Sealed Containers in Z

Eerke Boiten[1] and Jeremy Jacob[2]

[1] 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. Reasoning about this has so far been informal.

We give a model of distinguishable sealed envelopes in Z, exploring design decisions and further analysis and development of such models.

1 Introduction

Physical mechanisms for securing information such as sealed envelopes and scratch cards have powerful properties. They contain information, which remains hidden until an explicitly decided moment. Up to then, anyone who has not constructed the item can plausibly argue ignorance. Reasoning about such mechanisms so far has been done informally, which is insufficient for complex applications like voting and polling protocols [9], and most fundamentally: general cryptographic schemes [8]. In such schemes, scratch cards and sealed envelopes play an intricate role, and the notions of security are sophisticated, starting from possibilistic and extending to probabilistic and complexity aspects.

This paper explores the formal modelling and analysis of sealed distinguishable *envelopes* containing a single *bit*, applied in protocols for *bit commitment*[8]. Our emphasis will be on constructing the model and protocols, looking ahead to semantic requirements for systematic analysis and formal development.

2 Modelling Sealed Envelopes in Z

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

$$[Agent, Envelope]$$

Envelopes contain bits, and may be uncreated, closed or open. The value in a 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. We model an agent knowing the content of an envelope by 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*. The predicates below state that all open envelopes are held by some agent, the content of every created envelope is known by some agent, and all agents who know the content

Y. Ait Ameur and K.-D. Schewe (Eds.): ABZ 2014, LNCS 8477, pp. 136–141, 2014.

of an envelope agree on it. Operations on this state need to satisfy the criteria that open envelopes cannot be closed, and agents' knowledge never decreases.

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

There are three operations, to create, move and open sets of envelopes. When an agent $a?$ creates envelopes it does so for a set of envelopes to store 0-bits, $zs?$, and for a set of 1-bits, $os?$ – either may be singleton or empty. New envelopes are held by their creator, are closed, and their values are known to their creator only.

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

A set of envelopes 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.

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

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

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

We 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 models, but a good way of encoding non-standard observations of abstract data types. The variables with 0-subscripts

are those which represent $b?$'s view of (an instance of) state S.

View

$$S;\ S_0;\ b? : Agent$$

$$holder_0 = holder \rhd \{b?\} \ \land\ open_0 = open \cap \mathrm{dom}\, holder_0$$
$$zero_0 = \{b?\} \lhd zero \ \land\ one_0 = \{b?\} \lhd one$$

3 Bit Commitment: A Challenge

Commitment is an essential cryptographic primitive between two parties. In the simplest case, where the value committed to is a single bit, it works as follows.

One party, the *sender*, executes an action $Commit(b)$ for given value b. From then, the *receiver* knows the sender has committed to a bit, but not which. This is the *hiding* property, which a cheating receiver may try to break. The situation now set up is typically exploited in other protocols, e.g. authentication, for further actions. After that, the sender can execute an *Open* action, upon which the receiver finds out b. The value learned should be the same value committed to – this is called the *binding* property, which a cheating sender may try to break. A surrounding protocol would typically be aborted on discovery of foul play.

The obvious but flawed implementation attempt using envelopes just has the sender (here $a?$) passing an envelope with the bit in to the receiver (here $b?$):

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

Now we can specialise to the case of committing to a zero-bit, and the recipient's view afterwards (for one-bit it's the analogous *CommitOneView*), and state and prove that the two cases are indistinguishable ("hiding").

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

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

The recipient, $b?$, can open the envelope to discover the value of the bit in it. We introduce a view for the receiver once that has happened, and a correctness statement for "binding" at this stage:

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

$$\forall\, CommitOpenView \bullet (es? \subseteq \mathrm{ran}\, zero_0 \setminus \mathrm{ran}\, one_0 \Rightarrow CommitZeroOpenView)$$
$$\land\ (es? \subseteq \mathrm{ran}\, one_0 \setminus \mathrm{ran}\, zero_0 \Rightarrow CommitOneOpenView)$$

However, this is not yet a correct implementation of commitment: *Open* is enabled as soon as the commitment has been made. Thus, it does not allow the sender to control *when* the receiver may learn the bit value. Removing the *Move* action from the commitment step also would not solve it, as this would break the *binding* property: the sender

could then postpone the choice of bit. In the physical world, as in the cryptographic one, a sledgehammer 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.

