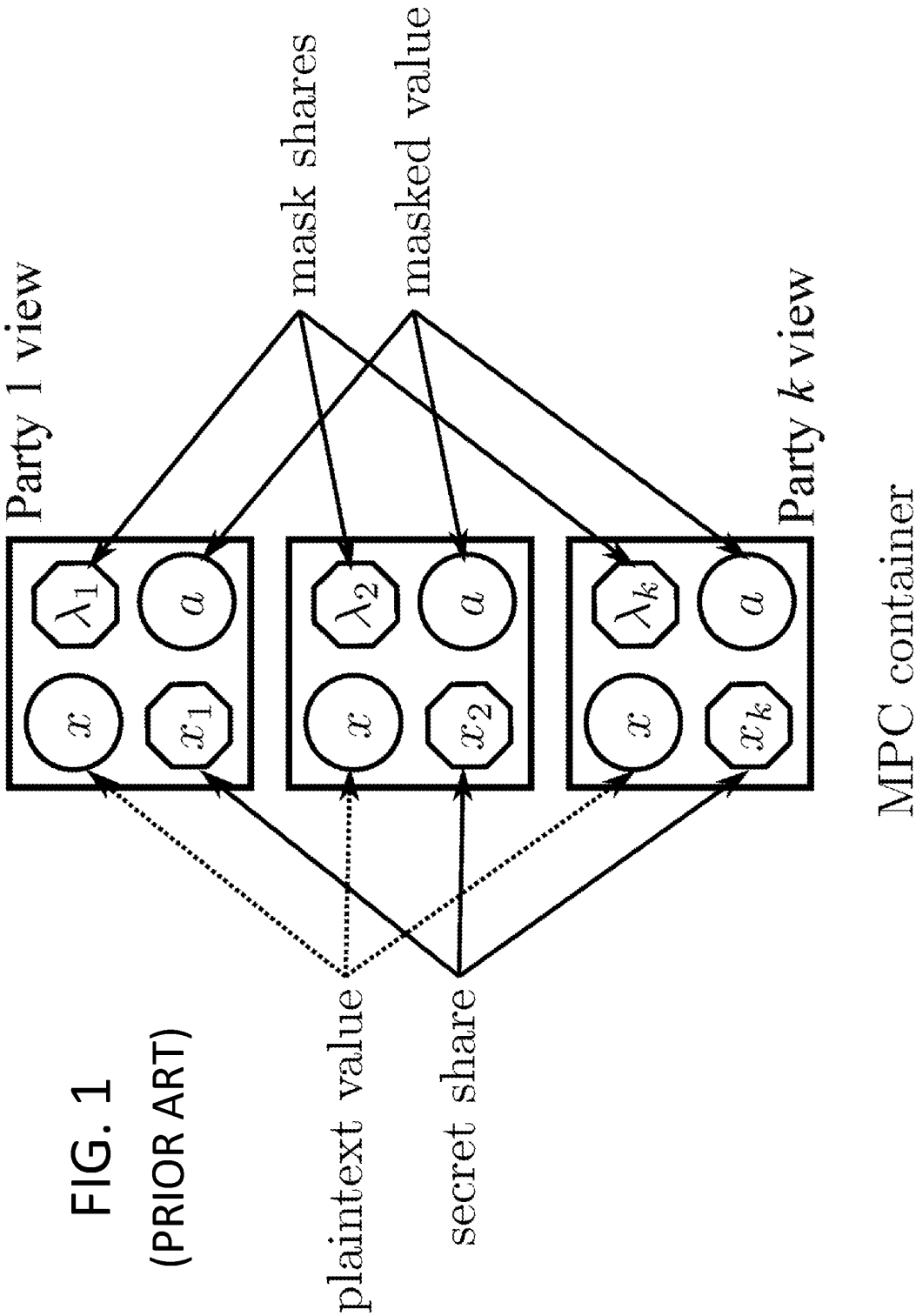




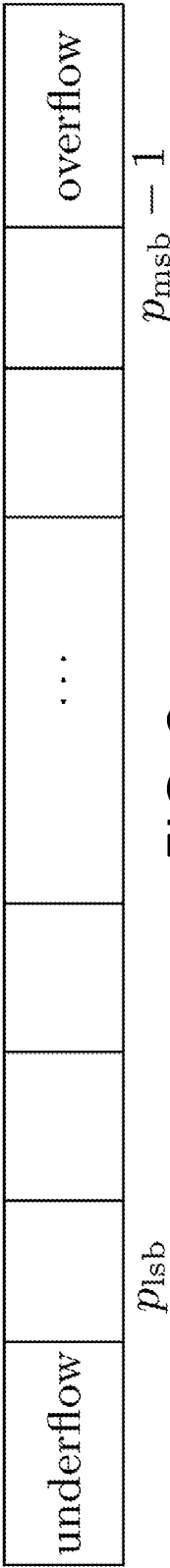


FIG. 2

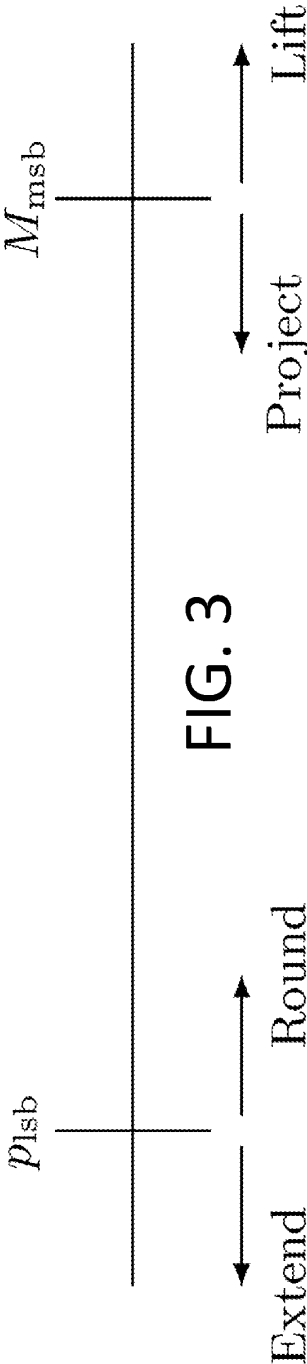


FIG. 3

FIG. 4

$\mathcal{M}_{M_{\text{msb}}, p_{\text{lsb}}}$	Operation	Value
$\mathcal{M}_{6,-2}$	Input	001011.10
$\mathcal{M}_{6,-4}$	Extend	001011.1000
$\mathcal{M}_{6,0}$	Round	001100
$\mathcal{M}_{4,-2}$	Project	1011.10
$\mathcal{M}_{8,-2}$	Lift	??001011.10

FIG. 5

Lift Operation: Offline Phase

Input: The mask $\lambda \in 2^{p_{\text{lsb}}} \mathbb{Z} / 2^{M_{\text{msb}}} \mathbb{Z}$ of a modular real number

Output: Precomputed shares $\llbracket b \rrbracket_{M'_{\text{msb}} - M_{\text{msb}}, 0}$ and $\llbracket \nu \rrbracket_{M'_{\text{msb}}, p'_{\text{lsb}}}$.

- 1: Set $b = 0$ if $\lambda \in \left[-\frac{2^{M_{\text{msb}}}}{4}, \frac{2^{M_{\text{msb}}}}{4} \right)$, $b = 1$ if $\lambda \in \left[\frac{2^{M_{\text{msb}}}}{4}, \frac{3 \cdot 2^{M_{\text{msb}}}}{4} \right)$.
 - 2: $\llbracket b \rrbracket_{M'_{\text{msb}} - M_{\text{msb}}, 0} = \text{secretShares}(b, \mathcal{M}_{M'_{\text{msb}} - M_{\text{msb}}, 0})$
