# Announcements

Homework 1

- Due ***today 11:59pm***
- Submit through **GradeScope** in **PDF**

Midterm exam

- **Next Thursday**, in class (2-3:20pm)

# Lecture 8

## Public Key Cryptography II: Signatures (cont'd) + Identification

[lecture slides are adapted from previous slides by Prof. Gene Tsudik]

# Digital Signatures

- Integrity
- Authentication
- Non-Repudiation
- Time-Stamping
- Causality
- Authorization




# RSA Signature Scheme

Use the fact that, in RSA, encryption reverses “decryption”

$$\text{Let } n = pq \ \text{ where } \ p \neq q \ \text{ are two (large) primes}$$

$$e \in Z^{*}_{\Phi(n)} \ \text{ and } \ e = d^{-1} \bmod \Phi(n) \text{ and } \ ed \equiv 1 \ \bmod \Phi(n)$$

$$\Phi(n) = (p-1)(q-1)$$

$$Secrets : p, q, d$$

$$Publics : n, e$$

$$\text{Signing} : message = m$$

$$Sign(m) : y = m^{d} \ mod \ n$$

$$\text{Verification} : signature = y$$

$$Verify(y, m) : (m = y^{e})???$$

# RSA Signature Scheme (contd)

- The Good:
  - Verification can be cheap (like RSA encryption)
  - Mechanically same as RSA decryption function
  - Security based on RSA encryption
  - Signing is harder but #verify-s > 1 …
  - Deterministic
- The Bad:
  - RSA is malleable: signatures can be "massaged"
    - $m_1^d * m_2^d = (m_1 * m_2)^d$
  - Phony "random" signatures
    - compute $Y=RSA(e,X)=X^e \bmod n$
    - X is a signature of Y because $Y^d=X \bmod n$

| Plaintext | SIG |
|---|---|
| $X^e$ ← | X |

- The Ugly:
  - Signing requires integrity!
  - How to sign multiple blocks when m > n?
  - Deterministic – needs additional randomization!

# El Gamal Signature Scheme

$p -$ *large prime*
$b -$ *base, generator*
$x -$ *private exponent*
$y -$ *public residue*; $y \equiv b^x \bmod p$
$P = Z_p^*$
$A = Z_p^* \times Z_p^*$
*publics* : $p, b, y$
*secrets* : $x$

*Signing* :
1. *generate random* $r \in Z_{p-1}$
2. *compute* : $k = b^r \bmod p$
3. *compute* : $c = (m - xk)r^{-1} \bmod p - 1$
4. *signature* $= \{k, c\}$

*Verifying* :
$y^k k^c \bmod p = b^m \bmod p \quad ???$

*notice that* :
$y^k k^c = b^{xb^r} (b^r)^{(m/r - xk/r)} = b^{xb^r + m - xb^r} = b^m$

# El Gamal PK Cryptosystem

$p$ – *large prime*
$b$ – *base, primitive element, generator*
$x$ – *private exponent*
$y$ – *public residue*; $y \equiv b^x \bmod p$
$P = Z_p^*$
$C = Z_p^* \times Z_p^*$
*publics* : $p, b, y$
*secrets* : $x$

*Encryption* :
1. *generate random* $r \in Z^*_{p-1}$
2. *compute* : $k = b^r \bmod p$
3. *compute* : $c = my^r \bmod p = mb^{xr} \bmod p$
4. *ciphertext* $= \{k, c\}$

*Decryption* :
1. *compute* $k^x \bmod p$
2. *compute* $(k^x)^{-1} \bmod p$
3. $m' = (k^x)^{-1} c = b^{-rx} m b^{xr} \bmod p = m$

# El Gamal Signature Scheme

$p$ – *large prime*
$b$ – *base, generator*
$x$ – *private exponent*
$y$ – *public residue*; $y \equiv b^x \bmod p$
$P = Z_p^*$
$A = Z_p^* \times Z_p^*$
*publics* : $p, b, y$
*secrets* : $x$

*Signing* :
1. *generate random* $r \in Z^*_{p-1}$
2. *compute* : $k = b^r \bmod p$
3. *compute* : $c = (m - xk) r^{-1} \bmod p - 1$
4. *signature* $= \{k, c\}$

*Verifying* :
$y^k k^c \bmod p = b^m \bmod p \quad ???$

*notice that* :
$y^k k^c = b^{xb^r} (b^r)^{(m/r - xk/r)} = b^{xb^r + m - xb^r} = b^m$

