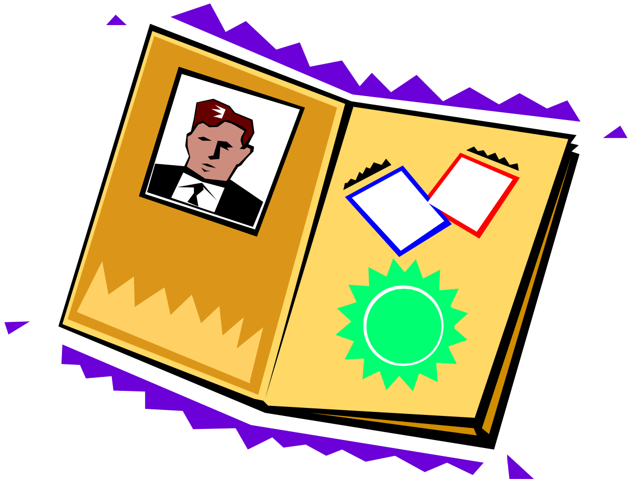
# Identification Schemes

# Lecture Outline

- Identification schemes
  - passwords
  - one-time passwords
  - challenge-response
  - zero knowledge proof protocols

# Authentication

- **Data source authentication (message authentication)**: a message is generated by a specific party.
- **Entity authentication (identification)**: the process whereby one party (the verifier) is assured of the identity of a second party (prover) involved in a protocol



## Requirements of Identification Protocols

- Requirements of identification protocols
  - for honest prover A and verifier B, A is able to convince B
  - no other party can convince B
  - in particular, B cannot convince C that it is A
- Kinds of attackers
  - passive and replay
  - active, man in the middle
  - the verifier

## Properties of Identification Protocols

- Computational efficiency
- Communication efficiency
- Security requirement of communication channels
- Trust in verifier
- Storage of secrets
- Involvement of a third party
- Nature of trust in the third party
- Nature of security: provable security

# Authentication Using Fixed Passwords

- Prover authenticates to a verifier using a password.
- Require secure communication channels
- Total trust in verifier
- Passwords must be kept in encrypted form or just digests of passwords are kept.
- Attacks:
  - Replay of fixed passwords
  - Online exhaustive password search
  - Offline password-guessing and dictionary attacks

## Unix crypt Algorithm

- Used to store Unix passwords
- Information stored is /etc/passwd is:
  - Iterated DES encryption of 0 (64 bits), using the password as key
  - 12 bit random salt taken from the system clock time at the password creation
- Unix use salting to change the expansion function in DES
  - to make dictionary attacks more difficult.
  - also to prevent use of off-the-shelf DES chips

## One-time passwords

- Each password is used only once
  - Defend against passive adversaries who eavesdrop and later attempt to impersonate
- Variations
  - shared lists of one-time passwords
    - challenge-response table
  - sequentially updated one-time passwords
  - one-time password sequences based on a one-way function

# Lamport's One-Time Password

Stronger authentication than password-based

- One-time setup:
  - A selects a value w, a hash function H(), and an integer t, computes $w_0 = H^t(w)$ and sends $w_0$ to B
  - B stores $w_0$
- Protocol: to identify to B for the ith time, $1 \le i \le t$
  - A sends to B: A, i, $w_i = H^{t-i}(w)$
  - B checks $i = i_A$, $H(w_i) = w_{i-1}$
  - if both holds, $i_A = i_A+1$

## Challenge-Response Protocols

- Goal: one entity authenticates to other entity proving the knowledge of a secret, ‘challenge’
- Time-variant parameters used to prevent replay, interleaving attacks, provide uniqueness and timeliness : nounce (used only once)
- Three types:
  – Random numbers
  – Sequences
  – Timestamp

# Challenge-Response Protocols

- **Random numbers**:
  - pseudo-random numbers that are unpredictable to an adversary;
  - need strong pseudo-random strings;
  - must maintain state;
- **Sequences**:
  - serial number or counters;
  - long-term state information must be maintained by both parties+ synchronization
- **Timestamp**:
  - provides timeliness and detects forced delays;
  - requires synchronized clocks.

