Several Weak Bit-Commitments Using Seal-Once Tamper-Evident Devices*

Ioana Boureanu and Serge Vaudenay

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
{ioana.boureanu,serge.vaudenay}@epfl.ch

Abstract. Following both theoretical and practical arguments, we construct UC-secure bit-commitment protocols that place their strength on the sender's side and are built using tamper-evident devices, e.g., a type of distinguishable, sealed envelopes. We show that by using a second formalisation of tamper-evident distinguishable envelopes we can attain better security guarantees, i.e., EUC-security. We show the relations between several flavours of weak bit-commitments, bit-commitments and distinguishable tamper-evident envelopes. We focus, at all points, on the lightweight nature of the underlying mechanisms and on the end-to-end human verifiability.

1 Introduction

Most of the recent approaches to primitive-construction employ the universal composability (UC) framework [6] in order to specify and prove the correctness/security of their cryptographic designs. The UC framework is a formalism that allows for cryptographic protocols to be computationally analysed in a single session, yet the security guarantees thereby obtained are preserved when multiple sessions are composed concurrently, in parallel and/or sequentially. In [6], Canetti shows that any polynomial-time multi-party functionality is feasible in the UC framework if the majority of participants are honest. Otherwise, feasibility is usually attained if the models are augmented with "setup-assumptions", obtaining the so-called "UC hybrid models" (i.e., extra ideal functionalities are made available to the parties).

UC-formalisations of tamper-evident/tamper-resistant hardware devices have been used as setups to UC-realize different cryptographic primitives, from bit-commitment to polling schemes [13,15,12,16,17,19,18]; the tamper-evidence of a device implies that, if tampered with, the device will signal the inflicted abnormalities, whereas tamper-resistance denotes the impossibility of tampering with the device. Tamper-evidence-based UC-secure protocols [16,17,19] also bear lightweight, humanly constructible/verifiable cryptographic mechanisms. To realize UC-secure weak bit-commitment (WBC) protocols, a type of distinguishable tamper-evident envelopes were shown sufficient and necessary (in the sense

* Full version of this paper: [3].

T. Takagi et al. (Eds.): ProvSec 2012, LNCS 7496, pp. 70–87, 2012.

that simpler functionalities of distinguishable tamper-evident containers are not sufficient to realize bit-commitments) [18]. The protocols thereby constructed placed their "strength" on the receiver's side, i.e., it is the receiver who creates the tamper-evident devices or prepares them. In [18], Moran and Naor raise the question of finding such lightweight, UC-secure (weak) bit-commitment protocols that in turn place their strength on the sender's side, i.e, *sender-strong* protocols. Along similar lines, Brassard, Chaum and Crépeau in foundation papers have long made the question: "Is it preferable to trust Vic or Peggy? We do not know, but it sure is nice to have the choice. [4]".

Contributions. The contributions of this paper are as follows:

- We create weak bit-commitments that place the (adversarial) strength on the committer side, i.e., sender-strong WBC, and that are UC-secure. To achieve this, we require a new formalisation of distinguishable envelopes and use it as a UC setup functionality (see the motivations below).
- We describe a hierarchy of ideal functionalities for sender-strong weak bit-commitments and UC-realize them. In this, we relate better with the existing literature in the field (see Section 2.2 for details).
- We relate our first functionality of distinguishable envelopes ($\mathcal{F}^{DE}_{\texttt{OneSeal}}$), the standard UC-functionality of bit-commitment ($\mathcal{F}^{BC}$) and those of WBC (already existing and newly introduced herein), showing most implication-relations between them.
- We introduce a second distinguishable envelope functionality ($\mathcal{F}^{\texttt{purpoted}DE}_{\texttt{OneSeal}}$), which allows for the corresponding DE-based WBC protocols herein and the ones in [18] to be enjoyed a stronger security notion: be not only UC-secure, but also EUC-secure.

Motivation for Our Formalisation of Tamper-Evident Envelopes. As Moran et al. state in [18], there are many ways to formalise tamper-evident containers, reflecting the different requirements of the possible physical implementations of such devices. The *sole* motivation given in [18] for allowing creator-forgeability is the desiderata of creating more complex, somewhat stronger protocols. But, when it comes to placing this sort of asymmetric strength on the sender's side, it only makes sense to construct commitment protocols that are, in the standard sense, *computationally hiding* and somewhat binding, i.e., the receiver is powerless and the sender can possibly equivocate his commitments. (By contrast, in [18] are both *partially* hiding and *partially* binding and are then amplified.) In this context, we conjecture that it is not possible to be based only on tamper-evident envelopes à la Moran et al. [18] and construct *hiding* sender-strong bit-commitment protocols, which would further be UC-secure in the same time. To overcome this shortcoming, we have herein slightly modified the original, tamper-evident envelope functionality from [18], preventing the creator from resealing envelopes. Hence, we model *seal-once* distinguishable tamper-evident envelopes (or, envelope allowing *one-seal* only). By contrast, the functionality in [18] formalises a *multi-seal* distinguishable tamper-evident envelope.

Furthermore, the previous protocols designed using tamper-evident envelopes à la Moran et al. [18] were only UC-secure and not EUC-secure. We noted that if we relaxed the forging abilities of the envelope-creator in the aforementioned way and we furthermore allow for purported destinator for envelopes the corresponding DE-based protocols obtained both here and in [18] attain EUC-security and not only UC-security.

Our Weak Bit-Commitments from a theoretical viewpoint. Alongside the UC-framework, sender-strong weak bit-commitments are also interesting by traditional theoretical lines, where they are easier to construct (see Section 3.4). In [4], outside of the UC-framework, Brassard et al. proved that the existence of "chameleon" bit-commitments[1] implies the existence of zero-knowledge (ZK) proofs of knowledge which were MA-protocols (i.e., where the verifier sends independent bits). Moreover, in [2] Beaver proved that in order for the aforementioned ZK proofs of knowledge (PoK) to be provable secure against adaptive adversaries, the chameleon bit-commitments (BC) are not enough, but content-equivocable bit commitments are needed (i.e., the equivocation is possible only if a record of the traffic between the sender and the receiver is available to the sender and not other types of witnesses, like parts of messages). One of our weak BC functionalities, $\mathcal{F}^{q-\mathrm{WBC}}_{\mathtt{LearnAtOpening}}$, models this last type of important weak bit-commitments.