# El Gamal Signature Scheme (cont'd)

The good:

- Signing is cheap(er)
- Designed as a signature function
- Non-deterministic (randomized)

The bad:

- Need GOOD source of random numbers
- Randomizers cannot be revealed (trace)
- Randomizers cannot be reused

# The Digital Signature Standard (DSS)

- Why DSS?
- RSA issues: patents, malleability, etc.
- A variant of El Gamal, but better performance
- Originally for |p|=512 bits, now up to 1024
- Optimized for signature size (320- vs. 1024-bit)
- Signing - 1 exp, 1 inv, verification - 2 exps, 1 inv
- No attacks thus far

# DSS (contd)

$p - large\ prime$
$b - base, generator$
$x - private\ exponent$
$y - public\ residue;\ y \equiv b^x \bmod p$
$P = Z_p^{\ *}, A = Z_p^{\ *} \times Z_p^{\ *}$
$publics: p, b, y \quad secrets: x$

$Signing:$

1. $generate\ random \quad r \in Z^*_{\ p-1}$
2. $compute: k = b^r \bmod p$
3. $compute: c = (m - xk)r^{-1} \bmod p - 1$
4. $signature = \{k, c\}$

$Verifying:$

$y^k k^c \bmod p = b^m \bmod p \quad ???$

$p - 512 - bit\ prime$
$q - 160 - bit\ prime, (p-1)\%q = 0$
$b - base, b^q \equiv 1 \bmod p \quad (b = \delta^{(p-1)/q})$
$x - private\ exponent$
$y - public\ residue;\ y \equiv b^x \bmod p$
$P = Z_p^*, A = Z_q \times Z_q$
$publics: p, q, b, y \quad secrets: x$

$Signing:$

1. $generate\ random \quad r \in Z^*_{q-1}$
2. $compute: k = (b^r \bmod p) \bmod q$
3. $compute: c = (m + xk)r^{-1} \bmod q$
4. $signature = \{k, c\}$

$Verifying:$

$(b^{mc^{-1}} k^{kc^{-1}} \bmod p) \bmod q = b^k \bmod p \quad ???$

$notice\ that:$

$$b^{mc^{-1}} y^{kc^{-1}} = b^{mr/(m+xb^r)} (b^x)^{(b^r r/(m+xb^r)}$$
$$= b^{(mr+xb^r r)/(m+xb^r)} = b^r$$

# Other interesting constructions around our topic…

# Interactive (Public) Key Exchange: Diffie-Hellman

Choose random v

$$y_a = a^v \bmod p$$

$$y_b = a^w \bmod p$$

Choose random w, Compute $K_{ba} = (y_a)^w \bmod p$

Compute $K_{ab} = (y_b)^v \bmod p$

Secure communication with $K_{ab}$

Eve is passive …

Use symmetric crypto to exchange keys?

