Formal Verification and Simulation of the NetBill Protocol Using SPIN[1]

Jose Garcia-Fanjul, Javier Tuya and Jose Antonio Corrales

Computer Science Department
Campus de Viesques at Gijon
Oviedo University

{fanjul | tuya | ja}@lsi.uniovi.es

Abstract. Electronic Commerce and Internet are deeply changing the way goods are exchanged, but there is still short confidence among users about the security of their data. The application of formal techniques to the modelling and design of electronic commerce protocols should help improving their reliability and so enhance the choices of these new technologies. In this paper basic concepts about security and payment protocols are described, the NetBill protocol is graphically specified and its behaviour in terms of reliability is formally verified using the SPIN tool.

Keywords. SPIN, Verification, Formal Methods, Payment Protocols, NetBill, Electronic Commerce, Internet

1 Introduction

Internet, and more precisely the WEB, leads electronic commerce to its full extent, allowing solutions ranging from a single commerce to the creation of an online technical service. Furthermore, the benefits are higher both from the point of view of the merchant and the consumer. This is the reason for the appearance of a large amount of tools that facilitate the creation of Web based sales services [2] and different and

[1] This work has been funded by the "III Plan de Investigacion del Principado de Asturias" (Spain) and SATEC S.A. under project CYBERMERCADO (PA-CYR96-03)

innovative payment methods. However, only 11.1% of the Internet users go shopping online and 60% of them claim to be worried about the privacy of their data [8].

Formal techniques of specification and verification, that have soundly proved their capacity to analyse communication protocols (see [10] and [7]) may be used to better electronic payment protocols and so recover the users confidence.

The present paper shows our work verifying the NetBill protocol. Our approach, as in [11], is the combination of two notations: widely spread graphical tools taken from Object Modelling Technique (OMT, defined in [9]) to support the development of a model and a model checker (SPIN) to prove the necessary properties.

As a summary of the paper, a section of basic concepts about security and payment protocols (section 2) is included. NetBill is shortly commented and described (section 3) and later modelled and graphically specified in section 4; at the end of this part, the restrictions and design choices for the model are enumerated. After that, in section 5, the implementation issues of the model are discussed. The logical properties to check are formally described and the results of the verification using the SPIN tool are exposed in section 6. Then we propose (section 7) a change to the merchant behaviour, trying to attack the protocol. This attack is not successful, so we redo the protocol description to make it fail and simulate the flaw with SPIN. The paper ends with sections about related information, conclusions and future work.

2 Basic Concepts of Security and Payment Protocols

The security of a computer system may be viewed from two points: **safety** or the possibility that the system may harm the environment and **security** or avoiding the chance that the environment may harm the system. Thinking about security, we should bear in mind these different aspects:

- **Authentication**: establishing the identity of the users of the system
- **Privacy**: assuring that data can not be viewed by non-authorized users
- **Integrity**: avoiding data to be modified by non-authorized users
- **Non-repudiability**: the capacity to prove the identity of the source or the receiver of a message

Assuming that cryptographic techniques assure the fulfilment of the aspects aforementioned, other security characteristics arise at a higher level and specific to payment protocols. For example, Tygar [12] has described the importance of **atomicity**, as the property consisting on logically joining some operations in such a way that all of them or none are to be carried out. Taking this concept to electronic commerce, money atomicity is described as “money can’t be created nor destroyed by electronic commerce protocols”. For goods atomicity, it should be assured that “the merchant receives payment if and only if the consumer receives the goods”. And for certified delivery: “merchant and consumer must hold non refutable proof of the nature of the delivered goods”.

3 NetBill Protocol

NetBill [3] is an electronic commerce protocol designed to be used in commercial transactions of information through Internet. It implies three parties: consumer, merchant and bank; and can be divided into two stages (goods deliver and payment) and eight steps that are graphically represented in **Figure 1** and **2**.

The consumer (throughout the paper and to avoid confusion we will use male to denote the consumer and female to denote the merchant) begins the deliver stage with the message “Goods Request” –GR, see **Table 1**-. The merchant answers with the ordered goods encrypted with a key k (message EG): it’s important to understand that up to the moment the consumer has the goods but can not make any use of them, as he has not yet received the key for them to be deciphered. After that, the consumer must deliver an electronic payment order that the merchant redirects to the bank along with the key k, this is the start of the payment stage. The bank checks whether the transaction can be completed or not. In the former, the bank sends a “Payment Slip” that includes the key k. If there are no funds the message sent is called “No Payment”. Whatever the message may be, the destination is the merchant, who must forward it to the consumer.

