

Using Redundancy to Increase Survivability*

Matti A. Hiltunen and Richard D. Schlichting
AT&T Shannon Laboratory
180 Park Avenue
Florham Park, NJ 07932

Carlos A. Ugarte
Department of Computer Science
The University of Arizona
Tucson, AZ 85721

1 Introduction

Secure communication services—that is, communication services that provide attributes such as confidentiality, integrity, and authenticity—typically implement each attribute using a single method for each connection. For example, confidentiality may be provided by DES and integrity by keyed MD5. Although such an approach may be secure in the traditional sense, it is not survivable—once a method is compromised, all security guarantees on the connection related to that attribute are gone. Each method is, in essence, a single point of vulnerability very much analogous to a single point of system failure when considering fault-tolerance attributes. This problem is the same for many other aspects of security, including authentication and access control.

This position paper advocates the use of a standard fault-tolerance technique—redundancy—to increase the survivability of communication. For example, using this approach, message integrity can be implemented by calculating redundant independent signatures, while confidentiality can be implemented by encrypting the message with a combination of methods with keys established using different methods. As a result, even if an intruder manages to find one key or break one algorithm, the security guarantees may remain intact. The task of the intruder can be complicated further by using secret combinations of methods or by dynamically altering the set of methods during the lifetime of the connection. By using multiple methods and doing so in ways that can vary unpredictably, the space of possibilities that must be considered by an attacker and the effort expended to compromise the attribute expands combinatorially. The approach also allows the tradeoff between the cost of the survivability and the protection to be managed explicitly and dynamically in response to changing threat scenarios.

This paper focuses on two key requirements for using redundancy to improve survivability, the development of appropriate techniques and the availability of suitable system support. We begin by discussing some specific redundancy techniques for both communication security and other security services, and then turn to the issue of system support. As an example of a system that has the necessary characteristics, we give an overview of Cactus, a system for building modular and configurable protocols and services, and SecComm, a highly configurable secure communication service implemented using Cactus. Other important aspects of the problem such as quantifying levels of survivability remain as future work.

2 Techniques

Implementing privacy and other attributes. The basic idea behind using redundancy to improve communication survivability is simple—with multiple methods enforcing a given attribute, the attribute should remain valid if at least one of the methods remains uncompromised. As with fault tolerance, however, the

*This work supported in part by the Defense Advanced Research Projects Agency under grant N66001-97-C-8518 and the National Science Foundation under grants CDA-9500991 and ANI-9979438.

effectiveness of the approach depends on the details of how it is used. As an example of this approach, we focus on using redundant cryptographic methods to ensure communication privacy.

In this context, a number of specific techniques can be devised based on redundant encryption. The simplest, of course, is to apply the different methods successively on the same data. For example, a message might first be encrypted using DES then IDEA. However, there are many other approaches, including:

- Alternating the order in which methods are applied, e.g., apply DES before IDEA for some messages and IDEA before DES for others.
- Applying different methods to different parts of the data, e.g., encrypt different parts of one message or different messages in a stream using different methods.

An important factor influencing the effectiveness of these approaches is the *independence* of the methods used, where two methods A and B are independent if compromising A provides no information that makes it easier to compromise B, and vice versa. A simple example of non-independence is when two encryption methods use the same key, since if one method is broken or the key stolen, privacy is completely compromised. To maximize independence in this type of situation, the keys should be established using different key distribution methods, for example, one using Diffie-Hellman and the other using Kerberos. As a result, the system may even be able to tolerate an attack where one or more of the key distribution centers are compromised.

Note that this type of independence is very much analogous to the fault-tolerance concept of independent failure modes for redundant hardware or software components. Components are independent in this sense when the failure of one component does not affect the correct execution of any other component.

While the independence of encryption methods is difficult to argue rigorously, the risk of methods not being independent is likely to be minimized if the methods are substantially different or if they encrypt data in different size blocks. It is also possible to develop combinations that attempt to maximize independence by not simply encrypting the same data multiple times, but by finding other ways to combine different encryption methods. For example, suppose that m is a cleartext message and E_1 and E_2 are different encryption methods. A ciphertext message cm could be constructed as $cm = \{E_1(m \otimes r), E_2(r)\}$, where \otimes is the exclusive-or operation and r is a random bit sequence the same length as the message. Given this method, breaking only E_1 or E_2 does not produce any useful information, which means that the attacker has to break both simultaneously to know if the system has been compromised. As a result, the effort required is multiplicative.

Determining independence of methods is easier for other security attributes such as message integrity. Let m be the message to be protected and $d_1(m), d_2(m), \dots$ be different cryptographic message digests of m . Since the message digest algorithms operate on the message independently, an attacker would need to compromise each integrity algorithm separately. In this case, the increase in the breaking effort is additive since the attacker knows when each method has been broken.

Finally, it is also possible to exploit analogues of fault-tolerance techniques that operate on a sequence of messages rather than on individual messages. For example, using techniques similar to forward error correction (FEC), message modification or modifications of the message stream (i.e., insertions and deletions) could be detected. Such techniques can naturally be used together with message-based methods to increase survivability further.

Other security services. These ideas can be applied to other types of security services in distributed systems as well. For example, redundancy can be used to increase the survivability of certification agencies and the PKI. If multiple certificates are required from multiple independent certification agencies or a user's public key is verified with a number of public key servers, the chance that an intruder can cause extensive damage by compromising one agency is reduced.