Our Weak Sender-Strong *Bit-Commitments from a practical viewpoint.* A real-life situation were the committer/sender should be given the chance to "change his mind" is the case in negotiation-based protocols where the receiver is known or thought to be corrupt (e.g, hostage-release cases, reputations [1], anonymous special auctioning [21], etc.).

To sum up, we are motivated to present certain means of attaining different UC and EUC-secure, bit-commitment protocols placing their strength on the sender's side, i.e., *sender-strong (SS).*

Related Work. A series of works on designing UC-secure protocols using tamper-resistant building blocks have recently emerged [13,8,15,12,19]. For example, the formalism by Katz, in [13], opens for the creation and exchange of tamper-proof hardware tokens used in a commitment protocol, which is UC-secure if the tokens are stateful and the DDH assumption holds. In [8], the two-party computation can equally be UC-realized, but the model is relaxed: the tokens are stateless and the assumption is switched to the existence of oblivious transfer protocols in the UC plain model. Similar results are obtained using tamper-resistant devices as building blocks in a model called the trusted agent model [15]. Like in [8] and unlike in [13], Mateus and Vaudenay [15] permit a freer flow of devices from their creator to their users and backwards. Similar protocols are constructed by Moran et al., in [19], using tamper-resistant hardware tokens that can be passed

[1] These are commitments where the sender could cheat at the decommitment phase if given extra information.

in one direction only. We note that the distinction of having UC-commitments which place the strength on the sender or, on the contrary, place their strength on the receiver has also been underlined [19] within this context of using tamper-resistant hardware as UC-setup.

Simpler cryptographic protocols UC-constructed using not tamper-resistant devices, but tamper-evident devices in form of sealed envelopes and sealed locks have been studied in [18,16,17]. All the protocols thereby presented place their strength on the receiver's side.

2 Setup and Target UC Functionalities

2.1 UC-Setup Functionalities Modelling Tamper-Evident Envelopes

The $\mathcal{F}_{\texttt{OneSeal}}^{DE}$ Functionality. In general, a functionality for tamper-evidence stores a table of envelopes, indexed by their unique *id*. More precisely, an entry in this table is of the form (*id*, *value*, *holder*, *state*). The values in one entry indexed by *id* are respectively denoted $value_{id}$, $holder_{id}$ and $state_{id}$.

In particular, the functionality $\mathcal{F}_{\texttt{OneSeal}}^{DE}$ models a tamper-evident "envelope", distinguishable by some obvious mark (e.g., barcode, serial number, colour, etc.). Protocol parties can simply open such containers, but any such opening will be obvious to other parties who receive the "torn" envelope. The $\mathcal{F}_{\texttt{OneSeal}}^{DE}$ ideal functionality, running in the presence of parties $P_1, \ldots, P_n$ and an ideal adversary $\mathcal{I}$ is described in the following.

Seal(*id*, *value*). Let this command be received from party P_i. It creates and seals an envelope. If this is the first **Seal** message with id *id*, the functionality stores the tuple (*id*, *value*, P_i, **sealed**) in the table. If this is not the first command of type **Seal** for envelope *id*, then the functionality halts.

Send(*id*, P_j). Let this command be received from party P_i. This command encodes the sending of an envelope held by P_i to a party P_j. Upon receiving this command from party P_i, the functionality verifies that there is an entry in its table which is indexed by *id* and has $holder_{id} = P_i$. If so, it outputs (**Receipt**, *id*, P_i, P_j) to P_j and $\mathcal{I}$ and replaces the entry in the table with (*id*, $value_{id}$, P_j, $state_{id}$).

Open *id*. Let this command be received from party P_i. This command encodes an envelope being opened by the party that currently holds it. Upon receiving this command, the functionality verifies that an entry for container *id* appears in the table and that $holder_{id} = P_i$. If so, it sends (**Opened**, *id*, $value_{id}$) to P_i and $\mathcal{I}$. It also replaces the entry in the table with (*id*, $value_{id}$, $holder_{id}$, **broken**).

Verify *id*. Let this command be received from party P_i. This command denotes P_i's verification of whether or not the seal on an envelope has been broken. The functionality verifies that an entry indexed by *id* appears in the table and that $holder_{id} = P_i$. It sends (**Verified**, *id*, $state_{id}$) to P_i and to $\mathcal{I}$.

One of the differences from the corresponding functionality presented in [18] is that the one introduced above does not output tuples containing the creator's

identity. This would have been of no interest for the protocols constructed in the following and would hinder EUC-security proofs given herein. However, a more important difference is that the creator of an envelope cannot re-seal it, i.e., he cannot forge the value stored initially inside the envelope. Hence, we use the syntagm "`OneSeal`" to refer to the functionality herein and, sometimes, we use the expression "`MultiSeal`" to designate the tamper-evident envelopes in [18]. This modification is driven by the fact that we could not yet prove or disprove the existence of a sender-strong, somewhat binding and *not partially hiding, but computationally hiding* bit commitment *that is also simulatable within UC*, using only creator-forgeable/multi-seal tamper-evident envelopes.

It is relatively easy to see that *regular* bit-commitments can be immediately constructed using one distinguishable tamper-evident envelope, see Section 4. The relation with the regular commitment functionality is however not symmetric, as will detail (i.e., if $\mathcal{F}^{DE}_{\texttt{OneSeal}}$ implies BC, it is not necessarily the case that $\mathcal{F}^{BC}_{\texttt{OneSeal}}$ implies DE).

The tamper-evident envelope functionality in [18] is denoted as $\mathcal{F}^{DE}_{\texttt{MultiSeal}}$.

2.2 Target UC Functionalities of Bit-Commitment

We now describe our target functionalities $\mathcal{F}^{q-\mathrm{WBC}}_{\star}$ that model different weak bit-commitment (WBC) protocols, where

$$\star \in \{\texttt{EscapeThenMayCheat}, \texttt{LearnAtCommitment}, \texttt{LearnAtOpening}\}.$$

In this fashion, we can relate the WBCs UC-realized herein both with traditional weak bit-commitments [2] of theoretical importance (e.g., see our $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$), and with weak bit-commitments UC-created in [18] with distinguishable envelopes (see our $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$). The differences between these functionalities lie mainly in learning that equivocation is possible (yet not obligatory) at the commitment phase ($\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtCommitment}}$) or the opening phase ($\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$) vs. cheating only when the committer has not yet been caught abusing the protocol ($\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$).

