Modelling Sealed Containers in Z

Eerke Boiten[1] and Jeremy Jacob[2]

[1] Centre for Cyber Security & School of Computing
University of Kent, UK
[2] Department of Computer Science, University of York, UK

Abstract. Physical means of securing information, such as sealed envelopes and scratch cards, can be used to achieve cryptographic objectives, including ones that are hard or impossible by electronic means. So far, descriptions of such mechanisms have been informal, and as a consequence so has the associated reasoning.
This paper[3] takes a formal methods approach to these physical cryptography primitives. A model of distinguishable sealed envelopes is given in Z, exploring some of the design decisions and avenues for further analysis and development of such models.

1 Introduction

Physical mechanisms for securing information such as sealed envelopes and scratch cards have powerful properties already in their simplest possible forms. They contain information, which remains hidden until an explicit moment where the seal is broken or the card is scratched. Up to that point, anyone not involved in manufacturing the item can plausibly argue ignorance of the information. More complex forms include multiple scratchable areas on a single card (as used in lottery games), and overprinted scratch cards [21].

Reasoning about such mechanisms so far has been done only informally. Applications such as in lotteries and games are probably simple enough for this to suffice. However, it has also been shown that these mechanisms can be used in voting protocols [1,21], polling protocols [19], and most fundamentally: to implement general cryptographic schemes [18], including bit commitment and oblivious transfer. In such schemes, scratch cards and sealed envelopes play a much more intricate role, and the notions of security are sophisticated. At the most abstract ("possibilistic") level, this involves reasoning about information flow and confidentiality. A more detailed analysis will also have to consider explicit probabilistic and computational complexity based aspects [5].

This paper embarks on the formal modelling and analysis of these mechanisms. The physical mechanism considered will be sealed distinguishable

[3] Version of 28 March 2014 – incorporating ABZ 2014 reviewers' comments before abbreviation to 6 pages.

envelopes containing a single *bit* of information. This will be applied in a number of protocols for *bit commitment*, leading up to the one given in Moran and Naor's paper [18]. The emphasis will be for this paper on constructing the model of the envelopes and the protocols based on it, and looking ahead to the semantic requirements for using such models in systematic analysis and formal development.

Section 2 describes a model for distinguishable sealed envelopes in Z. The problem of bit commitment, and how its modelling and analysis stretches the standard formal methods toolbox, is described in Section 3. We specify and discuss a number of bit commitment protocols based on envelopes in Section 4. Finally, Section 5 reflects on observational semantics and refinement, and sets out an agenda for advancing this work.

2 Modelling Sealed Envelopes in Z

This is a story about *Agent*s who pass *Envelope*s about:

$$[Agent, Envelope]$$

The presence of these two as given types represents relevant information, namely that both of these have "identities". For *Agent*s this will show in particular agents holding specific and different roles in protocols; for *Envelope*s it means that different envelopes can be distinguished. Moran and Naor [18] describe two different models of envelopes, of which we are covering the more powerful *distinguishable* envelope model.

The development of this specification has been supported with the Z-Eves proof tool [22], in particular syntax-checking, type-checking and applicability checking have been applied to definitions and theorems.[4] Most of the theorems' proofs have not been discharged yet.

2.1 The State

Envelopes contain bits, and may be: uncreated; created and closed; or created and open. The value in a created and closed envelope may only be known by its creator; the value in an open envelope is known by anyone who possesses it. A created envelope is in the possession of exactly one agent in any state. Thus, envelopes which have not been created are exactly those not held by anyone.

We model an agent knowing the content of an envelope by a pair of relations *zero* and *one*. The predicate $a \mapsto e \in zero$ encodes that agent a has direct evidence (it created the envelope, or has seen it when open) that e contains a 0-bit; and similarly for *one*.

[4] Note to reviewers: this comment was fully correct for the version four days before submission, and is intended to be fully correct again for any published version. At the time of submission, some revised and additional definitions and theorems are included that have not been checked, yet.

