Report on Committed Oblivious Tranfer

Jiahong Shen, Kairong Luo, Yushen Wu
Institute for Interdisciplinary Information Sciences
Tsinghua University

1 Introduction

Besides cards, there's also many ways to implement bit commitment. In this report, we will view bit commitment protocol as a black box, igoring how it's actually implemented, and discuss some algorithms on committed bits:

- k-XOR operation
- Coping
- Oblivious Transfer

Alice, who makes the commitments, can neither change her mind nor cheat during these operations. Also, the other agent Bob cannot learn anything unexpected about the committed bits, except the outputs of specified operations.

2 Notations

- In ordinary **Bit Commitment(BC)**, we use the notation $\{a\}$ the the committed bit that Alice sends to Bob, if she is able to reveal it as a later, but Bob is not able to learn anything about a by himself. Also, Alice cannot change her mind to open $\{a\}$ as $\bar{a}$.
- Also, we use the notation $\boxed{a}$ to represent a **Bit Commitment with XOR relation (BCX)** on the bit a, which will be introduced in section ?.

3 A Special Structure of BC: Bit Commitment with XOR relation (BCX)

Here we introduce a special BC protocol, called **Bit Commitment with XOR(BCX)**, in order to make use of its nice properties:

1. To commit to b using a BCX $\boxed{\text{b}}$, Alice uses n pairs of plain BCs

$$(\{b_{1L}\}, \{b_{1R}\}), (\{b_{2L}\}, \{b_{2R}\}), \cdots, (\{b_{nL}\}, \{b_{nR}\})$$

such that each pair XORs to b:

$$b_{iL} \oplus b_{iR} = b, \forall i \in \{1, \cdots, n\}.$$

2. To open $\boxed{b}$, Alice opens all $2n$ plain BCs, and Bob accepts if all pairs XOR to the same value (thus accept it as b).

3.1 Protocol of proving an XOR relation

A nice property of BCX is that, Alice is able to prove to Bob that some BCXs $\boxed{b^k}$ satisfy a k-XOR relation

$$\bigoplus_{i=1}^{k} b^i = c$$

without giving away any one of their values. In the next, we'll present the protocol:

Denote the k BCXs as plain BCs:

$$(b^1_{1L}, b^1_{1R}), (b^1_{2L}, b^1_{2R}), \cdots, (b^1_{nL}, b^1_{nR}) \tag{1}$$

$$(b^2_{1L}, b^2_{1R}), (b^2_{2L}, b^2_{2R}), \cdots, (b^2_{nL}, b^2_{nR}) \tag{2}$$

$$\vdots$$

$$(b^k_{1L}, b^k_{1R}), (b^k_{2L}, b^k_{2R}), \cdots, (b^k_{nL}, b^k_{nR}) \tag{3}$$

Bob generates k random n-permutations $\sigma_1, \sigma_2, \cdots, \sigma_k$ to Alice, and asks Alice to shuffle the BCX table in the following way:

$$(b^j_{iL}, b^j_{iR}) \mapsto (b^j_{\sigma_j(i)L}, b^j_{\sigma_j(i)R}), 1 \le i \le n, 1 \le j \le k. \tag{4}$$

That is, to shuffle the j-th row using the j-th permutation σ_j. For simplicity, we still use hte table above to denote the result the result of permutations.

After the permutation, Alice calculates and announces XOR results of each column:

$$c_{iL} = \bigoplus_{j=1}^{k} b^j_{iL}, c_{iR} = \bigoplus_{j=1}^{k} b^j_{iR}, 1 \le i \le n$$

For each $i = 1, 2, \cdots, n$, Bob randomly asks Alice to reveal all $\{b^j_{iL}\}$ or $\{b^j_{iR}\}$, $1 \le j \le k$, and check whether they're consistent with c_{iL} or c_{iR}. Also, Bob checks whether $c_{iL} \oplus c_{iR}$ turns out to be the same for all i.

If so, then Bob is convinced that Alice has done permutations and XOR calculations correctly, and

$$c = c_{iL} \oplus c_{iR} = \oplus^k_{j=1}(b^j_{iL} \oplus b^j_{iR}) = \oplus^k_{j=1} b^j, \forall i$$

is expected to be the correct XOR result of all BCXs.

3.2 Validty

The problem is, with what probability can Alice prove to Bob a wrong XOR result?

First of all, notice that Bob accepts only if every pair of c_{iL}, c_{iR} XORs to the same result, and accept is as the result. So if Alice wants to prove a wrong result, at least one of c_{iL}, c_{iR} should be inconsistent with the column entries in the table. When Bob chooses either the left column or the right column to check, he has at least one half probability to find the inconsistency.

