CS306: Introduction to IT Security

Fall 2020

Lectur ography

Instructor: **Nikos**

October 20, 2020

STEVENS
INSTITUTE *of* TECHNOLOGY
THE INNOVATION UNIVERSITY®

CS306: Announcements

- HW2 did not come – too much in view of next week's midterm exam
- Road ahead
  - ~~no lecture on October 13 (next week, classes will run on Monday schedule)~~
  - regular lecture on Oc
  - midterm exam on
    - online exam, quiz format
    - accommodations to be provided as nee
    - covers all materials discussed so far: lectures 1-7, labs 1-7, HW1
    - Lab 7 will offer a general revision on most important topics
    - exact list of topics to be provided tomorrow

CS306: Tentative Syllabus

Week	Date	Topics	Reading	Assignment
1	Sep 1	Introduction	Lecture 1	-
2	Sep 8	Symmetric-key encryption	Lecture 2	Lab 1
3	Sep 15		Lecture 3	Lab 2, HW 1
4	Sep 22		Lecture 4	Lab 3, HW 1
5	Sep 29	Ciphers in pra	ecture 5	Lab 4
6	Oct 6	MACs & hashing	Lecture 6	Lab 5
-	Oct 13	No class (Monday schedule)		Lab 6
7	Oct 20	Public-key cryptography	Lecture 7	Lab 7, HW2

CS306: Tentative Syllabus (continued)

Week	Date	Topics	Reading	Assignment
8	Oct 27	**Midterm**	All materials covered	
9	Nov 3			
10	Nov 10			
11	Nov 17	Cloud secu		
12	Nov 24	AC/Authenticatio		
13	Dec 1	Economics		
14	Dec 8	Legal & ethical issues		
15	Dec 10 (or later)	**Final** (closed "books")	All materials covered*	

* w/ focus on what covered after midterm

Two weeks ago

- Message authentication
  - MACs
  - Replay attacks
  - Constructions
- Cryptographic hashi
  - Hash functions
  - Constructions
- Demo
  - Hash functions in practice

Today

- Revision on message authentication & cryptographic hashing
  - Practical applications
    - authenticated encryption, hash functions security strength, HMAC
- Public-key (PK) cryp
  - Motivation, PK Infrastructure, PK encryp atures
  - Discrete log problem, DH key agreemen tion
- Demo
  - The length-extension attack…

ric-key encryption

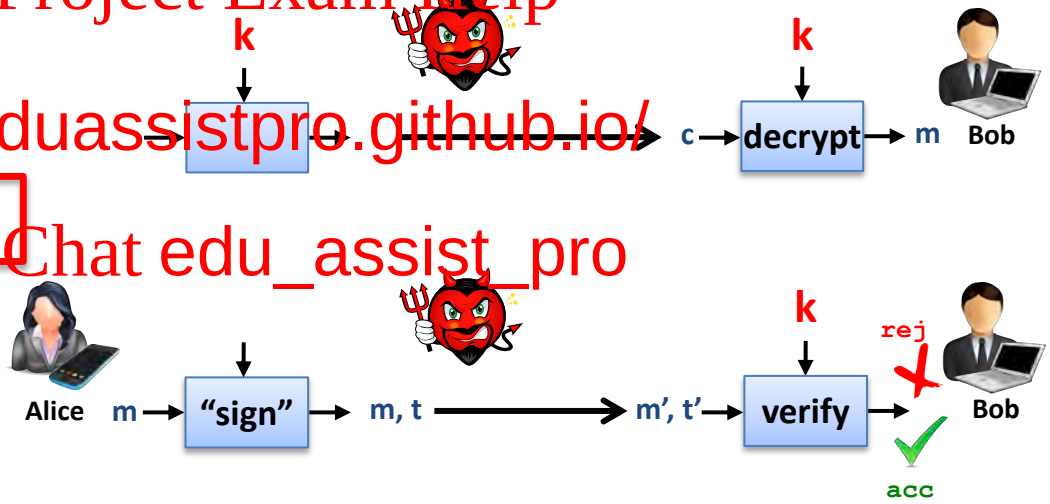
ignatures

Recall: Principles of modern cryptography

(A) security definitions, **(B) precise assumptions**, (C) formal proofs

For **symmetric-key** message encryption/authentication

- adversary
  - types of attacks
- trusted set-up
  - secret key is distributed securely
  - secret key remains secret
- trust basis
  - underlying primitives are secure
  - **PRG, PRF, hashing, ...**
    - **e.g., block ciphers, AES, SHA-2, etc.**



