Building Oblivious Transfer on Channel Delays

Paolo Palmieri and Olivier Pereira

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

Abstract. In the information-theoretic setting, where adversaries have unlimited computational power, the fundamental cryptographic primitive Oblivious Transfer (OT) cannot be securely achieved if the parties are communicating over a clear channel. To preserve secrecy and security, the players have to rely on noise in the communication. Noisy channels are therefore a useful tool to model noise behavior and build protocols implementing OT. This paper explores a source of errors that is inherently present in practically any transmission medium, but has been scarcely studied in this context: delays in the communication.
In order to have a model for the delays that is both general and comparable to the channels usually used for OT – such as the Binary Symmetric Channel (BSC) – we introduce a new noisy channel, the Binary Discrete-time Delaying Channel (BDDC). We show that such a channel realistically reproduces real-life communication scenarios where delays are hard to predict and we propose a protocol for achieving oblivious transfer over the BDDC. We analyze the security of our construction in the semi-honest setting, showing that our realization of OT substantially decreases the protocol sensitivity to the user's knowledge of the channel compared to solutions relying on other channel properties, and is very efficient for wide ranges of delay probabilities. The flexibility and generality of the model opens the way for future implementation in media where delays are a fundamental characteristic.

Keywords: Oblivious transfer, secure multi-party computation, information theoretic security, cryptography on noisy channels.

1 Introduction

The first uses of cryptography arose from the necessity of sending a secret message to some trusted correspondent in a way that only the intended receiver could learn the information. However, we may sometime be interested in communicating with someone we do not trust. Secure multi-party computation allows several parties to perform a shared computation while preserving the secrecy of their respective inputs and the correctness of the results [2].

In the case of two-party computation, where only two players are involved in the communication, a primitive of central importance is Oblivious Transfer (OT).

In a protocol that realizes OT, a sender sends some information to a receiver, which is however able to learn only part of it, while the sender remains oblivious as to what is received. The relevance of OT is due to its universality: any other two-party computation can be achieved on top of it [10]. However, if we make no computational assumption, that is, if we assume adversaries have unlimited computational capabilities, such a fundamental primitive cannot be implemented with unconditional security over a standard, error-free communication medium. Thus is the importance of using noisy channels, where we can exploit errors in the communication to our advantage in order to implement oblivious transfer in an unconditionally secure fashion. In general, any non-trivial noisy channel can be used for this purpose [6, 11].

The first protocol for OT was built on the Binary Symmetric Channel (BSC) [5, 4], a noisy 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. Of the fair number of noisy channels proposed over the years, most are derived from the BSC itself. The Unfair Noisy Channel (UNC), a weaker and therefore less assuming noisy channel, was introduced by Damgård, Kilian and Salvail [8]. 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" is derived). In [17], 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 is 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.

Despite the differences in the channels, the respective protocols designed to build OT usually follow the same scheme: the channel is used repeatedly by the parties, to benefit from privacy amplification, and error correcting codes (ECC) are used to ensure the correctness of the communication. Unfortunately, the use of ECC's also limits the flexibility of the construction by reducing the ranges of acceptable error probabilities, while applying privacy amplification techniques implies that a considerable amount of data needs to be transmitted through the channel for each single bit of private information we want to send. These factors, along with the strong requirements still imposed by current noisy channel models, prevent any real application of oblivious transfer protocols not based on computational assumptions.

1.1 Contribution

In order to decrease the sensitivity of OT protocols to the precise knowledge of channel characteristics and make actual implementation a more realistic prospect, we propose a new noisy channel primitive, called Binary Discrete-time Delaying Channel (BDDC). The BDDC preserves the basic characteristics of the BSC: it is a binary, discrete and memoryless channel; but it is based on a common but rarely used error source, the delays in communication. Delays happen in almost any telecommunication medium, both wired and wireless, but, to the best of our knowledge, have never been used in the design of oblivious transfer protocols in the information theoretic setting before.

To show how the channel can be used to achieve any secure two-party computation, we propose a protocol that implements oblivious transfer over the BDDC, and we provide a proof of the security of our realization in a scenario where players are honest-but-curious. The protocol design has two original features that largely increase its flexibility and efficiency compared to current constructions. First, the information sent by the sender through the channel is structured in a specific way in order to exploit the peculiarities of the channel and reduce the amount of communication required. Second, the protocol does not need error correcting codes to preserve the correctness of the communication. This allows for a much larger tolerance of variations in the error probability of the channel, even during the protocol execution.