 - 3: Compute $\nu = \text{roundTo}(\text{quartermod}_{2^{M_{\text{msb}}}}(\lambda), 2^{p'_{\text{lsb}}}) \bmod 2^{M'_{\text{msb}}}$.
 - 4: $\llbracket \nu \rrbracket_{M'_{\text{msb}}, p'_{\text{lsb}}} = \text{secretShares}(\nu, \mathcal{M}_{M'_{\text{msb}}, p'_{\text{lsb}}})$
 - 5: **return** $\llbracket b \rrbracket_{M'_{\text{msb}} - M_{\text{msb}}, 0}$ and $\llbracket \nu \rrbracket_{M'_{\text{msb}}, p'_{\text{lsb}}}$.
-

FIG. 6

Lift Operation: Online Phase

Input: For given parameters $p_{\text{lsb}}, M_{\text{msb}}, p'_{\text{lsb}}, M'_{\text{msb}}$:

- A pair $(\llbracket x \rrbracket_{M_{\text{msb}}, p_{\text{lsb}}}, \llbracket \lambda \rrbracket_{M_{\text{msb}}, p_{\text{lsb}}}, a = x + \lambda)$ of a secret shared modular real number and a mask for a plaintext $x \in \mathcal{P}_{m, p_{\text{lsb}}}$ and some $m \leq M_{\text{msb}} - 2$,
- Triplet shares $\llbracket b \rrbracket_{M'_{\text{msb}} - M_{\text{msb}}, 0}, \llbracket \nu \rrbracket_{M'_{\text{msb}}, p'_{\text{lsb}}}$.

