An operational model of QuickPay

Pieter H. Hartel,Jake Hill[*] and Matt Sims [*]

Introduction

QuickPay is a micro payment scheme with prepayment [1, 6]. The customer must register with the broker to obtain an electronic *carnet* with *value* tokens before a sale may take place. It is possible to acquire additional value tokens, or to refresh the value tokens at any time. A merchant must also register with the broker to obtain an electronic *till* with *authentication* tokens. This will enable the merchant to validate the tokens presented by the customer. Once customer and merchant are registered, a sale may proceed as follows. First the customer presents one or more value tokens to the merchant. The merchant then authenticates himself [†] with the broker and presents the customers value tokens to the broker for validation. The broker decides whether the customers tokens are valid. If the merchant is satisfied that the tokens are valid, goods/services may be delivered. The customer takes delivery but does not receive a receipt.

The QuickPay philosophy is to make transactions cheap in two ways. Firstly, offline payments are allowed although they are less secure than online payments. Secondly, payment does not require cryptographic computations. This is achieved by using real (as opposed to pseudo) random numbers as tokens. A sequence of random numbers is generated, encrypted and transferred when the customer or the merchant register with the broker. Creating the random numbers is efficient, as a hardware device (a noisy diode) is used rather than an computationally intensive algorithm. Encrypting and transmitting a sequence of random numbers can be relatively inefficient. However, this is only done during registration; during payment transactions only a single random number is transferred in clear. Systems that use cryptographic computations during every transaction, such as Millicent [2] and Mini-Pay [5], are inherently less efficient than QuickPay but possibly more secure.

Micro payment systems raise a number of interesting questions because they differ from normal payment systems. The present paper makes the following contributions to the understanding of micro payment systems in general, and to that of QuickPay in particular:

- to introduce an operational model for the QuickPay protocols, which allows real life scenarios to be studied at an appropriate level of abstraction.
- to investigate the correctness of an essential optimisation to the basic protocol, which allows multiple token payments to be replaced by a single token payment.

The operational model of QuickPay is written with the aid of the `latos` [3] tool, which provides type checking and animation of specifications. The type checking facility of the tool has been used to avoid inconsistencies in the model, and the animation facilities have been used to explore various transaction sequences.

The next section briefly presents the prototype implementation of QuickPay. Section sketches the model of QuickPay; the complete specification is given in the full paper [4]. Section presents a sample scenario, showing how QuickPay transactions can be studied using the model and the `latos` tool. Further scenarios are explored in full paper. Two problems and a number of solutions to one of the problems are briefly sketched in Section ; the full paper giving the details and the proof. The last section presents conclusions and discusses further work.

Prototype

The QuickPay prototype has been tested on making payments over the internet. The prototype works as follows. The customer first starts her carnet application, which puts up a window to inform the customer about her balance as shown below.

The customer also starts up an unrelated application that might require payment, such as an intelligent agent that is going to make some purchases on her behalf. For the purpose of this example we will use a web browser, which is being used to select a page of information from a web server, for example `http://www.merchant.com`. Figure 1 shows the information provided by the server, as it appears on the customers work station. The Web page she is looking at is actually a schematic diagram of the QuickPay prototype. The diagram shows the three main parties and the (TCP/IP) connections between them.

If a page contains links that require payment, the web browser sends an http request to `http://till.merchant.com/`. The merchants till application opens a TCP/IP connection to the carnet at the customer site (identified by her IP address) and the carnet puts up a window asking the customer to confirm the sale. When the customer agrees, the merchant receives the tokens over the TCP/IP connection and clears them with the vault application at the brokers site. The carnet can be customised to designate merchants as permanently trusted, or trusted for the current session. Such merchants can help themselves to tokens without confirmation from the customer.

[*] BT Laboratories, Martlesham Heath, Ipswich, UK

[†] We refer to a customer using the words 'her' or 'she' and we refer to a merchant using the words 'his' or 'he'.

