CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Unconditionally secure oblivious transfer from real network behavior
Authors	Palmieri, Paolo;Pereira, Olivier
Publication date	2013-11
Original Citation	Palmieri, P. and Pereira, O. (2013) 'Unconditionally Secure Oblivious Transfer from Real Network Behavior', in Sakiyama, K. & Terada, M. (eds.) Advances in Information and Computer Security: 8th International Workshop on Security, IWSEC 2013, Okinawa, Japan, November 18-20, 2013, Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 168-182. doi:10.1007/978-3-642-41383-4_11
Type of publication	Conference item
Link to publisher's version	https://link.springer.com/chapter/10.1007/978-3-642-41383-4_11 - 10.1007/978-3-642-41383-4_11
Rights	© Springer-Verlag Berlin Heidelberg 2013. The final publication is available at Springer via https://doi.org/10.1007/978-3-642-41383-4_11
Download date	2026-11-15 02:13:01
Item downloaded from	https://hdl.handle.net/10468/4769



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Unconditionally Secure Oblivious Transfer from Real Network Behavior

Paolo Palmieri[1,⋆] and Olivier Pereira[2]

[1] Delft University of Technology, Parallel and Distributed Systems Group
Mekelweg 4, 2628 CD Delft, The Netherlands
p.palmieri@tudelft.nl
[2] Université catholique de Louvain, UCL Crypto Group
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium
olivier.pereira@uclouvain.be

Abstract. Secure multi-party computation (MPC) deals with the problem of shared computation between parties that do not trust each other: they are interested in performing a joint task, but they also want to keep their respective inputs private. In a world where an ever-increasing amount of computation is outsourced, for example to the cloud, MPC is a subject of crucial importance. However, unconditionally secure MPC protocols have never found practical application: the lack of realistic noisy channel models, that are required to achieve security against computationally unbounded adversaries, prevents implementation over real-world, standard communication protocols.

In this paper we show for the first time that the inherent noise of wireless communication can be used to build multi-party protocols that are secure in the information-theoretic setting. In order to do so, we propose a new noisy channel, the Delaying-Erasing Channel (DEC), that models network communication in both wired and wireless contexts. This channel integrates erasures and delays as sources of noise, and models reordered, lost and corrupt packets. We provide a protocol that uses the properties of the DEC to achieve Oblivious Transfer (OT), a fundamental primitive in cryptography that implies any secure computation. In order to show that the DEC reflects the behavior of wireless communication, we run an experiment over a 802.11n wireless link, and gather extensive experimental evidence supporting our claim. We also analyze the collected data in order to estimate the level of security that such a network can provide in our model. We show the flexibility of our construction by choosing for our implementation of OT a standard communication protocol, the Real-time Transport Protocol (RTP). Since the RTP is used in a number of multimedia streaming and teleconference applications, we can imagine a wide variety of practical uses and application settings for our construction.

⋆ This work was accomplished while the author was at the Crypto Group of the Université catholique de Louvain.

1 Introduction

Multi-party computation protocols that are secure against computationally unbounded adversaries have seen, up until now, little or no practical use. This is mainly due to the strong assumptions that need to be satisfied for them to work. In particular, they require the availability of a noisy channel, the theoretical abstraction of an error-prone communication medium, since security can not be achieved over a clear channel. The aim of this paper is to show that, through the use of realistic channel models and efficient constructions, we can achieve secure multi-party computation over standard, commonly used network protocols today.

In a 1-out-of-2 oblivious transfer protocol, Rachel (the receiver) wants to learn one of the two secret bits b_0, b_1 that Sam (the sender) knows, but without revealing to him her selection s. Sam, on the other hand, wants to make sure that Rachel will not get any information about the other bit in the process. The first protocol to achieve this over a noisy channel was designed by Crépeau and Kilian, and used the Binary Symmetric Channel (BSC) [4]. The BSC is a simple channel model where each binary input has a probability p of being "flipped" when output: a 0 flipped becomes a 1 and vice versa. Since the BSC does not provide a realistic model of communication, new channel models have been subsequently proposed. Most of these models are modifications of the BSC itself, that introduce more freedom for the attacker in order to increase the generality of the construction. In particular, the Unfair Noisy Channel (UNC), proposed by Damgard et al. in 1999 [6] and later improved in 2004 [5], lets the adversary choose the error probability within a specific (narrow) range. The Weak Binary Symmetric Channel (WBSC), designed by Wullschleger in [22], lets a dishonest player know with a certain probability if a bit was received correctly.

