Basing Cryptographic Protocols on Tamper-Evident Seals*

Tal Moran and Moni Naor**

Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel

Abstract. In this paper we attempt to formally study two very intuitive physical models: sealed envelopes and locked boxes, often used as illustrations for common cryptographic operations. We relax the security properties usually required from locked boxes (such as in bit-commitment protocols) and require only that a broken lock or torn envelope be identifiable to the original sender. Unlike the completely impregnable locked box, this functionality may be achievable in real life, where containers having this property are called "tamper-evident seals". Another physical object with this property is the "scratch-off card", often used in lottery tickets. We show that scratch-off cards can be used to implement bit-commitment and coin flipping, but not oblivious transfer. Of particular interest, we give a strongly-fair coin flipping protocol with bias bounded by $O(1/r)$ (where r is the number of rounds), beating the best known bias in the standard model even with cryptographic assumptions.

1 Introduction

In this paper we consider the use of "tamper-evident seals" in cryptographic protocols. A tamper-evident seal is a primitive based on very intuitive physical models: the sealed envelope and the locked box. These are often used as illustrations for a number of basic cryptographic primitives: For example, encryption is often depicted as placing a message in a locked box (that cannot be opened without a key), while bit commitment is usually illustrated as a sealed envelope.

In a bit-commitment protocol one party, Alice, commits to a bit b to Bob in such a way that Bob cannot tell what b is. At a later time Alice can reveal b, and Bob can verify that this is indeed the bit to which she committed. The standard illustration used for a bit-commitment protocol is Alice putting b in a sealed envelope, which she gives to Bob. Bob cannot see through the envelope (so cannot learn b). When Alice reveals her bit, she lets Bob open the envelope so he can verify that she didn't cheat.

The problem with the above illustration is that a physical "sealed envelope" is insufficient for bit-commitment: Bob can always tear open the envelope before Alice officially allows him to do so. Even a locked box is unlikely to suffice; many

* This work was partially supported by the Minerva Foundation
** Incumbent of the Judith Kleeman Professorial Chair.

protocols based on bit-commitment remain secure only if no adversary can *ever* open the box without a key. A more modest security guarantee seems to be more easily obtained: an adversary may be able to tear open the envelope but Alice will be able to recognize this when she sees the envelope again.

"Real" closures with this property are commonly known as "tamper evident seals". These are used widely, from containers for food and medicines to high-security government applications. Another common application that embodies these properties is the "scratch-off card", often used as a lottery ticket. This is usually a printed cardboard card which has some areas coated by an opaque layer (e.g., the possible prizes to be won are covered). The text under the opaque coating cannot be read without scratching off the coating, but it is immediately evident that this has been done.

In this paper we attempt to clarify what it means to use sealed envelopes in a cryptographic protocol. In particular, we study their applicability to coin flipping (CF), zero-knowledge protocols, bit commitment (BC) and oblivious transfer (OT), some of the most fundamental primitives in modern cryptography.

Our Results. In this paper we show that the sealed envelopes we consider can be used to implement standard cryptographic protocols. We construct protocols for "weakly fair" CF (in which the result is 0, 1 or *invalid*) and BC.

The existence of BC immediately implies the existence of a zero-knowledge proof system for any NP language [16]. In the full version, we also show a non-interactive zero knowledge proof system (with a preprocessing stage) for any NP language that can be implemented using pre-printed scratch-off cards.

A possibly practical application of our model is the "cryptographic randomized response technique" (CRRT), defined by Ambainis et al. [2]. "Randomized response" is a polling technique used when some of the answers to the poll may be stigmatizing (e.g., "do you use drugs?"). The respondent lies with some known probability, allowing statistical analysis of the results while letting the respondent disavow a stigmatizing response. In a CRRT, there is the additional requirement that a malicious respondent cannot bias the results more than by choosing a different answer. The techniques described by Ambainis et al. achieve this, but require "heavy" cryptographic machinery (such as OT), or quantum cryptography. In the full version of the paper we give a simple protocol for CRRT using scratch-off cards.

