Lab #5 - October 4, 2018

Due Dec 31 at 11:59pm **Points** 7 **Questions** 7
Available Oct 4 at 9:20am - Dec 31 at 11:59pm 3 months **Time Limit** None
Allowed Attempts Unlimited

Instructions

[1] Good morning and welcome to the fifth lab session!

Lab sessions provide the opportunity for recitation and more in-depth understanding of the materials covered in class, as well as preparation for upcoming homework assignments.

Your attendance only of a given lab session (and, thus, your participation in the assignments and/or discussions) gives you full credit. You are expected to stay in the lab for the entire session, or until the TAs release the class possibly earlier than scheduled, and to actively participate in the discussions (e.g., asking questions, answering to questions, etc.).

Typically, lab assignments are offered in the form of an ungraded quiz, which should not be interpreted as a test or a mini exam.

[2] Today's quiz is planned so that it reviews materials covered in our last lecture in class.

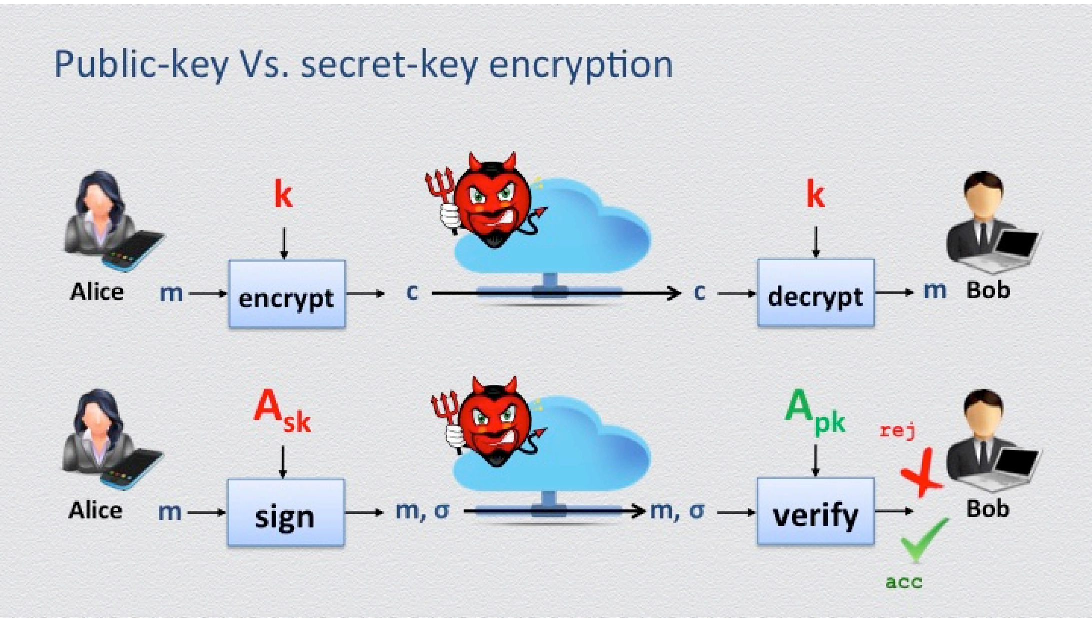
Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	6 minutes	2.67 out of 7

Submitted Oct 4 at 1:46pm

Question 1 1 / 1 pts



Public-key Vs. secret-key encryption

Recall the "key" difference between asymmetric and symmetric encryption.

What is the main advantage of using public keys?

○ Secure setup is no longer required.

Correct!

● Key management is easier using user-specific keys.

