The One-Time Pad Revisited

The One-Time Pad Revisited

Christian Matt and Ueli Maurer
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
Email: {mattc, maurer}@inf.ethz.ch

***Abstract*—The one-time pad, the mother of all encryption schemes, is well known to be information-theoretically secure, in contrast to most encryption schemes used in practice, which are at most computationally secure. In this paper, we focus on another, completely different aspect in which the one-time pad is superior to normal encryption, and which surfaces only when the receiver (not only the eavesdropper) is considered potentially dishonest, as can be the case in a larger protocol context in which encryption is used as a sub-protocol.**

For example, such a dishonest receiver (who is, say, coerced by the eavesdropper) can in normal encryption *verifiably* leak the message to the eavesdropper by revealing the secret key. While this leakage feature can provably not be avoided completely, it is more limited if the one-time pad is used. We use the constructive cryptography framework to make these statements precise.

I. Introduction

This paper follows the paradigm of *constructive cryptography* introduced in [1], [2]. A central idea behind it is that a cryptographic primitive or protocol can be seen as providing a construction of a so-called ideal resource from a so-called real resource, for a well-defined notion of construction. One main advantage of this approach is that when designing a complex protocol involving several cryptographic primitives or sub-protocols, one defines the security by the specification of the ideal resource one obtains when using the protocol (e.g. a secure channel), and one can therefore directly use this ideal resource in the construction of a new protocol for a more advanced ideal resource (e.g. a voting functionality). A second main advantage is that the security proof of the overall protocol follows directly, due to a composition theorem of the framework, from the proofs for the individual construction steps, i.e., for the individual cryptographic primitives.

In this approach, the purpose of encryption is to construct a secure channel from a sender Alice to a receiver Bob from a shared secret key and an authenticated channel. Given a key as long as the message, Alice can encrypt her message by bitwise XORing the key to the message, which yields the corresponding ciphertext. She then sends this ciphertext over the channel to Bob, who can recover the message by bitwise XORing the key to the ciphertext. If the key is uniformly random and used only once, this cryptosystem is called *one-time pad*. It was shown in [3] that an eavesdropper Eve does not learn anything about the message given only the ciphertext. This means constructively that the one-time pad can be used to construct a secure channel from a shared secret key and an authenticated channel if Eve is the only dishonest party, regardless of her computational power.

This analysis assumes that Alice and Bob are always honest, which is a standard assumption when analyzing symmetric encryption schemes. However, when channels are used in a more complex system with several parties, this assumption does not always hold. In this paper, we analyze the one-time pad when Bob is potentially dishonest. This allows us to understand situations in which Bob is coerced to give the secret key to another party or in which Bob wants to betray Alice by convincing a third party that she has sent a specific message. If Bob sends Eve the key and Alice sends a message to Bob, Eve can learn the message. Of course, Bob can just send Eve the message, but if she does not trust him, there is no reason for her to believe that he sent her the correct message. However, receiving a key that later decrypts the ciphertext to a meaningful message is more convincing because it might be much harder or even impossible for Bob to find such key. Hence, the resource generally constructed by encryption schemes cannot exclude that Bob convincingly leaks the message to Eve.

It is known that the one-time pad shares a feature with so-called *deniable encryption* schemes introduced in [4] which allows one to find a key for each pair of message and ciphertext such that the ciphertext decrypts to the given message with this key. Hence, once Bob knows the message, he can create a fake key that yields an arbitrary message of his choice. Intuitively, this means that receiving a key from a dishonest Bob after a message was sent is meaningless. We show that this intuition can be formalized in the constructive cryptography framework as follows: The one-time pad can be used to construct a resource which potentially allows Bob at the beginning to decide whether he wants to leak the message to Eve or not. However, he has to make this decision before Alice sends the message, i.e., his choice to leak the message or not cannot depend on the message. In contrast, when using ordinary encryption, one cannot exclude that Bob is still able to verifiably leak the message after it was sent. Hence, in addition to perfect secrecy, the one-time pad provides stronger guarantees than other encryption schemes.

