Retrenching the Purse: Hashing Injective CLEAR Codes, and Security Properties

Richard Banach
Czeslaw Jeske
School of Computer Science,
University of Manchester,
Manchester M13 9PL, UK
{banach,cj}@cs.man.ac.uk

Michael Poppleton
School of Electronics and
Computer Science,
University of Southampton,
Southampton SO17 1BJ, UK
mrp@ecs.soton.ac.uk

Susan Stepney
Department of Computer Science,
University of York,
York YO10 5DD, UK
susan.stepney@cs.york.ac.uk

***Abstract*— The Mondex Electronic Purse is an outstanding example of industrial scale formal refinement, and was the first verification to achieve ITSEC level E6 certification. A formal abstract model and a formal concrete model were developed, and a formal refinement was hand-proved between them. Nevertheless, certain requirements issues were set beyond the scope of the formal development, or handled in an unnatural manner. The retrenchment *Tower Pattern* is used to address one such issue in detail: the use of a hash function rather than a total injective function when clearing the highly constrained purse logs. A retrenchment is constructed from the lowest level model to a model using a hash, and is then lifted to create two refinement developments, working at different levels of detail, and connected via retrenchments. The tower development is appropriately validated, vindicating the design used.**

I. Introduction

The Mondex Electronic Purse [1], produced by the NatWest Development Team in the mid 1990s, is a system of Smartcard-based electronic purses, carrying currency for use in every normal kind of transaction. Clearly, any electronic purse is a security-critical application. For this reason, the developers of Mondex (formerly a part of NatWest Bank), employed state of the art methods to ensure the implementation was as robust as possible in the face of the most inviting of attacks — ones attempting to forge nonexistent money on the purses. When it was created, the Mondex Purse achieved an ITSEC [2] rating of E6 (nowadays equivalent to a Common Criteria EAL7 rating [3]). ITSEC E6 requires a formal abstract model, a formal concrete model and a proof of correspondence between them. In the case of Mondex this proof of correspondence was a refinement, formally proved to be correct by hand.

The abstract model of the Mondex Purse system describes a world of purses which exchange monetary value through atomic transactions, and specifies the security properties demanded of this world: firstly, and above all, the impossibility of creating nonexistent value; and secondly, the traceable accounting of all value in the system, whether correctly transacted or lost in transit.

The concrete design model describes a system of purses which is distributed, transferring value via an insecure and potentially lossy medium using a three-step protocol. Security features are implemented locally on each purse. In the field the purses must be self-sufficient, defending the integrity of their monetary contents in the face of the most pessimistic assumptions regarding their environment that can still lead to the maintenance of the security properties. Transactions that appear not to be proceeding as required (from an individual purse's point of view) are aborted, and their details logged locally. Central reconciliation of purses' local logs can lead to the retrieval of money genuinely lost in transit, while still defending against fraudulent attempts to create nonexistent value.

Since the purses regard their environment as hostile, every atomic operation that they can perform, must in and of itself, preserve the security invariants. Given this, and the intention to use refinement as the means of achieving correctness, the most straightforward way of assuring the robustness required, is to have each concrete atomic operation be the refinement of some abstract atomic operation; this was the strategy pursued in Mondex.

The separation between the abstract and concrete levels in Mondex is significant, in a logical as well as a functional sense, and it is this separation that contributes in large part to the validation obtainable from the formal proof. Nevertheless, the necessity of having a refinement, taking into account that refinement's proof obligations can be quite demanding in how abstract and concrete models are permitted to be related, meant that a number of requirements issues, legitimately the concern of the formal development, had to be passed over in silence — i.e. set beyond the scope of the formal development, or handled in an unnatural manner — since they would strictly speaking have broken the validity of the refinement had they been incorporated in the models that were used.

Retrenchment [4], [5] was introduced as a framework that weakens and generalizes refinement, essentially in allowing the main refinement operation proof obligation (PO) to be weakened in the postcondition by a *concession*. Things that come within the scope of retrenchment as a result, include the impossibility of refining infinite to finite types, or the continuous variables of real-world physical models (so commonly occurring in the safety-critical applications for which rigorous software techniques are utilised) to discrete ones. As well as such originally envisaged applications [6], retrenchment has also proved useful as a vehicle for the flexible layering in of contrasting, even conflicting requirements in a formal development [7].

If refinement offers strong guarantees of correctness, but is limited as regards the ideal remit of its applicability, and retrenchment forfeits correctness guarantees but is much more widely applicable, then the most profitable strategy would be to employ a judicious combination of the two. One way of doing this is via the *Tower Pattern*, a systematic arrangement of refinements and retrenchments, which allows refinement developments incorporating different but incompatible levels of real-world detail, to be related via suitable retrenchments. In this paper we focus on one of the issues imperfectly covered by refinement in the Mondex development — the logical imperfection of the mechanism for clearing purses' local logs once their contents have been safely archived — and illustrate how the *Tower Pattern* allows for a less unnatural treatment.

The rest of the paper is structured as follows. In section II we give an overview of the Mondex refinement development, and identify the requirements issues that motivate the application of retrenchment, the Mondex 'retrenchment opportunities'. Section III introduces the *Tower Pattern* and how it is applied to Mondex. The tower's constituent notions of refinement and retrenchment are also made precise. Section IV focuses on the hash function for authorising the clearing of purses' logs, the reality that overrides the ideal design using a total injective function assumed in [1]. A retrenchment is built from the lowest level model to a new model featuring the hash. Given these ingredients, Section V overviews how the pieces thus far assembled can be used to complete the building of the rest of the tower for this case study. Section VI gives a validation of the lifted retrenchment, by arguing about the intrinsic security properties of the purse and the protection (both logical and physical) that use of the clear operation enjoys. Section VII pursues this line to further reflect on the Mondex protocol as a whole. Section VIII concludes.