At any point in our story we have agents holding envelopes, some of which are open, and agents knowing the contents of some envelopes.

$$
\begin{array}{|l}
\hline
S \\
holder : Envelope \pfun Agent \\
open : \mathbb{F}\, Envelope \\
zero, one : Agent \rel Envelope \\
\hline
open \subseteq \dom holder \\
\ran(zero \cup one) = \dom holder \\
\ran zero \cap \ran one = \emptyset \\
\hline
\end{array}
$$

The predicates state that:

1. all open envelopes are held by some agent;
2. the content of every created envelope is known by some agent (at least its creator); and
3. agents' views of envelope contents are consistent: all agents who know the content of an envelope agree on it.

In this story we do not care about the initial state. An empty initial state is easy to describe if wished.

2.2 Good Operations

Operations on the state —happenings in our story— must be well behaved. We introduce a schema, *OpS*, to capture this.

$$
\begin{array}{|l}
\hline
OpS \\
\Delta S \\
\hline
open \subseteq open' \\
zero \subseteq zero' \\
one \subseteq one' \\
\hline
\end{array}
$$

The predicates state that:

1. envelopes cannot be resealed (which is a crucial property of our model); and
2. agents cannot forget what they know (acknowledging that *zero* and *one* are essentially epistemic predicates).

2.3 Operations on the State

There are three operations, to create, move and open envelopes. Each of these is a bulk operation, in that it applies to sets of envelopes.

We also allow empty sets everywhere, and do not insist that agents are different, so that an agent may 'move' envelopes to itself. These conventions allow slightly simpler formulæ.

Create When an agent $a?$ creates new envelopes it does so for a set of envelopes to store 0-bits, $zs?$, and for a set of 1-bits, $os?$. Of course, either of these could be empty and the other have cardinality 1.

Create

$$
\begin{array}{l}
OpS \\
a? : Agent \\
zs?, os? : \mathbb{F}\, Envelope \\
\hline
(zs? \cup os?) \cap \operatorname{dom} holder = \emptyset \\
zs? \cap os? = \emptyset \\
holder' = holder \cup ((zs? \cup os?) \times \{a?\}) \\
open' = open \\
zero' = zero \cup (\{a?\} \times zs?) \\
one' = one \cup (\{a?\} \times os?)
\end{array}
$$

Note that the predicates record that the new envelopes are new, held by their creator, are closed, and their values are known to their creator only.

Move A set of envelopes (perhaps just one) may be moved to a named agent as long as they have all been created, and are held by one agent. The receiving agent learns the values of any open envelopes. No envelope is opened by this operation.

Move

$$
\begin{array}{l}
OpS \\
b? : Agent \\
es? : \mathbb{F}\, Envelope \\
\hline
es? \subseteq \operatorname{dom} holder \\
\exists\, a : Agent \bullet holder(\!|\, es? \,|\!) = \{a\} \\
holder' = holder \oplus (es? \times \{b?\}) \\
open' = open \\
zero' = zero \cup (\{b?\} \times (es? \cap open \cap \operatorname{ran} zero)) \\
one' = one \cup (\{b?\} \times (es? \cap open \cap \operatorname{ran} one))
\end{array}
$$

Open The holder of a set of envelopes can open them. The holder learns their values, they do not change hands.

Open

OpS

$es? : \mathbb{F}\, Envelope$

$es? \subseteq \operatorname{dom} holder$

$es? \cap open = \emptyset$

$holder' = holder$

$open' = open \cup es?$

$\exists a : Agent \bullet holder(\!|\, es? \,|\!) = \{a\} \land$
$\quad zero' = zero \cup (\{a\} \times (es? \cap \operatorname{ran} zero)) \land$
$\quad one' = one \cup (\{a\} \times (es? \cap \operatorname{ran} one))$

Note that the precondition $es? \cap open = \emptyset$ (none of the envelopes are already open) could be omitted without harm; the operation would then be required to treat re-opening envelopes as a null operation.

2.4 An Agent's View

We can define a schema that reports an agent's view of a state. This is a *finalisation* operation, not commonly used in Z states-and-operations specifications, but a good way of encoding non-standard observations of abstract data types. See Section 5 for further discussion.

View

S

S_0

$b? : Agent$

$holder_0 = holder \rhd \{b?\}$

$open_0 = open \cap \operatorname{dom} holder_0$

$zero_0 = \{b?\} \lhd zero$

$one_0 = \{b?\} \lhd one$

The variables with 0-subscripts are those which represent $b?$'s view of (an instance of) state S.