The main contribution of this work is the description of an ideal resource that can be constructed using the one-time pad in a setting with a potentially dishonest receiver. This is the first result of this form involving more than one potentially dishonest party, which is of independent interest as a new type of example in the constructive cryptography framework.

II. Preliminaries

A. Resources and Converters

The results in this paper are formulated using the theory of constructive cryptography. In this section, we introduce the relevant concepts, following [2] and the exposition given in [5]. We consider different types of *systems*, which are objects with *interfaces* via which they interact with their environment. Interfaces are denoted by uppercase letters. One can compose two systems by connecting one interface of each system. The composed object is again a system.

Two types of systems we consider here are *resources* and *converters*. Resources are denoted by small capitals or special symbols such as $\bullet\!\!\longrightarrow$ and have a finite set of interfaces $\mathcal{I}$. Resources with interface set $\mathcal{I}$ are called *$\mathcal{I}$-resources*. Converters have one *inner* and one *outer interface* and are denoted by lowercase Greek letters. The inner interface of a converter α can be connected to interface $I \in \mathcal{I}$ of a resource R. The outer interface of α then serves as the new interface I of the composed resource, which is denoted by $\alpha^I\mathrm{R}$. We also write $\alpha_I\mathrm{R}$ instead of $\alpha_I^I\mathrm{R}$ for a converter α_I. For a vector of converters $\alpha = (\alpha_{I_1}, \ldots, \alpha_{I_n})$ with $I_1, \ldots, I_n \in \mathcal{I}$ and a set $\mathcal{P} \subseteq \{I_1, \ldots, I_n\}$ of interfaces, $\alpha_{\mathcal{P}}\mathrm{R}$ denotes the $\mathcal{I}$-resource that results from connecting α_I to interface I of R for every $I \in \mathcal{P}$. Moreover, $\alpha_{\overline{\mathcal{P}}}\mathrm{R}$ denotes the $\mathcal{I}$-resource one gets when α_I is connected to interface I of R for every $I \in \{I_1, \ldots, I_n\} \setminus \mathcal{P}$.

For two $\mathcal{I}$-resources R and S, the *parallel composition* $\mathrm{R} \parallel \mathrm{S}$ is defined as the $\mathcal{I}$-resource where each interface $I \in \mathcal{I}$ allows to access the corresponding interfaces of both subsystems R and S. The *sequential composition* of converters α and β is denoted by $\alpha\beta$ and defined via $(\alpha\beta)^I\mathrm{R} := \alpha^I\left(\beta^I\mathrm{R}\right)$ for all $\mathcal{I}$-resources R and interfaces $I \in \mathcal{I}$.

B. Basic Resources

We now describe basic resources needed later, starting with communication channels. The channels we consider allow the sender A to send a single message from a fixed message space $M := \{0,1\}^l$ for $l \in \mathbb{N}$ to the receiver B. We define two such channels, which differ in the information an eavesdropper E learns about the message. The notation used here was introduced in [6].

Definition 1. An *authenticated channel*, denoted as $\bullet\!\!\longrightarrow$, is a resource with three interfaces A, B, and E. On input a single message $m \in M$ at interface A, the same message is output at interfaces B and E. Further inputs are ignored.

This channel is called *authenticated* because E cannot modify the message. If an eavesdropper can only learn the length of the transferred message, we get the following resource.

Definition 2. A *secure channel*, denoted as $\bullet\!\!\longrightarrow\!\!\bullet$, is a resource with three interfaces A, B, and E. On input a single message $m \in M$ at interface A, the same message is output at interface B and the length $|m|$ of the message is output at interface E. Further inputs are ignored.

We further introduce a resource which outputs a random value at the interfaces A and B.

Definition 3. A *shared secret key*, denoted as $\bullet\!\!=\!\!\bullet$, is a resource with three interfaces A, B, and E. It outputs a uniformly random value at the interfaces A and B and does not output anything at interface E. All inputs are ignored.

In this notation, the symbol "$\bullet$" can intuitively be interpreted as indicating exclusive access of the party at that side of the resource to the corresponding functionality. For example, in an authenticated channel $\bullet\!\!\longrightarrow$, sending a message is exclusive to Alice, but receiving the message is not exclusive to Bob because it leaks to Eve.