The $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$ functionality idealising q-weak bit-commitment. Let $q \in (0,1)$. The functionality maintains a variable *bit*, where *bit* ranges over $\{0,1,\square\}$.

Commit b. When the **Commit** b command ($b \in \{0,1\}$) is sent to the functionality by a sender S, the value b is recorded in the variable *bit*. The functionality of $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$ outputs **Committed** to the receiver R and to the ideal adversary $\mathcal{I}$[2]. Further commands of this type or of type **EquivocatoryCommit** below are ignored by the functionality.

EquivocatoryCommit. When the **EquivocatoryCommit** command is sent to the functionality, the $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$ functionality replies to the sender and

[2] Throughout, the fact that the output is sent to the ideal adversary as well is inherent to the UC framework, i.e., see the UC-notion of "delayed output".

the ideal adversary with a $\perp$ message, with probability $1 - q$. With probability q, the functionality sets the variable *bit* to the value $\square$, outputs **Committed** to the sender, the receiver and to the ideal adversary. Further commands of this type or of type **Commit** above are ignored by the functionality.

AbortCommit. When the **AbortCommit** command is sent to the functionality, the $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$ functionality replies to the sender, to the receiver, and to the ideal adversary with a $\perp$ message (denoting an abnormal end of the execution). Further commands are ignored.

Open. Upon receiving the command **Open** from the sender, the functionality verifies that the sender has already sent the **Commit** b command. Then, the $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$ functionality outputs (**Opened**, *bit*) to the receiver and to the ideal adversary. Further commands are ignored by the functionality.

EquivocatoryOpen c. Upon receiving this command from the sender, with $c \in \{0, 1\}$, the functionality verifies that $bit = \square$. Then, the functionality $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{EscapeThenMayCheat}}$ outputs (**Opened**, c) to the receiver and to the ideal adversary. Further commands are ignored by the functionality.

In this functionality, the binding property of commitments can be defied. It corresponds to the weak bit-commitment functionality used by Moran and Naor [18], but it applies to the sender-strong case. In that sense, a dishonest player decides to try and open his commitment to any value even from the very beginning of the protocol and he can be successful in doing so with a probability of $q \in (0, 1)$, once he has not been caught red-handed.

Note that the WBC functionality presented above and the ones to be presented further model single bit commitments. Yet, they can easily be extended to respective functionalities for multiple commitments: i.e., each **Commit** b command sent by a sender S aimed at a receiver R would become **Commit**(id, b, R) and each corresponding functionality would, for each commitment, store a tuple $(id, sender, receiver, value)$ doing the respective checks.

The $\mathcal{F}^{q-\mathrm{WBC}}_{\textit{LearnAtCommitment}}$ *functionality idealising q-weak bit-commitment.* Let $q \in (0, 1)$. The functionality maintains a tuple (*bit*, *equiv*), where *bit* ranges over $\{0, 1\}$ and *equiv* ranges over {"Yes", "No"}.

Commit b. When the **Commit** b command ($b \in \{0, 1\}$) is sent to the functionality, the value b is recorded in the variable *bit*. With probability q the value "Yes" is stored in *equiv* or, with probability $1 - q$ the value "No" is stored in *equiv*. The $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtCommitment}}$ functionality outputs **Committed** to the receiver and to the ideal adversary. The $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtCommitment}}$ functionality outputs the updated value of *equiv* to the sender and to the ideal adversary. Further commands of this type are ignored by the functionality.

Open. Upon receiving this command, the functionality verifies that the sender has already sent the **Commit** b command. Then, the $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtCommitment}}$ functionality outputs (**Opened**, *bit*) to the receiver and to the ideal adversary. Further commands are ignored by the functionality.

EquivocatoryOpen. Upon receiving this command, the functionality verifies that the sender has already sent the **Commit** b command. Then, the functionality checks the value of *equiv*. If the value is "Yes", then $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtCommitment}}$ outputs (**Opened**, $\overline{bit}$) to the receiver and to the ideal adversary. If the value is "No", then $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtCommitment}}$ halts. Further commands are ignored by the functionality.

The $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtCommitment}}$ functionality mirrors a protocol which allows the sender to cheat by breaking the binding property of the protocol. Note that this cheating possibility is "decided" at the commitment phase, i.e., it is at some point during the commitment phase that the potential cheater *learns* about his opportunity. Also, note that while the cheating is allowed, it does not necessarily need to happen (i.e., there are two distinct opening commands).

Next, we give a similar functionality where in turn the possibility of equivocation becomes clear only at the opening phase.

The $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$ *functionality idealising q-weak bit-commitment.* Let $q \in (0,1)$. The functionality maintains a variable *bit*, ranging over $\{0,1\}$.

Commit. When the **Commit** b command ($b \in \{0,1\}$) is sent to the functionality, the value b is recorded in the variable *bit*. The $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$ functionality outputs **Committed** to the receiver and to the ideal adversary. Further commands of this type are ignored by the functionality.

Open. Upon receiving this command, the functionality verifies that the sender has already sent the **Commit** b command. Then, the $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$ functionality outputs (**Opened**, *bit*) to the receiver and to the ideal adversary. Further commands are ignored by the functionality.

EquivocatoryOpen. Upon receiving this command, the functionality verifies that the sender has already sent the **Commit** b command. With probability q, the $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$ outputs (**Opened**, $\overline{bit}$) to the receiver and to the ideal adversary. With probability $1-q$, the $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$ sends $\perp$ to the sender S and the ideal adversary $\mathcal{I}$. Further commands of this type are ignored by the functionality (but commands of type **Open** are still allowed).

The $\mathcal{F}^{q-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$ functionality mirrors a protocol which allows the sender to cheat by breaking the binding property of the protocol, knowingly at some point during the opening phase, i.e., i.e., it is at some point during the opening phase that the potential cheater *learns* about his opportunity, similarly to traditional lines in [2]. As aforementioned, note that while the cheating is allowed, it does not necessarily need to happen.

Note that sender-strong weak bit-commitments protocols with distinguishable, tamper-evident envelopes that allow only partial hiding are of course easier to UC-construct than those that require perfect hiding. Amplifications techniques could then be applied. However, we take the view that once the protocols that we seek are sender-strong, UC-realizing a functionality which is only partially hiding is would contradict the aim for the senders' strength (hence, the "*computationally hiding*" UC functionalities that we have given above).

