Punit Mehta

Threats to Archival systems

Introduction

- Data is preserved for longer time than traditional storage
  - (controlled access to its long-term contents)
- secure archival storage systems must store data over much longer periods of time, new threats emerge that affect the security landscape.
- Adversaries: Active (attempts to break in and compromise an archive). Passive (restricted to eavesdropping on communication channels)

Security Threats : Long term secrecy

- Long lived encryption: for an encrypted file in an archival storage system, the data is persistent and the key must be preserved for an indefinite period of time.
- Short lived encryption: a communication channel that is secured through the use of encryption and a key that is relevant only for the duration of the session.
- Archive (long-lived) : Issues,
  - 1: store keys for longer duration (often single point of failure).
  - 2: Decides to change the key/ new algorithm??
  - 3: Tech. such as quantum computing can break things!?
  - 4: key loss ~ data deletion
  - 5: re-encrypt the contents of long-term file storage ? (sometimes many petabytes of data might need re-encryption)

ST: **Locating Data**

- not able to locate in archival is an issue?
- Retrievable in a timely manner
- Central Index : Bank lockers
  - Compromise of central index is more scary
  - Option: personal index strategy (onus of maintaining an index upon the client)
    - + : an attacker that compromises one user's index knows very little about other users' data.
    - - : client then has more long-term responsibility.

ST: **Authentication and User Accounts**

- Users must show who they are before the system can determine what they are allowed to do.
- Challenge (Uniquely to archival system)
  - Suppose a user wants to securely store their will in an archival storage system
  - Subsequent read may take place decades later by the owner's next of kin after the owner had passed away.
  - A secure, archival storage system must thus be able to authenticate new users and establish their relationship to resources attached to existing users.
  - CATCH: user with rights to the data and no access to the encryption key -> effectively NO access.
  - If a deceased user's next of kin proves his legal right to the data, the secrecy mechanism must function in the complete absence of the user that wrote the file.

ST: **Integrity Guarantees**

- "Short-term" systems: Won't be a big issue. Updates happen frequently.
- Archival system?
  - Integrity of data actively checked at regular intervals
  - Solution: *disk scrubbing (parity + checksum check)*
    - A scrubbing procedure periodically scans sections of the data and uses strong hashes to detect corruption, recovering data that is damaged.
    - overactive scrubbing : contribute to media failures.
    - under-active scrubbing : no utility over the absence of disk scrubbing.
- Remember LONG term!
  - cryptographic hashes may not be sufficient. Adversary to find collisions in the hash used for disk scrubbing, can update the original data with incorrect data, thus fooling the integrity checking procedure.

ST: **Slow Attacks**

- several decades of time to conduct an attack
- Issues
  - 1. Intrusion detection: Signature matching algorithms (i.e. compare audit data and network activity to a database of known attacks' signatures) wont work if the attack is methodical enough to make only the slightest of changes at any one time and each step was spaced far enough apart
  - 2. Maintaining attack history: critical for slow attack. Security logging is important. Maintenance of a history of compromises.
    - Example: File is split into 5 blocks and we need at least 3 of them to completely understand the content. If attacker gets one and later after decade gets another, system should trigger the potential threat/breach!

ST: **Migration and Recovery**

- Long term storage – data needs to be moved across systems/infras.
- Archival (long-term) systems : harware failures are inevitable.
  - must thus be immune to the failure of any given component.
  - recovery and migration of effected data to new hardware.
    - SECURITY here?
      - – migration/recovery should happen without allowing any party to actually recover data
- Moving data: create a moving target that could help to limit an adversary’s ability to launch a targeted attack. (Recall: slow-attack)

Real life archival systems

FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment

- Farsite is a secure, scalable file system that logically functions as a centralized file server but is physically distributed among a set of untrusted computers.
- Farsite provides file availability and reliability through randomized replicated storage; it ensures the secrecy of file contents with cryptographic techniques; it maintains the integrity of file and directory data with a (~~Byzantine-fault-tolerant)~~ protocol.