If the consumer does not receive the transaction result, the protocol shows he can directly query the bank. The appropriate response will be PS, NP or a message called “No Record” in case the transaction never reached the bank.

Table 1. Messages passed between parties

Id	*Message Description*
GR	"Goods Request"
EG	"Encrypted Goods"
EPO	"Electronic Payment Order"
PS	"Payment Slip"
NP	"No Payment"
TE	"Transaction Enquiry"
NR	"No Record"
EEPO	"Endorsed EPO"

4 Logical Model

In our logical model, each NetBill party is represented by an object, and an event flow diagram (**Figure 1**) and event trace (**Figure 2**) are used to view the general characteristics of the protocol.

Figure 1. Event Flow Diagram

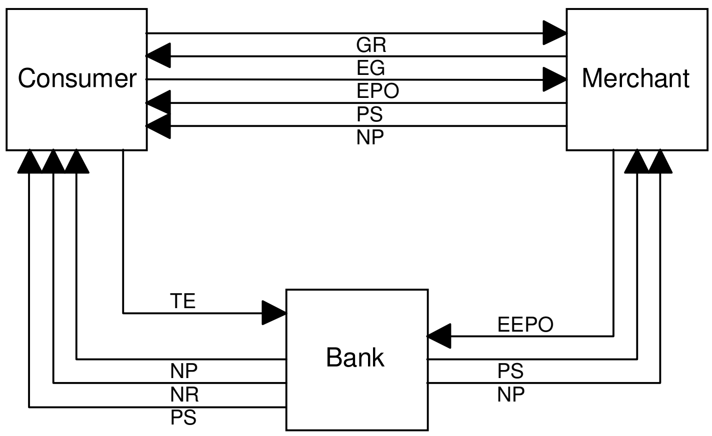
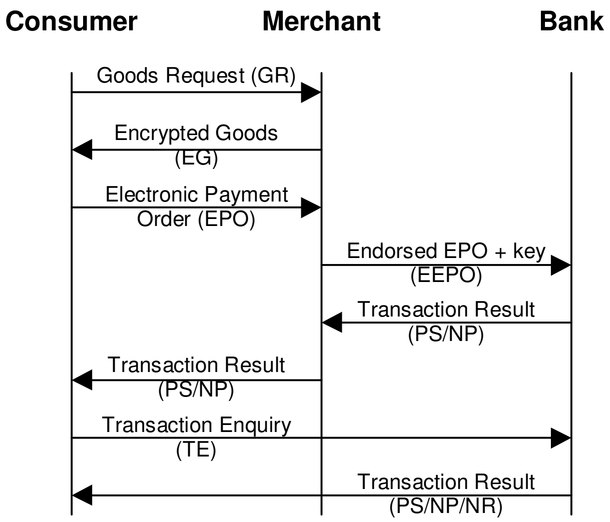
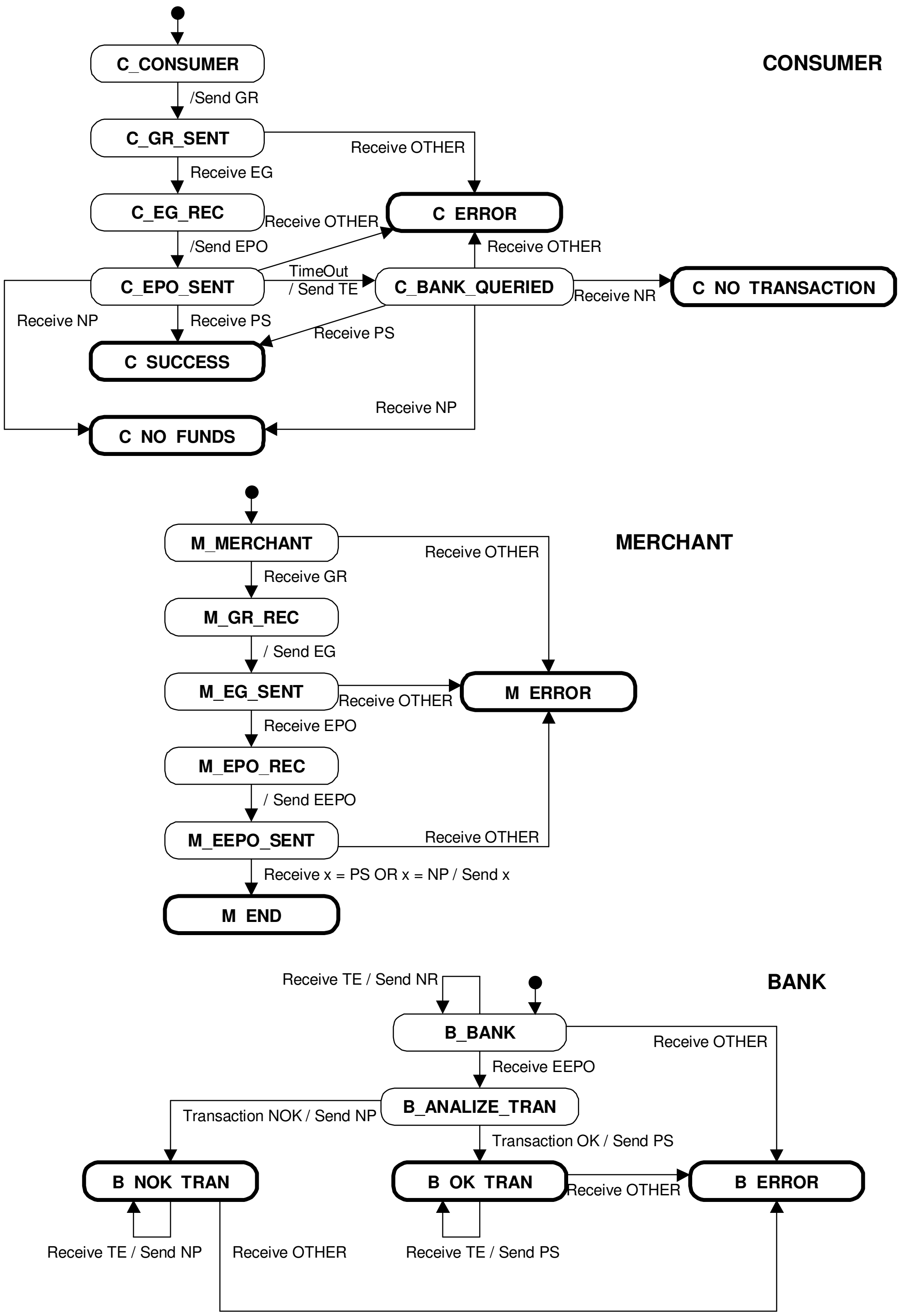