Output: Output secret shares $\llbracket x \rrbracket_{\text{mod}, M'_{\text{msb}}, p'_{\text{lsb}}}$ of the same plaintext x .

- 1: (if it is not already done) Mask and reveal $a = (x + \lambda) \bmod 2^{M_{\text{msb}}}$.
- 2: Player 1 computes $a_0 = \text{roundTo}(\text{centermod}_{2^{M_{\text{msb}}}}(a), 2^{p'_{\text{lsb}}}) \bmod 2^{M'_{\text{msb}}}$
- 3: All compute $\llbracket x \rrbracket_{M'_{\text{msb}}, p'_{\text{lsb}}} = a_0 + (a_1 - a_0) \boxtimes \llbracket b \rrbracket_{M'_{\text{msb}} - M_{\text{msb}}, 0} - \llbracket \nu \rrbracket_{M'_{\text{msb}}, p'_{\text{lsb}}}$, where

$$a_1 = \text{roundTo}(\text{posmod}_{2^{M_{\text{msb}}}}(a), 2^{p'_{\text{lsb}}}) \bmod 2^{M'_{\text{msb}}}$$

4: **return** $\llbracket x \rrbracket_{M'_{\text{msb}}, p'_{\text{lsb}}}$

FIG. 7 Share Refresh Operation ModReal to ModReal: Offline Phase

Input: Parameters params and params' , and a mask $\lambda \in \mathcal{M}_{\text{params}}$.

Output: Precomputed shares $\llbracket \nu' \rrbracket \in \mathcal{M}_{\text{params}'}^k$ of $\nu' := \text{projectAndRound}_{\text{params}'}(\lambda)$.

- 1: Compute $\nu' = \text{projectAndRound}_{\text{params}'}(\lambda)$.
- 2: $\llbracket \nu' \rrbracket = \text{secretShares}(\nu', \mathcal{M}_{\text{params}'})$
- 3: **return** $\llbracket \nu' \rrbracket$.

FIG. 8 Share Refresh Operation ModReal to ModReal: Online Phase

Input: For given parameters $\text{params} = (M_{\text{msb}}, p_{\text{lsb}})$ and $\text{params}' = (M'_{\text{msb}}, p'_{\text{lsb}})$:

- A secret-shared modReal container $(\llbracket x \rrbracket, \llbracket \lambda \rrbracket, a = x + \lambda)$ for $x \in \mathcal{P}_{m, p_{\text{lsb}}}$ for some $m \leq M'_{\text{msb}}$,
- Triplet shares $\llbracket \nu' \rrbracket \in \mathcal{M}_{\text{params}'}^k$.

Output: Output secret shares $\llbracket z \rrbracket \in \mathcal{M}_{\text{params}'}^k$ of the same plaintext x .

- 1: (if it is not already done) Mask and reveal $a = (x + \lambda) \bmod 2^{M_{\text{msb}}}$.
- 2: Player 1 computes $a_{\text{rounded}} = \text{projectAndRound}_{\text{params}'}(a)$
- 3: All Compute $\llbracket z \rrbracket = a_{\text{rounded}} - \llbracket \nu' \rrbracket$
- 4: **return** $\llbracket z \rrbracket$

FIG. 9

Beaver multiplication on modular backend (Secret-Secret)

Input: Given secret shares $\llbracket x \rrbracket_{\text{mod}, M_{\text{msb}}^A, p_{\text{lsb}}^A}$ and $\llbracket y \rrbracket_{\star, M_{\text{msb}}^B, p_{\text{lsb}}^B}$, working parameters $(M_{\text{msb}}^{AW}, p_{\text{lsb}}^{AW}), (M_{\text{msb}}^{BW}, p_{\text{lsb}}^{BW}), (M_{\text{msb}}^W, p_{\text{lsb}}^W)$ of same width $\eta := M_{\text{msb}} - p_{\text{lsb}}$, and output parameters $(M_{\text{msb}}^z, p_{\text{lsb}}^z)$

Output: Compute secret share $\llbracket z \rrbracket_{\text{mod}, M_{\text{msb}}^z, p_{\text{lsb}}^z}$