3 UC (Sender-Strong) Bit-Commitments

Driven by the theoretical and practical motivations presented in the introduction, we are now going to present protocols that UC-implement the weak bit-commitment (WBC) functionalities above, use the herein introduced functionalities of distinguishable envelopes as the UC-setup and place their strength on the senders' sides.

We start with the protocol `Pass&MayCheat` which UC-realizes the functionality $\mathcal{F}^{\frac{1}{2}-\text{WBC}}_{\texttt{EscapeThenMayCheat}}$ using the $\mathcal{F}^{DE}_{\texttt{OneSeal}}$. We continue the protocols `CommitEnablesCheat` and `OpenEnablesCheat` which respectively UC-realize the functionalities of $\mathcal{F}^{\frac{2}{3}-\text{WBC}}_{\texttt{LearnAtCommitment}}$ and $\mathcal{F}^{\frac{2}{3}-\text{WBC}}_{\texttt{LearnAtOpening}}$, using the $\mathcal{F}^{DE}_{\texttt{OneSeal}}$. We then present amplification techniques of such weak BC protocols. The techniques maintain the lightweight character of the constructions. We conclude the section by a strengthening of the $\mathcal{F}^{DE}_{\texttt{OneSeal}}$ functionality such that we attain EUC-security [7], i.e., not only UC-security.

3.1 The `Pass&MayCheat` Protocol

The Commitment Phase.

1. A sender S seals four envelopes and creates two pairs out of them such that each pair contains the set $\{x, x\}$ of values, for a random $x \in \{0, 1\}$. Each pair "contains" its own value x. He sends two envelopes, one from each pair, to the receiver R. (E.g., the pairs are $pair_1 = (E_1, E_2)$, $pair_2 = (E_3, E_4)$ and S sends, e.g., E_1, E_3 to R).
2. The receiver R stores the identifiers of the envelopes in a register W. (I.e., it stores $(1, 3)$, given the illustrated execution by S above.). Then, R sends them back without opening them.
3. The sender S verifies that the recently returned envelopes have the seals unbroken. If this is not so, he halts. Otherwise, he sends the two envelopes not sent before. (I.e., If seals are unbroken, then S sends the remaining E_2, E_4.)
4. The receiver verifies that the envelopes received do not have the ids stored already. If they do, he halts. Otherwise, he opens one of these envelopes, sends back the other one without opening it, together with the value of an *id* stored already in W to request back one envelope. The receiver also stores the ids of the envelopes seen this time round. (I.e., R opens, e.g., E_2, sends back E_4 and, e.g., 1, thus requesting back envelope E_1.) *Given the steps of the protocol so far, note that the opened envelope together with the requested one form an initial pair. Also, once the sender has sent this requested envelope, the sender will be left with the other of the initial pairs at his end. These comments equally apply to the equivocatorial commitment phase to follow.*
5. The sender S verifies that the recently returned envelope has the seal unbroken. If this is not so, he halts. Otherwise, he sends the one requested envelope to R. He also sends the value $d = b \oplus x$, where b is the bit he is

committing to and x is the bit hidden inside each envelope in the pair to be found at his side. (I.e., If the seal is unbroken, then S sends the requested E_1, $d = b \oplus x$, where x is in E_3 and/or in E_4).
6. The receiver R opens the last envelope received and checks that the value at its side are equal. If not, he aborts. (I.e., If E_1 and E_2 do not contain the same value, then R aborts).

The EquivocatoryCommitment Phase.

1. A sender S seals four envelopes and creates two pairs out of them such that one pair contains the set $\{x, x\}$ of values, for a random $x \in \{0, 1\}$ and the other pair contains the values $\{0, 1\}$. He sends two envelopes, one from each pair, to the receiver R. (E.g., The pairs are $pair_1 = (E_1, E_2)$, $pair_2 = (E_3, E_4)$ and S sends E_1, E_3 to R).
2. Same as in the commitment phase.
3. Same as in the commitment phase.
4. Same as in the commitment phase.
5. Same as in the commitment phase.
6. Same as in the commitment phase.

Let A denote the pair of envelopes to be found at this stage on the sender side.

The Opening Phase.

1. The sender S sends one envelope E_k in the remaining pair, i.e., $E_k \in A$ or $k \in \{i, j\}$.
2. The receiver R checks that E_k is in the set A (by checking the ids). If so, he opens the envelope E_k to find the value b_k hidden inside and then he sets the commitment-bit b' to $d \oplus b_k$. Otherwise, the receiver halts.

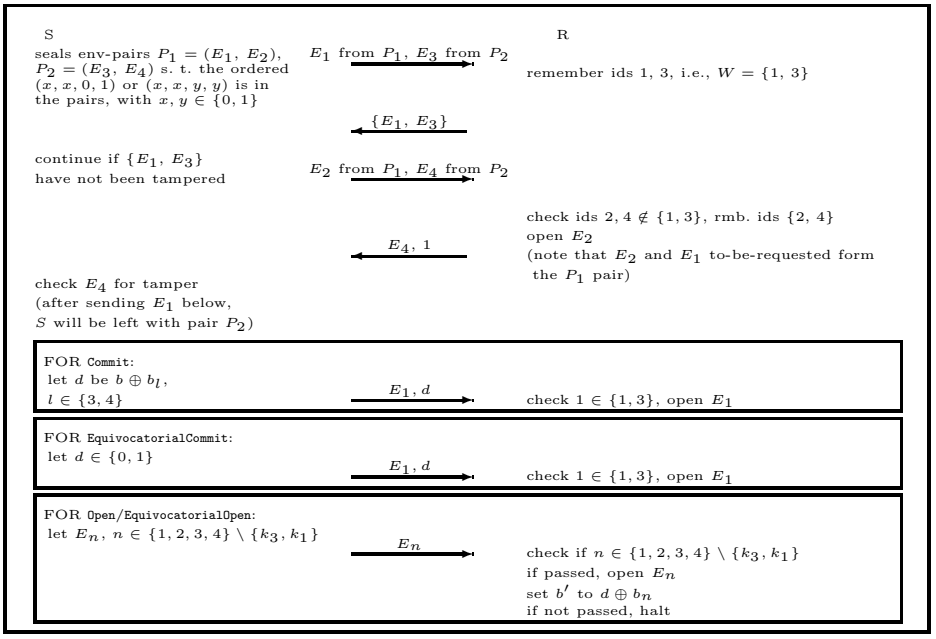
The EquivocatoryOpening Phase.