II. The Mondex Purse: Refinements, Retrenchment Opportunities

The Mondex purse formal development [1] consists of three models: A(abstract), B(between), and C(concrete). The A model is a highly abstract expression of atomic value transfer between purses, allowing for an atomic notion of loss in transit. The atomicity makes the security invariants, 'No value created' and 'All value accounted' trivial to prove. It is important to note that the A model is targetted purely at these security properties, which defined the scope of the Mondex formal verification. So it does not address all the many other system requirements. Model B captures the elements of the value transfer protocol, and is thus nonatomic. It is also enhanced with extra structure and constraints needed to achieve a backward refinement from model A; a backward refinement was the strategy used in the Mondex development to get from model A to model B. Model C is model B without the extra structure and constraints. These can be established by an induction on the length of the execution, leading to a forward refinement between models B and C. It is thus shown that model C is a refinement of model A.

Since [1] was aimed at the security properties alone, it is no surprise that many important aspects of Mondex do not get a proper treatment within [1]. Perhaps more surprising is that even taking this into account, some requirements aspects, in principle deserving to be included within the formal development (since they potentially impact on the security properties if mishandled), were nevertheless omitted or handled unnaturally in the modelling, in order to establish the refinement. One of the aims of our work on Mondex, is to show how such rather brittle aspects of formal development via refinement, can be mollified by making use of retrenchment. Not only does this improve the overall quality of the formal development, but it also provides excellent vindication for the retrenchment technique itself, especially when it is used appropriately in tandem with refinement. Here is a brief summary of the Mondex 'retrenchment opportunities'; in this paper we focus in detail on the hash function issue; the other retrenchment opportunities are treated elsewhere.

- Sequence Number: The integrity of the protocol depends partly on the sequence number of the transaction in progress. Sequence numbers occur in the B, C models where they are naturals; in reality they are bounded numbers, but large. See [8].
- Log Full: Transfers completing abnormally are aborted and logged locally by purses. Of course, purses' log contents are vital. Logs occur in the B, C models where they are unbounded; in reality they are finite, and small. See [9].
- Hash Function: The concrete models implement the abstract 'lost value' component in terms of an off-card archive into which purses' log contents are saved. A purse needs to be assured that the data is safely in the archive before it can clear it from its own, highly constrained, log memory. Safe archival is signalled to the purse using a 'clear' code. The purse log contents are assumed to be in total injective correspondence with the clear codes, as that property is required in the proof of the maintenance of the security properties. In reality of course a cryptographic hash function is used, which is neither total, nor injective, but is informally argued to be 'sufficiently injective'.
- Balance Enquiry: Each purse has a balance enquiry operation. If this is invoked in the middle of a B (or C) model transaction, a discrepancy can occur between the model A and model B balances since the model A transaction is atomic and the model B one isn't. This is handled formally by a somewhat unnatural modelling trick; so unnatural in fact, that in [1] balance enquiries were completely omitted. See [10].

III. The Tower Pattern, Refinements and Retrenchments

Mondex was developed via refinement [1], specifically the version for the Z language. Retrenchment is a different but compatible formal technique designed to address the kind of difficulties pure refinement can struggle with. They interact productively in the *Tower Pattern* [8], a grid-like arrangement in which models are connected by refinements in the vertical direction, and by retrenchments in the horizontal one. Moreover, the specific requirements issue dealt with via the retrenchment is largely decoupled from the overall tower structure, underlining its generality and wide utility. The theory permitting the building of the tower is taxing; see [11].

The tower is instantiated for Mondex in Fig. 1, where we see the development of [1] down the left hand side, and retrenchments connecting the A, B, C models to a more detailed refinement development in models F, E, D in the right hand side. Fig. 1 is built bottom up. So the first step is the C to D retrenchment, and this is followed by a lifting of model D to the level of abstraction of B, giving model E. The tower is finally completed with model F. This last step turns out to be easier to do in an ad hoc manner, and in fact, there is more than one sensible option, as discussed at the end of Section V.

$$\begin{tikzcd}
\textbf{A-bstract} \arrow[r, "\text{id}"] \arrow[d, "\text{ref}"'] & \textbf{F-iltered} \arrow[d, "\text{ref}"] \\
\textbf{B-etween} \arrow[r, "\text{ret}"] \arrow[d, "\text{ref}"'] \arrow[dr, dotted, "\text{ref ; ret}"'] & \textbf{E-levated} \arrow[d, "\text{ref}"] \\
\textbf{C-oncrete} \arrow[r, "\text{ret}"'] & \textbf{D-iscrete}
\end{tikzcd}$$

Fig. 1. The *Tower Pattern* applied to Mondex.

We now review the notions of refinement and retrenchment we need. We just give the forward rules for both refinement and retrenchment, since these are sufficient for the BCDE part of the tower.

Let model B be given by the ADT $(B, BInit, \{BOp, BI_{Op}, BO_{Op} \mid Op \in \mathsf{Ops}\})$, and let model C be given by the ADT $(C, CInit, \{COp, CI_{Op}, CO_{Op} \mid Op \in \mathsf{Ops}\})$. So schemas B, C give the abstract and concrete state spaces respectively for the forward refinement from B to C, and the corresponding per-operation I/O spaces are given by schemas BI_{Op}, BO_{Op} and CI_{Op}, CO_{Op}. We assume a retrieve relation $R_{BC} : [B; C]$ between the two state spaces, and for each operation Op, input and output mapping relations $RI_{BC,Op} : [BI_{Op}; CI_{Op}]$ and $RO_{BC,Op} : [BO_{Op}; CO_{Op}]$. Forward refinement is given by three proof obligations (POs), *initialization*, *applicability* and *correctness*:

$$\forall C' \bullet CInit \Rightarrow \exists B' \bullet BInit \wedge R'_{BC} \tag{1}$$

$$\begin{aligned}&\forall B;\ BI_{Op};\ C;\ CI_{Op} \bullet R_{BC} \wedge RI_{BC,Op} \wedge \text{pre}\, BOp \\ &\quad \Rightarrow \text{pre}\, COp\end{aligned} \tag{2}$$

$$\begin{aligned}&\forall B;\ BI_{Op};\ C;\ CI_{Op};\ C';\ CO_{Op} \bullet R_{BC} \wedge RI_{BC,Op} \wedge \text{pre}\, BOp \wedge \\ &\quad COp \Rightarrow \exists B';\ BO_{Op} \bullet BOp \wedge R'_{BC} \wedge RO_{BC,Op}\end{aligned} \tag{3}$$