C. Indistinguishability

A *distinguisher* D for resources with n interfaces is a system with $n+1$ interfaces, where n of them connect to the interfaces of a resource and a bit B is output at the remaining one. We write $\mathrm{P}^{D\mathrm{R}}(B=1)$ to denote the probability that D outputs the bit 1 when connected to resource R. The goal of a distinguisher is to distinguish two resources by outputting a different bit when connected to a different resource. We measure its success by the distinguishing advantage.

Definition 4. The *distinguishing advantage* of a distinguisher D for resources R and S is defined as

$$\Delta^D(\mathrm{R},\mathrm{S}) := \left|\mathrm{P}^{D\mathrm{R}}(B=1) - \mathrm{P}^{D\mathrm{S}}(B=1)\right|.$$

If $\Delta^D(\mathrm{R},\mathrm{S}) = 0$ for all distinguishers D, we say R and S are *indistinguishable*, denoted as $\mathrm{R} \equiv \mathrm{S}$.

Remark. The definition of indistinguishability above captures *perfect security*. One can also consider *statistical security* by allowing a small distinguishing advantage and *computational security* by additionally restricting the distinguishers to be computationally bounded. *Information theoretic security* refers to perfect or statistical security.

D. Filtered Resources

In some situations, specific interactions with a resource might not be guaranteed but only potentially available. As an example, consider a channel that potentially allows the receiver to leak the message to an eavesdropper, but does not guarantee the availability of this feature.

To model such situations, we extend the concept of a resource. Let R be an $\mathcal{I}$-resource and let $\phi = (\phi_I)_{I\in\mathcal{I}}$ be a vector of converters. We define the *filtered resource* R_ϕ as a resource with the same set of interfaces $\mathcal{I}$. For a party connected to interface I of R_ϕ, interactions through the converter ϕ_I are guaranteed to be available, while interactions with R directly are only potentially available to dishonest parties. The converter ϕ_I can be seen as a filter shielding specific functionality of interface I. Dishonest parties can potentially remove the filter to get access to all features of the resource R. Formally, R_ϕ is defined as a set of resources; see [2] for more details.

E. Construction of Resources

A *protocol* is a vector of converters with the purpose of constructing a so-called ideal resource from an available real resource. We now define what it means to construct a resource. Depending on which parties are considered potentially dishonest, we get a different notion of construction.

Definition 5. Let R_ϕ and S_ψ be filtered $\mathcal{I}$-resources and let $\pi = (\pi_I)_{I\in\mathcal{I}}$ be a protocol. Further let $\mathcal{U} \subseteq \mathcal{I}$ be the set of interfaces with potentially dishonest behavior. We say π *constructs* S_ψ from R_ϕ *with potentially dishonest* $\mathcal{U}$ if there exist converters $\sigma = (\sigma_U)_{U\in\mathcal{U}}$ such that

$$\forall \mathcal{P} \subseteq \mathcal{U} : \pi_{\overline{\mathcal{P}}}\phi_{\overline{\mathcal{P}}}\mathrm{R} \equiv \sigma_{\mathcal{P}}\psi_{\overline{\mathcal{P}}}\mathrm{S}.$$

The converters σ_U are called *simulators*.

To apply the above definition to an unfiltered resource R, one can formally introduce trivial filters $\phi_I = \mathbf{1}$ for $I \in \mathcal{I}$ that have no effect and consider the filtered resource R_ϕ which is identical to R. In such cases, we will omit the filters. We refer the reader to [2] for more details.

