# Introduction to Information Security

## Lecture 8: Cryptographic Protocols

2007. 6.

**Prof. Byoungcheon Lee**
sultan (at) joongbu . ac . kr

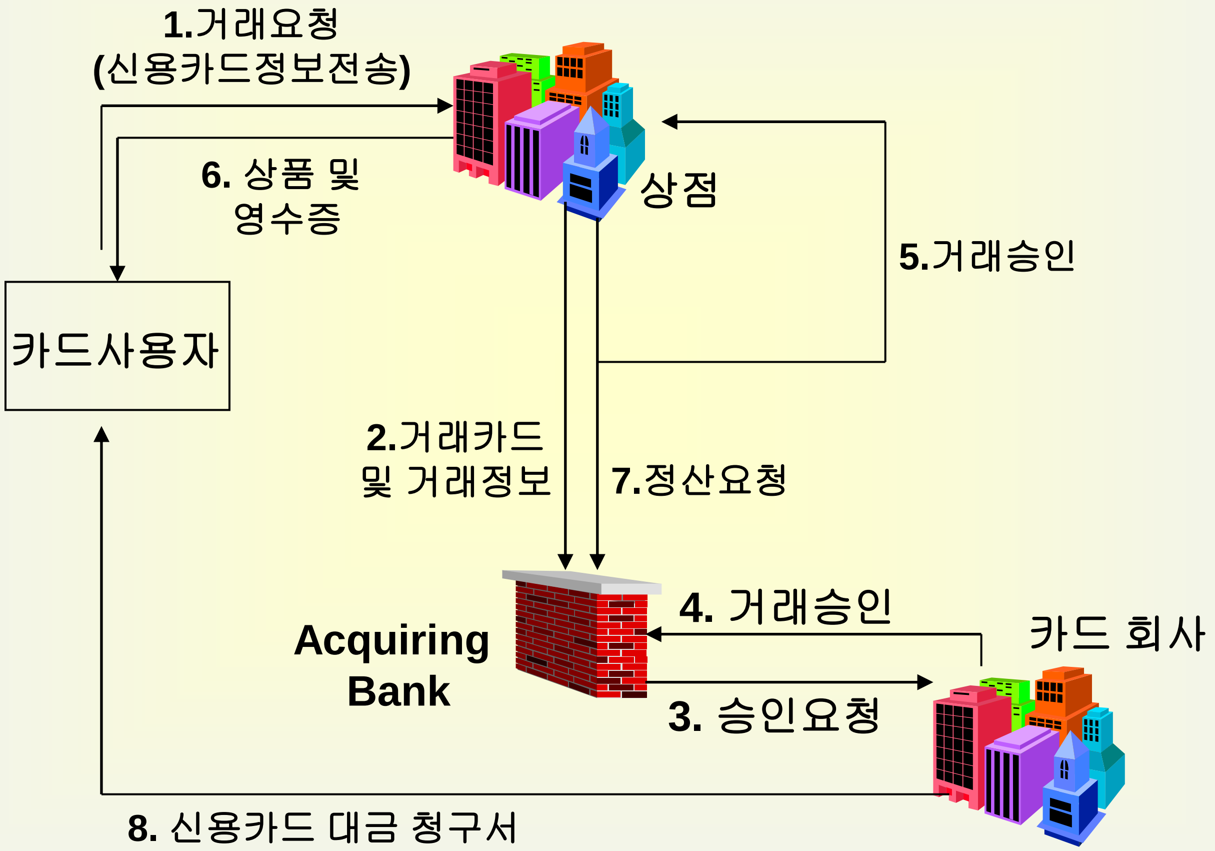
**Information and Communications University**

# Contents

1. Cryptographic Protocols
2. Flipping Coins over the Telephone
3. Special Signatures
4. Secret Sharing and Threshold Cryptography
5. Zero-knowledge Proofs
6. Identification, Authentication

# 1. Cryptographic Protocols

# Typical E-commerce Scenario



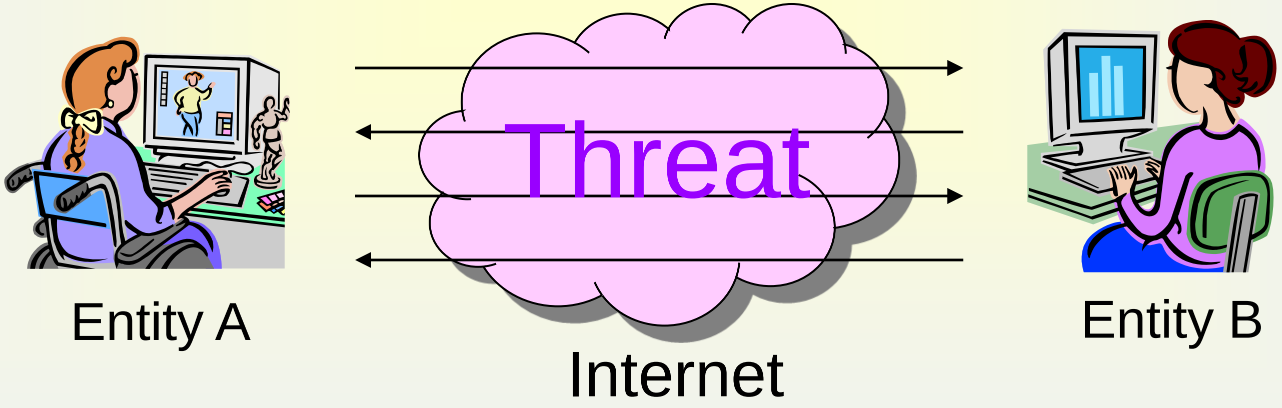
- Combination of lots of computation / communication.
- Must be fare to all participating entities

# Cryptographic Protocols

- **Cryptographic algorithms**
  - ✓ Algorithm executed by a single entity
  - ✓ Algorithms performing cryptographic functions
  - ✓ Encryption, Hash, digital signature, etc…
- **Cryptographic protocols**
  - ✓ Protocols executed between multiple entities through pre-defined steps of communication performing security-related functions
  - ✓ Perform more complicated functions than what the primitive algorithms can provide
  - ✓ Primitives: Key agreement, secret sharing, blind signature, coin toss, secure multiparty computations, etc …
  - ✓ Complex application protocols: e-commerce, e-voting, e-auction, etc …

# Cryptographic Protocols

- ➢ **Protocols**
  - ✓ **Designed to accomplish a task through a series of communication steps, involving two or more entities**
- ➢ **Cryptographic Protocols**
  - ✓ **Protocols that use cryptography**
  - ✓ **Non-face-to-face interaction over an open network**
  - ✓ **Cannot trust other entities**



Threat

Entity A

Entity B

Internet

# Security Requirements in Protocols

- ✓ Confidentiality
- ✓ Integrity
- ✓ Authentication
- ✓ Non-repudiation
- ✓ Correctness
- ✓ Verifiability
- ✓ Fairness
- ✓ Anonymity
- ✓ Privacy
- ✓ Robustness
- ✓ Efficiency
- ✓ Etc……

**Combinations of these requirements according to applications**

# Protocol Primitives

- **Coin Toss game over Communication Network**
  - ✓ Two parties play coin toss game over the communication network
  - ✓ Can it be made fair?
- **Blind Signatures**
  - ✓ Signer signs a document without knowledge of the document and the resulting signature
  - ✓ Message and the resulting signature are hidden from the signer
  - ✓ Many applications which require anonymity or privacy
  - ✓ Digital cash, e-voting
- **Key Agreements**
  - ✓ Two or more parties agree on a secret key over communication network in such a way that both influence the outcome.
  - ✓ Do not require any trusted third party (TTP)

# Protocol Primitives

- **Secret Sharing**
  - ✓ Distribute a secret amongst a group of participants
  - ✓ Each participant is allocated a share of the secret
  - ✓ Secret can be reconstructed only when the shares are combined together
  - ✓ Individual shares are of no use on their own.

- **Threshold Cryptography**
  - ✓ A message is encrypted using a public key and the corresponding private key is shared among multiple parties.
  - ✓ In order to decrypt a ciphertext, a number of parties exceeding a threshold is required to cooperate in the decryption protocol.

# Protocol Primitives

- **Zero-knowledge Proofs**
  - ✓ **An interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the validity of the statement.**

- **Identification, Authentication**
  - ✓ **Over the communication network, one party, Alice, shows to another party, Bob, that she is the real Alice.**
  - ✓ **Allows one party, Alice, to prove to another party, Bob, that she possesses secret information without revealing to Bob what that secret information is.**

# Protocol Primitives

- ➢ Private Information Retrieval (PIR)
  - ✓ allow a client to query a database without the server learning what the query is.

- ➢ Secure Multiparty Computation (SMC)
  - ✓ A set of parties with private inputs wish to compute some joint function of their inputs.
  - ✓ Parties wish to preserve some security properties. E.g., privacy and correctness.

# Application Protocols

- Electronic Commerce
  - ✓ SET (Secure Electronic Transaction) – Credit card transaction
  - ✓ Digital cash, micropayment, e-check, e-money
  - ✓ e-auction
  - ✓ e-banking
- e-government
- e-voting
- Fair exchange of digital signature (for contract signing)

- Application Scenarios
  - Traditional applications transfer to electronic versions
  - New applications appear with the help of crypto

# 2. Flipping Coins over the Telephone

# Coin Toss Game

- **Scenario**

  **Alice and Bob are getting a divorce and have to discuss who gets what. . .**

  **. . . and they can’t stand facing each other. . .**

  **. . . they don’t seem to agree about one thing: who gets the car?**

  **Finally they decide to flip a coin. . .**

- **The problem:**

  **If they don’t trust each other, how can they flip a coin over the telephone?**

**Head, Alice**

**Tail, Bob**

# Bit Commitment (BC)

- **Scenario**

  **Alice makes a commitment simply by picking a value from a finite set and committing to her choice in a way such that she cannot change her mind later. Later she can, if she wants, reveal her choice.**

- **Protocol**

1. **Alice writes down a bit b on a piece of paper, puts it inside a box and locks the box;**
2. **Alice gives the box to Bob;**
3. **If Alice wants, she can reveal her commitment by opening the box in front of Bob.**

# Bit Commitment (BC)

- ➢ Required properties of bit commitment (From Alice to Bob)
    1. Binding property: Alice can’t change her mind;
    2. Hiding property: Bob can’t open the box, unless Alice unlocks it.

- ➢ Construction of BC
    - ✓ Using one-way function : Hash functions, Public key encryptions

# Flipping Coins using BC

- Set up (Bit commitment using Hash function)

  Alice and Bob agree that Alice will flip a coin and Bob will try to guess. They agree on a hash function h( ).

- The Coin Flipping Protocol is as follows:

1. (Coin Flip by Alice) Alice randomly chooses x and computes y=h(x). Alice commits to x by sending y to Bob;

2. (Call head or tail by Bob) Bob guesses and calls whether x is even or odd number;

3. (Find the result) Alice reveals x, then Bob checks y=h(x) holds. If Bob’s guess is correct, Bob wins, otherwise Alice wins.

# Play Further Games?

- Millionaire Problem (by Andrew Yao in 1982)
  - ✓ Two millionaires, Alice and Bob, want to know who is richer, without revealing their actual wealth.

- Mental Poker
  - ✓ Play a fair game (poker) over distance without the need for a trusted third party

- Secure multi-party computation
  - ✓ We have a given number of participants (p1, p2, ..., pN), each having a private data, respectively (d1, d2, ..., dN).
  - ✓ The participants want to compute the value of a public function F on N variables at the point (d1, d2, ..., dN).
  - ✓ No participant can learn more from the description of the public function and the result of the global calculation.

# 3. Special Signatures

- Blind Signature
- Proxy Signature

# Special Signatures

- Special Signatures
  - ✓ Digital signatures with additional features (anonymity, privacy, efficiency, delegation,…)
  - ✓ Digital signature variants considering various business application scenarios

- Blind signature
  - ✓ A user can receive a signature of a signer without revealing the message and the resulting signature to the signer
- Proxy signature
  - ✓ An original signer delegate his/her signing capability to a proxy signer, and then the proxy signer signs documents on behalf of the original signer
- Self-certified signature
  - ✓ Signature verification and certificate verification are done efficiently in a single logical step.

# Special Signatures

- ❖ Undeniable signature
  - ✓ A recipient of a signature cannot check the validity by himself
  - ✓ The recipient has to interact with the signer in order to be convinced of the validity of signature

- ❖ Designated confirmer signature
  - ✓ The recipient has to interact with an entity called the confirmer who has been designated by the signer

- ❖ Nominative signature
  - ✓ a nominator (signer) and a nominee (verifier) to jointly generate and publish a signature in such a way that only the nominee can verify the signature and if necessary, only the nominee can prove to a third party that the signature is valid.