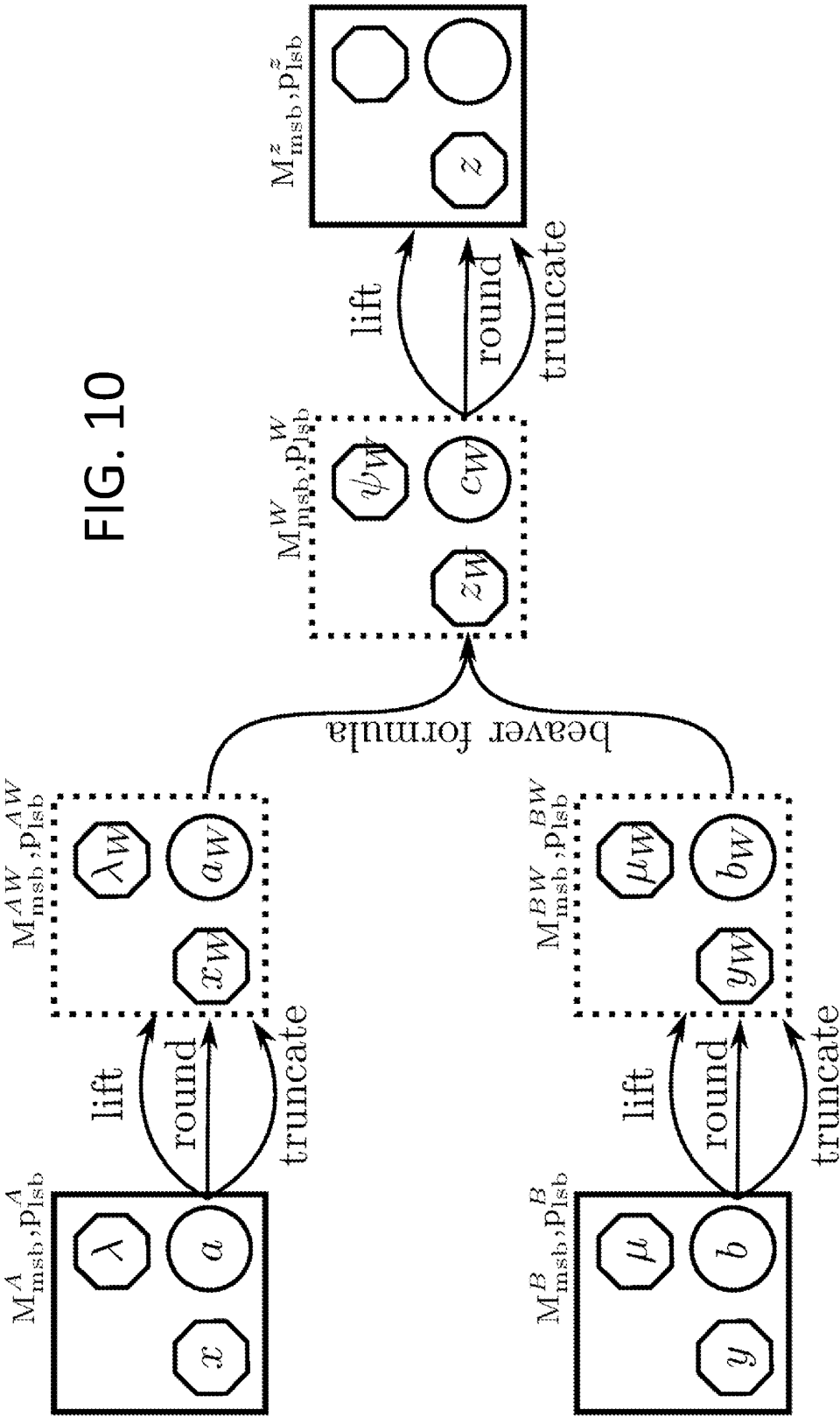
```

1: if  $M_{\text{msb}}^{AW} > M_{\text{msb}}^A$  then
2:   mask-and reveal  $a = x + \lambda$ 
3:    $\llbracket a_w \rrbracket = \text{centermod}_{2^{M_{\text{msb}}^A}}(a) + \text{diffmod}_{2^{M_{\text{msb}}^A}}(a) \cdot \llbracket b_\lambda \rrbracket - \llbracket \nu_\lambda \rrbracket + \llbracket \lambda_w \rrbracket$ 
4:   reveal  $a_w$ 
5: else if  $M_{\text{msb}}^{AW} \leq M_{\text{msb}}^A$  and  $p_{\text{lsb}}^{AW} > p_{\text{lsb}}^A$  then
6:   mask-and reveal  $a = x + \lambda$ 
7:    $\llbracket a_w \rrbracket = a \bmod 2^{M_{\text{msb}}^{AW}} - \llbracket \nu_\lambda \rrbracket + \llbracket \lambda_w \rrbracket$ 
8:   reveal  $a_w$ 
9: else
10:  mask-and reveal  $a = x + \lambda$ 
11:  Set  $a_w = a \bmod 2^{M_{\text{msb}}^{AW}}$ 
12:  Set each  $\lambda_{w,j} = \lambda_j \bmod 2^{M_{\text{msb}}^{AW}}$ 
13: end if
14: Do the same to compute  $b_w$  and  $\mu_w$ 
15: compute  $\llbracket z_w \rrbracket = a_w \times b_w - a_w \times \llbracket \mu_w \rrbracket - \llbracket \lambda_w \rrbracket \times b_w + \llbracket \lambda \mu_w \rrbracket$ 
16: if  $M_{\text{msb}}^z > M_{\text{msb}}^W$  then
17:  mask-and reveal  $c_w = z_w + \psi_w$ 
18:  Compute  $\llbracket z \rrbracket = \text{centermod}_{2^{M_{\text{msb}}^W}}(c_w) + \text{diffmod}_{2^{M_{\text{msb}}^W}}(c_w) \cdot \llbracket b_\psi \rrbracket - \llbracket \nu_\psi \rrbracket$ 
19: else
20:  Compute  $\llbracket z \rrbracket = \llbracket z_w \rrbracket \bmod 2^{M_{\text{msb}}^z}$ 
21: end if