While these constructions ease the assumptions needed to build OT from a theoretical point of view, they hardly make the channel models closer to any real communication channel. To address this problem, recent constructions use noisy channels that try to model common transmission errors occurring in actual networks. In particular, the use of transmission delays as source of noise has been proposed in [12], where Palmieri and Pereira provide a protocol for achieving oblivious transfer over the Binary Discrete-time Delaying Channel (BDDC). A modified version of the protocol, secure against malicious players, has later been introduced by Cheong and Miyaji [1].

The suitability of the BDDC to model packet reordering over IP networks has been shown in [13]. However, the BDDC does not take into account the possibility of packets being lost, which is a common occurrence in real communication settings. Moreover, it does not limit the number of times a packet can be delayed: however unlikely, it is possible for a packet to be delayed indefinitely. The behavior of a real packet-switching network would be instead to drop a packet after a certain time, usually called *time to live* (TTL).

1.1 Contribution

In this paper we propose a new noisy channel, the Delaying-Erasing Channel (DEC). The DEC integrates delays and erasures (lost packets) and introduces a limit to the number of possible delays. This channel, while being based on discrete times like its predecessors, addresses the lacks of the BDDC, and provides a realistic model for network communication, in both wireless and wired settings. We propose a protocol for achieving oblivious transfer over the DEC, and we study the security of the construction against both semi-honest and malicious adversaries.

The main goal of the DEC is to finally provide a realistic noisy channel model for network communication. In order to show that the DEC achieves this goal, we conduct an experiment simulating our OT protocol over a wireless network, and we collect extensive statistical evidence that supports our claims of security and flexibility for the construction. We analyze the collected data using several standard tools for entropy estimation, whose results confirm the suitability of the wireless medium to be used as a noisy channel. Our implementation of OT is based on the Real-time Transport Protocol (RTP), an application layer protocol frequently used for the streaming of multimedia content.

1.2 Outline of the Paper

In section 2 we give a security definition of oblivious transfer In section 3 we introduce the Delaying-Erasing Channel (DEC), and we provide a protocol implementing oblivious transfer over it. In section 3.2 we prove the security of the construction in the semi-honest setting, while in 3.3 we discuss the case of malicious adversaries. In section 4 we show that packets transmitted over a 802.11n wireless link show a behavior consistent with the channel definition. We analyze the experimental results and measure the entropy of the network errors in section 4.4.

2 Preliminaries

For a protocol to successfully implement oblivious transfer, three conditions must be satisfied after an execution: the receiver, Rachel, learns the value of the chosen bit b_s (correctness); the sender, Sam, learns nothing about the value of the selection bit s (security for Rachel); the receiver learns no further information about the value of the other bit b_{1-s} (security for Sam) [4]. When proving the security of our construction, we use the security definition of oblivious transfer provided in [12]. The definition uses the concept of *prediction advantage*, a measure of the advantage that an adversary has in guessing a secret bit by using all the information available to him. We use the notation found in [21].

Definition 1. ([21]) *Let P_{XY} be a distribution over $\{0,1\} \times \mathcal{Y}$. The* maximal bit prediction advantage *of X from Y is*

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

The *view* of a player consists of all the information that the player learns during the protocol execution. The sender, the receiver and the potential adversary all have different views. The security definition for OT follows.

Definition 2. [12] *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*

$$Pr\left[S = b_s\right] \geq 1 - \varepsilon \ . \tag{2}$$

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

$$\mathrm{PredAdv}\left(b_{1-s} \mid V, s\right) \leq \varepsilon \ . \tag{3}$$

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

$$\mathrm{PredAdv}\left(s \mid U, b_0, b_1\right) \leq \varepsilon \ . \tag{4}$$

3 Delaying-Erasing Channel

The channel model we propose combines the erasure and delaying channels. It takes into account the possibility for an input string to be delayed or to be lost (that is, erased). The channel also sets a limit to the number of delays that an input can suffer, and considers lost (erased) any string delayed a number of times equal or higher than that. We call p the delaying probability, r the maximum number of delays and q the erasing probability. Consequently, the probability that a string will be considered lost by a receiver is $(q + p^r)$, consisting of the erasing probability plus the probability for the string to be delayed r times.