# Special Signatures

- **Designated-verifier signature**
  - ✓ **the designated-verifier can be convinced of the validity of the signature, but he/she is unable to transfer the conviction to other entity.**

- **Limited-verifier signature**
  - ✓ **The limited verifier is able to transfer the proof to convince another entity (perhaps a judge). However, such a proof given to the judge is not transferrable to another third entities**

# Special Signatures

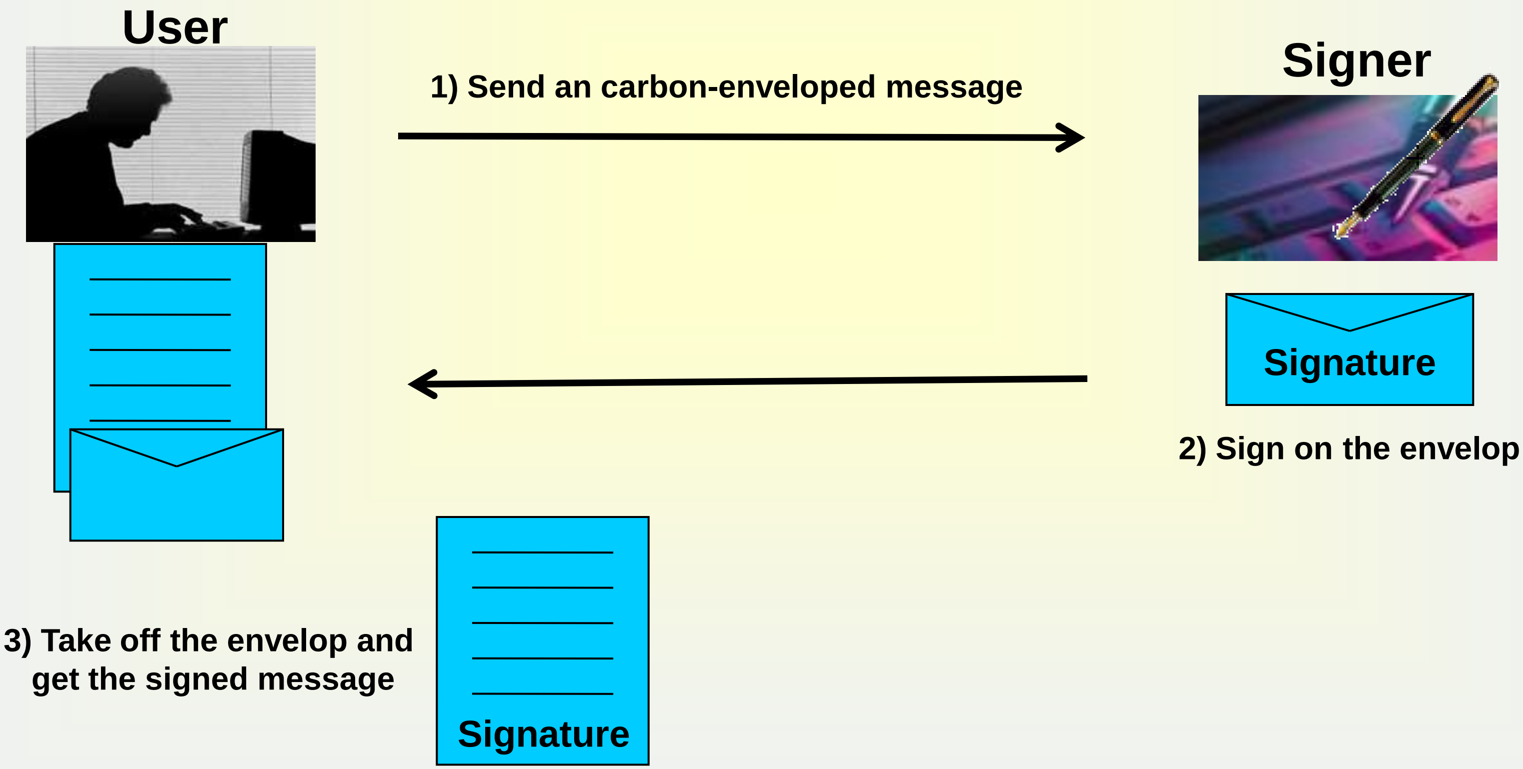
- **Group signature**
  - ✓ **a signature scheme which allows a member of a group to anonymously sign a message on behalf of the group.**
  - ✓ **A group manager can reveal the identity of the real signer**

- **Ring signature**
  - ✓ **A type of digital signature that can be performed by any member of a group of users. Therefore, a message signed with a ring signature is endorsed by someone in a particular group of people.**
  - ✓ **One of the security properties of a ring signature is that it should be difficult to determine *which* of the group members' keys was used to produce the signature.**

# Blind Signature

## Signing without seeing the message

- We should not reveal the content of the letter to the signer.
- For example, using a carbon-enveloped message



User

Signer

1) Send an carbon-enveloped message

Signature

2) Sign on the envelop

3) Take off the envelop and
get the signed message

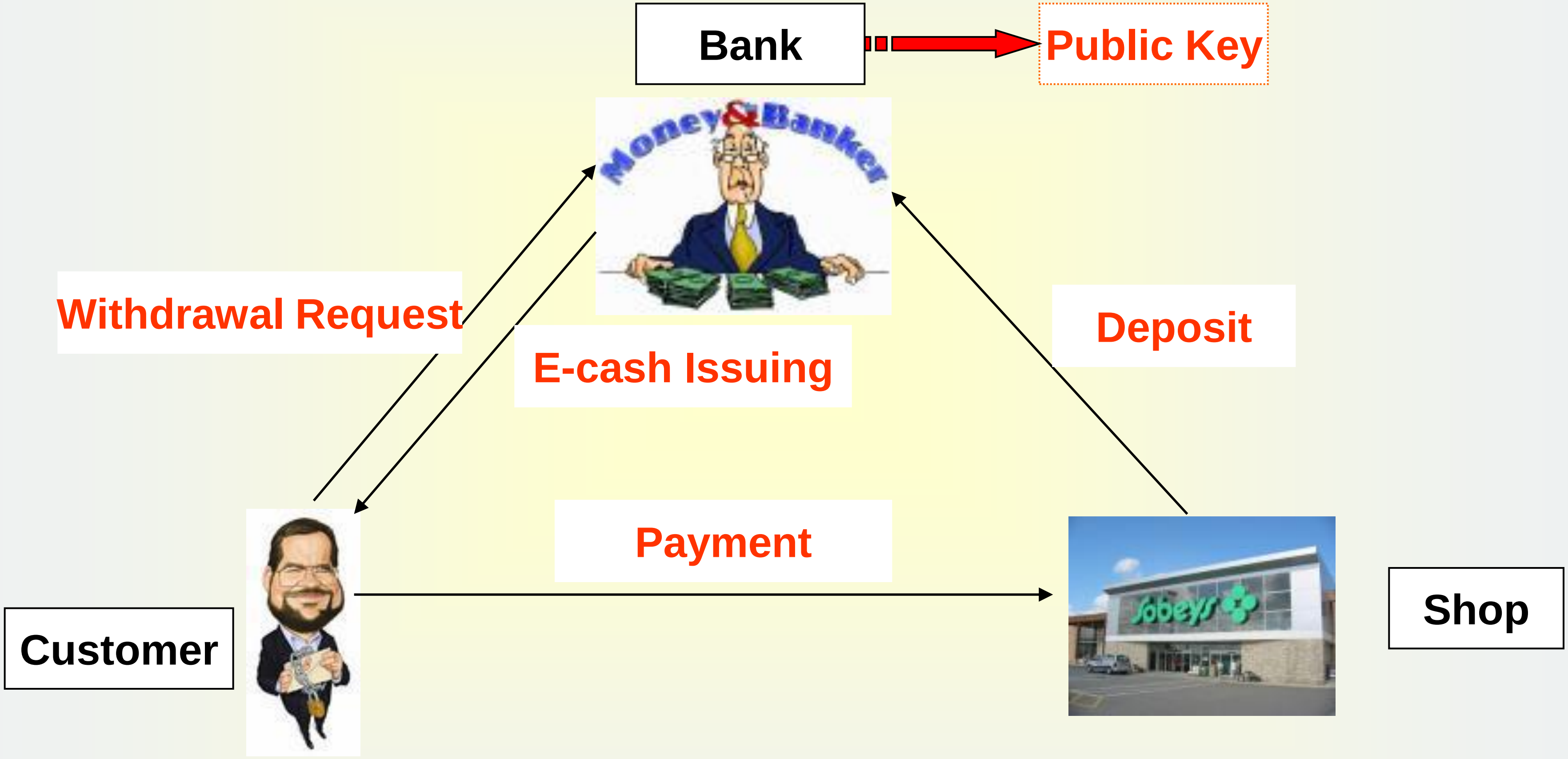
Signature

# Motivation of Blind Signature

- One interesting question of public key cryptosystem is whether we can use digital signature to create some form of digital currency. The scenario is described as follows:
  1) A bank published his public key.
  2) When one of his customer makes a withdrawal from his account, the bank provides it with a digitally signed note that specifies the amount withdrawn.
  3) The customer can present it to a merchant, who can then verify the bank's signature.
  4) Upon completing a transaction, the vender can then remit the note to the bank, which will then credit the vendor the amount specified in the note.
  5) This note is, in effect, a digital monetary instrument, we called it as "Electronic Cash or E-Cash".

- Privacy issue of digital cash???
  - ✓ The bank can easily trace a cash to a specific user.

# E-Cash Scenario



Bank

Public Key

Withdrawal Request

E-cash Issuing

Deposit

Payment

Customer

Shop

# David Chaum’s Blind Signature

- David Chaum proposed a very elegant solution to this problem, known as blind signature.

*He is also named as the “father of E-cash”*

# Blind Signature

**Blind signature scheme is a protocol that allows the provider to obtain a valid signature for a message *m* from the signer without him seeing the message and its signature.**

**If the signer sees message *m* and its signature later, he can verify that the signature is genuine, but he is unable to link the message-signature pair to the particular instance of the signing protocol which has led to this pair.**

**Many applications**

- ✓ **Useful when values need to be certified, yet anonymity should be preserved**
- ✓ **e-cash, e-voting**

# Blind Signature

Protocol Steps

1) Alice takes the document and uses a “blinding factor” to blind the document. (Blinding Phase)

2) Alice sends the blinded document to Bob and Bob signs the blinded document. (Signing Phase)

3) Alice can remove the blinding factor and obtain the signature on the original document. (Unblinding Phase)

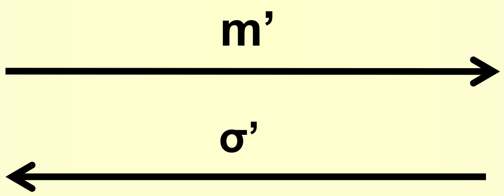
# RSA-based Blind Signature

**User** Get a signature for a message m. **Signer**

### (1) Blinding

$r \in Z_N^*$

$m' = H(m)\, r^e \bmod N$



m’

σ’

### (2) Signing

$\sigma' = m'^d \bmod N$

### (3) Unblinding

$\sigma = \sigma'\, r^{-1} \bmod N$

$$\sigma = \sigma'\, r^{-1} \bmod N = (H(m)\, r^e)^d\, r^{-1} \bmod N = H(m)^d \bmod N$$

σ is a valid signature of the signer
The signer cannot have any information on m and σ.

# Schnorr-based Blind Signature

$y = g^x \bmod p$

public: $p, q, g, y$

private: $x$

**User** **Signer**

**(1) Challenge**

$k \in_R Z_q^*$

$r = g^k \bmod p$

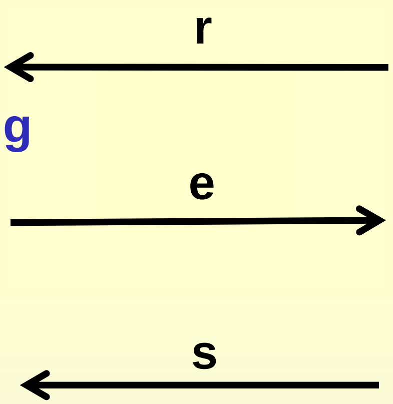
**(2) Blinding**

$a, b \in_R Z_q^*$

$r' = rg^a y^b \bmod p$

$e' = H(m, r')$

$e = e' + b \bmod q$



**(3) Signing**

$s = k + ex \bmod q$

**(4) Unblinding**

$g^s \overset{?}{=} ry^e \bmod p$

$s' = s + a \bmod q$

**(r’,s’) is an unknown signature**

**for the unknown message m**

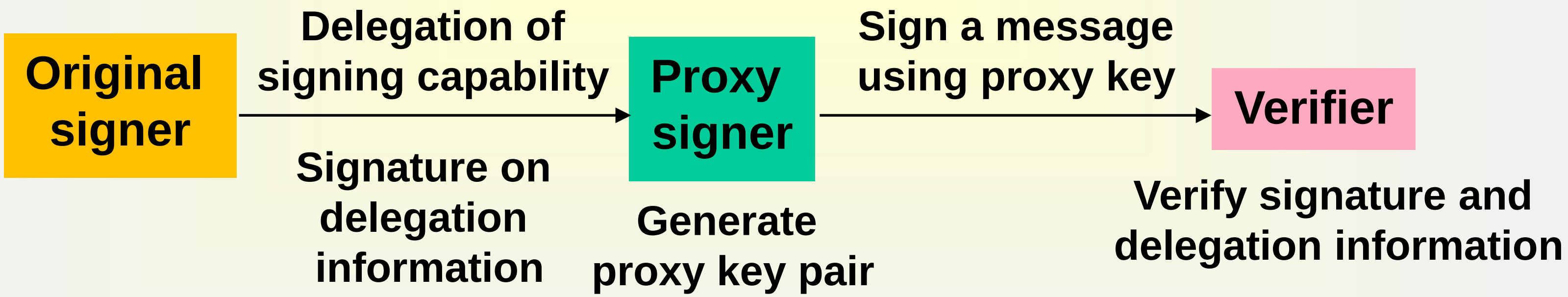
$$g^{s'} = g^{s+a} = g^k y^{e'+b} g^a = g^k g^a y^b y^{H(m,r')} = r' y^{e'} = r' y^{H(m,r')}$$

# Proxy Signature

- ❖ A scenario in the real world
  - ✓ Each student’s transcript of academic record should be signed by the department head.
  - ✓ The department head is too busy to sign all the transcripts , so he assigns a clerk to sign them.
  - ✓ How to delegate the right of signing transcripts to the clerk?
  - ✓ The department head gives a department chop(seal) to the clerk. The clerk signs the transcripts on behalf of the department head.
- ❖ Problems in handwritten proxy signature
  - ✓ It is difficult to prevent the proxy from signing documents unfavorable to the original signer.
  - ✓ It is also difficult to prevent the proxy signer from passing the chop to another person.
- ❖ Proxy signature: In digital case, these problems can be solved by using cryptographic means.

