Building Oblivious Transfer from communication delays

Paolo Palmieri Olivier Pereira
Université catholique de Louvain
UCL Crypto Group
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium
paolo.palmieri@uclouvain.be olivier.pereira@uclouvain.be

Abstract

In a scenario with two mutually distrusting players, Oblivious Transfer, a rather fundamental primitive in the design of cryptographic protocols, cannot be implemented with unconditional security over a standard, error-free communication medium. Various results, however, show that we can make use of *noisy channels*, where we can exploit errors in the communication to our advantage in order to implement OT in a secure fashion. First tested against standard primitives like the Binary Symmetric Channel, it has later been demonstrated how to build OT over a fair number of new noisy channels models proposed over the years. However, these models are usually derived from the BSC itself, and different sources of noise have been scarcely explored.

In this paper we propose a new noisy channel primitive, called Binary Discrete-time Delaying Channel. The aim is to model a realistic scenario in communication systems, while basing it on as few assumptions as possible. In particular, we make use of a rarely used error source, the delays in communication. We also provide a way to securely build OT over this new model in the semi-honest scenario, and then we add some robustness to the protocol, mitigating the influence of a cheating transmitter. The flexibility and generality of this new model may open the way for future implementation, especially in media where delays are a fundamental characteristic, as in the case of wireless communications.

1 Introduction

The relevance of Oblivious Transfer (OT), as well as of other primitives like Bit Commitment, in the design of cryptographic protocols is widely recognized [1, 2]. However, in a scenario with two mutually distrusting players, such a fundamental primitive cannot be implemented with unconditional security over a standard, error-free communication medium. Thus the importance of using noisy channels, where we can exploit errors in the communication to our advantage in order to implement Oblivious Transfer in a secure fashion. Crépeau and Kilian have in fact successfully demonstrated how to build the primitive over a Binary Symmetric Channel (BSC) [3], a theoretical communication channel where bits have some fixed probability of being flipped during the transmission. Other models of communication channels have since been designed and studied, in respect of their property of being a good medium over which to build OT. Crépeau later returned on the subject [4], and then, together with Morozov and Wolf [5], observing how OT can be built over any non trivial noisy channel. A more recent paper by Nascimento and Winter [6], explores the subject of which noisy channels are useful for obtaining OT.

Over the years, a fair number of noisy channel models have been proposed, most derived from the BSC itself, and it has been shown how to build OT over them. The

concept remains however similar: the channel is used many times by the parties, to benefit from privacy amplification, and some error correcting codes are used to ensure an error-free communication when needed. The Unfair Noisy Channel (UNC), a weaker and therefore less assuming noisy channel, was introduced by Damgård, Kilian and Salvail [7]. Instead of a fixed error probability, as in the case of a regular BSC, this channel allows for a known range of possible noise levels, and, to add more generality, it also let the potential attacker to be given the advantage of knowing exactly what the actual noise level is (from which the name "unfair"). In this context, Damgård et al. proved that a UNC can be used to build OT when the noise range is inside a specific threshold, widened in a further work [8], and that it is impossible for other values. Yet, there is still today a part of the graph (and hence a set of possible noise ranges) where it is uncertain whether or not OT could be possibly implemented with unconditional security.

In [9], Wullschleger proposes a new set of noisy channels, called Weak Noisy Channels (WNC). In particular, he revised two common primitives redesigning them into a new fashion: the Weak Erasure Channel (WEC) and the Weak Binary Symmetric Channel (WBSC). The aim of this work was to define the channels not with a predefined set of functionalities, but only by a set of conditions that the channels must satisfy. In this way, the primitives allow the attacker some more freedom. For instance, it is taken into account the possibility for a malicious player to know, with a certain probability, if the bit received through the channel was in fact correct or not. Wullschleger then successfully proves, for a certain range of parameters, that OT, or, more generally, any secure two-party computation, can be achieved on the channels.

Every new proposed channel try to ease the inherent assumptions needed to implement OT over it in a secure manner, but we are still far from reaching a practically implementable solution. Research has been, up until now, bound to information theoretic primitives like the BSC and has not provided a model apt to design a more realistic scenario, based on actual communication systems. Moreover, while it has been demonstrated that OT can be generally built on almost any noisy channel [5], as seen before, there still is lack of studies on how to use for that purpose the errors naturally present in various types of communication.

1.1 Contribution

In this paper, we present a new noisy channel primitive, called Binary Discrete-time Delaying Channel, whose aim is to model a realistic scenario in communication. In particular, we make use of a rarely used error source, the delays in communication. We then show how to build OT over this new model, and demonstrate that, in a scenario where players are honest-but-curious, the proposed channel allows for any secure two-party computation to be achieved. The last part of the paper analyzes how to add some robustness to the protocol, mitigating the influence of a cheating transmitter.

