Survey of Oblivious Transfer Extensions

Natnatee Dokmai

December 15, 2014

1 Introduction

Oblivious transfer (OT) is one of the most basic important building blocks in secure computation. However, computing a large number of OTs is expensive due to the complexity of its public-key encryption primitive. Therefore there is a high demand in protocols that compute OTs efficiently. Beaver (STOC 1996) showed that it is possible to obtain $poly(n)$ OTs from given n actually OT calls using one-way functions. In addition, he showed that it is impossible to extend OTs theoretically. Another groundbreaking work was proposed by Ishai et al. (CTYPTO 2003) to show how to practically extend OTs in the random oracle model assuming passive adversary. Since then, there have been many other purposed OT-extension construction based on Ishai et al.'s protocol in various security models and under various assumptions.

My goal in this project is to survey existing OT-extension protocols and assess different aspects of those protocols, including their security assumptions, security models, and efficiency, to provide researchers the state-of-the-art of OT-extension and give the clue to what aspects existing OT-extension protocols still lack and need to be further developed.

2 Definitions and Notations

In this report, we will use k to represent the number of underlying OTs and $n = poly(k)$ the number of OTs we want to obtain from the extension. $\mathcal{S}$ represents the *sender* who transfers strings chosen by the receiver, represented by $\mathcal{R}$.

We shall use the notions of underlying OTs as described in [1] and [3]:

1. $\frac{1}{2}$OT: One-out-of-two OT, in which $\mathcal{S}$ has input bits x_0 and x_1. $\mathcal{R}$ receives (c, x_c) for a random c outside his control, but $\mathcal{S}$ does not learn c.
2. $\binom{1}{2}$OT: Chosen one-out-of-two OT, in which $\mathcal{S}$ has input bits x_0 and x_1. $\mathcal{R}$ chooses $c \in \{0, 1\}$ and obtains x_c, but $\mathcal{S}$ does not learn c.
3. OT_ℓ^k: k independent instances of $\binom{1}{2}$OT, where $\mathcal{S}$ sends strings of length ℓ.

3 OT-extension Protocols

3.1 Beaver's Protocol

Assumptions: one-way function
Security Model: static or adaptive; semi-honest or malicious

Beaver's protocol as shown in [1] is one of the first OT-extension protocols ever described. The protocol shows how to generate OT–in the sense of random number generation–using any one-way function in a black-box manner.

Protocol Overview

Beaver's protocol relies on evaluating PRG (based on a one-way function) in Yao's garbled circuit to generate polynomially many $\frac{1}{2}$OTs. $\mathcal{S}$ acts as the circuit generator who inputs $(x_{j,0}, x_{j,1}), j \in [k]n$ and $\mathcal{R}$ as the circuit evaluator who inputs a random string $r \in \{0,1\}^k$. In circuit construction, PRG is expressed as a circuit which takes in r and outputs $c \in \{0,1\}^n$. c_j is then used as a selection bit to select x_{j,c_j}. The underlying k OTs are used to generate wire labels for r in the beginning, so as a result we extend k OTs to $n = poly(k)$ OTs. [1] shows that $\mathcal{O}(n^2)$ $\frac{1}{2}$OTs are needed to construct $\mathcal{O}(n)$ $\binom{2}{1}$OTs, so we need to set the parameter to produce $\mathcal{O}(n^2)$ $\frac{1}{2}$OTs.

We can achieve different security models of the same construction by using different variations of the garbled circuit. To achieve OT-extension assuming static malicious adversary, we can apply GMW compiler or the cut-and-choose method to the garbled circuit. To achieve adaptively secure OT-extension *with erasures*, we can replace the classic Yao's garbled circuit with the two-party protocol in [2].

Protocol Efficiency

Beaver's protocol appears to be inefficient in practice. In particular it requires that operations be performed for every gate of a circuit computing, among other things, a pseudo-random generator. Suppose we want to extend k OTs into n OTs, we will have to evaluate the underlying one-way function in the circuit $\mathcal{O}(n)$ times, where each time requires evaluating $poly(k)$ gates. The complexity is thus $\mathcal{O}(n \cdot poly(k))$.

3.2 IKNP Protocol

Assumptions: correlation-robust hash function
Security Model: static semi-honest

IKPN protocol as described in [3] is the first practical OT-extension protocol. Many improved OT-extensions, both in the semi-honest and malicious model, are constructed based on this protocol.

