Project Report[*]

Bingsen Wang - Zhile Jiang

December 13, 2020

Abstract

In this project, we focus on Oblivious Transfer (OT) Extension which is a important research area related to the efficiency of secure MPC. We survey and compare several OT extension protocols. We give our own implementation and test their performance. We start by following the original passively secure extension protocol before giving further abstraction and a general method to make them actively secure.

1 Introduction

Oblivious Transfer(OT) is a fundamental primitive widely used in a variety of protocols since it directly implies secure multiparty computation(MPC) and vice versa. [IR89] shows that it's very likely that OT requires public key cryptography, thus, even a single OT is always expensive, not to mention that the number of OT required to construct secure MPC is always polynomial to the complexity of the protocol. Therefore, in order to improve efficiency of secure MPC, one possible method is trying to build a large number of OT from a small number of OT, which is the topic of this project. In CRYPTO 2003, a first practical OT extension protocol was proposed in [IKNP03]. Further improvements and related work can be found in [KK13, KOS15, CCG18, ALSZ13, ALSZ15].

Roadmap: In section 2 we show some necessary preliminaries and our notation. In section 3, we give a very simple protocol which can extend the message length of OT primitive. section 4, 5 will give out the passively secure protocol proposed in [IKNP03] and the corresponding improved version in [KK13]. We leave all the efficiency analysis to section 6 followed by our implementation and experiments in section 7. Later, in section 8, we show some further abstraction and a efficient method to make actively secure OT extension protocol proposed in [KOS15].

2 Preliminaries & Notation

We first give a definition of basic OT, then our notation before a general definition of OT extension.

2.1 Oblivious Transfer

Let's first see the definition of basic 1 out of 2 oblivious transfer.

Definition of 1 out of 2 Oblivious Transfer

- Input to Sender: m_0, m_1
- Input to Receiver: a bit b

After running the protocol, Receiver should learn the message m_b corresponding to his choice.

We say such oblivious transfer is secure if Receiver learns nothing about the other message m_{1-b}, Sender learns nothing about the choice b of the Receiver.

[*]This is a group project adviced by Carsten Baum in the course Cryptographic Computing fall 2020 at Aarhus university.

We say it is passively secure if it is secure when both party does not deviate from the protocol. If it is secure even if we allow both party or adversary deviate from the protocol, we say it is actively secure.
We can define the message $m_i = f(x, i)$, and $b = y$. Then the above oblivious transfer is a secure function evaluation between the Sender and Receiver over function f with x as input of Sender and y as input of Receiver.
We present a general construction from public key encryption to passively secure oblivious transfer and leave the actively secure version to section 8.

A General Construction of 1 out of 2 Oblivious Transfer from public key encryption

Choose: With choice b, Receiver sample (pk_b, sk_b) and random a random pk_{1-b}. Send (pk_b, pk_{1-b}) to Sender.
Transfer: Sender computes $c_i = Enc_{pk_i}(m_i)$. Send (c_0, c_1) to Receiver.
Retrieve: Receiver computes $m_b = Dec_{sk_b}(c_b)$.
Security requires a method to sample random public key which is indistinguishable from public key generated normally from a secret key. This is not difficult in some public key cryptosystem like ElGamal.

2.2 Notation

We follow the notation used in [IKNP03] and [KK13] before section 8.
We use OT_l^m to denote m 1 out of 2 oblivious transfer with message length l bits. $\binom{n}{n'} - \mathsf{OT}_l^m$ to denote m n' out of n oblivious transfer with message length l bits. $[m] = \{1, 2, ..., m\}$. In the protocol description, we use capital letters to denote matrices like T, Q, small bold letters for vectors like $\mathbf{s}$. $\mathbf{t}_j$ for jth row of T, $\mathbf{t}^i$ for ith column of T. For a bit b, $b \cdot \mathbf{v}$ evaluates to $0^{|\mathbf{v}|}$ when $b = 0$ otherwise $\mathbf{v}$. $\oplus$ for entry-wise xor, $\odot$ for entry-wise and. PRG for pseudo random generator, RO for random oracle. $\mathcal{H}$ to denote instance of RO, $\mathcal{G}$ for PRG. Normally, we use $\mathcal{S}$ for Sender, $\mathcal{R}$ for Receiver.