Definition 3. *A* Delaying-Erasing Channel *(DEC) with delaying probability p, erasing probability q and maximum number of delays per single input r accepts as input a sequence $T = \langle t_1, t_2, \ldots \rangle$ of sets of strings $t_i \in (\{0,1\}^n)^*$, called input times, and outputs a sequence $U = \langle u_1, u_2, \ldots \rangle$ of sets of strings $u_i \in (\{0,1\}^n)^*$ called output times. Each string X admitted into the channel at input time $t_i \in T$ is output at most once by the channel, with probability of being output at time $u_j \in U$*

$$\Pr\left[X \in u_j | X \in t_i\right] = \begin{cases} (1-q) \cdot p^{j-i} \cdot (1-p) \\ \qquad\qquad \textit{with } 0 \leq j - i < r, \\ 0 \qquad\qquad\qquad\quad \textit{otherwise.} \end{cases} \tag{5}$$

In practice, the channel works as follows. An input string X that enters the channel at time t_i is due to be output by the channel at time u_i with probability $(1-p-q)$. The channel has probability q of erasing the string (we call this event *impromptu loss*), in which case the string is not output by the channel. If an impromptu loss does not occur, the string has a probability p of being delayed until the next output time. The delay event can happen multiple times: once a string is delayed, it can be delayed again. Therefore, the string has probability p^d of being delayed d times and being output at u_{i+d}, as long as $d < r$. Finally, the string is not output by the channel if it is delayed r times or more, which happens with probability p^r.

3.1 OT Protocol

Our oblivious transfer protocol follows the general scheme proposed by Crépeau and Kilian in [4], which has been at the base of every following OT construction. The idea behind the scheme is to generate, through a specific transmission strategy, a simple erasure channel over the available noisy channel (in our case the DEC), and then use it a number of times in order to realize OT and achieve security by privacy amplification. The protocol we propose is also similar to the one proposed in [12] for the Binary Discrete-time Delaying Channel, in the sense that it uses a precomputation phase during which two sets of packets are created. Contrary to the case of the BDDC, our protocol streams the two sets of packets by interleaving them: we send the first packet of the first set at t_1, then the second packet of the first set and the first of the second set at t_2 and so on. This allows us to exploit the uncertainty caused by the lost and delayed packets.

The protocol works as follows. First the sender, Sam, precomputes a sequence of packets. For simplicity, we can assume that the packets only contain their sequence number i. Then, he starts sending the packets over the DEC to the receiver, Rachel. Each packet is sent twice: the first packet at times t_1 and t_2, the second one at t_2 and t_3 and so on. Each of the two times the same packet is sent, Sam also attaches to it a unique identifier (e_i for the first transmission and e'_i for the second one), so that he will be able to tell them apart. However, he will not reveal to Rachel which identifier is used for which transmission of the packet. Rachel keeps track of the packets lost on the way and of the arrival times of those she receives. However, the channel does not give her any feedback on the delays that occur during transmission. Therefore, in the case of a packet received for the first time later than the expected time u_i, she is not able to tell which of the two copies of the packet was sent with the first transmission, and which with the second. The same is true in case only one copy arrives and it is received after the expected time u_i, or in case both copies are lost. At the same time, Sam does not know the arrival time and order of the packets. We use this uncertainty to build oblivious transfer. Rachel assigns to her selection bit s the packets for which she knows with certainty the identifier e, and to the other bit $(1-s)$ the other packets. Then, she sends her two selections of packets to Sam. Sam encodes the secret bits b_0 and b_1 using the identifiers e attached to

the packets during the first transmission, according to the selection operated by Rachel. Using the same identifiers, Rachel is able to decode b_s, but not b_{1-s}.

Protocol 1. The parties have a clear channel and a p-q-r-DEC with $0 < (p+q) < \frac{1}{2}$, $r > 1$ available for communication. Sam selects two disjoint sets E and E', each composed of n distinct binary strings of length l: $e_1, \ldots, e_n \in E$ and $e'_1, \ldots, e'_n \in E'$. From E and E' Sam builds the sets $C = \{c_1, \ldots, c_n\}$ and $C' = \{c'_1, \ldots, c'_n\}$, according to the following rules: $c_i := e_i \| i$ and $c'_i := e'_i \| i$. Then the parties communicate as follows:

1. Sam sends the set C to Rachel over the DEC, one string at each input time, starting at t_1. At t_2 he starts sending C' as well. This way, at each t_i, c_i and c'_{i-1} are sent.
2. Rachel receives over the DEC the strings in $\{C \cup C'\}$ that have not been erased, in the order produced by the channel.
3. Rachel selects the set I_s, where $s \in \{0, 1\}$ is her selection bit, such that $|I_s| = \frac{n}{2}$ and so that $i \in I_s$ only if she is able to distinguish $c_i \in C$ from $c'_i \in C'$. This happens in two cases: c_i has been received at u_i; or c_i, c'_i have not been erased and c'_i has been received at u_{i+r}. If less than $\frac{n}{2}$ strings can be placed in I_s, Rachel instructs Sam to abort the communication. Otherwise she selects $I_{1-s} = \{1, \ldots, n\} \setminus I_s$ and sends I_0 and I_1 to Sam over the clear channel. [3]
4. Sam receives the sets I_0 and I_1. Then, he chooses two universal hash functions f^0, f^1, 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

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

   Sam computes $h_0 = f^0(g_0)$, $h_1 = f^1(g_1)$ and sends to Rachel over the clear channel the functions f^0, f^1 and the two values

$$k_0 = (h_0 \oplus b_0) \ , \qquad k_1 = (h_1 \oplus b_1) \ . \tag{8}$$

5. Rachel computes her guess for b_s

$$b_s = f^s(g_s) \oplus k_s \ . \tag{9}$$

[3] In order to improve the efficiency of the protocol in a real setting, the receiver can send just one of these two sets, for example always I_0, as the sender can easily reconstruct the other.

3.2 Security: Honest-but-curious Adversaries

In the *semi-honest* setting, the players follow the protocol, but try to use any information available to them in order to guess the other player's secret. We prove the security of our construction by proving each of the three conditions of the security definition of oblivious transfer (Definition 2).

Correctness The first condition of Definition 2 states that, if both players behave in an honest way, the secret bit must be correctly received and decoded by the receiver party. In practice, the protocol succeeds when Rachel is able to identify with certainty at least $\frac{n}{2}$ strings from C among all the strings she receives. As stated in step 3 of the protocol, a string c_i is known by Rachel to be $\in C$ with certainty either when it is received at u_i; or when c_i is not erased and the corresponding string $c'_i \in C'$ is received at u_{i+r}. Therefore, the probability that a string c_i will not be identifiable as being part of C is upper-bounded by the probability $(p+q)$ that c_i is erased, or delayed at least once. Let us denote by X the random variable counting the number of strings not affected by the noise (that is, erased or delayed) out of the n total strings in C. We have that $\Pr\left[X \leq \frac{n}{2}\right]$, the probability that not enough strings in C are received correctly and on time for the protocol to succeed, follows the cumulative distribution function of the binomial distribution. For Hoeffding's inequality we have that

$$\Pr\left[X \leq \frac{n}{2}\right] \leq \exp\left(-2n\left(p+q-\frac{1}{2}\right)^2\right) \quad . \tag{10}$$

Therefore, the correctness condition is satisfied with overwhelming probability in n as soon as $p+q < \frac{1}{2}$, as per the protocol definition.

Security for Sam A curious Rachel is interested in learning b_{1-s}. She has two ways of obtaining the value: either by decoding k_{1-s} on the correct g_{1-s}, or by trying to guess it on a (partially) incorrect g_{1-s}. In the latter case, the probability of a correct guess is upper-bounded by $\frac{1}{2}$, for the properties of a universal hash function. In the following we evaluate the probability of the former.

For each pair of strings $(c_i \in C, c'_i \in C')$, Rachel receives two or less strings, in the order produced by the channel. She is interested in determining c_i, in order to learn e_i. We analyze in the following her ability of doing so, based on the different events that can happen after the transmission of the strings through the delaying-erasing channel. We suppose that, in case only one string is received, Rachel assumes to have received c_i.[4] For each (c_i, c'_i) we can have that:

- c_i is neither erased nor delayed. Independently of what happens to c'_i, Rachel learns e_i. This happens with probability $(1-p-q)$.

[4] This is always the best strategy, since a wrong assumption does not lower her probability of learning e_i: we assume that guessing e_i with no information has a negligible probability of succeeding.