The flexibility and generality of this new model may open the way for future implementation, especially in media where delays are a fundamental characteristic, as in the case of wireless communications.

2 Transmission delay as a source of noise

Reducing or limiting delays has always been one of the main challenges in the communication field. Delay is, by definition, difficult to predict and almost impossible to eliminate. Moreover, in real and non-isolated systems it usually depends on external, non-controllable factors. It is a fundamental matter in wireless communications (see, as reference, [10]), but it is also a common problem in wired IP networks.

While having these appealing characteristics, delay has not been systematically used as a source of noise in noise-demanding security application. In particular, complete

and specific studies are still missing. We will address this lack by providing a new channel model based on delay, and demonstrating how secure two-party computation can be achieved using it.

2.1 Binary Discrete-time Delaying Channel (BDDC)

Our model of communication channel is a box which accepts strings of input symbols and, for each string of symbols admitted, emits an identical output string after a certain delay. The channel operates at discrete times, which means that it is not continuously accepting inputs and emitting outputs, but these actions can only occur at specific points in time.

For simplicity, we assume that the action of accepting or emitting a string is instantaneous, that is, it takes no time to be accomplished. We also adopted the mathematical symbols $<$ and $>$, when comparing two instants in time t_0 and t_1, to express that t_0 occurs earlier in time, when using the former, or later, when using the latter, than t_1. Accordingly, when using $\leq$ and $\geq$, we are also taking into account the possibility of having t_0 and t_1 occurring at the same time.

Definition *Let $t_0 < t_1 < t_2 < \ldots$ be a sequence of distinct consecutive instants in time called* input times*, and let $u_0 < u_1 < u_2 < \ldots$ be a sequence of distinct consecutive instants in time called* output times*, where, for every input time t_i and its corresponding output time u_i, $u_i \geq t_i$. A* Binary Discrete-time Delaying Channel *with delaying probability p is a channel accepting, at discrete input times, inputs made of binary strings and emitting, at the corresponding output time, those same unchanged binary strings with a probability p, for every string admitted into the channel at any input time t_i and due to be emitted at output time u_i, to be delayed for emission until u_{i+1}. That is, if S is the string transmitted at input time t_0, and u_0 is the smallest output time for which $u_0 \geq t_0$, u_1 the second smallest and so on and so forth, then the channel is characterized by the conditional probabilities $Pr(u_0)$, $Pr(u_1)$, $Pr(u_2)$, ... where $Pr(u_i)$ is the probability of S being outputted at time u_i*

$$Pr(u_i) = p^i - p^{(i+1)}$$

Example The probability of a string S admitted at t_0 to be emitted without delay at u_0 is

$$Pr(u_0) = 1 - p$$

The channel is memoryless. A string of symbols is delayed with probability p independent of the history of symbols or delays. For instance, the probability for two strings sent at the same input time t_i of being both delayed while transmitted is p^2. Moreover, neither the sender nor the receiver gets any feedback about the transmission, i.e. they don't learn any information about whether or not a string sent or received was actually delayed.

It should be noted that there is no strict requirement regarding the discrete output times in relation to the input ones. For example, while logically u_i cannot precede on the time-line t_i, it is perfectly acceptable for the purpose of the channel both having u_i and t_i happen simultaneously, or having u_i happening later, even after any number of t_j with $j > i$.

3 Building Oblivious Transfer over a BDCC

The construction is inspired by the one proposed by Crépeau and Kilian while describing for the first time how to build OT over the BSC in [3].

As per the original concept of Oblivious Transfer (presented by Rabin in [11]) a sender Sam has two secrets bits, b_0 and b_1, and wants to communicate one of those two to a receiver Rachel, without at the same time revealing the other. Rachel wants to choose which one to receive without letting Sam know her choice s, but should not be able to learn any information other than the secret b_s she has selected.

3.1 OT from DCC in the Semi-Honest scenario

In this scenario, both parties are honest-but-curious, meaning that they follow the protocol, but try afterward to learn extra knowledge from their record of the conversation. In particular, Sam will try to guess which secret Rachel selected, while Rachel's aim is to get as much information as possible on the other secret.

Protocol

Before starting any communication, some preparatory computation needs to be completed. In particular, Sam forges a set of bit strings $C = c_1, c_2, \ldots, c_n$, all of the same length k. Then he fixes a constant $h < k$, such that, for every string c_i in the set

(a) the substring $sn\,(c_i)$ composed of the first h bits of c_i and called *sequence number*, is unique for every string in the set C;

(b) the substring $si\,(c_i)$ composed of the last $(k-h)$ bits of c_i and called *string identifier*, is unique for every string in the set C.