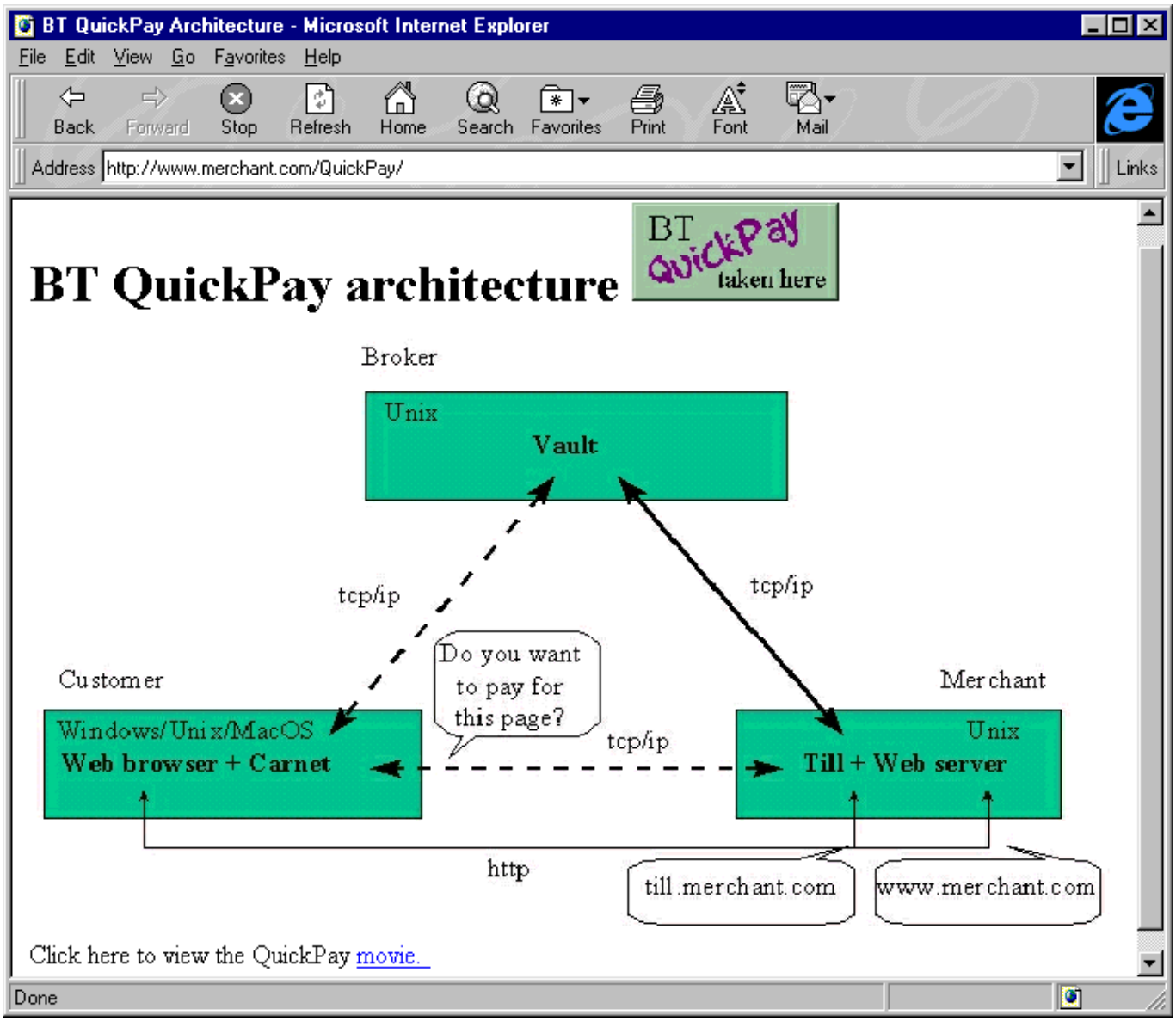

Figure 1: The architecture of the QuickPay prototype.

If the turn over of the merchant is high, a permanent TCP/IP connection (solid arrows) is used, otherwise a transient connection would be better. When the merchant is satisfied that the tokens have cleared, the till sends the page to the customer as a reply to the original http request.

The model to be described in the next section takes into account the core aspects of the implementation, but abstracts away from as much detail as possible. This strategy makes it possible to concentrate on the essentials, keeping the model simple and elegant. Once finished, it would be possible to refine the model so as to take on board more detail. Ultimately this process of refinement would lead to a finely detailed model that is able to describe every aspect of the implementation. The present work should be considered a starting point for the refinement process.

The model takes into account the transactions that rely on the TCP/IP connections, but abstracts away from the actual protocol implementation. The model also takes into account the three parties but it is not concerned with the Web browser/server. These components can be replaced by other client/server applications and are therefore not relevant to the model. The model abstracts away from the internal representations of data and messages in the carnet, till and vault applications.

