CPSC/PMAT 669

Cryptography in Practice and Current Trends

Mike Jacobson

Department of Computer Science
University of Calgary

Topic 11

Outline

Two Real Life Examples

We will now see two applications that put much of what we've learned together:

- **PGP**

  "**P**retty **G**ood **P**rivacy" — a secure e-mail system developed by Phil Zimmerman (`www.philzimmermann.com`)

- **SSH**

  Secure **Sh**ell – a PKC based access control system for remote login and file transfer

Features of PGP

Website: `www.pgp.com`

- Originally available for free world-wide, since 2010 owned and sold by Symantec
- Runs on most platforms
- Uses best available cryptographic primitives
- Not developed by government nor standards organization
- Compatible with most e-mail programs
- Automatically segments large messages (to accommodate message size limitations)
- Users may have multiple public keys, each identified by its 64 low order bits (*key ID*, denoted ID_K)

Cryptographic Services Provided by PGP

Authentication, data integrity, non-repudiation, using digital signatures

- DSS/SHA-1 (1024 bit keys), RSA/SHA-1 (768 to 3072 bit keys) and others are supported

Confidentiality, using a variation of CFB encryption (see documentation for details)

- A wide variety of block ciphers including AES are supported
- El Gamal and RSA for key transport (hybrid encryption)

Sending Secure E-Mail Using PGP

A sends an authenticated, encrypted message M to B as follows:

1. Computes the signature $S = D_A(H(M))$ on the SHA-1 hash of M
2. Compresses (S, M) using ZIP
3. Generates a random one time key K_{OT} to be used to encrypt *only* this message M
4. Uses K_{OT} to encrypt a time stamp TS, the key ID $ID_K(E_A)$ of her public key, the signature S, and the message M:

$$C = E_{K_{OT}}(TS, ID_K(E_A), S, M)$$

5. Encrypts K_{OT} using B's public key and sends the corresponding key ID $ID_K(E_B)$, encryption of K and C to B:

$$(ID_K(E_B), E_B(K_{OT}), C)$$

Receiving Secure E-Mail Using PGP

Upon receipt of $(ID_K(E_B), E_B(K_{OT}), C)$, B decrypts and verifies:

1. Decrypts the one time key $K_{OT} = D_B(E_B(K_{OT}))$
2. Recovers the signature S and message M by computing $D_{K_{OT}}(C) = (TS, ID_K(E_A), S, M)$
3. Checks that the time stamp TS is within an acceptable limit
4. Verifies the authenticity of M by comparing $H(M)$ with $E_A(S)$

Some features:

- The key IDs allow A and B to use the correct public keys when they have multiple public/private key pairs
- The use of time stamps prevents replay attacks
- No session keys are needed as each symmetric key is used to encrypt only one message

Types of Keys in PGP

One-time encryption keys

- Generated by PGP via PRNG
- Used for encryption of the current message only

Public/private keys

- User's public/private key pairs, other users' known public keys
- Generated by PGP via PRNG
- Used for signature generation and verification

Pass phrase

- Generated by user
- Used for encrypting and storing user's private keys

Private Key Rings

A user's public/private key pairs are stored in a *private key ring* maintained and stored by the user.

Each entry corresponds to one public/private key pair and contains:

- time-stamp (time of key creation)
- key ID
- public key (generated by PGP)
- private key (generated by PGP)
- user ID — different ID's, typically email addresses, may be assigned to different public/private keys

Stored in encrypted form using a block cipher. Access key is the SHA-1 hash value of a user generated secret pass phrase. To retrieve a private key, the user gets prompted to enter the pass phrase.

Public Key Rings

Other users' public keys known to a user are stored in a *public key ring*.

Can obtain keys from:

- secure public channels, CAs, etc...,
- *plus* from a mutually trusted individual.

Novel aspect of PGP: authenticity of public keys is **decentralized**

- PGP provides a mechanism for *quantifying* trust.
- no central trusted authority!

Main Problem

Public Key Ring Entries

Each entry corresponds to one known public key and contains:

- user ID (email address), time-stamp, key ID, public key.
- *owner trust field* — is this public key trusted to sign other certificates? User-assigned — higher value indicates higher degree of trust.
- *key legitimacy field* — higher value indicates higher trust in the binding of public key to user ID. Computed by PGP as a function of signature trust fields.
- *digital signatures* — zero or more signatures, each vouching for the authenticity of the key to ID binding of this public key.
- *signature trust fields* — each indicates the degree of trust in one signature. Higher value indicates a higher degree of trust in the signature's author. The key legitimacy field is a function of the signature trust fields.

