Homework 3 and 4

- Homework 3 is due now
- Homework 4 has been posted

Cryptography
CS 555

Topic 28: Key-Management, Diffie-Hellman Key Exchange

Recap

- Factoring Algorithms
- Discrete Log Attacks
- NIST Security Recommendations

Key-Exchange Problem

- **Key-Exchange Problem:**
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don’t have time to meet privately and generate one
    - Obi-Wan and Yoda share an asymmetric key with Anakin
    - Suppose that they fully trust Anakin

Key-Distribution Center (with Symmetric Key-Crypto)

Enc(K_{obiwan},"I would like to talk to Yoda")

Ok, here is a fresh key that no sith lord has seen

c_1=**Enc**($\mathbf{K}_{\mathbf{obiwan}}$,ts, $\mathbf{K}_{\mathbf{new}}$),
c_2=**Enc**($\mathbf{K}_{\mathbf{yoda}}$, ts, "Obiwan/Yoda", $\mathbf{K}_{\mathbf{new}}$)

$\mathbf{K}_{\mathbf{obiwan}}$: Shared key between Obiwan and Anakin

$\mathbf{K}_{\mathbf{yoda}}$: Shared key between yoda and Anakin

Key-Distribution Center (with Symmetric Key-Crypto)

Enc(K_{obiwan},"I would like to talk to Yoda")

Ok, here is a fresh key that no sith lord has seen

c_1=**Enc**($\mathbf{K_{obiwan}}$,ts, $\mathbf{K_{new}}$),
c_2=**Enc**($\mathbf{K_{yoda}}$, ts, "Obiwan/Yoda", $\mathbf{K_{new}}$)

Enc($\mathbf{K_{new}}$, "Its me, Obiwan, let's talk")
c_2=**Enc**($\mathbf{K_{yoda}}$,ts, "Obiwan/Yoda", $\mathbf{K_{new}}$)

Key-Distribution Center (with Symmetric Key-Crypto)

- **Vulnerability**: If Key-Distribution Center is compromised then **all** security guarantees are broken.
  - KDC is a valuable target for attackers
  - Possibility of insider attacks (e.g., employees)

- **Denial of Service (DOS) Attack**: If KDC is down then secure communication is temporarily impossible.

Key-Distribution Center (with Symmetric Key-Crypto)

- Benefit: Authenticated Encryption provides authentication as well
  - Yoda can be sure he is talking to Obiwan (assuming he trusts the KDC)

- Kerberos uses similar protocol
  - Yoda's key and Obiwan's key are typically derived from a password that they known.
  - **Vulnerability**: An eavesdropping attacker can mount a brute-force attack on the (low-entropy) passwords to recover $\mathbf{K_{yoda}}$ and $\mathbf{K_{obiwan}}$.

- **Recommendation**: Always use Public Key Initialization with Kerberos

Key-Explosion Problem

- To avoid use a trusted KDC we could have every pair of users exchange private keys
- How many private keys per person?
  - **Answer**: n-1
  - Need to meet up with n-1 different users in person!
- Key Explosion Problem
  - n can get very big if you are Google or Amazon!

Diffie-Hellman Key Exchange

1. Alice picks x_A and sends g^{x_A} to Bob
2. Bob picks x_B and sends g^{x_B} to Alice
3. Alice and Bob can both compute $K_{A,B} = g^{x_B \, x_A}$

Key-Exchange Experiment $KE_{A,\Pi}^{eav}(n)$:

- Two parties run Π to exchange secret messages (with security parameter 1^n).
- Let **trans** be a transcript which contains all messages sent and let k be the secret key output by each party.
- Let b be a random bit and let $\mathbf{k_b}$ = k if b=0; otherwise $\mathbf{k_b}$ is sampled uniformly at random.
- Attacker A is given **trans** and $\mathbf{k_b}$ (passive attacker).
- Attacker outputs b' ($KE_{A,\Pi}^{eav}(n)$=1 if and only if b=b')

Security of Π against an eavesdropping attacker: For all PPT A there is a negligible function **negl** such that

$$\Pr\left[KE_{A,\Pi}^{eav}(n)\right]=½+\mathbf{negl}(\mathrm{n}).$$

Diffie-Hellman Key-Exchange is Secure

Theorem: If the decisional Diffie-Hellman problem is hard relative to group generator $\mathcal{G}$ then the Diffie-Hellman key-exchange protocol Π is secure in the presence of an eavesdropper (*).