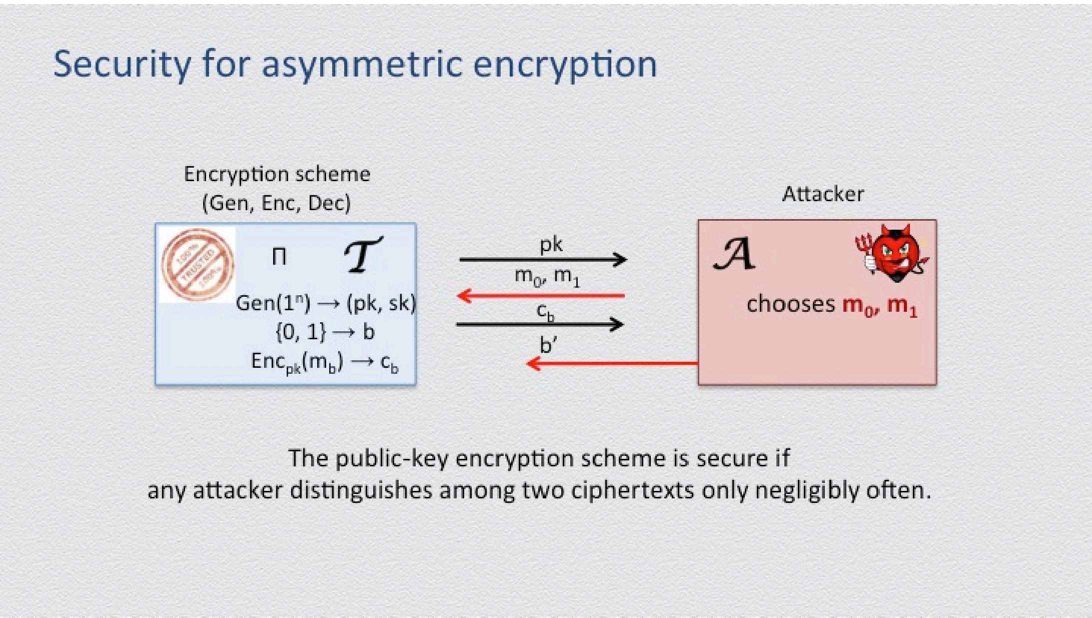
○ Stronger secury by operating over elements of a large group.

In the public-key setting:

- Key management becomes easier (i.e., more scalable);
- The setup security assumptions simply become different assumptions, not necessarily weaker (i.e., better);
- The security definitions are equally strong; it depends on the specific encryption scheme if it offer stronger or weaker security.

Question 2

0 / 1 pts



Security for asymmetric encryption

Encryption scheme
(Gen, Enc, Dec)

Π $\mathcal{T}$

$\mathrm{Gen}(1^n) \rightarrow (pk, sk)$
$\{0, 1\} \rightarrow b$
$\mathrm{Enc}_{pk}(m_b) \rightarrow c_b$

Attacker

$\mathcal{A}$

chooses m_0, m_1

The public-key encryption scheme is secure if
any attacker distinguishes among two ciphertexts only negligibly often.

Recall the "ciphertext indistinguishability" security definition for public-key encryption.

Why any secure public-key encryption scheme must necessarily be probabilistic?

○ Because all known public-key encryption schemes use a fresh random nonce to mask the plaintext message.

orrect Answer

○ Because otherwise function Enc() can be used by the attacker to trivially win the security game.

'ou Answered

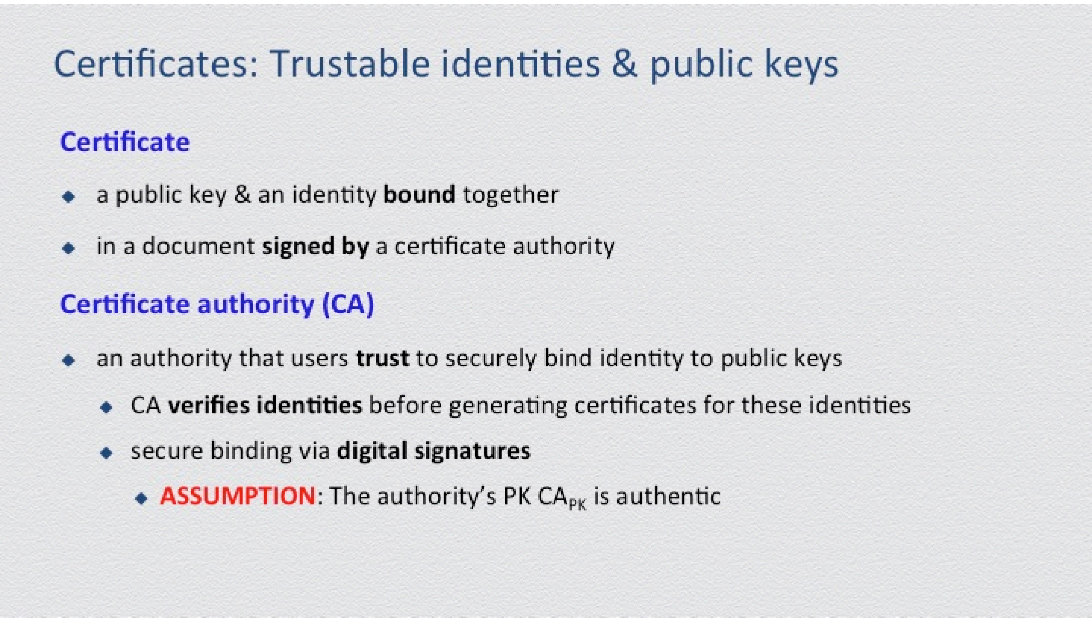
● Because chosen-plaintext attacks are not possible when public-key encryption is considered.

