Lecture 4 - Cryptography
September 14, 2006
URL: http://www.cse.psu.edu/~tjaeger/cse543-f06/
Review: secret vs. public key crypto.

- Secret key cryptography
  - Symmetric keys, where a single key (k) is used for encryption (E) and decryption (D)
    - $D(E(p, k), k) = p$
- All (intended) receivers have access to key
- Note: Management of keys determines who has access to encrypted data
  - E.g., password encrypted email
- Also known as symmetric key cryptography

- Public key cryptography
  - Each key pair consists of a public and private component: $k^+$ (public key), $k^-$ (private key)
    - $D(E(p, k^+), k^-) = p$
    - $D(E(p, k^-), k^+) = p$
  - Public keys are distributed (typically) through public key certificates
  - Anyone can communicate secretly with you if they have your certificate
  - E.g., SSL-base web commerce
The symmetric/asymmetric key tradeoff

- **Symmetric (shared) key systems**
  - Efficient (Many MB/sec throughput)
  - Difficult key management
    - Kerberos
    - Key agreement protocols

- **Asymmetric (public) key systems**
  - Slow algorithms (so far …)
  - Easy (easier) key management
    - PKI - public key infrastructures
    - Webs of trust (PGP)
Hash Algorithms

• Hash algorithm
  – Compression of data into a hash value
  – E.g., \( h(d) = \text{parity}(d) \)
  – Such algorithms are generally useful in programs

• … as used in cryptosystems
  – One-way - (computationally) hard to invert \( h() \), i.e., compute \( h^{-1}(y) \), where \( y = h(d) \)
  – Collision resistant hard to find two data \( x_1 \) and \( x_2 \) such that \( h(x_1) = h(x_2) \)

• Q: What can you do with these constructs?
• HMAC
  – Authenticates integrity for data d
  – Uses some key k and hash algorithm h
  – To simplify,
    \[ \text{hmac}(k,d) = h(k+d) \]

• Why does this provide authenticity?
  – Cannot produce \( \text{hmac}(k,d) \) unless you know \( k \), \( d \)
  – If you could, then can break \( h \)
  – Exercise for class: prove the previous statement

• Used in protocols to authenticate content
Birthday Attack

• A birthday attack is a name used to refer to a class of brute-force attacks.
  – birthday paradox: the probability that two or more people in a group of 23 share the same birthday is greater than 50%

• General formulation
  – function f() whose output is uniformly distributed
  – On repeated random inputs \( n = \{ n_1, n_2, \ldots, n_k \} \)
    • \( \Pr(n_i = n_j) = 1.2k^{1/2} \), for some \( 1 \leq i, j \leq k \), \( 1 \leq j < k \), \( i \neq j \)
    • E.g., \( 1.2(365^{1/2}) \approx 23 \)

• Q: Why is resilience to birthday attacks important?
Digital Signatures

• Models physical signatures in digital world
  – Association between private key and document
  – … and indirectly identity and document.
  – Asserts that document is authentic and non-reputable

• To sign a document
  – Given document d, private key k-
  – Signature $S(d) = E(k^-, h(d))$

• Validation
  – Given document d, signature $S(d)$, public key $k^+$
  – Validate $D(k^+, S(d)) = H(d)$
Basic truths of cryptography ...

- Cryptography is not frequently the source of security problems
  - Algorithms are well known and widely studied
    - Use of crypto commonly is … (e.g., WEP)
  - Vetted through crypto community
  - Avoid any “proprietary” encryption
  - Claims of “new technology” or “perfect security” are almost assuredly snake oil
Important principles

• Don’t design your own crypto algorithm
  – Use standards whenever possible
• Make sure you understand parameter choices
• Make sure you understand algorithm interactions
  – E.g. the order of encryption and authentication
    • Turns out that authenticate then encrypt is risky
• Be open with your design
  – Solicit feedback
  – Use open algorithms and protocols
  – Open code? (jury is still out)
Building systems with cryptography

• Use quality libraries
  – SSL/Leay, lim (from Lenstra), Victor Shoup’s library, RSAREF, cryptolib
  – Find out what cryptographers think of a package before using it

• Code review like crazy

• Educate yourself on how to use library
  – Caveats by original designer and programmer
Common issues that lead to pitfalls

• Generating randomness
• Storage of secret keys
• Virtual memory (pages secrets onto disk)
• Protocol interactions
• Poor user interface
• Poor choice of key length, prime length, using parameters from one algorithm in another
A really good book on the topic