Applied Cryptography and Computer Security
CSE 664 Spring 2017

Lecture 18: Key Distribution and Agreement

Department of Computer Science and Engineering
University at Buffalo

Key Distribution Mechanisms

- **Secret-key encryption is much faster than public-key encryption**
  - **to have efficiency, we are to deal with distribution of the shared keys**
- **Recall that public-key cryptography can bootstrap communication with symmetric keys**
  - **suppose Alice knows Bob's public key** pk_B
  - **Alice chooses a session key** s **and sends Bob** $E_{pk_B}(s)$
  - **Bob decrypts it and now they share the same key**
  - **this simple solution can work in some cases, but has disadvantages**

Key Distribution Mechanisms

- There are many possibilities for key distribution
  - assume that we have an insecure network of n users
  - there is also a trusted authority (TA)
    - the TA's responsibilities could include checking user identities, issuing certificates, transmitting keys, etc.
- We divide all approaches in 3 categories
  - key predistribution
    - a TA distributes keying information during the setup phase using a secure channel
    - a pair of users is then able to compute a key known only to them

Key Distribution Mechanisms

- **Types of key distribution (cont.)**
  - **session key distribution**
    - **on request, an online TA chooses a session keys and distributes it to two users**
    - **the TA communicates the new keys by encrypting them using previously distributed secret keys**
    - **session keys are used for a fixed, rather short period of time**
  - **key agreement (a.k.a. key establishment or key exchange)**
    - **network users employ an interactive protocol to construct a session key**
    - **no TA's help is used**
    - **can be based on secret-key or public-key schemes**

Key Distribution Mechanisms

- **The difference between key distribution and key agreement:**
  - **in key distribution, one party (e.g., a TA) chooses a key and transmits it to one or more parties**
    - **key transmission is performed in an encrypted form**
  - **in key agreement, two or more parties jointly establish a secret key**
    - **communication is performed over a public channel**
    - **each participant contributes to the value of the resulting key**
    - **the key is not sent from one party to another**

Key Distribution Mechanisms

- **In the network, users may have long-lived keys**
  - **they can be precomputed and stored securely**
  - **they could be secret keys known to a pair of users or to a user and the TA**
  - **they also could be private keys corresponding to public keys stored in users' certificates**
- **Pairs of users often employ short-lived session keys**
  - **a session key is used for a particular session and is discarded at the end of it**
  - **session keys are normally secret keys for a symmetric encryption scheme or MAC**

Key Distribution Mechanisms

- **Since the network is insecure, we need to protect against attackers**
  - **the adversary might be one of the users in the network**
- **An active adversary can:**
  - **modify messages being transmitted on the network**
  - **save messages for later use**
  - **try to masquerade as another user in the network**
- **Adversary's goal might be:**
  - **fool someone into accepting an invalid key as valid**
  - **learn some information about the key being established**
  - **use another user's identity to establish a shared key with someone**

Key Distribution Mechanisms

- **In real life applications, the adversary can have even more power**
  - **suppose that a session key has been exposed**
    - **we prefer to see no impact on the security of the long-lived key**
  - **suppose that an attacker gets ahold of your long-lived key**
    - **ideally this should not compromise the security of past session keys**
    - **this property is called perfect forward secrecy**
- **Often we also want parties to authenticate during the key agreement protocol**
  - **this is called authenticated key exchange**

Diffie-Hellman Key Predistribution

- The following key predistribution scheme is a modification of the Diffie-Hellman key exchange protocol
  - its security is based on the hardness of the Decision Diffie-Hellman (DDH) problem
- The setup
  - the public domain parameters consist of a group $(G, \cdot)$ and an element $g \in G$ of some order q
  - every user U in the network has a long-lived private key x_U $(0 < x_U \leq q - 1)$ and the corresponding public key $y_U = g^{x_U}$
  - the users' public keys are certified (signed) by the TA to guarantee their authenticity

