# Improving Security by Fault Tolerance

Yves Deswarte

LAAS-CNRS & INRIA
7, Avenue du Colonel Roche
31077 Toulouse Cedex, France
(deswarte@laas.fr)

March 6, 2001

Security techniques must accompany the current technological evolution: systems are more distributed and mobile, more complex, and more flexible to fit user requirements. Distribution and mobility imply that hardware faults are more frequent, more administrators and operators are involved, and more intrusions are likely to occur. Complexity and flexibility increase the probability of design faults. Secure systems have to deal with all these classes of faults: hardware faults, design faults, intrusions, malicious or careless administrators or operators. Fault tolerance should then help to make systems more secure.

Fault tolerance can be defined as a property of a computing system to perform its tasks correctly even if it is affected by faults. These faults can be either internal faults, i.e., failures of some system components, or external faults such as environment anomalies or erroneous interactions with users, operators or maintenance staff.

According to such definitions, some common practice techniques for security can be viewed as contributing to fault tolerance. For instance, protection mechanisms prevent unauthorized access, but also contribute to the detection and confinement of errors.

Moreover, since the correctness of some components is mandatory if security is to be enforced, it would be desirable to make these components fault tolerant. In particular, a *Trusted Computing Base* is a "single point of

failure", both for security and for reliability: if it fails, the TCB would either deny authorized access or enable unauthorized access. To prevent TCB failures, fault tolerance can be envisaged to deal with design faults, hardware faults, and even faulty interactions with trusted users.

However, application of fault tolerance techniques to security has to confront a fundamental contradiction: fault tolerance requires redundancy, and redundancy is detrimental to confidentiality. For example, data replication, while commonly used to tolerate accidental faults, implies that several information copies are present in the system. An intrusion into a part of the system is thus more likely to lead to sensitive information disclosure. Thus the challenge is to add redundancy while preserving confidentiality.

A first class of solutions consists of using coding theory to design error-correcting codes which would also be good ciphers. Threshold schemes [Sha79] are an example of such codes: the information is coded as a set of shadows such that gathering any $T$ shadows ($T$ being the threshold) is sufficient to reconstitute the information, while no information is obtained from less than $T$ shadows. This technique is efficient for the storage of small sensitive data items, but is inadequate for storing large files and for processing numerical data.

For the storage of large files, information dispersal techniques appear to be more efficient. In this case, files are split and stored on several storage units, and it is necessary to gather the information from several storage units to reconstitute the files. Two variants have been proposed. In one of these, Rabin suggests first to cipher the file, then to code it with a specific error-correcting code and to disperse the resulting data on different storage sites [Rab89]. LAAS proposed another variant called Fragmentation-Redundancy-Scattering (FRS) [FDP86]: the file is first ciphered by means of a stream cipher, the resulting data are distributed into fragments, and fragments are replicated and scattered among a set of storage sites. Confidentiality is increades due to the difficulty for an intruder to know in which order the different fragments have to be reassembled, rather than on the strength of the cipher.

For data processing itself, ciphering techniques are inadequate: most numerical operations and comparisons have to be run on deciphered data. LAAS and the University of Newcastle are currently developing an FRS technique for data processing [FDR94]. The application software is developed according to an object model and objects are decomposed into smaller objects recursively until each object either carries no significant information

(and then can be considered as a fragment), or cannot be decomposed any more (e.g., a character string). Undecomposable objects containing significant information have to be ciphered. Then redundancy can be added to fragments (e.g., by replication), and the resulting fragments can be scattered.

Fault tolerance techniques can also be applied to deal with the problem of malicious or careless operators and administrators: separation of duty can be viewed as a form of fault tolerance. Majority voting has also been successfully applied at LAAS to tolerate intrusions into security servers, including intrusions by privileged users [DBF91].

Even design faults can be tolerated by means of design diversity [JA88]. In this case, fault tolerance can be an attractive alternative to formal verification, when the complexity of the system or of its environment is close to or exceeds the current limitations of formal methods.

All these techniques show that fault tolerance can improve security.

# References


[DBF91] Yves Deswarte, Laurent Blain, and Jean-Charles Fabre. Intrusion tolerance in distributed systems. In *Proceedings of the 1991 International Symposium on Research in Security and Privacy*, pages 194–201. IEEE, May 1991.

[FDP86] Jean-Michel Fray, Yves Deswarte, and David Powell. Intrusion-tolerance using fine-grain fragmentation-scattering. In *Proceedings of the 1986 International Symposium on Security and Privacy*, pages 210–221. IEEE, April 1986.

[FDR94] Jean-Charles Fabre, Yves Deswarte, and Brian Randell. Designing secure and reliable applications using fragmentation-redundancy-scattering: an object oriented approach. In *Proceedings of the 1st European Dependable Computing Conference (EDCC-1)*. Springler-Verlag Lecture Notes in Computer Science, October 1994.

[JA88] Mark K. Joseph and Algirdas Avizienis. A fault-tolerance approach to computer viruses. In *Proceedings of the 1988 International Symposium on Security and Privacy*, pages 52–58. IEEE, May 1988.

[Rab89] Michael O. Rabin. Efficient dispersion of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335–348, April 1989.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.