One of the most interesting results is a protocol for "strongly fair" CF (where the result for an honest player must be either 0 or 1 even if the other player quits before finishing the protocol) with bias bounded by $O(\frac{1}{r})$, where r is the number of rounds. In the standard model, even with cryptographic assumptions, the best known bias for a protocol with r rounds is $O(\frac{1}{\sqrt{r}})$ (due to Cleve [8]). For a large class of protocols (including any that rely on bit-commitment or weakly fair coin flipping as a black-box), an unpublished result of Cleve and Impagliazzo [9] shows this is the best possible bias.

An important contribution of this paper is the *formal* analysis for the models and protocols we construct. We show that the protocols are *Universally Com-*

posable in the sense of Canetti [7]. This allows us to use them securely as "black-boxes" in larger constructions.

On the negative side, we show that our protocol for strongly fair CF using scratch-off cards is optimal: it is impossible to do better than $O(\frac{1}{r})$ bias (this follows from a careful reading of the proof in [8]; the proof appears in the full version). We show that OT cannot be implemented using scratch-off cards without additional assumptions (note that we show the impossibility of *any* type of OT, not just universally composable realizations).

Seals in Different Flavors. In the full version of the paper, we consider some additional variants of tamper-evident seals. Roughly speaking, the difference arises from whether or not a party can distinguish their own containers from those of the other party without opening the seal, and whether or not an honest player can "break" the seal. Scratch-off cards correspond to the distinguishable case in which the honest party *can* break the seal. We show that in a distinguishable "weak-lock" model, where the honest party cannot break the seal, even BC cannot be implemented (the proof is similar to the impossibility of OT , while in the indistinguishable case it is possible to implement both BC and OT. Due to space constraints, this paper will concentrate solely on the "distinguishable envelope" (scratch-off card) model. Note that in the standard model of cryptography, where the parties exchange messages and there is no access to outside physical resources, we do not know how to implement any of these closures.

Related Work. To the best of our knowledge, this is the first attempt at using tamper evident seals for cryptographic protocols. Ross Anderson discusses "packaging and seals" in the context of security engineering [3], however the use of tamper-evidence does not extend to more complex protocols. Blaze gives some examples of the reverse side of the problem: cryptanalysis of physical security systems using techniques from computer science [4, 5]. Using scratch-off cards in the lottery setting can be described as a very weak form of CF, however we do not believe this has ever been formally analyzed (or used in more complex protocols).

On the other hand, basing cryptographic protocols on physical models is a common practice. Perhaps the most striking example is the field of quantum cryptography. One of the inspirations for this work was the idea of "Quantum Bit Escrow" (QBE) [1], a primitive that is very similar to a tamper-evident seal and that can be implemented in a quantum setting. There are, however, significant differences between our definitions of tamper-evident seals and QBE. In particular, in QBE the adversary may "entangle" separate escrowed bits and "partially open" commitments. Thus, while unconditionally secure bit-commitment is impossible in the pure quantum setting [19, 18], it is possible in ours.

Much work has been done on basing BC and OT on the physical properties of communication channels, using the random noise in a communication channel as the basis for security. Both BC and OT were shown to be realizable in the *Binary Symmetric Channel* model [11, 10], in which random noise is added to the channel in both directions with some known, constant, probability. Later work shows that they can also be implemented, under certain conditions, in the weaker (but more convincing) *Unfair Noisy Channel* model [13, 12], where the error probability is not known exactly to the honest parties, and furthermore can be influenced by the adversary. Our construction for 1-2 OT uses some of the techniques and results from [13].

One of the motivations for this work was the attempt to construct cryptographic protocols that are implementable by humans without the aid of computers. This property is useful, for example, in situations where computers cannot be trusted to be running the protocol they claim, or where "transparency" to humans is a requirement (such as in voting protocols). Many other examples exist of using simple physical objects as a basis for cryptographic protocols that can be performed by humans, some are even folklore: Sarah Flannery [15] recounts a childhood riddle that uses a doubly-locked box to transfer a diamond between two parties, overcoming the corrupt postal system (which opens any unlocked boxes) despite the fact that the two parties have never met (and can only communicate through the mail). Fagin, Naor and Winkler [14] assembled a number of solutions to the problem of comparing secret information without revealing anything but the result of the comparison using a variety of different physical methods. Schneier devised a cipher [21] that can be implemented by a human using a pack of cards. In a lighter vein, Naor, Naor and Reingold [20] give a protocol that provides a "zero knowledge proof of knowledge" of the correct answer to the children's puzzle "Where's Waldo" using only a large newspaper and scissors. A common thread in these works is that they lack a formal specification of the model they use, and a formal proof of security.

