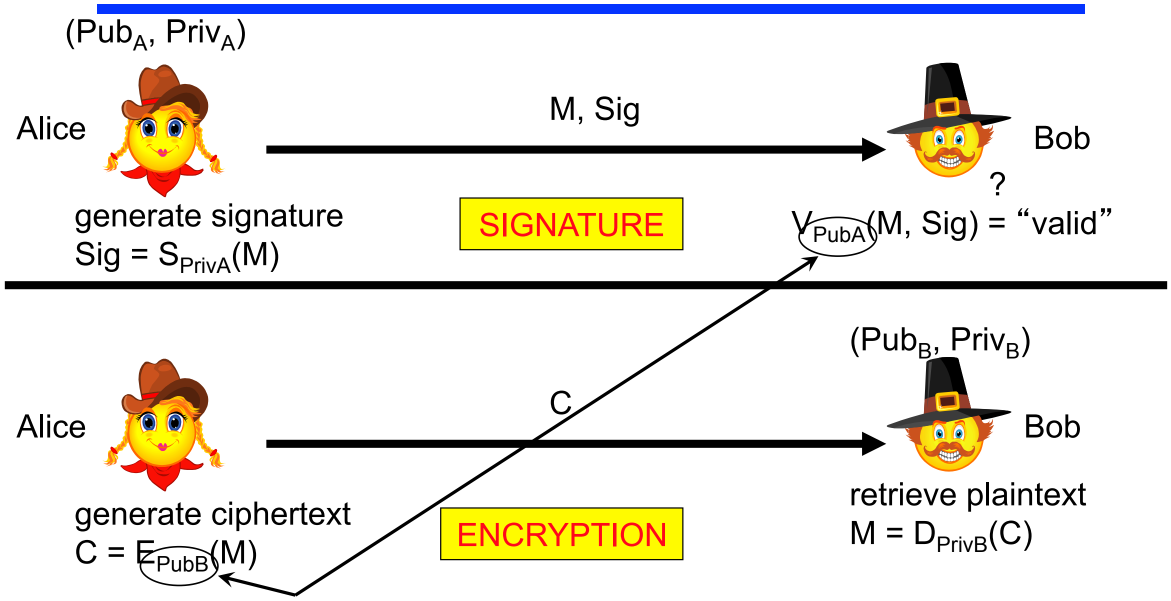
CS 408
Cryptography & Internet Security

Lecture 22:

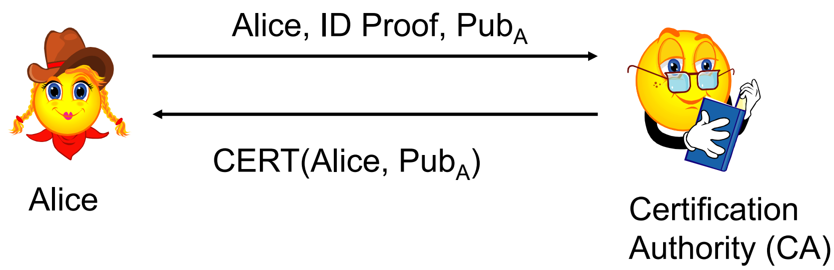
Key agreement based on asymmetric techniques

Public-key Cryptographic Primitives



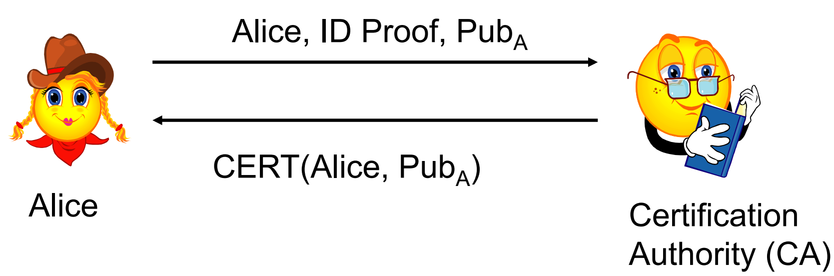
How to ensure the authenticity of the other's party public key?