– c_i is erased. Independently of what happens to c'_i, Rachel is not able to recover the identifier e_i. This happens with probability q.
– c_i is delayed. This happens with probability p. In this case, Rachel's probability to learn e_i depends on c'_i. We can have that:
  • c'_i is erased. Following the strategy of using the identifier she possesses, Rachel succeeds in guessing the right identifier. This happens with probability $p \cdot q$.
  • c'_i is not erased. This happens with probability $p(1-q)$. If c'_i is delayed $r-1$ times, Rachel learns the right identifier. This happens with probability $p^r(1-q)$. Otherwise, Rachel guesses the right identifier with probability $\frac{1}{2}$. In fact, the probability for the strings to arrive in the same order in which they are sent is equal to the probability for them to arrive in the reverse order ($\frac{p^2}{1+p}$). Therefore she does not have any strategy better than tossing a coin in both cases, as well as when the strings arrive at the same time.

Therefore, for each pair of strings (c_i, c'_i), Rachel does not learn e_i with probability

$$\Pr[\neg e_i] = q + \frac{p(1-q) - p^r(1-q)}{2} \quad , \tag{11}$$

which is > 0 as soon as $0 < (p+q) < \frac{1}{2}$ and $r > 1$ as per the protocol definition. Therefore, Rachel's probability of building the correct g_{1-s} by learning the correct e_i for every $i \in I_{1-s}$ is

$$\Pr[g_{1-s}] = (1 - \Pr[\neg e_i])^n \quad , \tag{12}$$

which is negligible in n.

Security for Rachel Since the delaying-erasing channel does not give any feedback to the sender on the state of transmitted strings, Sam ignores whether a string has been correctly received or not, and if it has been delayed during transmission. Therefore, from the point of view of a curious sender the distribution of (I_0, I_1) is independent of s.

3.3 Security: Malicious Adversaries

We observe that the semi-honest assumption of our construction is only required for the sender, but not for the receiver. This is also the case for the oblivious transfer protocol proposed for the BDDC [12]. In fact, a malicious Rachel can either send to Sam a malformed set I_{1-s}, where she puts only indices of strings not affected by the noise (for instance, by sending less i's than required or by including i's already in I_s), or swap strings affected by the noise with non-affected ones between the sets I_s and I_{1-s}. If Rachel chooses the former strategy, Sam can detect her malicious behavior by implementing a simple additional check on I_{1-s}, and abort the protocol in case the behavior of the receiver deviates from the protocol. The latter strategy, instead, increases Rachel's probability to learn

the other bit b_{1-s}, by moving delayed or erased strings from I_{1-s} to I_s, but only at the cost of lowering her probability to learn the selected bit b_s. In fact, the number of strings that have been delayed or erased by the channel, which is also the number of guesses that Rachel needs to make, remains the same. Therefore the probability for Rachel to decode both b_s and b_{1-s} is the same whether she acts honestly or in a malicious way.

As already noted in [10], we can use an oblivious transfer protocol secure against a malicious receiver and a semi-honest sender to obtain a protocol secure against a semi-honest receiver and a malicious sender. This is possible thanks to the symmetry property of oblivious transfer, proved for the first time in [20]. A black-box combiner for this reversal operation has been proposed in [8], where a compiler that combines the two protocols into one that is secure against generic malicious adversaries, originally designed for the case of OT based on trapdoor functions, is also presented.

4 From Noisy Channel to Real Network Behavior

The aim of this section is to show that the DEC realistically models actual network behavior. In order to do so, we simulate the OT protocol over a wireless point-to-point connection between two hosts, and we study the amount of errors that occur during the transmission and the predictability of such errors. We show the flexibility of our construction by implementing our OT protocol over a standard Internet protocol, the Real-time Transport Protocol.

4.1 Real-time Transport Protocol (RTP)

The Real-time Transport Protocol (RTP) [7,15] is an application layer protocol designed for the delivery of real-time information. Its typical use is the delivery of real-time audio and video, as in the case of multimedia streaming or teleconferencing. It is often used in conjunction with the Real Time Streaming Protocol (RTSP) [16], that provides a framework for controlling the data flow. RTP typically runs on top of the User Datagram Protocol (UDP). Both protocols are particularly suited to be used in our construction: they do not guarantee reliable transmission or quality-of-service and they do not support error correction and lost packet resending. The protocol specification for RTP expressly states that it "does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence" [15].

