CS530
Key Management & Distribution Issues
(Part 2)

Bill Cheng

http://merlot.usc.edu/cs530-s10

Key Management & Distribution Issues

- Practical issues
  - how to carry them
    - passwords vs. disks vs. smartcards
  - where do they stay, where do they go
  - how many do you have
  - how do you get them to begin with
- Classes of crypto
  - which type is right for your application
- Who needs strong secrets?
- How do you recover from exposed keys?
- Miscellaneous issues
  - security architectures

Key Management Overview

- Key management is where much security weakness lies
  - what types of keys to use for a system and how to choose keys
    - want large *key entropy* (amount of randomness in keys) nobody uses rot13 -- (inverse is itself!)
    - example of weak protocal: WEP
    - really short keys: PIN
    - verifiable plaintext attacks
      Ex: Does this look like English?
      If plaintext contains a checksum, great! Let's automate the attack!
    - known plaintext attacks
      Ex: precomputed dictionary attack
      - need to *salt* the password (then can only use dictionary attack)

Key Management Overview (Cont...)

- where do you store the keys?
  - floppy disks, USB harddrives (can be encrypted)
  - smartcard
    - -- key never leaves card
    - -- not vulnerable to even keyboard sniffer
    - -- not popular in US, probably because high costs (cost of cards + cost of infrastructure)
    - -- variety of smartcards: tamper proof, tamper resistant, tamper evident (tamper evident is good enough for end users)
  - post-it note?

Key Management Overview (Cont...)

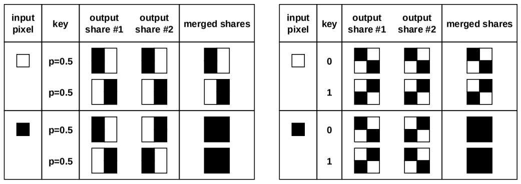
- how do you communicate about keys (key distribution)?
  - conventional: KDC
    - single key shared by both parties
    - generate and distribute keys
    - bind names to shared keys
  - public key: CA
    - public key published to the world
    - private key known only by owner
    - sign bindings of keys to names (protects integrity)
    - verifiable by multiple parties
  - third party certifies or distributes keys
    - certification infrastructure
    - authentication

Key Management Overview (Cont...)

- Classes of crypto
  - one-time pad (truely random)
    - most secure, not vulnerable to attacks
    - if pseudo-random number generator used, must have large IV
    - problem: key size must be as large as data size
    - limited applications
      Ex: submarines
  - visual cryptography (next page)
  - conventional: n^2 keys
  - public key: $2n$ keys
    - n is number of parties

Visual Cryptography

- Invented by Naor and Shamir (presented at EUROCRYPT '94)
  - see Doug Stinson's Visual Cryptography Page



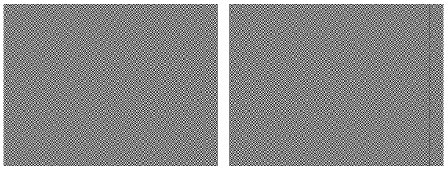
2x data expansion

4x data expansion

  - application: secret splitting
  - *perfect secrecy* (just like a one-time pad)

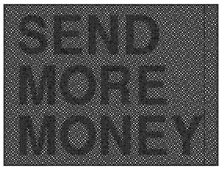
Visual Cryptography Example

- Ex:



Visual Cryptography Example (Cont...)

- Ex:



  - original image

SEND
MORE
MONEY

Grey or Color Image

- If a pixel value is not just black/white
  - grey images - real value between 0 (black) and 1 (white)
  - color images - RGB, real value between 0 and 1 in each component color
  - in both cases, can approximate with pure black and white values
- Two basic approaches
  - *thresholding* -- e.g., replace value by 1 if intensity ≥ 0.5 and replace value by 0 if intensity < 0.5
  - *error diffusion* -- start with thresholding, carry error into the next pixel

Error Diffusion

- Error diffusion
  - if a pixel value is 0.8, approximate it with 1, the difference (0.2) is the error
  - if the next pixel value is 0.3, the error in the previoius pixel is subtracted, the resulting pixel value is 0.1
  - 0.8 + 0.3 = 1 + 0.1
  - 0.1 is approximated by 0, the new error is 0.1
  - keep going

original

thresholding

error diffusion

Key Management Overview (Cont...)

- Who needs strong secrets anyway? (sometimes, secrets are not needed, what is really needed is *integrity of association*)
  - users?
    - need to prove identity
      - start with something not that confidential (SS#, mother's maiden name)
  - servers?
    - private key is usually sitting on the server!
      - not well protected
      - should probably put it on a smartcard
  - the Security System?
    - such as Kerberos/KDC, must have strong secrets

Key Management Overview (Cont...)

- Who needs strong secrets anyway? (cont...)
  - software?
    - DRM? (Digital Rights Management)
      - -- does it really work? (e.g., DVD player for Linux)
      - -- is it fair? (the entertainment industry wants everyone to pay for their weak copyright protection)
      - -- MS Palladium (Microsoft's secure computing base) place Microsoft as the gatekeeper of identification and authentication
  - end systems?
    - keys for hardware
- Secret vs. Public
  - public: integrity protected

Practical Use of Keys

- Email (PEM or S/MIME)
  - hashes and message keys to be distributed and signed
- Conferencing
  - group key management (discussed later)
- Authentication (discussed later)
- SSL (details later)
  - and other "real time" protocols
  - key establishment

Recovery from Exposed Keys

- Revocation lists (CRLs)
  - long lists
  - hard to propogate
- Lifetime / expiration
  - short life allows assurance of validity at time of issue
- Realtime validation
  - Online Certificate Status Protocol (OCSP)
    - privacy concerns? (server knows who you have been communicating with)
- What about existing messages?

Other Key Management Issues

- Key size vs. data size
  - affects security and usability
- Reuse of keys
  - multiple users, multiple messages
- Initial exchange
  - the bootstrap/registration problem
    - Ex: Web
      - use social security numbers?
      - use "personal" information?
        - -- 2002, Princeton admission official improperly logged into Yale website using "personal" info
  - confidentiality vs. authentication
    - sometimes you do not really need authentication

Other Key Management Issues (Cont...)

- sometimes you do not really need authentication (cont...)
  - -- client is often unauthenticated (server often does not know who the client is)
  - -- long term relationship more important
  - -- if the "real owner" hasn't complained and this client is paying the bills, this client is probably the "real owner"
- Security architectures
  - put some security requirements together

Security Architectures

- DSSA (Distributed Systems Security Architecture)
  - around 1987, originally from DEC
  - Ex: how to protect against booting from a CD and access all files on harddrive
    - hardware can checksum OS before loading the OS
      - -- if no match, don't load it
      - -- if match, create a certificate, pass it to the OS
  - delegation is the important issue
    - workstation can act as user
    - software can act as workstation
      - if given key
    - software can act as developer
      - if checksum validated
  - complete chain needed to assume authority
  - roles provide limits on authority - new sub-principal

Security Architectures (Cont...)

- Microsoft Authenticode
  - downloadable executables such as Java applets, Windows updates, ActiveX controls uses signed certificates
  - delegate trust to browser
- Proxies (also based on delegation)
  - limits on authority explicitly embedded in proxies
  - works well with ACL (access control list)
  - more on proxies in "authorization"