As an example, consider the typical setting for encryption where we want to construct a secure channel $\bullet\!\longrightarrow\!\bullet$ from a shared secret key $\bullet\!=\!\bullet$ and an authenticated channel $\bullet\!\longrightarrow$ and assume that A and B are always honest while E is potentially dishonest. Here, the real resource is $\mathrm{R} := \bullet\!=\!\bullet \parallel \bullet\!\longrightarrow$ and the ideal resource is $\mathrm{S} := \bullet\!\longrightarrow\!\bullet$. By Definition 5, a protocol $\pi = (\pi_A, \pi_B, \pi_E)$ constructs S from R with potentially dishonest $\mathcal{U} := \{E\}$ if there exists a simulator σ_E such that the following two conditions are satisfied:

$$\begin{aligned} \pi_A\pi_B\pi_E\mathrm{R} &\equiv \quad \mathrm{S} \\ \pi_A\pi_B\mathrm{R} &\equiv \sigma_E\mathrm{S} \end{aligned}$$

Intuitively, the first condition ensures that the protocol implements the required functionality and the second condition ensures that whatever Eve can do when connected to the real resource without necessarily following the protocol, she could do as well when connected to the ideal resource by using the simulator σ_E. Eve's protocol is here only hypothetical and would not be implemented in a real system.

III. Encryption with a Dishonest Receiver

In this section, we examine encryption schemes in general. All results hold with respect to information-theoretic security as well as computational security. We investigate which resources one can construct from a shared secret key and an authenticated channel using encryption in the setting in which not only Eve but also Bob could be dishonest. If this is used as part of a larger protocol which allows Bob to exchange messages with Eve, a dishonest Bob could send her the key, resulting in Eve learning the message. Since the security guarantees are preserved under composition in our framework, the ideal resource constructed by an encryption scheme has to reflect this. That is, the ideal resource potentially gives Bob the option to leak the message to Eve. However, this option is not guaranteed. One reason for this is that Bob cannot leak the message in a more restricted setting where he does not have a communication channel to Eve. Therefore, we have a filtered resource where Bob's leakage button is shielded by a filter.

Something else has to be taken into account. Consider for example the one-time pad. The ciphertext there is a uniformly random bit string. Hence, when Alice sends an encrypted message, Bob and Eve get common randomness. For other types of encryption, they could potentially extract the randomness of the key from the ciphertext and thereby also get common randomness. Even if this is not considered to be an issue, it has to be reflected in the ideal resource. Altogether, the resource a typical encryption scheme constructs from a shared secret key and an authenticated channel is a secure channel which potentially gives Bob the option to leak the message to Eve and which potentially gives common randomness to Bob and Eve.

We have just seen that ordinary encryption does not construct a secure channel without additional features from an authenticated channel and a shared secret key if Bob and Eve are potentially dishonest. We now show that no protocol can achieve this (a variant of this was already stated in the appendix of [2] without proof).

Theorem 1. *There exists no protocol $\pi = (\pi_A, \pi_B, \pi_E)$ that constructs a secure channel $\bullet\!\longrightarrow\!\bullet$ from a shared secret key $\bullet\!=\!\bullet$ and an authenticated channel $\bullet\!\longrightarrow$ with potentially dishonest B and E.*

Proof. Assume such protocol π exists. Then, there exist simulators σ_B and σ_E such that the following conditions hold:

$$\pi_A\pi_B\pi_E(\bullet\!=\!\bullet \parallel \bullet\!\longrightarrow) \equiv \quad \bullet\!\longrightarrow\!\bullet \tag{1}$$

$$\pi_A\pi_E(\bullet\!=\!\bullet \parallel \bullet\!\longrightarrow) \equiv \quad \sigma_B\bullet\!\longrightarrow\!\bullet \tag{2}$$

$$\pi_A\pi_B(\bullet\!=\!\bullet \parallel \bullet\!\longrightarrow) \equiv \quad \sigma_E\bullet\!\longrightarrow\!\bullet \tag{3}$$

$$\pi_A(\bullet\!=\!\bullet \parallel \bullet\!\longrightarrow) \equiv \sigma_B\sigma_E\bullet\!\longrightarrow\!\bullet \tag{4}$$

At the beginning, the resource $\pi_A(\bullet\!=\!\bullet \parallel \bullet\!\longrightarrow)$ outputs a key at interface B. Hence, (4) implies that $\sigma_B\sigma_E\bullet\!\longrightarrow\!\bullet$ also outputs a key at interface B. Furthermore, (4) implies that on input a message $m \in M = \{0,1\}^l$ at interface A of $\sigma_B\sigma_E\bullet\!\longrightarrow\!\bullet$ afterwards, the outputs at interfaces B and E agree. Since σ_E does not get any information about the message and the key had been output before the message was input, no output of σ_B depends on the message. However, we can conclude from (1) and (2) that $(\pi_B\sigma_B)^B\bullet\!\longrightarrow\!\bullet \equiv \bullet\!\longrightarrow\!\bullet$, i.e., applying protocol π_B to the output of σ_B gives the correct message. This is only possible with probability 2^{-l}, yielding a contradiction. □