On “secret key is distributed securely”

Alice & Bob (or 2 individuals) must **securely obtain** a **shared secret key**

- “securely obtain”
  - need of a secure channel
- “shared secret key”
  - too many keys

strong assumption to accept

a problem to manage

Public-key cryptography to the rescue…

On “secret key is distributed securely”

Alice & Bob (or 2 individuals) must **securely obtain** a **shared secret key**

- “securely obtain”

(A) strong assumption to accept

  - requires secure channel for key distribution (chicken & egg situation)
  - seems impossible f t relationship

  - not easily justifiabl

- “shared secret key” **ng problem** to manage
  - requires too many keys, namely $O(n^2)$ keys for n parties to communicate
  - imposes too much risk to protect all such secret keys
  - entails additional complexities in dynamic settings (e.g., user revocation)

Alternative approaches?

Need to securely distribute, protect & manage many **session-based** secret keys

- (A) for secure distribution, just "make another assumption…"
  - employ **"designated" secure channels**
    - physically protec d-proof" room)
  - employ **"trusted" p**
    - entities authorized to distribute keys ution centers (KDCs))
- (B) for secure management, just 'live w

Public-key cryptography to the rescue…

Public-key (or asymmetric) cryptography

disclaimer on names
private = secret

Goal: devise a cryptosystem where key setup is “more” manageable

Main idea: **user-specific** keys (that come in pairs)

- user U generates two keys (U_{pk}, U_{sk})
  - **U_{pk} is public** – e (even by the adversary)
  - **U_{sk} is private** – (even from other users)

Usage

- employ **public** key U_{pk} for certain “**public**” t rformed by **other users**)
- employ **private** key U_{sk} for certain “**sensitive**” tasks (performed by **user U**)

Assumption

- **public-key infrastructure (PKI)**: public keys become **securely** available to users

From symmetric to asymmetric encryption

secret-key encryption

- main limitation
  - **session-specific** keys

public-key encryption

- main flexibility
  - **user-specific** keys

- messages encrypted by receiver's PK can (only) be decrypted by receiver's SK

From symmetric to asymmetric message authentication

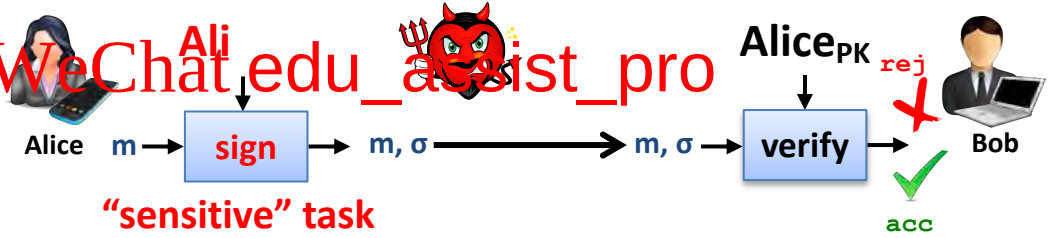
secret-key message authentication (or MAC)

- main limitation
  - **session-specific** keys

public-key message aut

(or **digital signatures**)