The flexibility and generality of the model opens the way for future implementation, especially in media where delays are a fundamental characteristic, as in the case of wireless communication, or wired IP networks.

1.2 Outline of the Paper

In Section 2 we introduce a new noisy channel based on data transmission delays, and we show how it actually models realistic communication scenarios. In Section 3 we provide a security definition for oblivious transfer, as well as some other useful definitions and preliminary notions which will be needed. We also propose a protocol that implements oblivious transfer over the new channel. In Section 4 we prove the security in the semi-honest model and we show the efficiency of our construction.

2 Transmission Delays as a Source of Noise

Digital communications are almost always affected by delays in data transmission, a fundamental characteristic of wireless communication, but also a common problem in wired IP networks [14]. Reducing or limiting delays has always been one of the main challenges in the communication field. Delays are quite often difficult to predict and almost impossible to eliminate. Moreover, in real and non-isolated systems, they usually depend on external, uncontrollable factors. But what can be a daunting property in the field of communication, can turn

out to be extremely useful in cryptography, where noisy channels have long been studied in order to achieve secure computation.

However, despite having these appealing characteristics, delays have not been systematically used as a source of noise in noise-demanding security applications. In particular, specific studies in the field of secure two-party computation against computationally unbounded adversaries are still missing. In this paper we address this by proving that oblivious transfer can be achieved on a channel whose only source of noise is transmission delays.

In order to obtain results as general and widely applicable as possible, we need a channel model that makes no unnecessary assumptions on the delay. At the same time, to be able to make meaningful comparisons, we want the channel to maintain the common features of the other channels currently used for oblivious transfer protocols in the information-theoretic setting – most notably the binary symmetric channel and its modifications. In information theory literature, there is an abundance of channel definitions that model most, if not all, forms of delay. However, those channels are designed around specific communication scenarios and for purposes different from those of cryptography. Therefore, we define a new noisy channel that is based on a small set of assumptions and is simple enough to allow for clear constructions and proofs, the Binary Discrete-time Delaying Channel (BDDC). In section 2.2 we show how the BDDC succeeds in modeling real-life communication scenarios.

2.1 Binary Discrete-time Delaying Channel (BDDC)

Our model of communication channel is a box accepting binary strings and emitting each accepted 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 instants 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.

Definition 1. *A* Binary Discrete-time Delaying Channel *with delaying probability p consists of*

- *an input alphabet* $\{0,1\}^n$,
- *an output alphabet* $\{0,1\}^n$,
- *a set of consecutive input times* $T = \{t_0, t_1, \ldots\} \subseteq \mathbb{N}$,
- *a set of consecutive output times* $U = \{u_0, u_1, \ldots\} \subseteq \mathbb{N}$ *where* $\forall u_i \in U, t_i \in T, u_i \geq t_i$.

Each input admitted into the channel at input time $t_i \in T$ is output once by the channel, with probability of being output at time $u_j \in U$

$$\Pr[u_j] = p^{(j-i)} - p^{(j-i+1)} \ . \tag{1}$$

Example 1. The probability of a string x, admitted into the channel 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 strings, 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. Neither the sender nor the receiver gets any feedback about the transmission, i.e. they do not learn any information about whether or not a string sent or received was actually delayed.

Informally put, the channel models a non-instantaneous communication between two parties, where the transmission takes a standard time $(t_i - u_i)$. Some of the content transmitted suffers unpredictable delays, which are usually short, but can sometimes take much longer.

Remark 1. 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 t_i on the time-line, 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 $j > i$. The channel also makes no claim whether or not the distance between each input (or output) time has to be fixed, but for clarity's sake we assume that to be the case.

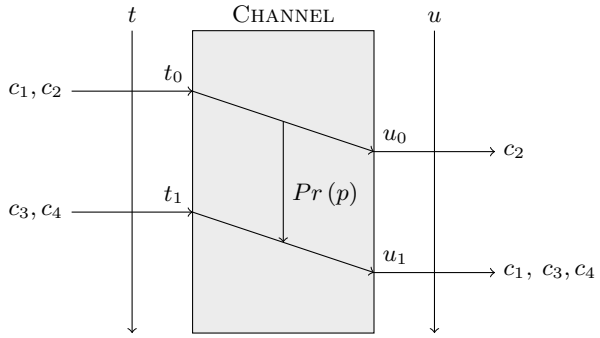

Fig. 1. A schematization representing a Binary Discrete-time Delaying Channel accepting two strings at time t_0, one of which gets delayed once, and two at time t_1, none of which gets delayed. This results in the channel emitting one string at time u_0 and three at u_1.

