4.4) Share Convert (for conversion of shares over $\mathbb{Z}_L$ to $\mathbb{Z}_{L-1}$ where $L = 2^{64}$, Algorithm 4)

* $\Pi_{SC}(\{P_0, P_1\}, P_2)$:

⊙ Input: P_0, P_1 hold $\langle a \rangle_0^L$ and $\langle a \rangle_1^L$, respectively such that $Reconst^L(\langle a \rangle_0^L, \langle a \rangle_1^L) \neq L-1$

⊙ Output: P_0, P_1 get $\langle a \rangle_0^{L-1}$ and $\langle a \rangle_1^{L-1}$

⊙ Common Randomness: P_0, P_1 hold a random bit n'', a random $r \in \mathbb{Z}_L$, shares $\langle r \rangle_0^L, \langle r \rangle_1^L$, $\alpha = wrap(\langle r \rangle_0^L, \langle r \rangle_1^L, L)$ and shares of 0 over $\mathbb{Z}_{L-1}$ denoted by u_0 and u_1.

⊙ Steps:

1) For each $j \in \{0,1\}$, P_j executes step 2-3
2) $\langle \tilde{a} \rangle_j^L = \langle a \rangle_j^L + \langle r \rangle_j^L$ and $B_j = wrap(\langle a \rangle_j^L, \langle r \rangle_j^L, L)$
3) Send $\langle \tilde{a} \rangle_j^L$ to P_2.
4) P_2 computes $n = Reconst^L(\langle \tilde{a} \rangle_0^L, \langle \tilde{a} \rangle_1^L)$ and $\delta = wrap(\langle \tilde{a} \rangle_0^L, \langle \tilde{a} \rangle_1^L, L)$
5) P_2 generates shares $\{\langle n[i] \rangle_j^P\}_{i \in [L]}$ and $\langle \delta \rangle_j^{L-1}$ for $j \in \{0,1\}$ and sends to P_j
6) P_0, P_1, P_2 invoke $\Pi_{PC}\{\{P_0, P_1\}, P_2\}$ with $P_j, j \in \{0,1\}$ having input $(\{\langle n[i] \rangle_j^P\}_{i \in [L]}, r-1, n'')$ and P_2 learns n'.

7) For $j \in \{0,1\}$, P_2 generates $\langle n' \rangle_j^{L-1}$ and sends to P_j.

8) For each $j \in \{0,1\}$, P_j executes steps 9-11

9) $\langle n \rangle_j^{L-1} = \langle n' \rangle_j^{L-1} + (1-j)\, n'' - 2n'' \langle n' \rangle_j^{L-1}$

10) $\langle \theta \rangle_j^{L-1} = B_j + (1-j) \cdot (-\alpha - 1) + \langle \delta \rangle_j^{L-1} + \langle n \rangle_j^{L-1}$

11) Output $\langle y \rangle_j^{L-1} = \langle a \rangle_j^{L} - \langle \theta \rangle_j^{L-1} + u_j$ (over $L-1$)

⑥ Explanation:

The goal of the protocol is to change shares of a over Z_L to Z_{L-1}. The thing to notice here is that if the original shares wraps around L, then we need to subtract 1 from it to convert it to shares over Z_{L-1}, else original shares are also valid shares of same value over $L-1$.

To calculate whether the original shares of a wrap around L or not, we define $\theta = \text{wrap}(\langle a \rangle_0^L, \langle a \rangle_1^L, L)$. Then we subtract shares of θ, $(\langle \theta \rangle_0^{L-1}, \langle \theta \rangle_1^{L-1})$

from shares of a to find its shares over Z_{L-1}. This is exactly what is done in step 11.

