What Can the π-calculus Tell Us About the Mondex Purse System?

Cliff B. Jones and Ken G. Pierce
School of Computing Science
Newcastle University, NE1 7RU, UK
{cliff.jones,k.g.pierce}@ncl.ac.uk

Abstract

This paper looks at the wider system surrounding a "Mondex" electronic purse. It does this from a process-oriented perspective using the π-calculus. Our model includes the issuing of purses by an authorised bank and the decisions of cardholders to participate in transactions.

1 Introduction

One of UKCRC's[1] "Grand Challenges"[2] is that on "Dependable Systems Evolution"[3] [16]. One objective is to experiment –with formal methods– on an "electronic purse" system whose details are derived from that which went under the name "Mondex". The common source document for the experiments is [14]. In particular that report sets out an important security property that should hold (i.e. it should not be possible to "print money") and there is interest in how this can be proved in various approaches.

It is our position that no one "formal method" is likely to be adequate for exploration of all facets of a problem like electronic purses. Henderson's thesis [5] is one example where a collection of formal approaches are used and each illustrates different facets of a difficult problem. One could also point to the work on "Dynamic Coalitions" in [1], which uses a state based approach to illuminate the space of options that hide behind the buzz phrase. Other aspects of elements such as the communication behaviour might well have been tackled using process algebraic approaches but the cited reference studies the space of systems using a model-oriented approach (VDM [8, 7]).

The π-calculus is described in [11, 12]; it is a development of CCS [10] that permits the passing of process names as parameters.

Our focus in this paper is on the way in which the ν name binding can be used to establish private communication between processes. In fact, one key issue that such a process algebraic approach clarifies is that it is necessary to be precise about what is included in or excluded from such a system. We are not, of course, suggesting that the ν operator is a solution to the delicate implementation issue of how to ensure that only authorised purses can be used in payment. We are only claiming that this is a useful abstraction whose properties express one facet of this problem rather naturally.

Section 2 of this paper gives a basic outline of the Mondex system and the π-calculus. Our model is presented in Section 3 and Section 4 includes an extension that addresses the physicality of our model. In Section 5 we consider tool support as a means of verification and we draw conclusions from our work in Section 6.

2 Background

2.1 Mondex: An Electronic Purse

The Mondex electronic purse system was a real development by the NatWest Development Team; the system was evaluated by the EU under the Information Technology Security Evaluation Criteria (ITSEC). ITSEC has seven levels of security certification — E0 through E6 (lowest to highest). The Mondex system achieved an E6 level of certification; this required both evaluation and *proof* of the design, hence a formal specification was undertaken using the Z method. A version of the Z specification was eventually published in a monograph from Oxford University [14].

The main component of the Mondex system is the purse. A purse holds monetary value and allows the holder to transfer sums of money to other Mondex purses. A purse is hosted on a smartcard. Transfers require card readers in order to allow these smartcards to communicate and these card readers are (confusingly) known as wallets.

Wallets have three modes of operation. With a single-slot card reader, the two purses are inserted sequentially and money is transferred first to the wallet and then to the receiving purse (this is possible because a wallet itself contains an integrated purse). A two-slot card reader allows the

[1] See www.ukcrc.org.uk
[2] See www.ukcrc.org.uk/grand_challenges/index.cfm
[3] See www.bcs.org/server.php?show=ConWebDoc.4721

12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)
0-7695-2895-3/07 $25.00 © 2007 IEEE

two purses to be inserted simultaneously. The third mode is a networked mode in which two card readers are connected via a phone line. This paper focuses on the two-slot mode.

The Mondex system is interesting because the smartcards are autonomous: they perform offline transactions without the aid of a central authority or logging ability. Security (i.e. cryptography) is performed on-card.

The security requirements of the system were captured by decribing a number of "security properties". The main property required by the bank is *no value creation*, i.e. the sum of all purse balances does not increase. The other main properties are *all value accounted*, the sum of all purse balances does not change; *authentic purses*, only authorised cards can participate in transactions (cryptography is assumed to work and 'strength of mechanism arguments' are absent from the monograph); and *sufficient funds*, a purse can only transfer value up to the amount of its current balance.

2.2 π-calculus

The π-calculus is a process algebra developed as a continuation of the work on CCS (Communication Concurrent Systems); like CCS and other process algebras, it allows us to model systems in terms of processes communicating with other process.