2.3 OT extension

Here comes our main topic, given a OT_k^k, can we build OT_l^m with $m \geq k$ and $l \geq k$? We say this is a reduction from OT_l^m to OT_k^k. The author in [IR89] shows a reduction from any OT_k^k to one way function imply $\mathsf{P} \neq \mathsf{NP}$, but [Bea96] shows that one way function is enough to reduce large number of OT to small number of OT.

3 Reducing OT_m^1 to OT_k^1

In this section, we present an extremely simple OT extension protocol which can extend the message length of any OT primitive.

Extending message length

- Input of $\mathcal{S}$: pair of m-bit string (m_0, m_1).
- Input of $\mathcal{R}$: a selection bit r.
- Common Input: security parameter k, PRG $\mathcal{G} : \{0,1\}^k \to \{0,1\}^m$, an ideal OT_k^1 primitive.

1. $\mathcal{S}$ samples pair of random k-bit seeds (s_0, s_1).
2. $\mathcal{S}, \mathcal{R}$ invoke the OT_k^1 primitive, where S acts as a sender with inputs (s_0, s_1) and $\mathcal{R}$ as receiver with selection r.
3. $\mathcal{S}$ computes $y_i = x_i \oplus G(s_i)$ and send (y_0, y_1) to $\mathcal{R}$.
4. $\mathcal{R}$ outputs $x_r = y_r \oplus G(s_r)$.

The correctness is straightforward. The security of OT_m^1 is also reduced to the security of OT_k^1 primitive when the PRG is good enough. This protocol is quite efficient as the only overhead is two call the the PRG. What we can not ignore is that the security of OT_m^1 actually decreased. Consider an ideal OT_m^1 with perfect security, the receiver should have probability only 2^{-m} to successfully guess another message. But in this protocol, the receiver only have to guess another seed for another message, his success probability increase from 2^{-m} to 2^{-k}.

4 Reducing OT_l^m to OT_m^k

In this section, we present the first practical OT extension protocol proposed in CRYPTO 2003 [IKNP03].

4.1 The protocol

IKNP protocol

- Input of $\mathcal{S}$: m pairs message $(x_{j,0}, x_{j,1})$ of l-bit strings for $j \in [m]$.
- Input of $\mathcal{R}$: m selection bits $\mathbf{r} = \{r_1, r_2, ..., r_m\}$.
- Common Input: security parameter k, RO $\mathcal{H} : [m] \times \{0,1\}^k \to \{0,1\}^l$, an ideal OT_m^k primitive.

1. $\mathcal{S}$ samples random $\mathbf{s} \in \{0,1\}^k$, $\mathcal{R}$ samples a random $m \times k$ bit matrix T.
2. $\mathcal{S}$, $\mathcal{R}$ invoke the OT_m^k primitive, where $\mathcal{S}$ acts as receiver with input $\mathbf{s}$, $\mathcal{R}$ acts as sender with input $(\mathbf{t}^i, \mathbf{r} \oplus \mathbf{t}^i)$, for $i \in [k]$. [a]
3. Let Q denote the matrix received by $\mathcal{S}$ in step 2. For $j \in [m]$, $\mathcal{S}$ sends $(y_{j,0}, y_{j,1}) = (x_{j,0} \oplus \mathcal{H}(j, \mathbf{q}_j), x_{j,1} \oplus \mathcal{H}(j, \mathbf{q}_j \oplus \mathbf{s}))$
4. For $j \in [m]$, $\mathcal{R}$ outputs $x_{j,r_j} = y_{j,r_j} \oplus \mathcal{H}(j, \mathbf{t}_j)$.