Figure 2. NetBill Steps (Event Trace Diagram)



To describe the insight details of behaviour, state transition diagrams are used (see **Figure 3**). Note we label the transition between two states with a pattern “A/B” where A is the event that makes the transition happen and B is an action that takes place when the transition is committed.

Figure 3. State transition diagrams describing the behaviour of the three objects



Discussion

The use of graphical notations always implies a reduction of complexity that makes workable the study of possible attacks or misuses of the protocol. As an example, consider the last two steps of NetBill: The consumer is waiting for the transaction result from the merchant, but a timeout occurs, so he sends a query about the state of the transaction to the bank. In case the timeout length is less than the response time of the merchant, the consumer would receive, in the end, two messages indicating the transaction result: one from the merchant and one from the bank. This is not a problem for the protocol: the consumer has to choose one of the messages because they should be the same under the assumptions we make, otherwise cryptoalgorithms would show the difference. But this is easier to track from a graphical specification (to reach state "C_SUCCESS" there are two transitions available) than the textual one provided in the original description of the protocol.

Note that certain restrictions and design choices apply to the model:

- It is thought to execute a single run of the protocol
- We try to discover mistakes inherent to the protocol description, so the parties of the protocol are modelled to be honest and communications are flawless. Anyway, the design is open to include dishonest parties or the appearance of intruders (see section 7 for a simulation of a protocol error based on changes to the definition of the protocol steps and the behaviour of the parties)
- The granularity hides the low-level security details of the cryptographic protocols: for example, every message will be accepted as coming from the correct source and we will not care about the contents of the message being sniffed. This decision is made to allow us verify a high level property as goods atomicity is
- The NetBill protocol involves a previous stage and two more steps to negotiate a price for the goods, but security characteristics are inherent to the steps we denote in the paper

5 Implementation of the Model

The use of graphical specification tools may be valuable to detect misuses or make the documentation of a protocol easy to understand to others. In addition, model checkers as SPIN allow the simulation and formal verification of logical properties.