Organization of the Paper. In Section 2, we give a formal definition for tamper-evident envelopes and the functionalities we attempt to realize using them. In Section 3 we discuss the capabilities of this model, showing that OT is impossible and giving protocols for BC and strongly-fair CF with bias $1/r$. Section 4 contains the discussion and some open problems.

2 The Model: Ideal Functionalities

Many two-party functionalities are easy to implement using a trusted third party that follows pre-agreed rules. In proving that a two-party protocol is secure, we often want to say that it behaves "as if it were performed using the trusted third party". A formalization of this idea is the "Universally Composable" model defined by Canetti [7]. In the UC model, the trusted third party is called the *ideal functionality*. The point of the model is that protocols that are secure in the UC have very strong security properties, such as security under composition and security that is retained when the protocol is used as a sub-protocol to replace an ideal functionality. This security guarantee allows us to simplify many of our proofs, by showing separately the security of their component sub-protocols.

Note that our impossibility results are not specific to the UC model: the impossibility results for BC (in the full version of the paper), OT (Section 3) and the lower bound for strongly fair CF (also in the full version) hold even for the much weaker "standard" notions of these functionalities[1].

[1] we *do* assume the "interface" supplied by the envelope primitives is the one defined by the ideal functionality (e.g., we do not allow putting one envelope inside another)

In this section we formally define tamper-evident envelopes in terms of their ideal functionalities. For completeness, we also give the definitions of the primitives we are trying to implement (CF, BC, and OT). We restrict ourselves to the two-party case, and to adversaries that decide at the beginning of the protocol whether to corrupt one of the parties or neither.

Distinguishable Envelopes. This functionality models an opaque envelope (or scratch-off card). Without opening the envelope it is impossible to determine its contents. Any party can decide to rip open the envelope (breaking the seal), but this will be evident to the envelope's creator if the envelope is returned.

In succeeding sections, we assume we are given a realization of this functionality and attempt to construct a protocol for a "target" functionality (these are described below)

Functionality $\mathcal{F}_{DE}$ contains an internal table that consists of tuples of the form $(id, value, creator, holder, state)$. The table represents the state and location of the tamper-evident envelopes, and contains one entry for each existing envelope, indexed by the envelope's id. We denote $value_{id}$, $creator_{id}$, $holder_{id}$ and $state_{id}$ the corresponding values in the table in row id (assuming the row exists). The table is initially empty. The functionality is as follows, running with parties $P_1, \ldots, P_n$ and adversary S:

Seal $(id, value)$ On receiving this command from party P_i, if this is the first message with id id store the tuple $(id, value, P_i, P_i, \text{sealed})$ in the table (if this is not the first message with id id, do nothing).

Send (id, P_j) On receiving this command from party P_i, the functionality checks if an entry for envelope id appears in the table and that $holder_{id} = P_i$. If so, it outputs $(\text{Receipt}, id, P_i, P_j)$ to P_j and S and replaces the entry in the table with $(id, value_{id}, creator_{id}, P_j, state_{id})$. Otherwise, it does nothing.

Open id On receiving this command from P_i, the functionality checks that an entry for envelope id appears in the table, that $holder_{id} = P_i$. If not, the message is ignored. Otherwise, it sends $(\text{Opened}, id, value_{id}, creator_{id})$ to P_i and S. It then replaces the entry in the table with $(id, value_{id}, creator_{id}, holder_{id}, \text{broken})$.