(*) Assuming keys are chosen uniformly at random from the cyclic group $\mathbb{G}$

Protocol Π

1. Alice picks x_A and sends g^{x_A} to Bob
2. Bob picks x_B and sends g^{x_B} to Alice
3. Alice and Bob can both compute $K_{A,B} = g^{x_B x_A}$

Diffie-Hellman Assumptions

Computational Diffie-Hellman Problem (CDH)

- Attacker is given $\mathrm{h}_1 = g^{x_1} \in \mathbb{G}$ and $\mathrm{h}_2 = g^{x_2} \in \mathbb{G}$.
- Attackers goal is to find $g^{x_1 x_2} = (\mathrm{h}_1)^{x_2} = (\mathrm{h}_2)^{x_1}$
- **CDH Assumption**: For all PPT A there is a negligible function negl upper bounding the probability that A succeeds

Decisional Diffie-Hellman Problem (DDH)

- Let $\mathrm{z}_0 = g^{x_1 x_2}$ and let $\mathrm{z}_1 = g^{r}$, where x_1,x_2 and r are random
- Attacker is given g^{x_1}, g^{x_2} and z_b (for a random bit b)
- Attackers goal is to guess b
- **DDH Assumption**: For all PPT A there is a negligible function negl such that A succeeds with probability at most ½ + negl(n).

Diffie-Hellman Key Exchange

1. Alice picks x_A and sends g^{x_A} to Bob
2. Bob picks x_B and sends g^{x_B} to Alice
3. Alice and Bob can both compute $K_{A,B} = g^{x_B x_A}$

Intuition: Decisional Diffie-Hellman assumption implies that a passive attacker who observes g^{x_A} and g^{x_B} still cannot distinguish between $K_{A,B} = g^{x_B x_A}$ and a random group element**.**

Diffie-Hellman Key-Exchange is Secure

Theorem: If the decisional Diffie-Hellman problem is hard relative to group generator $\mathcal{G}$ then the Diffie-Hellman key-exchange protocol Π is secure in the presence of an eavesdropper (*).

Proof:

$$\Pr\left[KE_{A,\Pi}^{eav}(n) = 1\right]$$
$$= ½\Pr\left[KE_{A,\Pi}^{eav}(n) = 1 | b = 1\right] + ½\Pr\left[KE_{A,\Pi}^{eav}(n) = 1 | b = 0\right]$$
$$= ½\Pr[A(\mathbb{G}, g, q, g^x, g^y, g^{xy}) = 1] + ½\Pr[A(\mathbb{G}, g, q, g^x, g^y, g^z) = 1]$$
$$= ½ + ½(\Pr[A(\mathbb{G}, g, q, g^x, g^y, g^{xy}) = 1] - \Pr[A(\mathbb{G}, g, q, g^x, g^y, g^z) = 1]).$$
$$\leq ½ + ½\text{negl}(n) \text{ (by DDH)}$$

(*) Assuming keys are chosen uniformly at random from the cyclic group $\mathbb{G}$