The exploration of bit commitment as a two party protocol, with the normal assumptions of a fixed number of messages of bounded size, has led to multiple negative results. First, it is impossible to achieve both perfect hiding and perfect binding. Approximate, computational, notions of security have been achieved with practical schemes, but the desirable compositionality property of "Universal Composability" is provably unattainable without further assumptions [5]. All this makes commitment a challenge for formal methods, requiring approximate notions of correctness and having an unsatisfiable "obvious ideal" specification.

4 Envelope Based Commitment Protocols

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, uniform probabilistic choice is abstracted to non-deterministic choice.) If the sender puts r **xor** $v?$ in the envelope, opening the envelope will not reveal $v?$. The sender will transmit r to open the commitment, which will allow the receiver to learn $v?$ by cancellation of **xor**. However, this allows the sender to cheat against binding, by transmitting $\neg r$ on opening.

The way out of this is for the *receiver* to prepare envelopes with random bits for the sender. With just two of these, cheating attempts always succeed, but with four it can be detected often enough. The receiver creates these, two with each bit value, and sends them to the sender. He opens three, and expects to find two zeros and a one, or vice versa – if not, the receiver's cheating has been detected. If the receiver biases the choice by sending (say) three zeroes plus a one, this is detected with a chance of one in four which seems small, but can be amplified by repeating the scheme to achieve any required level of security.

Thus, the protocol consists of three communications of envelopes between the parties, with typically a time gap between the second and the third in which the overarching protocol does its job. In traditional protocol notation it is given below, using two bits of extra notation:

- values listed between $[\![\ ]\!]$ brackets are *received* in a non-deterministic order;
- $[x]$ denotes a newly created closed envelope containing the bit x, a new open envelope with x is represented by $\langle x \rangle$.

Preparation: $B \rightarrow A : [\![[1], [1], [0], [0]]\!]$

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.

Commitment: $A \rightarrow B : \langle vv \rangle$

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: $A \rightarrow B : E4$

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).

The Preparation Phase The receiver creates four envelopes and sends them to the sender. An honest receiver balances the bits to be sent: two of each. A smart dishonest receiver sends three plus one (four the same would always be found out). The sender will detect this case with probability $\frac{1}{4}$. With $a?$ as the sender, and $b?$ as the receiver, we need shorthands for envelopes being created and moving in opposite directions from before:

$$CreateB \mathrel{\widehat=} Create[b?/a?] \qquad MoveBA \mathrel{\widehat=} Move[a?/b?]$$
$$SendFour \mathrel{\widehat=} [CreateB;\ es? : \mathbb{F}\, Envelope \mid es? = zs? \cup os? \land \#es? = 4] \fatsemi MoveBA$$
$$Honest \mathrel{\widehat=} [SendFour \mid \#zs? = 2]$$
$$Dishonest \mathrel{\widehat=} [SendFour \mid \#zs? = 1 \lor \#os? = 1]$$

We introduce $fs?$ for the set of envelopes to be opened, through renaming. With an honest receiver the sender knows the value in the unopened envelope:

$$OpenThree \mathrel{\widehat=} [Open[fs?/es?];\ es? : \mathbb{F}\, Envelope \mid \#fs? = 3 \land fs? \subseteq es?]$$
$$HonestOpenThree \mathrel{\widehat=} Honest \fatsemi OpenThree$$
$$\forall\, HonestOpenThree \bullet (\#(\mathrm{ran}\, zero \cap fs?) = 1 \Rightarrow es? \setminus fs? \subseteq \mathrm{ran}\, zero)$$
$$\land\ (\#(\mathrm{ran}\, one \cap fs?) = 1 \Rightarrow es? \setminus fs? \subseteq \mathrm{ran}\, one)$$