Verify id On receiving this command from P_i, the functionality checks that an entry for envelope id appears in the table and that $holder_{id} = P_i$. If not, the message is ignored. If so, it considers $state_{id}$. If $state_{id} = \text{broken}$ it sends $(\text{Verified}, id, \text{broken})$ to P_i and S. Otherwise, it sends $(\text{Verified}, id, \text{ok})$ to P_i and S.

A Note about Notation. In the interests of readability, we will often refer to parties "preparing", "verifying" and "opening" envelopes, instead of specifying that they send the corresponding messages (**Seal**, **Verify**, **Open**) to the functionality and wait for the appropriate response.

Weakly Fair Coin Flipping. This functionality models coin flipping in which the result of the coin flip can be 0, 1 or $\perp$. The result of the flip c should satisfy: $\Pr[c = 0] \leq \frac{1}{2}$ and $\Pr[c = 1] \leq \frac{1}{2}$. This is usually what is meant when talking about "coin flipping" (for instance, in Blum's protocol [6]). The $\perp$ result corresponds to the case where one of the parties deviated from (or prematurely aborted) the protocol. Under standard cryptographic assumptions (such as the existence of one-way functions), weakly fair coin flipping is possible. Conversely, in the standard model the existence of weakly fair coin flipping implies one-way functions [17].

Functionality $\mathcal{F}_{WCF}$ is as follows, with parties Alice and Bob (in this definition we only allow Bob to trigger an invalid output):

Value The sender of this command is called the *initiator*. The other party is the *receiver*. When this command is received, the functionality chooses a uniform value $d \in \{0, 1\}$. If the receiver is corrupted, the functionality then outputs "approve d" to the receiver. In that case, the functionality ignores all input until it receives either a **Continue** command or a **Stop** command from the receiver. If the receiver is not corrupted, the functionality proceeds as if he had sent a **Continue** command.

Stop When this command is received from a corrupt receiver (in response to an "approve d" message) the functionality outputs $\perp$ to all parties and halts.

Continue When this command is received from the receiver (in response to an "approve d" message), the functionality outputs "coin is d" to all parties and halts.

Strongly Fair Coin Flipping with Bias p (adapted from [7]). This functionality models a coin flip between two parties with a bounded bias. If both parties follow the protocol, they output an identical uniformly chosen bit. Even if one party does not follow the protocol[2], the other party outputs a random bit whose bias towards 0 or 1 is at most p.

Functionality $\mathcal{F}_{SCF}$ is as follows:

Value When this command is received for the first time from any party, the functionality chooses a value $d \in \{0, 1\}$. It then outputs "accept d?" to the adversary. If the adversary responds with **no**, with probability p the functionality outputs "value $1 - d$" to all parties and with probability $1 - p$ outputs "value d" to all parties. If the adversary responds with **yes** (or does not respond), the functionality outputs "value d" to all parties.

Bit Commitment (adapted from [7]). Functionality $\mathcal{F}_{BC}$:

Commit b The issuer of this command is called the sender, the other party is the receiver. On receiving this command the functionality records b and outputs "committed" to the receiver. It then ignores any other commands until it receives the **Open** command from the sender.

Open On receiving this command from the sender, the functionality outputs "opened b" to the receiver.

---

[2] in "not following the protocol" we include halting (failing to respond). The UC model does not explicitly handle this behaviour, however we can treat it as a special "halt" command that can be sent by the adversary

2.1 Intermediate Functionalities

In order to simplify the presentation, in the following sections we will present protocols that realize functionalities that are slightly weaker than the ones we want. We then use standard amplification techniques to construct the "full" functionalities from their weak version. In this section we define these intermediate functionalities and give the amplification lemmas we use to construct the stronger versions of these primitives. These definitions are in the spirit of [13].

(p, q)-Weak Bit-Commitment. This functionality models bit commitment where a corrupt sender can cheat with probability q while a corrupt receiver can cheat with probability p. The result of failing to cheat is asymmetric. In the case of a corrupt receiver, an unsuccessful attempt to cheat causes the sender to be notified, while a corrupt sender risks nothing in attempting to cheat. Note that an (ϵ, ϵ)-WBC protocol is a regular bit-commitment protocol when ϵ is negligible. Formally, functionality $\mathcal{F}_{(p,q)-WBC}$ proceeds as follows:

Commit b The issuer of this command is called the sender, the other party is the receiver. On receiving this command the functionality records b and outputs "committed" to the receiver. It then ignores any other commands until it receives an **Open** command from the sender, an **OpenFlip** command from a corrupt sender or a **Break** command from a corrupt receiver.