Diffie-Hellman Key Predistribution

- **Diffie-Hellman key predistribution**
  - A **and** B **would like to setup a joint key**
  - A **computes the key** $k_{A,B}$ **using** B**'s (signed) public key** y_B **and** A**'s private key** x_A**:**

$$k_{A,B} = y_B^{x_A} = g^{x_A x_B}$$

  - **likewise,** B**, using** A**'s (signed) public key** y_A **and** B**'s private key** x_B**, computes:**

$$k_{A,B} = y_A^{x_B} = g^{x_A x_B}$$

- **Each pair of users performs the same computation to obtain the key known only to them**

Diffie-Hellman Key Predistribution

- **Hardness assumptions**
  - **Computational DH: given g, g^a and g^b, it is hard to compute g^{ab}**
  - **Decision DH: given g, g^a, g^b, and g^c, it is hard to decide whether $g^c = g^{ab}$**
- **Security of DH key predistribution**
  - **since there is no interaction, an active adversary cannot do much**
  - **if CDH problem is hard, recovery of any key $k_{U,V}$ is infeasible**
  - **if DDH problem is hard, the keys are indistinguishable from random**

Session Key Distribution Schemes

- Assume that the TA has a shared key with each user on the network
  - k_A is the key shared with Alice, k_B is the key shared with Bob, etc.
- The TA chooses session keys and distributes them in encrypted form upon user requests
- How do we do this?
  - the simplest solution is for Alice to send a session key request for users A, B
  - the TA chooses a key k at random and sends $E_{k_A}(k||B)$ to Alice and $E_{k_B}(k||A)$ to Bob
  - each of them decrypt and start communicating using k
  - is this enough?

Session Key Distribution Schemes

- Needham-Schroeder SKDS was designed in 1978
  - uses fresh nonces, but stil doesn't provide adequate security
- Denning and Sacco discovered an attack on Needham-Schroeder SKDS
  - it is called known session key attack because it assumes the attacker obtains one of the past session keys k
- Kerberos is a series of related SKDSs developed at MIT in the 80-90s
  - it additionally uses validity period in security tokens
  - this limits the time period during which a Denning-Sacco type of attack can be carried out
- Neither solution has a security proof and both have security weaknesses

Bellare-Rogaway SKDS

- **Bellare and Rogaway proposed an SKDS in 1995 that has a proof of security**
  - **it has a different flow structure than the earlier schemes**
- **Bellare-Rogaway SKDS**
  - **Alice chooses random r_A and sends A, B, and r_A to Bob**
  - **Bob chooses random r_B and sends A, B, r_A, and r_B to the TA**
  - **the TA chooses a random session key k and computes $y_B = (E_{k_B}(k), MAC_B(A||B||r_B||E_{k_B}(k)))$ and $y_A = (E_{k_A}(k), MAC_A(B||A||r_A||E_{k_A}(k)))$**
  - **the TA sends y_B to Bob and y_A to Alice**

Bellare-Rogaway SKDS

- Alice and Bob need to verify that the messages have a correct form, the MAC is valid, and the proper values r_A and r_B were used
- No explicit key confirmation is provided
  - if Alice accepts, she believes that she has received a new session key from the TA
  - she doesn't know if Bob received everything as well, but she is confident that noone other than Bob can compute the session key
- We arrive at (informal) definition of a secure session key distribution scheme
  - if a protocol participant "accepts," then the probability that someone other than the intended peer knows the session key is negligible

Bellare-Rogaway SKDS

- **To show security, we make certain assumptions**
  - **Alice and Bob are honest**
  - r_A, r_B, **and** k **are chosen perfectly at random**
  - **the encryption scheme and MAC are secure**
  - **secret keys are known only to their intended owners**
- **Possibilities for an adversary**
  - **Mallory is a passive adversary**
  - **Mallory is an active adversary**
    - **she may impersonate Alice, Bob, or the TA; intercept and modify messages**