Note that (1)-(3) do not mention finalisation. We deal with the issue of observation 'on the fly', in line with the tack taken in retrenchment. Also, applicability turns out to be a trivial matter in Mondex since all operations are always enabled by having a 'skip' option.

The C to D development is a forward retrenchment. For this, the abstract model is the C ADT, and the concrete model is given by ADT $(D, DInit, \{(DOp, DI_{Op}, DO_{Op}) \mid Op \in \mathsf{Ops}\})$. Similar notational conventions apply as before. The retrenchment is given by firstly a *retrieve* relation $R_{CD} : [C; D]$ between the state spaces; and secondly by the within, output and concedes relations on a per-operation basis. The *within* relation is between the input-state spaces $W_{CD,Op} : [CI_{Op}; C; DI_{Op}; D]$. The *output* and *concedes* relations are normally defined over both full input-state-output frames with types $O_{CD,Op}; C_{CD,Op} : [CI_{Op}; C; C'; CO_{Op}; DI_{Op}; D; D'; DO_{Op}]$, though in practice, we often omit such parts of these signatures as are not needed. We call these three relations the *retrenchment data*.

Two POs define a retrenchment between two models: *initialisation* as for refinement (1), and *correctness* which is analogous to refinement correctness (3); note that applicability issues are understood to be subsumed in (5) via the within relation:

$$\forall D' \bullet DInit \Rightarrow \exists C' \bullet CInit \wedge R'_{CD} \tag{4}$$

$$\begin{aligned}&\forall C;\ CI_{Op};\ D;\ DI_{Op};\ D';\ DO_{Op} \bullet R_{CD} \wedge W_{CD,Op} \wedge DOp \\ &\quad \Rightarrow \exists C';\ CO_{Op} \bullet COp \wedge ((R'_{CD} \wedge O_{CD,Op}) \vee C_{CD,Op})\end{aligned} \tag{5}$$

IV. The Hash Retrenchment

When a user has made one or more erroneous transactions using the purse, he/she may want to recover the value lost. The only way to do this, is to have the purse log contents centrally archived and reconciled. If the archived log records of the two purses participating in a failed transaction are consistent in the appropriate way, then the bank has sufficient information to ascertain that the money has been lost in flight.[1] This provides one motivation for the need to archive and clear purse logs; another is the severely limited capacity of the logs themselves. Given the size of smartcard nonvolatile memory and the size of (production level) log entries, there is in reality only room for about 5 entries in a purse log.

The archiving works in two stages. In the first, purses send individual log entries to the archive (at the archive's behest), while still retaining them in the log. When this is finished, in the second, the archive sends a CLEAR code to the purse to signal that it is now safe to wipe the log. The proof of correctness for the latter demands that there is an injective function from all possible log contents to CLEAR codes, so that the purse is in no doubt that the correct log contents have been authorised for clearance. In reality, a hash is used, leading

[1]The actual relationship between purses' log contents and the overall system's security properties is rather complex, and is beyond the scope of this paper.

to a hypothetical security loophole. The object of this paper is to examine this using retrenchment.

In principle, the clearing of an individual purse's log is a purse-specific operation. However the impact of doing it imperfectly is system-wide, since it is the system-wide security invariants that are at stake. Therefore our treatment will examine first the clearing operation for an individual purse and its promotion into a system-level operation (in both model C and model D). Then we will relate the two system-level models via retrenchment.

We turn our attention to the purse-specific clearing operation of [1], *ClearException-LogPurseEafromOkay*; let us abbreviate this to *CClearPurseOkay* in this paper.[2]

The purse state schema is *CConPurse*, whose only component to change in operation *CClearPurseOkay* is the log *CexLog*. The remainder of the state, described by schema *CConPurseClear* (which is just *CConPurse* but with *CexLog* hidden), remains unchanged during the operation; this being described using the Ξ convention of Z.

$$\begin{array}{l}
\mathit{CConPurseClear} == \\
\quad \mathit{CConPurse} \setminus (\mathit{CexLog})
\end{array}$$

Clearing the log is only permitted if the purse is in its idle state *CeaFrom*, and if the correct message (a *CexceptionLogClear* message) has been received. This should correctly name the purse (*Cname*), and contain the correct CLEAR code (*Cimage CexLog*). The purse outputs a message ($\perp$) which is of-no-interest.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{CClearPurseOkay}} \\
\hline
\Delta \mathit{CConPurse} \\
\mathit{Cm}?, \mathit{Cm}! : \mathit{CMESSAGE} \\
\hline
\mathit{Cm}? = \mathit{CexceptionLogClear}(\mathit{Cname}, \mathit{Cimage}\ \mathit{CexLog}) \\
\Xi \mathit{CConPurseClear} \\
\mathit{Cstatus} = \mathit{CeaFrom} \\
\mathit{CexLog} \neq \varnothing \\
\mathit{CexLog}' = \varnothing \\
\mathit{Cm}! = \perp \\
\hline
\end{array}$$

In the D model the only difference is the use of *Dimage* rather than *Cimage*. The former is a total injection, the latter a total hash function.