[a] Here, both party only have to run the primitive OT_m^k one time.

To see the correctness, consider the matrix Q received by $\mathcal{S}$ in step 2, we can easily see that $\mathbf{q}^i = (s_i \cdot \mathbf{r}) \oplus \mathbf{t}^i$ and thus $\mathbf{q}_j = (r_j \cdot \mathbf{s}) \oplus \mathbf{t}_j$. Thus, there is always one input to $\mathcal{H}$ known by $\mathcal{R}$ and another one not known since $\mathcal{R}$ doesn't know $\mathbf{s}$. This protocol is practical since the overhead during reduction is majorly some matrix operation and $2m$ RO evaluation.

4.2 Passive Security

Let's prove the security of the original IKNP protocol by giving a simulation of view of any malicious $\mathcal{S}^*$ and semi-honest [1] $\mathcal{R}$.

Simulating $\mathcal{S}^*$ In the IKNP protocol, any $\mathcal{S}^*$ will only receive $\mathbf{q}^i = (s_i \cdot \mathbf{r}) \oplus \mathbf{t}^i$. Since T is random from $\mathcal{R}$, it will give a perfect hide of the choice $\mathbf{r}$. Thus, during the entire protocol, any $\mathcal{S}^*$ will only see some random vector independent with choice of $\mathcal{R}$ which can be directly simulated.

Simulating semi-honest $\mathcal{R}$ Given x_{j,r_j} which is supposed to learn by $\mathcal{R}$, let other message $x_{j,1-r_j} = 0^l$ and run the protocol. The only different variable between the simulator and real process is $\int$. Therefore, as long as the distinguisher can not guess $\int$ used in real process he can not win. The probability that distinguisher guess it right is only 2^{-k}. Thus, this simulation is statistically close to the real process.

[1] semi-honest means he will not deviate from the protocol

5 Reducing $\binom{n}{1} - \mathsf{OT}_l^m$ to OT_m^k

In this section, we present a further improved protocol proposed in [KK13] based on IKNP. This protocol extend IKNP to 1 out of n OT with little overhead. With the message combining technique, this will give significant efficiency improvement especially for short secrets.

5.1 The protocol

KK13 construction

- Input of $\mathcal{S}$: m tuples message $(x_{j,1}, ..., x_{j,n})$ of l-bit strings for $j \in [m]$.
- Input of $\mathcal{R}$: m selection integers $\mathbf{r} = (r_1, ..., r_m)$.
- Common Input: security parameter k, codeword $\mathbf{c} = (\mathbf{c}_1, ..., \mathbf{c}_k)$ [a] of k-bit string, RO $\mathcal{H} : [m] \times \{0,1\}^k \to \{0,1\}^l$, an ideal OT_m^k primitive.

1. $\mathcal{S}$ samples random $\mathbf{s} \in \{0,1\}^k$. $\mathcal{R}$ samples random $m \times k$ matrices T_0, T_1 in which $\mathbf{t}_{0,j} \oplus \mathbf{t}_{1,j} = \mathbf{c}_{r_j}$. [b]
2. $\mathcal{S}$, $\mathcal{R}$ invoke the OT_m^k primitive, where $\mathcal{S}$ acts as receiver with input $\mathbf{s}$, $\mathcal{R}$ acts as sender with inputs $(\mathbf{t}_0^i, \mathbf{t}_1^i)$ [c] for $i \in [k]$.
3. Let Q denote the matrix received by $\mathcal{S}$ in step 2. For $j \in [m], r \in [n]$, $\mathcal{S}$ sends $y_{j,r} = x_{j,r} \oplus \mathcal{H}(j, \mathbf{q}_j \oplus (\mathbf{c}_r \odot \mathbf{s}))$ to $\mathcal{R}$.
4. For $j \in [m]$, $\mathcal{R}$ outputs $x_{j,r_j} = y_{j,r_j} \oplus \mathcal{H}(j, \mathbf{t}_{j,0})$.