Bellare-Rogaway SKDS

- If Mallory is passive, Alice and Bob compute the same key and accept
  - Mallory cannot compute the key because encryption is secure
- Now assume that Alice is a legitimate user and Mallory is active
  - Alice doesn't know if she is really communicating with Bob or the TA
  - when Alice receives y_A, she checks that the MAC contains her r_A, the identities are A and B
    - this convinces her that the response is fresh and came from the TA
    - using r_A prevents replay attacks
    - also, including $E_{k_A}(k)$ under the MAC prevents its replacement by the attacker
- Similar reasoning applies to Bob's side

Key Distribution and Agreement

- **Recall that setting up a shared key between two users can be done by**
  - **predistributing keys to them**
  - **using a session key distribution scheme**
  - **engaging them in a key agreement protocol**
- **We next cover key agreement (or key exchange) schemes**
  - **a key exchange is an interactive protocol between two users without active participation of a TA**
  - **this is achieves by means of public-key cryptography**

Key Agreement Schemes

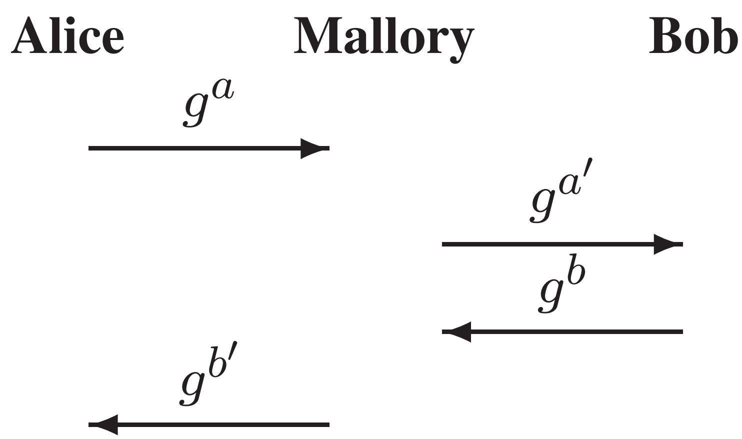
- **The best-known key exchange protocol is due to Diffie and Hellman**
  - **recall that Alice and Bob want to establish a shared key**
    - **the common parameters are** (G, q, g)
    - **Alice chooses a random number** a **from** $\mathbb{Z}_q$**, computes** g^a**, and sends** g^a **to Bob**
    - **Bob chooses a random number** b **from** $\mathbb{Z}_q$**, computes** g^b**, and sends** g^b **to Alice**
    - **Alice computes the shared key as** $(g^b)^a = g^{ab}$
    - **Bob computes the shared key as** $(g^a)^b = g^{ab}$

Diffie-Hellman Key Exchange

- **Diffie-Hellman key exchange**
  - Alice and Bob compute the same key, but it is computationally difficult for someone else to compute their key
  - the security property holds only against a passive attacker
  - the protocol has a serious weakness in the presence of an active adversary
    - this is called a **man-in-the-middle attack**
    - Mallory will intercept messages between Alice and Bob and substitute her own
    - Alice establishes a shared key with Mallory and Bob also establishes a shared key with Mallory

Diffie-Hellman Key Exchange

- **Man-in-the-middle attack on Diffie-Hellman key exchange**



  - **Alice shares the key $g^{ab'}$ with Mallory**
  - **Bob shares the key $g^{a'b}$ with Mallory**
  - **Alice and Bob do not share any key**
  - **what is Mallory capable of doing?**

Diffie-Hellman Key Exchange

- **Alice and Bob need to make sure they are exchanging messages with each other**
  - **there is a need for authentication**
  - **preceding this protocol with an authentication scheme is not guaranteed to solve the problem**
    - **after they authenticate, the same attack can be carried out**
- **We need a protocol that authenticates the participants at the same time the key is being established**
  - **such a protocol is called an authenticated key agreement scheme**
  - **it should simultaneously guarantee secure mutual authentication and secure key computation**