The Commitment Step. The relevant definitions (for a zero bit) for this phase are as follows.Details are in the full paper [4].

$$CreateOne \mathrel{\widehat=} \exists\, zs? : \mathbb{F}\, Envelope \bullet [Create[e?/os?] \mid \#e? = 1 \land \#zs? = 0]$$
$$SendOne \mathrel{\widehat=} CreateOne \fatsemi Open[e?/es?] \fatsemi Move[e?/es?]$$
$$CommitToOne \mathrel{\widehat=} OpenThree \fatsemi ([SendOne \mid \#(\mathrm{ran}\, zero \cap fs?) = 1]$$
$$\lor [SendZero \mid \#(\mathrm{ran}\, zero \cap fs?) = 2])$$
$$CommitOView \mathrel{\widehat=} \exists\, e?, es?, fs?, zs?, os? \bullet Honest \fatsemi CommitToOne \fatsemi View$$

Opening the Commitment. The relevant definitions for this phase are as below, with analogues for the one-bit:

$$MoveLast \mathrel{\widehat=} [es?, fs?, ef? : \mathbb{F}\, Envelope \mid Move[ef?/es?] \land ef? = es?/fs?]$$
$$Same \mathrel{\widehat=} [S';\ e?, ef? : \mathbb{F}\, Envelope \mid e? \cup ef? \subseteq \mathrm{ran}\, zero' \lor e? \cup ef? \subseteq \mathrm{ran}\, one']$$
$$OpenZero \mathrel{\widehat=} MoveLast \fatsemi ([Open[ef?/es?] \land Same)$$

5 Discussion and Further Work

We have not considered refinement for these specifications. Rather, we took an approach that is more common in security: using security properties over an abstract implementation. Better would be to state the security properties abstractly, and produce the implementation as a (gradual, stepwise) refinement of them, for a suitable refinement notion.

Some of our properties even refer directly to the specification internals, which is less abstract. The standard Z style is a natural choice for this work, as it considers state hidden, and these sealed containers are all about subtle information hiding varying over time. The use of finalisations ("views") is a first step towards providing a more abstract observational semantics, in line with previous work by both authors [3,1].

A number of additional aspects need to be considered in the refinement relation. Information flow from hidden to visible variables also needs to be incorporated, for example based on Morgan's shadow semantics [10] or the fog semantics [2] by Banks and Jacob. A promising theoretical framework for integrating the missing probabilistic aspect is the theory by McIver, Morgan and others [7,6]. Integrating also the computational complexity aspects inherent from modern cryptography remains an open problem.

Acknowledgement. This work is supported by the EPSRC CryptoForma Network, on formal methods and cryptography (`www.cryptoforma.org.uk`). The specification was developed with support from the Z-Eves proof tool [11].

References

1. Banks, M.J., Jacob, J.L.: On modelling user observations in the UTP. In: Qin, S. (ed.) UTP 2010. LNCS, vol. 6445, pp. 101–119. Springer, Heidelberg (2010)
2. 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`
3. Boiten, E., Derrick, J., Schellhorn, G.: Relational concurrent refinement part II: Internal operations and outputs. Formal Asp. Comput. 21(1-2), 65–102 (2009)
4. Boiten, E., Jacob, J.: Modelling sealed envelopes in Z (2014), `http://www.cs.kent.ac.uk/people/staff/eab2/papers/envlop.pdf`
5. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
6. 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)
7. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer (2004)
8. 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)
9. 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)
10. Morgan, C.: The shadow knows: Refinement of ignorance in sequential programs. Sci. Comput. Program. 74(8), 629–653 (2009)
11. Saaltink, M.: The Z/EVES system. In: Bowen, J.P., Hinchey, M.G., Till, D. (eds.) ZUM 1997. LNCS, vol. 1212, pp. 72–85. Springer, Heidelberg (1997)