An agent knows more than this. It also knows what closed envelopes have passed through it without being opened, and the order that envelopes passed through it. It also has partial knowledge of what other agents that it has communicated with know. None of that knowledge is important for the sequel, but could easily be added by new state components in S.

Remark Instead of using a 0-subscript we could have decorated the view variables with a prime ($'$); this has the advantage that we could save a line in the schema definition, but the disadvantage of confusion with state-change.

With a basic model of distinguishable sealed envelopes in place, we now set out the target: a bit commitment protocol.

3 Bit Commitment: a Challenge

Commitment is an essential cryptographic primitive between two parties. (The term carries several meanings in computer science – this one is somewhat different from "commit" actions that occur in database transaction management, and for which "distributed commitment" protocols have been developed [13].) In the simplest case, where the value committed to is a single bit, it works as follows.

One party, usually called the *sender*, executes an action $Commit(b)$ for a given bit value b. Once this has happened, the other party, usually called the *receiver*, knows that the sender has committed to a bit (and in a context of multiple commitments taking place, this commitment also has an identity), but does not know what b is. This prescribed ignorance on the part of the receiver is called the *hiding* property, which a cheating receiver may attempt to break. Once the commitment has been made, this starts a window of time in which the sender has committed to a value which the receiver does not yet know. In e.g. authentication and interactive zero knowledge proof protocols, essential interactions take place inside this window. At some later point in time, the sender can execute an *Open* action, after which the receiver finds out b. The value returned on opening should be the same value originally committed to – this is called the *binding* property, which a cheating sender may attempt to break. A surrounding protocol building on commitment would typically be aborted by the receiver if they discover foul play at this point.

The obvious attempt to implement commitment using envelopes runs as follows. On committing, the sender (here $a?$) creates an envelope with a bit in. Let's say the sender passes the envelope directly to the receiver (here $b?$).

$$Commit \mathrel{\widehat=} [Create;\ es? : \mathbb{F}\, Envelope \mid es? = zs? \cup os? \land \#es? = 1] \fatsemi Move$$

Note that we have introduced a name ($es?$) into *Create* to describe part of the link between it and the following *Move*.[5]

Now we can specialise to the two cases of committing to a zero-bit and to a one-bit, and the recipient's view after both:

$$CommitZero \mathrel{\widehat=} [Commit \mid os? = \emptyset]$$

$$CommitOne \mathrel{\widehat=} [Commit \mid zs? = \emptyset]$$

$$CommitZeroView \mathrel{\widehat=} \exists\, es?, zs?, os? : \mathbb{F}\, Envelope \bullet CommitZero \fatsemi View$$

$$CommitOneView \mathrel{\widehat=} \exists\, es?, zs?, os? : \mathbb{F}\, Envelope \bullet CommitOne \fatsemi View$$

[5] This is the first piece of Z in this specification where the Z-Eves theorem prover requires —a minimal amount of— help to discharge applicability proof obligations.

Of course, $b?$ cannot tell the difference between *CommitZero* and *CommitOne*:[6]

Theorem 1 (Commit_Indistinguishability).

$$\forall a?, b? : Agent \mid a? \neq b? \bullet \theta CommitZeroView = \theta CommitOneView$$

This theorem captures the hiding property.

Remark It is necessary to introduce the schemas *CommitZeroView* and *CommitOneView* as the θ construction can only be applied to a schema name, but not to an arbitrary schema expression. The schemas *CommitZero* and *CommitOne* are worth introducing as system-level descriptions of the two types of bit-commitment, independently of the vagaries of Z.

3.1 Discovering the value of a committed-to bit

The recipient, $b?$, can open the envelope to discover the value of the bit in it.