Protocol Overview

$\mathcal{S}$ first acts as the receiver and $\mathcal{R}$ as the sender in the underlying k OTs. $\mathcal{S}$ chooses a random string $s \in \{0,1\}^k$ as the selection bits and $\mathcal{R}$ inputs $(t^i, t^i \oplus r), i \in [k], t^i \in \{0,1\}^n$ where r are the actual selection bits of $\mathcal{R}$. $\mathcal{S}$ calls her ith chosen message in the underlying OTs $q^i = t \oplus s_i r$, where q^i is the ith column of a $n \times k$ matrix Q. Finally, $\mathcal{S}$ sends the actual messages by sending $z_{j,0} = H(q_j) \oplus x_{j,0}$ and $z_{j,1} = H(q_j \oplus s) \oplus x_{j,1}$ and $\mathcal{R}$ recovers x_{j,r_i} by computing $x_{j,r_i} = H(t_i) \oplus z_{j,r_i}$, where $j \in [n]$ and $H(\cdot)$ is the hash function.

Protocol Efficiency

The protocol makes a single call to OT_n^k. In addition, each party evaluates at most $2n$ times (an implementation of) a random oracle mapping $k + \log n$ bits to ℓ bits. All other costs are negligible. The cost of OT_n^k is no more than k times the cost of OT_k^1 (the type of OT realized by most direct implementations) plus a generation of $2n$ pseudo-random bits. In terms of round complexity, the protocol requires a single message from $\mathcal{S}$ to $\mathcal{R}$ in addition to the round complexity of the OT_n^k implementation.

3.3 NNOB Protocol

Assumptions: IKNP protocol, hash function (in the random oracle model)
Security Model: static malicious

This protocol is a malicious version of IKNP protocol and is publicly known as the state-of-the-art of malicious OT-extension. The protocol uses zero-knowledge proof on the IKNP protocol to prove the consistency among r in all $(t^i, t^i \oplus r), i \in [k]$. The protocol is described in [4].

Protocol Overview

The steps of this protocol are similar to those of the IKPN protocol, except that the consistency among r will be checked before used. After OT_n^k is invoked, $\mathcal{S}$ sends a permutation $\pi(\cdot)$ and $s_i \oplus s_{\pi(i)}, i \in [k]$ to $\mathcal{R}$. $\mathcal{R}$ then sends all $k/2$ values of $H\left(t_i \oplus t_{\pi(i)} \oplus (s_i \oplus s_{\pi(i)})r\right)$ (which could be easily computed if $\mathcal{R}$ is honest), where $H(\cdot)$ is the hash function, to $\mathcal{S}$. If the values that $\mathcal{R}$ sends do not agree with the values that $\mathcal{S}$ has, she halts the protocol, otherwise she proceeds the protocol with all the ith values and ignores the $\pi(i)$th.

Protocol Efficiency

To make comparison convenient, we refer to the proof in [5]. [5] shows that in order to achieve the security level of k, $4k$ underlying OTs are needed in the protocol. Thus if we compare NNOB to IKNP protocol, NNOB requires $3k$ extra OTs, $4k$ queries to $H(\cdot)$, and $2k$ queries to equality checking.

3.4 ZDE Protocol

Assumptions: IKNP Protocol, XOR-homomorphic hash function
Security Model: static malicious

This protocol is an improvement in terms of efficiency over the NNOB protocol. The paper presenting this protocol[5], however, is currently in the peer-review process.

Protocol Overview

This protocol replaces the hash function in NNOB protocol with an XOR-homomorphic hash function and also changes how r is verified. After OT_n^k is invoked, $\mathcal{R}$ sends all $H(t^i)$ to $\mathcal{S}$, where $H(\cdot)$ is the hash function. If $s_i = 0$, $\mathcal{S}$ checks if $H(q^i) = H(t^i)$, otherwise $\mathcal{S}$ checks if all $r = H(q^i) \oplus H(t^i)$ are consistent.

Protocol Efficiency

[5] shows that to achieve the security level of k, the protocol only requires $2k$ underlying OTs. Thus the protocol halves the bandwidth requires for OTs that of NNOB protocol and demonstrates 30% speedup even on a fast LAN connection for $k = 80$.

3.5 KK Protocol

Assumptions: hash function (in the random oracle model)
Security Model: static semi-honest

This protocol as described in [6] is adapted for sending short secrets using the same approach as IKNP with improved efficiency. It trades the length of messages for more gained OTs from the extensions.

Protocol Overview

KK protocol is similar to IKNP protocol except for some few steps. $\mathcal{R}$ lets $t_{j,0} \oplus t_{j,1} = c_{r_j}$, where c_{r_j} is a Walsh-Hadamard code. $\mathcal{R}$ then sends $\{(t_{j,0}, t_{j,1})\}_{j\in[k]}$ in place of $\{(t^j, t^j \oplus r\}_{j\in[k]}$. KK protocol realizes a $\binom{n}{1}$OT from each row of the sender's matrix analogous to Q matrix in IKNP protocol. Finally, $\mathcal{S}$ sends $z_{j,r} = x_{j,r} \oplus H\left(j, q_j \oplus (c_r \cdot s)\right)$ and $\mathcal{R}$ recovers $x_{j,r_j} = z_{j,r_j} \oplus H(j, t_{j,0})$. For full detail, please refer to [6].