Model

The model represents the QuickPay book keeping by a state, and the messages exchanged by the QuickPay protocols are represented by a list of transactions. Each transaction causes a transformation to be applied to the state, modelling the change in the book keeping as a result of a message exchange in the real system.

The state of the model is described by a number of data type definitions. The transactions that can take place are described by a set of logical inference rules operating on that data. Transactions and state are bound together in a configuration, which records the present state of the system as well as the sequence of transactions that have yet to take place. The collection of inference rules defines a relation over configurations. The model is animated by computing the transitive closure of the relation.

In subsequent sections we introduce the state (s), the transactions (tr), the relation ($\stackrel{qp}{\Rightarrow}$) and its closure ($\stackrel{qp}{\Rightarrow}*$).

State

The state s of the model is represented by a 3-tuple consisting of the book keeping of the customers (cs), brokers (bs), and merchants (ms).

$$s \equiv (cs,\ bs,\ ms);$$

This rather innocent looking tuple represents the major abstraction of the model with respect to the prototype. The latter distributes the information with the protocol taking care that appropriate information is exchanged between the parties, but no more. The model in principles allows unlimited access to information. However, the model has been designed in such a way that it is easy to see (and prove) where and when information is accessed.

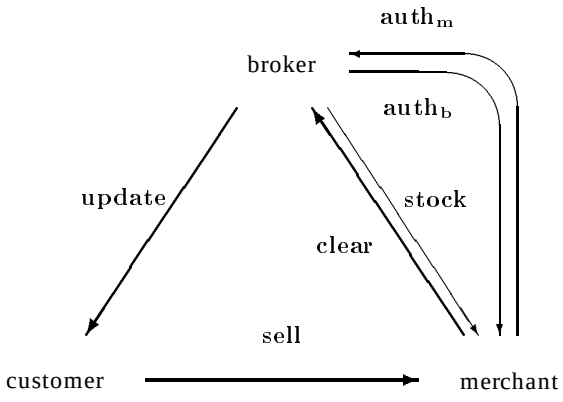

Figure 2: The QuickPay parties and transactions. Thick arrows represent value token transfer and thin arrows represent authentication token transfer.

The full paper gives the details of the mappings cs (from customer ids id_c to customer records), bs (from broker ids id_b to broker records), and ms (from merchant ids id_m to merchant records).

Transactions

The model represents transactions as separate actions and reactions. This permits actions be modelled whilst the reaction is not forthcoming. Each transaction is labelled and carries a number of parameters to identify the parties involved. Some transactions require further information, such as a token count n. The definitions below represent 'control' information of the messages transmitted in the actual prototype; we abstract away from actual 'payload' of the real messages (i.e., the lists of tokens).

$$\begin{aligned} tr \equiv\ & \mathbf{update}(id_c,\ id_b,\ n) \mid \mathbf{sell}(id_m,\ id_c,\ n) \mid \\ & \mathbf{clear}(id_b,\ id_m,\ id_c) \mid \mathbf{stock}(id_m,\ id_b,\ n) \mid \\ & \mathbf{auth}_m(id_m,\ id_b) \mid \mathbf{auth}_b(id_b,\ id_m); \end{aligned}$$

Figure 2 illustrates which parties are involved in each of the transactions. Value tokens (indicated by thick arrows) flow from the broker to the customer (**update**) and then via the merchant (**sell**) back to the broker (**clear**). The authentication tokens (thin arrows) flow from the broker to the merchant (**stock** and $\mathbf{auth}_b$) and from the merchant to the broker ($\mathbf{auth}_m$).

The notion of a configuration augments the (static) state s with a (dynamic) list of transactions [tr]. We can now formulate the model of QuickPay as a relation $\stackrel{qp}{\Rightarrow}$ over configurations. The type of the relation is:

$$\stackrel{qp}{\Rightarrow} :: \langle [tr],\ s \rangle \leftrightarrow \langle [tr],\ s \rangle;$$

The full paper gives the complete rules defining the effect on the state of the six transactions.

Closure

The model is animated by computing the transitive closure $\stackrel{qp}{\Rightarrow}*$ of the relation as shown by the function animate below. This function takes an initial configuration and delivers a list of configurations showing the remaining transactions and the state of the system after each transaction. The initial configuration is prepended to the result to show the starting point of the animation.

```
animate          :: ⟨[tr], s⟩→[⟨[tr], s⟩];
animate⟨trs, s⟩= ⟨trs, s⟩ : ⟨trs, s⟩ ⇒qp *;
```

A complete model of QuickPay has now been sketched. The next section presents an example of use.