To state the theorem that after opening the recipient knows the bit committed to, we introduce a view of: the sending of some bit, sending a zero-bit, and sending a one bit. The statement of the theorem encodes the manner by which $b?$'s knowledge allows it to deduce the value of the sent bit.[7]

$$CommitOpenView \mathrel{\widehat=} Commit \fatsemi Open \fatsemi View$$

$$CommitZeroOpenView \mathrel{\widehat=} CommitZero \fatsemi Open \fatsemi View$$

$$CommitOneOpenView \mathrel{\widehat=} CommitOne \fatsemi Open \fatsemi View$$

Theorem 2 (Opened_Distinguishability).

$$\begin{array}{l} \forall CommitOpenView \bullet \\ (es? \subseteq \operatorname{ran} zero_0 \setminus \operatorname{ran} one_0 \Rightarrow CommitZeroOpenView) \\ \land \\ (es? \subseteq \operatorname{ran} one_0 \setminus \operatorname{ran} zero_0 \Rightarrow CommitOneOpenView) \end{array}$$

Remark The inclusion of the *View* finalisation is necessary to ensure there is enough information for the type-checker. The schema names, *CommitOpenView*, and so on, cannot be replaced by their definitions as they are used in contexts where schema references, but not schema expressions are allowed.

However, the above does not yet correctly implement a commitment scheme. Crucially, *Open* is enabled as soon as the commitment has been made. Thus, it

[6] Z-Eves discharges this by one application of tactic "prove".

[7] Z-Eves cannot dismiss this using the tactic "prove by reduce". It needs further guidance, for example to expand operators such as $\lhd$. This we have not done.

does not satisfy the requirement that the sender controls *when* the receiver is allowed to learn the value of the commitment.

Even a solution where the sender would be able to *observe* when the receiver opens the commitment prematurely would be useful, as the sender could then abort the overarching protocol on making that observation. However, the assumption that sender and receiver are always in the same room and can always see what the other party is doing is too strong – we want security to be guaranteed by the sealing of envelopes, not by constant observation.

Removing the *Move* action from the commitment step also would not solve the problem, as this would break the *binding* property, as the sender could postpone the choice of bit and envelope until that moment without the receiver's noticing[8].

In the physical world, as in the cryptographic one, a standard solution to this is to assume a trusted third party, to which we can hand the envelope (or the bit) for safekeeping between committing and opening. This is a correct implementation, but somewhat circular, as the cryptographic machinery for establishing the trusted third party, and secure communication with it, is typically built on commitments again (they are for example commonly used in zero knowledge proof protocols [10]).

The exploration of bit commitment as a two party electronic protocol, with the normal assumptions of a fixed number of messages of bounded size, has led to a large number of negative results. First, it is fairly simple to see that it is impossible to achieve both perfect hiding and perfect binding – this is explained in detail elsewhere [4]. Even quantum computing will not change that [16]. As a consequence, the highest attainable notion of security should be a "computational" one. Indeed either perfect binding or hiding has been achieved in concrete constructions, with in each case the chance of breaking the other property negligibly small (in terms of a security parameter such as key length). It has been proved, however, that a version achieving the important compositionality property of "Universal Composability" [8] cannot be constructed unless the parties have some common knowledge to start with [9]. All this makes commitment an interesting challenge for formal methods, requiring approximate notions of correctness in the face of an "obvious ideal" specification which is unsatisfiable.

4 Envelope Based Commitment Protocols

4.1 Achieving Probabilistic Security

An envelope-based protocol that is more resistant against cheating needs three additional enhancements. First, the bit in the envelope needs to be masked with a "random" bit r. (In this paper, we do not continue the analysis onto

[8] The argument that this could be fixed by linking the bit to the envelope's identity and transferring the identity on committing is attractive, but leads straight to the issue of perfect bit commitment being impossible as a two party electronic protocol.

the probabilistic level, so the uniform probabilistic choice involved will be represented by a non-deterministic one instead.) If the sender puts r **xor** $v?$ in the envelope, the receiver will learn nothing about the value of $v?$ by prematurely opening the envelope; instead, the sender will transmit r when they open the commitment, which will allow the receiver to learn the value of $v?$ by taking the **xor** of r with the envelope's contents. In fact, as the value contains no information for anyone who does not know r, it does not even need to be put in an envelope.

Unfortunately this idea is flawed, too. As is typical for commitment schemes, an attempt to stop the receiver cheating against hiding leads to an opportunity for the sender to cheat against binding. There is nothing to stop the sender from transmitting $\neg r$ at opening time and thereby changing the value committed to. What we would really need here is for the sender to commit to r as well, but that makes it a circular construction again!