But now to securely compute shares of θ, the protocol uses these relations.

i) $r = \langle r \rangle_0^L + \langle r \rangle_1^L - \alpha L$

ii) $\langle \tilde{a} \rangle_j^L = \langle a \rangle_j^L + \langle r \rangle_j^L - \beta_j L$

iii) $\tilde{a} = \langle \tilde{a} \rangle_0^L + \langle \tilde{a} \rangle_1^L - \delta L$

iv) $\eta = a + r - (1-\eta) L$

v) $a = \langle a \rangle_0^L + \langle a \rangle_1^L - \theta L$

It picks a random number r and then computes $\tilde{a} \equiv a + r \bmod L$. It then uses Π_{PL} to see if $\tilde{a} > r-1$ and gets the output $\eta' = \eta'' \oplus (\tilde{a} > r-1)$ which is the masked result. To get the real result of comparison, it computes $\eta = \eta' \oplus \eta''$. Then, $\eta = wrap(a, r, L) = 0$ iff $\tilde{a} > r-1$.

Now, the paper claims that computing (i) − (ii) − (iii) + (iv) + (v) gives

$$\theta = \beta_0 + \beta_1 - \alpha + \delta + \eta - 1 \quad \text{------(1)}$$

Lets check it

$$r - \langle \tilde{a} \rangle_0^L - \langle \tilde{a} \rangle_1^L - \cancel{\kappa} + \cancel{\kappa} + a$$
$$= \cancel{\langle r \rangle_0^L} + \cancel{\langle r \rangle_1^L} - \alpha L - \cancel{\langle a \rangle_0^L} - \cancel{\langle r \rangle_0^L} + \beta_0 L$$
$$- \cancel{\langle a \rangle_1^L} - \cancel{\langle r \rangle_1^L} + \beta_1 L - \langle \tilde{a} \rangle_0^L - \langle \tilde{a} \rangle_0^L$$
$$+ \delta L + a + r - (1-\eta) L + \cancel{\langle a \rangle_0^L} + \cancel{\langle a \rangle_1^L}$$
$$- \theta L$$

$$r - \langle \tilde{a} \rangle_0^L - \langle \tilde{a} \rangle_1^L + a + \theta L$$
$$= -\alpha L + \beta_0 L + \beta_1 L - \langle \tilde{a} \rangle_0^L - \langle \tilde{a} \rangle_0^L + \delta L$$
$$+ a + r - (1-\eta) L - \theta L$$
$$= r - \langle \tilde{a} \rangle_0^L - \langle \tilde{a} \rangle_1^L + a + (\beta_0 + \beta_1 - \alpha + \delta - 1 + \eta - \theta) L$$

Then, comparing the corresponding terms, we get,

$$\beta_0 + \beta_1 - \alpha + \delta - 1 + \eta - \theta = 0$$

$$\theta = \beta_0 + \beta_1 - \alpha + \delta + \eta - 1 \quad \text{// True.}$$

⑥ Protocol:

+ Step 2 computes shares of $\tilde{a} = a + r$
$\langle \tilde{a} \rangle_j^L = \langle a \rangle_j^L + \langle r \rangle_j^L$ with β_j the wrap-around bit.

+ Step 4 reconstructs $x = \text{Reconst}^L(\langle \tilde{a} \rangle_0^L, \langle \tilde{a} \rangle_1^L)$ and $\delta = \text{wrap}(\langle \tilde{a} \rangle_0^L, \langle \tilde{a} \rangle_1^L, L)$

+ Step 6 computes $x > r-1$. But why $(r-1)$ and not just r? x is the sum of a and r, so if $x > r-1$, then $\eta = 1$ and the sum doesn't wrap around L and

$$x = a + r - (1-\eta)L$$
$$= a + r - (1-1)L$$
$$= a + r$$

But if $x \ngtr r-1$, then $\eta = 0$ and the sum does wrap around L and

$$x = a + r - L \quad (-L \text{ acts like mod } L)$$

+ Step 9 computes $\eta = \eta' \oplus \eta''$

+ Step 10 computes θ using the relation (1)

⊙ Question

- Why comparison with $(r-1)$ and not r
- Why use u_j in the step 11.