$$\begin{array}{l}
\mathit{Cimage} : \mathbb{P}_1\ \mathit{CpayDetails} \rightarrowtail \mathit{CLEAR} \\
\mathit{Dimage} : \mathbb{P}_1\ \mathit{DpayDetails} \rightarrow \mathit{CLEAR}
\end{array}$$

Aside fom this difference, each D component 'is as' its C counterpart so that *DClearPurseOkay* 'is as' *CClearPurseOkay*. Note that since a hash function will be many-one rather than one-one, more than one collection of (supposedly) archived log entries can map to the same CLEAR code. Thus the possibility arises that a CLEAR code could be received that reflects the contents of an archive that does not correctly contain the corresponding local log contents.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{DClearPurseOkay}} \\
\hline
\Delta \mathit{DConPurse} \\
\mathit{Dm}?, \mathit{Dm}! : \mathit{DMESSAGE} \\
\hline
\mathit{Dm}? = \mathit{DexceptionLogClear}(\mathit{Dname}, \mathit{Dimage}\ \mathit{DexLog}) \\
\Xi \mathit{DConPurseClear} \\
\mathit{Dstatus} = \mathit{DeaFrom} \\
\mathit{DexLog} \neq \varnothing \\
\mathit{DexLog}' = \varnothing \\
\mathit{Dm}! = \perp \\
\hline
\end{array}$$

An individual purse is embedded in the community of purses using Z promotion [12], [13], [14], [15]. In this a collection of identical components is aggregated via an indexing function. In the Mondex case, this state information is supplemented by specific world level state, namely the world level log contents archive and the world level message ether. We show the details just for the D world; the C world 'is as' the D world:

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{DConWorld}} \\
\hline
\mathit{DconAuthPurse} : \mathit{NAME} ⤘ \mathit{DConPurse} \\
\mathit{Dether} : \mathbb{P}\ \mathit{DMESSAGE} \\
\mathit{Darchive} : \mathbb{P}\ \mathit{DLogbook} \\
\hline
\forall\, n : \mathrm{dom}\, \mathit{DconAuthPurse} \bullet \\
\quad (\mathit{DconAuthPurse}\ n).\mathit{Dname} = n \\
\forall\, \mathit{nld} : \mathit{Darchive} \bullet \mathit{first}\ \mathit{nld} \in \mathrm{dom}\, \mathit{DconAuthPurse} \\
\hline
\end{array}$$

As well as state information, we have a framing schema, which allows the embedding of individual component operations into the indexed world context. Specifically this: (a) singles out which component will perform the local operation, via an input index *Dnm*?; (b) stipulates that all *other* components skip; (c) allows the selected component to perform a completely unrestricted state change on its local state, ready for further restriction for individual operations. Using this framing schema, the *DClear* function in the D world promotes the purse-specific clear operation, or skips (via *DIgnore*).

$$\begin{array}{|l}
\multicolumn{1}{l}{\Phi \mathit{DOp}} \\
\hline
\Delta \mathit{DConWorld};\ \Delta \mathit{DConPurse} \\
\mathit{Dm}?, \mathit{Dm}! : \mathit{DMESSAGE} \\
\mathit{Dnm}? : \mathit{NAME} \\
\hline
\mathit{Dm}? \in \mathit{Dether} \\
\mathit{Dnm}? \in \mathrm{dom}\, \mathit{DconAuthPurse} \\
\theta \mathit{DConPurse} = \mathit{DconAuthPurse}\ \mathit{Dnm}? \\
\mathit{DconAuthPurse}' = \mathit{DconAuthPurse} \oplus \\
\qquad\qquad\qquad \{\mathit{Dnm}? \mapsto \theta \mathit{DConPurse}\,'\} \\
\mathit{Darchive}' = \mathit{Darchive} \\
\mathit{Dether}' \subseteq \mathit{Dether} \cup \{\mathit{Dm}!\} \\
\hline
\end{array}$$

[2]We employ a convention of pre-capitalizing only the names of types (schema and other): we augment this convention by prefixing a single character $A, B, \ldots$ to a name as required, to denote the model in question. Thus *CThing* is a schema or other type in the C model, whereas *Dthing* is a variable, usually a schema component, in the D model. To save space we employ a further lexical schema convention by saying *DSchema* 'is as' *CSchema*, which indicates that the text of *DSchema* can be generated from that of *CSchema* by replacing all *Cthing*s by *Dthing*s.

$$\begin{array}{l} \mathit{DClear} == \mathit{DIgnore} \lor \\ \quad (\exists\, \Delta \mathit{DConPurse} \bullet \Phi \mathit{DOp} \land \mathit{DClearPurseOkay}) \end{array}$$

The relationship between the C and D worlds is a retrenchment. We turn now to the details of the retrenchment data for *CClear* and *DClear* (where *CClear* 'is as' *DClear*).

The retrieve relation asserts the equality of (most of the) corresponding state values in the two models. (However, for brevity, the syntactic details of this are hidden within the heavy inverted commas — some such device as this is needed since the component names in the two worlds differ in their initial letter.)

$$\begin{array}{|l} \multicolumn{1}{l}{R^{PS}_{CD}} \\ \hline \mathit{CConWorld};\ \mathit{DConWorld} \\ \hline \mathrm{dom}\, \mathit{CconAuthPurse} = \mathrm{dom}\, \mathit{DconAuthPurse} \\ (\forall\, n \in \mathrm{dom}\, \mathit{CconAuthPurse} \bullet \\ \quad \text{“}\ \mathit{CconAuthPurse}\ n = \mathit{DconAuthPurse}\ n\ \text{”}) \\ \text{“}\ \mathit{Cether} = \mathit{Dether}\ \text{”} \\ \hline \end{array}$$