- main flexibility
  - **user-specific** keys



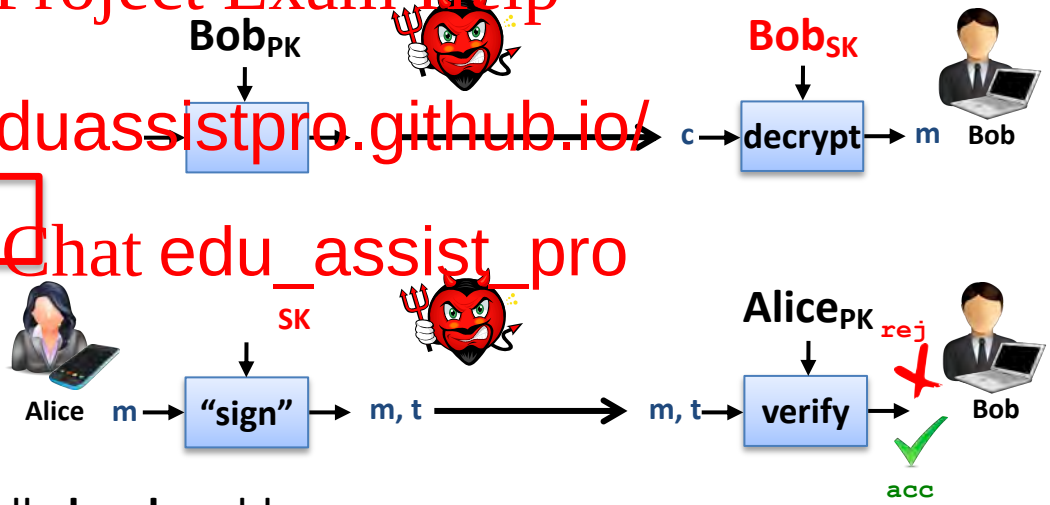
- (only) messages signed by sender's SK can be verified by sender's PK

Thus: Principles of modern cryptography

(A) security definitions, **(B) precise assumptions**, (C) formal proofs

For **asymmetric-key** message encryption/authentication

- adversary
  - types of attacks
- trusted set-up
  - PKI is needed
  - secret keys remain secret
- trust basis
  - underlying primitives are secure
  - typically, **algebraic** computationally-**hard** problems
    - e.g., **discrete log**, **factoring**, etc.



General comparison

Symmetric crypto

- key management
  - less scalable & riskier
- assumptions
  - secret & authentic co
  - secure storage
- primitives
  - generic assumptions
  - more efficiently in practice

Asymmetric crypto

- key management
  - more scalable & simpler
  - ty (PKI)
  - e
- math assumptions
- less efficiently in practice (2-3 o.o.m.)

Public-key infrastructure (PKI)

A mechanism for securely managing, in a dynamic multi-user setting, user-specific public-key pairs (to be used by some public-key cryptosystem)

- **dynamic**, **multi-user**
  - the system is open
- **user-specific public-ke**
  - each user U in the system is assigned a u U_{pk}, U_{sk})
- **secure management** (e.g., authenticated p
  - public keys are authenticated: current U_{pk} of user U is publicly known to everyone