A scenario: double spending

QuickPay has been designed to be customer friendly. She can refresh her carnet at any time, and even if she looses the carnet, she promptly receives fresh tokens. We will now use the model to study a scenario which mis-uses this facility. Suppose that a customer makes a purchase, and then immediately refreshes her carnet. Refreshing the carnet will invalidate the tokens just offered to the merchant, so that the latter is unable to clear the tokens. The customer will only benefit from her bad behaviour if the merchant delivers the goods / services before clearing the tokens. The merchant may wish to do so in order to clear tokens in batch, and thus to amortise the cost of a clear transaction over a larger number of tokens.

An animation as computed by the animate function consists of a series of snapshots, which show in detail how each transaction alters the state of the system. To save space customers or merchants with empty token lists are suppressed. Similarly, if the book keeping of any customer, broker or merchant is unaffected by a transaction then that information is suppressed.

The first step initialises the list of random numbers for the broker (called bank1), as shown below:

```
bank1    [10 11 12 13 14 15 16 17 18 19]
```

$\rightarrow$ update(alice, bank1, 2) $\rightarrow$

After the update transaction, the carnet of customer alice contains two value tokens, $[10, 11]$. The brokers vault contains the duplicates and shows that the customers budget is now 98:

```
alice    [10 11]
bank1    [12 13 14 15 16 17 18 19]
         alice    ([10 11],98)
```

$\rightarrow$ stock(shopx, bank1, 4) $\rightarrow$

After the stock transaction, the till of the merchant shopx contains four authentication tokens, $[12, 13, 14, 15]$. The brokers vault contains the duplicates and shows that the merchants account is 0:

```
bank1    [16 17 18 19]
         alice    ([10 11],98)
         shopx    ([12 13 14 15],0)
shopx    [12 13 14 15]
```

$\rightarrow$ sell(shopx, alice, 1) $\rightarrow$

The merchant has received the token 10 as payment from the customer:

```
alice    [11]
shopx    [12 13 14 15]
         alice    [10]
```

$\rightarrow$ update(alice, bank1, 0) $\rightarrow$

The customer updates her carnet, receiving two fresh tokens $[16, 17]$. The customer's budget is still 98. The broker's record also shows the new state of the carnet.

```
alice    [16 17]
bank1    [18 19]
         alice    ([16 17],98)
         shopx    ([12 13 14 15],0)
```

$\rightarrow$ sell(shopx, alice, 1) $\rightarrow$

The customer makes another purchase, this time spending token 16. The merchant holds the old, invalid token 10 as well as a new, valid token 16:

```
alice    [17]
shopx    [12 13 14 15]
         alice    [16 10]
```

$\rightarrow$ auth_{m}(shopx, bank1) $\rightarrow$

The broker accepts that the merchant is authentic by agreeing that 12 is the first authentication token:

```
bank1    [18 19]
         alice    ([16 17],98)
         shopx    ([13 14 15],0)
shopx    [13 14 15]
         alice    [16 10]
```

$\rightarrow$ auth_{b}(bank1, shopx) $\rightarrow$

The merchant also accepts that the broker is authentic, using token 13:

```
bank1    [18 19]
         alice    ([16 17],98)
         shopx    ([14 15],0)
shopx    [14 15]
         alice    [16 10]
```

$\rightarrow$ clear(bank1, shopx, alice) $\rightarrow$

At this stage the clearing fails because $[16, 10]$ does not match $[16, 17]$. The merchant account is not credited.

Problems and solutions

One of the main concerns in the design of QuickPay has been to make the protocols as efficient as possible. This

has resulted in a design that uses random numbers instead of compute intensive cryptography and small messages instead of large ones. The strive for efficiency has brought with it some (potential) problems that we should like to address in this section.

Asymmetric authentication

We discovered a hitherto unknown problem whilst building the model of QuickPay. The stock transaction serves to provide the merchant with authentication tokens. These tokens are used both to authenticate the broker with the merchant and vice versa. It is conceivable that a broker may take advantage of the knowledge how these tokens are computed to trick the merchant. This danger would not arise if both the broker and the merchant would create their own list of tokens, which they would then use to authenticate the other party. We have not found a practical example that exploits this weakness.