After that, Sam forges another set $C' = c'_1, c'_2, \ldots, c'_n$. All the strings in C' are of the same length k as those of C, and they can be logically divided into the two substrings as well, so that

(a) in any given string c'_i, the *sequence number* substring $sn\,(c'_i)$ is exactly the same as that of the string $c_i \in C$. That is, $\forall i \in 1, \ldots n, sn\,(c'_i) = sn\,(c_i)$;

(b) for every string c'_i, the *string identifier* substring $si\,(c'_i)$ is unique in the set C', and is also different from any other string identifier in C.

The values of k and h can be arbitrarily chosen, as long as they are long enough for the above properties to be respected, and the information is shared between the parties. In particular, we can gather that the values of k and h are subject, due to the requirement for the substrings of being unique, to the constraints

$$2^h \geq |C| = n$$

$$2^{k-h} \geq |C \cup C'| = 2n$$

Completed these preliminary steps, the parties are ready to proceed with the protocol as follows:

1. Sam sends the set C to Rachel using $BDDC(p)$, a Binary Discrete-time Delaying Channel with probability p. All the strings in the set are admitted into the channel at the same instant t_0.

2. Sam sends the set C' to Rachel using $BDDC(p)$ at instant t_1.

3. Rachel at instant u_0 receives over $BDDC(p)$ all the strings in the set C that have not been delayed by the channel. If less than half of the strings are received, then the number of delays is too high for the protocol to be secure, and Rachel sends instructions to Sam in order to abort the communication and start again from the beginning. In this case, even the sets C and C' have to be recreated from scratch.

4. Rachel at instant u_1 receives over $BDDC(p)$ the strings from set C delayed once, plus the strings of set C' that have not been delayed. She keeps listening on the channel at instants $u_2, u_3, \ldots$ until all the delayed strings have been received.

5. Rachel picks a set of sequence numbers I_s subject to the constraints

   (a) $|I_s| = \frac{|C|}{2} = \frac{n}{2}$

   (b) for every sequence numbers in I_s, the corresponding string $c_i \in C$ has been received at u_0

   Then, she picks a set I_{1-s} where she puts the remaining sequence numbers, regardless of their instant of arrival. Every sequence number should appear once and only once in one of the two sets, and not in the other.
   Rachel sends I_0 and I_1 to Sam over a clear channel.

6. Sam receives I_0 and I_1, and chooses a universal hash functions f, whose output is 1-bit long for any input. Let τ_c be the value corresponding to the time at which a string c has been sent (that is, $\tau_c = 0$ if the string c was sent at t_0, or $\tau_c = 1$ if the string was sent at t_1). For each set I_i, Sam computes the string str_i by concatenating each string identifier si in the set, XORed with τ_0 if the corresponding string was sent at t_0, or τ_1 if it was sent at t_1

   $$str_i = \big((si_1 \oplus \tau_1)|| \ldots ||(si_{|I_i|} \oplus \tau_{|I_i|})\big)$$

   The two strings str_0, str_1 are given in input to the hash function f to obtain the two values h_0 and h_1

   $$h_0 = f(str_0)$$
   $$h_1 = f(str_1)$$

   When the computation is complete, Sam sends to Rachel the function f and the two values

   $$d_0 = (h_0 \oplus b_0), d_1 = (h_1 \oplus b_1)$$

7. Rachel computes her guess for b_s, according to the formula

   $$b_s = f(str_s) \oplus d_s$$

Remark It should be noted that the steps 2 and 3 of the protocol could also happen in the inverse order, or simultaneously. This is due to the fact that there is no explicit constraint regarding the chronological order of t_1 and u_0.
In the same way, the string identifiers inside of the sets I_0 and I_1, and the sets themselves are sent in no particular order (so that Sam cannot get any information from this).

3.2 Why is it working?

As always in the study of security and cryptography, to show that an algorithm or protocol is *secure*, we first have to give a clear definition of what we mean by the word secure. In this case, the task is quite simple. In fact, with the proposed protocol, we are implementing the well-defined Oblivious Transfer problem. Consequently, we have to prove the two main goals of Oblivious Transfer itself:

1. the sender (Sam) sends b_0 and b_1, but learns nothing about the receiver's choice.
2. the receiver (Rachel), gets the bit b_s she has chosen, but no information about the other bit b_{1-s}.