To translate the logical model shown in section 4 into a SPIN model, we build a process for each object. Thus we have processes “Consumer”, “Merchant” and “Bank”.

Communications modelling in verification tools is usually accomplished by channels implemented as queues. In [4] the model represents a channel between every two processes and direction of communication. We believe that this approach is difficult to understand: for n processes there would be n!/(n-2)! channels involved (for example six channels for three processes or twelve for four). Furthermore, the ability to simulate an intruder to the system through the Internet is easier to implement if we centralize every message passed in a single process.

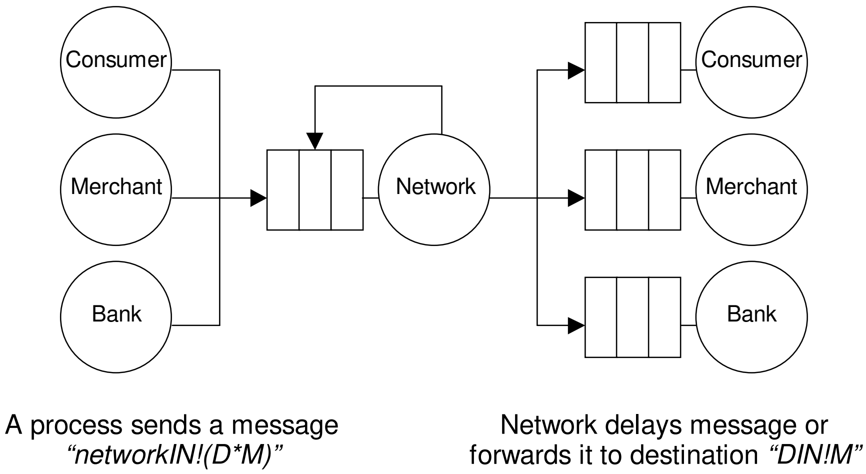
Thus, the proposed and implemented model is made up of an input channel for each process, that way the delivery of the messages is made to the channel corresponding to the party, as if it was its mailbox. Besides, we model the Internet as another process named “Network” that will receive and send messages from and to the parties of the protocol.

In our implementation, identifiers consumerIN, merchantIN, bankIN and networkIN suit to the channels of the processes. So when it is written in PROMELA (the language of the SPIN model checker) “consumerIN?x”, it means that the consumer receives a message that is stored in a variable called x. Similarly, “networkIN!x” shows that a message x is sent to the “mailbox” of the network.

It has been said that messages are sent between any two parties of the protocol through the network process. To accomplish this, we need an addressing architecture that is made up of “message constants” defined for every message of the protocol as in **Table 1**, and “party constants” identifying the three tiers. For example, GR is a constant defined for the message “Goods Request”; CON, MER and BAN identify the consumer, merchant and bank parties, respectively.

Any party that needs to send a message to another multiplies the message constant with the appropriate destination constant, delivering the result to the “network” process. If we call a generic party constant “D” and a generic message constant “M”, the way to send M to D is “networkIN!(D*M)”. When “network” receives that, it generates a new message in the channel of D “DIN!M” (see **Figure 4**).

Figure 4. Communication between parties through the “Network” process



Note that “network” could have a behaviour on its own to model the presence of potential attackers: query, loss, modification and/or generation of messages (in [1] a model with these characteristics is described). Anyway, at this level of granularity we have designed it as a message distributor, implementing the addressing scheme described above so it can single out the destination of the messages with the only added functionality of possibly delaying them.