# Proxy Signature

- ❖ Overview of proxy signature
  - ✓ An original signer delegates his/her signing capability to a proxy signer (issues a proxy key pair to proxy signer)
  - ✓ Proxy signer signs a message on behalf of the original signer using the proxy key pair
  - ✓ A receiver verifies the signature itself and original signer’s delegation together


Original signer
Delegation of signing capability
Signature on delegation information
Proxy signer
Generate proxy key pair
Sign a message using proxy key
Verifier
Verify signature and delegation information

# Classification of Proxy Signature

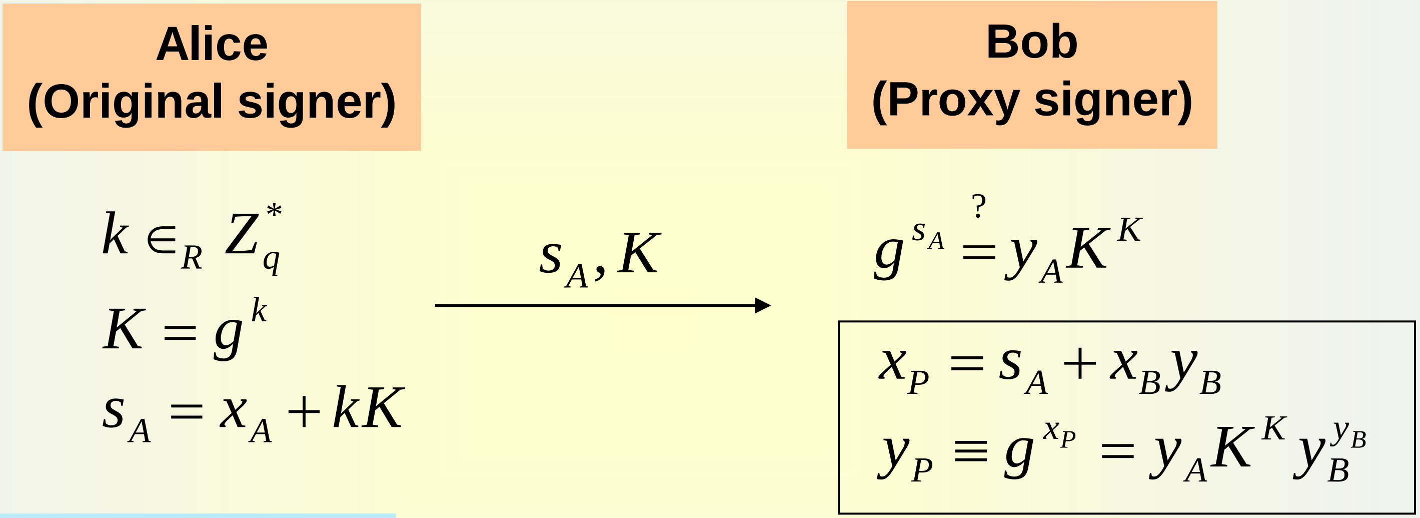
- ❖ Full delegation : gives the original signer's private key to proxy signer

- ❖ Partial delegation : generates a new proxy key pair
  - ✓ Proxy unprotected : original signer knows the proxy key pair
  - ✓ Proxy protected : proxy key pair is hidden from the original signer
  - ✓ Partial delegation with warrant : contains warrant information

- ❖ Delegation by warrant : the original signer gives a signed warrant to the proxy signer.

# Proxy Signature by MUO

❖ **Proposal by Mambo, Usuda, Okamoto in 1996**



**Alice (Original signer)**

$$k \in_R Z_q^*$$

$$K = g^k$$

$$s_A = x_A + kK$$

$$\xrightarrow{s_A, K}$$

**Bob (Proxy signer)**

$$g^{s_A} \overset{?}{=} y_A K^K$$

$$\boxed{\begin{array}{l} x_P = s_A + x_B y_B \\ y_P \equiv g^{x_P} = y_A K^K y_B^{y_B} \end{array}}$$

- **Use proxy signer's key pair**
- **Non-interactive proxy key issuing**

**Signature creation:** $m, S(x_P, m), K, y_B$

**Verification of delegation:** $y_P = y_A K^K y_B^{y_B}$

**Verification of signature:** $V(y_P, m, S(x_P, m)) \overset{?}{=} true$

# 4. Secret Sharing and Threshold Cryptography

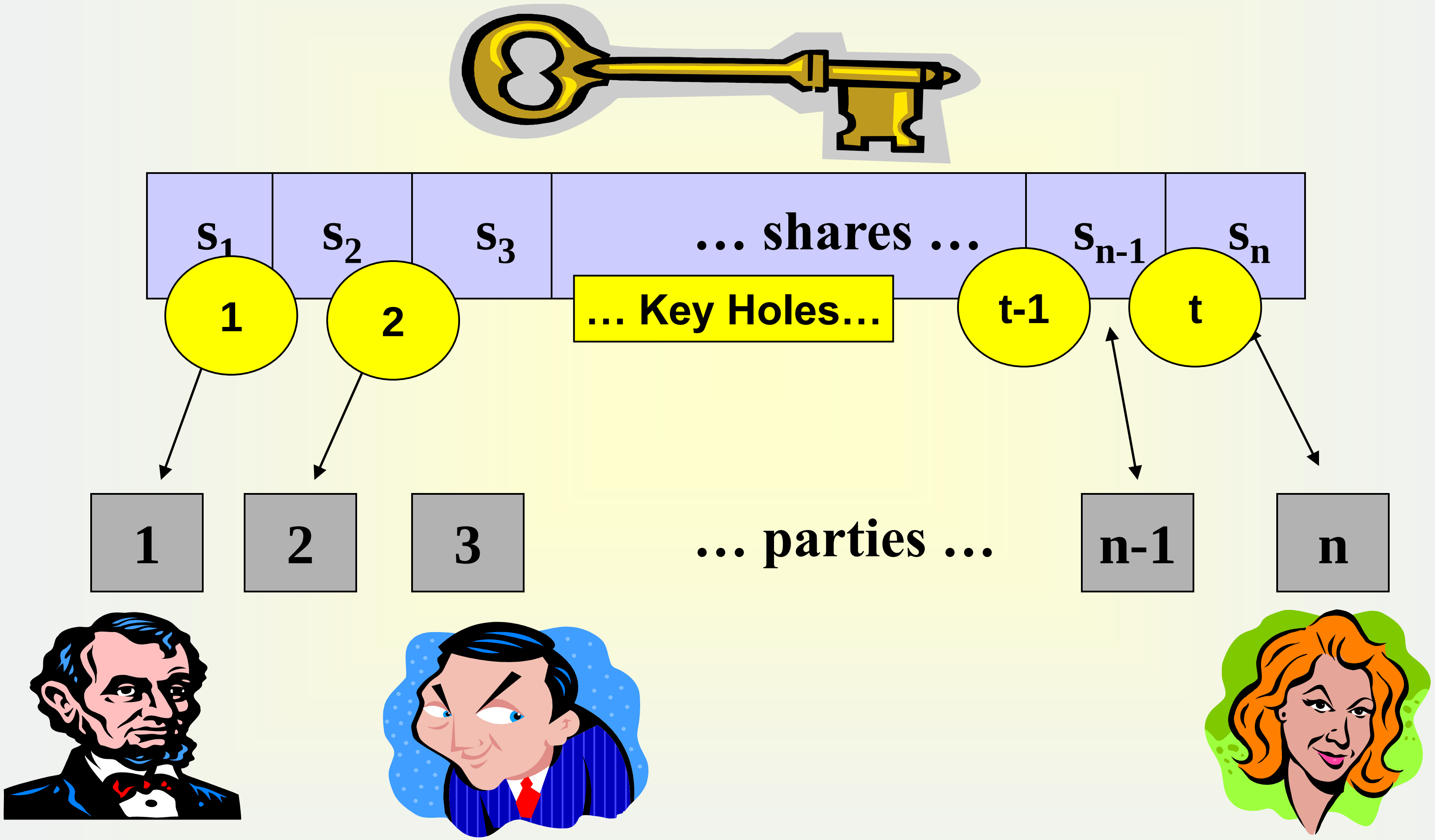
# Secret Sharing

- ➢ Background
  - ✓ Some secrets are too important to be kept by one person.
  - ✓ "*It is easier to trust the many than the few*"
  - ✓ Secrecy (trust) and robustness

- ➢ Example:
  - ✓ Purported by Time Magazine in 1992 that the Russian nuclear weapon systems were protected by a two-out-of-three access mechanism – President, Defense Minister and Defense Ministry

- ➢Secret Sharing
  - ✓ Distribute a secret amongst a group of participants
  - ✓ Each participant is allocated a share of the secret
  - ✓ Secret can be reconstructed only when the shares are combined together
  - ✓ Individual shares are of no use on their own.

# Secret Sharing



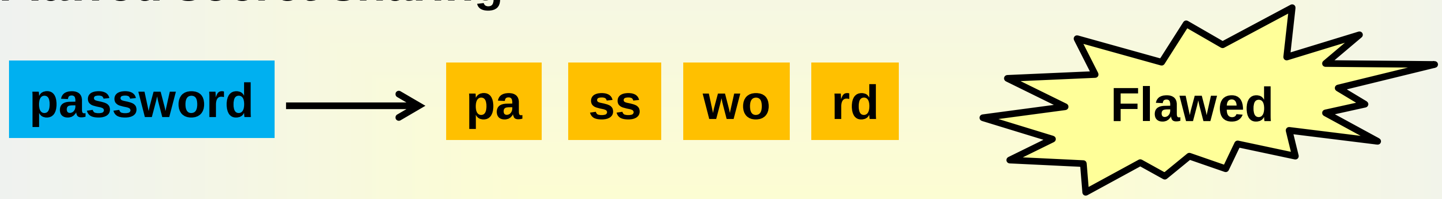
$s_1$ $s_2$ $s_3$ … shares … $s_{n-1}$ $s_n$

1 2 … Key Holes… t-1 t

1 2 3 … parties … n-1 n

# Secret Sharing

- **Flawed secret sharing**



password

pa ss wo rd

Flawed

- **Trivial secret sharing**

  A secret $s$ is distributed as $s = b_1 \oplus b_2 \oplus \ldots \oplus b_{n-1} \oplus b_n$

  1) Choose random numbers $b_1,\ldots,b_{n-1}$
  2) Compute $b_n = b_1 \oplus b_2 \oplus \ldots \oplus b_{n-1} \oplus s$

**All *n* shares should be present to recover the secret *s* (Not robust)**

# Threshold Secret Sharing

- **Scenario**

  **For example, imagine that the Board of Directors of Coca-Cola would like to protect Coke's secret formula. The president of the company should be able to access the formula when needed, but in an emergency any 3 of the 12 board members would be able to unlock the secret formula together.**
  **This can be accomplished by a secret sharing scheme with *t* = 3 and *n* = 15, where 3 shares are given to the president, and 1 is given to each board member.**

- **Security Issues**
  - **Secrecy: resistance against any misbehavior**
  - **Robustness: reliability against any possible error**

# Threshold Secret Sharing

- **(t, n) Secret Sharing with t<n**
  - ✓ A secret K is shared among n shares
  - ✓ Among n shares t shares have to cooperate to recover the secret K
  - ✓ Robust against partial error
  - ✓ Shamir's secret sharing, Blakley's secret sharing

- **The goal is to divide a secret K into n pieces $s_1, \ldots s_n$ in such a way that:**
  - ✓ Any group of *t* or more users can jointly obtain the secret; knowledge of any *t* or more $s_i$ pieces makes K easily computable.
  - ✓ Any group of *t-1* or less users cannot jointly obtain any information about the secret. Knowledge of any *t-1* or fewer $s_i$ pieces leaves K completely undetermined.

- **Provides tradeoff between security and reliability according to the choice of t and n.**
  - ✓ Higher t gives higher security, lower reliability
  - ✓ Lower t gives lower security, higher reliability

# Shamir's Secret Sharing

- (t, n) Secret Sharing
  - ✓Secret information *K*
  - ✓*n* share holders ($P_1,\ldots,P_n$)
  - ✓Using *t-1* degree random polynomial with random coefficient

(Step 1. Polynomial construction) A dealer selects a secret, *K* ( < *p* : prime) as a constant term and *t-1* degree random polynomial with arbitrary coefficients as :

$$F(x) = K + a_1x + a_2x^2 + \ldots + a_{k-1}x^{t-1} \bmod p$$

(Step 2. Share distribution) Distributes *F(i) (i=1,…,n)* securely to share holders $P_i$.