The way out of this is for the *receiver* to prepare envelopes with random bits for the sender. In the first attempt, the receiver creates one with zero, and one with one, and passes these to the sender. The sender opens a random one, getting x, concludes that the other contains $\neg x$, and takes $r = \neg x$, and transfers $v?$ **xor** r as before. On opening the commitment, the sender passes the closed envelope to the receiver, which should contain r as a proof that the sender did not cheat. Binding has been restored!

Unfortunately hiding is at risk again in this solution: the receiver could send envelopes with two identical bits, thereby learning the committed value prematurely. The sender opens only one envelope so can never find this out.

In fixing this, probabilistic reasoning really comes to the fore. Instead of creating and sending two envelopes, the receiver creates and sends a balanced collection of *four* envelopes. The sender opens three of these, and expects to find two zeros and a one, or vice versa – if not, the receiver's cheating has been detected. This still allows a fair amount of cheating to be undetected: if the receiver biases the choice by sending (say) three zeroes plus a one, this is detected with a chance of one in four, and where it is undetected gives receiver full knowledge of $v?$ as r will be zero. A chance of detection of one in four seems small, but by repeating this as often as necessary ("amplification"), the chance of successful cheating can be made as small as desired.

4.2 The Commitment Protocol

Summarising, the protocol consists of three communications of envelopes between the two parties, with typically a time gap between the second and the third in which the overarching protocol does its job knowing the commitment is in place.

Before we give a description in Z, we present it in the traditional protocol notation, in which we need two enhancements:

– normally a series of values separated by commas is sent and received in that order (concatenated); surrounding $[\![\ ]\!]$ brackets means they are *received* in some non-deterministic order;

– $[x]$ denotes a newly created closed envelope containing the bit x, a new open envelope with x is represented by $\langle x \rangle$.

The commitment protocol then goes as follows.

Preparation: $B \to A : [[[1], [1], [0], [0]]]$
A receives these as $E1 \ldots E4$, opens $E1 \ldots E3$ and takes the exclusive-or of their values returning b.
If $E1 \ldots E3$ all had the same value,A finds B has cheated and aborts the protocol.
The Z specifications below use $es?$ for $E1 \ldots E4$ and $fs?$ for $E1 \ldots E3$.

Commitment: $A \to B : \langle vv \rangle$
The value vv is computed as the exclusive-or of b and the value $v?$ that A wants to commit to. It is sent in an open envelope called $e?$ below.

Opening: $A \to B : E4$
B receives and opens this last closed envelope, the value found should be b, and computes the exclusive-or of b and vv which should be $v?$.
If the envelope received isn't one of the original four created by B, A has cheated and B aborts the protocol (and any surrounding ones).

We retain $a?$ as the sender, and $b?$ as the receiver, in the following descriptions. We will need a few shorthands for envelopes being created and moving in opposite directions from the previous protocol:

$$CreateB \mathrel{\widehat=} Create[b?/a?]$$
$$MoveBA \mathrel{\widehat=} Move[a?/b?]$$

4.3 The Preparation Phase

The receiver creates four envelopes and sends them to the sender. Note that we use here their names within the commitment protocol, with this particular transmission going in the opposite direction.

$$SendFour \mathrel{\widehat=} [CreateB;\ es? : \mathbb{F}\, Envelope \mid es? = zs? \cup os? \land \#es? = 4] \fatsemi MoveBA$$

An honest receiver balances the bits to be sent: exactly two of them are 0-bits, and hence exactly two are 1-bits:

$$Honest \mathrel{\widehat=} [SendFour \mid \#zs? = 2]$$

A dishonest receiver could send all the same or three the same. The protocol ensures that all four envelopes having the same value will be detected. Hence we only model the case where there is a one–three split; the protocol will detect this case with probability $\frac{1}{4}$.

$$Dishonest \mathrel{\widehat=} [SendFour \mid \#zs? = 1 \lor \#os? = 1]$$

Now we can state a couple of theorems: that with an honest receiver the sender knows the value in the unopened envelope, and with a dishonest receiver

the sender may be fooled.[9] We need to distinguish the set of envelopes opened from the set of envelopes sent by the receiver. We introduce *fs*? for the set of envelopes to be opened, using renaming of variables in *Open*.