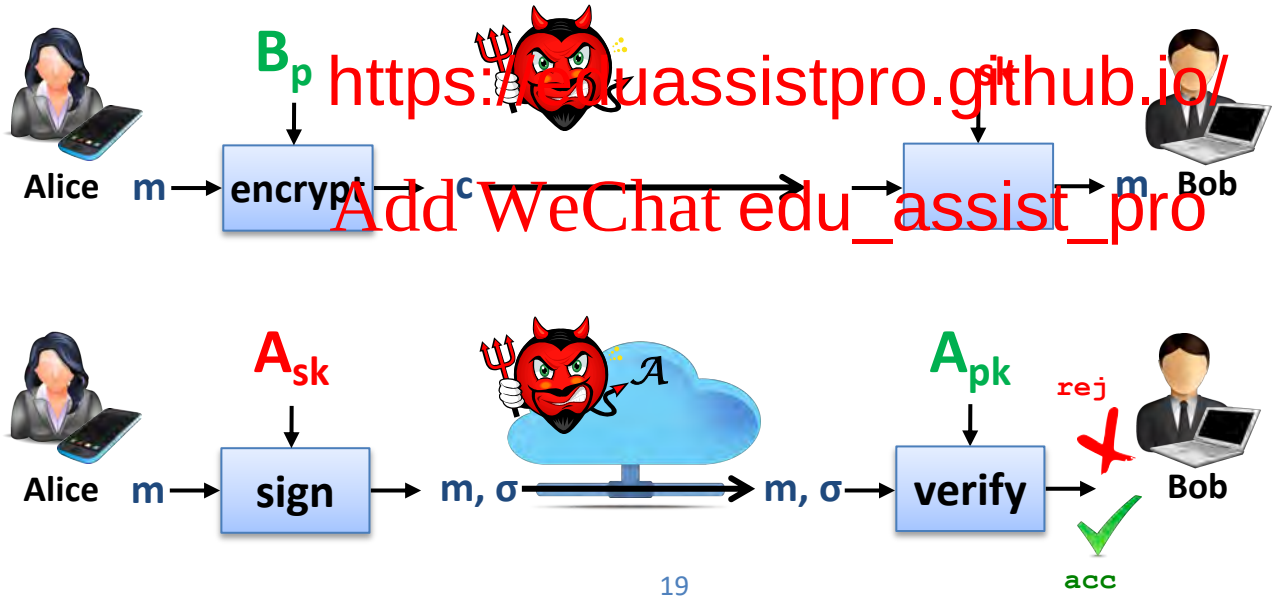
Very challenging to realize

- currently using **digital certificates**; ongoing research towards a better approach…

Overall: Public-key encryption & signatures

Assume a trusted set-up

- public keys are securely available (PKI) & secret keys remain secret



Secret-key vs. public-key encryption

Public-key cryptography: Early history

Proposed by Diffie & Hellman

- documented in “New Directions in Cryptography” (1976)
- solution concepts of public-key encryption schemes & digital signatures
- key-distribution systems
  - Diffie-Hellman k
    - “reduces” sy to

Public-key encryption was earlier (and ind ropposed by James Ellis

- classified paper (1970)
- published by the British Governmental Communications Headquarters (1997)
- concept of digital signature is still originally due to Diffie & Hellman

How to set up a PKI?

- How are public keys stored? How to obtain a user's public key?
- How does Bob know or 'trust' that A_{PK} is Alice's public key?
- How A_{PK} (a bit-string (user/identity)?

public key: A_{PK}
secret key: A_{SK}

public key: B_{PK}
secret key: B_{SK}

Achieving a PKI…

How can we maintain the invariant that at all times

- any given **user** U is **assigned** a **unique** public-private key pair; and
- any other user knows U's current public key?
  - secret keys can be l ked

Recall

- PK cryptosystems come with a Gen algorith y U
  - on input a security-strength parameter, it outputs a random valid key pair for U
- public keys can be made publicly available
  - e.g., sent by email, published on web page, added into a public directory, etc.

entails binding users/identities to public keys

Distribution of public keys

Public announcement

- users distribute public keys to recipients or broadcast to community at large

Publicly available directory

- can obtain greater s blic directory

Both approaches have pro ries

Do you trust your public key?

- Impostor claims to be a true party
  - true party has a public and private key
  - impostor also has a public and private key
- Impostor sends imp erifier
  - says, “This is the true party’s public key”
  - this is the critical step in the deception

Certificates: Trustable identities & public keys

Certificate

- a public key & an identity **bound** together
- in a document **signed by** a certificate authority

Certificate authority (

- an authority that users **trust** to securely bind c keys

  - CA **verifies identities** before generating these identities
  - secure binding via **digital signatures**
    - **ASSUMPTION**: The authority's PK CA_{PK} is authentic