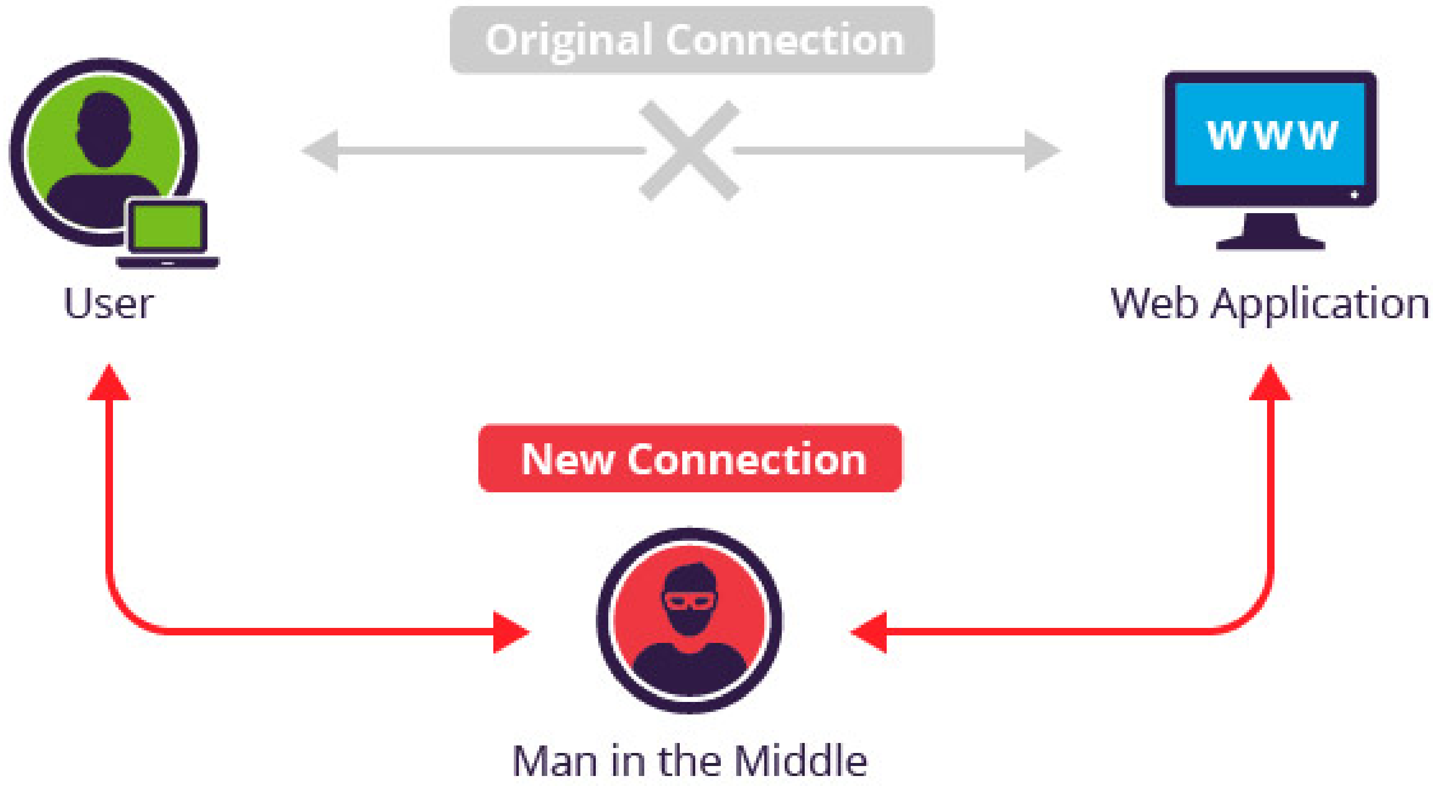
Diffie-Hellman Key Exchange

1. Alice picks x_A and sends g^{x_A} to Bob
2. Bob picks x_B and sends g^{x_B} to Alice
3. Alice and Bob can both compute $K_{A,B} = g^{x_B\, x_A}$

Intuition: Decisional Diffie-Hellman assumption implies that a passive attacker who observes g^{x_A} and g^{x_B} still cannot distinguish between $K_{A,B} = g^{x_B\, x_A}$ and a random group element**.**

Remark: The protocol is vulnerable against active attackers who can tamper with messages.