```

$\triangleright$ lift A
 $\triangleright$ round A
 $\triangleright$ truncate A
 $\triangleright$ lift output
 $\triangleright$ trunc/round output

Beaver multiplication on modular backend (Secret-Secret)

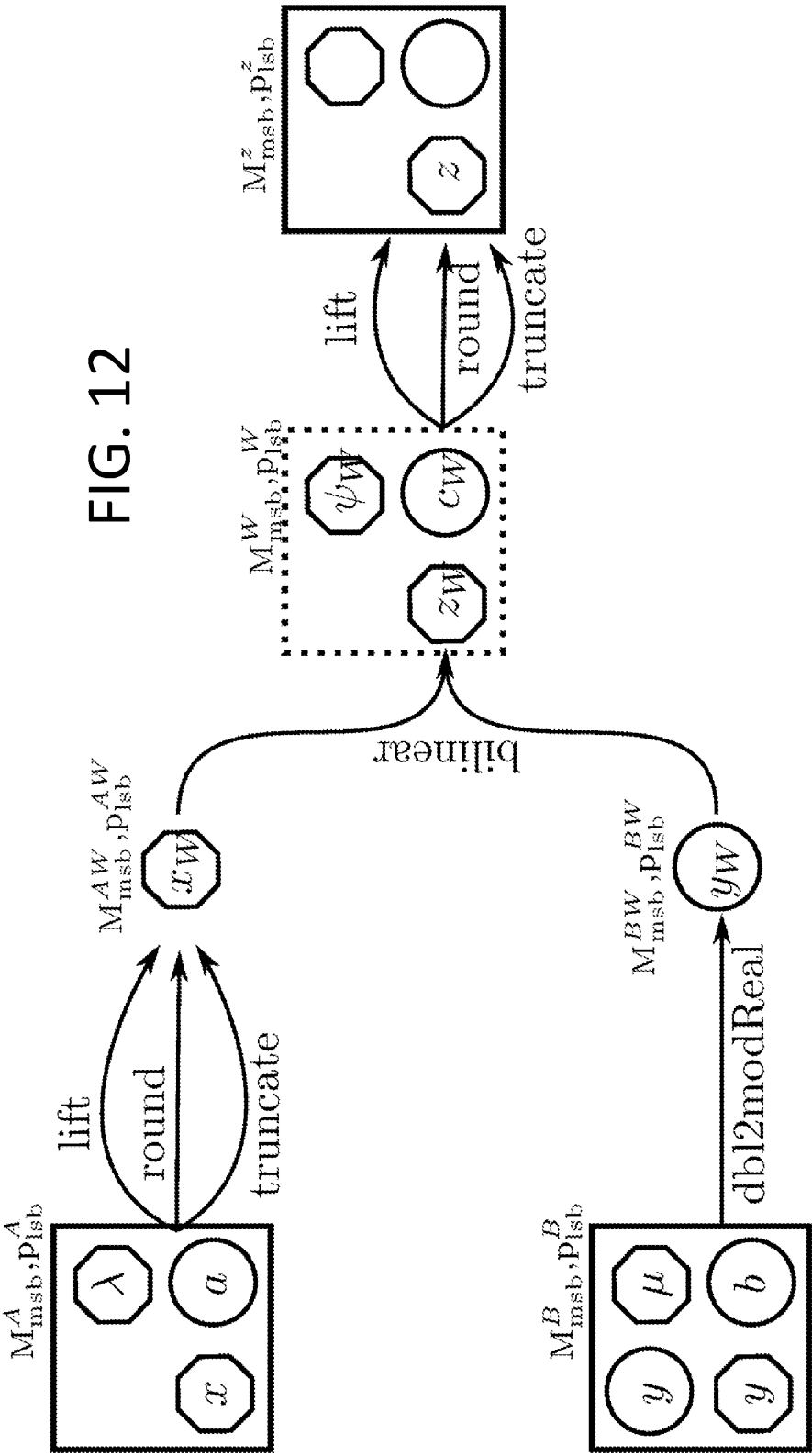


Beaver multiplication on modular backend (Secret-Public)

FIG. 11

Input: Given secret shares $\llbracket x \rrbracket_{\text{mod}, M_{\text{msb}}^A, p_{\text{lsb}}^A}$ and a public value y , working parameters $(M_{\text{msb}}^{AW}, p_{\text{lsb}}^{AW}), (M_{\text{msb}}^{BW}, p_{\text{lsb}}^{BW}), (M_{\text{msb}}^{W}, p_{\text{lsb}}^W)$ of same width, and output parameters $(M_{\text{msb}}^z, p_{\text{lsb}}^z)$	
Output: Compute secret share $\llbracket z \rrbracket_{\text{mod}, M_{\text{msb}}^z, p_{\text{lsb}}^z}$	
1: if $M_{\text{msb}}^{AW} > M_{\text{msb}}^A$ then	$\triangleright$ lift A
2: mask-and reveal $a = x + \lambda$	
3: $\llbracket x_w \rrbracket = \text{centermod}_{2^{M_{\text{msb}}^A}}(a) + \text{diffmod}_{2^{M_{\text{msb}}^A}}(a) \cdot \llbracket b_\lambda \rrbracket - \llbracket \nu_\lambda \rrbracket$	
4: reveal a_w	