1. The sender S sends from the remaining pair A the envelope E_k that contains the bit $d \oplus c$, where c is the bit that the sender wants to open to.
2. Same as in the opening phase.

In the following figure, we give an illustration of the protocol above, in a symmetric way (i.e., E_1 and E_2 could be interchanged in their appearances, etc.).

Explanations on the `Pass&MayCheat` *protocol.* Assume first that the sender S creates pairs of envelopes such that each contains the set $\{x, x\}$ of values and that the sender respects the calculation of d for the non-equivocable case. It is clear that at the end of the protocol, the sender has no choice but to open to the correct bit. If, in turn, S does not form d as specified and R does follow the protocol, then S may not be able to open.

Assume now that the sender S creates pairs of envelopes such that one contains the set $\{x, x\}$ of values and the other contains the set $\{0, 1\}$ of values. Depending



on the choice of R to open envelopes, the sender may continue the protocol; clearly this is possible in half of the cases (the possibility that a randomly chosen bit x is equal either to 0 or to 1, depending which was the opening of R). In such cases, S can clearly open the value d to any bit-value, since he is left with x and its negation $\overline{x}$ in the pair A at his end.

Note that the receiver R cannot cheat without being caught: i.e., torn envelopes are obvious to the sender and opening by R of more than two envelopes –to try and break the hiding property– is not possible due to the stage-by-stage unsealing enforced by the protocol.

Also, note that the envelopes used within are seal-once envelopes. Thus, the sender S is not able to change the values x stored inside the envelopes, after step 4 of the commitment phase (say, in order to avoid being caught by R).

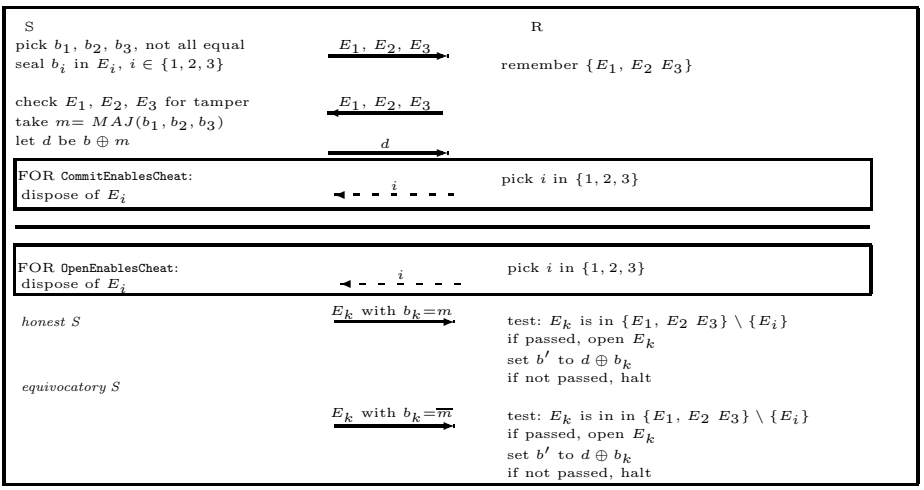
Theorem 1. *In a hybrid UC-model, where the setup is the* $\mathcal{F}_{\mathit{OneSeal}}^{DE}$ *functionality, the* `Pass&MayCheat` *protocol UC-realizes the* $\mathcal{F}_{\mathit{EscapeThenMayCheat}}^{\frac{1}{2}-\mathrm{WBC}}$ *functionality.*

Due to space constraints, the proof of Theorem 1 is given in the the full version of this paper [3].

3.2 The `CommitEnablesCheat` and `OpenEnablesCheat` Protocols

The `CommitEnablesCheat` Protocol

The Commitment Phase. The sender wants to commit to a bit b and proceeds as it follows.



1. The sender S creates 3 sealed envelopes denoted E_1, E_2, E_3 respectively containing the bits denoted b_1, b_2, b_3, such that not all bits are equal. The sender sends the envelopes over to the receiver R.
2. The receiver memorises the set $\{E_1, E_2, E_3\}$ of envelopes and sends them back to the sender
3. The sender verifies that the envelopes sent back are untampered with. Then, he computes m as the majority of the bits sealed inside, i.e., he computes $m = MAJ(b_1, b_2, b_3)$. The sender wants to commit to a bit b. He calculates $d = b \oplus m$. Then, the sender sends d to the receiver.
4. The receiver sends the identifier i of an envelope that the sender should dispose of, i.e., $i \in \{1, 2, 3\}$. Let the set $S = \{E_1, E_2, E_3\} \setminus \{E_i\}$ denote the set of remaining envelopes.
5. The sender disposes of envelope i. (Note that after this the sender can equivocate if the remaining envelopes contain different bits.)

The *equiv* value is $\frac{2}{3}$.

The Opening Phase.

1. The non-equivoquing sender sends an envelope E_k such that $b_k = m$.
   The equivoquing sender sends an envelope E_k such that $b_k = \overline{m}$.
2. The receiver tests that $E_k \in S$ and if so, he sets b', the commitment bit, as follows: $b' = d \oplus b_k$. If the test fails, the receiver halts.

Note that by being asked to discard[3] an envelope at the opening phase instead of in step 4 of the commitment phase, the idea behind protocol `CommitEnablesCheat` can be shaped to obtain a protocol where the equivocation becomes clear only at

[3] A possible way of implementing discarding is sending the emptied envelope back to the receiver.

the opening time. The protocol obtained in this way is hereby denoted `OpenEnablesCheat`. The protocols `CommitEnablesCheat` and `OpenEnablesCheat` are graphically represented in the previous figure.

Note once more that, unlike in the `Pass&MayCheat` protocol, in `CommitEnablesCheat` and `OpenEnablesCheat` protocols, the committer can cheat with some probability (i.e., $\frac{2}{3}$), yet this is not influenced by him being caught cheating, but rather by a mere choice of the receiver.

These requirements sound similar to looking for a means in which Alice would commit to a bit b using a BSC (binary symmetric channel) with noise level q [5]. Nevertheless, the existing solutions [5,10,20] to problems of the latter kind are receiver-strong, not sender-strong. Also, they are not constructed to be UC-secure, but secure by classical lines, which may be weaker. Moreover, those original constructions involve error-correction codes and/or pseudo-random generators being manipulated by the participants. Thus, those primitives are also beyond our cryptographically lightweight scope. Therefore, to obtain senders' strength, UC-security, simplicity and human operability we have proposed protocols `CommitEnablesCheat` and `OpenEnablesCheat` above.