# Merkle's Puzzles (1974)

$for\ 0 < i < 2^n = N$

$Pick\ random\ values\ X_i, Y_i, |X_i| \geq 2 * |Y_i|,\ where\ |Y_i| = n$

$Pick\ random\ index_i, |index_i| = n$

$Form\ Puzzle\ \ P_i = E(Y_i, \{index_i, Xi, S\})$

$where\ S\ is\ a\ fixed\ string, e.g., "Alice\ to\ Bob"$

$\{P_i \mid 0 < i < 2^n\}$ →

$Pick\ random\ j, 0 < j < 2^n$

$Select\ P_j$

$Break\ Y_j\ by\ brute\ force$

$Obtain\ \{index_j, X_j, S\}$

← $index_j$

$Lookup\ index_j$

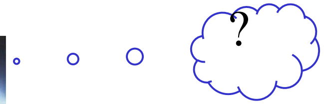
$Obtain\ \ X_j$

Encrypted communication with $X_j$ ↔

***Alice's effort: O(2^n)***

***Bob's effort: O(|Y_j|) = O(2^n)***

$$\textbf{\textit{Eve's effort:}}\quad O(2^n * |Y_i|) = O((2^n)^2) = O(|X_i|)$$





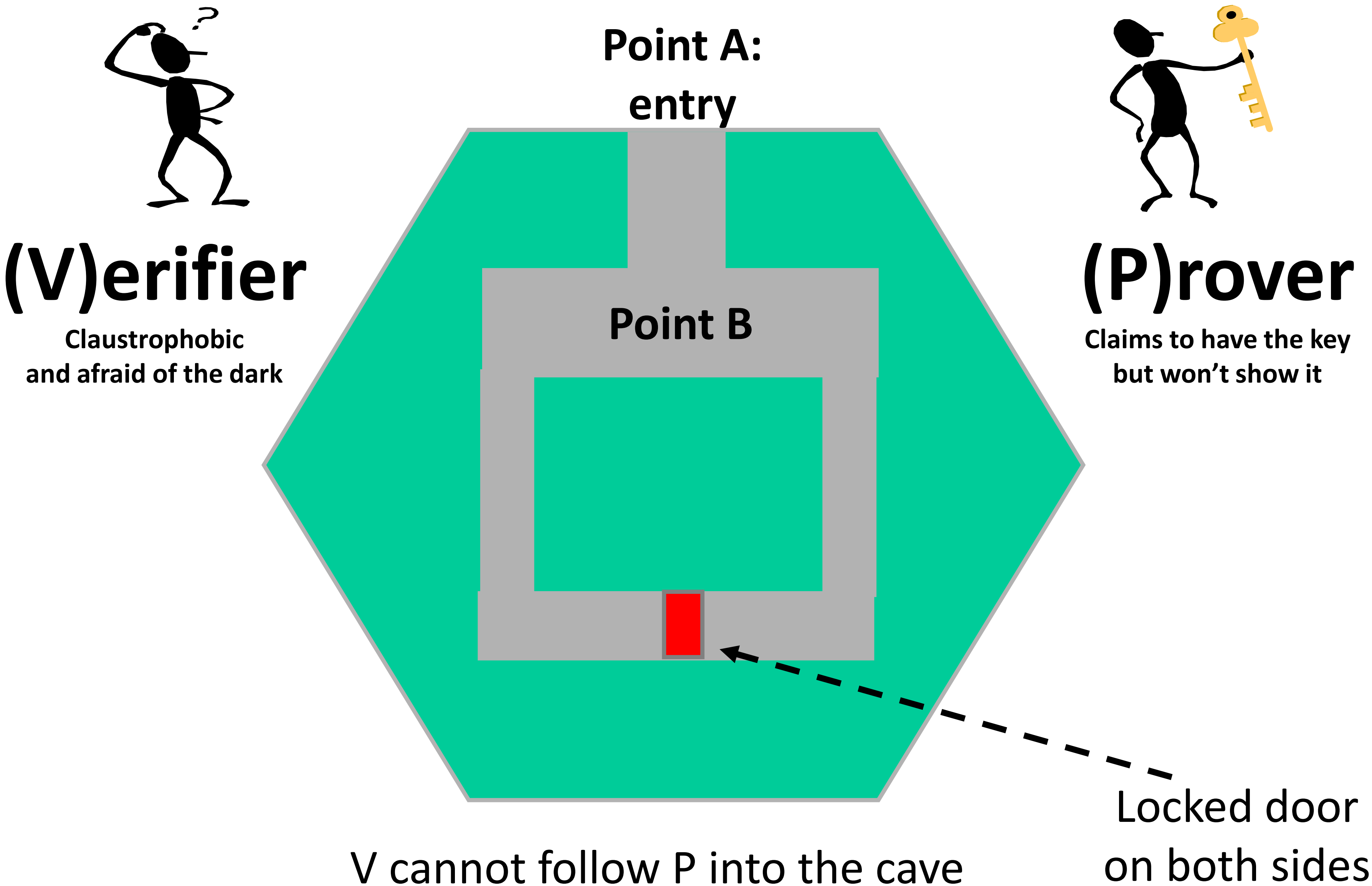
Is security computational or information theoretic?