Knowledge of pk allows anyone to learn the ciphertext of any plaintext. So, if the encryption scheme is not randomized, Enc() is a deterministic function, mapping always the same input to the same output. So, the attacker can learn the ciphertext of m_0 and the ciphertext of m_1, prior to sending these messages to the trusted party T, and thus use this knowledge to find which of the two messages was encrypted by T and win the game!

Question 3

0.67 / 1 pts

Lab5_Question8_CA.jpg



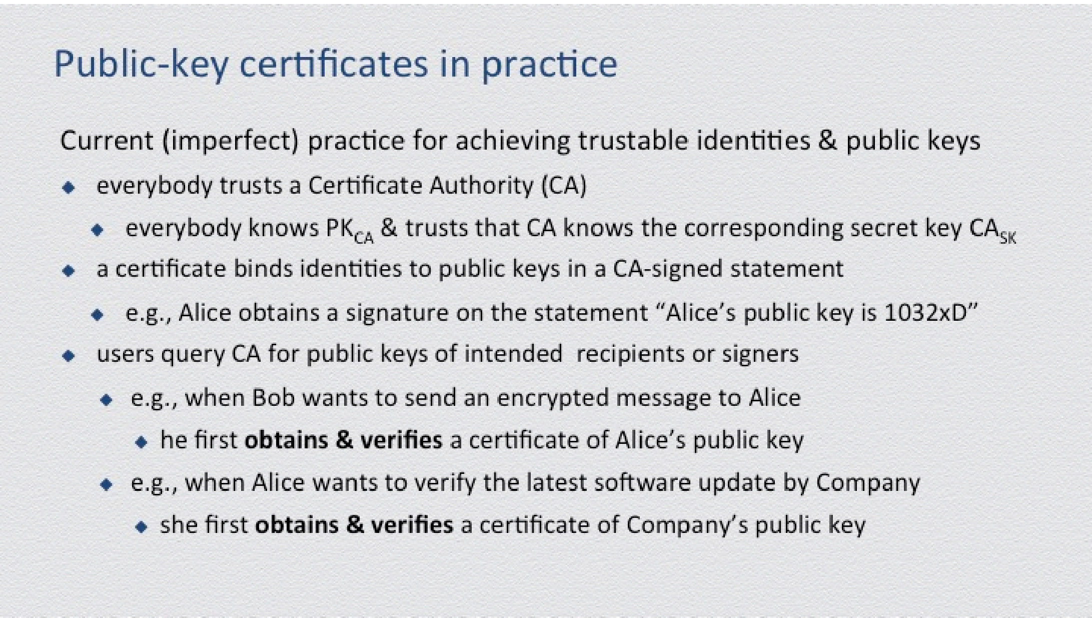
Certificates: Trustable identities & public keys

Certificate

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

Certificate authority (CA)

- an authority that users **trust** to securely bind identity to public keys
  - CA **verifies identities** before generating certificates for 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 identities to public keys in a CA-signed statement
  - e.g., Alice obtains a signature on the statement "Alice's public key is 1032xD"
- users query CA for public keys of intended recipients or signers
  - e.g., when Bob wants to send an encrypted message to 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

Lab5_Question8_CA3.jpg

Recall the problem that we discussed in the class related to the management of public keys, namely, how to reliably establish a trustworthy PKI. The main current solution is the use of public key certificates which are explained. The idea is to employ digital signatures, computed by a few trustworthy authorities, on statements that bind identities to their public keys.

Which of the following are disadvantages of such an approach?

orrect Answer

□ We solve a problem related to the authenticity of public keys, but we do so by employing a technology that is based on public-key message authentication.

Correct!

☑ Management of public keys is not necessarily more efficient or more secure.

Indeed, certificates typically have an expiration date, after which they are considered invalid, or they often must be revoked because an attack against a CA may become known or because a user may loose its corresponding secret key or my suspect it has leaked to an attacker. Thus, reissuing and revocation of public-key certificates make their secure management a complicated problem.