Proof (Sketch) We are first going to address the first requirement. Sam sends the two bits during step 6. Given the fact that the Binary Discrete-time Delaying Channel gives no feedback to the sender or the receiver about which strings are delayed and which not (as per the channel definition), he can make no assumption about that. Following this, he has no mean to get any information from his own transmissions (steps 1, 2 and 6 of the protocol). During steps 3, 4 and 7, no communication at all takes place, so these steps are of no use when trying to get extra information as well. All that is left for Sam to try to exploit is the step 5, where Rachel sends back to him the two sets of identifiers I_0 and I_1. The communication here happens over a clear channel, and the sets are sent in no particular order, so Sam's only hope is to analyze the content of the transmission. The contents of the two sets are, however, already known to Sam (they are the string identifiers he sent to Rachel himself). The *discrimen*, the difference between the two from which to disclose Rachel's choice is that she put in one, and only one of the two all the delayed strings. Sam could therefore guess s, if only he had any way of knowing a priori which strings have been delayed, but this is against the channel properties, and so Sam is left clueless.

From Rachel's point of view, what's important is to receive the bit b_s she has chosen. XOR is both an associative and a commutative operation, and for its own properties the formula she uses to compute b_s

$$f(str_s) \oplus d_s = f(str_s) \oplus (f(str_s) \oplus b_s) = (f(str_s) \oplus f(str_s)) \oplus b_s = 0 \oplus b_s = b_s$$

A clarification is needed about the operands of the formula. In particular, $f(str_s)$ appears twice. However, to get the result b_s, Rachel uses two different values for each of them. One of the two can be replaced by the value h_s sent to her by Sam at the step 6, which is equivalent by definition. The other $f(str_s)$ is calculated by Rachel herself, using the hash function f she received at the same step 6 and the string identifiers of the set I_s. She will compute the result with the same formula used by Sam

$$str_i = \big((d_1 \oplus \tau_1)|| \dots ||(d_{|I_i|} \oplus \tau_{|I_i|})\big)$$

In the case of I_s, she has all the needed values. She has chosen the string identifiers $d_1 \dots d_{|I_s|}$ herself, while the $\tau_1 \dots \tau_{|I_s|}$ are clearly all 0 (she choose the strings in I_s so that they all arrived at u_0).

But we know Rachel is a curious person, and she will try to guess the other bit b_{1-s} as well. Unfortunately for her, using the same procedure as above will do no good. In fact, during step 4, Rachel has no way of distinguishing the couples of strings (c_i, c'_i) sharing the same sequence number and for which c_1 was delayed by the channel and c'_1 was not. Therefore she cannot determine accurately the values $\tau_1 \dots \tau_{|I_{1-s}|}$, needed to solve the equation and obtain the value of b_{1-s} from h_{1-s}. Again, the property of the Binary Discrete-time Delaying Channel of giving no feedback to the sender or the receiver about which strings are delayed and which not is impossible to overcome.

3.2.1 About the values of p, n

When sending strings through a delaying channel, one would usually expect that channel to actually delay at least some of those strings. But our channel only has a *probability* of delaying each string, and, as always when dealing with probabilities, we cannot rule out completely the possibility of having all of them being emitted with no delay at the first available output time. This is a particularly bad case from our point of view, since it would mean that Rachel would be able to get the values of both bits b_0 and b_1. But how realistic is this scenario? When calculating Rachel's probability of decoding b_{1-s} just in the same way of b_s, we have two variables we need to take into account: the probability of the channel p, and the number n of sequence numbers sent back to Sam in the sets I_s and I_{1-s}. In fact, the higher this number is, the more strings will have to pass through the channel unaffected. Thus the probability is equal to $(1-p)^n$.

Making sure that at least some strings are delayed is however not enough. To understand why, we are going to adopt two new terms. We call a *collision* the event in which both sequence numbers of the strings $c_i \in C$ and $c'_i \in C'$ arrive at the same output time. We call a *switch* the event in which the sequence number of c'_i arrives earlier than the one of c_i. Rachel intuitively understands that a switch is not such a likely event. In fact if, for instance, c_i is delayed once, and c'_i is delayed once as well, it is quite a safe guess to assume that the first sequence number received is c_i and the second is c'_i, rather than the other way around. If she tries to predict the set a string comes from in this way, she will make a correct guess every time the delay of c_i is equal or lesser than the delay of c'_i (no switch occurs). Taking into account also the case for which c_i is not delayed (no need to guess), this happens for each string with probability

$$Pr = \frac{1}{(1+p)}$$

Besides, one can estimate that, for the strings c_i and c'_i, the probabilities of a collision and of a switch are respectively

$$Pr(collision) = \frac{p\,(1-p)}{1+p} \qquad Pr(switch) = \frac{p^2}{1+p}$$

Since Rachel makes a correct guess with a probability higher than $\frac{1}{2}$ only when there is no collision nor any switch, we obtain that the probability for her to guess b_{1-s} correctly is

$$Pr(b_{1-s}) = \frac{1}{2} + \frac{(1+p)^{-n}}{2}$$

which provides her with a negligible advantage as soon as $p > 0$.