The within relation asserts the equality of the two indexing names in the two worlds, asserts that for the named purse, the union of its log and the relevant part of the archive in the two models agrees,[3] and makes explicit the contents of the corresponding input messages:

$$\begin{array}{|l} \multicolumn{1}{l}{W^{PS}_{CD,Clear}} \\ \hline \mathit{CConWorld};\ \mathit{DConWorld} \\ \mathit{Cm}? : \mathit{CMESSAGE};\ \mathit{Dm}? : \mathit{DMESSAGE} \\ \mathit{Cnm}?, \mathit{Dnm}? : \mathit{NAME} \\ \hline \mathit{Cnm}? = \mathit{Dnm}? \\ (\{\mathit{Cnm}?\} \lhd \mathit{Carchive}) \cup \\ \qquad (\mathit{CconAuthPurse}\ \mathit{Cnm}?).\mathit{CexLog} = \\ \quad (\{\mathit{Dnm}?\} \lhd \mathit{Darchive}) \cup \\ \qquad\quad (\mathit{DconAuthPurse}\ \mathit{Dnm}?).\mathit{DexLog} \\ \mathit{Cm}? = \mathit{CexceptionLogClear}(\mathit{Cnm}?, \\ \qquad \mathit{Cimage}\ (\mathit{CconAuthPurse}\ \mathit{Cnm}?).\mathit{CexLog}) \\ \mathit{Dm}? = \mathit{DexceptionLogClear}(\mathit{Dnm}?, \\ \qquad \mathit{Dimage}\ (\mathit{DconAuthPurse}\ \mathit{Dnm}?).\mathit{DexLog}) \\ \hline \end{array}$$

The output relation is relevant when things are working properly. In that case, the log entries in the *before* state of the logs, are correctly duplicated in the archives. One can then deduce from the within relation, that clearing the log will maintain the pursewise *CDAllValueAccountedPurse Cnm*? property, which states that for purse *Cnm*?, no value has been lost track of in either model during the clearing of the logs.

$$\begin{array}{|l} \multicolumn{1}{l}{O^{PS}_{CD,Clear}} \\ \hline \Delta \mathit{CConWorld};\ \Delta \mathit{DConWorld} \\ \mathit{Cm}! : \mathit{CMESSAGE};\ \mathit{Dm}! : \mathit{DMESSAGE} \\ \mathit{Cnm}?, \mathit{Dnm}? : \mathit{NAME} \\ \hline (\mathit{CconAuthPurse}\ \mathit{Cnm}?).\mathit{CexLog} \\ \quad \subseteq \{\mathit{Cnm}?\} \lhd \mathit{Carchive} \\ (\mathit{DconAuthPurse}\ \mathit{Dnm}?).\mathit{DexLog} \\ \quad \subseteq \{\mathit{Dnm}?\} \lhd \mathit{Darchive} \\ \mathit{CDAllValueAccountedPurse}'\ \mathit{Cnm}? \\ \mathit{Cm}! = \mathit{Dm}! \\ \hline \end{array}$$

The interesting part lies with the concession. This is applicable when it is not the case that in both models, the local logs' contents are safely lodged in the respective archives. Although the local states of the abstract and concrete purse remain equal (since both of their logs are cleared), the fundamental *CDAllValueAccountedPurse*′ *Cnm*? security property, which asserts that the whereabouts of all the money in the system possessed by purse *Cnm*? (in either model) is known, has been violated. Given the ramifications of the within relation, this is a genuine possibility opened up by the non-injectivity of the model D hash function. We defer further discussion to the Validation section.

$$\begin{array}{|l} \multicolumn{1}{l}{C^{PS}_{CD,Clear}} \\ \hline \Delta \mathit{CConWorld};\ \Delta \mathit{DConWorld} \\ \mathit{Cm}! : \mathit{CMESSAGE};\ \mathit{Dm}! : \mathit{DMESSAGE} \\ \mathit{Cnm}?, \mathit{Dnm}? : \mathit{NAME} \\ \hline \mathit{CconAuthPurse}\ \mathit{Cnm}? = \mathit{DconAuthPurse}\ \mathit{Dnm}? \\ \neg((\mathit{CconAuthPurse}\ \mathit{Cnm}?).\mathit{CexLog} \\ \qquad \subseteq \{\mathit{Cnm}?\} \lhd \mathit{Carchive} \land \\ \quad (\mathit{DconAuthPurse}\ \mathit{Dnm}?).\mathit{DexLog} \\ \qquad \subseteq \{\mathit{Dnm}?\} \lhd \mathit{Darchive}) \\ \neg(\mathit{CDAllValueAccountedPurse}'\ \mathit{Cnm}?) \\ \mathit{Cm}! = \mathit{Dm}! \\ \hline \end{array}$$

V. The Rest of the Tower

Thus far we have retrenched the lowest level C model of the earlier development to the D model in which a hash was used instead of a truly injective function to authorise clearing of purses' local logs. This is the bottom rung of the tower described in Section III. Now we sketch how the new D model can be related to the other models in the Mondex development, by indicating how models E and F are constructed.

First we lift model D to the level of abstraction of model B, by using a generic construction for factoring retrenchments[4] into a 'maximally abstract' retrenchment followed by a suitable refinement. The details for this are in [11], building on earlier work in [16]. A system, typically called *U*, is constructed out of the two original systems, and the required level of abstraction is defined via a collection of construction-specific properties. *U* captures this level by being refinable to

[3]This of course allows the two archive components themselves to disagree in a limited number of ways, opening up the possibility that the two *clear* operations will reveal a genuine difference between them, since the non-injectivity of *Dimage* means that the right code can be generated from the wrong archive.

[4]In this case it is the retrenchment given by composing the B to C refinement with the C to D retrenchment. See Fig. 1.

any system that also enjoys the properties, making it the most abstract such system. Any system inter-refinable with U is just as good as U, so we have the option of replacing U with a more convenient system if we wish.