Basic setup

- Every machine in Farsite may perform three roles: It is a client, a member of a directory group, and a file host
- A client is a machine that directly interacts with a user. A directory group is a set of machines that collectively manage file information
- Every member of the group stores a replica of the information, and as the group receives client requests, each member processes these requests deterministically, updates its replica, and sends replies to the client.
- Data consistency: (as long as fewer than a third of the machines misbehave) Byzantine protocol.

Security Part-1 : Access Control

- Enforce write and read access controls.
- “directory metadata” includes an access control list (ACL) of public keys of all users.
- Cryptographically authenticated channels (involves a user’s private key) --- originating from a specific legitimate user.
- The directory group validates the authorization of a user’s update request before accepting the update.

Security Part-2 : Privacy

- Both file content and user-sensitive metadata (meaning file and directory names) are encrypted for privacy.
- Client creates a new file!
  - it randomly generates a symmetric file key
  - Encrypts the file with that key.
  - Encrypts the file key using the public keys of all authorized readers of the file
  - Stores the file key encryptions with the file
  - Retrieval? -- A user with a corresponding private key can decrypt the file key and therewith the file.
- To prevent members of a directory group from viewing file or directory names, they are encrypted too!

Security Part-3 : Integrity

- As long as fewer than one third of the members of a directory group are faulty or malicious, the integrity of directory metadata is maintained by the (~~Byzantinefault-tolerant~~) protocol.
- Integrity:
  - Computing a ~~Merkle~~ hash tree over the file data blocks,
  - Storing a copy of the tree with the file
  - keeping a copy of the root hash in the directory group that manages the file's metadata.
- Because of the tree, the cost of an in-place file-block update is logarithmic – rather than linear – in the file size. The hash tree also enables a client to validate any file block in logarithmic time, without waiting to download the entire file.

Misc Important Points

- Modifying a file (unlike creating, renaming, or deleting it) affects not only the file metadata but also the file content. It is necessary to update the content atomically with the metadata; otherwise, they may be left in an inconsistent state following a crash, and the file content will be unverifiable.
- Byzantine-fault-tolerant protocol achieves good performance by locally caching file data, lazily propagating file updates, and varying the duration.

Freenet: A Distributed Anonymous Information Storage and Retrieval System

Goals

- Anonymity for both producers and consumers of information
- Deniability for adversaries
- Resistance to attempts by 3rd parties to deny access to information
- Efficient dynamic storage and routing of information
- Decentralization of all network functions

Basic Idea

- Each node maintains a dynamic routing table
- This routing table contains addresses for other nodes and keys which these nodes are thought to hold
- Requests for keys are passed from node to node through a chain of proxy requests in which each node makes a local decision about where to send the request next – Routes for requests will vary
  - Each request has a HTL (hops-to-live)
  - Each request has a pseudo-random ID to prevent loops
  - No node is privileged over any other node

Security : File Keys

- 3 different types of file keys
  - – KSK (keyword-signed key)
    - Derived from a short description the user provides – text/philosophy/sun-tzu/art-of-war
    - Asymmetric key pair
      - – Public half – hashed to yield the file key
      - – Private half – used to sign the file to prevent others from tampering with the file
  - – SSK (signed-subspace key)
    - Public/private key pair created by user
    - Public key and descriptive text (like the text used for the KSK) hashed separately and then XORed together, then hashed again
    - Private key used to sign the file (like with the KSK)
    - To allow others to retrieve the file, simply publish the public key and descriptive text
  - – CHK (content-hash key)
    - Directly hash the file to get a CHK
    - Designed for files that aren't going to change (such as audio/video files)
    - Files encrypted with randomly generated encryption key
    - To allow others access to the file, simply publish the file's hash and the decryption key

Retrieving Data

- User sends request to own node with key and HTL
- Node checks its own data store
  - – If found, return data along with note saying it was the source
  - – If not found, look up nearest key (lexicographically) in its routing table and forward request to that node
  - If request is ultimately satisfied, pass the data to requester, cache data in own data store, and add data to own routing table
  - Subsequent requests for same file will be served from own data store

OceanStore: An architecture for global-scale persistent storage

- Refs from - http://www.powershow.com/view1/4eb7e-ZDc1Z/OceanStore_An_Architecture_for_Global-scale_Persistent_Storage_powerpoint_ppt_presentation