, \mathbf{x}_2, iter$)
1: $\mathbf{x} \leftarrow \mathbf{x}_1 - \mathbf{x}_2, i \leftarrow 0$
2: **while** $i < iter$ **do**
3: $\mathbf{x} \leftarrow f(\mathbf{x}), i \leftarrow i + 1$
4: **return** $(\mathbf{x} + 1)/2$

[도26]

2600

Algorithm 7 Computing the mean of a column

0: **procedure** MEAN($\mathbf{x}, \mathbf{v}$)
1: $\text{sum} \leftarrow \text{Enc}(0)$
2: $\mathbf{n} \leftarrow \text{Enc}(0)$
3: **for** $i = 0$ to $N_B - 1$ **do**
4: $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{Mult}(\mathbf{x}_i, \mathbf{v}_i))$
5: $\mathbf{n} \leftarrow \text{Add}(\mathbf{n}, \mathbf{v}_i)$
6: **for** $i = 0$ to $\log_2 M - 1$ **do**
7: $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{Rot}(\text{sum}, 2^i))$
8: $\mathbf{n} \leftarrow \text{Add}(\mathbf{n}, \text{Rot}(\mathbf{n}, 2^i))$
9: $\text{invn} \leftarrow \text{INVNTHSQRT}(\mathbf{n}, \text{Enc}(1^M), 1, 25)$
10: $\text{mean} \leftarrow \text{Mult}(\text{invn}, \text{sum})$
11: **return** mean

[도27]

2700

Algorithm 8 Computing variance of a column

```

0: procedure VAR( $\mathbf{x}, \mathbf{v}$ )
1:  $\text{sum} \leftarrow \text{Enc}(0)$ 
2:  $\mathbf{n} \leftarrow \text{Enc}(0)$ 
2:    $\text{mean} \leftarrow \text{mean}(\mathbf{x}, \mathbf{v})$ 
3: for  $i = 0$  to  $N_B - 1$  do
4:    $\text{tmp} \leftarrow \text{Sub}(\mathbf{x}, \text{mean})$ 
5:    $\text{tmp} \leftarrow \text{Mult}(\text{tmp}, \text{tmp})$ 
6:    $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{Mult}(\text{tmp}, \mathbf{v}))$ 
7:    $\mathbf{n} \leftarrow \text{Add}(\mathbf{n}, \mathbf{v}_i)$ 
8: for  $i = 0$  to  $\log_2 M - 1$  do
9:    $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{Rot}(\text{sum}, 2^i))$ 
10:   $\mathbf{n} \leftarrow \text{Add}(\mathbf{n}, \text{Rot}(\mathbf{n}, 2^i))$ 
11:  $\text{invn} \leftarrow \text{INVNTHSQRT}(\mathbf{n}, \text{Enc}(1^M), 1, 25)$ 
12:  $\text{var} \leftarrow \text{Mult}(\text{invn}, \text{sum})$ 
13: return  $\text{var}$ 

```

[도28]

2800

Algorithm 9 Computing correlation of a column

```

0: procedure CORR( $\mathbf{x}, \mathbf{y}, \mathbf{v}$ )
1:  $\text{sum} \leftarrow \text{Enc}(0)$ 
2:  $\text{mean1} \leftarrow \text{mean}(\mathbf{x}, \mathbf{v}), \text{mean2} \leftarrow \text{mean}(\mathbf{y}, \mathbf{v})$ 
3:  $\text{var1} \leftarrow \text{var}(\mathbf{x}, \mathbf{v}), \text{var2} \leftarrow \text{var}(\mathbf{y}, \mathbf{v})$ 
4:  $\text{invstdev1} \leftarrow \text{INVNTHSQRT}(\text{var1}, \text{Enc}(1), 2, 21)$ 
5:  $\text{invstdev2} \leftarrow \text{INVNTHSQRT}(\text{var2}, \text{Enc}(1), 2, 21)$ 
6: for  $i = 0$  to  $N_B - 1$  do
7:    $\text{tmp1} \leftarrow \text{Sub}(\mathbf{x}, \text{mean1}), \text{tmp2} \leftarrow \text{Sub}(\mathbf{y}, \text{mean2})$ 
8:    $\text{tmp} \leftarrow \text{Mult}(\text{tmp1}, \text{tmp2})$ 
9:    $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{Mult}(\text{tmp}, \mathbf{v}))$ 
10: for  $i = 0$  to  $\log_2 M - 1$  do
11:   $\text{sum} \leftarrow \text{Add}(\text{sum}, \text{Rot}(\text{sum}, 2^i))$ 
12:  $\text{corr} \leftarrow \text{Mult}(\text{sum}, \text{Mult}(\text{invstdev1}, \text{invstdev2}))$ 
13: return  $\text{corr}$ 

```

[도29]