Obtaining Public Keys, I

A inserts a new public key (certificate with attached signatures) into his public key ring as follows:

1. The owner trust field is assigned (byte value).
   - If the public key belongs to A, a value of "ultimate trust" is assigned.
   - Otherwise, user selects one of "unknown," "untrusted," "marginal trust," or "complete trust".

2. Signature trust fields are assigned (byte values). For each attached signature:
   - If the signature's author is unknown, the signature trust field is assigned "unknown user".
   - Otherwise, the signature trust field is assigned the corresponding owner trust field.

Obtaining Public Keys, II

3. The key legitimacy field is evaluated based on the signature trust fields:
   - If at least one signature trust field is "ultimate trust," the key legitimacy is assigned "complete".
   - Otherwise, key legitimacy is derived from a weighted sum, where weights are assigned to trust values as follows:

     weight of $1/X$ to "always trusted" signature trust fields,
     weight of $1/Y$ to "usually trusted" signature trust fields,

     where X and Y user-configurable parameters.
   - If total weight is ≥ 1, the key legitimacy is assigned "complete". So in the absence of "ultimate trust", one needs

     at least X "always trusted" signatures or
     at least Y "usually trusted" signatures or
     some suitable combination thereof.

     If total weight is < 1, the key legitimacy is assigned "not trusted" or "marginally trusted".

Consistency of Public Key Rings and Key Revocation

This scheme makes it possible to trust the authenticity of a user's public key, but not to trust the user to sign other users' keys.

Key revocation:
- PGP occasionally checks public key rings for consistency.
- A user can revoke his public keys by issuing a *key revocation certificate* — a signed certificate with a revocation flag set — and sending it to as many users as possible.
- The recipients must then update their public key rings accordingly.

PGP: Usability?

Not in wide-spread use (poor usability).

SSH (Secure Shell)

Website: `www.tectia.com` (`www.ssh.com` redirects to there)

SSH is an access control system consisting of 3 components:

1. **SSH Transport Layer Protocol**
   - unilateral authentication (server to client) — client downloads server's public key
   - establishment of shared session key for secure communication
2. **SSH User Authentication Protocol**
   - unilateral authentication (client to server) protected by shared session key
3. **SSH Connection Protocol**
   - interactive applications protected by shared session key

Once the secure channel is set up in step 1, the other two are relatively straightforward.

SSH TLP — TCP Connection Establishment

1. $C \to S : V_C$
2. $S \to C : V_S$
3. $C \to S : I_C$
4. $S \to C : I_S$

Steps 1 & 2: identification

- V_C, V_S: client's and server's SSH protocol and software versions

Steps 3 & 4: algorithm negotiation

- I_C, I_S: lists of algorithms supported for key agreement, encryption, MAC, compression

For each category, the algorithm chosen is the first one listed in I_C that is also listed in I_S.

SSH TLP — Key Agreement

Unilaterally authenticated Diffie-Hellman, server S to client C:

Protocol (all "mod p"s omitted):

1. $C \to S : g^x$
2. $S \to C : K_S,\ g^y,\ sig_S\big(H(V_C, V_S, I_C, I_S, K_S, g^x, g^y, K)\big)$ where
   - K_S — server's public key
   - $K = g^{xy}$ — session key
   - V_C, V_S — SSH protocol & Software versions
   - I_C, I_S — algorithm lists
3. C verifies authenticity of K_S and validates the server's signature

Note that K_S is not authenticated (beware of man-in-the-middle attacks!)

Management and Validation of Server's Public Keys

Two approaches:

1. Use public-key certificates
   - Problem: PKI not widely deployed
2. Current solution: each client maintains a local database (e.g. `$HOME/.ssh/known_hosts`) containing associations between servers and public keys.

   Suggested methods to ensure authenticity of stored public keys:
   - carry authenticated copy on removable storage media (e.g. a USB key or token)
   - obtain public key over an insecure channel, verify over phone (read out hash of obtained public key)

   Not perfect, but a huge improvement over `rlogin`, `rsh`, `rftp`, `telnet` etc (which have no security!)

SSH TLP Secure Channel

Once authenticated Diffie-Hellman is completed, server and client have a shared session key and hence a secure channel.

Services of the secure channel:

- Confidentiality
  - 3DES-CBC required
  - AES128-CBC recommended
- Data Integrity
  - HMAC-SHA1 required
  - HMAC-SHA1-96 recommended

SSH User Authentication and Connection Protocols

SSH User Authentication Protocol:

- Unilateral authentication (client to server) over the secure channel established by SSH TLP
- Authentication is based on the user proving possession of some cryptographic credential:
  - Public key challenge/response required (private key derived from user's pass phrase)
  - Password based authentication should also be supported

SSH Connection Protocol:

- standard interactive shell applications over the mutually authenticated secure channel established by the previous two components

Quantum Cryptography

Where is cryptography heading?

