OBLIVIOUS TRANSFER IN IP_2

JONATHAN LIU

1. Introduction

In 1988, Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson [BGKW88] provided a construction transforming a 2-prover Interactive Proof System to a zero-knowledge 2-prover Interactive Proof System which required no cryptographic assumptions or primitives. In doing so, they demonstrate a secure proof protocol that allows perfect zero-knowledge proofs for NP problems whose implementations require no reliance on one-way functions or any other cryptographic primitives. To do so, they begin by constructing Oblivious Transfer (OT) in the 2-prover setting, and then use this OT as a black box to design Zero-Knowledge proofs. We will focus solely on the Oblivious Transfer mechanism and its analysis, and refer the reader to the paper by James Hulett and Ruta Jawale for more detail on how this mechanism can be used. In particular, we will be revisiting and simplifying both the constructions and proof outlines given by [BGKW88], as well as noting some security issues overlooked in the bit commitment scheme of the protocol.

2. Bit Commitment

The construction of the Oblivious Transfer protocol relies on a Bit Commitment scheme that can be implemented in the 2-prover setting. We should first define the scheme.

2.1. Preliminaries.

Bit Commitment is a powerful primitive in cryptography where a Prover is forced to "commit" to a bit by sending a message to the verifier that locks the prover to that bit without revealing the bit until later in the protocol. This can be thought of analogously to bids in a silent auction: no bidder should be able to learn any other bidder's bid, but we should be able to reveal all bids once the auction is over without the ability for any bidder to change their bid. With this in mind, we can denote the first part of the protocol the "Commitment" phase, and the second part the "Reveal" phase. Takes as a whole, the scheme should satisy the following properties:

- Binding: No message m_b sent can reveal to bit b' with non-negligible probability.
- Hiding: No PPT adversary given m_b can guess b with non-negligible advantage.

Bit commitment is an important cryptographic primitive, upon which protocols like zero-knowledge proofs for NP and secure computation can be built. However, in the single-prover case, any construction of bit-commitment relies on cryptographic assumptions.

Date: May 9, 2019.

2.2. **Two-Prover Bit Commitment in [BGKW88].** In [BGKW88], the authors present a bit-commitment scheme requiring no cryptographic assumptions and operating only on the assumption that the two provers do not communicate with one another. A simplified version of their scheme to commit a bit b is presented here, split into the two parts mentioned earlier. We assume that the two provers agree beforehand on a number $s \in \{0, 1, 2\}$. Whether or not P_2 knows b has no bearing on the protocol.

Prover 1 (P_1)		**Verifier** (V)		**Prover 2** (P_2)
		Commitment Phase		
	$\xleftarrow{\pi^*}$	$\pi^* \xleftarrow{\$} \pi : \{0,1,2\} \to \{0,1,2\}$		
$m_b = (\pi^*(s) + b) \mod 3$	$\xrightarrow{m_b}$			
		Reveal Phase		
			$\xleftarrow{s}$	
$b = m_b - \pi^*(s)$				

We will now analyze the security of this procedure, beginning with the binding property. We note that because π^* is a permutation chosen by the verifier V and P_2 has received no information at all in this procedure, if we assume that V is honest then the only information P_2 can gain is that $\pi^*(s) \in \{0, 1\}$. (Note also that if this were not the case, then P_1 would halt the procedure.) Therefore, if P_2 returns some $s' \neq s$ there is a $1/2$ chance that $\pi^*(s')$ is the other bit and a $1/2$ chance that $\pi^*(s') = 2$. Thus, P_2 being dishonest about s results in a $1/2$ chance of being caught. Furthermore, this procedure can be repeated n of times in parallel if we replace s with a random ternary pad of length n and have the verifier send n random permutations. As P_2 still gets no information about the commitment phase, the prover has a $(1/2)^n$ chance of successfully breaking the commitment.

Next, we will look at the hiding property. We can see that prior to the reveal phase, the only information V gains is m_b. Knowing that $b \in \{0, 1\}$, the verifier can deduce that $s \in \{(\pi^*)^{-1}(m_b), (\pi^*)^{-1}(m_b - 1 \mod 3)\}$, but assuming that s was drawn randomly, this gives V a $1/2$ chance of guessing s and b. When run in parallel, V can narrow down each value of s in the pad to one of two options, but this is still no advantage as long as the pad consists of i.i.d. randomly chosen ternary values.

2.3. **Security Notes.** It is important to note that one bit of communication from P_1 to P_2 is all it takes to break the commitment: P_2 has narrowed the successfully cheating bit to one of two possibilities, and P_1 knows not only the two options but also which one would successfully cheat, and the selection of one item from two can be given as a single bit.