IV. The One-Time Pad with a Dishonest Receiver

In the preceding section, we have seen that ordinary encryption constructs a secure channel that potentially allows the receiver Bob to leak the message to an eavesdropper Eve and potentially gives common randomness to Bob and Eve if they are both dishonest. In this section, we show that, even though it is impossible to construct a secure channel without additional

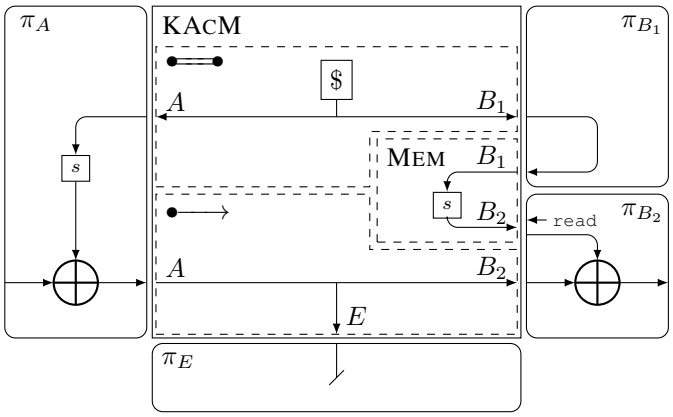

Fig. 1. The resource KAcM, which consists of $\bullet\!\!=\!\!=\!\!\bullet_{A,B_1}$, $\bullet\!\!\longrightarrow_{A,B_2}$, and MEM_{B_1,B_2} composed in parallel, with the protocol converters attached.

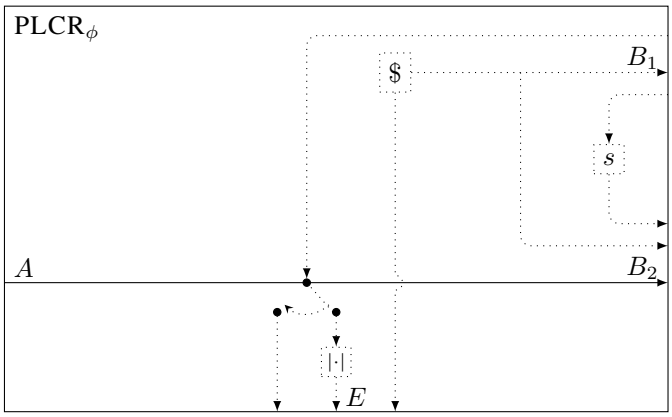

Fig. 2. The resource PLCR_ϕ. Interactions that are not guaranteed but only potentially available to dishonest parties are drawn with dotted lines.

features from a shared secret key and an authenticated channel, the one-time pad can be used to construct a stronger resource than the one constructed by ordinary encryption, namely a resource that does not allow Bob to leak the message anymore once he has received it.

Intuitively, this is because the one-time pad shares the feature with so-called *deniable encryption* [4] that for each pair of message and ciphertext one can find a key such that the ciphertext decrypts to the given message with this key. In case of the one-time pad such key can be obtained by computing the bitwise XOR of the message and the ciphertext. Therefore, Bob can generate a fake key to yield any message of his choice if he already knows the message. This makes receiving a key from Bob in that case useless because there is no way to verify whether it is the correct key. This translates to the ideal resource by not allowing Bob to leak the message after receiving it. Note that this does not work before Bob knows the message. Then, he cannot find a key that will decrypt the ciphertext to a message of his choice.

To capture the fact that Bob can leak the message before he receives it but not afterwards, we split the receiver into two phases, B_1 that is active at the beginning and B_2 that is active after receiving a message. This is necessary to model for example that a receiver is following the protocol at first but changes his strategy depending on the received message. We first describe the real resource used in our construction.

