Cryptographic Keys
CS 236
On-Line MS Program
Networks and Systems Security

Peter Reiher

Outline

- Properties of keys
- Key management
- Key servers
- Certificates

Introduction

- It doesn’t matter how strong your encryption algorithm is
- Or how secure your protocol is
- If the opponents can get hold of your keys, your security is gone
- Proper use of keys is crucial to security in computing systems

Properties of Keys

- Length
- Randomness
- Lifetime
- Secrecy

Key Length

- If your cryptographic algorithm is otherwise perfect, its strength depends on key length
- Since the only attack is a brute force attempt to discover the key
- The longer the key, the more brute force required

Are There Real Costs for Key Length?

- Generally, more bits is more secure
- Why not a whole lot of key bits, then?
- Some encryption done in hardware
  - More bits in hardware costs more
- Some software encryption slows down as you add more bits, too
  - Public key cryptography especially
- Some algorithms have defined key lengths only
- If the attack isn't brute force, key length might not help

Key Randomness

- Brute force attacks assume you chose your key at random
- If attacker learns how you chose your key
  - He can reduce brute force costs
- How good is your random number generator?

Generating Random Keys

- Well, don't use `rand()`[1]
- The closer the method chosen approaches true randomness, the better
- But, generally, don't want to rely on exotic hardware
- True randomness is not essential
  - Need same statistical properties
  - And non-reproducibility

[1]See http://eprint.iacr.org/2013/338.pdf for details

Cryptographic Methods

- Start with a random number
- Use a cryptographic hash on it
- If the cryptographic hash is a good one, the new number looks pretty random
- Produce new keys by hashing old ones
- Depends on strength of hash algorithm
- Falls apart if any key is ever broken
  – Doesn't have *perfect forward secrecy*

Perfect Forward Secrecy

- A highly desirable property in a cryptosystem
- It means that the compromise of any one session key will not compromise any other
  - E.g., don't derive one key from another using a repeatable algorithm
- Keys do get divulged, so minimize the resulting damage

Random Noise

- Observe an event that is likely to be random
  - Physical processes (cosmic rays, etc.)
  - Real world processes (variations in disk drive delay, keystroke delays, etc.)
- Assign bit values to possible outcomes
- Record or generate them as needed
- More formally described as *gathering entropy*
- Keys derived with proper use of randomness have good perfect forward secrecy

On Users and Randomness

- Some crypto packages require users to provide entropy
  - To bootstrap key generation or other uses of randomness
- Users do this badly (often very badly)
- They usually try to do something simple
  - And not really random
- Better to have crypto package get its own entropy

Don’t Go Crazy on Randomness

- Make sure it’s non-reproducible
  - So attackers can’t play it back
- Make sure there aren’t obvious patterns
- Attacking truly unknown patterns in fairly random numbers is extremely challenging
  - They’ll probably mug you, instead

Key Lifetime

- If a good key's so hard to find,
  - Why every change it?
- How long should one keep using a given key?

Why Change Keys?

- Long-lived keys more likely to be compromised
- The longer a key lives, the more data is exposed if it's compromised
- The longer a key lives, the more resources opponents can (and will) devote to breaking it
- The more a key is used, the easier the cryptanalysis on it
- **A secret that cannot be readily changed should be regarded as a vulnerability**

Practicalities of Key Lifetimes

- In some cases, changing keys is inconvenient
  - E.g., encryption of data files
- Keys used for specific communications sessions should be changed often
  - E.g., new key for each phone call
- Keys used for key distribution can't be changed too often

Destroying Old Keys

- Never keep a key around longer than necessary
  - Gives opponents more opportunities
- Destroy keys securely
  - For computers, remember that information may be in multiple places
    - Caches, virtual memory pages, freed file blocks, stack frames, etc.
  - Real modern attacks based on finding old keys in unlikely places

Key Storage

- The flip side of destroying keys –
  - You'd better be sure you don't lose a key while you still need it
- Without the key, you can't read the encrypted data
  - Kind of a bummer, if you wanted to
- Key storage is one approach

What Is Key Storage?

- Saving a copy of a cryptographic key "somewhere else"
- Securely store a key in some safe place
- If you lose it accidentally, get it back from storage location
- Prevents encrypted data from becoming unreadable

Where Should You Store Keys?

- Must not be accessible to an attacker
  - Don't want him to get hold of all your keys
  - Don't want them readily available if your machine is hacked
- But relatively accessible when needed
- Usually on a separate machine

How Do You Get Keys There?

- And back
- Each new key must be transported to the key server
- Not much saved if transport mechanism is compromised
- Need carefully designed/implemented mechanism for moving keys

Key Secrecy

- Seems obvious
- Of course you keep your keys secret
- However, not always handled well in the real world
- Particularly with public key cryptography

Some Problems With Key Sharing

- Private keys are often shared
  - Same private key used on multiple machines
  - For multiple users
  - Stored in "convenient" places
  - Perhaps backed up on tapes in plaintext form

Why Do People Do This?

- For convenience
- To share expensive certificates
- Because they aren't thinking clearly
- Because they don't know any better
- A recent example:
  - RuggedCom's Rugged Operating System for power plant control systems
  - Private key embedded in executable

To Make It Clear,

- **PRIVATE KEYS ARE PRIVATE!**
- They are for use by a single user
- They should <u>never</u> be shared or given away
- They must never be left lying around in insecure places
  - Widely distributed executables are insecure
  - Just because it's tedious to decipher executables doesn't mean can't be done
- The entire security of PK systems depends on the secrecy of the private key!