Furthermore, the authors of [CSST11] write that the bit commitment scheme defined in [BGKW88] is incomplete in the face of more nuanced procedures, arguing that a "Non-Locality box", a device that takes in inputs from separate sources and gives outputs that are individually randomly distributed yet still correlated (see [PR94]), is enough to break the commitment scheme despite not giving enough information to achieve communication. While Non-Locality boxes are believed to not be classically achievable (see [CHTW04]), they still raise a few concerns about the security of the procedure against quantum adversaries (explored in [BCMS98]), as well as the possibility of the non-locality box being simulated by a third party that the two provers are simultaneously interacting with. If the provers perform some computation with a third party unassociated with the provers in between the commitment and reveal phase, it is possible that

the result of that computation can leak enough information to P_2 to allow them to cheat and break the commitment.

To counteract this, they propose the notion of a "secure third party", that "tags" all information received from P_1 and P_2 and ensures that no computation resulting from information with the tag of one prover can go to the other prover. In the end, [CSST11] concedes that in the scenario where every third party resource available to the provers is "secure", the procedure meets the necessary security.

2.4. **Flipping Coins from Bit Commitment.** One important protocol that can be implemented from Bit Commitment is the ability for people to flip coins together. The procedure, using Bit Commitment as a black box, can be implemented as follows:

$$
\begin{array}{ccc}
P & & V \\
c_1 \xleftarrow{\$} \{0,1\} & & c_2 \xleftarrow{\$} \{0,1\} \\
& \xrightarrow{\textbf{Commit}(c_1)} & \\
& \xleftarrow{c_2} & \\
& \xrightarrow{\textbf{Reveal}(c_1)} & \\
\multicolumn{3}{c}{\textbf{coin} = c_1 \oplus c_2}
\end{array}
$$

First, given bit-commitment as a black box, we note that as long as either c_1 or c_2 is chosen at random, then $c_1 \oplus c_2$ will always be uniformly random. What is left is to examine the security in the two-prover bit commitment scheme that we use. WLOG, if P_1 and V are flipping a coin together, then P_2 gets no information at any stage in the process, and given that P_2 can only affect the reveal process successfully with probability 2^{-n}, this coin-flipping scheme is secure under our two-prover model.

3. Oblivious Transfer from Bit Commitment

We will proceed to construct Oblivious Transfer from Bit Commitment in the two-prover case. This result is surprising particularly because Oblivious Transfer has been shown to be strictly more complex than Bit Commitment (in the single-prover setting) by [IR88].

3.1. **Preliminaries.**

3.1.1. *Oblivious Transfer.* We will begin by setting up some preliminary definitions for criteria that we aim to achieve in our OT protocol. Oblivious Transfer is a protocol in which a sender sends information to a receiver, but the receiver only receives it with 1/2 chance. Furthermore, the receiver will know which outcome has occurred, but the sender will not, hence "oblivious." We specify the following criteria for our OT protocol:

- Transfer: With 1/2 probability, the receiver learns exactly what the sender sent. If this does not occur, then the receiver does not have better than a 3/4 chance of guessing what the sender sent.
- Obliviousness: The sender has no non-negligible advantage in guessing whether or not the receiver properly received the message.

Immediately, it should be surprising that the criterion for transfer allows the receiver up to 3/4 chance of guessing the sent value in the non-transfer case. We will see once we implement the protocol that it can be augmented to decrease this 3/4 chance to an arbitrarily small value.

3.1.2. *W5PBP.* A Width-5 Permutation Branching Program $\mathcal{P}$, or W5PBP, can be defined as a set of instructions $\langle i, f_i^0, f_i^1 \rangle$ and initial value $s \in [5]$ where:

- i takes values $1, 2, ..., \ell - 1$,
- f_i^0 and f_i^1 are permutations on [5] for all i,
- given some input bitstring x of length ℓ, the program is evaluated as

$$\mathcal{P}(x) = f_{\ell-1}^{x_{\ell-1}}(f_{\ell-2}^{x_{\ell-2}}(...f_1^{x_1}(f_0^{x_0}(a))...)).$$

This program can be viewed as a computation done by transitions between five states given by sequential bit values. Constructions for Width-3 PBPs have been shown to calculate modulo 3 and parity of an input in [Bar85], and the following result was given in [Bar89]:

Theorem (Barrington's Theorem)**.** *Every function computable by a circuit of depth d is computable by a W5PBP of length at most 4^d.*

We will next give a method to randomize a $W5PBP$, which we will use for our Oblivious Transfer protocol. We select a set of random permutations on [5] denoted by ϕ_i for each i, and replace our set of instructions $\langle i, f_i^0, f_i^1 \rangle$ with a new set of instructions $\langle i, g_i^0, g_i^1 \rangle$ such that

$$g_i^b(\phi_i(x)) = \phi_{i+1}(f_i^b(x)).$$

In essence, what we are doing is taking the branching program we have created, randomly permuting each set of 5 possible states, and rewriting our functions to follow these permutations.