The basic units of the π-calculus are *names*. Names are channels of communication; the prefix operators input — $x(y).P$ — and output — $\overline{x}\,y.P$ — allow processes to communicate synchronously. Two processes willing to synchronise on the same channel name can communicate (and pass variables as parameters). A key property of the π-calculus is that names can be passed as variables and used as channels for further communications.

Terms in the π-calculus can be composed sequentially, $P.Q$ ('then'); in parallel, $P \mid Q$ (in which P and Q are both able to communicate); or using external choice, **+** (in which only P *or* Q can communicate; in so doing the other term transitions to **0**, the nil process).

The . operator can be used to indicate that a process proceeds to behave like another named process; this property permits tail recursion. Replication allows us to say that a process can always perform a particular communcation; the ! ('bang') operator is effectively a shorthand for $P \mid !P$.

As discussed previously, the ν operator is key to our model. It creates a 'new', unique channel name and binds that name in the following process term. This name is private; it cannot be known to external processes (and thus used for communication) until it is explicitly provided to another process over a known channel.

Our notation also uses $\sum_{i\in\{1,\ldots,n\}} P_i$ as a short hand for multiple choice, P_0 **+** ... **+** P_n and $\prod_{i\in\{1,\ldots,n\}} P_i$ as a shorthand for multiple parallel composition, $P_0 \mid \ldots \mid P_n$.

3 A π-calculus Model

3.1 Design Choices

Previous treatments of the "Mondex" system have looked at transactions between purses predominantly from a state-based perspective (e.g. [13]). They aim to show that the security properties (e.g. *no money creation*) hold in the system.

We felt however that it would also be interesting to look at the customers' decisions to exchange money –deal brokering– and at the way in which the bank issues purses to customers.

3.1.1 Deal Brokering

Our initial approach was to model deal brokering as a summation of "deals" between each pair of customers. In this case, transactions could occur non-deterministically. This approach led, however, to a possible deadlock (reminiscent of the dining philosophers' problem) where a customer could attempt more than one simultaneous deal and deadlock the process. This is exposed as a problem if timeouts are introduced during refinement, because the abstract system cannot achieve resolution of deadlock whereas the implementation has extra behaviours.

To combat this, we changed our model so that customers can non-deterministically attempt a transaction with a distinct purse; a pair of customers must agree (i.e. synchronise) on the transaction. This synchronisation then prohibits either customer from attempting other transactions until the current transaction has ended — either through success, failure or one party deciding to abort.

3.1.2 The Bank

The bank has been modelled to accept requests from customers for purses. The key question we faced was the role of the bank in authenticating purses. Since the bank creates all purses, it is possible to hold a set of 'authorised purses' against which the bank could check a given purse before a transaction. This abstract authorisation would aid in showing that the security properties hold. Unfortunately, in the real system, purses exist "on their own" and wallets cannot defer to the bank, hence this authentication would have to be refined before implementation.

We thus decided to model the system more closely, by having the bank simply create purses and relinquish them to customers; it is then up to the wallets –presumably through a form of secret key authentication– to verify the authenticity of a purse.

3.2 Our Chosen π-calculus Definition

The top-level process in the model is $World$ – the world contains a bank (which can issue purses); a number of customers (who initially do not own purses); and a number of wallets (required for transactions):

$$Bank \mid \left(\prod_{i \in \{1,\ldots,n\}} CustNoP_i \right) \mid \left(\prod_{x \in \{1,\ldots,w\}} Wallet_x \right)$$

The bank is able to issue purses at the request of a customer.

$$Bank \stackrel{\text{def}}{=} !\,(\boldsymbol{\nu} p)\big(reqp(c_i).\overline{c_i}\,p\big).P_p$$

As in the physical system, the bank is unable to authenticate a purse once it has entered 'the real world'. Each purse is created with a new, unique channel — it is the intention that this can represent the secret of the purse.

A customer without a purse is able to request one; they provide a private channel to the bank over which they receive their purse's identity.

$$CustNoP_i \stackrel{\text{def}}{=} (\boldsymbol{\nu} c_i)\overline{reqp}\,c_i.c_i(p).Cust_{ip}$$

Once a customer has received a purse, they are able to participate in transactions. The channel p is their unique connection to the purse.