Correct!

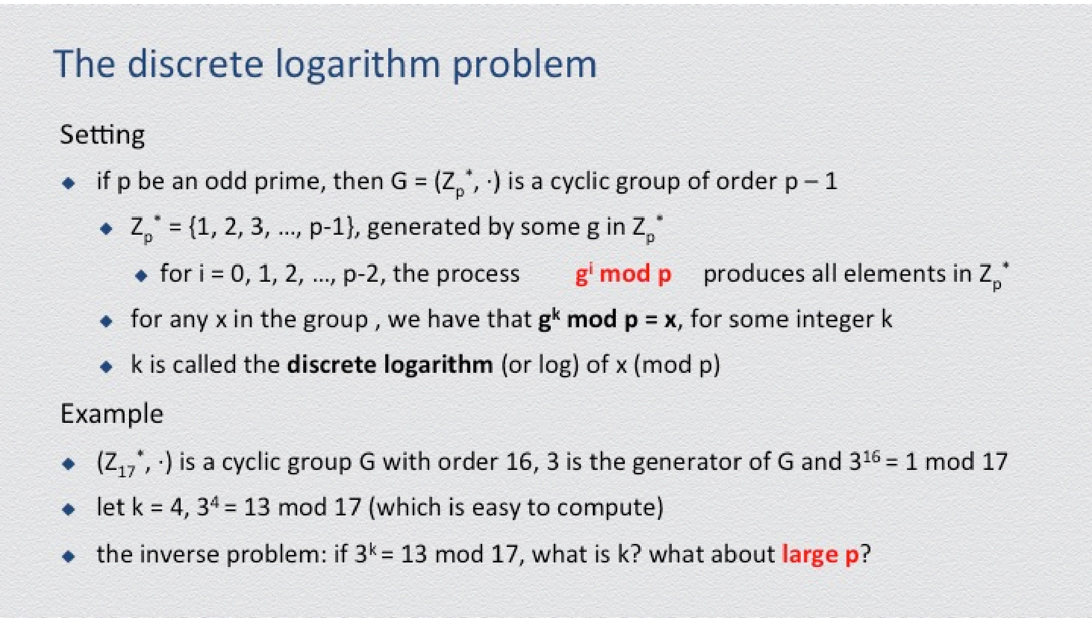
☑

Although a centralized trust model (the basis of trust is on one or few CAs) potentially reduces the attack surface, it also introduces a single (or few) points of failure.

Indeed, what happens if an attacker succeeds in impersonating a CA? This has happened many times in real life...

Question 4

0 / 1 pts



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, ..., p-2, the process g^i **mod p** produces all elements in Z_p^*
  - for any x in the group , we have that **g^k mod p = x**, for some integer k
  - k is called the **discrete logarithm** (or log) of x (mod p)

Example

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

Let's recall the discrete logarithm problem that we have described in class.

Why it is easy to compute $3^{1580212}$ mod 17?

○

Because the modulo is small - the larger the modulo the harder the computation's complexity.

ou Answered

●

Because 3 and 17 are co-primes, we know that 3^x = 1 mod 17 for any integer x.

No. The property holds only for those x that 16 divides.