[a] There is k code each of length k-bit.
[b] $\mathbf{t}_{i,j}$ here denote jth row of matrix T_i.
[c] $\mathbf{t}_j^i$ here denote the ith column of matrix T_j.

Just as the IKNP protocol, the correctness of KK13 construction is straightforward as $\mathbf{q}_j \oplus \mathbf{t}_{j,0} = \mathbf{c}_{r_j} \odot \mathbf{s}$. Since a semi-honest $\mathcal{R}$ can not get $\mathbf{s}$, he can not compute the mask over other message.

5.2 Passive Security

Let's prove the security of KK13 construction by giving a simulation of view of any malicious $\mathcal{S}^*$ and semi-honest $\mathcal{R}$.

Simulating $\mathcal{S}^*$ Just as in the IKNP protocol, any $\mathcal{S}^*$ only learns a random matrix Q during the entire process, which can be simulated directly.

Simulating semi-honest $\mathcal{R}$ Given the message x_{j,r_j} for $j \in [m]$ which is supposed to be learned by $\mathcal{R}$. Let other message $x_{j,[n]/r_j} = 0^l$ for $j \in [m]$ and run the protocol. Consider the mask $\mathcal{H}(j, \mathbf{q}_j \oplus (\mathbf{c}_{r_j} \odot \mathbf{s}))$ which is a mask $\mathcal{R}$ should know. Consider another mask $\mathcal{H}(j, \mathbf{q}_j \oplus (\mathbf{c}_{r'} \odot \mathbf{s}))$ for $r' \neq r_j$ which $\mathcal{R}$ doesn't know. If $\mathbf{c}_{r'}$ has only on bit different from $\mathbf{c}_{r_j}$, the distinguisher can easily guess it right and win, in a real process, $\mathcal{R}$ can also guess other mask without deviating from the protocol. Therefore, it is necessary that minimal hamming distance between any two different code is large enough, we denote this as d. Since there is m tuples, the probability that distinguisher win is $m2^{-d}$.

Codeword To achieve passive security, we need to make sure $d = \Omega(k)$. In [KK13], the author choose to use Walsh-Hadamard codes in which $d = k/2$.

6 Efficiency Analysis

6.1 Message Combining

In this subsection, we present a simple technique to get OT_l^m given $\binom{n}{1} - \mathsf{OT}_{l\log n}^{m/\log n}$ primitive. We assume n is an integer power of 2 and $(\log n)|m$.

Message combining

- Input of $\mathcal{S}$: m pair message $(x_{j,0}, x_{j,1})$ of l-bit strings for $j \in [m]$.
- Input of $\mathcal{R}$: m selection bits $\mathbf{r} = \{r_1, r_2, ..., r_m\}$.
- Common Input: an ideal $\binom{n}{1} - \mathsf{OT}_{l\log n}^{m/\log n}$ primitive.

1. $\mathcal{S}$ computes $m/\log n$ message $(x'_{j,0}, \ldots, x'_{j,n})$ of $l\log n$-bit strings for $j \in [m/\log n]$, $x'_{j,k} = \langle x_{j\log n+1,k_1}, \ldots, x_{(j+1)\log n,k_{\log n}} \rangle$, where $k \in [n]$ and k_i is the i-th bit in the binary expansion of k. $\mathcal{R}$ computes $m/\log n$ ingeters $\mathbf{r}'$, in which $r'_j = \sum_{i=1}^{\log n} r_{j\log n+i} 2^{i-1}$.
2. $\mathcal{S}, \mathcal{R}$ invoke the $\binom{n}{1} - \mathsf{OT}_{l\log n}^{m/\log n}$ primitive, where S acts as a sender with inputs $(x'_{j,0}, \ldots, x'_{j,n})$ and $\mathcal{R}$ as receiver with selection $\mathbf{r}'$, notate the output of $\mathcal{R}$ as $m/\log n$ messages z_j of $l\log n$ bit strings for $j \in [m]$.
3. For $j \in [m]$, $\mathcal{R}$ outputs the $(j \mod \log n)$-th substring of length l in $z_{\lfloor j/\log n \rfloor}$.