Open b On receiving this command from the sender, the functionality checks that the sender previously sent a **Commit** b command. If so, or if the sender is corrupt and previously sent a **CanEquivocate** command whose response was "Can Equivocate", the functionality outputs "opened b" to the receiver. Otherwise the command is ignored.

CanEquivocate On receiving this command from a corrupt sender, choose a value r uniformly in $[0, 1]$. If $r \geq q$ send "No Equivocation" to the sender and ignore further **CanEquivocate** messages. If $r < q$ send "Can Equivocate" to the sender.

Break On receiving this command from a corrupt receiver, choose a value r uniformly in $[0, 1]$. If $r \geq p$ send "cheating receiver" to the sender and receiver and then halt. Otherwise, send b to the receiver.

We can amplify any (p, q)-WBC protocol when $p, q < 1$, meaning that the existence of such a protocol implies the existence of regular BC.

Theorem 1. *Let P be a (p, q)-WBC protocol and $p, q < 1$. Then there exists an (ϵ, ϵ)-WBC for any $\epsilon > 0$ using $O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)$ invocations of P*

The proof for this theorem is straightforward, and will be given in the full version.

(p, q)-Remotely Inspectable Seal (RIS). This functionality is used in our protocol for strongly fair CF. It is a strengthened version of a tamper-evident seal. With a tamper-evident seal, only its holder can interact with it. Thus, either the sender can check if it was opened, or the receiver can verify that the sealed contents were not changed, but not both at the same time. A remotely inspectable seal is one that can be tested "remotely" (without returning it to the sender). Unfortunately, we cannot realize the ideal version, and therefore relax it somewhat: we allow a corrupt receiver to learn the committed bit during the verification process, and only then decide (assuming he did not previously break the seal) whether or not the verification should succeed. Our definition is actually a further relaxation[3]: both sender and receiver may cheat with some probability. A corrupt sender can cause the result of the **Open** command to be a uniformly random value (instead of a specific value determined in the commit stage). The receiver will catch the sender with probability at least $1-q$. A corrupt receiver who opens the commitment before the verify stage will be caught with probability $1-p$.

Formally, the functionality maintains an internal state variable $v = (v_b, v_s)$ consisting of the committed bit v_b and a "seal" flag v_s. It accepts the commands:

Commit b The issuer of this command is called the sender, the other party is the receiver. b can be either 0, 1 or (if the sender is corrupt) $\perp$. If $b \in \{0,1\}$ (the sender did not try to cheat), the functionality sets $v \leftarrow (b, \text{sealed})$. If $b = \perp$ (the sender tried to cheat) with probability q (the sender cheated successfully) $v \leftarrow (r, \text{sealed})$ (where r is randomly chosen from $\{0,1\}$) and with probability $1-q$ it sets $v \leftarrow (\perp, \text{sealed})$ In any case the functionality concludes by outputting "committed" to the receiver, and ignoring any subsequent **Commit** commands.

Open This command is sent by the receiver. If $v_b \in \{0,1\}$ the functionality outputs "opened b" to the receiver. Otherwise it outputs "invalid" to the receiver. If $v_s = \text{sealed}$, with probability $1-p$ the functionality sets $v_s \leftarrow \text{open}$

Verify If $v_s \neq \text{sealed}$, the functionality outputs "invalid" to the sender. Otherwise (no opening was detected), the functionality outputs "verifying value b" to the adversary and waits for a response. If the adversary responds with **ok**, the functionality outputs "sealed" to the sender, otherwise it outputs "invalid" to the sender. After receiving this command from the sender (and responding appropriately), the functionality ignores any subsequent **Verify** commands.