In our case, for the clear operation the construction yields:

$$\begin{array}{|l} \hline \textit{protoEClear} \\ \textit{BClear};\ \Delta \textit{DConWorld} \\ \textit{Dm}?, \textit{Dm}! : \textit{DMESSAGE} \\ \hline (R^{PS}_{BD} \wedge W^{PS}_{BD,Clear}) \wedge ((R'^{PS}_{BD} \wedge O^{PS}_{BD,Clear}) \vee C^{PS}_{BD,Clear}) \\ \hline \end{array}$$

where R^{PS}_{BD} 'is as' R^{PS}_{CD}, $W^{PS}_{BD,Clear}$ 'is as' $W^{PS}_{CD,Clear}$, $O^{PS}_{BD,Clear}$ 'is as' $O^{PS}_{CD,Clear}$, and $C^{PS}_{BD,Clear}$ 'is as' $C^{PS}_{CD,Clear}$, and where:

$$\begin{array}{l} \textit{BClear} == \textit{BIgnore} \vee \\ \quad (\exists\, \Delta \textit{BConPurse} \bullet \Phi \textit{BOp} \wedge \textit{BClearPurseOkay}) \end{array}$$

This is as expected, except that it conceals the fact that in *BClear* Φ*BOp*, instead of containing Δ*BConWorld* (as would be expected) actually has Δ*BetweenWorld*, where *BetweenWorld* features additional structure and constraints imposed on *BConWorld* in order to enable the A to B backward refinement to discharge. Otherwise the constituents of *BClear* 'are as' their corresponding *CClear* ones.

We do not have the space to convince the reader that the constraints in *BetweenWorld* do not materially affect our discussion. They express the consistency between the cryptographically protected messages in the ether and the purses' internal states. Interested readers can refer to [1].

In fact the E model operation just given contains a large amount of duplication of state and other information; the B and D parts of the state say much the same thing in slightly different ways, via the various equalities on state components buried beneath the surface. Above, we noted that it is sufficient to have a system which is inter-refinable with the U model given by the generic construction, so it is worth examining *protoEClear* to see if we can achieve some simplification. It so happens that we can, though we do not have space to go through the details. In point of fact we can replace *protoEClear* with the simpler:

$$\textit{EClear} \quad \text{'is as'} \quad \textit{DClear}$$

except that *DetweenWorld* (which now 'is as' *BetweenWorld*) replaces occurences of *DConWorld* in *DIgnore* and Φ*DOp* in *DClear*. Thus *DClear* is at the right level of abstraction after all, due above all, to the simplicity and minimalist nature of the D construction.

All that remains of building the tower is to construct model F. Here it is best to proceed in an ad hoc manner from model A, rather than pursue the lifting construction blindly. Firstly, note that funds lost track of through innapropriate clearance of purse logs are still 'lost', though in a different manner. One way forward is thus to try to subsume them under the abstract 'lost' banner. In fact one can replace the relevant *equality* between lost components in the A to B retrieve relation by a suitable *inequality*, and one can thereby derive a valid backward refinement from model A to model E. This leads to one option for making model F, i.e. to make it simply a copy of model A (as shown in Fig. 1).

A perhaps more honest approach though, is to distinguish between 'lost but traceable' and 'lost but untraceable' at the top level. This necessitates the introduction of a new abstract state component to hold the lost but untraceable funds. Also we would need a new abstract operation, corresponding to the clear operations lower down, that would nondeterministically choose between doing nothing, and transfering a portion of the lost but traceable funds into the lost but untraceable category. By this means, the equality between abstract and concrete traceably lost funds, characteristic of the model A to model B retrieve relation, could be maintained. So this leads to another option for making model F, as a copy of model A but including the enhancements just discussed. A straightforward retrenchment now takes model A to model F.

VI. Validation

In preceding sections we presented a selection of models for the purse, displaying varying degrees of accuracy regarding the clearing of a purse's log (compared, that is, to the real world). In this section, we discuss the adequacy of these in the light of real world requirements and assumptions. Given that retrenchment is such a flexible notion, permitting almost arbitrary model evolution, it behoves us to validate the way that retrenchment is used, to ensure that this flexibility does not lead us adrift of, rather than towards, our ultimate system objectives. Since the characteristics of the lowest level retrenchment of the tower propagate to the higher levels, we concentrate on the C to D retrenchment.

First we note that the purse does not contain any explicit interlocks at purse state level, which attempt to ensure that its log is cleared only when its contents are safely in the archive — aside that is, from those implicit in the properties of the clear code or hash function, which are correlated with the purse log contents. Why is this? The point is that if transactions are going wrong anyway, so that the log is in use, and thus needs to be cleared, then any aspect of the purse's environment may be responsible. The purse may even be imprisoned in a harness that feeds it misleading messages in an attempt to fool it into creating nonexistent value. Any state level interlock, since it must correlate with the archive state to be at all useful, must be mediated by the potentially threatening environment; therefore it is fundamentally unreliable. In order to provide useful backup, the log clear operation, should thus rely on *weaker* assumptions about the environment than the transaction protocol it supports, if this is at all possible.

What happens if the purse receives a false but credible clear message? If the current log contents are not all in the archive, then when they are erased, the ability to rescue the funds pertaining to the failed transactions referred to in the missing log entries is lost. Although the 'All value accounted' security property is compromised, the more important 'No value created' one, is maintained via the genuinely diminished total value described by the global state of the purse world.

So in reality, attackers stand to gain nothing financially by forging a clear message. Then again, a given attacker either might not know this, or might have some unusual motivation (for example, to undermine the credibility of the banks underwriting the Mondex system) for making his quarry lose funds by turning the recoverably lost into the irrecoverably lost.

