Chapter 6
Self-Protecting Key Management

“Adapt or perish is nature’s inexorable imperative”
– H.G. Wells

Abstract The idea of self-protecting key management is to extend standard cryptographic key management schemes with functions that allow them to adapt automatically to changes in security policies. The motivation for this approach is that adaptability can help significantly reduce the risk of security exposure, and remove the complexity that manual security management implies. This chapter presents a method of extending a cryptographic key management scheme to ensure self-protection in the case of key updates. Our method is based on using backup keys and data versions to anticipate key update requests instead of generating the keys on demand which is time-consuming and can create delays that result in security attacks.

6.1 Overview

In the previous three chapters we presented ways in which CKM schemes can be extended and/or implemented for better performance. Efficient CKM schemes are good for handling security, because they allow an access control scheme to match security with performance and so provide better guarantees of satisfying service level agreements. However, emerging applications based on paradigms like data outsourcing, service oriented architectures, and cloud computing, make it necessary to take adaptability into account in designing security schemes.

This need for adaptability in security schemes has arisen mainly because of the complexity of managing security in Web-applications, where security management requires dealing with changing conditions that are not easy to predict and hard code into the applications. For instance, in the CKM schemes, it is difficult to manually predict where and when a rekey request will occur. Standard CKM is difficult to apply to applications where the security conditions change frequently because man-
ually key updates result in delayed responses during which users wait for the new key.

This chapter presents a framework based on the Autonomic Computing paradigm [35, 88, 87] that allows a CKM scheme to adapt to changing scenarios by monitoring the rate at which key update requests occur and using this information to make predictions about the arrival rate of future rekey requests. The advantage of this solution is that it minimizes the response time and size of the vulnerability window created by delays that arise when rekey requests are handled manually in a CKM scheme.

Our framework is composed of six functionalities: the Sensor, Monitor, Analyzer, Planner, Executor, and Effector, that are connected together in the form of a feedback control loop (FBCL) [84, 85]. The FBCL continually monitors the arrival rate of rekey requests at the key server and, at regular intervals, computes an acceptable resource (keys and encrypted replicas/data versions) allocation plan to minimize the overall cost of rekeying. Each component of the framework contributes to enhancing a standard CKM scheme’s performance without changing its underlying characteristics. A prototype implementation and experiments showing performance improvements over a standard CKM scheme demonstrate the effectiveness of the proposed framework.

### 6.2 Self-Protecting Cryptographic Key Management (SPCKM) Framework

Our Self-Protecting Cryptographic Key Management (SPCKM) framework handles the rekey problem by supporting a standard KM scheme with an autonomic model. Each class in the access control hierarchy represents a group of users authorized to access a portion of the shared data. A rekey request is when a message is transmitted to the key server indicating a change in group membership. The message can be explicit in the sense that a user actively indicates his/her wish to leave the group, or it can be implicit, for instance if a user remains inactive for a long time then the KM system can lock him/her out for safety reasons.

In the framework, the adaptive rekey process is initiated by monitoring, over an initial period, the rate at which the key server (central authority) receives requests. This information is then used to generate keys and encrypted versions of the data to anticipate future rekey requests. In comparison to quasi-static key management approaches, the SPCKM framework allows a key management scheme to minimize the response time and the size of the vulnerability window created in handling rekey requests.

As in the schemes we have presented in previous chapters, we assume that there exists a single trusted central authority (key server) $U_0$ in charge of key generation/assignment and data encryption. Cases of central server crashes are assumed to be handled by some fault-tolerance solution like server replication. The key server encrypts the data on the data server according to the rules of access specified by the