5: else if $M_{\text{msb}}^{AW} \leq M_{\text{msb}}^A$ and $p_{\text{lsb}}^{AW} > p_{\text{lsb}}^A$ then	$\triangleright$ round A
6: mask-and reveal $a = x + \lambda$	
7: $\llbracket x_w \rrbracket = a \bmod 2^{M_{\text{msb}}^{AW}} - \llbracket \nu_\lambda \rrbracket$	
8: reveal a_w	
9: else	$\triangleright$ truncate A
10: Set each $x_{w,j} = x_j \bmod 2^{M_{\text{msb}}^{AW}}$	
11: end if	
12: Convert the public value y to a modreal value y_w of params $(M_{\text{msb}}^{AW}, p_{\text{lsb}}^{AW})$	
13: compute $\llbracket z_w \rrbracket = \llbracket x_w \rrbracket \times y_w$	
14: if $M_{\text{msb}}^z > M_{\text{msb}}^W$ then	$\triangleright$ lift output
15: mask-and reveal $c_w = z_w + \psi_w$	
16: Compute $\llbracket z \rrbracket = \text{centermod}_{2^{M_{\text{msb}}^W}}(c_w) + \text{diffmod}_{2^{M_{\text{msb}}^W}}(c_w) \cdot \llbracket b_\psi \rrbracket - \llbracket \nu_\psi \rrbracket$	$\triangleright$ trunc/round output
17: else	
18: Compute $\llbracket z \rrbracket = \llbracket z_w \rrbracket \bmod 2^{M_{\text{msb}}^z}$	
19: end if	

Beaver multiplication on modular backend (Secret-Public)