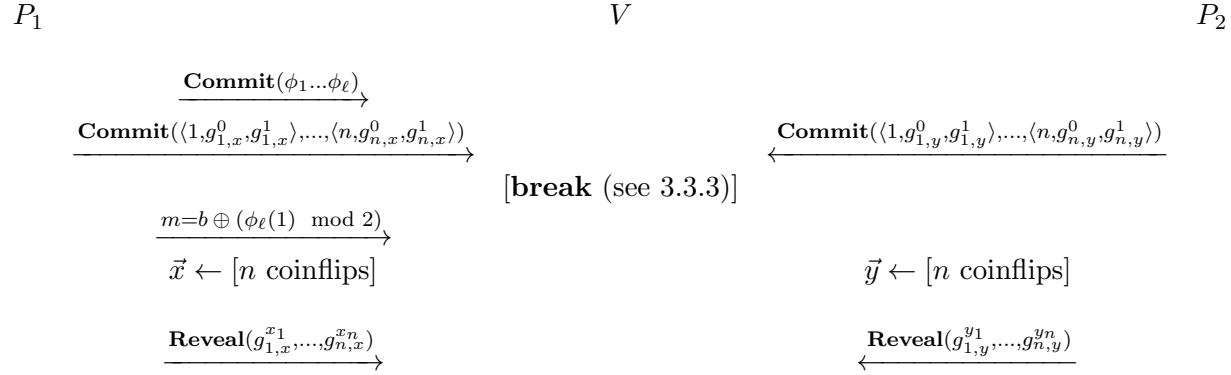
3.2. **The OT Protocol.** We now proceed to provide a description of the OT protocol constructed in [BGKW88].

3.2.1. *Setup.* The two provers and the verifier begin by agreeing on some standard W5PBP and sets $\langle i, f_i^0, f_i^1 \rangle$ for evaluating the function $\mathcal{P} : \{0,1\}^{2n} \to \{0,1\}$ for some n satisfying $\mathcal{P}(\vec{x} \mid\mid \vec{y}) = \sum \vec{x} \oplus \vec{y} \mod 2$. As half of these functions correspond to values of $\vec{x}$ and the other half correspond to values of $\vec{y}$, we will label those functions as $\langle i, f_{i,x}^0, f_{i,x}^1 \rangle$ and $\langle i, f_{i,y}^0, f_{i,y}^1 \rangle$ instead.

Note that by Barrington's Theorem this circuit can grow exponentially, but for any parameter k we can select $n = O(\log k)$ to ensure that the circuit is still of polynomial length.

Then, the two provers, without the verifier, agree on a set of permutations $\phi_1 ... \phi_\ell$, and agree on the sets $\langle i, g_{i,x}^0, g_{i,x}^1 \rangle$ and $\langle i, g_{i,y}^0, g_{i,y}^1 \rangle$ according to the specifications mentioned earlier.

3.2.2. *Interaction.* From this point, remember that we consider the two provers unable to talk to any non-secure third parties or each other. The procedure is given as follows:



At this point, the verifier can compute $g^{y_n}_{n,y}(...g^{y_1}_{1,y}(g^{x_n}_{n,x}...g^{x_1}_{1,x}(s)...))...) = \mathcal{P}(\vec{x}||\vec{y})$. Furthermore, the verifier knows the value of $\sum \vec{x} \oplus \vec{y}$ as well as $m = b \oplus \phi_\ell(1)$. Therefore, if $\sum \vec{x} \oplus \vec{y} \mod 2 = 1$, then the verifier can calculate b.

3.3. **Analysis.**

3.3.1. Transfer. Note that the probability that $\sum \vec{x} \oplus \vec{y} \mod 2 = 1$ is exactly $1/2$ given proper coinflips. We are more concerned with the ability for the verifier to recover b in the case where $\sum \vec{x} \oplus \vec{y} \mod 2 \neq 1$.