Quantum cryptography (QC) designs and analyzes cryptographic schemes whose security resides in the laws of quantum mechanics (e.g. Heisenberg's uncertainty principle), rather than some computationally difficult math problem.

The most famous QC scheme is the *BB84* quantum key distribution scheme (Bennett & Brassard 1984) for establishing a secret key (consisting of bits) by exchanging and measuring quantum bits sent across a public quantum channel.

- Information theoretically secure
- Distance record: 404 km over fiber optic cable (Hua-Lei Yin, U Science Technology China, June 2016)

Advantages and Disadvantages of QC

Main Advantage of QC:

- Theoretically impervious to attacks — must defeat laws of nature!

Problems with QC:

- Signal degradation
- Difficult to implement
- Implementations susceptible to physical attacks
- Unknown how to do authentication via quantum systems

Quantum Computing

The aim of *quantum computing* is to explore and build *quantum computers*, computing devices that make use of quantum mechanical phenomena for their operation (as opposed to transistor-based conventional computers).

There are different quantum architectures which are good at solving different problems and have been realized to different degrees:

- *Adiabatic* quantum computers are good at solving certain NP-hard problems such as satisfiability and combinatorial searches. They exist (e.g. D-Wave Systems' 2X system).
- Computers using *quantum gates* have been built, but only for a very small number of gates. Scaling them to practical size is a technologically hard (and some think unsolvable) problem.
  - They can be simulated on conventional computers, but the simulation requires exponential time.
  - They are not good at solving NP-hard problems

Quantum Computing and Cryptography

The discrete logarithm problem, integer factorization problem, and elliptic curve discrete logarithm problem are all special instances of an algebraic problem called the *hidden subgroup problem* (HSP).

- For these three problems, the hidden subgroup is abelian (commutative operation).
- There is a quantum algorithm due to Peter Shor (1997) for solving the HSP for finite abelian groups in polynomial-time.
- No polynomial-time quantum algorithm is known for solving the HSP for non-abelian groups.
  - Such an algorithm for the symmetric group would solve the graph isomorphism problem which is NP-hard.
  - Such an algorithm for the dihedral group would solve certain shortest vector problems in lattices; in general, the shortest vector problem is known to be NP-hard for randomized reductions.

Post-Quantum Cryptography

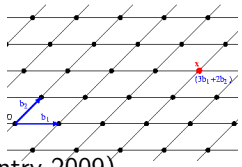
Post-quantum cryptography, aka *quantum-resistant cryptography* designs and analyzes cryptographic schemes whose security resides in the computational difficulty of certain mathematical (usually number theoretic) problems that to the best of our knowledge cannot be solved efficiently by a quantum computer.

On August 2, 2016, NIST began a competition for quantum-resistant public key algorithms; see `http://csrc.nist.gov/groups/ST/post-quantum-crypto/`.

Main Research Thrusts in Post-Quantum Cryptography

Lattice-based cryptography:

- *Learning With Errors*
- *Ring Learning With Errors*
- Lattices have other cryptographic applications:
  - NTRU PKC (Hoffstein, Piper, Silverman 1996)
  - Ideal lattices in *fully homomorphic encryption* (Gentry 2009)
  - and more ...



Code-based crypto (McEliece cryptosystem 1978)

Isogenies of supersingular elliptic curves (Diffie-Hellman, DSA)

Hash-based signature schemes (Merkle 1979, Lamport 1979)

Knapsack systems (after Merkle & Hellman 1978)

Multivariate equation based signature schemes (*Unbalanced Oil-and-Vinegar*, after Patarin 1997)

Summary of the Course

Things we covered:

- basics of perfect secrecy, one-time pad
- overview of symmetric key cryptography, AES
- public-key encryption, key exchange, digital signatures, formal security models
- overview of data integrity mechanisms
- overview of key management methodologies
- examples of applications using cryptography

Summary of the Course, cont.

Some (interesting!) things we did not:

- information-theoretic cryptography
- detailed design methodology for symmetric key primitives, attacks
- formal security notions for symmetric key crypto, data integrity
- *rigorous* security definitions and proofs for PKC
- variations of cryptographic primitives (block cipher modes with authentication, homomorphic encryption, signatures with additional properties, etc...)
- other cryptographic primitives (secret sharing, oblivious transfer, multiparty computation, etc...)
- quantum-resistant cryptography
- *detailed* discussion of using cryptography in practice
- ...

Cryptography in the Real World is Hard!