Clearly, this protocol is secure since there is no communication between sender and receiver except invoking the ideal primitive.

6.2 Security Loss

In the analysis of security of IKNP and KK13, we see there is a security loss in KK13 construction since the security of KK13 depends on the minimal hamming distance d of codeword. We make sure $d = \Omega(k)$ but d is at most k since ths length of code is k. And if $d = k$, KK construction is exactly the IKNP construction. So $d < k$, there always exists security loss no matter which kind of codeword we choose. To achieve k security, we need to set the security parameter of KK13 construction as $k' = \frac{k}{d} * k$. This brings an overhead.

6.3 Communication Complexity

The communication cost of OT_l^m using IKNP construction is $O(m(k+l))$. And the communication cost of $\binom{n}{1} - \mathsf{OT}_l^m$ using KK13 construction is $O(m(k+nl))$. By message combining, the cost of OT_l^m is equal to the cost of $\binom{n}{1} - \mathsf{OT}_{l\log n}^{m/\log n}$. So we get OT_l^m with cost $O((m/\log n) * (k + nl\log n))$. By choosing $n\log n = k/l$, it equals to $O(km/\log(k/l))$.

Hence, KK13 construction achieves logarithmic factor improvement while $l = 1$.

7 Implementation and Experiment

One goal of this project is to show that the construction proposed by [KK13] [2] could be used to improve the cost of 1-out-of-2 OT in practice. So we implement IKNP protocol and KK13 construction to compare their performance.

7.1 Implementation Details

The protocols are implemented in Go language version 1.14.10. We use elliptic curve cryptosystem as public-key cryptosystem in $\mathsf{OT}_{l\log n}^{m/\log n}$. We implement elliptic curve cryptosystem using curve P256. The curve and

[2]Our code is visible at https://github.com/Heisenberge/CCproject

the operations on curve are provide by *go/crypto* and *go-ethereum/crypto*. The SHA-256 is used as hashing oracle. And we use the TCP protocol to implement the network communication.

7.2 Experimental Setting

We use the security parameter of IKNP $k_{IKNP} = 128$. As we discussed above, to achieve exactly same concrete security as IKNP, the security paremeter of KK13 $k_{KK} = 256$. According to theoretical analysis, KK13 construction offers a $\log k$ improvements if we take the length of message $l = 1$ and $n \log n = k/l$. With our security parameter, we should set $n \approx 26.9$. So we run KK13 with both $n = 16$ and $n = 32$.

We variate the number of message m from 2^7 to 2^{22}. All the messages and choices are generated uniform randomly.

We run the protocols on LAN. Sender runs on Intel Core i8-8265U processors and 8GB of memory. Receiver runs on AMD Ryzen 7 3700X and 32 GB memory. All benchmarks were taken as an average of 20 runs.

7.3 Results

Figure 1 illustrates the running time of IKNP, KK13 with $n = 16$ and KK13 with $n = 32$. It shows KK13 runs slower than IKNP while m is very small. But the situation changes when m grows. When m is relatively large, KK13 is 20% faster than IKNP with both setting. And KK13 with $n = 32$ is a little faster than KK13 with $n = 16$.

