4.3) Private Compare (for comparison, Algorithm 3)

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

- **Input**: P_0, P_1 hold $\{\langle r[i]\rangle_0^P\}_{i\in[l]}$ and $\{\langle r[i]\rangle_1^P\}_{i\in[l]}$, respectively, a common input r (an l-bit integer) and a common random bit β.
- **Output**: P_2 gets a bit $\beta \oplus (x > r)$
- **Common Randomness**: P_0, P_1 hold l common random values $s_i \in \mathbb{Z}_p^*$ for all $i \in [l]$ and a random permutation π for l elements. P_0 and P_1 additionally hold l common random values $u_i \in \mathbb{Z}_p^*$

- **Steps**:

1) Let $t = r + 1 \bmod 2^l$
2) For each $j \in \{0,1\}$, P_j executes steps 3-14
3) for $i = \{l, l-1, \cdots, 1\}$ do
4) if $\beta = 0$ then

Part 1:

5) $\langle w_i\rangle_j^P = \langle x[i]\rangle_j^P + j\,r[i] - 2r[i]\langle x[i]\rangle_j^P$
6) $\langle c_i\rangle_j^P = j\,r[i] - \langle x[i]\rangle_j^P + j + \sum_{k=i+1}^{l} \langle w_k\rangle_j^P$

Part 2

7) else if $\beta = 1$ and $r \neq 2^\ell - 1$ then

8) $\langle w_i \rangle_j^P = \langle x[i] \rangle_j^P + j + [i] - 2 + [i] \langle x[i] \rangle_j^P$

9) $\langle c_i \rangle_j^P = -j + [i] + \langle x[i] \rangle_j^P + j + \sum_{k=i+1}^{\ell} \langle w_k \rangle_j^P$

Part 3

10) else

11) if $i \neq 1$,

$$\langle c_i \rangle_j^P = (1-j)(u_i + 1) - j u_i$$

else

$$\langle c_i \rangle_j^P = (-1)^j \cdot u_i$$

12) end if

13 end for

14) Send $\{\langle d_i \rangle_j^P\}_i = \pi\left(\{s_i \langle c_i \rangle_j^P\}_i\right)$ to P_2

15) For all $i \in [\ell]$, P_2 computes

$d_i = \text{Reconst}^P(\langle d_i \rangle_0^P, \langle d_i \rangle_0^P)$ and set

$\beta' = 1$ iff $\exists\, i \in [\ell]$ such that $d_i = 0$.

⊙ We want to compute $\beta' = \beta \oplus (x > r)$ where $(x > r)$ denotes the bit which is 1 when $x > r$ over the integers and 0 otherwise.

β \ $(x>R)$	$0 \Rightarrow x \leq R$	$1 \Rightarrow x > R$
$\beta = 0$	$\beta' = 0$	$\beta' = 1$
$\beta = 1$	$\beta' = 1$	$\beta' = 0$

⊙ Note:

i) Use of XOR is the mask the output with β and give β' to P_2

ii) Parties initially have shares of bits of x over Z_p ($p > 67$) which is provided to P_0, P_1 by P_2.

* Part 1 : Case $\beta = 0$

In this case, $\beta' = 1$ iff $(x > r)$ or at the leftmost bit where $x[i] \neq r[i]$, $x[i] = 1$

$x = 100111$
$r = 000101$

comparison of leftmost bits of x and r,
$x[i] = 1 \Rightarrow x > r$

⊙ we compute

Step 4) $w_i = x[i] \oplus r[i] = x[i] + r[i] - 2x[i]r[i]$

here $a \oplus b = a + b - 2ab = (a+b)^2 \pmod 2$

$(a+b)^2 \bmod 2$	$a=0$	$a=1$
$b=0$	0 / 0	1 / 1
$b=1$	1 / 1	0 / 0

⊙ 2-party setting

$$\langle w_i \rangle_j^P = \langle x[i] \rangle_j^P + j\, r[i] - 2r[i] \langle x[i] \rangle_j^P$$

here,

P_0: $\langle w_i \rangle_0^P = \langle x[i] \rangle_0^P - 2r[i] \langle x[i] \rangle_0^P$

P_1: $\langle w_i \rangle_0^P = \langle x[i] \rangle_1^P + r[i] - 2r[i] \langle x[i] \rangle_1^P$

then,

$$\langle w_i \rangle = \langle x[i] \rangle_0^P + \langle x[i] \rangle_1^P + r[i] - 2r[i] \{ \langle x[i] \rangle_0^P + \langle x[i] \rangle_1^P \}$$

$$= x[i] + r[i] - 2r[i]x[i]$$

⊙ step 5) then we compute,

$$c[i] = r[i] - n[i] + 1 + \sum_{k=i+1}^{L} w_k$$

and if $\exists\, i$ such that $c_i = 0$ iff $n > r$.

example: $n > r$

$n = 10001101$

$r = 10000110$

$w = n \oplus r = 00001011$

Hamming distance $H = \text{sum}(w[i]) = 3$

if $n > r$, $r[i] - n[i]$ is negative at the i-th place where n and r first differ

Note: Indexing starts from $i = 1$ according to step 3 of the protocol.