4.2 OT over RTP

Taking advantage of the fact that RTP does not prevent packet loss or reordering, we can use it as the base for our oblivious transfer construction. In particular, the parties use RTP at step 1 of the protocol, while communicating over a wireless (or wired) link, that acts as the noisy channel. The sender sends two distinct RTP streams composed of the same number of packets. The

content of each packet can be arbitrarily selected by the sender, as long as it is unique with respect to both streams, since it is to be used as the packet identifier. Some of the fields of a standard RTP packet header (see RFC 3550 [15] for reference) require special care in our application. The `Sequence Number` value will be used, with the same meaning, also in the OT protocol. Packets sharing the same position in the two streams will be forged by the sender in order to have identical headers. In particular, this has to be enforced for the `Timestamp` field. Similarly, the identifier of the synchronization source (that is, the sender) has to be replicated in both streams. The `Payload Type` field, which indicates the encoded format of the data sent with RTP, can be chosen arbitrarily. The underlying protocols (UDP or IP) do not add information that could make the streams distinguishable, so no specific intervention is needed at levels lower than the application layer, other than selecting the desired time-to-live at the IP level.

The contemporary transmission of multiple RTP streams from the same source does not reveal the specific use we make of the protocol. In fact, it is a common occurrence: for instance, in the transmission of multimedia content, audio and video streams usually have separate RTP sessions, enabling a receiver to deselect a particular stream.

The following steps of the protocol remain unchanged, as they are performed over a clear channel.

4.3 Experiment

The aim of this experiment is double: to show that the DEC realistically models the noise introduced during network communication, and to analyze the unpredictability, and therefore the suitability for secure computation of that noise. We conduct the experiment as follows.

The party acting as sender is simulated by a wireless router running the open source and Linux-based custom firmware OpenWRT. This particular configuration let us use the RTP/RTSP streaming server Live 555 directly on the device. The receiver party, a notebook computer, connects to the router before starting the OT protocol using the IEEE 802.11n-2009 wireless transmission method [9], and receives an IP address through a DHCP request. This way, the receiver and sender parties are directly connected by a wireless link.

The notebook computer simulating the receiver party is placed at about 12 meters of distance from the router. No physical obstacles block the line of sight between the two devices. Both parties are not engaged in any network communication other than the RTP streaming. The streaming session, initiated by the sender, runs for 158.28 seconds, with a total of 5629 packets sent. The packets reaching the receiver party are collected in the order they are received using the open source sniffing tool WireShark. [5] Steps 3 to 5 of the protocol (encoding and decoding of the secret bits and communication over a clear channel) are not simulated during the experiment.

[5] The sample data transmitted, and the dump of the packets received is available at the URL: http://www.uclouvain.be/crypto/ot-wireless-tests.tar.gz.

The results of the experiment are shown in Table 1, and appear to be consistent with relevant literature (see, for instance, [14]). The number of lost packets (erasures) and sequence errors (delays) has been obtained using the RTP Stream Analysis tool provided with WireShark. In the following we analyze the results from the security point of view.

Total RTP packets:	5629	
Erasures:	65	1.15%
Delays:	109	1.94%
Total errors:	174	3.09%

Table 1. Average lost (erased) and displaced (delayed) packets during video streaming using the RTP protocol over a wireless link.

4.4 Analysis

The amount of noise that we observed during the experiment indicates that both lost packets and sequence errors are relatively common occurrences, as shown in Table 1.

The security of our construction, however, also depends on the (im)possibility, for an attacker, of being able to predict errors. In other words, we want the distribution of the displaced and lost packets into the sequence to be as uniform as possible. In order to evaluate how much this assumption reflects the reality of wireless communication, we convert the sequence of packets generated during the experiment into a binary string, using the following strategy: the packets affected by the noise are represented by a bit of value 1, those not affected by a bit of value 0. Then, we estimate the entropy of the generated binary string using a set of standard test suites, in particular: `ent` [17], Maurer's test including Coron's modification [11,3,2] and the Context-Tree Weighting (CTW) method [19,18]. The main idea behind these tools for entropy estimation is to compare the length of an input sequence with its output after compression. Since the probability of errors (and therefore of 1's) is lower than 0.5, we compare it to the Shannon entropy normalized to the actual probability, calculated using the standard definition

$$H_{\mathrm{b}}(p) = -p \log_2 p - (1-p) \log_2 (1-p) \tag{13}$$

and the amount of noise observed during our experiment. Since we fix the probability p to the observed value, $H_{\mathrm{b}}(p)$ is the maximum possible entropy, and not an upper-bound. This does not affect the reliability of the results, since our goal is to detect the presence of any pattern in the error distribution that might lead to predictability, and not to evaluate the error probability itself. In the case of packet delays, we have $p = 0.0194$, and therefore $H_{\mathrm{b}}(p) = 0.1392$. Entropy

estimations calculated by the three tests mentioned above are shown in Table 2: the closer to the maximum entropy $H_{\mathrm{b}}(p)$ the estimated values are, the less likely we are to find any pattern in the sequence.