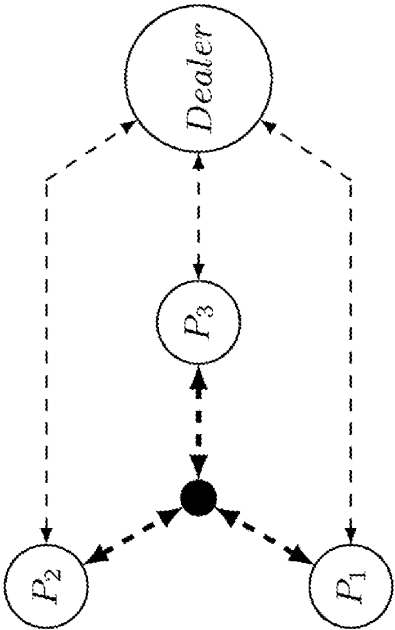
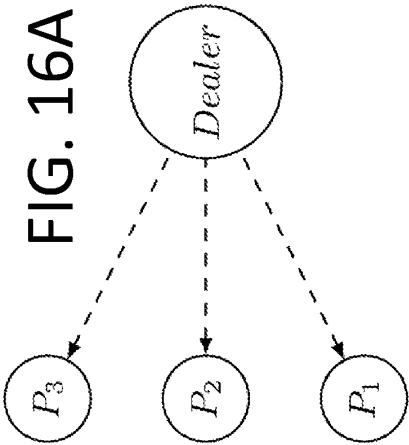
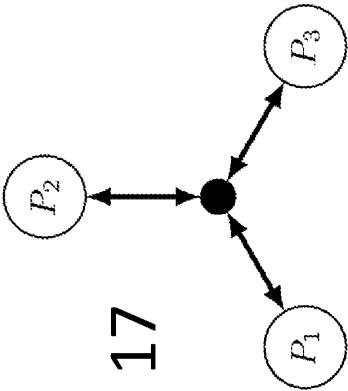
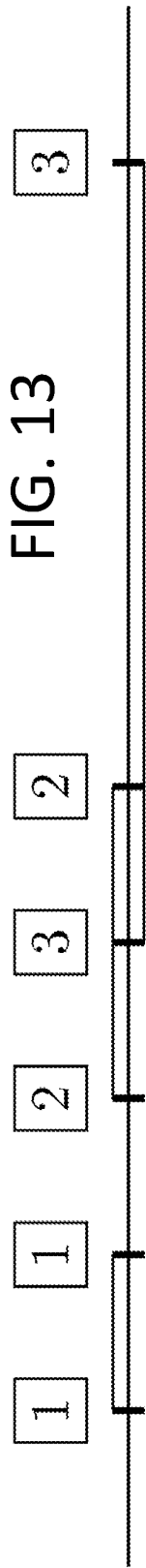


FIG. 16B

FIG. 14

Linear combination operation with ModReal representation

Input: Input shares $\llbracket x^{(i)} \rrbracket$ with associated masking data $\llbracket \lambda^{(i)} \rrbracket$ and masked value $a^{(i)}$; coefficients $(\alpha^{(i)}, k^{(i)})$, working parameters $M_{\text{msb}}^w, p_{\text{lsb}}^w$, output parameters $M_{\text{msb}}^z, p_{\text{lsb}}^z$ lift triplets

Output: output shares $\llbracket z \rrbracket$ where $z = \sum_{i=1}^n \frac{a^{(i)}}{2^{k^{(i)}}} x^{(i)}$

- 1: **for** $i = 1$ to n **do**
 - 2: **if** $M_{\text{msb}}^w > M_{\text{msb}}^{(i)} - k^{(i)}$ **then** $\triangleright$ lift case
 - 3: players **jointly** mask and reveal $a^{(i)} = x^{(i)} + \lambda^{(i)}$
 - 4: compute $\llbracket y^{(i)} \rrbracket_{\text{mod}} = \text{centermod}_{2^{M_{\text{msb}}^w}}(a^{(i)}) + \text{diffmod}_{2^{M_{\text{msb}}^w}}(a^{(i)}) \odot \llbracket b^{(i)} \rrbracket$
 in $\mathcal{M}_{M_{\text{msb}}^w + k^{(i)}, p_{\text{lsb}}^w + k^{(i)}}$
 - 5: This reinterprets into $\llbracket c^{(i)} \rrbracket_{\text{mod}} = \llbracket \frac{1}{2^{k^{(i)}}} y^{(i)} \rrbracket_{\text{mod}}$ in $\mathcal{M}_{M_{\text{msb}}^w, p_{\text{lsb}}^w}$.
 - 6: compute $\llbracket u^{(i)} \rrbracket_{\text{mod}} = a^{(i)} \cdot \llbracket c^{(i)} \rrbracket - \llbracket \nu^{(i)} \rrbracket$ in $\mathcal{M}_{M_{\text{msb}}^w, p_{\text{lsb}}^w}$.
 - 7: **else**
 - 8: players locally compute $\llbracket u^{(i)} \rrbracket_{\text{mod}} = a^{(i)} \cdot \llbracket \frac{1}{2^{k^{(i)}}} x^{(i)} \rrbracket_{\text{mod}}$ projected and rounded to $\mathcal{M}_{M_{\text{msb}}^w, p_{\text{lsb}}^w}$ (no triplet is used).
 - 9: **end if**
 - 10: **end for**
 - 11: Players add the shares $\llbracket u^{(i)} \rrbracket_{\text{mod}}$ locally for $i \in [1, n]$ to obtain the shares of the result $\llbracket w \rrbracket_{\text{mod}}$
 - 12: cast $\llbracket w \rrbracket_{\text{mod}}$ to the forced output $\llbracket z \rrbracket_{\star, M_{\text{msb}}^z, p_{\text{lsb}}^z}$ (using the general cast algorithm)
-