It is in the context of the above that we validate the C to D retrenchment. The main objective is to estimate how much greater the chance of inappropriately wiping the log is, when a hash function is used compared to when a truly injective one is used. We can do this under two broad classes of assumptions: firstly, that any inappropriate clear message received, arrived by chance; secondly, that any such message is the result of malicious attack. A third possibility, that the archive itself sends a clear message at an inappropriate time, can be discounted: the archive is a critical system component, and the security of the system as a whole can only be established on the assumption that the critical components do not deliberately misbehave.

In the first case (pure chance), all message bits can be assumed to be random variables. In the second case (malicious attack), only the hash value itself might be assumed cryptographically protected, and the rest of the message could be constructed by a knowledgeable attacker. Therefore in the worst case, only the hash itself offers any protection against inappropriate use, and thus to offer a useful level of protection, one would assume that at least 256 bits of hash value would be used. That many bits of hash would offer a good degree of protection against an attack based on random trials, since at one trial per millisecond, exhausting all the possibilities would take many orders of magnitude longer than the age of the universe. Obviously though, an attacker with some inside knowledge of the hash algorithm could conceivably do quite a bit better than chance, depending on the depth of his knowledge. So, in reality, an even greater number of bits could easily be used, for the sake of robustness, and as insurance against future advances in cryptanalysis.

As well as the purely statistical argument just outlined, a proper validation of the hash function approach ought to take the physical context into account. Unless an attacker has built a bespoke environment for the purse in order to assault it (and the banks underwriting the Mondex system might well consider it worthwhile to make the Mondex design robust against such a possible if unlikely attack, if only to protect their credibility, as indicated above), then clear messages will usually be issued only in the surroundings of the bank, when an honest reconciliation of failed transactions is being attempted. Under such circumstances, not only is the possibility of malicious attack much reduced, but the purse will be connected to the archive by a highly reliable communications link, reducing to vanishingly small, the likelihood of chance errors arising through transmission.[5] This strongly reinforces the argument that the design using a hash instead of a genuinely injective clear function is adequately dependable, and the retrenchment that describes the passage from the ideal design to the more realistic one is therefore vindicated.

[5]Given the figures already cited above, we omit any quantative estimates.

VII. Further Considerations

The reasoning of the previous section prompts further re-evaluation of the Mondex system as a whole. Consider the other 'retrenchment opportunities' noted in Section II, namely the sequence number limit, the log size limit, and the balance enquiry. Although the reader will largely have to take our word for it due to lack of space, it is the case that in all of these scenarios, the difficulties addressed could arise via the 'natural' running of the Mondex protocol as it was originally intended to function. Thus one could conceive of hitting the sequence number limit if enough transactions were to take place; or (even more easily) that the log size limit would be reached if enough failing transactions were to happen; or that a transaction of the protocol which was long-lived due to communication latency, could, if a balance enquiry operation was called in the middle of it by an impatient recipient, yield a balance that was out of step with what the abstract A model would predict at such a moment.

The retrenchment opportunity opened up by the use of a hash rather than an injective function is of a different nature, since clearing the log in the intended way only takes place when it is safe to do so. It is only if the capabilities of the purse are exploited in a non-intended way that the problem comes to light. Furthermore, to the extent that the problem discussed is a real one, then there are repercussions to be considered for the rest of the Mondex system as follows.

Suppose the probability that the protection offered by the hash in the clear log case could be broken is indeed nonzero. This would arise because the environment would be capable of inventing the right message by some means. However if the environment were capable of inventing that particular message, then there is no reason to suppose that it could not invent other cryptographically protected messages on which the security of the Mondex system relies. Thus the problem opened up by a putative weakness in the clear log case spreads to the whole protocol since there is no reason to suppose that significantly different protection mechanisms would be implemented, within the restricted code area available on a smartcard, for different parts of the Mondex system (the true details of the Mondex system's cryptographic protection mechanisms are not in the public domain).

Consider the Mondex protocol, the essentials of which are illustrated in Fig. 2 for the case of two purses. A successful run of the protocol requires the successful exchange of three messages, each of which is security critical. If the possibility of breaking the cryptographic protection mechanisms is indeed nonzero, then one would be able to break not only the integrity of log clearing, and thence the 'All value accounted' security property, but also the more fundamental 'No value created' security property, via a sequence of events as follows.

The scenario consists of one purse, the recipient, or 'To' purse, and a suitable harness in which to put it, and within which its operations may be called. The harness takes over the role that the ‘From’ purse in a genuine transaction would play, except that it does so fraudulently. The harness firstly calls the ‘StartTo’ operation in the To purse, quoting transaction details for a fictitious transfer of funds to the To purse. (Since the security of the Mondex system does not rely on the secrecy of this information, a determined attacker could in principle create it.) The To purse completes the call with an encrypted ‘req’ (request) message, which is discarded by the harness, and enters the ‘Bepv’ (expecting payment value) state. The harness then sends a suitably encrypted ‘val’ (value) message to the To purse, acting as if it were a genuine ‘From’ purse. The To purse accepts the message as input for the ‘Val’ operation, during which it increments its balance by the requisite amount (and it completes the transaction by sending an ‘ack’ (acknowledgement) message). By this means, via an unjustified increase in the To purse’s balance, nonexistent money has been created.

The sequence of events just outlined shows us that the security issues surrounding the integrity of the log clearing mechanism are pertinent to the integrity of the whole Mondex system. One can pursue this line of thought to analyse, for example, the risk associated with random receipt of the required faked messages by a To purse, rather as we did above for the hash.

The assumptions in [1] on the communication medium which transports the protocol messages of the Mondex system are that the medium is unreliable (i.e. that messages can be lost), but that the cryptographically protected messages cannot be faked — without the latter nothing could be proved. But obviously this last assumption cannot be absolutely true. Thus a probabilistic risk anlysis as we have suggested, to support the strength of the assumption and to supplement the pure logic of the security proof, is in fact needed.