# Challenge-response based on symmetric-key encryption

- Unilateral authentication, timestamp-based
  - A to B: $E_K(t_A, B)$
- Unilateral authentication, random-number-based
  - B to A: $r_B$
  - A to B: $E_K(r_B, B)$
- Mutual authentication, using random numbers
  - B to A: $r_B$
  - A to B: $E_K(r_A, r_B, B)$
  - B to A: $E_K(r_B, r_A)$

## Challenge-Response Protocols Using Digital Signatures

- unilateral authentication with timestamp

  $A \rightarrow B$: $cert_A, t_A, B, S_A(t_A, B)$
- unilateral authentication with random numbers

  $A \leftarrow B$: $r_B$

  $A \rightarrow B$: $cert_A, r_A, B, S_A(r_A, r_B, B)$
- mutual authentication with random numbers

  $A \leftarrow B$: $r_B$

  $A \rightarrow B$: $cert_A, r_A, B, S_A(r_A, r_B, B)$

  $A \leftarrow B$: $cert_B, A, S_B(r_B, r_A, A)$

## Zero-Knowledge Protocols

- **Motivation:**
  - Password-based protocols: when Alice authenticates to a server, she gives her password, so the server can then impersonate her.
  - Challenge-response improves on this, but still reveals partial information.
- **Zero-knowledge protocols**: allows a prover to prove that is posses a secret without revealing any information of use to the verifier.

## Observations on the Protocol

- Multiple rounds
- Each round consists of 3 steps
  - commit
  - challenge
  - respond
- If challenge can be predicted, then cheating is possible.
  - cannot convince a third party (even if the party is online)
- If respond to more than one challenge with one commit, then the secret is revealed.

## Zero Knowledge Proofs

- A kind of interactive proof system
  - – proof by interaction
- Involves a prover and a verifier
- Proving without revealing any other information

# Two Kinds of Zero-Knowledge Proofs

- ZK proof of a statement
  - convincing the verifier that a statement is true without yielding any other information
  - example of a statement, a propositional formula is satisfiable
- ZK proof of knowledge of a secret
  - convincing the verifier that one knows a secret, e.g., one knows the square root modulo N=pq

## Properties Zero-Knowledge Proofs

- Properties of ZK Proofs:
  - completeness
    - honest prover who knows the secret convinces the verifier with overwhelming probability
  - soundness
    - no one who doesn't know the secret can convince the verifier with nonnegligible probability
  - zero knowledge
    - the proof does not leak any additional information
- How to formalize soundness and ZK?

## Formalizing the Soundness Property

- The protocol should be a “proof of knowledge”
- A knowledge extractor exists
  - that given a prover who can successfully convince the verifier, can extracts the secret

## Formalizing ZK property

- For every possible verifier algorithm, a simulator exists
  - taking what the verifier knows before the proof, can generate a communication transcript that is indistinguishable from one generated during ZK proofs
  - honest verifier ZK considers only the verifier algorithm in the protocol
- Three kinds of indistinguishability
  - perfect (information theoretic)
  - statistical
  - computational

# Schnorr Id protocol (ZK Proof of Discrete Log)

- System parameter: p, q, g
  - q | (p-1) and g is an order q element in $Z_p^*$
- Public identity: v
- Private authenticator: s  $v = g^{-s} \bmod p$
- Protocol

1. A: picks random r in [1..q], sends $x = g^r \bmod p$,
2. B: sends random challenge c in $[1..2^t]$
3. A: sends y=sc+r mod q
4. B: accepts if $x = (g^y v^c \bmod p)$

# Security of Schnorr Id protocol

- probability of forgery: $1/2^t$
- soundness:
- ZK property
  - honest verifier ZK
  - not ZK if $2^t$>log n is used

## Converting Interactive ZK to Non-interactive ZK

- The only interactive role played by the verifier is to generate random challenges
  - challenges not predictable by the prover
- The same thing can be done using one-way hash functions

# Interactive ZK Implies Signatures

- Given a message M, replace the random challenge of the verifier by the one-way hash c=h(x||M)