Explanations on the `CommitEnablesCheat` and `OpenEnablesCheat` protocols. We detail on the `CommitEnablesCheat` protocol above, the explanations on `OpenEnablesCheat` being very similar and immediately following. Let us consider the case where the parties follow the protocol. We can see that if S prepares the envelopes correctly (i.e., they contain a permutation of $\{x, x, \overline{x}\}$, $x \in_U \{0, 1\}$) and he adheres to step 2, then at step 3, the value $m = x$. No matter what value b_i has (i.e., x or $\overline{x}$), in the set S of remaining envelopes there is always an envelope E_k with the value x inside that opens the commitment correctly. With probability $\frac{2}{3}$, the set S still contains an envelope with value $\overline{x}$. In this last case, S could open his commitment to the flipped bit (i.e., point 2 in the opening phase). By the above, the protocol is complete. One can see that the case where S does not follows the protocol in terms of envelope sealing does not bring him any benefit. In Theorem 2, we formalise the above explanations, in the context of the UC framework.

Theorem 2. *In a hybrid UC-model, where the setup is the $\mathcal{F}_{OneSeal}^{DE}$ functionality, the `CommitEnablesCheat` and `OpenEnablesCheat` protocols UC-realize the $\mathcal{F}_{LearnAtCommitment}^{\frac{2}{3}-\mathrm{WBC}}$ and the $\mathcal{F}_{LearnAtOpening}^{\frac{2}{3}-\mathrm{WBC}}$ functionalities, respectively.*

Due to space constraints, this proof is given in the full version of this paper [3].

3.3 Amplifying q-WBC Sender-Strong Protocols

Let ✠ ∈ {`Pass&MayCheat`, `CommitEnablesCheat`, `OpenEnablesCheat`}.
Let ⋆ ∈ {`EscapeThenMayCheat`, `LearnAtCommitment`, `LearnAtOpening`}.

By using k instances of a q-weak sender-strong protocol of the ✠-kind of protocols, we can obtain a protocol `Amplified_`✠ protocol that UC-realizes $\mathcal{F}_{\star}^{q^k-WBC}$. Hence, we can attain regular bit-commitments for a conveniently large k. See the formalisations below.

The `Amplified_Pass&MayCheat` Protocol:

(Equivocatory) Commitment Phase. The sender commits, all equivocally or all normally, to a bit b_j in k sequential rounds, each time using the $\mathcal{F}^{q-WBC}_{\texttt{EscapeThenMayCheat}}$ functionality, $j \in \{1, \ldots, k\}$. The j-th such functionality is denoted $\mathcal{F}^{q-WBC}_{\texttt{EscapeThenMayCheat};\, j}$.

Each functionality $\mathcal{F}^{q-WBC}_{\texttt{EscapeThenMayCheat};\, j}$ to which **EquivocatoryCommit** was sent, outputs to its sender `Committed`, with probability q and $\perp$ otherwise. If $\perp$ is sent, then the receiver aborts.

(Equivocatory) Opening Phase. The sender opens (equivocally or not) all commitments using the functionalities of $\mathcal{F}^{q-WBC}_{\texttt{EscapeThenMayCheat};\, j}$. The receiver halts if the openings are not all the same.

Theorem 3. *Let $q \in (0,1)$ and λ be a security parameter. By using $k = \Omega(\lambda)$ instances of an $\mathcal{F}^{q-WBC}_{\star}$ functionality, we can construct a protocol `Amplified_`$\maltese$ that UC-realizes the $\mathcal{F}^{BC}$ functionality, where*
$\star \in \{$`EscapeThenMayCheat`, `LearnAtCommitment`, `LearnAtOpening`$\}$ and
$\maltese \in \{$Pass&MayCheat, CommitEnablesCheat, OpenEnablesCheat$\}$.

In particular, the protocol `Amplified_Pass&MayCheat` UC-realizes the functionality of $\mathcal{F}^{q^k-WBC}_{\texttt{EscapeThenMayCheat}}$.

For the regular BC functionality, $\mathcal{F}^{BC}$, see the full version of this paper [3]. The `Amplified_Pass&MayCheat` BC protocol is trivially following out of `Amplified_Pass&MayCheat`, i.e., where equivocation is not possible. By letting $k = \frac{\log \varepsilon}{\log q}$ in Theorem 3, we make `Amplified_Pass&MayCheat` a ε-WBC, with ε arbitrarily close to 0. However, for `Amplified_Pass&MayCheat` to UC-realize $\mathcal{F}^{BC}$, we need a k to be of linear-size in the security parameter λ. Proofs that weak bit-commitment protocols in the above sense can be amplified to regular bit-commitments exist already, e.g., [20]. The proofs therein follow long-established lines, i.e., not the UC framework [4]. Also, they often refer to receiver-strong protocols and generally use more convoluted primitives, e.g., pseudo-random generators, error-correcting codes, outside our lightweight interests. Our proof is done in the UC framework and, as we can see, the protocol respects the sender-strong aspects sought-after herein. Due to space constraints, the actual proof of Theorem 3 is given in the full version of this paper [3].

3.4 (Stronger) Universally Composable Security

A UC-oriented note is that something as little as the order of the messages in the commitment-phase of the weak protocol `CommitEnablesCheat` above and/or the amount of randomness given to the sender does impact the UC-simulatability. A protocol only different from `CommitEnablesCheat` in that it inverts the order of events 3 and 4 in the commitment phase does not UC-realize the $\mathcal{F}^{\frac{2}{3}-\text{WBC}}_{\texttt{LearnAtCommitment}}$

[4] Similar proofs of amplifications may exist in the UC framework, however they would not be with respect to the $\mathcal{F}^{q-WBC}_{i}$ functionalities as introduced in Section 2.

functionality, while it is perfectly hiding and binding with probability $\frac{2}{3}$ in the classical sense. Thus, it seems to be easier to construct q-weak bit-commitments using just a formalisation of distinguishable envelopes, but when sender-strength and UC-security are both sought after subtle difficulties arise ($q \in (0, 1)$).