Max. normalized entropy $H_{\mathrm{b}}(p)$: 0.1392	
Ent	0.1392
*Maurer**	0.0994
CTW	0.1144

Table 2. Entropy estimation for one bit, given $p = 0.0196$, as observed during the experiment.

While in the case of the `ent` test the entropy estimation is virtually identical to the maximum value, the context-tree weighting method is able to compress to a higher ratio. In fact, the CTW algorithm produces an output whose size is 82% of the one that would be obtained compressing an input where errors are uniformly distributed. The Maurer-Coron test is the most effective, reaching a compression ratio of 71%. However, this is partly due to a requirement in the algorithm that imposes a minimum input length higher than the size of our test string. Therefore, during the test execution, about 800 bits of the input string are read twice, since the test loops the input in case of an insufficient amount of data to elaborate. Overall, these results confirm that, even in a setting where a low amount of noise can be expected, errors are both enough frequent and randomly distributed to allow for a significant security margin to be achieved.

5 Conclusion

In this paper we propose a noisy channel model that reflects, for the first time, the behavior of real networks. We present experimental evidence collected during an experiment over wireless communication supporting this claim, and we show the flexibility of the model by running the experiment using a commonly used Internet streaming protocol, the Real-time Transport Protocol.

Analysis of the noise introduced by the wireless medium during the experiment supports the assumptions that the channel makes in terms of unpredictability of that noise. In fact, using standard entropy estimation tools, we estimate the normalized entropy to be between 71% and 100% of the theoretical maximum, depending on the test, even for a relatively clean channel where the amount of noise observed is, on average, 3.09%. This allows us to construct, for the first time, an oblivious transfer protocol secure against computationally unbounded adversaries over a real network. We believe that the flexibility of our model and construction will help open the way to widespread implementation of secure multi-party computation.

6 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.

References

1. Cheong, K.Y., Miyaji, A.: Unconditionally secure oblivious transfer based on channel delays. In: Qing, S., Susilo, W., Wang, G., Liu, D. (eds.) ICICS. Lecture Notes in Computer Science, vol. 7043, pp. 112–120. Springer (2011)
2. Coron, J.S.: On the security of random sources. In: Imai, H., Zheng, Y. (eds.) Public Key Cryptography. Lecture Notes in Computer Science, vol. 1560, pp. 29–42. Springer (1999)
3. Coron, J.S., Naccache, D.: An accurate evaluation of maurer's universal test. In: Tavares, S.E., Meijer, H. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 1556, pp. 57–71. Springer (1998)
4. Crépeau, C., Kilian, J.: Achieving oblivious transfer using weakened security assumptions (extended abstract). In: FOCS. pp. 42–52. IEEE (1988)
5. 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)
6. 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)
7. Group, A.V.T.W., Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard) (Jan 1996), `http://www.ietf.org/rfc/rfc1889.txt`, obsoleted by RFC 3550
8. Haitner, I.: Semi-honest to malicious oblivious transfer - the black-box way. In: Canetti, R. (ed.) TCC. Lecture Notes in Computer Science, vol. 4948, pp. 412–426. Springer (2008)
9. IEEE-SA: Ieee 802.11n-2009 amendment 5: Enhancements for higher throughput. (October 2009)
10. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Kleinberg, J.M. (ed.) STOC. pp. 99–108. ACM (2006)
11. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 1st edn. (1996)
12. Palmieri, P., Pereira, O.: Building oblivious transfer on channel delays. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt. Lecture Notes in Computer Science, vol. 6584, pp. 125–138. Springer (2010)
13. Palmieri, P., Pereira, O.: Implementing information-theoretically secure oblivious transfer from packet reordering. In: Kim, H. (ed.) ICISC. Lecture Notes in Computer Science, vol. 7259, pp. 332–345. Springer (2011)
14. Salyers, D., Striegel, A., Poellabauer, C.: Wireless reliability: Rethinking 802.11 packet loss. In: WOWMOM. pp. 1–4. IEEE (2008)
15. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard) (Jul 2003), `http://www.ietf.org/rfc/rfc3550.txt`
16. Schulzrinne, H., Rao, A., Lanphier, R.: Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard) (Apr 1998), `http://www.ietf.org/rfc/rfc2326.txt`