A customer with a purse is able to participate in transactions. They may request a transfer of arbitrary value from any other purse –or– accept an incoming request. Performing either of these actions causes the customer to enter a state in which they can no longer perform further transactions until the current one is complete.

$$Cust_{ip} \stackrel{\text{def}}{=} \left(\sum_{j \in \{1,\ldots,n\}} \overline{req_j}\,v \;+\; req_i(v) \right).CustT_{ip}$$

Note that customers are indexed by i and p, where i is effectively a public identity used by other purses to request a transaction and p is the private identity of the purse that is only used when it is explicitly shared with a wallet.

Although not considered here, our model could be extended to allow a customer to have multiple purses (i.e. a set of purse identities). As long as each purse only participated in a single transaction at any one time, this would not affect the integrity of the model.

Once a pair of customers decide to initiate a transfer, they must find a wallet with which to perform the transfer. This is modelled as an external choice; although it is possible that the customers may choose different wallets, deadlock is avoided because they can always abort the transaction.

A customer participating in a transaction ($CustT_{ip}$) is released after the transaction has completed successfully (ok), fails (f) or they decide to abort (ab). Once released, they return to being a customer able to participate in further transactions ($Cust_{ip}$).

$$CustT_{ip} \stackrel{\text{def}}{=} \left(\sum_{x \in \{1,\ldots,w\}} \overline{insert_x}\,p \right).\big(ok_p() \;+\; f_p() \;+\; \overline{ab_p}\,\big).Cust_{ip}$$

A wallet is initially defined to be ready to accept one purse.

$$Wallet_x \stackrel{\text{def}}{=} insert_x(p).Wallet_{xp}$$

Once the first is purse inserted, it can be removed (and the wallet returns to being empty) or a second purse can be inserted — in which case the transaction continues.

$$Wallet_{xp} \stackrel{\text{def}}{=} ab_p().Wallet_x \;+\; insert_x(q).\big(\cdots\big)$$

For the transfer to occur, each purse must be authentic — we know that the wallet must authorise a purse without deferring to a central authority. If either purse is not considered authentic, the transfer fails. It should be noted that the wallet only checks the authenticity of the purses once both have been inserted. It could be modelled to challenge (and possibly reject) the first purse as soon as it is inserted, however this does not affect the integrity of the model. Authentication is defined in $Wallet_{xp}$ as...

$$\big(\textbf{if } [\![auth(p,q)]\!] \textbf{ th } WalletC_{xpq} \textbf{ el } \big(\overline{f_p}\,|\overline{f_q}\,\big).Wallet_x\big)$$

At this stage, the wallet needs to perform some additional checking before the transfer can occur. The first part of the definition for $WalletC$ (below) allows either customer to abort the deal at any point up to the actual transfer taking place in $WalletT$. The ability of either customer to abort the process has been a difficult issue. In "real life", a customer could pull their card out at any stage, but modelling this as a parallel process (as above) during the transfer is tricky.

$$WalletC_{xpq} \stackrel{\text{def}}{=} \Big(\big(ab_p().\overline{f_q} \;+ ab_q().\overline{f_p}\,\big).Wallet_x \;\Big|\; \cdots\Big)$$

Ideally, we would like an abort to release both purses, the second customer and the wallet; more importantly, we need to ensure that the value of the transfer has either not left the original purse (residing in the value or lost component), or has successfully been transfered to the requesting purse. We also wish to model the failure of the transfer for other reasons (modelled as a non-deterministic choice at this stage), while ensuring the same properties hold and to introduce timeouts at a later stage.

While it is possible to devise a representation for this, we suspect that it would be messy. In order to avoid the issue at this stage, we simply do not allow the transfer to be aborted after a certain point.

The wallet needs to know the value of the transfer *and* the direction (e.g. p pays q). Also, you may recall that a customer is able to *attempt* a transaction of arbitrary value, hence the wallet must also ensure that there is enough money in the paying purse to complete the transfer. These issues are represented by the abstract processes $[\![KEY]\!]$ (input from the keypad that yields a value v for the transfer and swaps p and q as necessary) and $[\![BAL]\!]$ (balance check for sending purse).