Public-key certificates in practice

Current (imperfect) practice for achieving trustable identities & public keys

- everybody trusts a Certificate Authority (CA)
  - everybody knows PK_{CA} & trusts that CA knows the corresponding secret key CA_{SK}
- a certificate binds iden statement
  - e.g., Alice obtains a 's public key is 1032xD"
- users query CA for public keys of intended gners
  - e.g., when Bob wants to send an encrypt Alice
    - he first **obtains & verifies** a certificate of Alice's public key
  - e.g., when Alice wants to verify the latest software update by Company
    - she first **obtains & verifies** a certificate of Company's public key

Example

a certificate is a public key and an identity bound together and signed by a certificate authority (CA)

document signed by CA

a certificate authority is an **authority** that users **trust** to accurately verify identities before generating certificates that bind those identities to keys

Certificate hierarchy

Single CA certifying every public key is impractical

Instead, use trusted root certificate authorities

- root CA signs certifica
  they sign certificates f
  - certificate "chain of trust"
    - $sig_{Symantec}$("Stevens", $PK_{Stevens}$)
    - sig_{UMD}("faculty", $PK_{faculty}$)
    - $sig_{faculty}$("Nikos", PK_{Nikos})

Example 1: Certificate signing & hierarchy

To create Diana's certificate:

Diana creates and delivers to Edward:

Name: Diana Position: Division Manager Public key: 17EF83CA ...

Edward adds:

Name: Diana Position: Division Public key: 17EF83	hash value anager

Edward signs with his priv

Name: Diana Position: Division Manager Public key: 17EF83CA...	hash value 128C4

Which is Diana's certificate.

To create Delwyn's certificate:

Delwyn creates and delivers to Diana:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...

Diana adds:

Name: Delwyn 3882C ..	hash value 48CFA

rivate key:

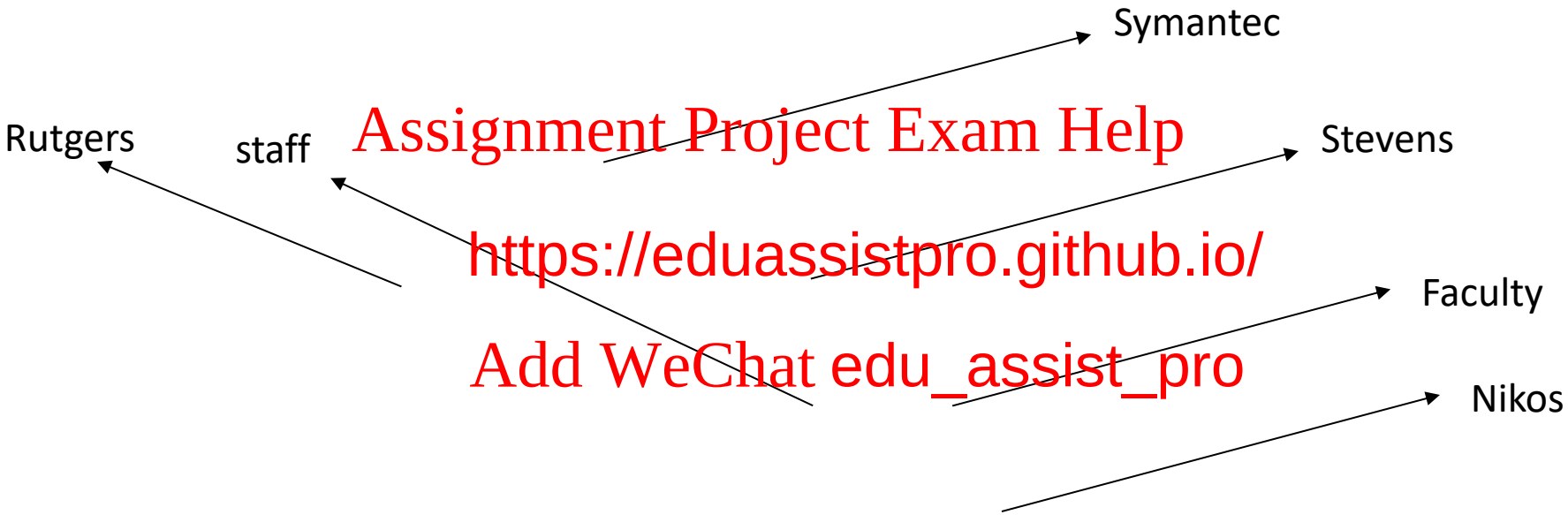
ager 82C ...	hash value 48CFA

And appends her certificate:

Name: Delwyn Position: Dept Manager Public key: 3AB3882C ...	hash value 48CFA
Name: Diana Position: Division Manager Public key: 17EF83CA ...	hash value 128C4

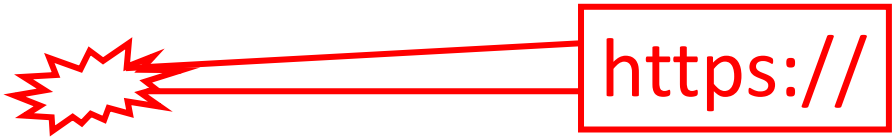
Which is Delwyn's certificate.

Example 2



What bad things can happen if the root CA system is compromised?

Secure communication over the Internet



What cryptographic keys are used to protect communication?

X.509 certificates

Defines framework for authentication services

- defines that public keys stored as certificates in a public directory
- certificates are issued and signed by a CA

Used by numerous app

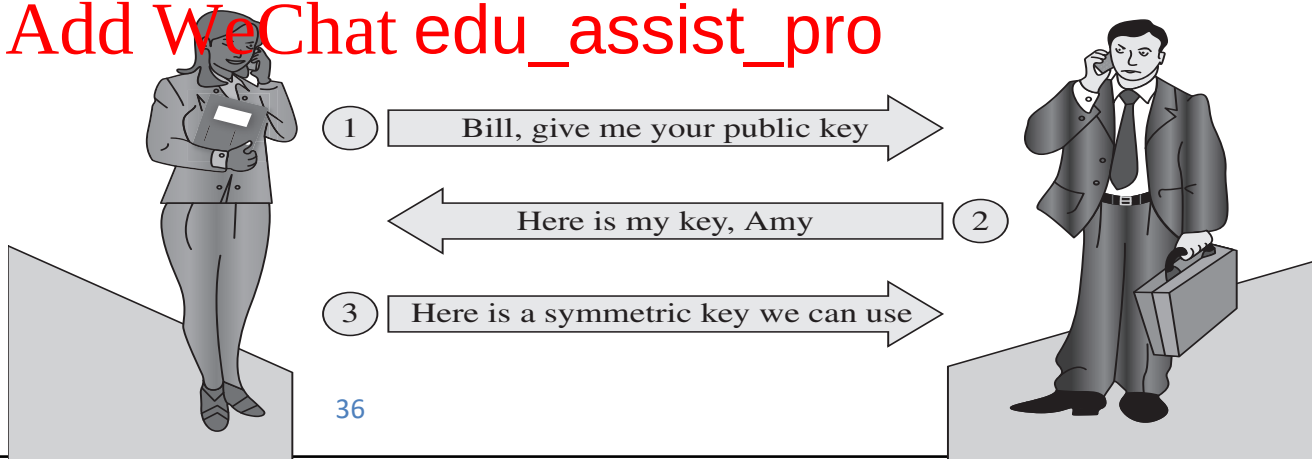
Example: see certificates accepted by you

Secret-key cryptography is “reduced” to public-key

PK encryption can be used “on-the-fly” to securely distribute session keys

Main idea: Leverage PK encryption to securely distribute session keys

- sender generates a fresh session-specific secret key k and **learns** receiver’s public key R_{pk}
- session key k is sent to
- session key k is emplo
  - e.g., how **not** to run above protocol



Hybrid encryption

"Reduces" secret-key crypto to public-key crypto

- better performance than block-based public-key CPA-encryption
- main idea
  - apply PK encryptio
  - use k for secret-key