(Step 3. Secret recovery) When *t* shares $\Lambda=(K_1, K_2,\ldots,K_t)$ among *n* are given, recover *K* by using the ***Lagrange Interpolation***

$$K = \sum_{j\in\Lambda} K_j \lambda_{j,\Lambda} \bmod p, \quad \text{where } \lambda_{j,\Lambda} = \prod_{l\in\Lambda\setminus\{j\}} \frac{l}{l-j}$$

# Shamir's Secret Sharing

- **Example**
  - ✓(3,5) secret sharing
  - ✓K=11, p=17
  - ✓Construct a degree 2 random polynomial

    $F(x) = K + a_1x + a_2x^2 \bmod p$
  - ✓For a random choice $a_1=8, a_2=7$

    $F(x) = 11 + 8x + 7x^2 \bmod 17$
  - ✓Share distribution

    $K_1 = F(1) = 7\times1^2 + 8\times1 + 11 \equiv 9 \quad \bmod 17$

    $K_2 = F(2) = 7\times2^2 + 8\times2 + 11 \equiv 4 \quad \bmod 17$

    $K_3 = F(3) = 7\times3^2 + 8\times3 + 11 \equiv 13 \quad \bmod 17$

    $K_4 = F(4) = 7\times4^2 + 8\times4 + 11 \equiv 2 \quad \bmod 17$

    $K_5 = F(5) = 7\times5^2 + 8\times5 + 11 \equiv 5 \quad \bmod 17$

    $K_1, K_2, K_3, K_4, K_5$ : shares given to $(P_1,\ldots,P_5)$

# Shamir's Secret Sharing

- **Example**
  - **Secret recovery by equation solving**

    **From $K_2$, $K_3$, $K_4$, we can recover $K = 11$**

    $$a \times 2^2 + b \times 2 + K \equiv 4 \quad \text{mod } 17$$

    $$a \times 3^2 + b \times 3 + K \equiv 13 \quad \text{mod } 17$$

    $$a \times 4^2 + b \times 4 + K \equiv 2 \quad \text{mod } 17$$

    **Solve the 3 polynomial equations with 3 variables to get $K$.**

  - **Using the Lagrange interpolation**

    **For $\Lambda=(K_1, K_2, K_3)$**

$$K = K_1(\frac{2}{2-1}\frac{3}{3-1}) + K_2(\frac{1}{1-2}\frac{3}{3-2}) + K_3(\frac{1}{1-3}\frac{2}{2-3})$$

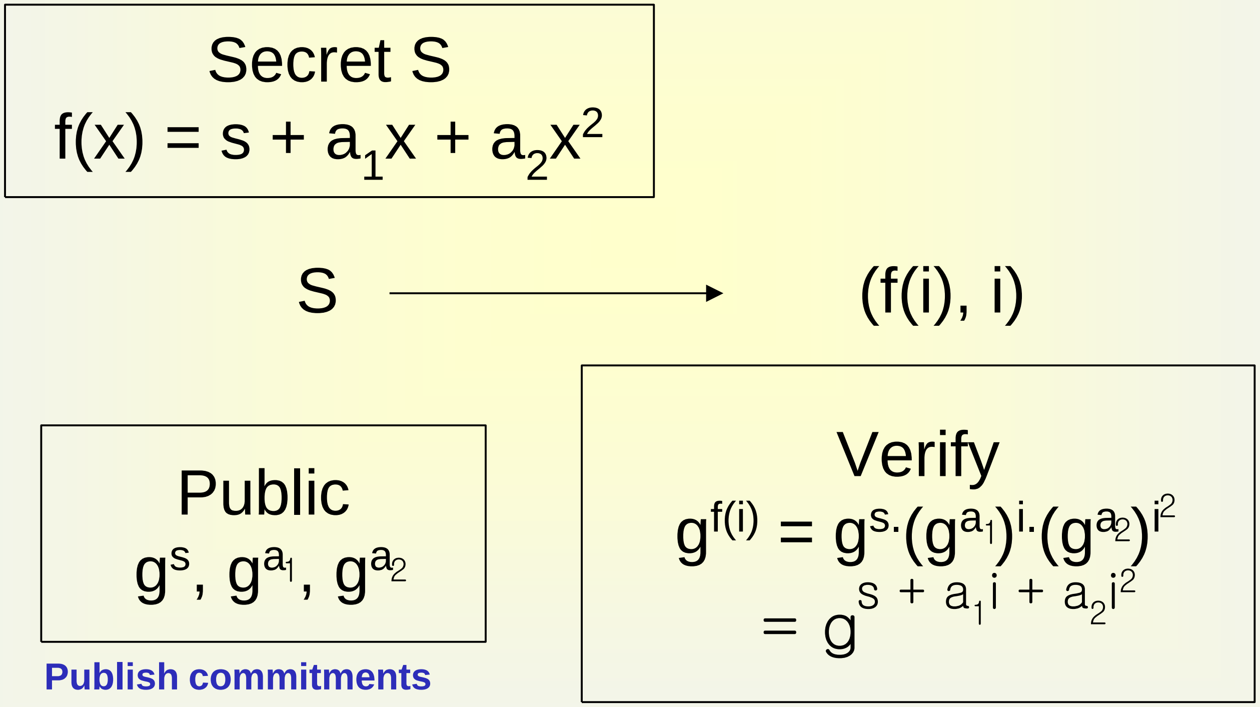
$$= 9 \cdot 3 + 4 \cdot (-3) + 13 \cdot 1 \bmod 17 = 11$$

# Shamir's Secret Sharing

- **Exercise. Construct a Shamir's secret sharing scheme in the following setting --- (5,7) secret sharing with p=23, K=19**
  1. **Construct a random polynomial**
  2. **Share distribution**
  3. **Secret recovery**

# Verifiable Secret Sharing

- How to have a confidence that your share is a correct one?
- Feldman's Verifiable Secret Sharing (VSS)



Secret S

$$f(x) = s + a_1x + a_2x^2$$

$$S \longrightarrow (f(i), i)$$

Public

$$g^{s},\ g^{a_1},\ g^{a_2}$$

Verify

$$g^{f(i)} = g^{s}\cdot(g^{a_1})^{i}\cdot(g^{a_2})^{i^2}$$

$$= g^{s + a_1 i + a_2 i^2}$$

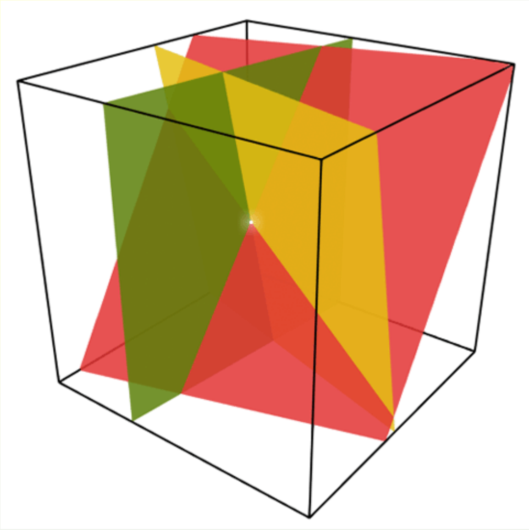
Publish commitments to the coefficients

Verify the correctness of his share f(i)

# Blakley's Secret Sharing Scheme

- **Two nonparallel lines in the same plane intersect at exactly one point.**
- **Three "nonparallel" planes in space intersect at exactly one point.**
- **More generally, any *n-dimensional hyperplanes* intersect at a specific point.**

- **The secret may be encoded as any single coordinate of the point of intersection.**



# Threshold Cryptography

**A public key is published, but the corresponding private key is shared among multiple parties.**

- **Threshold Encryption Scheme**
  - ✓ **A message is encrypted using the public key**
  - ✓ **In order to decrypt a ciphertext, a number of parties exceeding a threshold is required to cooperate in the decryption protocol.**

- **Threshold Signature Scheme**
  - ✓ **To sign a message, a number of parties exceeding a threshold is required to cooperate in the signing protocol.**
  - ✓ **A signature can be verified using the public key.**

# 5. Zero-Knowledge Proofs

# What does one learn from a proof?

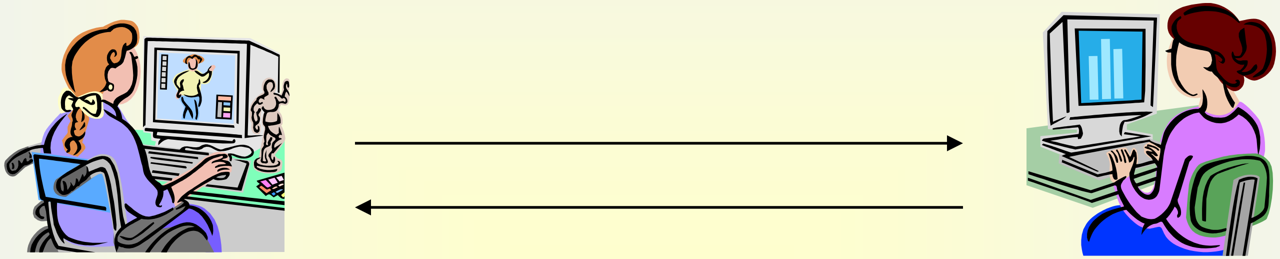
- The validity of the assertion being proven (by definition).

  Anything else?

- Classical (NP) proofs: Upon receiving a proof of statement *x*, one gains the ability to prove *x* to others.
  - Theorem and proof in math textbook
  - You learn to get Knowledge.

- Interactive proofs: Can be “zero-knowledge”, i.e. reveal nothing other than the validity of the assertion being proven. ⇒ verifier does not gain ability to prove same assertion to others!
  - The assertion is a precious information (your password)
  - Your protocol is designed to achieve Zero-Knowledge
  - Proofs can be used again

# Interactive Proof Systems



Prover

Verifier

- ➢ **Prover knows a secret (precious) information.**
- ➢ **Wants to prove that he knows it, but do not want to reveal it.**

- ➢ **Verifier is curious about prover’s knowledge.**
- ➢ **He will query difficult questions, s.t. the secret should be used to answer.**
- ➢ **Should be random questions**

**The verifier’s strategy is a probabilistic polynomial-time (PPT) procedure.**

# Interactive Proof Systems

- **An Interactive Proof System for a language L is a two-party game between a prover and a verifier that interact on a common input in a way satisfying the following properties:**

  - **Completeness: There exists a prover strategy P, such that for every x∈L, when interacting on a common input x, the prover P convinces the verifier with probability at least 2/3.**

  - **Soundness: For every x∉L, when interacting on the common input x, any prover strategy P* convinces the verifier with probability at most 1/3.**

# Zero-Knowledge Proofs

- Interactive proofs that reveal nothing other than the validity of assertion being proven
- A zero-knowledge proof is a way that a “prover” can prove possession of a certain piece of information to a “verifier” without revealing it.
- This is done by manipulating data provided by the verifier in a way that would be impossible without the secret information in question.
- Central tool in study of cryptographic protocols

# Complexity Theory

- **A complexity class is the set of all of the computational problems which can be solved using a certain amount of a certain computational resource.**

- **The complexity class *P* is the set of decision problems that can be solved by a deterministic machine in polynomial time.**

- **The complexity class *NP* is the set of decision problems that can be solved by a *non*-deterministic machine in polynomial time.**

# Complexity Class NP

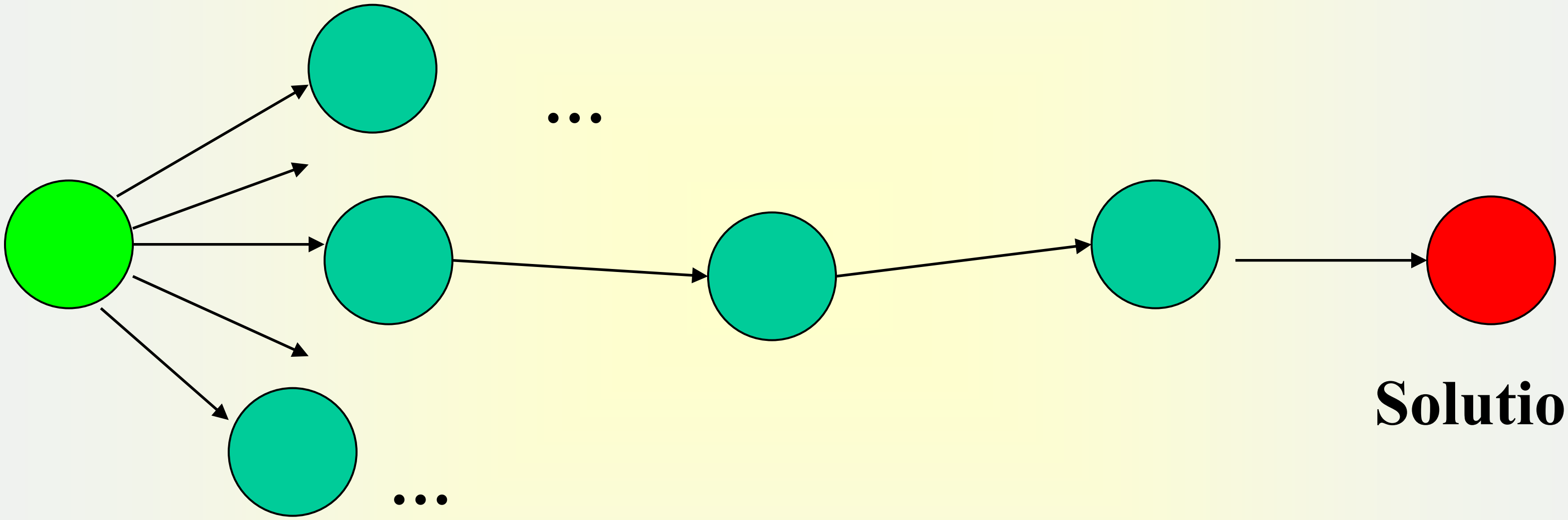
- NP ("Non-deterministic Polynomial time") is the set of decision problems solvable in polynomial time on a non-deterministic Turing machine.
  - It is the set of problems whose solutions can be "verified" by a deterministic Turing machine in polynomial time.
  - It takes exponential time to prove/find a solution, but it takes polynomial time to verify the correctness of a candidate solution.

Examples of NP problems

Boolean satisfiability problem,
Hamilton cycle for a large graph
Graph coloring
Quadratic nonresidue
Circuit satisfiability
Vertex-cover
Knapsack
Subset-sum
Integer Factorization Problem (IFP)
Discrete Logarithm Problem (DLP)

# Complexity Class NP

NP “search tree”



Solution!