2.2 Real-World Communication Scenarios

While delays generally occur in most forms of telecommunication, a digital networking communication method that is particularly sensitive to them is packet switching. Packets moving through a shared network are usually delivered to destination passing by a variable number of nodes, routers and switches. At each hop a packet may be buffered and queued, building up a variable delay

depending on the traffic load of the network. For a deliberate design choice, the Internet Protocol (IP) does not guarantee that packets are delivered in the same order in which they were originally sent. The behavior of a network resulting in out-of-order delivery of packets is known as *packet reordering*. A 2004 study by Zhou and Van Mieghem found that, tracing sets composed of 50 100-byte UDP packets between 12 Internet test boxes, around 56% of the streams were subject to packet reordering [18], while Bellardo and Savage found in [1] that miniumsized TCP packets are reordered more than 10 percent of the time. Mesurements techniques are available to assess the impact of this phenomenon [1], and the analysis of the reordering caused by *multipath forwarding* (the choice of different routes for packets in the same stream) indicates that the current trend of increase in parallelism necessary to handle high speed links is also increasing the occurrence of packet reordering [12].

The binary discrete-time delaying channel is well suited to simulate the behavior of an IP network affected by packet reordering. The model approximates reality by introducing the requirement of discrete times for inputs and outputs, which allows for a remarkably more flexible and easier to study noisy channel. Generally, any packet switching network where a packet has some probability of being delayed during the transmissions can be modeled using a BDDC.

3 Building Oblivious Transfer over a BDDC

In the original concept of oblivious transfer, as presented by Rabin [15], the sender, Sam, sends his secret bit b to the receiver, Rachel. Rachel receives the bit with probability $\frac{1}{2}$ and, whether or not she receives it, she will not tell Sam. A variant of the primitive, named *chosen one-out-of-two oblivious transfer*, or simply 1-2 oblivious transfer, was later presented by Even, Goldreich and Lempel [9]. In this case Sam has two secrets bits, b_0 and b_1, and wants to communicate one of them to Rachel, without at the same time revealing the other. Rachel wants to choose which one to receive without letting Sam know her selection s, but should not be able to learn any information other than the secret bit b_s she has selected. The two versions of the primitive were shown to be equivalent by Crépeau [3]. We choose to focus on 1-2 oblivious transfer, and in the following, for simplicity, we refer to it simply as oblivious transfer.

3.1 A Security Definition for Oblivious Transfer

A protocol implements OT in a secure manner if three conditions are satisfied after a successful execution: Rachel learns the value of b_s (correctness); Rachel gains no further information about the value of b_{1-s} (security for Sam); Sam learns nothing about the value of s (security for Rachel) [5]. We give a formal definition of these security conditions by using the concept of *prediction advantage*. The prediction advantage is a measure of the advantage an adversary has in guessing a secret bit when using all the information available to her. We use the notation found in [16].

Definition 2. ([16]) *Let P_{XY} be a distribution over $\{0,1\} \times \mathcal{Y}$. The* maximal bit prediction advantage *of X from Y for a function f is*

$$\mathrm{PredAdv}\left(X \mid Y\right) = 2 \cdot \max_{f} \Pr\left[f\left(Y\right) = X\right] - 1 \ . \tag{2}$$

We call *view* of a player all the information that the player obtains during an execution of the protocol. For each execution there are both a receiver's view and a sender's view. In the *semi-honest model*, the adversary is *passive*: she follows the protocol, but outputs her entire view [16].

When proving the security of our construction, we use the following definition of oblivious transfer.