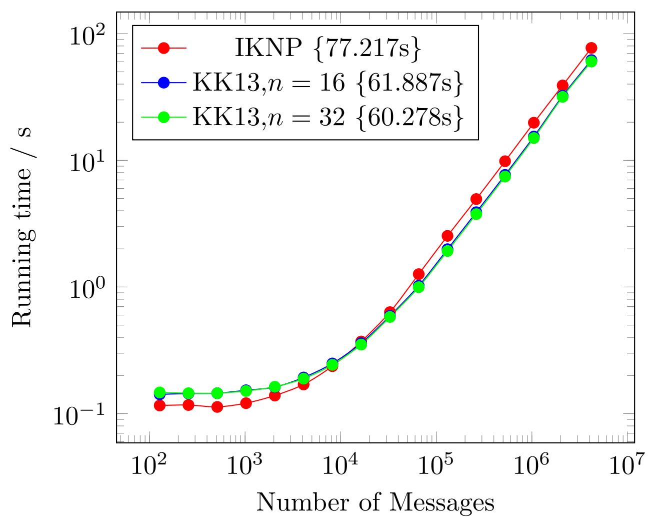

Figure 1: Performance of our implementations with various number of messages. The time of $m = 2^{22}$ is given in {}.

7.4 Analysis

The improvement of KK13 is not significant in practice. The theoretical communication cost of IKNP in our setting is $2mk + 2ml = 268m$. The theoretical communication cost of KK13 $(m/logn) * (2k + nllogn)$. Hence, the cost of KK13 with $n = 16$ is $144m$ and the cost of KK13 with $n = 32$ is $134.2m$. If the theoretical computation cost is proportional to the communication cost or communication cost dominates the total cost, the expected improvement is about $268m/144m \approx 1.86$ with $n = 16$ and $268m/134.2m \approx 2$ with $n = 32$. However, the real improvement in our experiment is just 1.25. We guess it might be caused by following reasons:

1. The computational cost depends on implementation. And our implementation is not good enough;

2. In LAN environment, the communication cost is not the majority cost.

The results from our previous experiments can support our guess. Figure 2 show the results. The main difference between these programs and final programs is the *byte* type is used to store the messages in these programs while the *bool* type is used in final programs. If the communications dominates the running time, the running time should increase significantly because this change is similar to increase the length of messages by 8. But our experiments give more mild results. So the communication cost contributes to the running time since the running time increases, but it does not dominate the running time. For computational cost, this change brings benefits. If we use bool type to store the messages, the concatenate operation considering different length of message is unnecessary. Instead, we just need a very simple type conversion before we send the messages through network. And this change also benefits the matrix operations. The operations on bool matrix are much more direct than those on byte matrix.

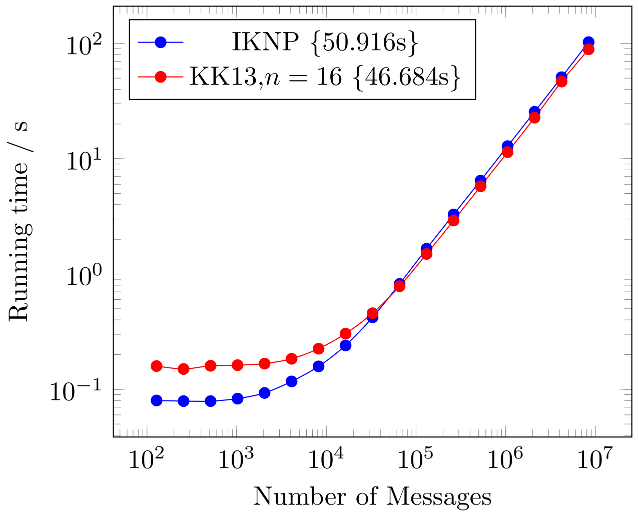

Figure 2: The performance of implementations using *byte* type to store messages. The time of $m = 2^{22}$ is given in {}.

8 Active Securiry

In this section, we talk about how to make the above mentioned OT extension protocols actively secure. Eventhough there is tool allows us to translate passively secure protocol into actively secure protocol, for better efficiency, it is interesting to try other possible method. We will first show how to do an active attack to the IKNP protocol. In [IKNP03], the author makes IKNP protocol actively secure by a cut and choose technique. Later, [KOS15, CCG18] proposed several models which give further abstraction of IKNP protocol, based on that, they give a quite efficient and general technique to make it actively secure.