17. Walker, J.: Ent: A pseudorandom number sequence test program., http://www.fourmilab.ch/random/
18. Willems, F.M.J.: The context-tree weighting method : Extensions. IEEE Transactions on Information Theory 44(2), 792–798 (1998)
19. Willems, F.M.J., Shtarkov, Y.M., Tjalkens, T.J.: The context-tree weighting method: basic properties. IEEE Transactions on Information Theory 41(3), 653–664 (1995)
20. Wolf, S., Wullschleger, J.: Oblivious transfer is symmetric. In: Vaudenay, S. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 222–232. Springer (2006)
21. Wullschleger, J.: Oblivious-transfer amplification. In: Naor, M. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4515, pp. 555–572. Springer (2007)
22. 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)

A Equipment and Configuration

The wireless router used for the purpose of the experiment is a Netgear N600 (WNDR3800). It is a dual band (2.4 or 5.0 GHz), 802.11a/b/g/n capable device. It is powered by an Atheros AR7161 rev. 2 680 MHz CPU, and has 128MiB of RAM and 16MiB of flash memory. [6]

The OpenWRT version installed on the router is `10.03.1`, the latest at the time of writing. The open source LIVE555™ Media Server (updated to version `2012.05.17`) was installed, and used for streaming packets with the RTP/RTSP protocol.

The USB Wireless LAN adapter used during the experiment is a Linksys AE2500 (branded Cisco). This adapter is capable of working according to the latest WIEEE 802.11n standard (but can also work in 802.11b or 802.11g compatible modes). It supports dual band communication (2.4 GHz or 5 GHz).[7]

On the client side, the stream was displayed using the open source media player VLC (version 2.0.2 "Twoflower"), and packets were dumped using the WireShark open source sniffing tool.

The wireless configuration used for the router/access point (AP) during the experiment is shown in table 3.

B RTP Packet Header

The header of an RTP packet is shown in Figure 1, as described in [15]. The first field specifies the protocol revision used (the current version is 2). The padding

[6] Full specifications are available at the manufacturer's website: `http://www.netgear.com/home/products/wirelessrouters/high-performance/WNDR3700.aspx`.

[7] Full specifications are available at the manufacturer's website: `http://home.cisco.com/en-eu/products/adapters/AE2500`.

W-LAN:	IEEE 802.11n (2.4 GHz band)
AP Channel:	6 (2437 MHz)
AP Security:	WPA-CCMP(AES) Pre-Shared Key (PSK)
Active STA's:	1

Table 3. Configuration of the Wireless router-AP for the experiment.

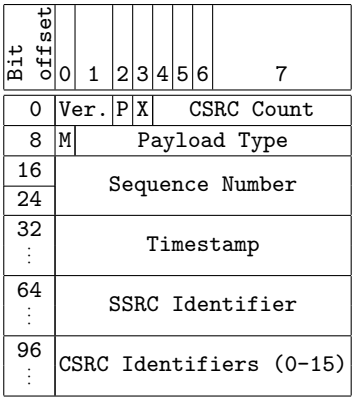


Fig. 1. RTP Packet Header.

(`P`) field indicates if there are extra padding bytes at the end of the packet. `X`, extension, indicates the presence of application or profile specific headers between the standard header and the payload data. Extensions of the protocol can also use the marker (`M`) field, to indicate that the current packet has some special relevance for the application. The Real-time Transport Protocol allows the transmitted information to be generated by multiple sources. In this case, the packet flow will be synchronized by a unique *synchronization source* (SSRC), while any additional source will act as *contributing source* (CSRC). Both SSRC and CSRC's have unique identifiers, whose value is contained in the `SSRC Identifier` and `CSRC Identifiers` fields respectively. The maximum number of CSRC's is 16, and the actual number for a specific stream is defined in the `CSRC Count` field. For the purpose of our oblivious transfer protocol, only the synchronization source is used. The RTP header also contains information about the format used for the payload data (`Payload Type`) and specifies for each packet a `Sequence Number` and a `Timestamp`.