Hybrid encryption using the KEM/DEM approach

"Reduces" secret-key crypto to public-key crypto

- main idea
  - **encapsulate** secret key k into c
  - use k for secret-key
  - KEM: key-encapsul
  - DEM: data encapsulation mechanism - E
- KEM/DEM scheme
  - CPA-secure if KEM is CPA-secure and Enc' EAV-secure
  - CCA-secure if KEM and Enc' are CCA-secure

Discrete Log & its ns

The discrete logarithm problem

Setting

- if p be an odd prime, then G = (Z_p^*, ·) is a cyclic group of order p – 1
  - Z_p^* = {1, 2, 3, …, p 1} generated by some g in Z_p^*
    - for i = 0, 1, 2, …, roduces all elements in Z_p^*
  - for any x in the gro some integer k
  - k is called the **discrete logarithm** (or log)

Example

- (Z_{17}^*, ·) is a cyclic group G with order 16, 3 is the generator of G and $3^{16} = 1 \bmod 17$
- let k = 4, $3^4 = 13 \bmod 17$ (which is easy to compute)
- the inverse problem: if $3^k = 13 \bmod 17$, what is k? what about **large p**?

Computational assumption

Discrete-log setting

- cyclic $G = (Z_p^*, \cdot)$ of order $p - 1$ generated by g, prime p of length t ($|p|=t$)

Problem

- given G, g, p and x in Z (mod p)
- we know that $x = g^k$ m p-2}… but

Discrete log assumption

- for groups of specific structure, **solving the discrete log problem is infeasible**
- any efficient algorithm finds discrete logs negligibly often (prob = $2^{-t/2}$)

Brute force attack

- cleverly enumerate and **check $O(2^{t/2})$ solutions**

ElGamal encryption

Assumes discrete-log setting (cyclic $G = (Z_p^*, \cdot) = \langle g \rangle$, prime p, message space Z_p)

Gen

- secret key: random number $x \in Z_p^*$ public key: $A = g^x \bmod p$, along w/ G, g, p

Enc

- pick a fresh random r
- send ciphertext $\mathbf{Enc_{PK}}$ **1 2** **$c = m \cdot R \bmod p$**

Dec

- **$\mathbf{Dec_{SK}(c_1,c_2) = c_2\ (1/c_1^x) \bmod p}$** w **1**

Security is based on **Computational Diffie-Hellman** (CDH) assumption

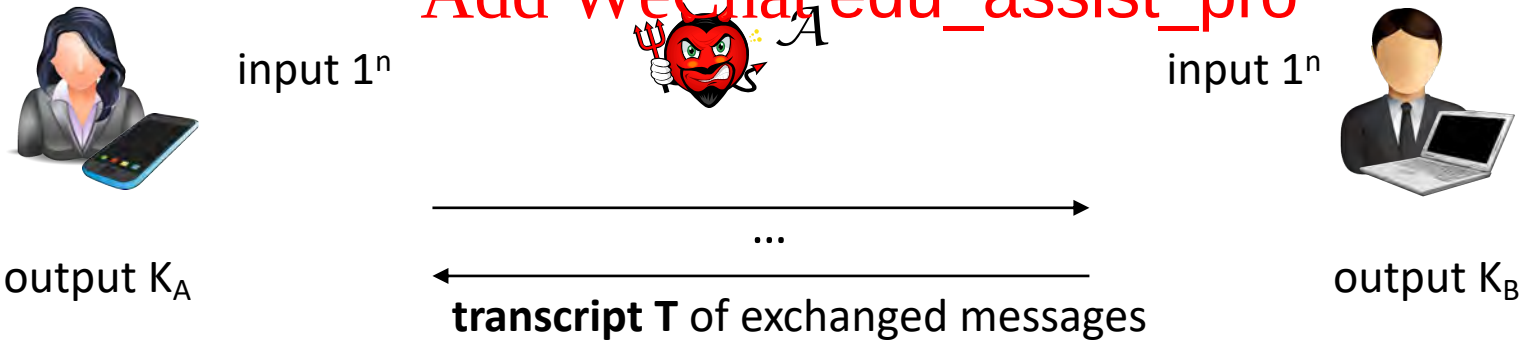
- given (g, g^a, g^b) it is hard to compute g^{ab}

