Chapter 4
Timestamped Key Management

“Have a vision. Be demanding.”
—Colin Powell

Abstract Our Timestamp key management scheme, like the Atallah et al. scheme aims to replace only the key associated with the group affected by a change in security policy. In this scheme, we use the idea that each key is associated to a unique value that is computed by a certified and trusted authority. Key replacements are handled by updating this value instead of key thereby significantly reducing the cost of updates.

4.1 On Timestamps and Key Updates

This chapter describes a second approach to efficient rekeying. As mentioned before, one of the main issues that arises in rekeying is that of minimizing the size of the window of vulnerability created during the rekey process. In the previous chapter, we noted that the requirement of rekeying and re-encrypting the associated data to preserve data security creates a wide vulnerability window [12, 83]. Since this process is time consuming and causes delay, the approach we propose in this chapter associates a timestamp to each key and uses both pieces of information to compute a verification signature that can be used for authenticating users [81]. Replacement is handled by updating both the timestamp and the verification signature as opposed to updating the whole hierarchy and re-encrypting all the associated data, as is the case in the SPKM scheme that we proposed previously.

As illustrated in Figure 4.1, the assumption is that there exists a single trusted secure central authority $U_0$, as well as a trusted (secure) timestamp authority (TSA), in charge of key generation and timestamp creation respectively. The timestamp authority communicates the timestamps to $U_0$, where they are associated with the keys generated and transmitted securely to the user groups in the hierarchy. Data is
only accessible to users if his/her key and timestamp yield a verification signature (hash value) that corresponds to the one associated with the data.

The central authority $U_0$, ensures secure access by transmitting the key and timestamp pair for each group to the data server where a hash value (verification signature) is computed for each key and timestamp pair, and stored in a secret registry $R_H$. It is assumed that the registry is kept secret by encrypting its contents with a secret key that is held by the data server. Each slot in the registry points to the data encrypted under the group key, associated with the signature. For instance, in Figure 4.1, the verification signature $H_{K_0,T_0}$, grants access to $D_{K_0},...,$ $D_{K_{n-1}}$ because all the other keys $K_1,...,K_{n-1}$ are derivable from the key $K_0$ associated with the verification signature $H_{K_0,T_0}$.

In order to access data, members of a security class (group) $U_j$, transmit their key $K_j$ and timestamp $T_j$, securely to the data server where a pre-defined one-way hash function, $H$, is applied to the pair to compute a verification signature $H_{K_j,T_j}$. The computed signature $H_{K_j,T_j}$ is then compared to the value in $R_H$ for $U_j$ and if the computed signature matches the currently valid signature for $U_j$ then access is granted, otherwise it is denied. In the case where access is granted, the user in $U_j$ can then access all data packages that obey the precedence relation, $U_i \preceq U_j$.

When the membership of a group changes, $U_0$ contacts the TSA for a new timestamp to replace the old value of $T_j$. On reception of the new timestamp, $U_0$ transmits this securely to the members of $U_j$ and recomputes a new hash value $H_{K_j,T_j}$ which is