Here we include the translation of the state transition diagram for the consumer process included in **Figure 3** to PROMELA.

```
proctype consumer ()
{
/* Consumer states */
#define C_CONSUMER 0
#define C_GR_SENT 1
#define C_EG_REC 2
#define C_EPO_SENT 3
#define C_SUCCESS 4
#define C_ERROR 5
#define C_NO_FUNDS 6
#define C_BANK_QUERIED 7
#define C_NO_TRANSACTION 8

/* Variables needed ********************/
short state, timeoutC;
int x;

/* Initialisations *********************/
state = C_CONSUMER;

/* Transitions *************************/
do
```

```
::  if  /* timeoutC will be randomly TRUE or FALSE
            to simulate timeouts*/
      :: timeoutC = FALSE
      :: timeoutC = TRUE
    fi;

    if
      :: state == C_CONSUMER ->
         atomic { networkIN!(MER*GR); state = C_GR_SENT }

      :: state == C_GR_SENT ->
         consumerIN?x;
         if ::x == EG ->
           atomic { state = C_EG_REC; EGConsumer = TRUE }
         ::else ->
           state = C_ERROR
         fi

      :: state == C_EG_REC ->
         atomic { networkIN!(MER*EPO); state = C_EPO_SENT }

      :: state == C_EPO_SENT && timeoutC == FALSE ->
         consumerIN?x;
         if ::x == NP ->
           state = C_NO_FUNDS
         :: x == PS ->
           atomic { state = C_SUCCESS; PSConsumer = TRUE }
         ::else ->
             state = C_ERROR
         fi

      :: state == C_EPO_SENT && timeoutC == TRUE ->
         atomic { networkIN!(BAN*TE); state = C_BANK_QUERIED}

      :: state == C_BANK_QUERIED ->
         consumerIN?x;
         if ::x == PS ->
           atomic { state = C_SUCCESS; PSConsumer = TRUE }
         :: x == NP ->
           state = C_NO_FUNDS
         :: x == NR ->
           state = C_NO_TRANSACTION
         :: else ->
           state = C_ERROR
         fi

      :: state == C_SUCCESS || state == C_ERROR
         || state == C_NO_FUNDS || state == C_NO_TRANSACTION ->
         break /* Final states: processing ends */

    fi;

od;
} /* consumer proctype */
```

6 Properties to be Satisfied by the Model and Results of the Verification

The property we want to check is goods atomicity, so we have to formalize the previously included statement “the merchant receives payment if and only if the consumer receives the goods”. There are, trivially, two parts involved in the property:

we must tell (1) when the merchant "receives payment" and (2) when the consumer "receives the goods". For the former (1), the Payment Slip message includes the amount deposited and is signed by the bank, so the merchant is paid the moment she receives the message PS from the bank. Note that it does not suffice that the money is deposited, as the consumer may request a refund and the merchant could not prove, if she has not received message PS, the agreement of the consumer to the transaction. For the latter (2), PS also includes the key k, so the consumer receives the goods once he has the encrypted goods (EG) and the key (PS).

Formally speaking, if *M* is the PROMELA model for the NetBill protocol, *E* the set of all reachable states for the model, e_i each state in *E* and $\sigma : e_1, e_2, ..., e_i, ...$ a sequence of states that complies with the model *M*, then we will define the following predicates in the set of states *E*:

$$\textbf{Def. } \forall e_i \in E,\ PSC(e_i) \Leftrightarrow \qquad (1)$$
$$\exists \sigma\ \exists e_k \mid e_i, e_k \in \sigma, e_k \le e_i \text{ in } \sigma \text{ and } e_k \text{ verifies ConsumerIN?PS}$$

$$\textbf{Def. } \forall e_i \in E,\ EGC(e_i) \Leftrightarrow \qquad (2)$$
$$\exists \sigma\ y\ \exists e_k \mid e_i, e_k \in \sigma, e_k \le e_i \text{ in } \sigma \text{ and } e_k \text{ verifies ConsumerIN?EG}$$

$$\textbf{Def. } \forall e_i \in E,\ PSM(e_i) \Leftrightarrow \qquad (3)$$
$$\exists \sigma\ y\ \exists e_k \mid e_i, e_k \in \sigma, e_k \le e_i \text{ in } \sigma \text{ and } e_k \text{ verifies MerchantIN?PS}$$

These three predicates are implemented in PROMELA by three variables named PSConsumer, EGConsumer and PSMerchant that are initially set to FALSE and are assigned TRUE the moment PS or EG are received by "Consumer" or "Merchant" respectively.