The rest of the definition for $WalletC$ is:

$$\Big([\![KEY]\!].\textbf{if } [\![BAL]\!]_v \textbf{ th } WalletT_{xpqv} \textbf{ el } (\overline{f_p}\,|\overline{f_q}\,).Wallet_x\Big)$$

The wallet is now able to perform the transfer. The wallet creates a new private channel c, which it uses for transfers. It acquires send and receive channels from each purse, thus allowing it to instruct the purses to pass money.

$$WalletT_{xpqv} \stackrel{\text{def}}{=} (\boldsymbol{\nu} c)\Big(\overline{p}\,c.c(S_p,R_p).\overline{q}\,c.c(S_q,R_q).\cdots\Big).Wallet_x$$

From the wallet's perspective, the actual transfer occurs as below. If successful, it releases the customers indicating this success (ok), if not, it releases the customers with a fail (f).

$$\Big(\overline{S_p}\,v.\big(S_p(ACK).\overline{R_q}\,v.(\overline{ok_p}\,|\overline{ok_q}\,) + (\overline{f_p}\,|\overline{f_q}\,)\big)\Big)$$

The purse itself consists of a value and a lost component (represented by abstract processes). The value is the current balance of the purse; if a transfer didn't complete, the value of that transfer is recorded in the lost component (to be restored to the balance at a later date).

$$P_p \stackrel{\text{def}}{=} [\![V_p]\!] \;\Big|\; [\![L_p]\!] \;\Big|\; !(\boldsymbol{\nu} sr)\Big(p(c).\overline{c}\,sr.\big(\cdots\big)\Big)$$

A purse performs transfers by synchronising with a wallet and providing channels for sending and receiving money. When the purse receives a request to send money, the amount is taken from the value component — it then non-deterministically succeeds (and acknowledges this fact to the wallet) or fails (adding the amout to the lost component). A request to receive money adds the amount to the value component.

$$s(v).[\![V_p - v]\!].(\overline{s}\,ACK + [\![L_p + v]\!]) + r(v).[\![V_p + v]\!]$$

4 Physicality of the model

One drawback of our model, as it stands, is the distance between the representation and the reality of the physical system. It is not exactly clear what p represents: in the present model, p is a unique, private link to a purse that allows a customer to perform transactions.

In reality, the original Mondex cards were intended to replace cash. They could therefore be passed around, become lost or stolen and used by whoever currently held the card (which might not necessarily be the original owner of the money). There is no notion of an authentic user in the system.

It is possible to extend our model to include a notion of loss of ownership of purses. It is probably unnecessary to include this in the full model, but is an interesting aside to show how the model might be brought closer to the physicality of the real system.

$$World \stackrel{\text{def}}{=} \cdots Lost_{\{\}}$$

$$Lost_L \stackrel{\text{def}}{=} lose(p).Lost_{L\cup\{p\}} + \overline{find}\,p.Lost_{L\setminus\{p\}}$$

$$CustNoP_i \stackrel{\text{def}}{=} \cdots + (find(p) + steal(p)).Cust_{ip}$$

$$Cust_{ip} \stackrel{\text{def}}{=} (\overline{lose}\,p + \overline{steal}\,p).CustNoP_i + \cdots$$

In this extension, a customer with a purse can lose that purse or have it stolen. In both cases, the customer loses the ability to interact over p (and thus can no longer use the purse). When a purse is lost, p enters "lost property", an abstract place where purse 'handles' go when they are lost. When a purse is stolen, control of p is directly transferred to the unscrupulous customer who stole it.

Customers without purses can still request one from the bank (whose definition remains unchanged), or they may steal one. They may also find a purse and gain control of a p from lost property (and subsequently p is no longer lost). In all cases, they become customers with purses and they can perform transactions by means of p.

In all three cases (losing, stealing and finding a purse), this extension assumes that the physical act occurs simultaneously (e.g. a customer leaves their purse on a bus, or has their pocket picked).

Our model uses the notion of p as a private channel of communication to a purse in order to abstract away message authentication. This extension shows that "ownership" of p by a customer ($Cust_p$) is simply a matter of circumstance that allows that customer to use the purse at the current time and is not necessarily a representation of legal ownership. It could be argued that p is a property of the purse P and not of the customer using it.

12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)
0-7695-2895-3/07 $25.00 © 2007 IEEE

5 Tool Support

The obvious tool to use when working with the π-calculus is the Mobility Workbench (MWB) [15], a tool for analysing systems described in the polyadic pi-calculus. The Mobility Workbench can be used to interactively simulate agents (by allowing the user to select commitments) and to find and report deadlocks within an agent. MWB can also perform model checking using modal logic assertions and decide *open bisimulation equivalences* (for agents with *finite control*).