$$OpenThree \mathrel{\widehat=} [Open[fs?/es?]; \; es? : \mathbb{F}\, Envelope \mid \#fs? = 3 \land fs? \subseteq es?]$$

$$HonestOpenThree \mathrel{\widehat=} Honest \fatsemi OpenThree$$

Theorem 3 (Honest_Receiver).

$$\forall\, HonestOpenThree \bullet \\ (\#(\operatorname{ran} zero \cap fs?) = 1 \Rightarrow es? \setminus fs? \subseteq \operatorname{ran} zero) \\ \land \\ (\#(\operatorname{ran} one \cap fs?) = 1 \Rightarrow es? \setminus fs? \subseteq \operatorname{ran} one)$$

Because the value of the sealed envelope is known to its creator it is in the range of either *zero* or *one*, we use this as a proxy for knowing its value. The phrase $\operatorname{ran} zero \cap fs?$ is the set of newly-opened envelopes that contain a 0-bit, and similarly for $\operatorname{ran} one \cap fs?$ and 1-bits. The sealed envelope is the only member of the set $es? \setminus fs?$, so the phrase $es? \setminus fs? \subseteq \operatorname{ran} zero$ tells us that it holds a 0-bit.

There are two symmetric theorems about dishonest receivers fooling the sender. We only state one, about being fooled into believing that the value in the sealed envelope is a 0-bit when it is actually a 1-bit.

$$DishonestOpenThree \mathrel{\widehat=} Dishonest \fatsemi OpenThree$$

Theorem 4 (Dishonest_Receiver).

$$\exists\, DishonestOpenThree \mid \#(\operatorname{ran} zero \cap fs?) = 1 \bullet es? \setminus fs? \subseteq \operatorname{ran} one$$

4.4 The Commitment Step

After the receiver has prepared and sent the four envelopes, we have already seen that the sender opens three of them. The full action of the sender when they commit to a bit is now as follows. We define shorthands for creating a singleton set of envelopes renamed as *e*?:

$$CreateOne \mathrel{\widehat=} \exists\, zs? : \mathbb{F}\, Envelope \bullet [Create[e?/os?] \mid \#e? = 1 \land \#zs? = 0]$$

$$CreateZero \mathrel{\widehat=} \exists\, os? : \mathbb{F}\, Envelope \bullet [Create[e?/zs?] \mid \#e? = 1 \land \#os? = 0]$$

These single envelopes get opened and sent in the protocol step as follows:

$$SendOne \mathrel{\widehat=} CreateOne \fatsemi Open[e?/es?] \fatsemi Move[e?/es?]$$

$$SendZero \mathrel{\widehat=} CreateZero \fatsemi Open[e?/es?] \fatsemi Move[e?/es?]$$

[9] Both of these theorems require much work to prove in Z-Eves.

Putting this phase together, the decision on which bit to send depends on which bit to commit to (encoded in the operation name) and the masking bit (the bit that is implied to be in the remaining closed envelope, decided from the values of the three envelopes opened): it is the exclusive-or of those two bits.

$$\begin{aligned} CommitToOne \mathrel{\widehat=} OpenThree \fatsemi ([SendOne \mid \#(\operatorname{ran} zero \cap fs?) = 1] \\ \lor [SendZero \mid \#(\operatorname{ran} zero \cap fs?) = 2]) \end{aligned}$$

$$\begin{aligned} CommitToZero \mathrel{\widehat=} OpenThree \fatsemi ([SendZero \mid \#(\operatorname{ran} zero \cap fs?) = 1] \\ \lor [SendOne \mid \#(\operatorname{ran} zero \cap fs?) = 2]) \end{aligned}$$

The situation where the sender finds three identical envelopes on opening (and thus discovers the receiver's cheating in preparing the envelopes) is represented by empty behaviour.

