Today

Introduction

Asymmetric Encryption

Public Key Infrastructure (PKI)

Cryptanalysis

An Example: WS-Security

---

**Applied Cryptography**

- Use of cryptography to build secure systems
- Symmetric encryption gives us tools
  - hash functions and message digests
  - encryption and decryption ciphers
- … but suffer two fundamental problems
  - the key distribution problem
  - the key management problem

---

**Fundamental Concepts**

- **Confidentiality**
  - limitation of access to information to authorized parties
  - symmetric encryption
- **Integrity**
  - prevention of unauthorized alteration of information
  - message authentication codes
- **Authentication**
  - identity and information origin verification
  - challenge-response
- **Non-repudiation**
  - inability to deny previous commitments or actions
  - ?

---

**Asymmetric Encryption**

- Commonly referred to as public key encryption
- Messages encrypted using key pairs (public & private)
- One key used for encryption, the other for decryption
- Public key distributed as much as possible
- Private key kept secret
- Versatile and more secure than symmetric algorithms
- Comparatively slow
Today
Introduction
Asymmetric
Encryption
Public Key Infrastructure (PKI)
Cryptanalysis
An Example:

- Cryptanalysis
- Asymmetric Encryption
- Public Key Infrastructure (PKI)

Public Key Infrastructures (PKI)

- Virtual infrastructures consisting of clients, servers and Certification Authorities (CAs)
- CAs are trusted third parties which provide signed certificates (i.e., signs public keys)
- CA certificates are distributed in browsers and similar tools (trusted and considered known by all)
- Since CA public keys are known, (signed) certificates can be validated offline (without connecting to the CA)
- Secure connections are established between parties using certificates and encryption algorithms
- Network traffic tunneled through encrypted channels

Asymmetric Encryption

- Based on mathematical one-way functions
  - (relatively) easy to compute with known parameters
  - computationally infeasible to recover parameters
- Three main classes of problems
  - integer (prime) factorization
  - discrete logarithm
  - elliptic curve
- Comparatively slow, often combined with other tools
  - exchange of session keys in transport protocols
  - sign hashes of messages rather than messages

Asymmetric Encryption Trust

- As long as the keys can be trusted
  - messages can be kept secret (only receiver can decrypt)
  - senders and receivers can be authenticated
  - message content can be trusted
  - (acknowledged) messages cannot be revoked
- Can be used to address the key distribution problem (does not solve the key management problem)

Certificates

- Certificate = signed tuple of public key & identity
- Certificates can be self-signed or signed by others
- Certificates signed by trusted parties can be used for encryption, authentication and message integrity checks
- Self-signed certificates can only be used for encryption (they suffer from the key distribution problem)

Certification Authorities

- Trusted third party organizations
- Provides certificate issuing services
- CA public keys assumed to be well known (distributed with browsers and operating systems)
- Commercial operators (2014)
  - Symantec (owns VeriSign, Thawte, Geotrust)
  - Comodo SSL
  - Go Daddy
  - GlobalSign
**Introduction**

As it is used, with the average user not bothering to verify the certificates and no revocation mechanism, SSL is simply a (very slow) Diffie-Hellman key-exchange method. Digital certificates provide no actual security for electronic commerce: it’s a complete sham. (Bruce Schneir, Secrets and Lies)

---

**Cryptanalysis**

- **Ciphertext-only attack**
  - recover plaintexts (or key) using ciphertexts encrypted with the same key
- **Known-plaintext attack**
  - recover key using ciphertexts and corresponding plaintexts
- **Chosen-plaintext attack**
  - recover key using ciphertexts from selected plaintexts
- **Adaptive-chosen-plaintext attack**
  - chosen-plaintext attack where plaintexts are updated during analysis

**Brute-Force Attacks**

- **Exhaustive key search** ("try every combination")
- **Can be used to attack (almost) any encrypted data** (exception: information-theoretically secure ciphers)
- **Feasibility primarily determined by two factors:**
  - key length and algorithm complexity
    - key length determines search space size
    - algorithm complexity determines cost of test
- **Cost of test grows linearly with key length**
  - 512 bit key $2^{512}$ times as complex as 256 bit key
- **Search space grows exponentially with key length**
  - 512 bit key $2^{512}$ times as complex as 256 bit key

---

**Secure Socket Layer (SSL)**

- A protocol for establishing secure connections using certificates and cryptography algorithms
- Transport Level Security (TLS) ≈ SSL v3.0
- Clients use server certificate to authenticate server
- Servers use client certificate to authenticate client (optional)
- Once identities are established, symmetric session keys are exchanged
- Keystores to manage certificates and keys

---

**Application of SSL**

"As it is used, with the average user not bothering to verify the certificates and no revocation mechanism, SSL is simply a (very slow) Diffie-Hellman key-exchange method. Digital certificates provide no actual security for electronic commerce: it’s a complete sham." (Bruce Schneir, Secrets and Lies)
Brute-Force Attacks
- exploits reduce search complexity
  - reduced key spaces
- heuristic functions reduce search space
  - statistical analysis (frequency analysis, birthday attacks)
  - known data formats (e.g., protocol headers)
  - try common values first (e.g., dictionary attacks)
- tools and resources widely available today
  - attack software mature and widely available
  - specialized databases (dictionaries)
  - application-specific integrated circuits (ASICS)
  - field-programmable gate arrays (FPGAs)
  - graphics processor units (GPU) farms

Hash Collisions
- checksums and hashes are designed to minimize the likelihood of collisions
  - approaching rectangular distribution (uniform probability)
- pigeonhole principle
  - $n$ items stored in $m$ containers: $n > m$ gives at least one container contains > 1 item
  - not possible to uniquely hash any large data set to a smaller representation
- collisions are (unlikely but) unavoidable