We have taken a basic version of the model presented in Section 3.2 and run it through the MWB, looking for deadlocks. The version used includes the issuing of purses to customers by the bank and customers brokering deals (in the simple example, synchronisation between customers). Not surprisingly, no deadlocks were detected up to the maximum number of processes that the tool can handle. The tool reported that a system with a single customer will deadlock, which is correct – once the customer has been issued a purse, it has no one to deal with and the bank has no one to whom it can issue further purses.

The table below shows the results of the experiments, including the number of customers in the model, the "size" of the state space as reported by the tool and the time taken to check for deadlocks.

Customers	"Size"	Time (s)
1	3	0
2	11	0
3	55	0.031
4	357	1.422
5	n/a	n/a

As one must expect, the time taken and/or storage required increases exponentially with the number of processes. When the system was increased to five customers (and the bank), the tool crashed through lack of memory before a result was returned.

The ability of the tool to check modal logic assertions about agents could prove useful in checking other properties of the system, although it is currently unclear if this would allow us to say anything of interest about the protocol.

If our model was extended with the message transfer protocol of the original monograph –by refining the abstract transfer that occurs between two purses in a wallet– the equivalence checking ability of MWB could be used to show the observational equivalence of any refinement that is made, but again this is currently unclear.

The translation of π-calculus terms into Petri-nets presented in [3, 4] could allow model checking to be performed in the "as yet unnamed" tools being developed by that group. The translation does not currently permit recursion and we have so far not attempted a petri-net representation of our model.

6 Conclusions

Our model uses a process algebraic notation to represent not only the transfer of money between purses, but also a way in which the bank, customers and wallets might interact within the system. It illustrates how the ν name binding can be used to model private communication between these processes and that the ability to pass these private channel names between processes provides us with a concise abstraction from the issues of implementation.

In allowing customers to abort transactions, we have included behaviours in the abstract model that may be required when timeouts are introduced during refinement; this however causes problems if a customer aborts at a key stage of the transfer. Solutions we have considered include some form of exception handling, such as a $catch$; or perhaps an asymmetric operator that is similar to $\mid$, which allows one term to be declared as uninterruptable after a certain point.

When discussing our model, we realised that introducing a representation of customers allows us to consider the authenticity of users in the system as well as the authenticity of purses: is the holder of the current purse authorised to use it? Again, the problem here is the isolation of wallets from a central authority; our model could be extended to allow reasoning about the authentication of customers, i.e. using PIN numbers.

As well as customer authentication, we could expand the system to include "hostile" processes that attempt to forge communications within the system – man-in-the-middle attacks, for example – and determine their ability to affect the system.

If we wanted to reason about the implementation of our model, one interesting possibility would be to look at developments of the techniques proposed in [9] and developed in [2]; Koutny's notion of "interface refinement" is presented in a CSP [6] framework so either new research is required or we would have to consider the sub-problem that could be handled in CSP.

7 Acknowledgements

This work has been supported by the UK EPSRC under the "Splitting (Software) Atoms Safely" project. We would like to thank Björn Victor for his help with the Mobility Workbench and Koutny, Niaouris et al. with regards to their Petri-net translation of π-calculus terms.

References

[1] J. W. Bryans, J. S. Fitzgerald, C. B. Jones, and I. Mozolevsky. Dimensions of dynamic coalitions. Technical Report CS-TR-963, School of Computing Science, University of Newcastle, May 2006.

[2] J. Burton. *The Theory and Practice of Refinement-After-Hiding*. PhD thesis, University of Newcastle upon Tyne, 2004.

[3] R. Devillers, H. Klaudel, and M. Koutny. Petri net semantics of the finite π-calculus terms. *Fundam. Inf.*, 70(3):203–226, 2006.

[4] R. Devillers, H. Klaudel, and M. Koutny. A petri net translation of π-calculus terms. In *3rd International Colloquium on Theoretical Aspects of Computing'2006*, volume 4281 of *Lecture Notes in Computer Science*, pages 138–152. Springer Verlag, 2006.

[5] N. Henderson. *Formal Modelling and Analysis of an Asynchronous Communication Mechanism*. PhD thesis, University of Newcastle upon Tyne, 2004.