# Other use of public key crypto (except encryption & signature)?

# Identification/Authentication

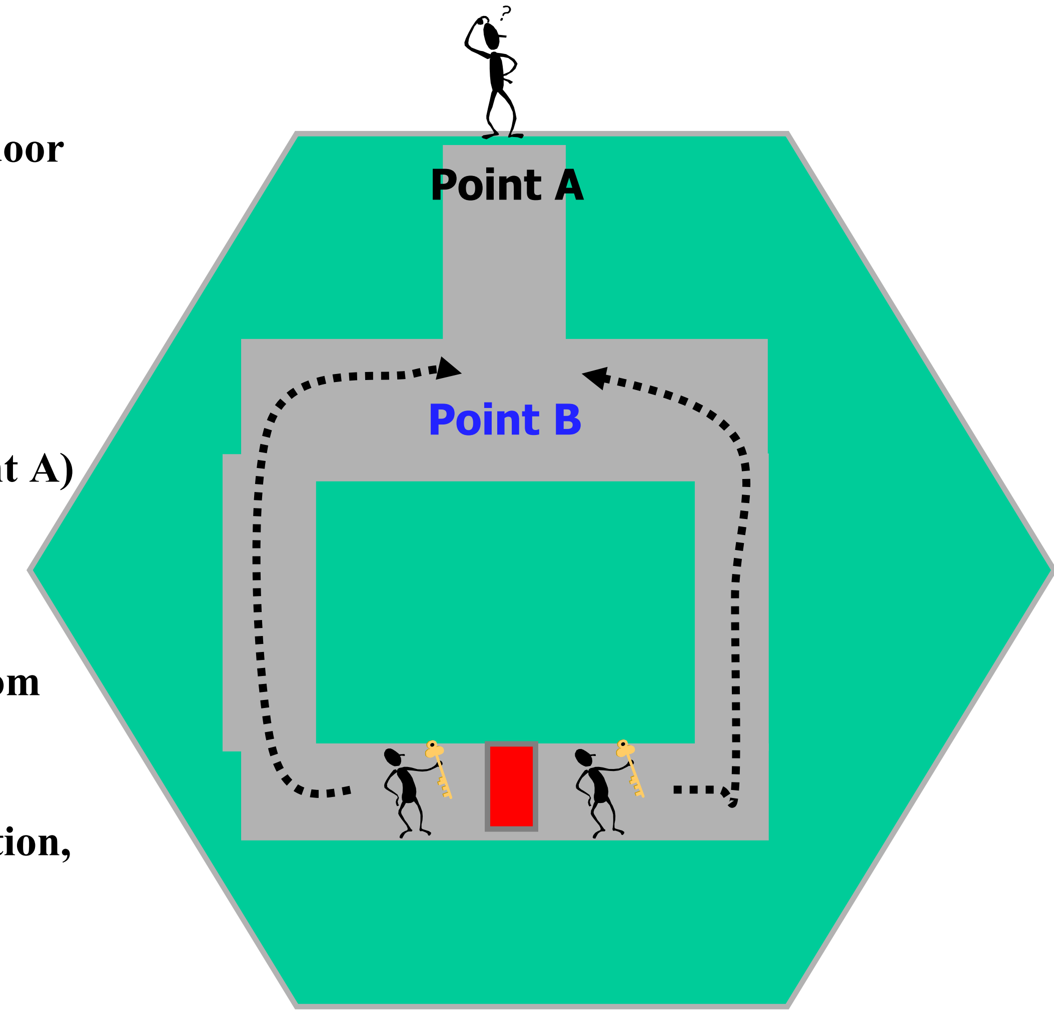
- Identification/authentication is an interactive protocol whereby one party: “prover” (who claims to be, say, Alice) convinces the other party: “verifier” (Bob) that she is indeed Alice
- Identification/authentication can be accomplished with public key digital signatures
  - However, signatures reveal information about private key
  - Also, signatures are “transferrable”, e.g., anyone who has Alice’s signature can use it to prove that he/she is Alice
- Can we provide identification/authentication without revealing any info about the secret?
  - *Zero-knowledge proof*: prove ownership of a secret without revealing any info about the secret

# The Cave Analogy of Zero-Knowledge




V cannot follow P into the cave

# The Cave Analogy of Zero-Knowledge




The Protocol:

1) V asks someone he trusts to check that the door is locked on both sides.

2) P goes into the maze past point B (heading either right or left)

3) V looks into the cave (while standing at point A)

4) V randomly picks right or left

5) V shouts (very loudly!) for P to come out from the picked direction

6) If P doesn't come out from the picked direction, V knows that P is a liar and protocol terminates

**REPEAT steps (2)-(6)**
**k TIMES**

# Fiat-Shamir Identification Scheme

- In Fiat-Shamir, prover has an RSA-like modulus **n = pq** where p and q are large primes and factorization of n is secret

- Primes themselves are not used in the protocol
  - Unlike RSA, a trusted center can generate a global **n**, used by everyone, as long as nobody knows its factorization. Trusted center can then “forget” the factorization after computing **n**