Thus, from what we have said, the following formulae should be verified globally for goods atomicity to be proved:

$$\forall \sigma \text{ state sequence by model } M \qquad (4)$$
$$(\exists e_i \in \sigma \mid PSC(e_i)\ \&\&\ EGC(e_i)) \Leftrightarrow (\exists e_k \in \sigma / PSM(e_k))$$

which is the same as the following two liveness properties expressed in LTL logic [6], where the notation is taken from SPIN documentation ([] the temporal operator always and <> the temporal operator eventually):

$$[\,]\,(\,PSC\ \&\&\ EGC \Rightarrow \langle\rangle\ PSM\,) \quad \&\& \quad [\,]\,(\,PSM \Rightarrow \langle\rangle\,(\,PSC\ \&\&\ EGC\,)\,) \tag{5}$$

SPIN properties are expressed in a negative way, so we will say that it should be impossible in any state:

$$(\,PSC\ \&\&\ EGC \Rightarrow [\,]\,(\,!PSM\,)\,) \quad || \quad (\,PSM \Rightarrow [\,]\,(\,!\,(\,PSC\ \&\&\ EGC\,)\,)\,) \tag{6}$$

So we translate the formulae to SPIN (see [5]) as a NEVER clause, a situation that can never hold in the set of reachable states.

After implementing the model and the property in the checker, it can be assured that NetBill protocol is goods atomic: no errors detected in a full state space search; with the following results (using 1.0 seconds in a SUN Ultra 1 with 128 MB RAM):

```
(Spin Version 2.9.4 -- 4 November 1996)
        + Partial Order Reduction

Full statespace search for:
        never-claim             +
        assertion violations    + (if within scope of claim)
        acceptance   cycles     - (not selected)
        invalid endstates       - (disabled by never-claim)

State-vector 348 byte, depth reached 648, errors: 0
  173676 states, stored
  343678 states, matched
  517354 transitions (= stored+matched)
   27170 atomic steps
hash conflicts: 441885 (resolved)
(max size 2^18 states)

6.60095e+07     memory usage (bytes)
```

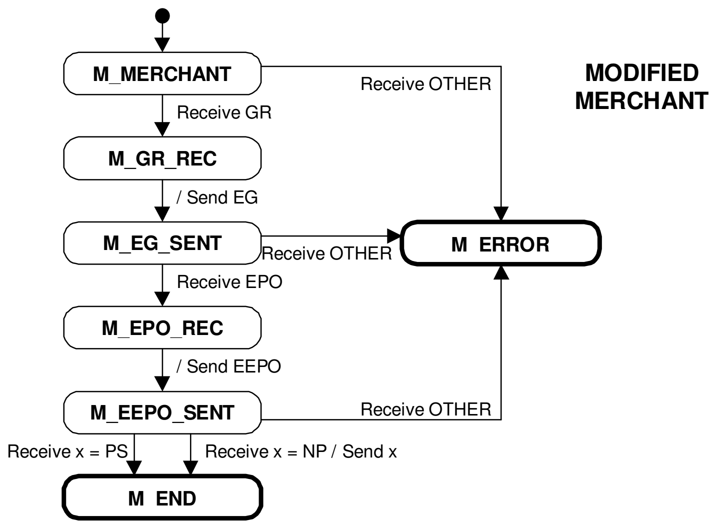
7 Simulating a Protocol Error

With the model exposed above, it has been proved that NetBill is goods atomic. But we have made several strong assumptions to isolate characteristics inherent to the protocol. In this section we will try to make our analysis more realistic taking out some of them

The honesty of the parties was one of the suppositions described in section 4. What would happen if we modify their behaviour? NetBill clearly states that the merchant must forward the response from the bank to the consumer (messages PS or NP). But as the consumer is able to obtain that information later through message TE, merchants may claim that forwarding the transaction result is a loss of time and money. Or a merchant may tray to fraud an unadvertised consumer by this action.

So the merchant process will be modified to not forward PS messages to the consumer when it is in “M_EEPO_SENT” state (see **Figure 5**).

Figure 5. State transition diagram describing the modified behaviour of the merchant



Of course, after implementing this change in PROMELA, SPIN showed a result similar to the one obtained before, because the consumer queried the bank to find out the transaction result.

We have to consider, too, the differences between the protocol description and its implementation. What if the consumer software had a bug? Due to current technology we assume many times that some things just do not happen, so the last two steps of NetBill, those regarding bank querying when a timeout occurs, may be seen by solution implementers as unlikely. Relaying on the good behaviour of the protocol parties and the network, this section of the software might be poorly proved. Thus we will model and simulate that the consumer, under certain circumstances, might not send TE. The model resulting from this buggy implementation will be called “NetBillModified Protocol”. The specification of this new protocol is the one shown in **Figures 6, 7** and **8**. Note that this bug, a timeout that never occurs, induces a lot of changes in the protocol definition.