A signature scheme can be also derived based on above discussion

Application: Key-agreement (KA) scheme

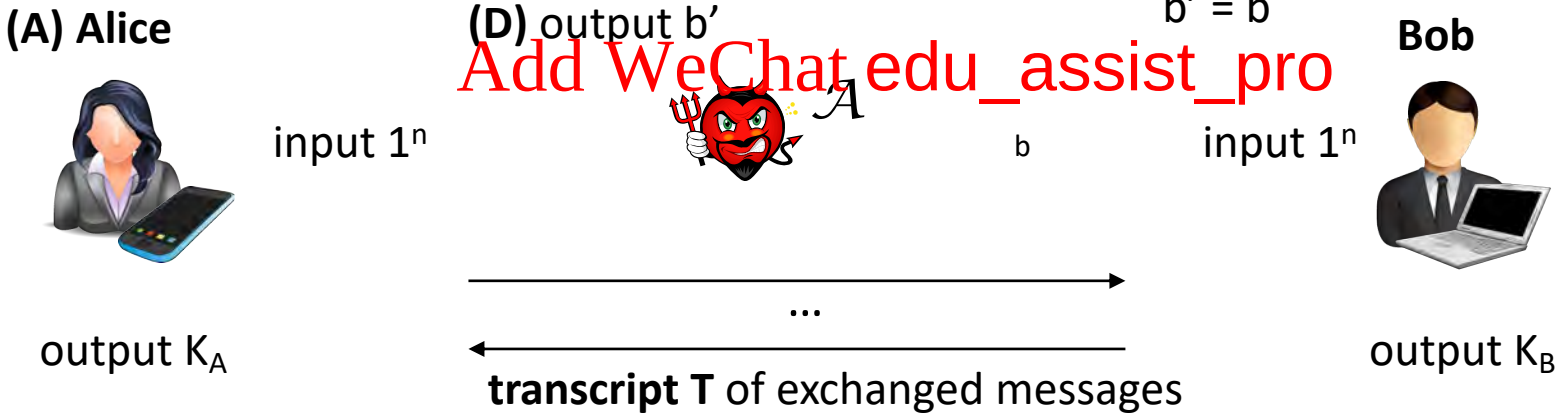
Alice and Bob want to securely establish a **shared key** for secure chatting over an **insecure** line

- instead of meeting in person in a secret place, they want to use the insecure line…
- KA scheme: they run a key agreement protocol Π to contribute to a shared key K
- correctness: $K_A = K_B$
- security: no PPT adversar in a truly random one



Key agreement: Game-based security definition

- scheme $\Pi(1^n)$ runs to generate $K = K_A = K_B$ and transcript T; random bit b is chosen
- adversary $\mathcal{A}$ is given T and k_b; if b = 1, then $k_b = K$, else k_b is random (both n-bit long)
- $\mathcal{A}$ outputs bit b' and wins if b' = b
- then: **Π is secure if no P**



(B) b is randomly chosen

The Diffie-Hellman key-agreement protocol

Alice and Bob want to securely establish a **shared key** for secure chatting over an **insecure** line

- DH KA scheme Π
  - discrete log setting: p, g public where $\langle g \rangle = Z^*_p$ and p prime

input 1^n input 1^n

(1) randomly pick secret a

(3) send g^a mod p

(2) randomly pick secret b

(4) send g^b mod p

(5) set **K = g^{ab} mod p** = $(g^b \bmod p)^a \bmod p$

(6) set **K = g^{ab} mod p** = $(g^a \bmod p)^b \bmod p$

Security

- discrete log assumption is necessary but not sufficient
- decisional DH assumption
  - given g, g^a and g^b, g^{ab} is computationally indistinguishable from uniform

Authenticated Diffie-Hellman