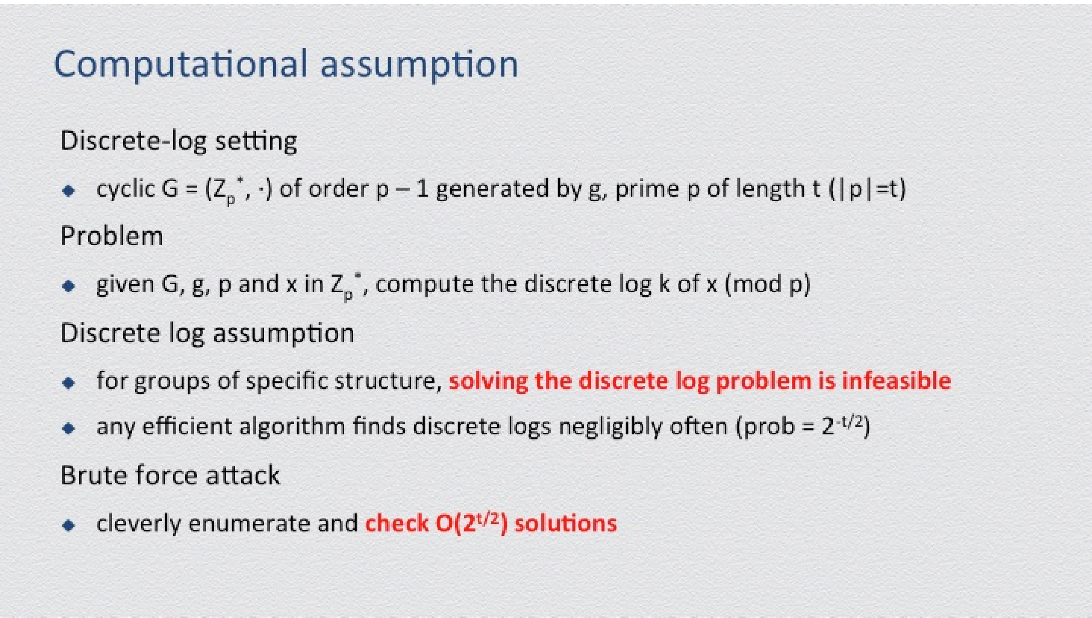
orrect Answer

○ Because we know how 1580212 mod 16 is written in binary.

○ Because 3 and 17 are small integers.

Question 5

0 / 1 pts



Computational assumption

Discrete-log setting

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

Problem

- given G, g, p and x in Z_p^*, compute the discrete log k of x (mod p)

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

Is the discrete log problem well defined?

'ou Answered

● No. Element x should be relative prime to g.

No such restriction exists.

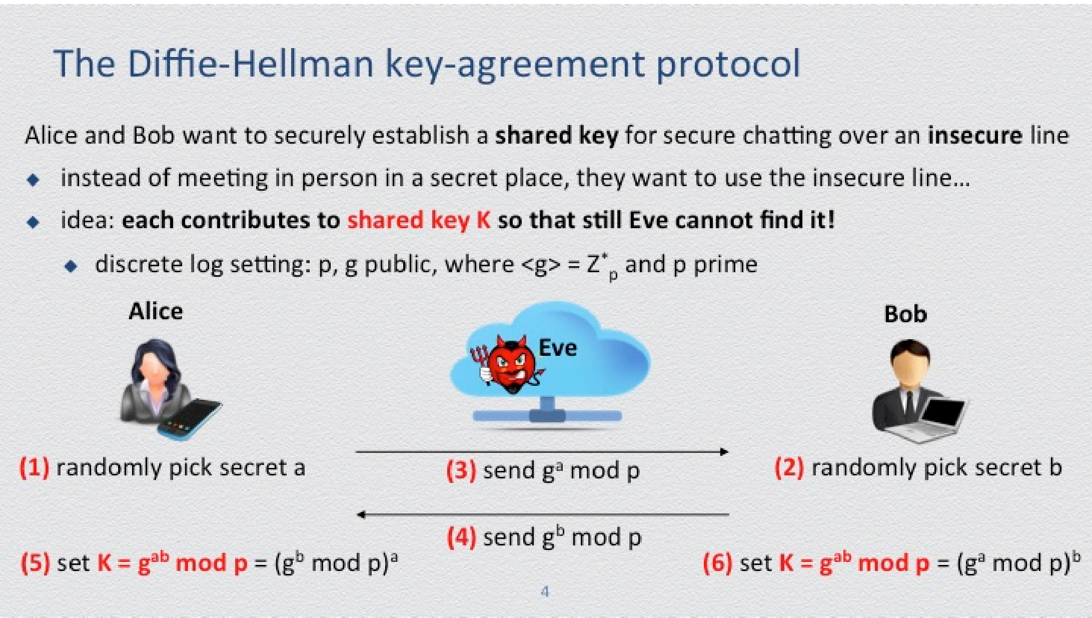
orrect Answer

○ Yes. Because Z^*_p is a cyclic group.

○ Not always. Unless element x divides g, the problem is vacuous.

Question 6

1 / 1 pts



The Diffie-Hellman key-agreement protocol

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…
- idea: **each contributes to shared key K so that still Eve cannot find it!**
  - discrete log setting: p, g public, where $<g> = Z^*_p$ and p prime

(1) randomly pick secret a

(2) randomly pick secret b

(3) send g^a mod p

(4) send g^b mod p

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

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

4

Consider the above simple and elegant protocol - suggested by Diffie and Hellman - for securely setting up a shared secret key over an insecure channel. Here, Eve is assumed to be able to eavesdrop but be incapable of launching a woman-in-the-middle attack.