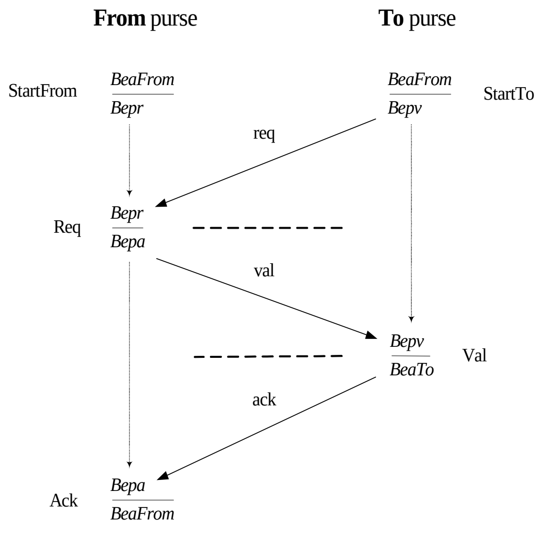

Fig. 2. The Mondex Protocol.

VIII. Conclusions

In the preceding sections, we introduced the Mondex development and its ‘retrenchment opportunities’; and also the *Tower Pattern* and supporting refinement and retrenchment notions. The paper then focused on one of the retrenchment opportunities, the hashing of log clear messages, and examined its treatment via the tower in detail. What we got was a typical *Tower Pattern* arrangement, namely two refinement developments, dealing with different (and from a refinement point of view incompatible) levels of detail, connected via a collection of retrenchments into a large commuting diagram. The latter shows how the different stages of the two developments can be related to one another, and the theory underpinning the tower [11] can even help in the automatic construction of part of it.

Crucial to the utility of such a state of affairs is validation, since retrenchment by itself is such an extremely permissive technique. For this reason, a validation from the domain perspective is vital in corroborating any formal development done with its help, to ensure that the results of its use are helping rather than hindering the attainment of overall system objectives. By reasoning about the context of the log clear operation, we were able to provide the needed validation, and the arguments advanced were developed into further insights about the integrity of the Mondex system as a whole. The whole process was an excellent illustration of the ability of retrenchment to bridge the gap between ‘academic’ or purely logical reasoning about systems, and the often niggly details that have to be considered during the business of engineering a real product within its pragmatic context.

References

[1] S. Stepney, D. Cooper, and J. Woodcock, “An electronic purse: Specification, refinement and proof,” Oxford University Computing Laboratory, Tech. Rep. PRG-126, 2000.
[2] D. of Trade and Industry, “Information Technology Security Evaluation Criteria,” 1991, http://www.cesg.gov.uk/site/iacs/itsec/media/formal-docs/Itsec.pdf.
[3] *Common Criteria for Information Security Evaluation*, ISO 15408, v. 3.0 rev. 2, 2005.
[4] R. Banach and M. Poppleton, “Retrenchment: An engineering variation on refinement,” in *2nd International B Conference*, ser. LNCS, D. Bert, Ed., vol. 1393. Montpellier, France: Springer, April 1998, pp. 129–147.
[5] ——, “Sharp retrenchment, modulated refinement and simulation,” *Formal Aspects of Computing*, vol. 11, pp. 498–540, 1999.
[6] M. Poppleton and R. Banach, “Controlling control systems: An application of evolving retrenchment,” in *Second International Conference of B and Z Users*, ser. LNCS, D. Bert, J. Bowen, M. Henson, and K. Robinson, Eds., vol. 2272. Grenoble, France: Springer, January 2002, pp. 42–61.
[7] R. Banach and M. Poppleton, “Retrenching partial requirements into system definitions: A simple feature interaction case study,” *Requirements Engineering Journal*, vol. 8, no. 2, 2003, 22pp.
[8] R. Banach, M. Poppleton, C. Jeske, and S. Stepney, “Retrenching the purse: Finite sequence numbers and the tower pattern,” in *Formal Methods 2005*, ser. LNCS, J. Fitzgerald, I. Hayes, and T. A., Eds., vol. 3582. Newcastle, UK: Springer, 2005, pp. 382–398.
[9] R. Banach, C. Jeske, M. Poppleton, and S. Stepney, “Retrenching the purse: Finite exception logs, and validating the small,” in *Workshop on Software Engineering 2006*, M. Hinchey, Ed. Loyola College, MD: IEEE Computer Society Press, 2006.
[10] R. Banach, M. Poppleton, C. Jeske, and S. Stepney, “Retrenching the purse: The balance enquiry quandary, and generalised and (1,1) forward refinements,” *Fundamenta Informaticae*, 2006, (to appear).

[11] C. Jeske, “Algebraic integration of retrenchment and refinement,” Ph.D. dissertation, University of Manchester, 2005.
[12] J. Woodcock and J. Davies, *Using Z: Specification, Refinement and Proof*. Prentice-Hall, 1996.
[13] J. Derrick and E. Boiten, *Refinement in Z and Object-Z*, ser. FACIT. Springer, 2001.
[14] S. Stepney, F. Polack, and I. Toyn, “An outline pattern language for Z,” in *Third International Conference of B and Z Users*, ser. LNCS, D. Bert, J. Bowen, S. King, and M. Waldén, Eds., vol. 2651. Turku, Finland: Springer, June 2000, pp. 2–19.
[15] ——, “Patterns to guide practical refactoring,” in *Third International Conference of B and Z Users*, ser. LNCS, D. Bert, J. Bowen, S. King, and M. Waldén, Eds., vol. 2651. Turku, Finland: Springer, June 2000, pp. 20–39.
[16] R. Banach, “Maximally abstract retrenchments,” in *Proc. IEEE ICFEM2000*. York: IEEE Computer Society Press, August 2000, pp. 133–142.
[17] D. Bert, J. Bowen, S. King, and M. Waldén, Eds., *Proc. ZB2003: Formal Specification and Development in Z and B*, ser. LNCS, vol. 2651. Turku, Finland: Springer, June 2000.