The verifier would be able to cheat if they could determine the value of $\phi_\ell(1)$, and assume WLOG that the last bit to evaluate is 0, so the verifier knows $g^0_{\ell-1}$. Due to the fact that $g^b_i(\phi_i(x)) = \phi_{i+1}(f^b_i(x))$ for all values of i, and that ϕ, f, and g are all permutations, the verifier knows all values $f^b_i(x)$, as well as $g^0_{\ell-1}$ and $\phi_\ell(0)$. Note that knowing $g^1_{\ell-1}$ would allow the verifier to determine $\phi_\ell(1)$, as flipping this one bit will flip the answer. However, the equation tells us that

$$g^1_{\ell-1}(\phi_{\ell-1}(x)) = \phi_\ell(f^1_{\ell-1}(x)),$$

and while the verifier knows $(f^b_{\ell-1})^{-1}(1)$ as well as $\phi_{\ell-1}(x)$ from the calculation, the verifier is still missing both $g^1_{\ell-1}$ and $\phi_\ell(1)$, both of which rely on ϕ_ℓ. As long as ϕ_ℓ is a randomly generated permutation, there will be no way for the verifier to recover any of the g functions not given, and as a result will not be able to learn any information about ϕ_ℓ, other that $\phi_\ell(1) \neq \phi_\ell(0)$ for the value of $\phi_\ell(0)$ found.

However, the verifier is still afforded some room for predictive ability in guessing b. Note that if $\phi_\ell(0)$ is even, then of the four possibilities left for $\phi_\ell(1)$, only one of the four is even. Thus, assuming that ϕ_ℓ is randomly chosen, there is a $3/4$ chance that $\phi_\ell(1) \mod 2 \equiv 1$, which means that there is a $3/4$ chance that $b = m \oplus 1$. This still falls within the transfer criterion we defined earlier. Furthermore, note that if we were to expand this PBP to one of larger width, we can greatly diminish the advantage granted here with little cost to runtime.

3.3.2. Obliviousness. From the same coin-flipping argument earlier, we note that the knowledge of half of any XOR gives no advantage in finding the value, and given that the sum is simply the XOR of each individual XOR, there is no way for either prover to guess the outcome with probability greater than $1/2$ without knowing the entire other vector. We note also that the

coin-flipping mechanism, despite involving both parties, leaks no information to the non-flipping prover. A random bit vector of length n can not be communicated in time $o(n)$, so at least n bits of communication are required between the two provers to break obliviousness.

3.3.3. *Malicious Provers.* Now, we consider the ability for the provers to cheat in the protocol. We note that the ability for the provers to alter the permutations once the vectors $\vec{x}$ and $\vec{y}$ have been computed relies on their ability to break the commitment scheme, which we have argued earlier is not possible with non-negligible probability. What remains is to examine the ability for the provers to give a "cheating" W5PBP to begin with. Note that once m has been sent to the verifier, if the entire circuit was to be revealed to V as well, then the verifier can always determine b; it is the fact that V only learns half of the PBP that allows the transfer to occur with probability $1/2$.

For this reason, we give the verifier the ability to break the protocol once the commitments have been sent, forcing the provers to reveal all commitments instead of sending m. In doing so, the verifier can verify that the committed W5PBP is completely valid. By introducing this ability to check, it forces any provers trying to cheat by sending an incorrect W5PBP to take the risk of failing the protocol altogether. If the W5PBP is indeed valid, then the provers must provide a new set of permutations ϕ for the protocol and try again. Because the verifier knows f anyway, no new function is necessary.

References

[BCMS98] Gilles Brassard, Claude Crepéau, Dominic Mayers, and Louis Salvail. "Defeating classical bit commitments with a quantum computer". In: *ArXiv Quantum Physics e-prints* (1998).

[BGKW88] Michael BenOr, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. "Multi-Prover Interactive Proofs: How to Remove Intractibility Assumptions". In: *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of Computing* (1988), pp. 113–131.

[Bar85] Steven A. Barrington. "Width-3 Permutation Branching Programs". In: (1985).

[Bar89] Steven A. Barrington. "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1". In: *Journal of Computer and System Sciences* 38 (1989), pp. 150 –164.

[CHTW04] R. Cleve, P. Hoyer, B. Toner, and J. Watrous. "Consequences and limits of nonlocal strategies". In: *CCC '04: Proceedings of the 19th IEEE Annual Conference on Computational Complexity* (2004), pp. 236–249.

[CSST11] Claude Crepéau, Louis Salvail, Jean-Raymond Simard, and Alain Tapp. "Two Provers in Isolation". In: *Advances in Cryptology - ASIACRYPT 2011* (2011), pp. 407–430.

[IR88] Russell Impagliazo and Steven Rudich. "Limits on the Provable Consequences of One-Way Permutations". In: *Advances in Cryptology CRYPTO 88. CRYPTO 1988* (1988), pp. 8–26.

[PR94] S. Popescu and D. Rohrlich. "Nonlocality as an axiom." In: *Foundations of Physics* 24 (1994), p. 379.