FIG. 15

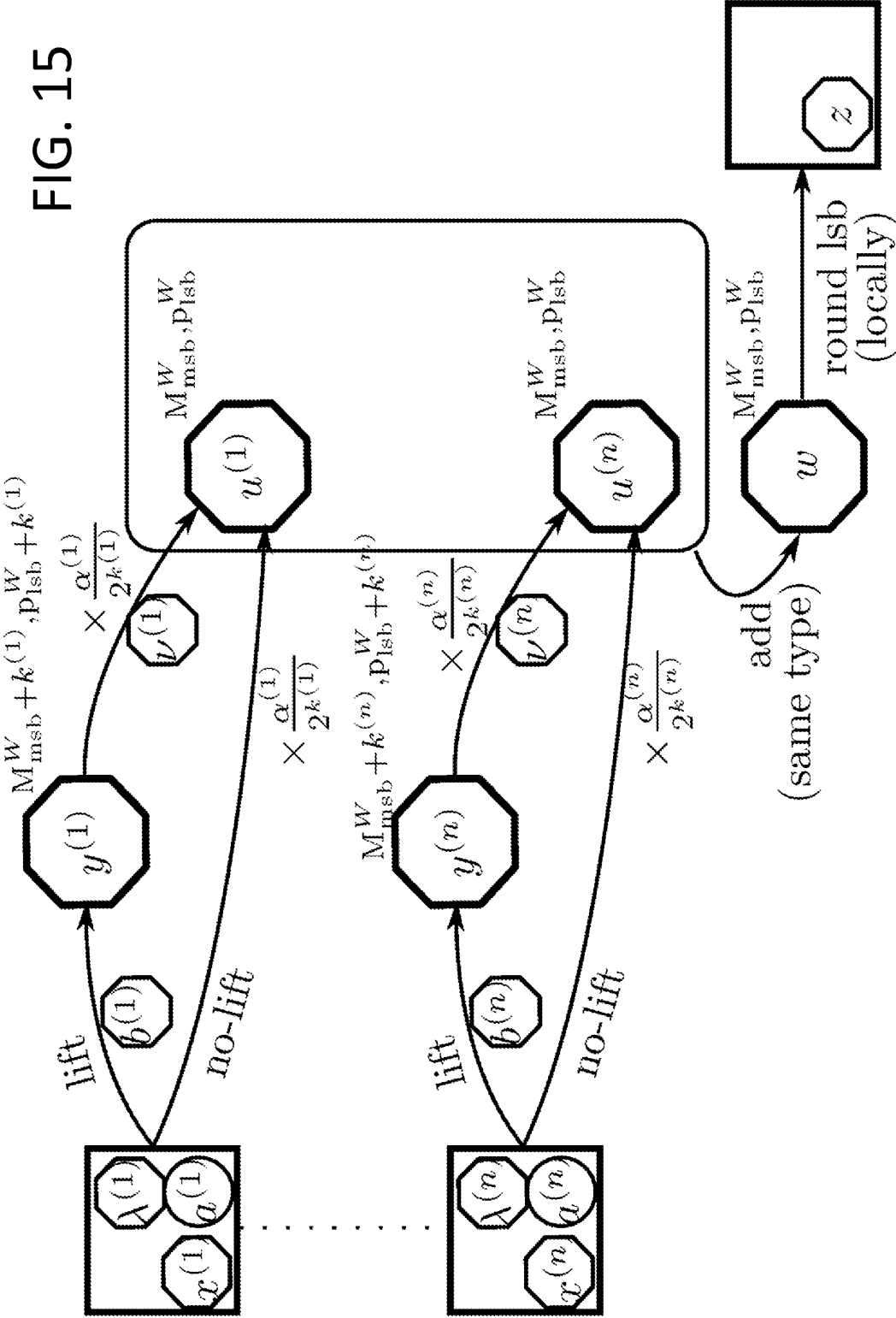


FIG. 18