8.1 An active attack to IKNP protocol

Consider step 2 in the IKNP protocol in which both party invoke the ideal OT primitive and $\mathcal{R}$ acts as sender. If we allow $\mathcal{R}$ to deviate from the protocol, he can use any vector as input message which determine the matrix Q in step 3. We denote the first input message in step 2 by $(\mathbf{a}, \mathbf{b})$, $\mathcal{R}$ can choose $a_1 = 0, b_1 = 1$ and let $a_i = b_i$ for $i > 1$. Then, the first bit of $\mathbf{q}_0$ equals to first bit of $\mathbf{s}$. Since $\mathcal{R}$ choose all the possible message, he can get other bits of $\mathbf{q}_0$. Thus, if $\mathcal{R}$ know the corresponding message in advance, he can recover the first bit of $\mathbf{s}$ by two call over $\mathcal{H}$. Using same method, $\mathcal{R}$ can let second tuple of message different in second bit and so on. If he get all the message supposed to be learned by him, he can recover entire $\mathbf{s}$ with $2m$ RO evaluation and learn all the message.
This is hard to be an attack in real process since it is not possible that $\mathcal{R}$ always learn what he should learn before the process. But this attack makes it impossible to give simulation under actively secure model.

8.2 Further abstraction of IKNP

Following the notation in [KOS15] , let's denote correlated oblivious transfer with error as COTe_k^m and give the definition.

Definition of COTe_k^m

- Input of $\mathcal{S}$: $\mathbf{s} \in \{0,1\}^k$.
- Input of $\mathcal{R}$: m vectors $\mathbf{x}_j = r_j \cdot (1, 1, ..., 1)$ [a] for $j \in [m]$.
- $\mathcal{S}$ learns matrix Q, $\mathcal{R}$ learns matrix T such that $\mathbf{q}_j \oplus \mathbf{t}_j = \mathbf{x}_j \odot \mathbf{s}$[b].

[a] r_j is a bit from $\mathcal{R}$.
[b] When $\mathcal{R}$ follow the protocol, this equals to $\mathbf{q}_j \oplus \mathbf{t}_j = r_j \cdot \mathbf{s}$.

This definition can be regarded as an abstraction of step 1 and 2 of IKNP protocol, specifically, it allow $\mathcal{R}$ do active attack by letting $\mathbf{x}_j$ has 1 on first bit and 0 on other bits. If we can give an actively secure implementation of COTe_k^m, then the step 3 and 4 of IKNP only use RO to compute mask over message, clearly give us perfect active security. In [KOS15], they achieve active security by doing a consistency check over finite field, other part is actually not much different from step 1 and 2 of IKNP protocol. They prove that an malicious $\mathcal{R}$ deviated from the protocol has only negligible probability to pass even one check.

9 Conclusion

In this project, we compared IKNP protocol and KK13 protocol from theoretical and practical perspectives. In theoretical part, we discussed the correctness, the security and the efficiency. In practical part, we focus on the efficiency. And we find although KK13 protocol brings an significant improvement theoretically, it is not easy to achieve. The communication cost is a major part of the total cost, but the computational cost cannot be ignored.

References

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer amp; Communications Security*, CCS '13, page 535–548, New York, NY, USA, 2013. Association for Computing Machinery.

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 673–701, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 479–488, New York, NY, USA, 1996. Association for Computing Machinery.

[CCG18] Ignacio Cascudo, René Bødker Christensen, and Jaron Skovsted Gundersen. Actively secure ot-extension from q-ary linear codes. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 333–348, Cham, 2018. Springer International Publishing.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 145–161, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 44–61, New York, NY, USA, 1989. Association for Computing Machinery.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved ot extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 54–70, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure ot extension with optimal overhead. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 724–741, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.