16-124-836
Marcel Matti ZAUDER

8.1 Reversing Oblivious Transfer

8.1.1 Correctness

We can raise the following equation:

$$\begin{aligned} \alpha &= m \oplus r \\ &= z \oplus y_0 \oplus r \\ &= x_c \oplus y_0 \oplus r \\ &= x_{y_0 \oplus y_1} \oplus y_0 \oplus r \end{aligned}$$

If $y_0 = y_1$, then the $\binom{2}{1}$-OTS, will return the value of $x_0 = r$, so for our equation we get:

$$\begin{aligned} \alpha &= x_0 \oplus y_0 \oplus r \\ &= r \oplus y_0 \oplus r \\ &= y_0 = y_1 \end{aligned}$$

,which is obviously the correct value $\mathbb{S}$ wants to have.
If $y_0 = \neg y_1$, then the $\binom{2}{1}$-OTS, will return the value of $x_1 = r \oplus d$, so for our equation we get:

$$\begin{aligned} \alpha &= x_1 \oplus y_0 \oplus r \\ &= r \oplus d \oplus y_0 \oplus r \\ &= d \oplus y_0 \end{aligned}$$

, which is $\alpha = y_0$, if $d = 0$, and $\alpha = y_1$, if $d = 1$.

8.1.2 Security for S

The receiver $\mathbb{R}$ only learns the blinded values of either x_0 or x_1. Therefore it cannot derive from these values what the chosen value of d was.

8.1.3 Security for R

Because $\mathbb{R}$ is calculating $x_c \oplus y_0$, $\mathbb{S}$ cannot calculate the other value, which $\mathbb{S}$ did not receive, because it cannot work back the index of which x_c $\mathbb{R}$ has gotten from the $\binom{2}{1}$-OTS. Therefore also the security for $\mathbb{R}$ is given.

8.2 More efficient Oblivious Transfer

8.2.1 Protocol

Key Generation

We know that $n = 2^k$ is the number of inputs. Let $l_1, ..., l_k$ and $r_1, ..., r_k$ be random bits. Furthermore $j_0 = 0$. We now consider a balanced binary tree of depth k and j_0 as its root node. Let $j_{i,x}$ be any non-root node in the tree, whereas $i \geq 1$ and $x \in \{l, r\}^i$ which indicates which left and right edges to travers to reach each node.
All $j_{i,x}$ are defined as follows:

$$j_{i,x'l} \;=\; j_{i-1,x'} \oplus l_i \qquad \textit{For left children nodes}$$

$$j_{i,x'r} \;=\; j_{i-1,x'} \oplus r_i \qquad \textit{For right children nodes}$$

$x' \in \{l, r\}^{i-1}$ describes the sequence of edges to reach the predecessor of node $j_{i,x}$.
The leaf nodes of the tree are named $k_0, ..., k_{n-1}$. This leaves are random bits, which are the solution of an XOR operation of random bits. Furthermore for each k_i, k_j, where $j \neq i$ the XOR operation sequence differs in at least one instance applied to j_0 to produce it. Because all l_i and r_i are randomly chosen the knowledge of any k_i does not leak any information of another k_j where $i \neq j$.

$\binom{n}{1}$-Oblivious Transfer

We are now constructing a protocol for an $\binom{n}{1}$-OT, where $n = 2^k$:

S$(x_0, ..., x_{n-1})$		**R(y)**
$j_0 = 0$		$k_y = 0$ Binary representation of y: $(y_1 ... y_k)_2 := y$
Key generation as shown in the previous section FOR $i = 1$ TO k: $l_i \leftarrow \{0, 1\}$ $r_i \leftarrow \{0, 1\}$ $j_{i,x'l} \;=\; j_{i-1,x'} \oplus l_i$ $j_{i,x'r} \;=\; j_{i-1,x'} \oplus r_i$ *Setting up the leaves:* $k_0, ..., k_{n-1} \;=\; j_{k,l...l}, ..., j_{k,r...r}$		
$c_i = x_i \oplus k_i$	$\xrightarrow{c_0,...,c_{n-1}}$	
FOR $i = 1$ TO k:	$\xrightarrow{l_i, r_i}$ $\boxed{\binom{2}{1}\text{-OT}}$ $\xleftarrow{y_i}$ $\xrightarrow{j_i}$	FOR $i = 1$ TO k:
		$k_y = k_y \oplus j_i$ RETURN $c_y \oplus k_y$

8.2.2 Cost

We assume that the $\binom{2}{1}$-OT is implemented as defined in the lecture. Therefore this protocol uses a total of:

- k $\binom{2}{1}$-OT are used, therefore:
  - $2k$ public-key operations
  - $3k$ message rounds

  - $O(k(1+\lambda)) = O(k + k\lambda)$ bits
- $2k$ random bits
- $2^{k+1} - 2$ XOR operations for key generation
- 2^k XOR operations for encryption
- k XOR operations for key retrieval
- 1 XOR operation for decryption
- $2^k = n$ bits and one message round for the transfer of the ciphertexts

Therefore our total costs are:

Computational Cost
$2k = 2\log_2(n)$ expensive public-key operations and $O(n)$ cheap XOR operations

Latency
$3k + 1$ message rounds

Communication Complexity
$O(n + k\lambda)$ bits transferred