2900
}

Algorithm 10 Computing the maximum in a block

```

0: procedure APPROXMAXSLOT(x)
1: max  $\leftarrow$  x
2: for  $i = \log_2 M - 1$  to 0 do
3:   y  $\leftarrow$  Rot(max,  $2^i$ )
4:   z  $\leftarrow$  COMP(max, y)
5:   max  $\leftarrow$  Mult(max, z)
6:   max  $\leftarrow$  Add(max, Rot(max,  $2^i$ ))
7: return max

```

[도30]

3000
}

Algorithm 11 Computing the maximum in a whole column of multiple blocks

```

0: procedure APPROXMAX(x)
1: max  $\leftarrow$  APPROXMAXSLOT(x0)
2: for  $i = 1$  to  $N_B - 1$  do
3:   y  $\leftarrow$  APPROXMAXSLOT(xi)
4:   z  $\leftarrow$  COMP(max, y)
5:   w  $\leftarrow$  Sub( $1^M$ , z)
6:   max  $\leftarrow$  Mult(max, z), y  $\leftarrow$  Mult(y, w)
7:   max  $\leftarrow$  Add(max, y)
8: return max

```

[도31]

3100
}**Algorithm 12** k-percentile function for ordinal variables

```

0: procedure PERCENTILE( $B, \mathbf{v}^{(f_0)}, S_{f_0}, k$ )
1:  $\mathbf{c}_1 \leftarrow \text{Enc}(\mathbf{0}^M)$ 
2: for  $i \leftarrow 1, S_{f_0}$  do
3:    $\mathbf{c}_{\text{tmp}} \leftarrow \mathbf{b}_{i,0}^{(f_0)}$ 
4:   for  $j \leftarrow 1, N_B - 1$  do
5:      $\text{Add}(\mathbf{c}_{\text{tmp}}, \mathbf{b}_{i,j}^{(f_0)}, \mathbf{c}_{\text{tmp}})$ 
6:   for  $j \leftarrow 0, \log_2(M) - 1$  do
7:      $\mathbf{c}_{\text{tmp}} \leftarrow \text{Add}(\mathbf{c}_{\text{tmp}}, \text{Rot}(\mathbf{c}_{\text{tmp}}, 2^j))$ 
8:    $\vec{m} \leftarrow \mathbf{0}^M$ 
9:   for  $j \leftarrow i - 1, S_{f_0} - 1$  do
10:     $\vec{m}(j) \leftarrow 1$ 
11:    $\mathbf{c}_1 \leftarrow \text{Add}(\mathbf{c}_1, \text{Mult}_{\text{evk}}(\mathbf{c}_{\text{tmp}}, \vec{m}))$ 
12:  $\mathbf{c}_2 \leftarrow \mathbf{v}_0^{(f_0)}$ 
13: for  $i \leftarrow 1, N_B - 1$  do
14:    $\text{Add}(\mathbf{c}_2, \mathbf{v}_i^{(f_0)}, \mathbf{c}_2)$ 
15: for  $i \leftarrow 0, \log_2(M) - 1$  do
16:    $\mathbf{c}_2 \leftarrow \text{Add}(\mathbf{c}_2, \text{Rot}(\mathbf{c}_2, 2^j))$ 
17:  $\mathbf{c}_2 \leftarrow \text{INVNTHSQRT}(\mathbf{c}_2, \text{Enc}(\mathbf{1}^M), 1, 25)$ 
18:  $\mathbf{c}_1 \leftarrow \text{Mult}(\mathbf{c}_1, \mathbf{c}_2)$ 
19:  $\mathbf{c}_1 \leftarrow \text{APPROXSIGN}(\text{Sub}(\mathbf{c}_1, \frac{k}{100}))$ 
    ▷ operate  $f(x) = -0.5(x - 0.5)^2 + 1.125$ 
20:  $\mathbf{c}_1 \leftarrow \text{Square}_{\text{evk}}(\text{Sub}(\mathbf{c}_1, 0.5))$ 
21:  $\mathbf{c}_1 \leftarrow \text{Add}(\text{Mult}(\mathbf{c}_1, -0.5), 1.125)$ 
    ▷ find  $l$  s.t.  $l$ -th to  $S_{f_0}$ -th slots have 1 as value
22: for  $j \leftarrow 0, \log_2(M) - 1$  do
23:    $\mathbf{c}_1 \leftarrow \text{Add}(\mathbf{c}_1, \text{Rot}(\mathbf{c}_1, 2^j))$ 
24:  $\mathbf{c}_1 \leftarrow \text{Negate}(\mathbf{c}_1)$ 
25:  $\mathbf{c}_1 \leftarrow \text{Add}(\mathbf{c}_1, S_{f_0} + 1)$ 
26: return  $\mathbf{c}_1$ 

```

[도32]