Public Key Infrastructure (PKI)



- CERT(Alice, Pub_A) is Alice's public key certificate, which binds Alice's identity to her public key
  - signed by the CA (using the CA's private key)
- Anyone can verify authenticity of $CERT_A$ by using the CA's public key
- The CA's public key is readily available in a ***root certificate***
  - Included in the browser, published online, or in a newspaper, or on a CD etc.
  - The root certificate is a ***self-signed certificate*** (signed with the private key corresponding to the actual public key contained in the certificate)

Public Key Infrastructure (PKI)



- To verify a signature from Alice:
  - Bob retrieves Alice's certificate $CERT_A$ = CERT(Alice, Pub_A)
  - Bob can verify $CERT_A$ by using the CA's public key
  - Bob can verify the signed message using Pub_A

PKI

- A public key certificate contains several fields:
  - The identity of the public key's owner
  - The public key
  - Serial number
  - Expiration date
  - Other useful fields

Public Key Infrastructure

- When Alice needs Bob's public key, she retrieves Bob's certificate: CERT(Bob, Pub_B)
  - Alice has the authentic public key of the CA, so she can verify the authenticity of Bob's certficate
  - This validates the authenticity of Bob's public key, Pub_B, which is contained inside Bob's certificate
- A Root Certificate acts as an ***anchor point*** in the *chain of trust*
  - They are used to validate certificates lower in the PKI hierarchy
- PKI = the entire infrastructure needed to support public key cryptography
  - Includes organizations (CAs), principals, and their interactions

Remember this Group and its Properties?

- Let p be a prime integer and let Z^*_p be the set {1, 2, …, p-1}.
  $(Z^*_p, \bullet)$ is a group, where • is multiplication modulo p.

- Properties of $(Z^*_p, \bullet)$:
  - The order of $(Z^*_p, \bullet)$ is p-1
  - $(Z^*_p,\bullet)$ is always cyclic (this means that this group admits a generator)
  - In $(Z^*_p,\bullet)$, a generator element is an element whose order is equal to p-1
    - Every element in $(Z^*_p,\bullet)$ can be written as a "power" of a generator element

Discrete Logarithm

Definition

Let *p* be a prime, G = $(Z^*_p, \bullet)$ be a cyclic group, and *g* be a generator (primitive element) of G. Then, every element *a* of G can be written as $g^k \equiv a \bmod p$ for some integer *k*.
k is called the the discrete logarithm of *a* to base *g* modulo *p*.

Example

Z^*_{97} is cyclic group of order 96. A generator of Z^*_{97} is *g=5*.
Since $5^{32} \equiv 35 \pmod{97}$, we have that $\log_5 35 = 32$ in Z^*_{97}

Discrete Log Problem

Discrete Log Problem (DLP):

Given a prime p, a generator g of Z^*_p, and an element $y \in Z^*_p$, find the integer x, $0 \leq x \leq p-2$, such that $g^x \equiv y \pmod{p}$

Difficulty of solving DLP: when p is large enough, no efficient algorithms are known to solve the DLP problem

- p should have at least 1024 bits

Diffie-Hellman Problem

Diffie-Hellman Problem (DHP):

Given a prime p, a generator g of Z^*_p, and elements $g^x \pmod{p}$ *and* $g^y \pmod{p}$, find $g^{xy} \pmod{p}$

Difficulty of solving DHP: when p is large enough, no efficient algorithms are known to solve the DHP problem

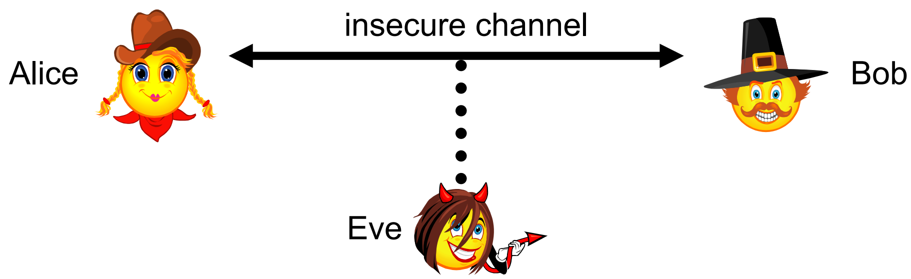
- p should have at least 1024 bits

FACT

If one can solve the DLP problem, then one can also solve the DHP problem. Why?

- we say that the DHP problem reduces to the DLP problem

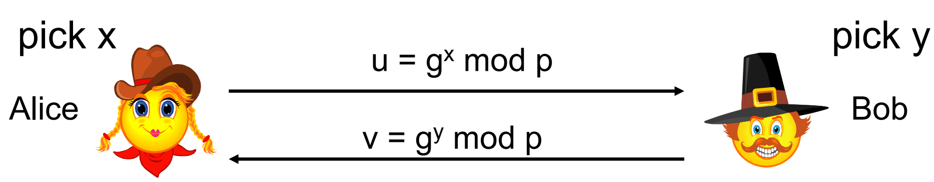
Diffie-Hellman Setting



- No previous contact between A and B
- Both A and B have a computer
- Eve can hear every single message exchanged between A and B

Can A and B establish a secret key (which Eve doesn't know)?