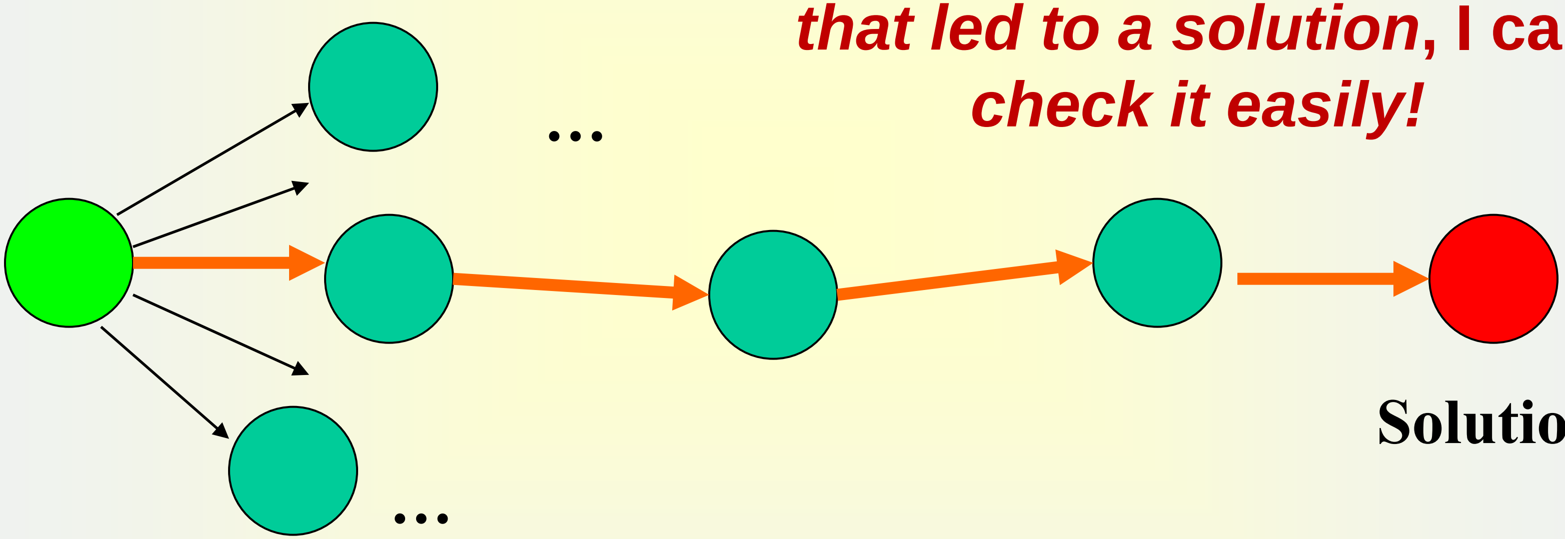
Mostly “dead ends”

**Hard to find a solution by just searching the tree!**

# Complexity Class NP

**NP “search tree”**

<span style="color:#c00000">**But if you *just tell me the path in the search tree that led to a solution*, I can *check it easily!***</span>



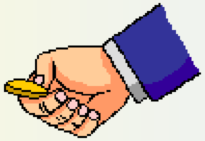
**Solution!**

**Mostly “dead ends”**

# New Ingredients for Interactive Proofs

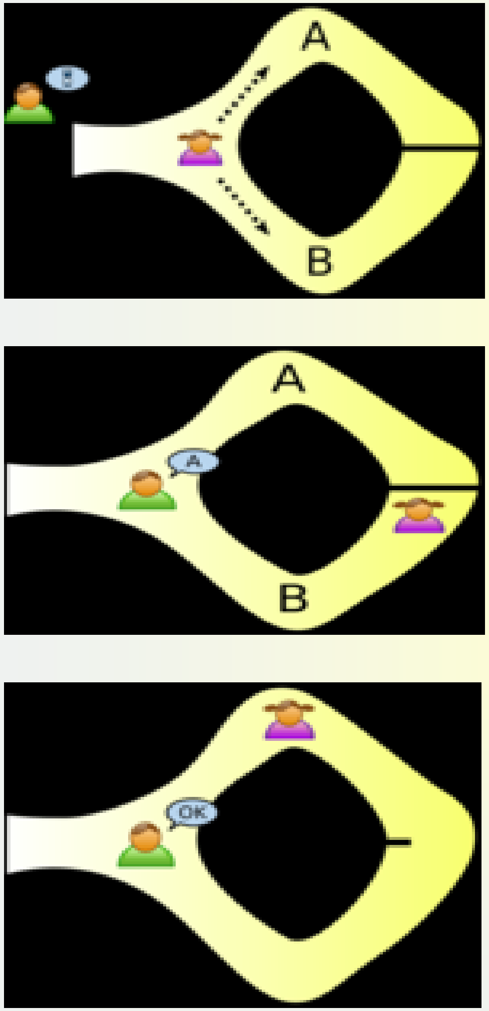
- **Classical NP proofs inherently non-zero-knowledge. Verifier gains ability to prove the assertion to others.**

- **Randomization: verifier can "toss coins"**
  **Allow verifier to error with small probability**



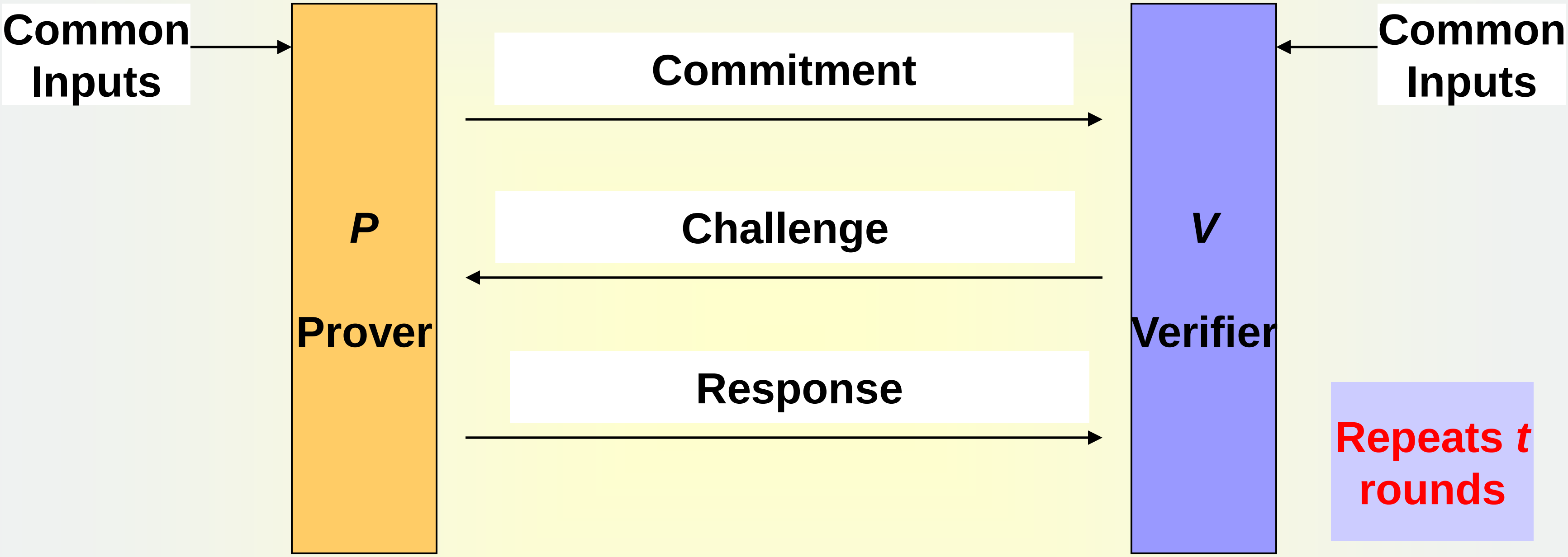
- **Interaction: replace static *proof* with dynamic, interactive *proof* with all-powerful *prover***
  - **Will "interact" with verifier and try to "convince" it that assertion is true.**
  - **Answer correctly for any question of the verifier (unpredictable questions)**

# Ali Baba’s Cave



- Alice wants to prove to Bob that she knows how to open the secret door between A and B, but will not reveal the secret itself.
- Procedure
  - Alice and Bob go to cave
  - Alice goes to A or B randomly (Bob cannot see)
  - Bob tells Alice to come from A or B
  - If Alice knows the secret, she can appear from the correct side of the cave every time
- Bob repeats as many times until he believe Alice knows the secret to open the secret door
- How about Trudy? Can he convince Bob without knowing the secret?

# Interactive Proof Protocol



- Prover and verifier share *common inputs* (functions or values)
- The protocol yields Accept if every Response is accepted by the Verifier
- Otherwise, the protocol yields Reject

# Requirements of Interactive Proofs

- ***Completeness***
  - **If the statement is true, the honest verifier will be convinced of this fact by an honest prover.**
  - **Prob[($P$,$V$)($x$) = Accept | $x \in L$] ≥ ε   where  ε ∈ (½,1]**

- ***Soundness***
  - **If the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.**
  - **Prob[(¬$P$,$V$)($x$) = Accept | $x \notin L$] ≤ δ   where  δ ∈ [0,½)**

# Zero-Knowledge Proofs

- Instances of interactive proofs with the following properties:
  - Completeness – true theorems are provable
  - Soundness – false theorems are not provable
  - Zero-Knowledge – No information about the prover's private input (secret) is revealed to the verifier

- GMR(Goldwasser, Micali, Rackoff)
  1. "The knowledge complexity of interactive-proof systems", Proc. of 17th ACM Sym. on Theory of Computation, pp.291-304, 1985
  2. "The knowledge complexity of interactive-proof systems", Siam J. on Computation, Vol. 18, pp.186-208, 1989 (revised version)

Fundamental Theorem [GMR]:
"Zero-knowledge proofs exist for all languages in NP"

# Flavors of Zero-Knowledge Proofs

- **Quality of ZK/Simulation:**
  - **Perfect (PZK)**
  - **Statistical (SZK)**
  - **Computational (ZK)**

- **Verifier strategies considered:**
  - **Honest-verifier zero knowledge (HVZK)**
  - **General zero knowledge (ZK)**

- **Soundness:**
  - **Proof systems: unbounded provers**
  - **Arguments: poly-time provers**

# Defining Zero-Knowledge

- **How to formalize “Verifier learns nothing”?**

**Simulation Paradigm (informally):**

- **Require: anything that can be computed in poly-time by interacting with prover can also be computed in poly-time without interacting with prover.**

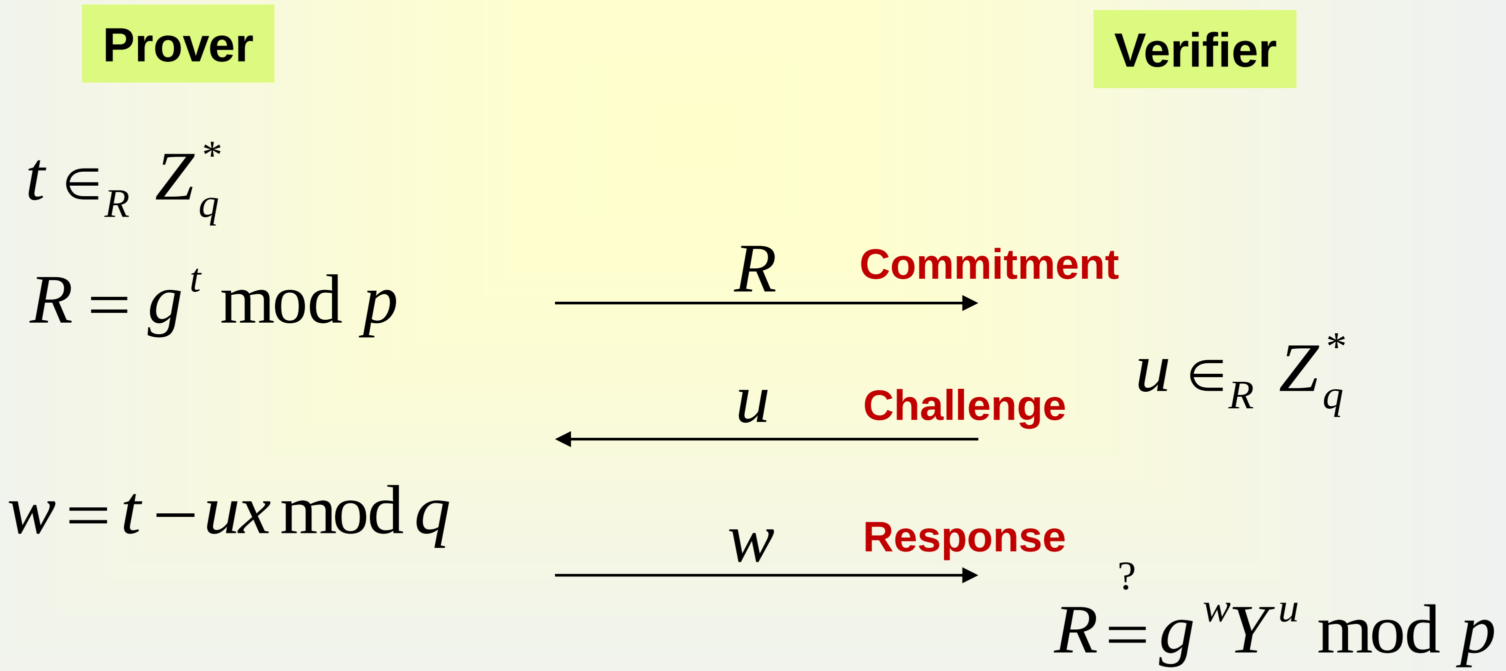
- **That is, for every poly-time verifier $V^*$, there exists a poly-time simulator $S$ s.t.**