And that the n columns are randomly permuted, so the event of finding error in each column is independent, thus the probability that Bob is convinced is no more that 2^{-n}, which is negligible.

3.3 Coping a BCX

The proof above destroys the BCXs involved, so we need to introduce a copying protocol. In this protocol we also need a supervisor Bob that checks whether Alice copies the committed bit honestly:

Denote the BCX to be copied as $\boxed{b}$. Alice creates $3n$ pairs of BCs s.t. each pairs XORs to b. Bob randomly partitions them into 3 subsets $\boxed{b^0}$, $\boxed{b^1}$, $\boxed{b^2}$ with equal cardinality. Then Alice proves to Bob that $\boxed{b} \oplus \boxed{b^0} = 0$ (i.e. $b = b_0$) using the protocol above.

If so, Bob should be convinced that $\boxed{b^1} = \boxed{b^2} = \boxed{b}$, thus the two copies are successfully produced.

And the validty is trivial: If Alice intends to produce a wrong copy, she may add some wrong pairs into the $3n$ pairs. However, when Bob randomly partitions $\boxed{b^0}$, $\boxed{b^1}$, $\boxed{b^2}$, where's only exponantially small probability that $\boxed{b^0}$ contains all correct pairs while one/both of $\boxed{b^1}$, $\boxed{b^2}$ contains all incorrect pairs.

4 Committed Oblivious Tranfer

In the next, we will focus on the main topic: oblivious transfer of committed bits (**Committed Oblivious Transfer, COT**): Suppose Alice is committed to $\boxed{a_0}$, $\boxed{a_1}$, and Bob is committed to $\boxed{b}$. After running COT($\boxed{a_0}$, $\boxed{a_1}$)($\boxed{b}$), Bob will be committed to $\boxed{a} = a_b$.

4.1 Preliminaries

- Oblivious Transer(OT): we need the assumption that we can de ordinary OT on a_0, a_1, b, so this is not suitable for cards. (Q: can we do OTs with cards?)
- Linear coding: a $[n, k, d]$ linear coding is C a way to code k-bit messages into a n-bit codeword, s.t. each pair of distinct codewords has Hamming distance at least d. All possible outputs of coding are called **Codewords**.

The main idea of Linear coding is to add extra error-correcting bits, and we expect that the code we choose is linear-time decodable.

4.2 Linear Coding

Formal Definition. Let H be any binary matrix. The *linear code* with *parity check matrix* H consists of all binary vectors x s.t.

$$Hx^T = 0.$$

In this paper *Linear-Time Encodable and Decodable Error Correcting Codes*, the author proposed a linear coding, named **Superconcentrator Codes**, which is quite efficient such that all of these three operations can be achieved in linear time $O(n)$:

- Constructing codewords,
- Decoding codewords,
- Proving a word is a codeword,

In the next, we'll always use this coding:

4.3 Abstract Idea of the Protocol

Suppose Alice is committed to $\boxed{a_0}$, $\boxed{a_1}$, and Bob is committed to $\boxed{b}$. After running COT($\boxed{a_0}$, $\boxed{a_1}$)($\boxed{b}$), Bob will be committed to $\boxed{a} = a_b$.

- Step 1: Determine a public linear coding C.
- Steps 2-9: Alice randomly pick two codewords c_0, c_1. After rounds of ordinary OTs and proofs, we allow bob to learn c_b, where b is promised to be his committed bit $\boxed{b}$.
- Step 10: Alice find a linear function h, so Bob can calculate a_b from $h(c_b)$.

4.4 Protocol

In the next, we'll display all details and necessary validty proof. There're also comments on the intention of each part, and necessary validty proofs.

Step 1: Bob determines a $[n, k, d]$ linear code C and announces to Alice. (In fact we require that $d > 2(\sigma+\varepsilon)n, k > (1/2 + 2\sigma)n$ for some small but constant ε, σ, which will be discussed later).

Step 2: Alice randomly picks some $c_0, c_1 \in C$, make commitments to all bits $\boxed{c_0^i}$ and $\boxed{c_1^i}$, and proves that $\boxed{c_0^1}\boxed{c_0^2}\cdots\boxed{c_0^n} \in C$, and $\boxed{c_1^1}\boxed{c_1^2}\cdots\boxed{c_1^n} \in C$.

Step 3: Bob randomly picks subsets $I_0, I_1 \subset \{1, \cdots, n\}$, with $|I_0| = |I_1| = \sigma n, I_0 \cap I_1 = \emptyset$, and sets $b_i = \bar{b}$ for $i \in I_0$, while $b_i = b$ if $i \notin I_0$.

Step 4: Alice runs OT$(c_0^i, c_1^i)(b_i)$ and Bob gets w^i for $1 \leq i \leq n$.