[6] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
[7] ISO. VDM-SL. Technical Report Draft International Standard, ISO/IEC JTC1/SC22/WG19 N-20, 1995.
[8] C. B. Jones. *Systematic Software Development using VDM*. Prentice Hall International, second edition, 1990. ISBN 0-13-880733-7.
[9] M. Koutny and G. Pappalardo. A model of bahaviour abstraction for communicating processes. In *STACS'99*, volume 1563 of *Lecture Notes in Computer Science*, pages 313–322. Springer-Verlag, 1999.
[10] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
[11] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–77, 1992.
[12] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes*. Cambrisge University Press, 2001.
[13] G. Schellhorn, H. Grandy, D. Haneberg, and W. Reif. The mondex challenge: Machine checked proofs for an electronic purse. In *FM*, pages 16–31, 2006.
[14] S. Stepney, D. Cooper, and J. Woodcock. An electronic purse: Specification, refinement, and proof. Technical monograph PRG-126, Oxford University Computing Laboratory, July 2000.
[15] B. Victor. *A Verification Tool for the Polyadic π-Calculus*. Licentiate thesis, Department of Computer Systems, Uppsala University, Sweden, May 1994. Available as report DoCS 94/50.
[16] J. Woodcock. First steps in the verified software grand challenge. *IEEE Computer*, 39(10):57–64, October 2006.

12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)
0-7695-2895-3/07 $25.00 © 2007 IEEE

8 Appendix: Model

$$World \stackrel{\text{def}}{=} Bank \mid \left(\prod_{i\in\{1,\ldots,n\}} CustNoP_i\right) \mid \left(\prod_{x\in\{1,\ldots,w\}} Wallet_x\right)$$

$$Bank \stackrel{\text{def}}{=} !\,(\boldsymbol{\nu} p)\big(reqp(c_i).\overline{c_i}\,p\big).P_p$$

$$CustNoP_i \stackrel{\text{def}}{=} (\boldsymbol{\nu} c_i)\overline{reqp}\,c_i.c_i(p).Cust_{ip}$$

$$Cust_{ip} \stackrel{\text{def}}{=} \left(\sum_{j\in\{1,\ldots,n\}} \overline{req_j}\,v \;\boldsymbol{+}\; req_i(v)\right).CustT_{ip}$$

$$CustT_{ip} \stackrel{\text{def}}{=} \left(\sum_{x\in\{1,\ldots,w\}} \overline{insert_x}\,p\right).\big(ok_{p}() \;\boldsymbol{+}\; f_{p}() \;\boldsymbol{+}\; \overline{ab_p}\,\big).Cust_{ip}$$

$$Wallet_x \stackrel{\text{def}}{=} insert_x(p).Wallet_{xp}$$

$$Wallet_{xp} \stackrel{\text{def}}{=} ab_{p}().Wallet_x \;\boldsymbol{+}\; insert_x(q).\big(\textbf{if } [\![auth(p,q)]\!] \textbf{ th } WalletC_{xpq} \textbf{ el } \big(\overline{f_p}\,|\overline{f_q}\,\big).Wallet_x\big)$$

$$WalletC_{xpq} \stackrel{\text{def}}{=} \Big(\big(ab_{p}().\overline{f_q} + ab_{q}().\overline{f_p}\,\big).Wallet_x \;\Big|\; \Big([\![KEY]\!].\textbf{if } [\![BAL]\!]_v \textbf{ th } WalletT_{xpqv} \textbf{ el } \big(\overline{f_p}\,|\overline{f_q}\,\big).Wallet_x\Big)\Big)$$

$$WalletT_{xpqv} \stackrel{\text{def}}{=} (\boldsymbol{\nu} c)\Big(\overline{p}\,c.c(S_p,R_p).\overline{q}\,c.c(S_q,R_q).\Big(\overline{S_p}\,v.\big(S_p(ACK).\overline{R_q}\,v.(\overline{ok_p}\,|\overline{ok_q}\,) \;\boldsymbol{+}\; (\overline{f_p}\,|\overline{f_q}\,)\big)\Big)\Big).Wallet_x$$

$$P_p \stackrel{\text{def}}{=} [\![V_p]\!] \;\Big|\; [\![L_p]\!] \;\Big|\; !\,(\boldsymbol{\nu} sr)\Big(p(c).\overline{c}\,sr.\big(s(v).[\![V_p - v]\!].(\overline{s}\,ACK \;\boldsymbol{+}\; [\![L_p + v]\!]) \;\boldsymbol{+}\; r(v).[\![V_p + v]\!]\big)\Big)$$