A (p,q)-Remotely Inspectable Seal can be amplified for any $p < 1$ and $q \leq 1$:

Theorem 2. *Let P be a (p,q)-RIS protocol $p < 1$ and $q \leq 1$. Then there exists an (ϵ,ϵ)-RIS for any $\epsilon > 0$ using $O\left(\log^2\left(\frac{1}{\epsilon}\right)\right)$ invocations of P*

Note that the amplification works even if $q = 1$: this is because the adversary doesn't have full control over the revealed bit but can only cause it to be a random bit. The proof of this theorem will appear in the full version.

3 Capabilities of the Distinguishable Envelope Model

Oblivious Transfer is Impossible. Let Alice be the sender and Bob the receiver. Consider Alice's bits a_0 and a_1, as well as Bob's input c, to be random

[3] This second relaxation is only for convenience; we can remove it using amplification as noted in Theorem 2

variables taken from some arbitrary distribution. Alice's view of a protocol execution can also be considered a random variable $V_A = (a_0, a_1, r_A, N_1, \ldots, N_n)$, consisting of Alice's bits, random coins (r_A) and the sequence of messages that comprise the transcript as seen by Alice. In the same way we denote Bob's view with $V_B = (c, r_B, M_1, \ldots, M_n)$, consisting of Bob's input and random coins and the sequence of messages seen by Bob.

The essence of oblivious transfer (whether universally composable or not) is that Bob gains information about one of Alice's bits, but Alice does not know which one. We can describe the information Bob has about Alice's bits using Shannon entropy, a basic tool of information theory. The Shannon entropy of a random variable X, denoted $H(X)$ is a measure of the "uncertainty" that resides in X. When X has finite support: $H(X) = -\sum_x \Pr[X = x] \log \Pr[X = x]$.

Suppose Bob's view of a specific protocol transcript is v_B. What Bob learns about a_i ($i \in \{0, 1\}$) can be described by the conditional entropy of a_i given Bob's view of the protocol. We write this $H(a_i \mid V_B = v_B)$. If Bob knows a_i at the end of the protocol then $H(a_i \mid V_B = v_B) = 0$ since there is no uncertainty left about the value of a_i given Bob's view. If Bob has no information at all about a_i then $H(a_i \mid V_B = v_B) = 1$, since there are two equally likely values of a_i given Bob's view.

We show that in any protocol in the DE Model, Alice can calculate the amount of information Bob has about each of her bits:

Theorem 3. *For any protocol transcript where $V_A = v_A$ and $V_B = v_B$, both $H(a_0 \mid V_B = v_B)$ and $H(a_1 \mid V_B = v_B)$ are completely determined by v_A*

In any OT protocol, Bob must have full information about a_c (c is Bob's "choice" bit): $H(a_c \mid V_B = v_B) = 0$, and no information at all about a_{1-c}: $H(a_c \mid V_B = v_B) = 1$. The theorem states that in this case Alice can determine c, violating the "obliviousness" requirement of the protocol.

Due to space considerations, we do not give the proof of Theorem 3 here. The basic idea is that for any injection f and any random variable Y, the event $Y = y$ is identical to the event $f(Y) = f(y)$. Therefore, for any two random variables X and Y, it holds that $H(X \mid Y = y) = H(X \mid f(Y) = f(y))$. We show an injection from V_B to V_A, r_B, c. Since r_B and c must be independent of (a_0, a_1), we can conclude that Alice can compute $H(a_0 \mid V_B = v_B)$ and $H(a_1 \mid V_B = v_B)$ for any transcript. A similar argument holds for Bob.

A $(\frac{3}{4}, \frac{1}{2})$-Weak Bit Commitment Protocol. We will show a weak bit commitment protocol, and apply Theorem 1 to amplify the protocol and construct a "standard" bit commitment protocol.

To implement **Commit** b:

1. Bob prepares four sealed envelopes, two containing a 0 and two a 1 in random order. Bob sends the envelopes to Alice
2. Alice opens three envelopes (chosen randomly), and verifies that they are not all the same. Let r be the value in the remaining (sealed) envelope. Alice sends $d = b \oplus r$ to Bob.