Definition 3. *A protocol Π between a sender and a receiver, where the sender inputs $(b_0, b_1) \in \{0,1\}$ and outputs nothing, and the receiver inputs $s \in \{0,1\}$ and outputs S, securely computes 1-2 oblivious transfer with an error of at most ε, assuming that U and V represent the sender and receiver views respectively, if the following conditions are satisfied:*

- (Correctness) *If both players are honest, we have*

$$Pr\left[S = b_s\right] \geq 1 - \varepsilon \ . \tag{3}$$

- (Security for Sam) *For an honest sender and an honest (but curious) receiver we have*

$$\mathrm{PredAdv}\left(b_{1-s} \mid V, s\right) \leq \varepsilon \ . \tag{4}$$

- (Security for Rachel) *For an honest receiver and an honest (but curious) sender we have*

$$\mathrm{PredAdv}\left(s \mid U, b_0, b_1\right) \leq \varepsilon \ . \tag{5}$$

3.2 A Protocol for Oblivious Transfer over a BDDC

The protocol we introduce allows the construction of oblivious transfer over a BDDC. The protocol is composed of a first phase, during which the sender Sam transmits through the channel multiple times and the receiver Rachel listen, and a second phase, where communication happens on a clear channel and the parties exploit the noise introduced by the channel to achieve their goals of secrecy and security. Before any communication can actually begin, some introductory computation by the sending party is needed, in order to craft the strings that will be sent later on to the receiver through the channel.

This construction follows the basic concepts introduced by Crépeau and Kilian while describing for the first time how to build OT over the BSC [5].

Protocol 1. Before starting any communication, some preparatory computation needs to be completed. Sam selects two disjoints sets E and E' of n distinct binary strings of length l: $e_1, \ldots, e_n$ and $e'_1, \ldots, e'_n$.

Then, Sam builds the following sets:

– C, that contains the strings $c_1, \ldots, c_n$ defined as the concatenation $c_i := e_i \| i$;
– C', that contains the strings $c'_1, \ldots, c'_n$ defined as $c'_i := e'_i \| i$.

We call the i's *sequence numbers*, while the strings in $E \cup E'$ are used as *string identifiers*. The values n and l are shared between the parties. The players can communicate either using a binary discrete-time delaying channel with probability p, called p-BDDC, or a clear channel.

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

1. Sam sends C to Rachel using the p-BDDC at instant t_0.
2. Sam sends the set C' to Rachel using the p-BDDC at instant t_1.
3. At instant u_0 Rachel receives over the p-BDDC all the strings in C that have not been delayed by the channel. If less than $\frac{n}{2}$ strings are received Rachel instructs Sam to abort the communication.
4. At instant u_1 Rachel receives over the $p-$BDDC the strings from 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 selects a set of string identifiers I_s, where $s \in \{0, 1\}$ is her selection bit, such that $|I_s| = \frac{n}{2}$ and so that every string $c \in C$ with $i \in I_s$ has been received for the first time at u_0. Then she puts the remaining i's in I_{1-s} and sends I_0 and I_1 to Sam over a clear channel.[1]
6. Sam receives I_0 and I_1, and chooses two universal hash functions f and f', whose output is 1-bit long for any input. Let $E_j \subset E$ be the set containing every $e_i \in E$ corresponding to an $i \in I_j$, such that

$$e_i \in E_j \Leftrightarrow i \in I_j \ . \tag{6}$$

   For each set I_j, Sam computes the string g_j by concatenating each $e^j_k \in E_j$, ordering them for increasing binary value, so that

$$g_j = \left(e^j_1 \parallel \ldots \parallel e^j_{\frac{n}{2}}\right) \quad \text{with } e^j_1, \ldots, e^j_{\frac{n}{2}} \in E_j \ . \tag{7}$$

   The two strings g_0, g_1 are given in input to the hash functions f, f' to obtain the two values

$$h_0 = f\left(g_0\right) \ , \qquad h_1 = f'\left(g_1\right) \ . \tag{8}$$

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

$$i_0 = \left(h_0 \oplus b_0\right) \ , \qquad i_1 = \left(h_1 \oplus b_1\right) \ . \tag{9}$$

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

$$b_s = f^s\left(g_s\right) \oplus i_s \ . \tag{10}$$

[1] Or Rachel can just send one of these two sets in order to save bandwidth as Sam can easily reconstruct the other.

Remark 2. 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.

Remark 3. Since the elements in $E \cup E'$ have to be distinct, we gather that

$$2^l \geq |E \cup E'| = 2n \ . \tag{11}$$

Remark 4. While in our constructions we use the sequence numbers i's, it should be noted that any set D of n distinct binary strings $d_1, \ldots, d_n$ might be used in their place in a setting where using unordered strings may be preferred.

4 Security in the Semi-honest Scenario

In the semi-honest setting, 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 wants to guess which secret Rachel selected, while Rachel's aim is to get as much information as possible on the other secret.

