CS628A: Assignment 1

Introduction

- Implement a secure Key-Value store
- Key-Value map = map[string][ ]byte
- KeyStore = map[string]rsa.PublicKey
- DataStore → Can delete/modify any entry in the map
- KeyStore → Trusted storage of public keys
- User → Store, share/revoke, delete, append

Intro(contd.)

- “A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.” - Kerckhoffs's principle
- Implementation should not maintain any global state other than the DataStore and KeyStore
- A user needs to remember only the username and password for performing the ops.

Bad Design #1

- File → stored as (filename, value) in DataStore
- Problems?
  - filename is public => other users know where the file is!
  - Other users can modify/delete the file!

Bad Design #2

- File →stored as (filename, E(value)) in DataStore
- Problems?
- filename is not confidential, other users know location of file, can delete it

Bad Design #3

- File → stored as (unguessable_loc, E(value))
- Problems?
- The DataStore can replace E(value) with E(value_new)
- Lacks integrity!

Bad Design #4

- File1 →stored as (unguessable_loc1, E(k, value1) w. integrity)
- File2 → stored as (unguessable_loc2, E(k, value2) w. integrity)
- Problems?
- DataStore can replace E(k, value1) w. integrity with E(k, value2) w. integrity

Inefficient Appends

- Filecontent = LoadFile(“foobar”)
- Filecontent += “appended content”
- StoreFile(“foobar”, Filecontent)
- Problems?
  - Whole file have to be transferred from DataStore server to Client

Other “Useful” Things

- Can you store User credentials in Datastore in order to verify User?
  - It’s ok if you can verify with very high probability.
- There is no differentiation between the data that is stored → Purely implementation specific
- Writing modular code will help you a ton!