Step 5: Bob tells $I = I_0 \cup I_1$ to Alice. Alice opens c_0^i and c_1^i for each $i \in I$, and Bob checks whether w^i he receives in step 4 are correct.

Step 6: If so, Bob is able to decode the w^i he receives, to make the whole word a codeword w with the decoding algorithm.

In steps 3-6, Alice transferred n bits to Bob, and Bob verified $2\sigma n$ bits among them. If Alice transferred εn bits incorrectly, the probability the Alice's mistakes are not discovered by Bob

$$p \leq (1-2\sigma)^{\varepsilon n} \tag{5}$$

is exponentially small in n.

This shows Bob can obtain $(n - \sigma n - \varepsilon n)$ bits of correct information with high probability. So, if we take $d > 2(\sigma + \varepsilon)n$, Bob can decode w correctly with high probability.

Step 7: Bob commits to this $\boxed{w^i}$ (his decoding ouput) for all $i \in \{1, \cdots, n\}$, and give a zero-knowledge proof that $w = \boxed{w^1}\boxed{w^2}\cdots\boxed{w^n}$ is indeed a codeword.

Step 8: Alice randomly chooses $I_2 \subset \{1, \cdots, n\}$ such that $I_2 \cap I = \emptyset$, $|I_2| = \sigma n$, announces I_2 to Bob and opens c_0^i, c_1^i for $i \in I_2$.

Step 9: Bob proves that $\boxed{w^i} = c_b^i$ for $i \in I_2$.

This procedure is to make sure that in steps 3-6 Bob decodes the word w honestly. If it is not the case, then Alice accepts with negligible probability:

Bob has already proved that w is a codeword, so at least $d > 2(\sigma + \varepsilon)n$ bits are different between w and c_b. Alice verified σn bits, so similarly we can see she accepts with probability

$$p \leq \left(\frac{n-d}{n}\right)^{\sigma n} \leq (1-2\sigma-\varepsilon)^{\sigma n} = \varepsilon(n) \tag{6}$$

exponentially small in n.

After the procedures above, we have already established codewords c_0, c_1 for Alice, and $w = c_b$ for Bob, without leaking any information.

Step 10: Alice chooses an amplification function h s.t. $a_0 = h(c_0), a_1 = h(c_1)$, which does linear combination on bits. Then he announces this h to Bob, and Bob is able to learn $a_b = h(c_b)$ in this way. At last Bob commits to this $\boxed{a} = a_b$.

4.5 Proofs Involved in the Protocol

In the protocol above, we didn't dig into the details of two proofs. Let's discuss about that here:

The first thing is that, proof of a word $\boxed{c^1}\boxed{c^2}\cdots\boxed{c^n}$ being a codeword. According to the property of linear codes, we only need to prove some linear combinations of $c^1, c^2, \cdots, c^n$ comes to zero:

$$Hc = 0 \Leftrightarrow \sum_j H_{ij} c^j = 0, \forall i \Leftrightarrow \sum_{j:H_{ij}=1} c^j = 0, \forall i \tag{7}$$

The second is to prove that $\boxed{z} = h(\boxed{x^1}\boxed{x^2}\cdots\boxed{x^n})$ for committed z, x^i. We know that h is a linear function, so the proof also reduces to prove the value of a linear combination using XOR argument.

4.6 Complexity

We analyze the complexity in terms of the number of BCXs and OTs used by the participants:

- In step 4, they performed $O(n)$ OTs, when c_0 and c_1 are transferred.
- Also, require $O(n)$ BCXs for both Alice and Bob to commit.
- All proofs can be achieved only with $O(n)$ BCXs.

Additionally, we know we can build a BCX out of $O(n)$ ordinary BCs. So the overall complexity is $O(n)$ OTs and $O(n^2)$ BC operations.

5 Other Problems

Here we proposed some relavent questions that might be unsolved:

- In the first paper (discussed by Kairong Luo and Jiahong Shen), they're several protocols for cards. We mentioned that the way to do committed oblivious tranfer in this report isn't adaptive to cards, so whether we can do COT with cards?
- We discussed the way to prove an XOR argument with BCX, can we do more such as proving and AND/OR result on BCX? If not, can we introduce new structures to archieve this goal?

6 References

- Crépeau, C., van de Graaf, J., and Tapp, A. Committed oblivious transfer and private multi-party computation. *Advances in Cryptology - CRYPTO '95* (1995), pp. 110-123.
- F. J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.
- D. Spielman, *Linear-Time Encodable and Decodabble Error-Correcting Codes*, 27th ACM Symposium on Theory of Computing, 1995, pp. 388-397.