Theorem 1. *The protocol described in Section 3.2, securely computes 1-2 oblivious transfer with error probability ε when it is executed on a p-BDDC with $0 < p < \frac{1}{2}$ and*

$$n > \max\left(\frac{-2\log(\varepsilon)}{(1-2p)^2}, \frac{\log\left(\frac{\varepsilon}{2}\right)}{\log\left(1-\frac{p}{2}\right)}\right) \ . \tag{12}$$

Proof. We prove the security of our construction by showing that each of the three conditions of Definition 3 hold.

Correctness Rachel is able to compute the bit b_s when she receives, at step 3 of the protocol, a number of non-delayed strings that is greater than $\frac{n}{2}$. If we use X to denote the random variable counting this number, we see that $\Pr\left[X < \frac{n}{2}\right]$, that is, the probability that too many strings are delayed for the protocol to succeed, follows the cumulative distribution function of the binomial distribution. Using Hoeffding's inequality, we then observe that

$$\Pr\left[X < \frac{n}{2}\right] \leq \exp\left(-2n\left(\frac{1}{2}-p\right)^2\right) \ , \tag{13}$$

which shows that the correctness condition is satisfied by our protocol with overwhelming probability in n when $p < \frac{1}{2}$.[2] By extracting n in this inequality, we obtain the first argument of the maximum function in the theorem statement.

[2] Note that, for channels where $\frac{1}{2} \leq p < 1$, the correctness condition on p can be relaxed by requiring Rachel to build sets containing less than half of the strings, which would allow the protocol to succeed even if more than half of the strings are delayed.

Security for Sam We evaluate the probability that Rachel is able to compute both b_s and b_{1-s} in a protocol session. In the semi-honest setting, which we consider here, this probability is upper-bounded by the probability that Rachel is able to compute b_{1-s}. Let us call this event $\mathsf{Success}$.

Rachel has two ways to compute b_{1-s}: by evaluating the appropriate universal hash function on the correct inputs, as Sam does in Step 6 of the protocol (let us call $\mathsf{GuessInputHash}$ this event), or by not doing so. So, $\Pr[\mathsf{Success}] = \Pr[\mathsf{Success} \wedge \mathsf{GuessInputHash}] + \Pr[\mathsf{Success} \wedge \neg\mathsf{GuessInputHash}]$. The probability of the second alternative is upper-bounded by $\frac{1}{2}$, due to the properties of the universal hash function. The probability of the first alternative is in turn upper-bounded by $\Pr[\mathsf{GuessInputHash}]$. Let us now evaluate that probability.

For each pair of strings sharing the same sequence number i, four events can happen:

1. The first string of the pair is not delayed, which happens with probability $1 - p$.
2. The first string of the pair is delayed, but the two strings still reach Rachel in the same order they were sent. This happens with probability $\frac{p^2}{1+p}$.
3. Those two strings are delivered to Rachel in reverse order, which also happens with probability $\frac{p^2}{1+p}$.
4. The two strings are delivered to Rachel at the same time, which happens with probability $\frac{p(1-p)}{1+p}$.

When the first string is not delayed, Rachel can be sure of which was sent first. When the first string is delayed, and the two strings are delivered at different times, Rachel cannot guess with a probability better than $\frac{1}{2}$ whether the two strings are switched or delivered in the sending order: both events happen with the same probability. This is obviously also true when the two strings are delivered at the same time.

So, as soon as the first of the two strings is delayed, no strategy can provide a probability higher than $\frac{1}{2}$ to guess which string was sent first, meaning that Rachel is able to guess with probability $1 - \frac{p}{2}$ which one among any two strings with identical sequence number was sent first. Let us denote by $\mathsf{GuessCorrectOrder}$ the number of such correct guesses among n pairs of strings.

We have that $\Pr[\mathsf{GuessInputHash}] = \Pr[\mathsf{GuessInputHash} \wedge \mathsf{GuessCorrectOrder} = n] + \Pr[\mathsf{GuessInputHash} \wedge \mathsf{GuessCorrectOrder} < n]$. Let us now observe that the first term of this sum is upper bounded by:

$$\Pr\left[\mathsf{GuessCorrectOrder} = n\right] = \left(1 - \frac{p}{2}\right)^n \quad , \tag{14}$$

which is negligible in n as soon as $p > 0$. Besides, since the input of the hash function is not correctly guessed when $\mathsf{GuessCorrectOrder} < n$, we have that the second term of the sum is null. This shows that:

$$\Pr\left[\mathsf{Success}\right] \leq \frac{1}{2} + \left(1 - \frac{p}{2}\right)^n \quad . \tag{15}$$

By using the definition of prediction advantage and extracting n in this inequality, we obtain the second argument of the maximum function in the theorem statement.