Diffie-Hellman Key Agreement Protocol



$K = v^x = g^{xy}$ $K = u^y = g^{xy}$

- One-time setup:
  - A large prime p and a generator g of Z^*_p are selected and published in advance
- x and y are randomly chosen by the parties and are kept secret

Diffie-Hellman Key Agreement Protocol

pick x

Alice

$u = g^x \bmod p$ →

← $v = g^y \bmod p$

pick y

Bob

$K = v^x = g^{xy}$

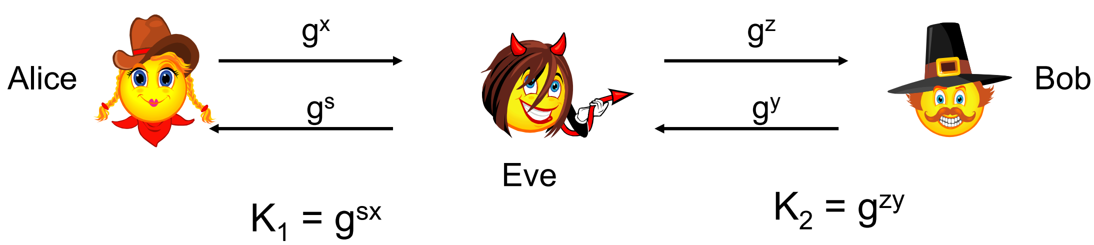
$K = u^y = g^{xy}$

- The established session key is $K = g^{xy} \bmod p$
- From p, g, u, v, Eve cannot deduce K !
  - Security is based on the difficulty of the DHP and DLP problems
- Protocol trivially achieves perfect forward secrecy because there are no long-term keys to be compromised
  - But x and y must be discarded at the end of the session

Diffie-Hellman: What Can Go Wrong?

- Simple DH protocol is only secure against passive adversaries
- With <u>active</u> adversaries, the protocol is vulnerable to Man-In-The-Middle (MITM) attacks
- Also, simple DH is anonymous (A and B know they establish a key with somebody, but they don't know with whom!)

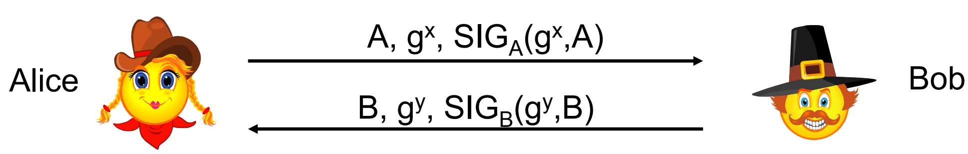
DH: Man-In-The-Middle Attack



- Eve can change messages between A and B
- Eve can forge messages from either party to the other
- Protocol is broken! (A and B believe they talk to each other, when in fact, each one of them talks to Eve)

- How to achieve *mutual authentication*? (each party can verify the identity of the peer with whom the session key is established)

Proposal 1 for Authenticated DH Protocol



- A and B have long-term signature keys
- Protocol satisfies the perfect forward secrecy requirement
  - Exponents must be chosen fresh and independent for each session
  - Exponents must be erased immediately after computation of the key g^{xy}

Proposal 1 for Authenticated DH Protocol

Alice 

A, g^x, $SIG_A(g^x,A)$ →

← B, g^y, $SIG_B(g^y,B)$

 Bob

- What is wrong?

  (remember, in addition to mutual authentication, also want to achieve:
  - perfect forward secrecy
  - protection against known-key attacks
  - protection against replay attacks

  (and protection against any combination of the above)

Proposal 1 for Authenticated DH Protocol

Alice 

A, g^x, $SIG_A(g^x,A)$ →

← B, g^y, $SIG_B(g^y,B)$

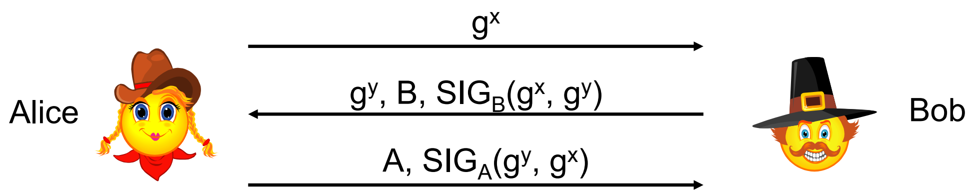
 Bob

- What is wrong? Known-key attack!