  **[output of $S(x)$] ≈ [output of $V^*$ after interacting with $P$ on $x$].**

# Proof of Knowledge (of discrete logarithm)

- A prover tries to prove that he knows a discrete logarithm x

$$\boxed{x = \log_g Y \bmod p, \qquad (Y = g^x \bmod p)}$$



Prover

Verifier

$$t \in_R Z_q^*$$

$$R = g^t \bmod p$$

$R$ Commitment

$$u \in_R Z_q^*$$

$u$ Challenge

$$w = t - ux \bmod q$$

$w$ Response

$$R \overset{?}{=} g^w Y^u \bmod p$$

g

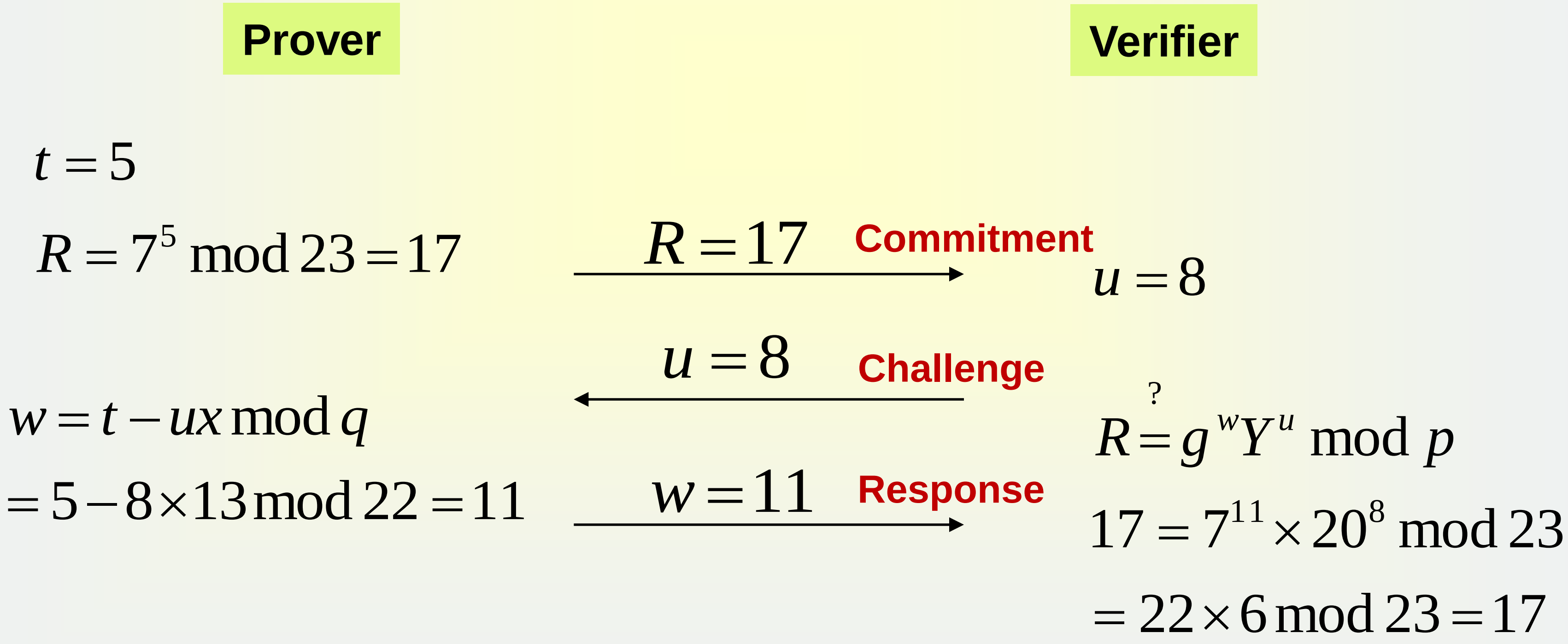
x

| ^ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 2 | 4 | 9 | 16 | 2 | 13 | 3 | 18 | 12 | 8 | 6 | 6 | 8 | 12 | 18 | 3 | 13 | 2 | 16 | 9 | 4 | 1 |
| 3 | 8 | 4 | 18 | 10 | 9 | 21 | 6 | 16 | 11 | 20 | 3 | 12 | 7 | 17 | 2 | 14 | 13 | 5 | 19 | 15 | 22 |
| 4 | 16 | 12 | 3 | 4 | 8 | 9 | 2 | 6 | 18 | 13 | 13 | 18 | 6 | 2 | 9 | 8 | 4 | 3 | 12 | 16 | 1 |
| 5 | 9 | 13 | 12 | 20 | 2 | 17 | 16 | 8 | 19 | 5 | 18 | 4 | 15 | 7 | 6 | 21 | 3 | 11 | 10 | 14 | 22 |
| 6 | 18 | 16 | 2 | 8 | 12 | 4 | 13 | 3 | 6 | 9 | 9 | 6 | 3 | 13 | 4 | 12 | 8 | 2 | 16 | 18 | 1 |
| 7 | 13 | 2 | 8 | 17 | 3 | 5 | 12 | 4 | 14 | 7 | 16 | 9 | 19 | 11 | 18 | 20 | 6 | 15 | 21 | 10 | 22 |
| 8 | 3 | 6 | 9 | 16 | 18 | 12 | 4 | 13 | 2 | 8 | 8 | 2 | 13 | 4 | 12 | 18 | 16 | 9 | 6 | 3 | 1 |
| 9 | 6 | 18 | 13 | 11 | 16 | 15 | 9 | 2 | 20 | 19 | 4 | 3 | 21 | 14 | 8 | 7 | 12 | 10 | 5 | 17 | 22 |
| 10 | 12 | 8 | 6 | 9 | 4 | 13 | 3 | 18 | 16 | 2 | 2 | 16 | 18 | 3 | 13 | 4 | 9 | 6 | 8 | 12 | 1 |
| 11 | 1 | 1 | 1 | 22 | 1 | 22 | 1 | 1 | 22 | 22 | 1 | 1 | 22 | 22 | 1 | 22 | 1 | 22 | 22 | 22 | 22 |
| 12 | 2 | 3 | 4 | 18 | 6 | 16 | 8 | 9 | 13 | 12 | 12 | 13 | 9 | 8 | 16 | 6 | 18 | 4 | 3 | 2 | 1 |
| 13 | 4 | 9 | 16 | 21 | 13 | 20 | 18 | 12 | 15 | 17 | 6 | 8 | 11 | 5 | 3 | 10 | 2 | 7 | 14 | 19 | 22 |
| 14 | 8 | 4 | 18 | 13 | 9 | 2 | 6 | 16 | 12 | 3 | 3 | 12 | 16 | 6 | 2 | 9 | 13 | 18 | 4 | 8 | 1 |
| 15 | 16 | 12 | 3 | 19 | 8 | 14 | 2 | 6 | 5 | 10 | 13 | 18 | 17 | 21 | 9 | 15 | 4 | 20 | 11 | 7 | 22 |
| 16 | 9 | 13 | 12 | 3 | 2 | 6 | 16 | 8 | 4 | 18 | 18 | 4 | 8 | 16 | 6 | 2 | 3 | 12 | 13 | 9 | 1 |
| 17 | 18 | 16 | 2 | 15 | 12 | 19 | 13 | 3 | 17 | 14 | 9 | 6 | 20 | 10 | 4 | 11 | 8 | 21 | 7 | 5 | 22 |
| 18 | 13 | 2 | 8 | 6 | 3 | 18 | 12 | 4 | 9 | 16 | 16 | 9 | 4 | 12 | 18 | 3 | 6 | 8 | 2 | 13 | 1 |
| 19 | 3 | 6 | 9 | 7 | 18 | 11 | 4 | 13 | 21 | 15 | 8 | 2 | 10 | 19 | 12 | 5 | 16 | 14 | 17 | 20 | 22 |
| 20 | 6 | 18 | 13 | 12 | 16 | 8 | 9 | 2 | 3 | 4 | 4 | 3 | 2 | 9 | 8 | 16 | 12 | 13 | 18 | 6 | 1 |
| 21 | 12 | 8 | 6 | 14 | 4 | 10 | 3 | 18 | 7 | 21 | 2 | 16 | 5 | 20 | 13 | 19 | 9 | 17 | 15 | 11 | 22 |
| 22 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

$g^x$ mod 23

# Proof of Knowledge (of discrete logarithm)

- Example: p=23, g=7, q=22
- Key generation x=13, y=20
- Prover proves that he knows x=13 corresponding to y=20 without revealing x



**Prover**

**Verifier**

$t = 5$

$R = 7^5 \bmod 23 = 17$

$\xrightarrow{R=17}$ **Commitment**

$u = 8$

$\xleftarrow{u=8}$ **Challenge**

$w = t - ux \bmod q$

$= 5 - 8 \times 13 \bmod 22 = 11$

$\xrightarrow{w=11}$ **Response**

$R \overset{?}{=} g^w Y^u \bmod p$

$17 = 7^{11} \times 20^8 \bmod 23$

$= 22 \times 6 \bmod 23 = 17$

# Proof of Equality of two discrete logarithms

- Prover tries to prove that two discrete logarithms are equal without revealing x

$$\boxed{\begin{array}{l} Y = g^x, Z = c^x \\ \log_g Y = \log_c Z \end{array}}$$

**Prover**

**Verifier**

$t \in_R Z_q^*$

$R_1 = g^t \bmod p$

$R_2 = c^t \bmod p$

$\xrightarrow{R_1, R_2}$ **Commitment**

$u \in_R Z_q^*$

$\xleftarrow{u}$ **Challenge**

$w = t - ux \bmod q$

$\xrightarrow{w}$ **Response**

$R_1 \overset{?}{=} g^w Y^u \bmod p$

$R_2 \overset{?}{=} c^w Z^u \bmod p$

# Proof of Equality of two discrete logarithms

$$Y = g^x, Z = c^x \qquad 7^5 = 17, 11^5 = 5$$

$$\log_g Y = \log_c Z \qquad \log_7 17 = \log_{11} 5$$

**Prover**

**Verifier**

$t = 3$

$R_1 = g^t \bmod p = 7^3 = 21$

$R_2 = c^t \bmod p = 11^3 = 20$

$R_1 = 21, R_2 = 20$ **Commitment** $\xrightarrow{\hspace{3em}}$

$u = 6$

$\xleftarrow{\hspace{3em}}$ $u = 6$ **Challenge**

$w = t - ux \bmod q$

$= 3 - 6 \times 5 \bmod 22 = 17$

$w = 17$ **Response** $\xrightarrow{\hspace{3em}}$

$R_1 \overset{?}{=} g^w Y^u \bmod p$

$= 7^{17} \times 17^6 \bmod 23 = 19 \times 12 = 21$

$R_2 \overset{?}{=} c^w Z^u \bmod p$

$= 11^{17} \times 5^6 = 14 \times 8 = 20$

# Proving the Correctness of ElGamal Decryption

- The prover tries to prove that his decryption is correct and the plaintext is m without revealing his private key x
- Prover's key $Y = g^x \bmod p$
- ElGamal Encryption: m→(U,V) $U = g^r \bmod p$

  $V = mY^r \bmod p$
- ElGamal Decryption $V / U^x \to m$

# Proving the Correctness of ElGamal Decryption

- Prover proves that the following two discrete logarithm is equal using the previous proof

$$\boxed{\begin{array}{l} Y = g^{x}, \dfrac{V}{m} = U^{x} \\ \log_{g} Y = \log_{U} \dfrac{V}{m} \end{array}}$$

# Non-Interactive Zero-Knowledge Proof

- **Non-interactive Zero-knowledge (NIZK) proofs using Fiat-Shamir Heuristic**

$$\boxed{x = \log_g Y \bmod p, \qquad (\mathrm{Y} = g^x \bmod p)}$$

**Prover**

$$t \in_R Z_q^*$$

$$R = g^t \bmod p$$

$$u = H(Y, R)$$

$$w = t - ux \bmod q$$

$$\xrightarrow{(R, w)}$$

**Verifier**

$$u = H(Y, R)$$

$$R \overset{?}{=} g^w Y^u \bmod p$$

# 6. Identification, Authentication

# Authentication

- ❖ Entity Authentication (Identification)
  - Over the communication network, one party, Alice, shows to another party, Bob, that she is the real Alice.
  - Authenticate an entity by presenting some identification information
  - Should be secure against various attacks
  - Through an interactive protocols using secret information

- ❖ Message Authentication
  - Show that a message was generated by an entity
  - Using digital signature or MAC