Security for Rachel The only step in the protocol in which Rachel uses her selection s to generate messages to Sam is number 5, when she sends back I_0 and I_1. During any other step Rachel is not sending any information at all to Sam. A BDDC gives no feedback to the sender or the receiver about which strings are delayed: each string c is delayed at least once with probability p independent of c. Therefore, from Sam's point of view, the distribution (I_0, I_1) is independent of s, and Sam's prediction advantage on s given his view and his input bits is null. □

Remark 5. We observe that the semi-honest assumption is only required for the sender, but not for the receiver. When acting as a malicious receiver, Rachel can either produce a malformed set I_{1-s} (reducing the number of strings included, or including non-delayed strings already present into I_s) in order to put only non-delayed strings into the set, or swap delayed strings with non-delayed ones between the sets I_s and I_{1-s}. In the first case, a simple additional check on the sender's side of the protocol will prevent any response to a malformed I_{1-s}. In the second case Rachel, by moving delayed strings from I_{1-s} to I_s, increases her probability to get the other bit b_{1-s} at the cost of lowering her probability to get the selected bit b_s. In fact, the number of delayed strings, which is also the total number of guesses needed by Rachel, remains the same. Therefore the probability of decoding both bits is the same whether she acts honestly or in a malicious way.

5 Conclusion

In this paper, we proposed using channel delays as a source of uncertainty to realize oblivious transfer. To this purpose, we introduced a new channel, called Binary Discrete-time Delaying Channel (BDDC), and propose an OT protocol built on this channel.

We believe that building OT on communication delays provides important benefits compared to the existing solutions. In particular, our protocol has a remarkably low sensitivity to the precise knowledge of the channel parameters, a factor that often constitutes one of the main inconveniences of cryptographic protocols relying on communication channel properties.

Figure 2 illustrates this little sensitivity by plotting the two curves of which the maximum is taken in the statement of Theorem 1, for a security parameter $\varepsilon = 10^{-9}$. The curve that grows when p tends to 0 shows that the number of strings to be sent must increase when p is small in order to ensure that Rachel is not able to decode both of the sender bits of the OT protocol. The curve that grows with p shows that the number of strings to be sent must increase when p tends to $\frac{1}{2}$ in order to ensure that Rachel gets one of the two sender bits.

This graph shows that our protocol is able to tolerate a very wide range of uncertainty on the channel parameters: the exchange of 1000 strings (that is, approximately 42000 bits transferred on the BDDC channel and less than 12000 bits sent on the noiseless channel) guarantees oblivious transfer with error $\varepsilon = 10^{-9}$ for values of p ranging from 0.05 to 0.4 approximately. This practically means that an active adversary able to set the probability to a desired level within this range does not reduce the security of the construction. The idea of letting the adversary choose the channel probability was first introduced with the Unfair Noisy Channel (UNC) [8], a binary symmetric channel where the error rate is only known to be in a certain interval $[\gamma \cdots \delta]$. This work shows that OT cannot be achieved as soon as the difference $\delta - \gamma$ becomes too large, namely, if $\delta \geq 2\gamma(1-\gamma)$. This interval has a maximum width equal to 0.125 when $\gamma = 0.25$ and $\delta = 0.375$, even though no protocol is known that can tolerate such a wide interval width on a UNC. Interval widths for which OT can be achieved on UNCs have also been studied [7], showing experimentally that OT can be built on a UNC for intervals of maximum width around 0.04.

We believe that these figures show a crucial benefit of exploiting delays on channels: delays provide the uncertainty that is needed to build security, but they also offer the possibility to be sure that some strings have been sent before others (if they are received before other strings are sent, for instance). This is not the case on other channels that have been considered until now, like the BSC channel and its variants, where one can never be sure that a string is delivered correctly, raising the need to precisely calibrate error correction mechanisms.