  (exposure of session keys (secrets) for a specific session should not affect the security of other sessions)
- Eve gets some secrets for an old session (e.g., the secret exponent of one of the parties)
- Eve impersonates Alice by replaying g^x, $SIG_A(g^x)$ and by using knowledge of x

  (Eve can do this without even breaking the long-term signature key of Alice)

Proposal 2 for Authenticated DH Protocol



- What is wrong?
- Does not meet *consistency* property

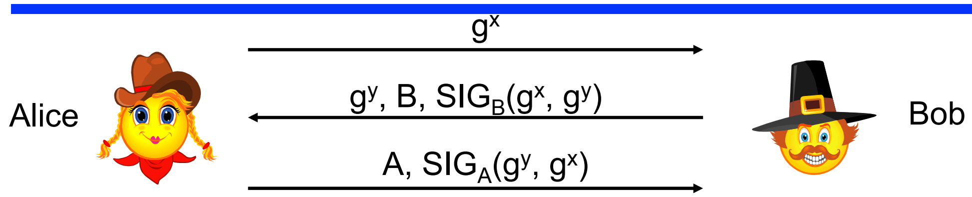
  (if two honest parties establish a common session key, then both parties need to have a consistent view of who the peers to the session are)

More on *consistency*

If two honest parties establish a common session key, then both parties need to have a consistent view of who the peers to the session are

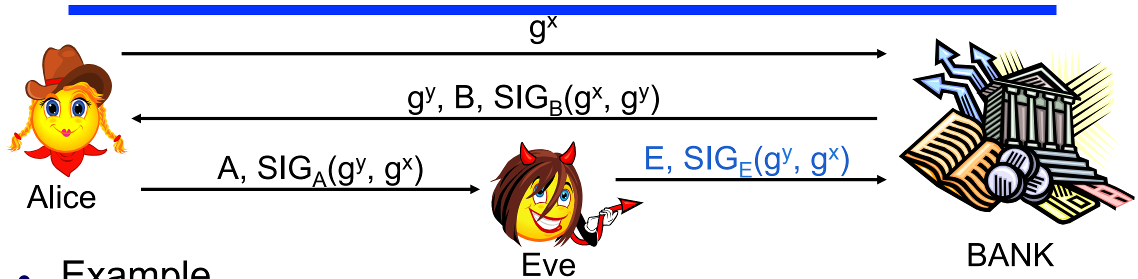
- If a party A establishes a session key K and believes that the peer to the exchange is B, then if B establishes the same session key K then B needs to believe that the peer to the exchange is A
- And Vice-versa

Proposal 2 for Authenticated DH Protocol



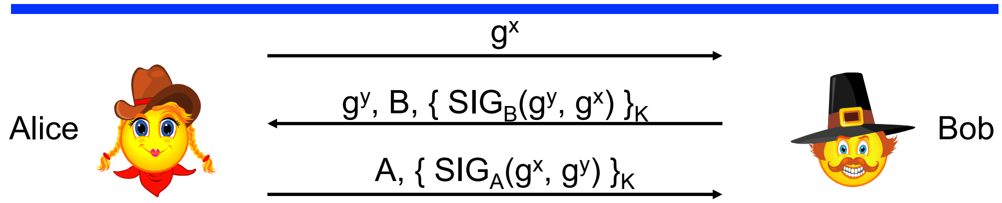
- Eve lets the first two messages go through and replaces the third message with:
  E ⇒ B: E, $SIG_E(g^y, g^x)$
- A believes it has exchanged key K with B
- B believes it has exchanged key K with E
- This is not a breach of secrecy, but a severe breach of authentication (A and B will use the same key with different understandings of who the peer exchange is)
  ⇒ protocol doesn't meet *consistency*

Proposal 2 for Authenticated DH Protocol



- Example
  - Bob is a bank
  - Alice is a customer that wants to send to the bank a monetary deposit
- After key K is established:
  - Alice sends deposit securely using key K
  - Bank believes the deposit is coming from Eve
  - Money will be considered to belong to Eve (rather than to Alice)
- This is an *identity misbinding attack*
  (protocol fails to provide an authenticated binding between the key and the honest identities)

Station-to-Station Protocol (STS)



where $K = g^{xy}$

Protocol provides:

- session key secrecy
- perfect forward secrecy
- protection against known-key attacks
- protection against replay attacks
- consistency

Recommended Reading

- Parts of Chapter 10 (for Key establishment, Needham-Schroeder, and Public key-based key agreement)