$c[8] = (1-1) + 1 + 0 = 1$

$c[7] = (0-0) + 1 + 0 = 1$

$c[6] = (0-0) + 1 + 0 = 1$

$c[5] = (0-0) + 1 + 0 = 1$

$c[4] = (0-1) + 1 + 0 = 0 \rightarrow$ point where r and n first differ, confirmed by $\sum_{i=5}^{8} w_k = 0$, before 4th bit, hamming distance is 0.

$c[3] = (1-1) + 1 + 1 = 2$

$c[2] = (1-0) + 1 + 1 = 3$

$c[1] = (0-1) + 1 + 2 = 2$

Now, example: $n < r$

$n = 110001$

$r = 110101$

$w = 000100$; $H(n, r) = 1$

$c[6] = (1-1) + 1 + 0 = 1$

$c[5] = (1-1) + 1 + 0 = 1$

$c[4] = (0-0) + 1 + 0 = 1$

$c[3] = (1-0) + 1 + 0 = 2$

$c[2] = (0-0) + 1 + 1 = 2$

$c[1] = (1-1) + 1 + 1 = 2$

} No 0's in c

⊙ 2-party setting for $c[i]$

$$\langle c_i \rangle_j^P = j\, r[i] - \langle a[i] \rangle_j^P + j + \sum_{k=i+1}^{l} \langle w_k \rangle_j^P$$

$$P_0 : \langle c_i \rangle_0^P = -\langle a[i] \rangle_0^P + \sum_{k=i+1}^{l} \langle w_k \rangle_0^P$$

$$P_1 : \langle c_i \rangle_1^P = r[i] - \langle a[i] \rangle_1^P + 1 + \sum_{k=i+1}^{l} \langle w_k \rangle_1^P$$

$$\langle c_i \rangle = r[i] - \{\langle a[i] \rangle_0^P + \langle a[i] \rangle_1^P\} + 1 + \sum_{k=i+1}^{l} (\langle w_k \rangle_0^P + \langle w_k \rangle_1^P)$$

$$= r[i] - a[i] + 1 + \sum_{k=i+1}^{l} w_k$$

* Part 2:

⊙ if $\beta = 1$, we compute

$1 \oplus (a > r) \equiv (a \leq r) \equiv a < r+1$ over integers

⊙ corner case of $r = 2^l - 1$ which is always true dealt in part 3.

⊙ So, in this case $\beta = 1$ and $r \neq 2^l - 1$, we compute $(r+1 > a)$ using same methods as in part 2.

$$1 \oplus (a > r) \equiv (a \leq r)$$

$a > r$: True

$$1 \oplus 1 \equiv 0$$

$a > r$: False

$$1 \oplus 0 \equiv 1$$

* Part: 3) In worser case of $r = 2^k - 1$, $n \leq r$ is always true and both parties create shares of c_i having only one 0. P_0, P_1 use common values u_i to create a valid share of a 0 and $k-1$ share of 1.

If $i \neq 1$,

$$\langle c_i \rangle_j = (1-j)(u_i+1) - i u_i$$

Then,

$$\langle c_i \rangle_0^P = (1-0)(u_i+1) - 0 \times u_i$$
$$= (u_i+1)$$
$$\langle c_i \rangle_1^P = (1-1)(u_i+1) - u_i$$
$$= -u_i$$

if $i = 1$,

$$\langle c_i \rangle_j^P = (-1)^j \cdot u_i$$
$$\langle c_1 \rangle_0^P = (-1)^0 \cdot u_0$$
$$= u_0$$
$$\langle c_1 \rangle_1^P = (-1) \cdot u_0$$
$$= -u_0$$

Then,

$$\langle c_i \rangle = \langle c_i \rangle_0 + \langle c_i \rangle_1$$

or $c = 111\ldots10$

* Step 14) To ensure security against a corrupt P_2, we hide exact values of non-zero c_i's and position of possible 0 by multiplying with random s_i and permuting these values by common permutation π.