# Fiat-Shamir Identification Scheme

- Secret Key: Prover (P) chooses a random value

  $1 < S < n$ (to serve as the key)

  such that $\gcd(S,n) = 1$
- Public Key: P computes $I=S^2 \bmod n$, publishes $(I,n)$ as his public key.
  - Assumption: Finding square roots mod n is at least as hard as factoring n

- Purpose of the protocol: P has to convince verifier (V) that he knows the secret S corresponding to the public key (I,n),
  - i.e., to prove that he knows a square root of I mod n, without revealing S or any portion thereof

# Fiat-Shamir

**Prover (Alice)**

***n***, ***I***, ***S***

*pick random **R**;*

*set **x**=**R**$^2$ mod **n***

**Verifier (Bob)**

***n***

***I***, ***x*** →

← *query* = 0 / 1

***R*** → Check that: ***R***$^2$ = ***x*** *mod* ***n***

***R * S*** *mod* ***n*** → (***RS***)$^2$ = ***xI*** *mod* ***n***

# Fiat-Shamir Identification Scheme

V wants to authenticate identity of P, who claims to have a public key I. Thus, V asks P to convince him that P knows the secret key S corresponding to I .

1. P chooses at random $1 < R < n$ and computes: $X = R^2 \bmod n$

2. P sends X to V

3. V randomly requests from P one of two things (0 or 1):
   (a) R
   or
   (b) RS mod n

4. P sends requested information

# Fiat-Shamir ZK Identification Scheme

5. V checks the correct answer:

   a) $R^2$ ?= X (mod n)

      or

   b) $(R*S)^2$ ?= X*I (mod n)

6. If verification fails, V concludes that P does not know S

7. Protocol is repeated t (usually 20, 30, or log n) times, and, if each one succeeds, V concludes that P is the claimed party.

# What if Prover knows the challenge ahead of time: Case 0

***n***, ***I*** *(doesn't know S)*

***n***

*pick random* ***R****;*
*set* $\boldsymbol{x}=\boldsymbol{R}^2$ *mod* ***n***

***I***, ***x***

*query* = 0

***R***

Check that:
$\boldsymbol{R}^2 = \boldsymbol{x} \text{ mod } \boldsymbol{n}$

# What if Prover knows the challenge ahead of time: Case 1

***n***, ***I*** *(doesn't know S)*

*pick random* ***R****;*

*set* $\boldsymbol{x}=\boldsymbol{R}^2*\boldsymbol{I}$ *mod* ***n***

***n***

$\boldsymbol{I}, \boldsymbol{x}=\boldsymbol{R}^2*\boldsymbol{I}$

*query* = 1

Check that:

$(\boldsymbol{R}*\boldsymbol{I})^2 = \boldsymbol{x}*\boldsymbol{I} \bmod \boldsymbol{n}$

$\boldsymbol{R}*\boldsymbol{I} \bmod \boldsymbol{n}$

***(Instead of: R*S*** *mod* ***n)***

# Fiat-Shamir Identification Scheme

CLAIM: Protocol does not reveal ANY information about S,

or

The Fiat-Shamir protocol is **ZERO-KNOWLEDGE**

**Proof**: We show that no information on S is revealed:

- Clearly, when P sends X or R, it does not reveal any information about S
- When P sends *RS mod n*:
  - **RS mod n** is random, since R is random and gcd(S, n) = 1.
  - If adversary can compute any information about S from

I, n, X and RS mod n

it can also compute the same information on S from I and n, since it can choose a random $T = R'S \bmod n$ and compute:

$$X' = T^2 I^{-1} = (R')^2 S^2 I^{-1} = (R')^2$$

# Security

Clearly, if P knows S, then V is convinced of P's identity

If P does not know S, it can either:

1. know R, but not RS mod n. Since P is choosing R, it cannot multiply it by the unknown value S

   or

2. choose RS mod n, and thus can answer the second question: RS mod n. But, in this case, P cannot answer the first question R, since to do so, needs to divide by unknown S

# Security

- In any case, adversary cannot answer both questions, since otherwise he can compute S as the ratio between the two answers.

- But, we assumed that computing S is hard, equivalent to factoring n.

- Since P does not know in advance (when choosing R or RS mod n) which question that V will ask, he cannot foresee the required choice. He can succeed in guessing V's question with probability 1/2 for each question.

- The probability that V fails to catch P in all runs is thus: $2^{-t}$
  - e.g., 1 in 1,000,000,000 for t=20