To implement **Open**:

1. Alice sends b and the sealed envelope to Bob.
2. Bob verifies that it is sealed, then opens it. He checks that $d = b \oplus r$.

Intuitively, Bob can cheat only by sending three envelopes with the same bit in the commit phase. However, Alice will catch him with probability $\frac{1}{4}$. Alice can cheat by opening less than three envelopes (and guessing d). She will then be caught in the open phase with probability at least $\frac{1}{2}$. A proof of security will appear in the full version.

A Strongly Fair Coin Flipping Protocol with Bias $O(\frac{1}{r})$. The construction uses remotely inspectable seals (defined in Sec. 2.1), which we then show how to implement in the DE model. The idea is similar to the "standard" CF protocol using BC: Alice commits to a random bit a. Bob sends Alice a random bit b, after which Alice opens her commitment. The result is $a \oplus b$.

The reason that this is not a strongly fair CF protocol is that Alice learns the result of the toss before Bob, and can decide to quit before opening her commitment. Using RIS instead of BC solves this problem, because Bob can open the commitment without Alice's help.

Ideally, we would like to replace BC with RIS (and have Alice verify that Bob didn't break the seal before sending b). This seems to work; If Bob quits before verification, or if the verification fails, Alice can use a as her bit, because Bob had to have decided to quit before seeing a. If Bob quits after verification (and the verification passed), Alice can use $a \oplus b$, since Bob sent b before learning a. This idea fails, however. The reason is that RIS allows Bob to see the committed bit *during* verification. If he doesn't like it, he can cause the verification to fail.

We can overcome the problem with probability $1 - \frac{1}{r}$ by doing the verification in r rounds. The trick is that Alice secretly decides on a "threshold round": after this round a failure in verification won't matter. Bob doesn't know which is the threshold round (he can guess with probability at most $1/r$). If Bob decides to stop before the threshold round, either he did not attempt to illegally open a commitment (in which case his decision to stop cannot depend on the result of the coin flip), or he illegally opened all the remaining commitments (opening less than that gives no information about the result). In this case all subsequent verifications will fail, so he may as well have simply stopped at this round (note that the decision to open is made before knowing the result of the coin flip). Clearly, anything Bob does after the threshold round has no effect on the result. Only if he chooses to illegally open commitments during the threshold round can this have an effect on the outcome (since in this case, whether or not the verification fails determines whether Alice outputs a or $a \oplus b$). The full protocol is as follows:

1. Alice chooses r random bits $a_1, \ldots, a_r$ and commits to each bit using the RIS scheme (this is done in parallel). Denote $a = a_1 \oplus \cdots \oplus a_r$,
2. Bob chooses a random bit b. If Alice quits before finishing the commit stage, Bob outputs b. Otherwise, he sends b to Alice.
3. If Bob quits before sending b, Alice outputs a. Otherwise, Alice chooses a secret index $j \in \{1, \ldots, r\}$.

4. The protocol now proceeds in r rounds. Round i has the following form:
   (a) Alice verifies that Bob did not open the commitment for a_i.
   (b) Bob opens the commitment for a_i.
5. If the verification is successful up to round j (and Bob didn't quit up to round j), Alice outputs $a \oplus b$. Otherwise, Alice outputs a.
6. Bob always outputs $a \oplus b$ (If Alice quits before completing the verification rounds, Bob opens the commitments without verification).

Implementation of Remotely Inspectable Seals. We start by giving a $(\frac{1}{2}, 1)$-RIS protocol. We can then apply Theorem 2 to amplify it to (ϵ, ϵ)-RIS for some negligible ϵ. To implement **Commit** b:

1. Alice sends two envelopes to Bob, each containing the bit b.

To implement **Verify**:

1. Alice initiates a (weakly fair) coin flip with Bob.
2. Denote the result of the coin flip r. Bob returns envelope r to Alice.
3. Alice waits for the result of the coin flip and the envelope from Bob. If the result is $\perp$, or if Bob does not return an envelope, Alice outputs "opened". Otherwise, Alice verifies that Bob returned the correct envelope, and that it is still sealed. If both of these conditions are not satisfied, she outputs "sealed", otherwise she outputs "opened"

To implement **Open**:

1. Bob randomly chooses one of the envelopes in his possession (if he already returned one to Alice during the **Verify** stage then he opens the one he has left). He outputs the contents of the envelope.