Figure 6. Event Flow Diagram for the NetBillModified protocol

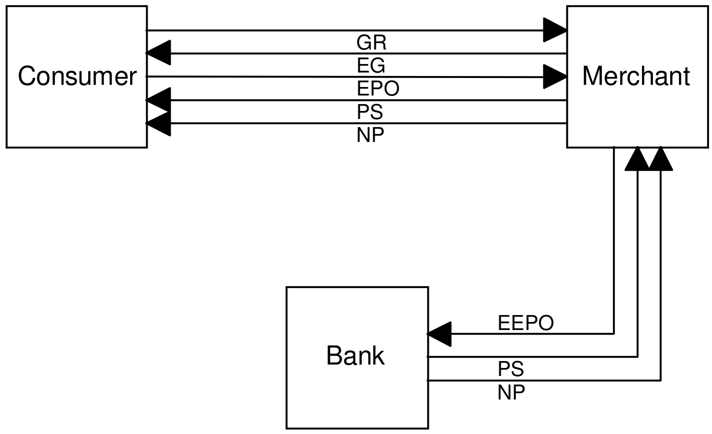


Figure 7. NetBillModified Steps (Event Trace Diagram)

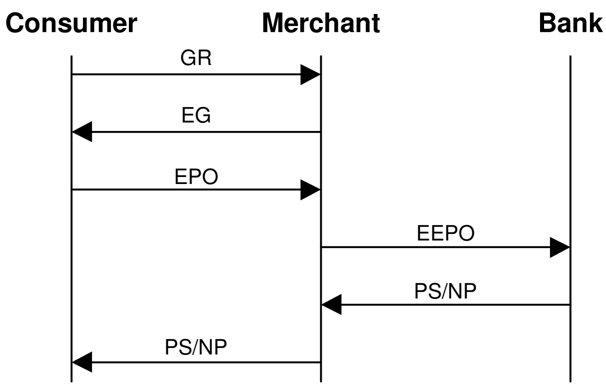
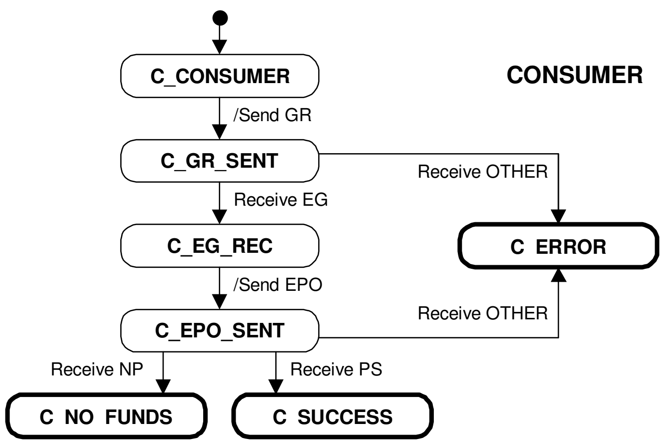
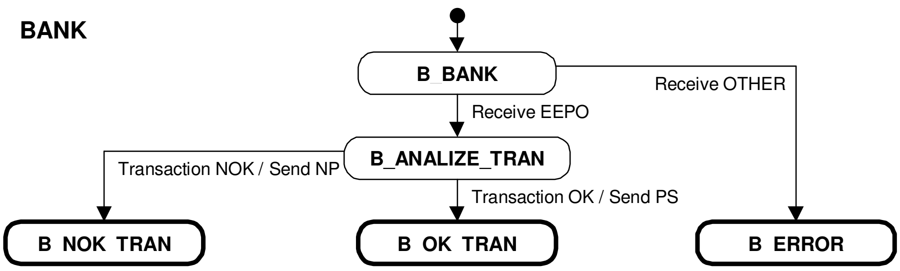


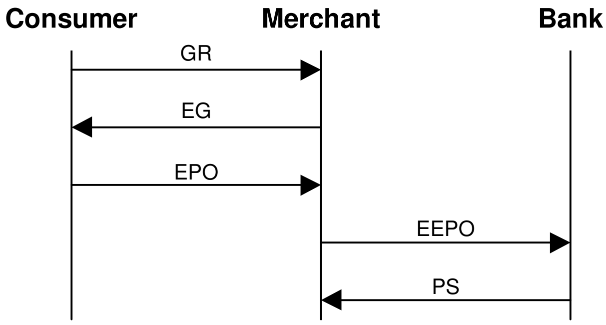
Figure 8. State transition diagrams describing the behaviour of the objects in the NetBillModified protocol (merchant does not change)





As a result of merging the modification in the merchant behaviour with the protocol vulnerability created by the bug in the implementation before described, NetBillModified becomes a non goods-atomic protocol, confirmed by SPIN with the counter-example shown in **Figure 9**. Merchant receives PS from the bank, so she is paid, but consumer receives the encrypted goods and not the key to decipher them: "the merchant receives payment **but** the consumer does not receive the goods".