At this point, an honest receiver should not be able to observe a difference between committing to zero and committing to one: the hiding property.

$$CommitZView \mathrel{\widehat=} \exists e?, es?, fs?, zs?, os? \bullet Honest \fatsemi CommitToZero \fatsemi View$$

$$CommitOView \mathrel{\widehat=} \exists e?, es?, fs?, zs?, os? \bullet Honest \fatsemi CommitToOne \fatsemi View$$

Theorem 5 (Commit4_Indistinguishability).

$$\forall a?, b? : Agent \mid a? \neq b? \bullet \theta CommitZView = \theta CommitOView$$

4.5 Opening the Commitment

To open the commitment, the sender sends the remaining closed envelope to the receiver. The receiver finds the masking bit in there and deduces by taking its exclusive-or (i.e. inequality checking) with the bit sent earlier on the bit committed to.

$$MoveLast \mathrel{\widehat=} [es?, fs?, ef? : \mathbb{F}\, Envelope \mid Move[ef?/es?] \land ef? = es?/fs?]$$

$$Same \mathrel{\widehat=} [S'; e?, ef? : \mathbb{F}\, Envelope \mid e? \cup ef? \subseteq \operatorname{ran} zero' \lor e? \cup ef? \subseteq \operatorname{ran} one']$$

$$\begin{aligned} OpenZero &\mathrel{\widehat=} MoveLast \fatsemi ([Open[ef?/es?] \land Same) \\ OpenOne &\mathrel{\widehat=} MoveLast \fatsemi ([Open[ef?/es?] \land \neg Same) \end{aligned}$$

Correctness theorems for this could be given to state that, with a honest receiver, only behaviours where the commitment step and the opening step use the same bit are possible. In fact, it is impossible for the sender to successfully cheat in this particular protocol. This can be stated by demonstrating that alternative sender behaviours (e.g. replacing *CommitToOne* and/or *MoveLast*) do not enable behaviours with mismatching commitments and openings (attacks against the binding property).

5 Conclusions and Further Work

The above specifications reflect our attempt to model sealed envelopes and their application in a bit commitment protocol. In doing so, we used a common Z style of describing state spaces and operations on them. This we did not link yet with any refinement theory. Rather, we took an approach that is more common in formal methods for security: giving (an abstraction of) an implementation, and then stating some security properties that we would like to hold. Clearly the more sophisticated approach would be to state the security properties in an abstract way, and produce the implementation as a (gradual, stepwise) refinement of them [5], for some suitable advanced refinement notion.

Some of our security properties refer directly to the internals of the specification. As well as being less abstract, this may also inhibit reasoning over multiple concurrent instances of protocols. One reason for using Z for this work (as opposed to, say, refinement calculus) is that information hiding is part of its common usage [12,23] – and these sealed containers are all about subtleties of information hiding varying over time. The use of finalisations ("views") in some of the security statements is a first step towards providing a more abstract observational semantics, in line with mechanisms previously explored by both authors in different contexts [15,6,11,7,2]. So far, the views we have used only described the variables visible to an agent. The established theory of information flow recognises that the values of visible variables may also allow for inferences on the values of hidden ones. Morgan's shadow semantics [20] and the related fog semantics [3] by Banks and Jacob would provide a way of addressing this with finalisations – possibly in UTP rather than Z.

A major semantic aspect that also needs to be addressed is the probabilistic one. Most of the non-determinism in the specifications in this paper is an abstraction of probabilistic choice. The full analysis should not only show that the four-envelope protocol allows the receiver to cheat, but also confirm that the probability of cheating being detected is one in four. A promising theoretical framework for integrating this aspect is available. McIver and Morgan [17] have developed refinement theory for probabilistic systems, and together with Hoang et al. they have subsequently achieved some results in linking this to shadow semantics [14]. A related aspect is that protocol descriptions also need to be enhanced to explicitly encode what cheating is, and how this is best modelled in specifications – here, we have modelled some of it in an ad-hoc way. The standard style of protocol description tends to avoid the description of behaviour in error cases altogether.

For work in this area to become usable for modern cryptographic protocols in general, it will also need to address computational aspects. For many problems, the only possible or practicable solutions are ones that have a risk of failure that is non-zero, but negligible as a function of an adversary's computational resources. Analysing and deriving such solutions also requires further enhancement to our formal methods toolbox.

Acknowledgement

This work is supported by the EPSRC Network of Excellence CryptoForma, on formal methods and cryptography (www.cryptoforma.org.uk) with the collaboration between the authors resulting directly from a presentation at the January 2014 network meeting.