# Approach for Identification

- ❖ **Using Something Known**
  - **Password, PIN**

- ❖ **Using Something Possessed**
  - **IC card, Hardware token**

- ❖ **Using Something Inherent**
  - **Biometrics**

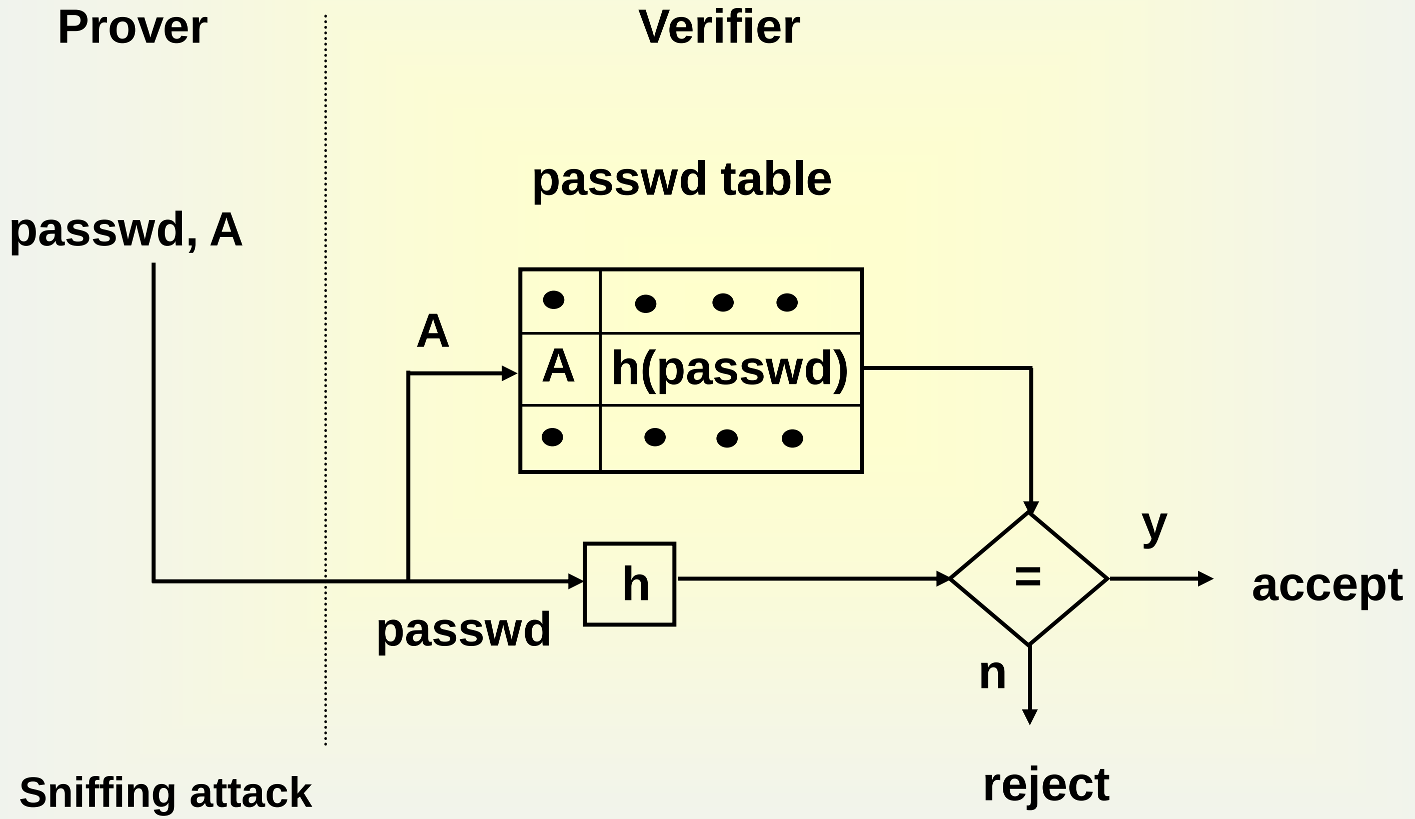
# Approach for Identification

| Method | Examples | Reliability | Security | Cost |
|---|---|---|---|---|
| *What you Remember (know)* | Password<br>Telephone #<br>Reg. # | M/L | M (theft)<br>L (impersonation) | Cheap |
| *What you have* | Registered Seal<br>Magnetic Card<br>IC Card | M | L (theft)<br>M (impersonation) | Reasonable |
| *What you are* | Bio-metric (Fingerprint, Eye, DNA, face, Voice, *etc*) | H | H (theft)<br>H (Impersonation) | Expensive |

# Approach for Identification

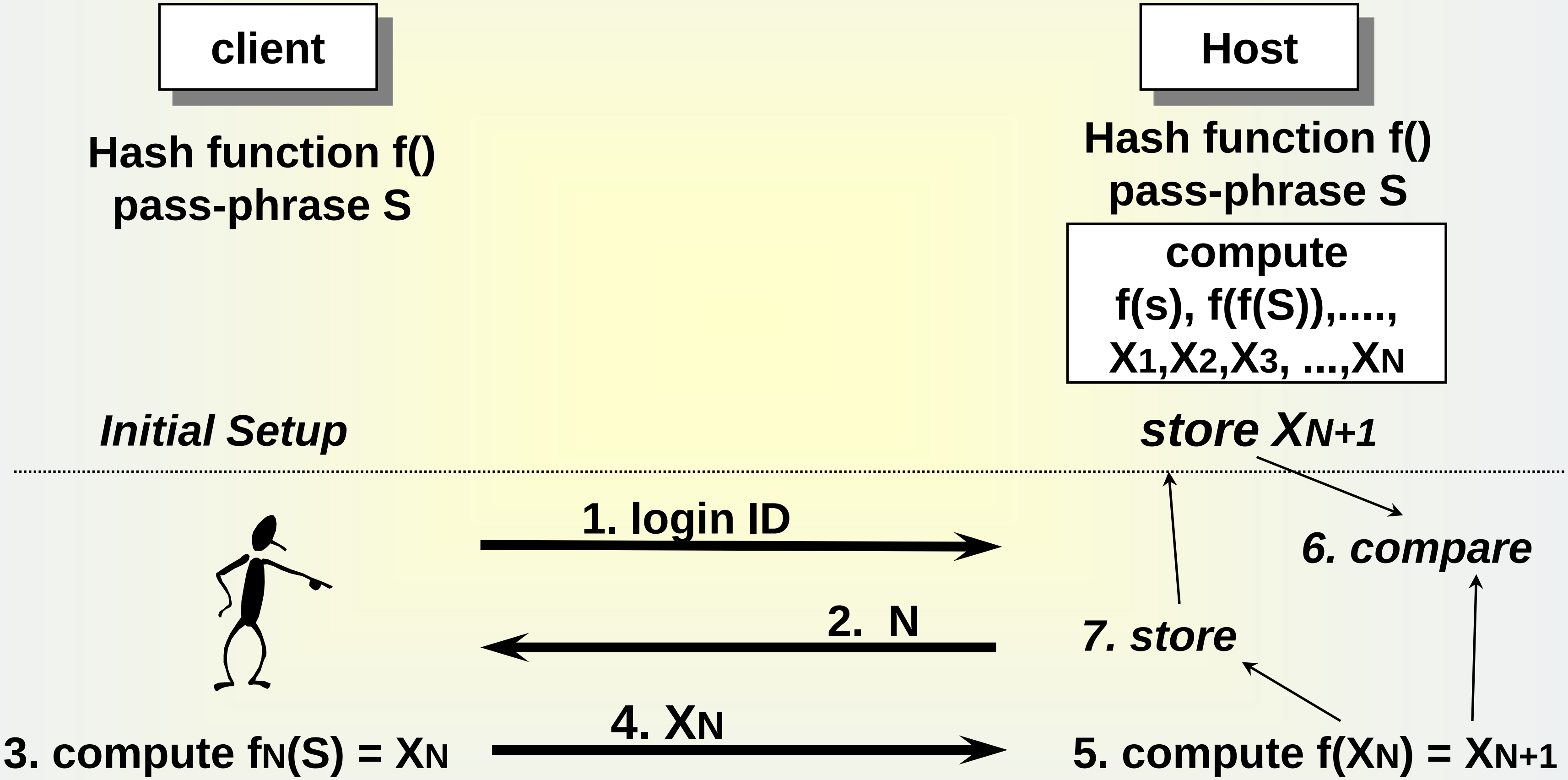
- **Password-based scheme (weak authentication)**
  - **crypt *passwd* under UNIX**
  - **one-time password**
- **Challenge-Response scheme (strong authentication)**
  - **Symmetric cryptosystem**
  - **MAC (keyed-hash) function**
  - **Asymmetric cryptosystem**
- **Using Cryptographic Protocols**
  - **Fiat-Shamir identification protocol**
  - **Schnorr identification protocol, *etc***