In areas of security where the existing operating system already provides mechanisms—e.g., authentication and file access control—overall system survivability can sometimes be improved by introducing redundant independent methods to enforce the desired security guarantees or to detect violations. For example, access control can be augmented with encryption, in which case a user can only read a file if allowed by the access control system and with the necessary key. Similarly, an intrusion detection system can be viewed as a redundant component that monitors user behavior to detect illegal activities that are acceptable to the standard OS security mechanisms.

Redundancy mechanisms can also be developed to address specific security problems. For example, the Cactus framework discussed in the next section has been used to develop a distributed system monitoring tool that was extended with a module designed to deal with an intruder modifying the web pages of an organization. This module monitors a subtree of the directory structure by comparing a checksum of each file against the previous checksum of the same file. If a file is modified, the system administrator is notified. This technique allows unauthorized web page modifications to be detected quickly using redundant checks.

3 System support

The realization of services that use redundancy-based survivability techniques such as those described above can be simplified using an appropriate software customization framework. Here, we discuss one such system called Cactus, together with SecComm, a specific instance of a highly-customizable secure communication service implemented using Cactus.

Cactus. Cactus is a system for constructing configurable network protocols and services where each service property or functional component is implemented as a separate software module called a *micro-protocol* [3]. A customized instance of a service is then created by choosing a collection of micro-protocols based on the properties to be enforced, and configuring them together with the Cactus runtime system to form a *composite protocol* that implements the service. A micro-protocol is structured as a collection of *event handlers* that are executed when a specified event occurs. Events can be raised explicitly by micro-protocols or by the runtime. The runtime also provides a variety of operations for managing events and handlers, such as binding and unbinding event handlers and events. In addition, Cactus supports shared data that can be accessed by all micro-protocols configured into a composite protocol.

The flexibility of the Cactus model allows abstract service properties and functions to be implemented as independent modules without enforcing artificial ordering between modules as in hierarchical composition frameworks such as the *x-kernel* [4]. Furthermore, the indirection provided by the event mechanism makes it easy to change the collection of micro-protocols dynamically without affecting other micro-protocols.

Several prototype implementations of Cactus have been constructed, including one in C that runs on Linux and another in Java that runs on multiple platforms. Cactus and the predecessor Coyote system [1] have been used to construct a number of other communication-oriented network services, including group RPC, group membership, and communication channels that provide combinations of real-time and reliability guarantees.

SecComm service. SecComm is a secure communication service that executes in user space with either IP, UDP, or TCP as the underlying protocol. SecComm allows fine-grain customization of a range of security attributes including privacy, authenticity, message integrity, replay prevention, non-repudiation, and key distribution [2]. A secure connection created through SecComm is customizable in the sense that only the required security attributes are guaranteed. Furthermore, the strength of guarantee associated with each attribute can be customized at a fine grain level. The service offers the choice of the cryptographic techniques

and key lengths used to implement each attribute, and each attribute can be guaranteed using complex combinations of security algorithms making it easy to utilize redundancy techniques.

The set of micro-protocols designed and implemented for SecComm consists of *basic security* micro-protocols that perform security operations such as DES encryption or MD5 message digest and *meta security* micro-protocols that construct more complex security algorithms out of the basic micro-protocols. Meta security micro-protocols are especially useful for implementing redundancy-based survivability techniques, since each such micro-protocol encodes one method of combining different security mechanisms. For example, they can apply multiple or alternating security algorithms on a message, or can break a message into multiple parts and then apply different techniques to each part. The event-oriented execution paradigm is a significant advantage here, since a meta security micro-protocol only encapsulates the control structure of *how* mechanisms are combined, with no explicit references to which micro-protocols are being combined. Rather, it simply raises events that it receives as arguments at session creation time that are bound to event handlers in the micro-protocols implementing the specific mechanisms to be combined.

4 Conclusions and future work

This paper has discussed the use of redundant techniques as the basis for improving the survivability of security services. While the idea of combining fault tolerance and security is not new, this position paper promotes a more general application of redundancy techniques in different areas of security and introduces a convenient implementation platform for such techniques. Our approach can also be viewed as a way of artificially increasing the diversity of the system, which has been advocated elsewhere as a potential approach to improving survivability [5].

Future work will focus on using the Cactus framework to implement dynamically adaptable security services, where the security mechanisms are changed at runtime in reaction to changed security requirements (e.g., suspected intrusion) or changes in available resources. The Cactus framework makes it easy to activate and deactivate micro-protocols at runtime, and we have designed and implemented coordination mechanisms for distributed adaptation that allow adaptations to occur smoothly without interrupting normal operation. We intend to apply these techniques to explore survivability in the context of the SecComm service.

References

- [1] N. Bhatti, M. Hiltunen, R. Schlichting, and W. Chiu. Coyote: A system for constructing fine-grain configurable communication services. *ACM Transactions on Computer Systems*, 16(4):321–366, Nov 1998.
- [2] M. Hiltunen, S. Jaiprakash, R. Schlichting, and C. Ugarte. Fine-grain configurability for secure communication. Technical Report 00-05, Department of Computer Science, University of Arizona, Tucson, AZ, Jun 2000.
- [3] M. Hiltunen, R. Schlichting, X. Han, M. Cardozo, and R. Das. Real-time dependable channels: Customizing QoS attributes for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):600–612, Jun 1999.
- [4] N. Hutchinson and L. Peterson. The *x*-kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, Jan 1991.
- [5] F. Schneider, editor. *Trust in Cyberspace*. Committee on Information Systems Trustworthiness, National Research Council, National Academy Press, Washington, D.C, Sep 1998.