Man in the Middle Attack (MITM)



Man in the Middle Attack (MITM)

1. Alice picks x_A and sends g^{x_A} to Bob
   - Eve intercepts g^{x_A} , picks x_E and sends g^{x_E} to Bob instead
2. Bob picks x_B and sends g^{x_B} to Alice
   1. Eve intercepts g^{x_B}, picks $x_{E'}$ and sends $g^{x_{E'}}$ to Alice instead
3. Eve computes $g^{x_{E'}x_A}$ and $g^{x_E x_B}$
   1. Alice computes secret key $g^{x_{E'}x_A}$ (shared with Eve not Bob)
   2. Bob computes $g^{x_E x_B}$(shared with Eve not Alice)
4. Eve forwards messages between Alice and Bob (tampering with the messages if desired)
5. Neither Alice nor Bob can detect the attack

Password Authenticated Key-Exchange

- Suppose Alice and Bob share a low-entropy password pwd and wish to communicate securely
  - (without using any trusted party)
  - Assuming an active attacker may try to mount a man-in-the-middle attack
- Can they do it?

Tempting Approach:

- Alice and Bob both compute K= KDF(pwd)=H^n(pwd) and communicate with using an authenticated encryption scheme.
- **Midterm Exam:** Secure in random oracle model if attacker cannot query random oracle too many time.

Password Authenticated Key-Exchange

Tempting Approach:

- Alice and Bob both compute K= KDF(pwd)=H^n(pwd) and communicate with using an authenticated encryption scheme.
- **Midterm Exam:** Secure in random oracle model if attacker cannot query random oracle too many time.
- **Problems:**
  - In practice the attacker can (and will) query the random oracle many times.
  - In practice people tend to pick very weak passwords
  - Brute-force attack: Attacker enumerates over a dictionary of passwords and attempts to decrypt messages with $K_{pwd'}$=KDF(pwd') (only succeeds if $K_{pwd'}$=K).
  - An offline attack (brute-force) will almost always succeed

Password Authenticated Key-Exchange (PAKE)

Better Approach (PAKE):

1. Alice and Bob both compute $\mathrm{W} = g^{pwd}$
2. Alice picks x_A and sends "Alice", $X = g^{x_A}$ to Bob
3. Bob picks x_B computes $\mathrm{r} = \mathrm{H}(1, Alice, Bob, X)$ and $Y = (X \times (W)^r)^{x_B}$ and sends Alice the following message: "Bob," Y
4. Alice computes $\mathrm{K} = Y^Z = g^{x_B}$ where $z = 1/\left((pwd \times r) + x_A\right)\ mod\ p$. Alice sends the message V_A= H(2,Alice,Bob,X,Y,K) to Bob.
5. Bob verifies that V_A== H(2,Alice,Bob,X,Y,K) where K = g^{x_B}. Bob generates V_B= H(3,Alice,Bob,X,Y,K) and sends V_B to Alice.
6. Alice verifies that V_B==H(3,Alice,Bob,X,Y, Y^Z) where $z = 1/\left((pwd \times r) + x_A\right)$.
7. If Alice and Bob don't terminate the session key is H(4,Alice,Bob,X,Y, K)

Security:

- No offline attack (brute-force) is possible. Attacker get's one password guess per instantiation of the protocol.
- If attacker is incorrect and he tampers with messages then he will cause the Alice & Bob to quit.
- If Alice and Bob accept the secret key K and the attacker did not know/guess the password then K is "just as good" as a truly random secret key.

See RFC 6628

Key-Explosion Problem

- So far neither Diffie-Hellman Key Exchange nor PAKEs completely solved the problem

- PAKEs require a shared password
  - (n-1) shared passwords?
- Diffie-Hellman Key Exchange is vulnerable to man-in-the-middle

- Can use KDC to store database of public-keys (e.g., g^{x_A}) for each party.
  - Breached KDC doesn't reveal secret keys

Public Key Revolution

- Digital Signatures can help
  - Private-Key Analogue: MAC
  - Private Key required to produce signature for a message m
  - Anyone with Public Key can verify the message
- An authority could sign the message "Alice's public key is g^{x_A}"
- Anyone could use the authority's public key to validate Alice's public key
- The authority does not actually need to store g^{x_A}.
- In fact, if Alice has signature then she can use this to prove her identity to Bob (and Bob doesn't need to interact the authority)

Next Class: Formalizing Public Key Encryption

- Formalizing Public Key Encryption
- Read Katz and Lindell: 11.1-11.2