Another important UC note is that the protocols above (and in fact all weak bit-commitment protocols constructed previously for the receiver-strong case in Moran and Noar's work [18]) are not secure in stronger versions of the UC framework, e.g., GUC (Generalised UC) or EUC (externalised UC) [7]. For a wrap-up on GUC (Generalised UC) or EUC (externalised UC), see the full version of this paper [3]. To support this claim, it is enough to show that the protocols are not secure in the EUC framework. So, in an EUC model with the $\mathcal{F}_{\texttt{OneSeal}}^{DE}$-setup consider an environment that prepares the envelopes and feeds them to the adversary. It is clear that the ideal adversary cannot "extract" the bit b to commit to and thus he cannot indistinguishably simulate the commitment phase.

Lemma 4. *In a hybrid EUC-model, where the setup is the $\mathcal{F}_{\texttt{OneSeal}}^{DE}$ functionality, the `CommitEnablesCheat` protocol does not EUC-realizes the $\mathcal{F}_{\texttt{LearnAtCommitment}}^{\frac{2}{3}-\text{WBC}}$ functionality.*

Due to space constraints, the proof of this lemma is given in the full version of this paper [3].

As aforementioned, along very similar lines the receiver-strong protocols in previous works [18] are not EUC-secure either. We modify the $\mathcal{F}_{\texttt{OneSeal}}^{DE}$ functionality slightly such that when used as a setup, we attain EUC-security of the protocols herein and those WBC protocols in Moran and Noar's work [18].

$\mathcal{F}_{\texttt{OneSeal}}^{\texttt{purpotedDE}}$: A Stronger Functionality for Tamper-Evident Distinguishable Sealed Envelopes. This functionality stores tuples of the form (*id*, *value*, *holder*, *state*). The values in one entry indexed with *id*, like before.

SealSend(*id*, *value*, P_j). Let this command be received from an envelope-creator party P_i. It seals an envelope and sends its id to the future holder P_j. If this is the first **Seal** message with id *id*, the functionality stores the tuple (*id*, *value*, P_j, **sealed**) in the table. The functionality sends (*id*, P_i) to P_j and to $\mathcal{I}$. (Optionally, it can send (*id*, *sealed*) to P_i and to $\mathcal{I}$). If this is not the first command of type **Seal** for envelope *id*, then the functionality halts.

Send(*id*, P_j). Let this command be received from a holder-party P_i. This command encodes the sending of an envelope held by P_i to a party P_j. Upon receiving this command from party P_i, the functionality verifies that there is an entry in its table which is indexed by *id* and has $holder_{id} = P_i$. If so, it outputs (**Receipt**, *id*, P_i, P_j) to P_j and $\mathcal{I}$ and replaces the entry in the table with (*id*, $value_{id}$, P_j, $state_{id}$).

Open *id*. Let this command be received from a holder-party P_i. This command encodes an envelop being opened by the party that currently holds it. Upon receiving this command, the functionality verifies that an entry for container *id*

appears in the table and that $holder_{id} = P_i$. If so, it sends (**Opened**, id, $value_{id}$) to P_i and $\mathcal{I}$ and replaces the entry with (id, $value_{id}$, $holder_{id}$, **broken**).

Verify id. Let this command be received from a holder-party P_i. This command denotes P_i's verification of whether or not the seal on an envelope has been broken. The functionality verifies that an entry indexed by id appears in the table and that $holder_{id} = P_i$. It sends (**Verified**, id, $state_{id}$) to P_i and to $\mathcal{I}$.

The main difference between $\mathcal{F}_{\texttt{OneSeal}}^{\texttt{purpoted}DE}$ and the original $\mathcal{F}_{\texttt{OneSeal}}^{DE}$ functionality is that an envelope is created for a specifically intended holder and this holder is consequently notified with a message of the form (id, $creator$). Note that this enhancement is realistic (i.e., if to be used in a protocol, the delivery address of the receiver is to be specified by a manufacturing body). However, note that the functionality does not store or reveal publicly the creators of the envelopes (i.e., that would be a stronger enhancement, akin to signing the tamper-evident devices). With this modification, the holder-to-be knows that a specific envelope has been freshly produced by a specific creator. Intuitively, this prevents the weakness in the proof of Lemma 4 from happening, i.e., R cannot accept envelopes that are not (newly) meant for him. Also, the creator is authenticated by the functionality, in the sense that he cannot use envelopes made by others. In a larger sense, this can prevent relay attacks. More formally, the following holds.

Theorem 5. *In a hybrid EUC-model, where the setup is the* $\mathcal{F}_{\mathit{OneSeal}}^{\mathit{purpotedDE}}$ *functionality, the* `CommitEnablesCheat` *protocol EUC-realizes the* $\mathcal{F}_{\mathit{LearnAtCommitment}}^{\frac{2}{3}-\mathrm{WBC}}$ *functionality.*

The proof of the above theorem follows from the proofs of Theorem 2 and that of Lemma 4, combined with the fact that $\mathcal{F}_{\texttt{OneSeal}}^{\texttt{purpoted}DE}$-envelopes have a specified entity as their destination and this entity knows this fact upon the creation of the envelopes. We conjecture that Theorem 5 holds even in the case of adaptive adversaries.

4 Relations between (Weak) Bit-Commitments and Distinguishable Envelopes in UC

Given the results above and those in [18], we have that $\mathcal{F}_{\texttt{OneSeal}}^{DE}$ (or $\mathcal{F}_{\texttt{OneSeal}}^{\texttt{purpoted}DE}$) can create receiver-strong and sender-strong weak UC bit-commitments, which in turn can be amplified to obtain regular UC bit-commitments. The $\mathcal{F}^{BC}$ functionality or a flavour of it (see $\mathcal{F}_{COM}$ in [6]) constitutes a sufficient setup to UC-realize a ZK protocol [6]. Under these circumstances, it is definitely interesting to investigate the existent implications between different sort of weak bit-commitment, regular bit-commitment and tamper-evident envelopes, in the UC framework.

