A Byzantine-Resilient Protocol using the TTCB*

Miguel Correia, Lau Cheuk Lung, Nuno Ferreira Neves, and Paulo Veríssimo

Faculdade de Ciências da Universidade de Lisboa
Bloco C5, Campo Grande, 1749-016 Lisboa, Portugal
{mpc,lau,nuno,pjv}@di.fc.ul.pt

1 Introduction

The Trusted Timely Computing Base (TTCB) is a distributed security kernel with novel features: it is a distributed component with its own secure network; and it is real-time, i.e., it is a synchronous subsystem capable of timely behavior. These two characteristics together are uncommon in security kernels. The TTCB provides only a limited set of services. It has been implemented and shown that this implementation has a high coverage of its assumptions [3].

The TTCB is being designed on the context of the EU project MAFTIA [4]. The objective is to support the implementation of Byzantine-resilient (intrusion-tolerant) protocols. This abstract shows a conceptually simple Byzantine-resilient protocol based on the TTCB. An optimized but more complex version of the protocol can be found elsewhere [2].

2 TTCB and System Model

A TTCB is composed by local parts in hosts, *local TTCBs*, interconnected by a *control network*. A *local TTCB* is a computational component with activity, conceptually separated from the host operating system. Hosts are connected by a network, that we denominate *payload network* to differentiate from the TTCB control network. The TTCB is assumed to fail only by crashing, while the rest of the system, hosts and payload network, is assumed to fail arbitrarily.

The TTCB provides a set of *security-related services*. Processes use the *local authentication service* to authenticate the TTCB and to establish a *secure communication channel* (or trusted path) with it. The *trusted block agreement service* is the main building block for the construction of Byzantine-resilient protocols. The service delivers a value obtained from the agreement of values proposed by a set of processes. The values are small blocks and the TTCB has limited resources, so the objective is not to execute all agreement related operations inside the TTCB but only crucial steps of the protocols. Processes use the function *TTCB_propose* to propose a value to the service and *TTCB_ decide* to get the result.

The TTCB provides also a set of *time-related services*. These services can be used to support the implementation of partially-synchronous protocols [3].

* This work was partially supported by the EC, through project IST-1999-11583 (MAFTIA), and by the FCT, through the Large-Scale Informatic Systems Laboratory (LASIGE) and the project POSI/1999/CHS/33996 (DEFEATS).

3 The BRM Protocol

Reliable multicast is a classical distributed systems problem, informally described by two properties [1]: 1) all correct processes deliver the same messages, and 2) if a correct sender transmits a message then all correct processes deliver this message. The Byzantine-Reliable Multicast protocol is shown in Figure 1. The basic idea is that the sender uses a TTCB agreement to give the recipients a *hash* of the message, used to verify the integrity and authenticity of what is received. Messages are multicast $Od + 1$ times. Od is the omission degree, i.e., the maximum number of messages that can be lost in an interval of time due to accidental faults. It is not possible to define a similar value for malicious faults.

BRM-T Sender protocol

```
tstart = TTCB_getTimestamp() + T0;
M := (elist, tstart, data);
propose := TTCB_propose(elist, tstart, TTCB_TBA_RMULTICAST, H(M));
repeat Od+1 times do multicast M to elist except sender;
deliver M;
```

BRM-T Recipient protocol

```
read_blocking(M);
propose := TTCB_propose(M.elist, M.tstart, TTCB_TBA_RMULTICAST, ⊥);
do decide := TTCB_decide(propose.tag);
    while (decide.error ≠ TTCB_TBA_ENDED);
while (H(M) ≠ decide.value) do read_blocking(M);
repeat Od+1 times do multicast M to elist except sender;
deliver M;
```

Fig. 1. BRM-T protocol.

4 Conclusion

BRM is the first protocol designed with a *hybrid failure model* in which most of the system can be attacked and fail arbitrarily, while the TTCB can only fail by crashing. This protocol tolerates any number of failed processes, on the contrary to the proved maximum of f out of $3f + 1$ in purely asynchronous systems [1]. A variant of the protocol was implemented and shown to be efficient [2].

References

1. G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4):824–840, October 1985.
2. M. Correia, L. C. Lung, N. F. Neves, and P. Veríssimo. Efficient byzantine-resilient reliable multicast on a hybrid failure model. In *Proceedings of the 21th IEEE Symposium on Reliable Distributed Systems*, October 2002.
3. M. Correia, P. Veríssimo, and N. F. Neves. The design of a COTS real-time distributed security kernel. In *Proceedings of the Fourth European Dependable Computing Conference*, October 2002.
4. D. Powell and R. J. Stroud, editors. *MAFTIA: Conceptual Model and Architecture. Project MAFTIA IST-1999-11583 deliverable D2.* November 2001.