Birthday Paradox
- the probability that randomly selected independent values in a finite set have the same value grows much faster than the probability of selecting a specific value from the set
  - 30 people -> 7.9% chance for a specific birthday
  - 30 people -> 70% chance for the same birthday
  - the likelihood of a hash collision is around 50% when the number of data items hashed approaches the square root of the number of unique data representations

Birthday Attack
- force hash collisions using (the mathematics of) the birthday paradox
  - evaluate value permutations that produce a certain hash
    - non-rectangular value distributions reduce search space
    - value space restrictions reduce search space
- spoofing signatures
  - $A$ sends a signed message to $B$ (hash used in signature)
  - $C$ intercepts the message in transit
  - $C$ finds a permutation of a different message with the same hash
  - $C$ sends the altered message to $B$ (pretending to be $A$)
- naive brute force attack requires (at most) $2^n/2$ tests
- hash attacks require (at most) $2^{n/2}$ tests

Approximate Hash Strength
- approximate hash strength calculations
  - assume rectangular distribution and probabilities $\leq 0.5$
  - $m$: possible hash values
  - $n$: non-colliding data values
  - $P(n)$: probability of a value
  - $P(n) \approx \frac{n}{2^m} \frac{m}{2^m}$
  - $n \approx \sqrt{2^m \cdot P(n)}$
- given a 32-bit hash, what is the maximum number of values we can store if we want to keep the chance of a hash collision $\leq$ one in a million ($P \approx 2^{-30}$)
  - $n \approx \sqrt{2^m \cdot \frac{1}{2^{30}}} \approx 90$
- SHA-512 (512-bit hash, $2^{512}$ possible values), $P \approx 2^{-20}$
  - $n \approx \sqrt{2^{512} \cdot \frac{1}{2^{20}}} \approx 2^{246}$
  - approximate #atoms in our galaxy: $2^{266}$
**Hash Attack Example: Passwords**

- 2013: Ars Technica gave 3 experts a file with 16k passwords (MD5 hash)
  - 90% recovered in 20 hours
  - commodity system with AMD Radeon 7970 graphics card
- Attack surface
  - broken hash (MD5)
    - fast hashes (MD5): ≈ 8B tests/s 20 hours
    - slow hashes (SHA512): 2k tests/s 2.5 years
  - policy limitations
    - reduce search space based on patterns (8 chars, 3 digits)
    - dictionaries
    - reduce search space based on commonalities


---

**Password Strength**

- Password attacks are heuristics brute-force based
  - dictionary and combinator attacks
  - targeting recovered password files or online systems
- Complexity = | password |
  - 8 lowercase letters: $26^8 \approx 2^{36}$
  - 8 upper/lowercase letters or numbers: $62^8 \approx 2^{38}$
  - 4 words from dictionary of 1000: $1000^4 \approx 2^{36}$
  - 4 words from dictionary of 100000: $100000^4 \approx 2^{67}$
- Probability
  - ...assuming rectangular value distributions
  - ...difficulty to guess vs difficulty to use (remember)

---

**Information-Theoretic Security**

- Not enough information to decipher messages without key
- Perfect security (special case)
  - an encryption algorithm is perfectly secure if a ciphertext produced using it provides no information about the plaintext without knowledge of the key
- One-time pad (1882) (bit / char level)
  - stream cipher with perfectly random (secret) keystream
  - lifetime limited by the amount of tokens on the pad
- Even with perfect ciphers, key problems remain
  - key distribution
  - key management

---

**Key Management**

- Keystores: encrypted databases for keys and certificates
- Usually stored in a single file called .keystore
- Applications must provide database decryption key
  - (username & password) to access keystore content
- Keystores only containing public keys and certificates are commonly referred to as truststores
- Keystores can be shared between SSL applications
  - (usually only done for truststores)
WS-Security

- A standard for secure SOAP point-to-point messaging
- XML Security protects XML
  - XML-Encryption
  - XML-Signature
- WS-Security protects Web Services by securing SOAP
  - links XML Security with tokens
    (K.509, Kerberos, SAML)
  - provides SOAP bindings and defines SOAP security headers
- Extensible
- Customizable

Transport-Level Security

- Provides a secure channel for SOAP message transport via HTTP over TLS (HTTPS)
- all communication is encrypted
- Hybrid encryption scheme (good performance)
- Point-to-point security
  - protection over a single network connection
  - does not work across intermediaries

Message-Level Security

- Sign and/or encrypt portions of the SOAP message
  - asymmetric encryption (poor performance)
  - end-to-end security (security across intermediaries)
- WS-SecureConversation
  - "SSL for application layer" (hybrid scheme)
  - apply security on a set of SOAP message exchanges
  - establish an initial security context (session key) that is used to protect subsequent messages

Apache Rampart

- WS-Security implementation for Axis2
- Supported specs (Rampart 1.4)
  - WS-Security
  - WS-SecureConversation
  - WS-Trust (partial support)
- No need to modify service code
  - security callbacks in message handler chains
  - configured in services.xml (service-side) and in client-side configuration file
  - advanced stub generators can generate security code
- Installation guide and examples on course web page

WS-Security Message Structure

Apache Rampart Message Processing
Today
Introduction
Asymmetric Encryption
Public Key Infrastructure (PKI)
Cryptanalysis
An Example: WS-Security

Advanced Distributed Systems
P-O Östberg
Applied Cryptography
Department of Computing Science, Umeå University