References

1. Adida, B., Rivest, R.L.: Scratch & vote: self-contained paper-based cryptographic voting. In: Proceedings of the 5th ACM workshop on Privacy in electronic society. pp. 29–40. ACM (2006)
2. Banks, M.J., Jacob, J.L.: On modelling user observations in the UTP. In: Qin, S. (ed.) UTP. Lecture Notes in Computer Science, vol. 6445, pp. 101–119. Springer (2010)
3. Banks, M.J., Jacob, J.L.: On integrating confidentiality and functionality in a formal method. Formal Asp. Comput. (2013), http://link.springer.com/article/10.1007%2Fs00165-013-0285-4
4. Boiten, E.: Commitment: A challenge for formal methods, http://www.cs.kent.ac.uk/people/staff/eab2/crypto/commit.pdf, unpublished, viewed 27 Jan 2014
5. Boiten, E.: From ABZ to cryptography. In: Börger, E., Butler, M., Bowen, J., Boca, P. (eds.) ABZ: Abstract State Machines, B and Z. Lecture Notes in Computer Science, vol. 5238, p. 353. Springer (2008)
6. Boiten, E., Derrick, J.: Grey box data refinement. In: Grundy, J., Schwenke, M., Vickers, T. (eds.) International Refinement Workshop & Formal Methods Pacific '98. pp. 45–59. Discrete Mathematics and Theoretical Computer Science, Springer-Verlag, Canberra (September 1998)
7. Boiten, E., Derrick, J., Schellhorn, G.: Relational concurrent refinement part II: Internal operations and outputs. Formal Asp. Comput. 21(1–2), 65–102 (2009)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. pp. 136–145. IEEE Computer Society (2001), updated version at http://eprint.iacr.org/2000/067.
9. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2139, pp. 19–40. Springer (2001)
10. Damgård, I., Nielsen, J.: Commitment schemes and zero-knowledge protocols (2011), https://services.brics.dk/java/courseadmin/CPT/documents/getDocument/ComZK08.pdf?d=114251, material for a course on Cryptologic Protocol Theory, Aarhus University, viewed 27 Jan 2014
11. Derrick, J., Boiten, E.: Relational concurrent refinement. Formal Asp. Comput. 15(2), 182–214 (November 2003), http://www.cs.kent.ac.uk/pubs/2003/1751
12. Derrick, J., Boiten, E.: Refinement in Z and Object-Z: Foundations and Advanced Applications. Springer, 2nd edn. (2014), http://www.cs.kent.ac.uk/pubs/2001/1200
13. Fischer, M.: The consensus problem in unreliable distributed systems (a brief survey). In: Karpinski, M. (ed.) FCT. Lecture Notes in Computer Science, vol. 158, pp. 127–140. Springer (1983)
14. Hoang, T.S., McIver, A.K., Meinicke, L., Morgan, C.C., Sloane, A., Susatyo, E.: Abstractions of non-interference security: probabilistic versus possibilistic. Formal Asp. Comput. 26(1), 169–194 (2014)

15. Jacob, J.L.: Refinement of shared systems. In: McDermid, J.A. (ed.) The Theory and Practice of Refinement: Approaches to the Development of Large-Scale Software Systems. pp. 27–36. Butterworths (1989)
16. Mayers, D.: Unconditionally secure quantum bit commitment is impossible. Physical Review Letters 78(17), 3414–3417 (1997)
17. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer (2004)
18. 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. Lecture Notes in Computer Science, vol. 3580, pp. 285–297. Springer (2005)
19. Moran, T., Naor, M.: Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In: Vaudenay, S. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 88–108. Springer (2006)
20. Morgan, C.: The shadow knows: Refinement of ignorance in sequential programs. Sci. Comput. Program. 74(8), 629–653 (2009)
21. Randell, B., Ryan, P.: Voting technologies and trust. IEEE Security and Privacy 4(5), 50–56 (2006)
22. Saaltink, M.: The Z/EVES system. In: Bowen, J.P., Hinchey, M.G., Till, D. (eds.) ZUM. Lecture Notes in Computer Science, vol. 1212, pp. 72–85. Springer (1997)
23. Woodcock, J., Davies, J.: Using Z: Specification, Refinement, and Proof. Prentice Hall (1996)