3.3 Robustness against cheating

Up to this point we did not take into account the possibility of having Sam or Rachel actively trying to get extra information by deviating from the protocol. We call this behavior *cheating*.

A way Sam could try to influence how Rachel builds the sets I_0 and I_1, and then try to guess which one refers to the strings she received with some delay, is to willfully hold back one or more strings from the set C and artificially "delaying" them by sending them only at instant t_1. When, at step 6, he receives I_0 and I_1, he will recognize I_{1-s} as the one containing those strings, thus being able to guess s. While we cannot prevent this behavior completely, we can make it easier for Rachel to detect Sam's cheating and act accordingly. In particular we can impose an upper bound over the number of strings Sam can hold back, by having Rachel check at steps 3 and 4 that there are not more than a certain number of delayed strings. If this happens, Rachel will reject the transmission and instruct Sam to proceed with a new one. However much we limit Sam's cheating, it is still true that only one artificially delayed string can be enough to reveal I_{1-s}. A way to address this issue is to introduce more noise in the communication, raising p and having Rachel accept at point 3 only sets with a number of delays in

$$\left[\frac{n}{2}, \frac{n}{2} + \epsilon\right]$$

At the same time we will make use of strong enough error correcting codes to ensure that Rachel always gets the correct result even with the ϵ delayed strings selected for I_s. Since both I_0 and I_1 contain delayed strings, Sam cannot compute Rachel's choice.

Rachel can also move from the protocol trying to enhance her possibilities of correctly guessing b_{1-s}. However, her options are more limited. Being only responsible, during the interaction with Sam, of choosing which string identifiers are placed either into I_s or I_{1-s}, she can try to improve her chances of decoding b_{1-s} by placing differently the string identifiers into the sets. But this has a cost, and whenever she puts one more non-delayed string into I_{1-s}, she is taking one out of I_s, thus reducing her probability of correctly decoding even the chosen bit b_s.

Acknowledgments

This research work was supported by the SCOOP Action de Recherche Concertées. Olivier Pereira is a Research Associate of the F.R.S.-FNRS. We also want to thank Abdellatif Zaidi and Luc Vandendorpe for interesting discussions on the subject.

References

[1] Y. Ishai, M. Prabhakaran, and A. Sahai, "Founding cryptography on oblivious transfer - efficiently," in *CRYPTO*, vol. 5157 of *Lecture Notes in Computer Science*, pp. 572–591, Springer, 2008.

[2] D. Chaum, I. Damgård, and J. van de Graaf, "Multiparty computations ensuring privacy of each party's input and correctness of the result," in *CRYPTO*, vol. 293 of *Lecture Notes in Computer Science*, pp. 87–119, Springer, 1987.

[3] C. Crépeau and J. Kilian, "Achieving oblivious transfer using weakened security assumptions," in *FOCS*, pp. 42–52, IEEE, 1988.

[4] C. Crépeau, "Efficient cryptographic protocols based on noisy channels," in *EUROCRYPT*, pp. 306–317, 1997.

[5] C. Crépeau, K. Morozov, and S. Wolf, "Efficient unconditional oblivious transfer from almost any noisy channel," in *SCN*, vol. 3352 of *Lecture Notes in Computer Science*, pp. 47–59, Springer, 2004.

[6] A. C. A. Nascimento and A. Winter, "On the oblivious-transfer capacity of noisy resources," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2572–2581, 2008.

[7] I. Damgård, J. Kilian, and L. Salvail, "On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions," in *EUROCRYPT*, pp. 56–73, 1999.

[8] I. Damgård, S. Fehr, K. Morozov, and L. Salvail, "Unfair noisy channels and oblivious transfer," in *TCC*, vol. 2951 of *Lecture Notes in Computer Science*, pp. 355–373, Springer, 2004.

[9] J. Wullschleger, "Oblivious transfer from weak noisy channels," in *TCC*, vol. 5444 of *Lecture Notes in Computer Science*, pp. 332–349, Springer, 2009.

[10] J. G. Proakis, *Digital Communications (4th edition)*. McGraw-Hill Science Engineering, August 2000.

[11] M. O. Rabin, "How to exchange secrets by oblivious transfer." Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981. Manuscript.