# Identification by Password



Sniffing attack
Replay attack - Static password

# S/Key (One-Time Password System)



client

Hash function f()
pass-phrase S

Host

Hash function f()
pass-phrase S

compute
f(s), f(f(S)),....,
$X_1$,$X_2$,$X_3$, ...,$X_N$

*Initial Setup*

*store $X_{N+1}$*

1. login ID

2. N

3. compute $f_N(S) = X_N$

4. $X_N$

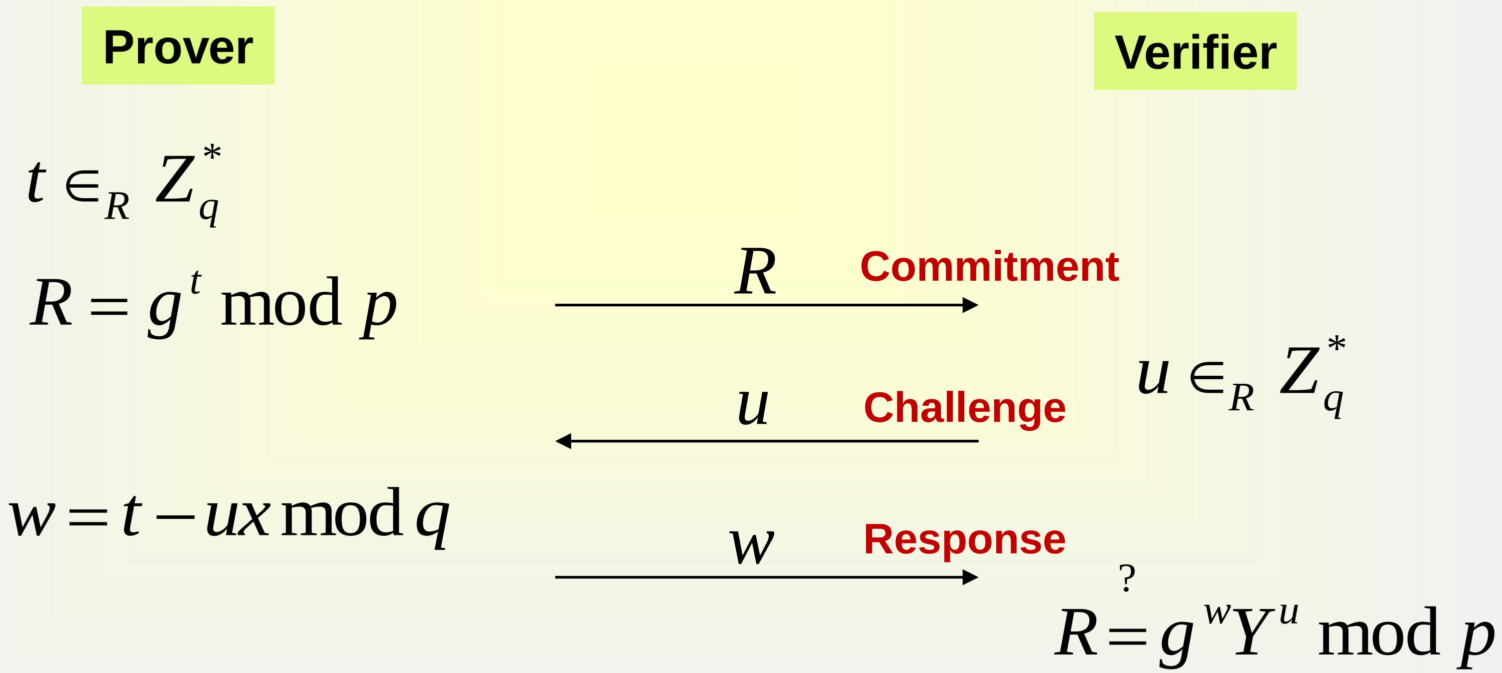
5. compute $f(X_N) = X_{N+1}$

*6. compare*

*7. store*

# Schnorr Identification

$$\boxed{x = \log_g Y \bmod p, \qquad (Y = g^x \bmod p)}$$



**Prover**

**Verifier**

$t \in_R Z_q^*$

$R = g^t \bmod p$

$\xrightarrow{\quad R \quad}$ Commitment

$u \in_R Z_q^*$

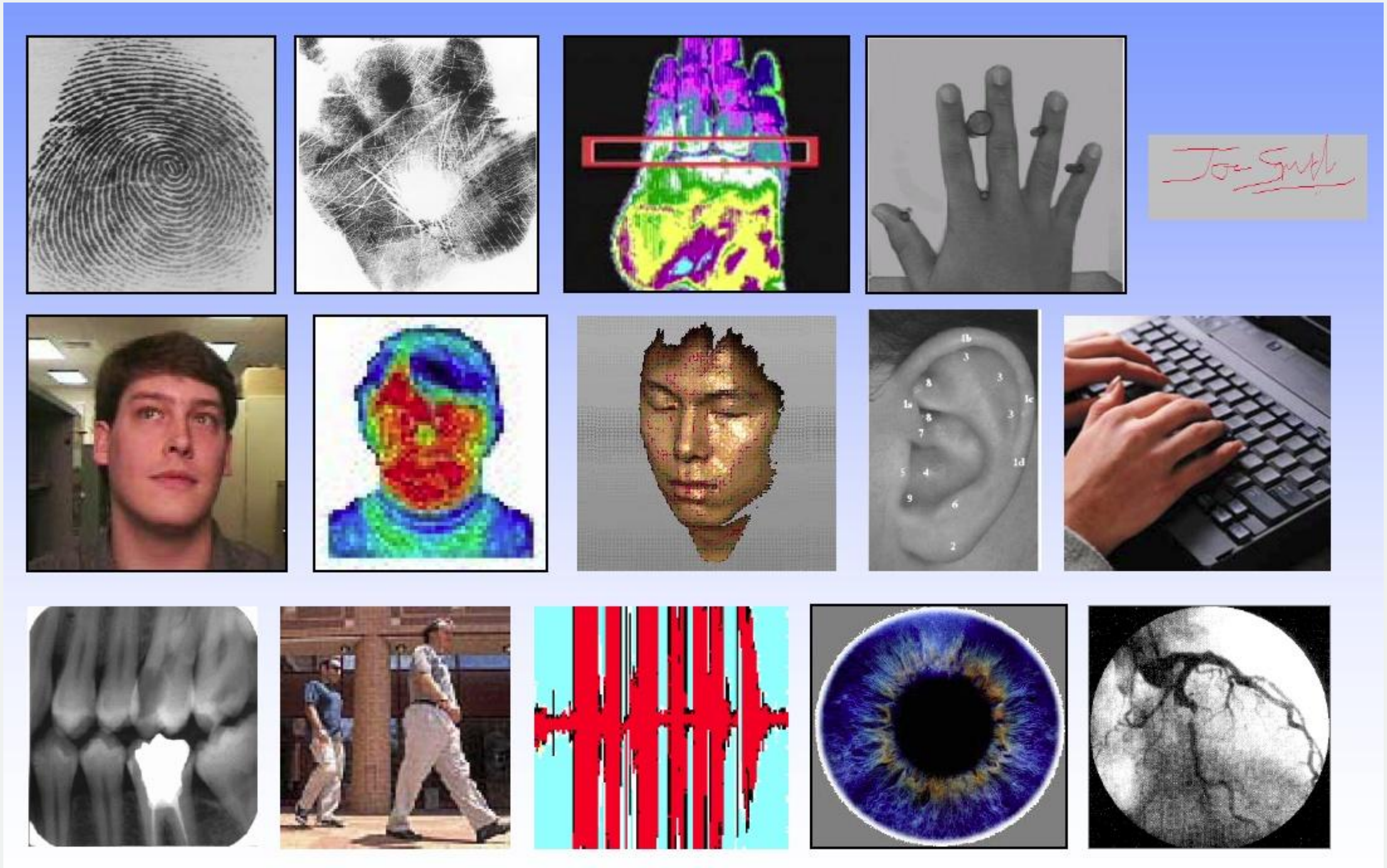
$\xleftarrow{\quad u \quad}$ Challenge

$w = t - ux \bmod q$

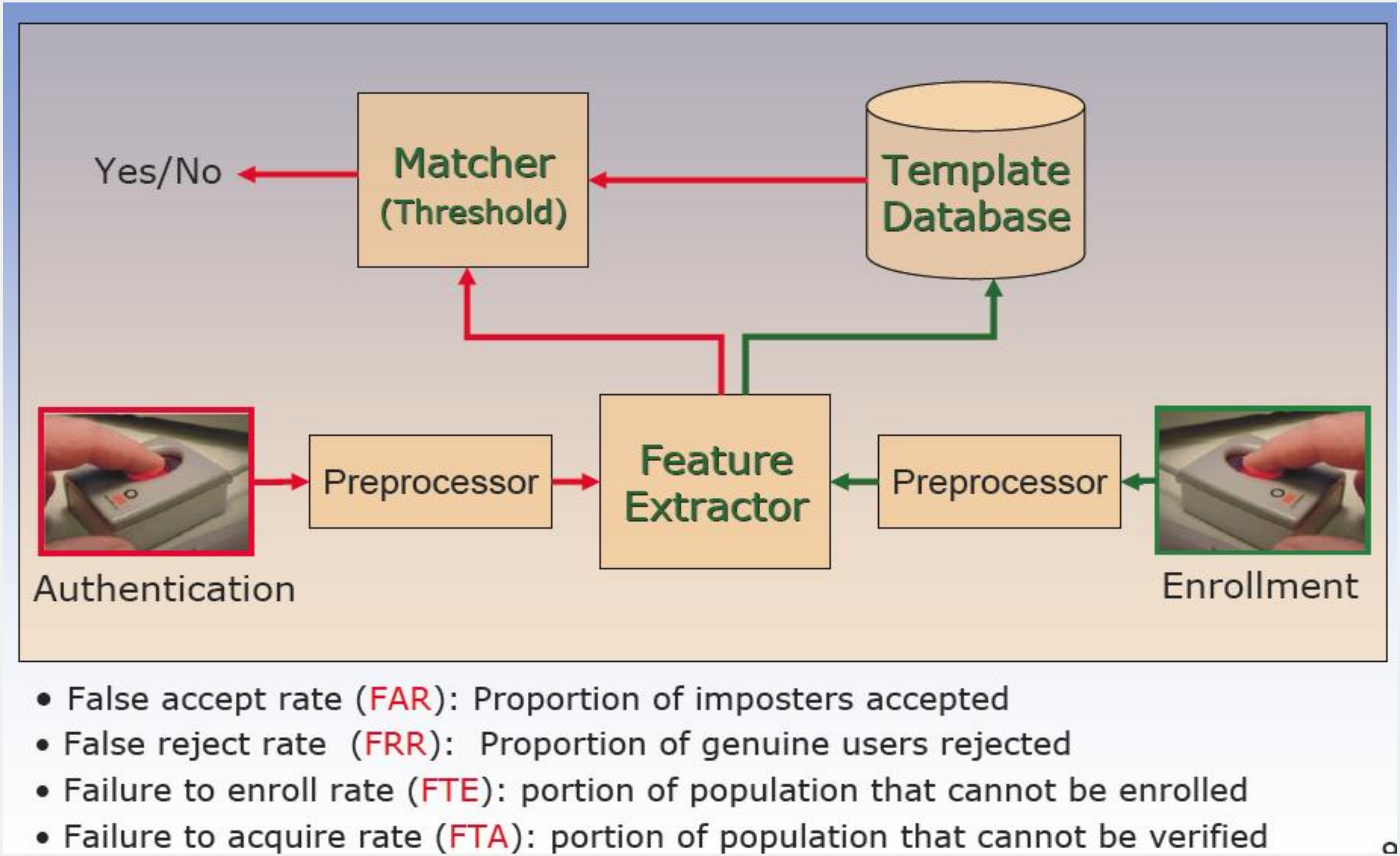
$\xrightarrow{\quad w \quad}$ Response

$R \overset{?}{=} g^w Y^u \bmod p$

# Identification using Biometric Trails

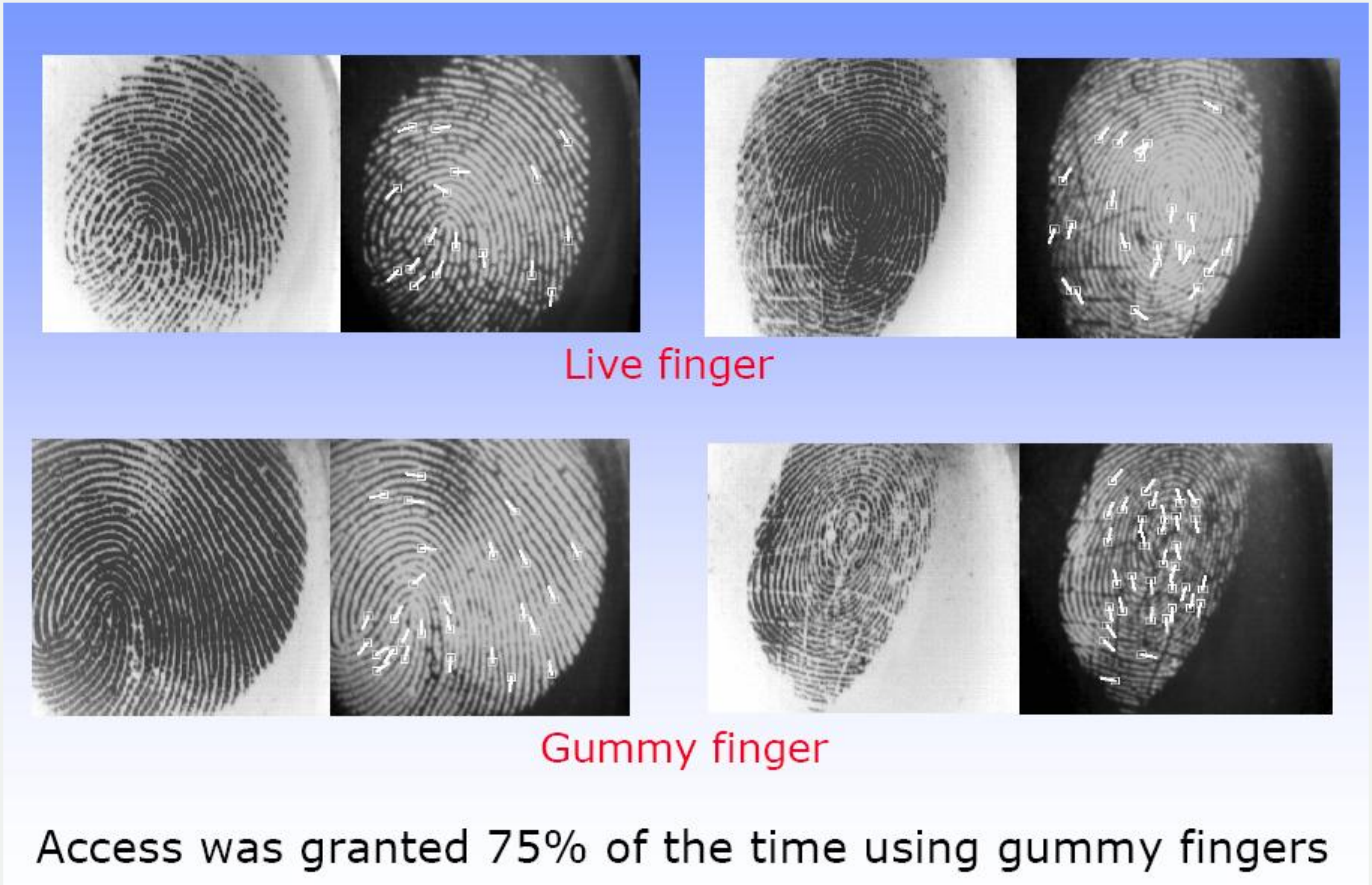


# Biometric Recognition System



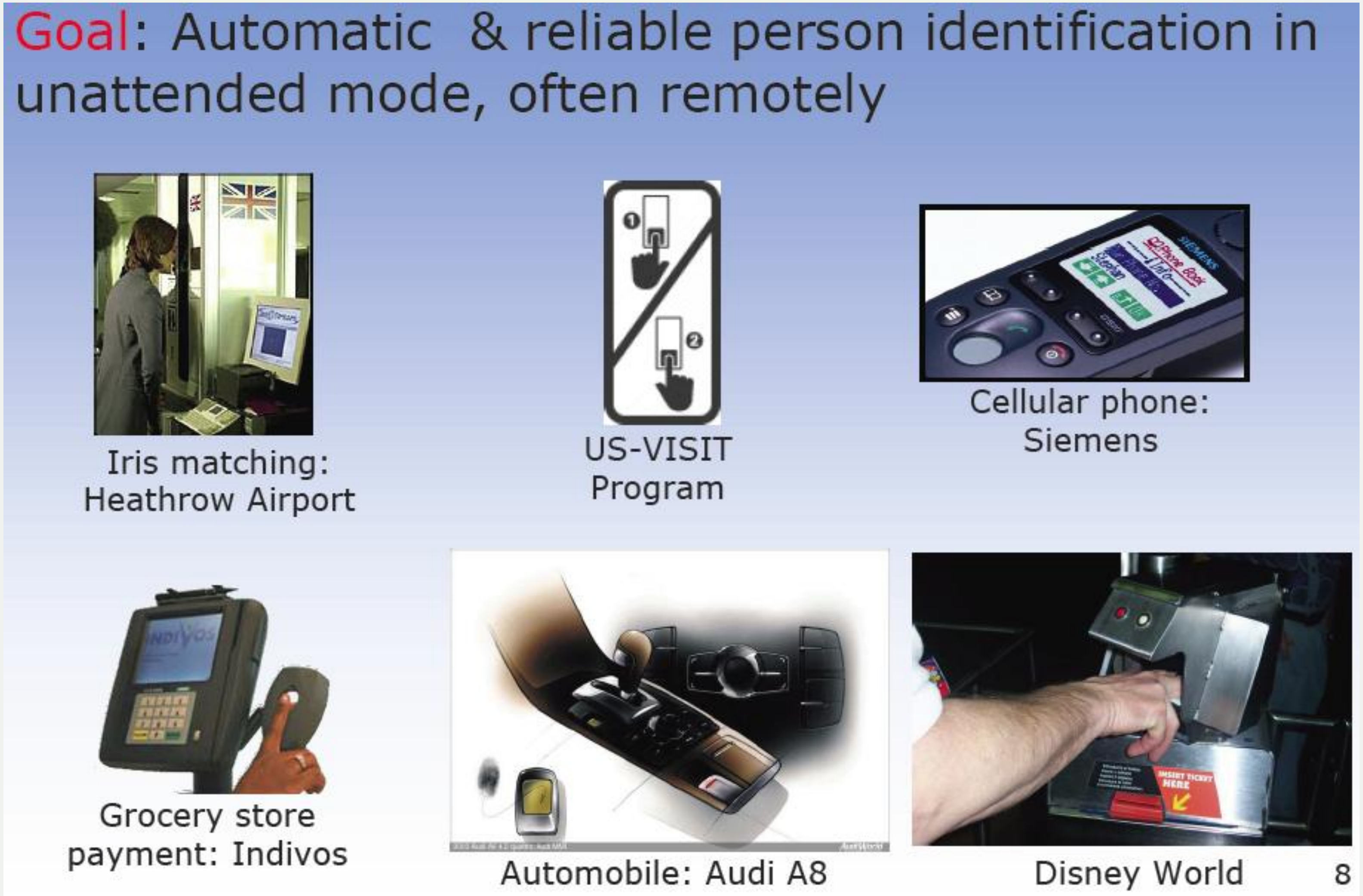
- False accept rate (FAR): Proportion of imposters accepted
- False reject rate (FRR): Proportion of genuine users rejected
- Failure to enroll rate (FTE): portion of population that cannot be enrolled
- Failure to acquire rate (FTA): portion of population that cannot be verified

# Fake Fingerprint



Access was granted 75% of the time using gummy fingers

# Applications



Goal: Automatic & reliable person identification in unattended mode, often remotely

Iris matching: Heathrow Airport

US-VISIT Program

Cellular phone: Siemens

Grocery store payment: Indivos

Automobile: Audi A8

Disney World

8