Firstly, note that a multiple commitment $\mathcal{F}_{MCOM}$ [6] setup suffices to UC-realize an $\mathcal{F}_{\texttt{EscapeThenMayCheat}}^{q-\mathrm{WBC}}$, $\mathcal{F}_{\texttt{LearnAtCommitment}}^{q-\mathrm{WBC}}$ or an $\mathcal{F}_{\texttt{LearnAtOpening}}^{q-\mathrm{WBC}}$ functionality, for some $q \in (0, 1)$. (We recall the $\mathcal{F}_{MCOM}$ [6] functionality in the full version of this paper [3]). In other words, several instances of the regular bit-commitment

functionality $\mathcal{F}^{BC}$ suffice to UC-construct a sender-strong weak bit-commitment. In particular, three regular bit-commitment $\mathcal{F}^{BC}$ functionalities (see the full version of this paper [3]) UC-construct a $\frac{2}{3}$-WBC which is sender-strong. Let a protocol $\mathcal{P}$ be obtained from protocol `CommitEnablesCheat` where the creation and transmission of the three envelopes is respectively replaced by creation and transmission of three commitments using $\mathcal{F}_{MCOM}$ or using three respective instances of $\mathcal{F}^{BC}$. An analogous fact holds also for $\mathcal{F}^{\frac{2}{3}-\mathrm{WBC}}_{\texttt{LearnAtOpening}}$, where to construct the protocol $\mathcal{P}$ we use `OpenEnablesCheat` instead of `CommitEnablesCheat`.

Secondly, as a consequence of Theorem 3, note that all sender-strong weak BCs UC-imply regular BCs.

Thirdly, it should be answered whether bit-commitment setup suffice to UC-realize the $\mathcal{F}^{DE}_{\texttt{OneSeal}}$ distinguishable tamper-evident envelope functionality. We conjecture that question 1 has a negative answer. This is intuitively due to the fact that in bit-commitments it is the sender who opens the commitment and, in an envelope-emulating protocol, it should be the envelope's creator who needs to perform the corresponding opening. But, in order for this to be generally possible, an envelope creator ought to know the current envelope holder, which is not the case in our formalisations (i.e., envelopes can be passed on from holder to holder, without the notification of the creator).

To sum up, amplification proofs considered, we have completed the picture of UC-realisability of different flavours of sender-strong weak BC with tamper-evident envelopes and of their relation to (almost) regular BC and receiver-strong weak BCs by Moran and Naor [18]. To some level, we can say that all weak BCs are equivalent to regular BCs. We leave the EUC or the GUC correspondents of the implications enumerated above as open questions.

5 Conclusions

Answering a variant of the open question in Moran and Naor's work [18] and several practical needs [9,14,11], we conclude that simple, sealed envelopes can also create sender-strong (weak) bit-commitments protocols. In the process, we have also discussed the fact that the protocols in [18] are not EUC-secure but only UC-secure. We mainly focused on creating sender-strong bit-commitments with the same level of security. Nevertheless, we showed how to modify the $\mathcal{F}^{DE}_{\texttt{OneSeal}}$ functionality given in Moran and Naor's work [18] such that we also create (weak) bit-commitment protocols that are EUC-secure. We showed lightweight amplification proofs of our WBC protocols. We lastly discussed some of the implications between UC weak BCs, UC regular BCs and distinguishable tamper-evident UC envelopes. The interest in *weak* BC protocol per se was motivated by both theoretical and practical reasons. The GUC-security of our schemes remains to be discussed.

Acknowledgements. The authors acknowledge the support of the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a centre supported by the Swiss National Science Foundation.

References

1. Awerbuch, B., Patt-Shamir, B., Peleg, D., Tuttle, M.: Collaboration of Untrusting Peers with Changing Interests. In: Proceedings of the 5th ACM Conference on Electronic Commerce, EC 2004, pp. 112–119. ACM, New York (2004)
2. Beaver, D.: Adaptive Zero Knowledge and Computational Equivocation (Extended Abstract). In: The 28th Annual ACM Symposium on Theory of Computing (STOC), pp. 629–638 (1996)
3. Boureanu, I., Vaudenay, S.: Several weak bit-commitments using seal-once tamper-evident devices. Cryptology ePrint Archive, Report 2012/380 (2012), http://eprint.iacr.org/2012/380
4. Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. Journal of Computer Systems Science 37, 156–189 (1988)
5. Crépeau, C.: Efficient Cryptographic Protocols Based on Noisy Channels. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 306–317. Springer, Heidelberg (1997)
6. Canetti, R.: A Unified Framework for Analyzing Security of Protocols. Electronic Colloquium on Computational Complexity (ECCC) 8(16) (2001)
7. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally Composable Security with Global Setup. Cryptology ePrint Archive, Report 2006/432 (2006), http://eprint.iacr.org/
8. Chandran, N., Goyal, V., Sahai, A.: New Constructions for UC Secure Computation Using Tamper-Proof Hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)
9. Chin-Chen, C., Ya-Fen, C.: Efficient Anonymous Auction Protocols with Freewheeling Bids. Computers & Security 22(8), 728–734 (2003)
10. Damgård, I.: On the Existence of Bit Commitment Schemes and Zero-Knowledge Proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 17–27. Springer, Heidelberg (1990)
11. Dane, G.: The Implementation of an Auction Protocol over Anonymous Networks (2000), http://research.microsoft.com/en-us/um/people/gdane/papers/partiiproj-anonauctions.pdf
12. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding Cryptography on Tamper-Proof Hardware Tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010)
13. Katz, J.: Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
14. Kikuchi, H., Harkavy, M., Tygar, J.D.: Multi-round Anonymous Auction Protocols. In: Proceedings of the 1st IEEE Workshop on Dependable and Real-Time E-Commerce Systems, pp. 62–69. Springer (1998)
15. Mateus, P., Vaudenay, S.: On Tamper-Resistance from a Theoretical Viewpoint. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 411–428. Springer, Heidelberg (2009)
16. Moran, T., Naor, M.: Basing Cryptographic Protocols on Tamper-Evident Seals. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 285–297. Springer, Heidelberg (2005)

17. Moran, T., Naor, M.: Polling with Physical Envelopes: A Rigorous Analysis of a Human-Centric Protocol. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 88–108. Springer, Heidelberg (2006)
18. Moran, T., Naor, M.: Basing Cryptographic Protocols on Tamper-Evident Seals. Theoretical Computer Science 411, 1283–1310 (2010)
19. Moran, T., Segev, G.: David and Goliath Commitments: UC Computation for Asymmetric Parties Using Tamper-Proof Hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008)
20. Naor, M.: Bit Commitment Using Pseudo-Randomness. Journal of Cryptology 4, 151–158 (1991)
21. Stajano, F., Anderson, R.: The Cocaine Auction Protocol: On the Power of Anonymous Broadcast. In: Pfitzmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 434–447. Springer, Heidelberg (2000)