Protocol Efficiency

The protocol implies $\mathcal{O}(\log k)$ factor communication and computation improvement over IKNP protocol for slightly shorter secrets.

4 Assessing Differences Across Various Assumptions and Security Models

4.1 Information-theoretic security, one-way function, and the random oracle model

[1] has provided the proof that it is impossible to extend OTs in the information-theoretic setting. The known weakest assumption to extend OTs using one-way function is also presented in [1] as Beaver's protocol. However, the Beaver's protocol is still impractical due to the cost of evaluating PRG in Yao's garbled circuit. The only known practical OT-extension protocol is IKNP protocol, which is proven secure in the random oracle model.

4.2 Semi-honest and Malicious Security

[5] has demonstrated that ZDE protocol can perform less than a factor of two of IKNP protocol in term of execution time for a large number of OTs. This provides encouraging evidence that there are many opportunities for improving performance of malicious extended OT and that malicious OTs can be practically used in secure computation protocols.

4.3 Static and Adaptive Security

There are many practical OT-extension protocols assuming static adversary, however, OT-extension assuming adaptive security still has not gained much attention. An adaptively secure OT-extension protocol assuming erasures can be constructed using Beaver's protocol and the two-party protocol of [2], but it is impractical due to the nature of Beaver's protocol. The best we currently know about adaptively secure OT-extension is the result proven by [7], that the existence of OT-extension in the presence of adaptive adversaries implies the existence of an OT protocol that is secure in the presence of static adversaries. Thus, any protocol for extending OT that maintains adaptive security needs to assume, at the very least, the existence of a statically secure protocol for OT.

4.4 Shorter length of messages

[6] has shown that it is possible to gain improvement in the OT-extension if shorter messages are assumed. This can be particularly beneficial for the semi-honest GMW protocol where OTs of length 1 are used to compute AND gates. In fact, [6] has demonstrated the result of improved semi-honest GMW protocol that outperforms Yao's garbled circuit in practice.

4.5 Smaller number underlying OTs

In particular, we are interested whether it is possible to extend OTs using $\mathcal{O}(\log(k))$ underlying OTs rather than $\mathcal{O}(k)$ as in all the protocols presented in this report. The problem with $\mathcal{O}(\log(k))$ is that bounded adversaries still have the potential to brute-force all the elements in the key space. [7] demonstrates in order to extend only a logarithmic number of oblivious transfers (with security against malicious adversaries), one has to construct an oblivious transfer protocol from scratch. Thus, meaningful OT extensions exist only if one starts with a superlogarithmic number of oblivious transfers.

Protocol	**Efficiency**	**Security Model**	**Assumptions**
Beaver's	$\mathcal{O}(n \cdot poly(k))$ (for semi-honest)	semi-honest or malicious; static or adaptive	one-way functions
IKNP	$\mathcal{O}(n) + \mathcal{O}(k)$	static semi-honest	correlation-robust hash function
NNOB	$\mathcal{O}(n) + \mathcal{O}(k)$	static malicious	random oracle, IKNP
ZDE	$\mathcal{O}(n) + \mathcal{O}(k)$	static malicious	homomorphic hash function, IKNP
KK	$\mathcal{O}(n/\log(k)) + \mathcal{O}(k)$	static semi-honest	random oracle

5 References

1. D. Beaver. Correlated Pseudorandomness and the Complexity of Private Computations. In *the 28th STOC*, pp. 479–488 (1996)
2. A.Y. Lindell. Adaptively Secure Two-Party Computation with Erasures. In *Fischlin, M. (ed.) CT-RSA 2009.* LNCS, vol. 5473, pp. 117–132. Springer, Heidelberg (2009)
3. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, August 2003.
4. J.B. Nielsen, P.S. Nordholt, C. Orlandi, and S.S. Burra. A new approach to practical active-secure two-party computation. In *IACR Cryptology ePrint Archive*, 2011:91 (2011)
5. S. Zahur, N. Dokmai, and D. Evans. Faster OT-Extension Against Malicious Adversaries. Anonymous submission to *IEEE Security and Privacy 2015.*
6. V. Kolesnikov and R. Kumaresan. Improved OT extension for transferring short secrets. In *Advances in Cryptology–CRYPTO* 2013 (pp. 54-70). Springer Berlin Heidelberg.
7. A.Y. Lindell and H. Zarosim. On the feasibility of extending oblivious transfer. In *Theory of Cryptography.* Springer Berlin Heidelberg, 2013. 519-538.