Based on the fact that successive exponentiations constitute a commutative operation (i.e., the order by which we exponentiate does not matter), Alice and Bob correctly agree on key K = g^{ab} mod p, where a, b are secret exponents contributed respectively by Alice and Bob. These secret values are securely exchanged by "hiding" them in the exponent, i.e., by transmitting g^a mod p and g^b mod p, respectively. Because discrete logs are hard to compute, Eve cannot directly compute a or b!

Is the assumption that discrete logs are hard to compute a sufficient or necessary condition for the security of the above protocol?

Correct!

◉ A necessary condition.

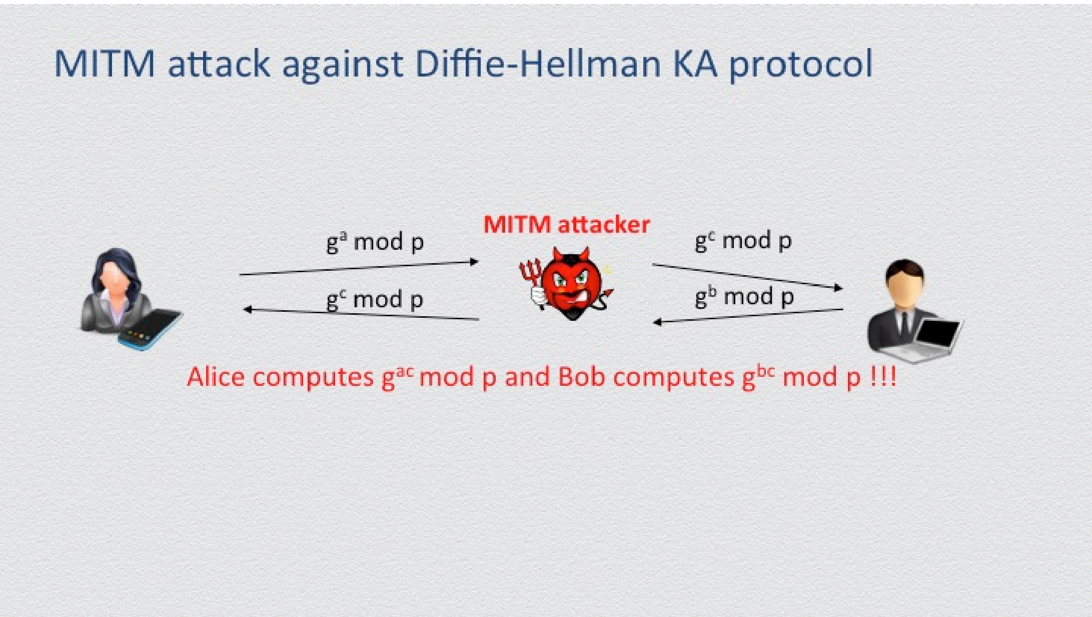
Indeed, if discrete logs are easy to find then the protocol is trivially insecure, as Eve learns directly the secret values a and b. However, even when such direct computation of a, b is not possible (in particular, when discrete logs are hard to compute), we can still not assess the security of the protocol: Eve knows something about a and b, namely g^a mod p and g^b mod p, and this information may be useful in efficiently computing the shared key gab mod p.

○ A sufficient condition.

Thus, we need to assume that given (g, p, g^a mod p, g^b mod p), it is (still) computationally hard to compute g^{ab} mod p, that is a stronger assumption than the hardness of finding discrete logs. This assumption is called Computational DH assumption (and implies discrete logs are hard to find - or else computing g^{ab} mod p is easy).

Question 7

0 / 1 pts



MITM attack against Diffie-Hellman KA protocol

Alice computes g^{ac} mod p and Bob computes g^{bc} mod p !!!

In the above key agreement protocol (and in a very similar way with the one we described in class when we discussed how a shared key can be simply established via public key encryption), things can go bad if Eve is actually Mallory, acting as a woman-in-the-middle, thus being able to not only intercept but also tamper with the exchanged messages.

If we assume that a PKI has been securely established, how we can prevent such man or woman-in-the-middle attacks?

○ By having each party additionally encrypting their sent messages.

'ou Answered

◉ In the case of an active adversary, key establishment is inherently impossible to securely achieve.

No. Authenticated key agreement protocols provide a secure solution simply by using signatures for message authentication.

orrect Answer

○ By having each party additionally signing their sent messages.