Definition 6. The resource KAcM has the four interfaces A, B_1, B_2, and E and consists of a shared secret key $\bullet\!\!=\!\!=\!\!\bullet_{A,B_1}$ between A and B_1, an authenticated channel $\bullet\!\!\longrightarrow_{A,B_2}$ from A to B_2, and an l-bit memory MEM_{B_1,B_2} writable by B_1 and readable by B_2. At interface B_1, one can input $x \in \{0,1\}^l$ to store x in the memory. On input `read` at interface B_2, the beforehand stored value x is returned to B_2. To model that B_1 and B_2 exist in different phases of the protocol, all inputs at interface B_1 after something is input at interface A and all inputs at interface B_2 before are ignored.

Now we describe the protocol $\pi = (\pi_A, \pi_{B_1}, \pi_{B_2}, \pi_E)$. Let π_A internally store the key it receives at its inner interface and on input a message at the outer interface, compute the bitwise XOR of this message and the key and input the result into the authenticated channel to B_2. When π_{B_1} receives the key, it stores it in the memory. When π_{B_2} receives a ciphertext from A, it reads the key from the memory, computes the bitwise XOR of the key and the ciphertext and outputs the result at its outer interface. Eve's (hypothetical) protocol $\pi_E := \bot$ ignores all inputs. See Fig. 1 for an overview of the protocol and the involved resources.

We now describe the ideal resource which is constructed from KAcM by that protocol. The guaranteed functionality allows A to send a message $m \in M$ to B_2. Moreover, it potentially allows E to learn the length of the message and B_1 to flip a switch such that E afterwards potentially receives the message instead of only its length. Also, B_2 can potentially see whether the switch was flipped. For the same reason we explained in the case of general encryption, B_1, B_2, and E could potentially get common randomness. If B_1 and B_2 are both dishonest, they cannot be prevented from using the memory in KAcM to exchange information. Hence, the ideal resource potentially also allows B_1 to store an l-bit string which can later be read by B_2. See Fig. 2 for an illustration of this resource. The formal definition of it follows.

Definition 7. The resource PLCR has the four interfaces A, B_1, B_2, and E and works as follows:

Initialization

$b \leftarrow$ `false`
$t \leftarrow 1$
$r \leftarrow l$-bit string chosen uniformly at random
output r at interface B_1

Interface A

Input: $m \in M$
$t \leftarrow 2$
output (r, b, m) at interface B_2
if b **then**
  output (r, m) at interface E
else
  output $(r, |m|)$ at interface E

Interface B_1

Input: `leak`
  if $t = 1$ **then**
    $b \leftarrow$ `true`
Input: $(\texttt{store}, x)$, $x \in \{0,1\}^l$
  if $t = 1$ **then**
    $s \leftarrow x$

Interface B_2

Input: `read`
  if $t = 2$ **then**
    output s at interface B_2

Since some of the functionality of PLCR is not guaranteed by the protocol but only potentially available, we introduce the following filters: $\phi_{B_1} := \bot$ and $\phi_E := \bot$ ignore all inputs, ϕ_{B_2} converts inputs of the form (r, b, m) at its inner interface to m and ignores other inputs, and $\phi_A := \mathbf{1}$ forwards all inputs. Let $\phi := (\phi_A, \phi_{B_1}, \phi_{B_2}, \phi_E)$.

Theorem 2. *The protocol π defined above constructs* PLCR_ϕ *from* KAcM *with potentially dishonest B_1, B_2, and E.*

Proof. Let $\sigma := (\sigma_{B_1}, \sigma_{B_2}, \sigma_E)$ for the simulators σ_{B_1}, σ_{B_2}, and σ_E defined below.