4 Discussion and Open Problems

The protocols we describe in this paper can be performed by unaided humans, however they require too many "envelopes" to be practical for most uses. It would be useful to construct protocols that can be performed with a smaller number of envelopes or with a smaller number of rounds.

Another point worth mentioning is that the protocols we construct only require one of the parties to seal and verify envelopes. Thus, the binding property is only used in one direction, and the tamper-evidence and hiding properties in the other. This property is useful when we want to implement the protocols in a setting where one of the parties may be powerful enough to open the seal undetectably. (for instance, in the context of voting, where one of the parties could be "the government" while the other is a private citizen)

In both the weakly and strongly fair CF protocols, only the first round requires envelopes to be created, and their contents do not depend on communication with the other party. This allows the protocols to be implemented using scratch-off cards (which must be printed in advance). In particular, the weakly fair CF protocol can be implemented with a scratch-off card using only a small number of areas to be scratched (this protocol is given in the full version).

In the case of BC, our protocol requires the powerful party to be the receiver. It would be interesting to construct a BC protocol for which the powerful party is the sender (i.e., only the sender is required to to seal and verify envelopes).

References

1. D. Aharonov, A. Ta-Shma, U. V. Vazirani, and A. C. Yao. Quantum bit escrow. In *STOC '00*, pages 705–714, 2000.
2. A. Ambainis, M. Jakobsson, and H. Lipmaa. Cryptographic randomized response techniques. In *PKC '04*, volume 2947 of *LNCS*, pages 425–438, 2004.
3. R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., 2001.
4. M. Blaze. Cryptology and physical security: Rights amplification in master-keyed mechanical locks. *IEEE Security and Privacy*, March 2003.
5. M. Blaze. Safecracking for the computer scientist. *U. Penn CIS Department Technical Report*, December 2004. http://www.crypto.com/papers/safelocks.pdf.
6. M. Blum. Coin flipping over the telephone. In *Proceedings of IEEE COMPCON '82*, pages 133–137, 1982.
7. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.
8. R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *STOC '86*, pages 364–369, 1986.
9. R. Cleve and R. Impagliazzo. Martingales, collective coin flipping and discrete control processes. http://www.cpsc.ucalgary.ca/ cleve/pubs/martingales.ps, 1993.
10. C. Crépeau. Efficient cryptographic protocols based on noisy channels. In *Eurocrypt '97*, volume 1233 of *LNCS*, pages 306–317, 1997.
11. C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *FOCS '88*, pages 42–52, 1988.
12. I. B. Damgård, S. Fehr, K. Morozov, and L. Salvail. Unfair noisy channels and oblivious transfer. In *TCC '04*, volume 2951 of *LNCS*, pages 355–373, 2004.
13. I. B. Damgård, J. Kilian, and L. Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Eurocrypt '99*, volume 1592 of *LNCS*, pages 56–73, 1999.
14. R. Fagin, M. Naor, and P. Winkler. Comparing information without leaking it. *Commun. ACM*, 39(5):77–85, 1996.
15. S. Flannery and D. Flannery. *In Code: A Mathematical Journey*. Algonquin Books of Chapel Hill, 2002.
16. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. of the ACM*, 38(3):691–729, July 1991.
17. R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *FOCS '89*, pages 230–235, 1989.
18. H.-K. Lo and H. F. Chau. Why quantum bit commitment and ideal quantum coin tossing are impossible. In *PhysComp '98*, pages 177–187, 1998.
19. D. Mayers. Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.*, (78):3414–3417, 1997.
20. M. Naor, Y. Naor, and O. Reingold. Applied kid cryptography or how to convince your children you are not cheating, Mar. 1999. http://www.wisdom.weizmann.ac.il/ naor/PAPERS/waldo.ps.
21. B. Schneier. The solitaire encryption algorithm, 1999. http://www.schneier.com/solitaire.html.