Selecting lossy compression

To reduce the amount of data transmitted during a multiple token transfer, the prototype implementation of QuickPay optimises payments of $n > 1$ value tokens in the following way. Instead of transferring a list of tokens $[v_1, \ldots v_n]$ the implementation transfers just the last one, v_n. We have termed this the *selecting lossy compression*, because the optimisation compresses by selecting a token.

When clearing, the broker has a duplicate of the customers carnet. The broker is thus able to look the token v_n up in the duplicate. Normally, the token will be found at the n-th place, thus confirming both that the token is valid, and that it represents n tokens.

In the full paper, we give a scenario that shows why the optimisation is not correct.

Whilst building the QuickPay prototype we (re) discovered that the selecting lossy compression optimisation was incorrect. We also found a different optimisation (adding lossy compression) which causes the scenario above to behave correctly. To study this new optimisation we applied it to the model, uncovering another problem. This lead us to a generalisation of compressing tokens, as well as a range of optimisations.

The full paper explores these issues in detail, giving necessary conditions on the compression function so that the optimised protocol can be proved correct with respect to the unoptiomes protocol.

Conclusions and future work

A formal model of the QuickPay micro payment system has been built that makes it possible:

- to clearly and concisely describe the transactions of the system. The core transactions of the protocol have been fully specified.
- to animate sample transaction sequences so as to illustrate the concepts and to explore scenarios of incorrect uses of the system. Seven such sequences have been presented.
- to identify potential problems and to study possible solutions to these problems. A new problem has been identified and an old problem (selecting lossy compression) has been re-discovered. Various solutions to the old problem are given, ranging from a provably correct solution (lossless compression) to an efficient solution (adding lossy compression).

The model has helped us to analyse the behaviour of the protocols, leading to the conclusion that QuickPay is:

- attractive for the customer. Even if she looses her tokens, they will be promptly replaced.
- attractive for the broker. Like all pre-paid schemes, the broker is able to dispose of the (real) money of the customer for a certain period of time.
- less attractive for the merchant. All problems that we have studied are to the disadvantage of the merchant. However, presently a merchant providing electronic services receives voluntary contributions at best. With QuickPay the merchant might expect an increase in revenue.

The model has been formulated rather abstractly, so that in the prototype implementation there may well be problems that are not captured by the model. The model could be extended to cover more detail.

The model could also be extended to study the costs of the transactions and to compare this costs to that of the tokens being processed.

The model could be extended to capture histories of customer and merchant behaviour. Such histories could be used to decide if and when clearing of tokens is needed, as well as other possible optimisations to the protocol. The histories could also be used to assess to what extent the merchant may take advantage of the knowledge of customer behaviour.

Finally, the model could be generalised to capture other micro payment systems, such as Mini-Pay, Millicent and Payword / Micromint. Such a general model would enable different micro payment systems to be compared on the same formal footing.

Acknowledgements

This work was supported by a short-term research fellowship from BT laboratories, Martlesham Heath, UK, contract number STRF97 / 33. The help and support of Michael Butler, John Regnault and Tim Hart is gratefully acknowledged.

References

1 UK BT Research laboratories, Martlesham Heath. Transaction system. In *GB Patent application no. 9624127.8*, 1996.

2 S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. G. Sobalvarro. The millicent protocol for inexpensive electronic commerce. In *4th International World Wide Web Conf.*, pages 603–618, Boston, Massachusetts, Dec 1995. World Wide Web Journal. `www.research.digital.com/ SRC/ staff/ msm/ bio.html`.

3 P. H. Hartel. LATOS – a lightweight animation tool for operational semantics. Technical report DSSE-TR-97-1, Dept. of Electr. and Comp. Sci, Univ. of Southampton, England, Oct 1997. `www.ecs.soton.ac.uk/ ~phh/ latos.html`.

4 P. H. Hartel, J. Hill, and M. Simms. An operational model of QuickPay. Declarative Systems & Software Engineering Technical Reports DSSE-TR-98-4, Dept. of Electr. and Comp. Sci, Univ. of Southampton, England, 1998. `www.dsse.ecs.soton.ac.uk/ techreports/ 98-4.html`.

5 A. Herzberg and H. Yochai. Mini-Pay : Charging per click on the web. In *6th WWW conference*, Santa Clara, California, Apr 1997.

6 J. Hill and M. Sims. QuickPay: Prepaid micropayments. Technical report in preparation, BT Research laboratories, Martlesham Heath, UK, 1998.