Inner Interface of σ_{B_1}

Input: r
  output `leak` at inner interface
  output r at outer interface

Outer Interface of σ_{B_1}

Input: $x \in \{0,1\}^l$
  output $(\texttt{store}, x)$ at inner interface

Initialization of σ_{B_2}

$s \leftarrow$ `null`

Inner Interface of σ_{B_2}

Input: (r, b, m)
  if b **then**
    $s \leftarrow$ returned value from `read` at inner interface
    output $m \oplus r$ at outer interface
  else
    $s \leftarrow m \oplus r$
    output r at outer interface

Outer Interface of σ_{B_2}

Input: `read`
  if $s \neq$ `null` **then**
    output s at outer interface

Inner Interface of σ_E

Input: (r, m)
  output $m \oplus r$ at outer interface
Input: $(r, |m|)$
  output r at outer interface

We have to show that

$$\forall \mathcal{P} \subseteq \{B_1, B_2, E\} : \pi_{\overline{\mathcal{P}}}\text{KAcM} \equiv \sigma_{\mathcal{P}}\phi_{\overline{\mathcal{P}}}\text{PLCR}.$$

We first verify the conditions with $B_1 \in \mathcal{P}$, i.e., with σ_{B_1} present on the right hand side. In this case, σ_{B_1} outputs a uniformly random l-bit string r at the beginning, as the resource on the left hand side does at interface B_1. Since σ_{B_1} outputs `leak` at its inner interface, the local variable b in PLCR is `true` when a message m is input at interface A. Hence, σ_{B_2} and σ_E (if present) both output $m \oplus r$. As in the resource on the left hand side, the bitwise XOR of the outputs at interfaces B_1 and B_2 as well as the outputs at interfaces B_1 and E yields the input message $m = r \oplus (m \oplus r)$. On input `read` at interface B_2, σ_{B_2} returns the value stored before a message was input at interface A. Therefore, the resources on the left and those on the right hand side are indistinguishable in these four cases.

Now consider the cases with $B_1 \notin \mathcal{P}$. There, b is `false` when a message m is input at interface A, so σ_E will output a uniformly random l-bit string r in case $E \in \mathcal{P}$. If $B_2 \notin \mathcal{P}$, this is indistinguishable from the output at interface E of the resource on the left hand side. Otherwise, σ_{B_2} outputs r as well and sets its internal variable s to $m \oplus r$. Hence, inputting `read` and computing the bitwise XOR of the returned value s and the previous output at interface B_2 results in the message $m = (m \oplus r) \oplus r$, as in the resource on the left hand side. Therefore, the resources are indistinguishable in all cases. □

V. Conclusion and Future Work

We have shown in the constructive cryptography framework that it is impossible to construct a secure channel without additional features from a shared secret key and an authenticated channel when the receiver is potentially dishonest. Furthermore, we have described which weaker channel typical encryption schemes construct and how the one-time pad can be used to construct a stronger resource.

An interesting problem which could not be discussed in this paper is the one-time pad and encryption in general with a potentially dishonest sender.

Acknowledgment

The work was supported by the Swiss National Science Foundation (SNF), project no. 200020-132794.

References

[1] U. Maurer, "Constructive cryptography – a new paradigm for security definitions and proofs," in *Theory of Security and Applications*, ser. Lecture Notes in Computer Science, S. Mödersheim and C. Palamidessi, Eds. Springer Berlin Heidelberg, 2012, vol. 6993, pp. 33–56.

[2] U. Maurer and R. Renner, "Abstract cryptography," in *The Second Symposium in Innovations in Computer Science, ICS 2011*, B. Chazelle, Ed. Tsinghua University Press, Jan. 2011, pp. 1–21.

[3] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell Systems Technical Journal*, vol. 28, pp. 656–715, 1949.

[4] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable encryption," in *Advances in Cryptology — CRYPTO '97*, ser. Lecture Notes in Computer Science, B. S. Kaliski Jr., Ed., vol. 1294. Springer Berlin Heidelberg, 1997, pp. 90–104.

[5] U. Maurer, A. Rüedlinger, and B. Tackmann, "Confidentiality and integrity: A constructive perspective," in *Theory of Cryptography — TCC 2012*, ser. Lecture Notes in Computer Science, R. Cramer, Ed., vol. 7194. Springer Berlin Heidelberg, 2012, pp. 209–229.

[6] U. Maurer and P. Schmid, "A calculus for security bootstrapping in distributed systems," *Journal of Computer Security*, vol. 4, no. 1, pp. 55–80, 1996.