Our protocol also appears to be very efficient for important delay probability ranges: we observe that sending 250 pairs of strings on the p-BDDC channel, that is, around 8500 bits, is enough to realize OT for $0.17 \leq p \leq 0.29$ and $\varepsilon = 10^{-9}$. We eventually observe that, in many practical applications, the protocol parameters might be adapted in order to influence the delaying probability if needed. For instance, it appears that packet size has an important impact on reordering occurrences in IP networks [1].

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. Bellardo, J., Savage, S.: Measuring packet reordering. In: Internet Measurement Workshop. pp. 97–105. ACM (2002)
2. Chaum, D., Damgård, I., van de Graaf, J.: Multiparty computations ensuring privacy of each party's input and correctness of the result. In: Pomerance [13], pp. 87–119

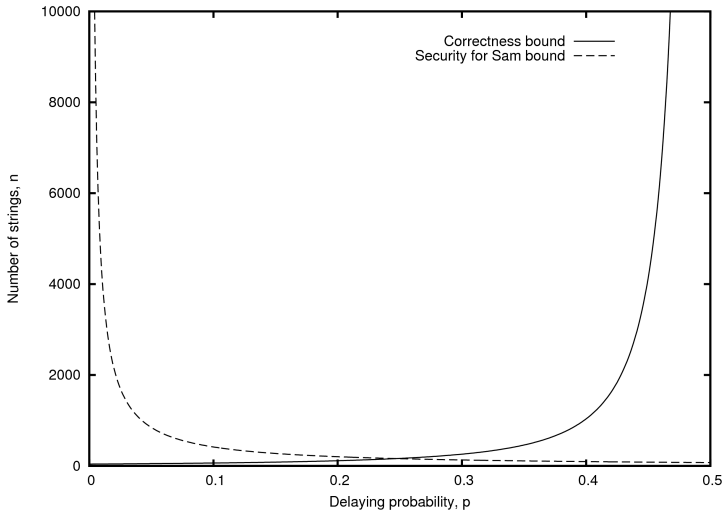

Fig. 2. n as a function of p for $\varepsilon = 10^{-9}$.

3. Crépeau, C.: Equivalence between two flavours of oblivious transfers. In: Pomerance [13], pp. 350–354
4. Crépeau, C.: Efficient cryptographic protocols based on noisy channels. In: EUROCRYPT. pp. 306–317 (1997)
5. Crépeau, C., Kilian, J.: Achieving oblivious transfer using weakened security assumptions (extended abstract). In: FOCS. pp. 42–52. IEEE (1988)
6. Crépeau, C., Morozov, K., Wolf, S.: Efficient unconditional oblivious transfer from almost any noisy channel. In: Blundo, C., Cimato, S. (eds.) SCN. Lecture Notes in Computer Science, vol. 3352, pp. 47–59. Springer (2004)
7. Damgård, I., Fehr, S., Morozov, K., Salvail, L.: Unfair noisy channels and oblivious transfer. In: Naor, M. (ed.) TCC. Lecture Notes in Computer Science, vol. 2951, pp. 355–373. Springer (2004)
8. Damgård, I., Kilian, J., Salvail, L.: On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In: EUROCRYPT. pp. 56–73 (1999)
9. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28(6), 637–647 (1985)
10. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC. pp. 20–31. ACM (1988)
11. Nascimento, A.C.A., Winter, A.: On the oblivious-transfer capacity of noisy resources. IEEE Transactions on Information Theory 54(6), 2572–2581 (2008)
12. Piratla, N.M., Jayasumana, A.P.: Reordering of packets due to multipath forwarding - an analysis. In: Proc. IEEE Int. Conf. on Communications (ICC 2006). pp. 28–36 (2006)
13. Pomerance, C. (ed.): Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, Lecture Notes in Computer Science, vol. 293. Springer (1988)

14. Proakis, J.G.: Digital Communications (4th edition). McGraw-Hill Science Engineering (August 2000)
15. Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University (1981), manuscript
16. Wullschleger, J.: Oblivious-transfer amplification. In: Naor, M. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4515, pp. 555–572. Springer (2007)
17. Wullschleger, J.: Oblivious transfer from weak noisy channels. In: Reingold, O. (ed.) TCC. Lecture Notes in Computer Science, vol. 5444, pp. 332–349. Springer (2009)
18. Zhou, X., Mieghem, P.V.: Reordering of ip packets in internet. In: Barakat, C., Pratt, I. (eds.) PAM. Lecture Notes in Computer Science, vol. 3015, pp. 237–246. Springer (2004)