Figure 9. Counter-example found by SPIN



8 Related Information

A similar work is referred in [4] by Heintze et al from Carnegie Mellon University, but they do not emphasize the use of graphical specification and deal exclusively with the formal verification issues. The way they model communications makes it difficult to include typical Internet attacks such as sniffing to the model (see section 5 for a description of a different implementation and further discussion). Besides, the tool used and the assumptions vary considerably.

In [1] Audun describes a model to verify security protocols that is under the implementation choice of a network process against individual channels for each direction of communication between parties. Audun deals with a finer granularity, so his work will be valuable to refine our model in the future.

9 Conclusions and Future Work

It is notorious the lack of consensus in the scientific community devoted to security protocols and cryptography in terms of the notation to be used to describe ciphered messages, digital signs and similar aspects. It is the most usual to find a section in all specialized papers describing the notation that the author will use throughout the exposition. It is evident, too, the lack of use of graphical specification methods that could be considered standard for protocol description. Well-known software engineering graphical notations may be used in this field, as we have made in this paper.

Formal techniques have soundly proved their capacity in protocol analysis and design, but are still infraused by designers in these stages of protocol development. The trend shows that they are only used when the protocol has been defined and designed, to validate or discover errors. It does not make sense, in our opinion, invest time and money in informal analysis and designs that will be later invalidated or remade. This team's work is devoted to research the use of notations and formal methods in the development of electronic commerce protocols. We believe that using the tools at hand not just in the last steps, but in the first ones of analysis and design, would enhance the protocol quality, shortening the total development effort at the same time.

The granularity used in this work seems to be adequate to check high level properties of the protocols, discarding implementation details of the cryptoalgorithms and the presence of attackers. This affirmation, however, should be proved with techniques of refinement and compositional verification to hold the system at different levels of granularity.

10 References

[1] Audun, J. "Security Protocol Verification using SPIN". Proceedings of the First SPIN Workshop, INRS-Telecommunications, Montreal, Canada, 1995.

[2] Cano, A.; Blanco, E.; Garcia-Fanjul, J.; Goitia, A.; Tuya, J. and Corrales, J.A. "CITIES: a model for the development of Web-based sales services". Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics '98 - Orlando (Florida). Pages 387-393. July, 1998.

[3] Cox, B.; Tygar, J.D. and Sirbu, M. "NetBill Security and Transaction Protocol". Proceedings of the First USENIX Workshop in Electronic Commerce. Pages. 77-88. July, 1995.

[4] Heintze, N.; Tygar, J.D.; Wing, J. and Wong, H.C. "Model Checking Electronic Commerce Protocols". Proceedings of the 2nd Usenix Workshop on Electronic Commerce. Pages. 147 - 164. November, 1996.

[5] Holzmann, G.J. "Design and Validation of Computer Protocols". Prentice-Hall, 1991.

[6] Manna, Z. and Pnueli, A. "The Temporal Logic of Reactive and Concurrent Systems Specification". Springer-Verlag. 1991.

[7] Mocas, S. and Schubert, T. "Formal Analysis of IP Layer Security". Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols. September, 1997

[8] Pitkow, J.E. and Kehoe, C.M. "Emerging trends in the WWW user population". Communications of the ACM. Pages 106-108. Volume 39, number 6. June, 1996.

[9] Rumbaugh, J. et al. "Object Oriented Modelling and Design". Prentice-Hall. 1991.

[10] Schieferdecker, I. "Abruptly Terminated Connections in TCP - A Verification Example". Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design, Maribor, Slovenia. Edited by Z. Brezocnik and T. Kapus, editors. Pages 136-145. June, 1996.

[11] Tuya, J.; De Diego, J.R.; De la Riva, C. and Corrales, J.A. "Dynamic Analysis of SA/RT Models Using SPIN and Modular Verification". The Spin Verification System. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Volume 29, 1997.

[12] Tygar, J.D. "Atomicity in Electronic Commerce". Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing. Pages 8-26. May, 1996.