

MOBY DICK

The Mobile Digital Companion

LTR 20422

Deliverable II.3.3

Demonstrator II: Digital cheque-book

July 1997

University of Tromsø: Tage Stabell-Kulø

University of Twente: Arne Helme

In this section we will describe the overall design of the demonstrator and the technical details of its implementation. We start out by presenting an overview of the demonstrator as to what will be demonstrated and why this is relevant. Then we will embark on a technical description of the solution.

1 Overview of the demonstrator

We have realized the digital cheque-book demonstrator in a more general setting in which contacts are the basis of a transaction. In a setting in which for example cheques are used for buying goods from a store, the cheque can be (part of) the contract. Such a setting with a digital cheque-book involves three transactions. The first transaction is between the Bank and the Client so that the Client can obtain valid cheques. Then there is a transaction between the Client and the Merchant in which something can be purchased. The final transaction is between the Merchant and the Bank, so that the Merchant can receive the money from the Bank.

Our suggested solution to the problem of securely transaction effectively divides the Pocket Companion into two: the trusted part (called Trusted Computing Base (TCB)) and the non-trusted one. More details of this architecture can be found in chapter 4 of deliverable II.3.1. This demonstrator shows that the non-trusted part can be realised.

The sequence of events as shown in the demonstrator are as follows:

- The demonstrator application is started on the client machine. Which service provider to contact is given on the command line. This start-up sequence would in a real scenario probably be replaced by following a WWW link.
- Then, by means of untrusted software the contract is negotiated. As usual in the 'real world' contracts are detailed and specify everything of interest. The cheque from the bank can be part of the contract. This phase could, for example, be implemented in a WWW browser. For the demonstrator, a contract is designed at the server, containing the users' name and other details. This contract is then sent to the users machine.
- In order to obtain the user's signature on the contract, the untrusted software gives the contract to the security module.
- Now, the security module is in control of the TCB and displays the contract to the user in a dedicated window on the screen, together with two buttons, labelled 'Sign' and 'Refrain', respectively.
- If the users presses 'Sign' the security module opens up another window in which the user can type his password.
- The *image* of the window as seen by the user, with borders to outline the contract properly, is saved for future reference (it can be displayed at will) and a hash of the image is signed with the user's private key. The security module has completed its task and untrusted software can once again take over control.
- The image, followed by the signed hash of it, is returned to the service provider.

The outcome is that the service provider can prove that the user entered the contract while neither the secret key of the user (stored in the smart card) or the PIN to release the key (typed on the keyboard) has left the TCB. This implies that the user can securely sign contracts even if untrusted software is running on the machine.

Up to this point, we have glossed over an important aspect, the binding between a user and an encryption key. At present it is unclear how one in the future will establish this binding, but

we notice that X.509 has been chosen in the implementation of SSL (which is supported by popular browsers such as Netscape and Microsoft Explorer). This aspect of the security environment is outside the scope of this demonstrator, but we note that whatever solution *the user* wants to use, the security module as such should be unaffected.

2 Technical description of the demonstrator

In this section we will describe the realization of the demonstrator with particular emphasis on the messages that are exchanged.

- 1 The contract is negotiated. This also includes identifying the user for two purposes. First, the name must be included in the contract, and the service provider must ensure that the 'correct' user is negotiating. The latter includes sending the user's public key together with a chain of certificates linking to some common trusted third party. This is the domain of the service provider, and it might, for example, require that some certification verifier is on-line.

Also, either the user or the service provider might require that all communication is encrypted for privacy. If this is the case, a session key must be exchanged.

The user sends his name together with his public key (in PGP syntax) together with certificates. This functionality is provided by a library we have written as part of the security effort in the first phase.

- 2 The contract is formulated and sent to the client where it is delivered to the security module for signing. The contract is sent as a human readable text in order for the user to be able to read.
- 3 The string constituting the contract is given to the security module. At this point, the security module would switch into 'secure mode'. A new window is displayed with the contract, the date and time of day and two buttons (`Sign` and `Refrain`). When the user presses the `Sign` button the 'contents' of the window is saved in a buffer. A new window is opened (by the security module) and the user is asked to supply the PIN to unlock the secret key in the smartcard. Since the demo runs without a smart card module, this PIN is used to decrypt the secret components of a secret (RSA) key. The buffer is then digitally signed with the secret key of the user, and the result stored in a file for later reference.
- 4 The output from the security module (the signed image) is sent to the service provider. Sending of the signed image can safely be left to the untrusted part (outside the TCB).

As we have seen, the exchange of messages is very simple. This is so since the Trusted Computing Base (TCB) is designed to bridge the gap usually bridged by elaborate authentication protocols. By reading the contract before signing, the user can verify the correctness of the participants, and may even reply on the contract to have a specific 'look' (tickets to ride a train should look like a train ticket). In this stage in the project we assume that contracts are presented to the Pocket Companion as pure text. However, we envision that some page layout language (Postscript or PDF) can be used to give contracts the 'look and feel' one wants. This needs to be investigated further as graphical languages might introduce subtle issues about representation.

Regarding authentication of the service provider, we believe that the best we can do is to either include the chain of certificates in the contract, or having the security module displaying them as part of the signing procedure. This also needs to be investigated further. In particular, if one decides that X.509 is too large and complex, other feasible alternatives must be found.

3 Conclusion

The demonstrator, together with the solution to the problem of signing contracts and the assessment criteria, is a good indication that we can overcome the security problems related to a Personal Companion.