CHAPTER 6

Conclusions

Fine-grained access control to outsourced data using a symmetric key-based cryptosystem requires each data file (that can be accessed individually) is encrypted with a distinct key. A user authorize for accessing a set of files needs to store each file’s secret key. Key management hierarchies are used to efficiently manage a large set of secret keys with each user and enforce data access control. Two types of key management hierarchy are used for managing keys in data outsourcing scenario: user-based and resource-based. We critically analyzed the two types of key management hierarchy and show that the storage requirement for resource hierarchies will be same as user-based hierarchies. They perform better in the case of dynamic operations such as extending read authorization and revoking a user without affecting other required functionalities. However, average key derivation-time in resource-based hierarchies is more than user-based hierarchies. We have implemented the two hierarchies and shown the results experimentally for the sake of our arguments. We conclude that the resource-based hierarchies can be a good candidate for key management in data outsourcing scenario.

The goal of a subscription-based key management hierarchy is to enforce time-limited (or subscription-based) access control. The key assignment for time-limited access is done using a subscription-based HKAS (SBHKAS). Existing SBHKASs exhibit a trade-off between private storage requirement by a user, system public storage requirement, and key derivation cost. Reducing public storage is not generally emphasized in designing traditional HKAS but it is relevant in data outsourcing where consumer is paying for storage-as-a-service. We have proposed a simple and efficient hash-based SBHKAS using dependent keys. The proposed
SBHKAS reduces the secret storage cost at the central authority responsible for managing the keys and system public storage without increasing other costs such as secret storage per user and key derivation cost. However, the average re-keying operation cost due to access right revocation is more in HKAS with dependent keys. In case of dependent keys, ascendant nodes keys are also sometime needs to be re-keyed. In our scheme, a parent node’s key needs to be re-keyed only if the key of child node is computed using the parent key.

Access right revocation is a desired operation in many access control systems. Traditionally, it is handled using the re-keying mechanism that assigns and encrypts each affected outsourced node with a new key. It prevents the revoked user with old secrets from accessing the resource which is now encrypted with the new key. Wang et al. [9] proposed an access right revocation mechanism for outsourced data considering honest but curious CSP. The important feature of their mechanism is that it does not require re-keying procedure (used in traditional systems). However, in their mechanism, a user’s access right revocation is dependent on all other users’ access. Therefore, the system does not scale. In our proposed system, access right revocation can be efficiently handled using the improved certificate-based data access mechanism where each revocation is independently handled.

Recently, Vimercati et al. [40] proposed a SBHKAS for outsourced data in a cloud scenario. We show that their scheme has a security flaw. In their scheme, a user after withdrawing his subscription can still have access to the resources associated with his old and revoked subscription interval.

Write access control for outsourced data is more challenging as compared to read access control since it requires more on the service provider. A small malicious change to the outsourced content can put a hugely adverse effect on the data owner’s business. Therefore, in the literature, a malicious but cautious CSP type is adopted for write access control. The data owner requires an auditing like mechanism to detect any misbehavior and will take appropriate action accordingly to avoid it in future. In existing schemes, it is possible that a user can modify their own written outsourced records any number of times in collusion with a service
provider, without being detected by the data owner. We first time consider this property as an important security requirement and propose an audit-based mechanism to handle it. We also provide a stronger freshness guarantee for distributed cloud scenario to assure a reader that the received data file is fresh at least until the time when it was dispatched by the service provider. We argue that for freshness property, storing timestamp and version number with the outsourced data record is not sufficient. It must require some proof mechanism that can be used later at the time of disputes. Finally, although the audit-based mechanisms will defend against many write access control issues, we realize there is still an open question that can we prevent unauthorized writes without using audit-based mechanisms?

Personal health record (PHR) is a well accepted patient-centric model for cloud-based e-health. It is one of the important privacy enabled data outsourcing application. We proposed a symmetric key based PHR management system (PHRMS) for the cloud using hash chains for key management. Two of the important requirements we addressed are forward secrecy and unobservability. As best of our knowledge, we first time addressed the forward secrecy requirement in PHRMS. It will be beneficial for patient’s outsourced data privacy. Privacy of PHR is a primary concern for the user it belongs. It becomes more challenging when it is outsourced to a untrusted cloud. Unobservability is a privacy property defends against traffic analysis and is important requirement when communication is through untrusted entity such as cloud service provider. We achieve unobservability using mix node and show that the forward secrecy can be achieved using one-way hash chains. Although the presence of mix node adds communication delay, it will be significantly less than the document generation time.