Diffie-Hellman Key Exchange

- **Authenticated Diffie-Hellman key exchange**
  - **each user U has a private signing key sk_U and the corresponding public verification key pk_U**
  - **there is a trusted authority TA that signs keys**
  - **user U holds a certificate $\mathsf{cert}(U)$ issued by the TA**

$$\mathsf{cert}(U) = (U, pk_U, \sigma_{TA}(U, pk_U))$$

  - **the protocol is also known as station-to-station key agreement**
  - **it combines the key exchange with a mutual authentication scheme**

Diffie-Hellman Key Exchange

- **Authenticated Diffie-Hellman key exchange (simplified)**
  - **public parameters are as before** (G, q, g)
  - **Alice chooses random** a**, computes** $x_A = g^a$**, and sends** $\mathsf{cert}(A)$ **and** x_A **to Bob**
  - **Bob chooses random** b**, computes**

    $$x_B = g^b,\ k = (x_A)^b = g^{ab} \text{, and } \ y_B = \sigma_B(A||x_B||x_A)$$

    and sends $\mathsf{cert(B)}$**,** x_B**, and** y_B **to Alice**
  - **Alice verifies** y_B**; if the signature is valid, she computes**

    $$k = (x_B)^a = g^{ab} \text{ and } \ y_A = \sigma_A(B||x_A||x_B)$$

    and sends y_A **to Bob**
  - **Bob verifies** y_A**; if the signature is valid, he accepts**

Diffie-Hellman Key Exchange

- **Security of authenticated Diffie-Hellman**
  - **the man-in-the-middle attack on DH key exchange no longer works**
  - **what happens now is:**

Alice		**Mallory**		**Bob**
	g^a →		$g^{a'}$ →	
	← $g^{b'}, \sigma_B(A\|\|g^{b'}\|\|g^a)$		← $g^b, \sigma_B(A\|\|g^b\|\|g^{a'})$	
	$\sigma_A(B\|\|g^a\|\|g^{b'})$ →		$\sigma_A(B\|\|g^{a'}\|\|g^b)$ →	

  - **Mallory cannot forge Alice's and Bob's signature, so she cannot be successful**

Diffie-Hellman Key Exchange

- **Security of authenticated Diffie-Hellman**
  - this protocol is a secure mutual identification scheme
    - this can be proven using the security definitions for mutual authentication
  - if an adversary is active, this will be detected by the participants
  - if the adversary is passive, both parties will accept with the same key
    - the adversary cannot compute any information about the key assuming that the DDH problem is hard

Diffie-Hellman Key Exchange

- Let's look at the level of assurance Alice and Bob receive
  - Alice accepts after sending g^a and receiving $\sigma_B(A||g^b||g^a)$ back
    - Alice is confident that she is really communicating with Bob
    - if Bob followed the instructions, he will be able to compute the key
    - Alice is confident that Bob can compute g^{ab} because g^a and g^b were in Bob's signature
  - Bob accepts after sending $\sigma_B(A||g^b||g^a)$ to Alice and receiving $\sigma_A(B||g^a||g^b)$ back
    - the analysis is similar for Bob, except that he knows that Alice already accepted
  - when Alice accepts, she doesn't know whether Bob will accept

Key Agreement Schemes

- **We can define different levels of assurance that Alice (or Bob) obtain during a key exchange protocol**
  - **implicit key authentication is provided if A is assured that noone other than B can compute the key**
  - **implicit key confirmation is provided if A is assured that B can compute the key and noone else can**
  - **explicit key confirmation is provided if A is assured that B computed the key and noone else can compute it**
- **Authenticated Diffie-Hellman provides implicit key confirmation to both parties**
- **Kerberos and Needham-Schroeder provide explicit key confirmation**

Key Agreement Schemes

- **We might want to consider possible influence that different sessions can have on each other in real life usage**
- **We'll next look at security under a known session key attack**
  - **Mallory observes several sessions with different users (which can involve Mallory as well) of her choice**
  - **Mallory is able to compromise session keys associated with some of the observed sessions of her choice**
  - **Mallory is then asked to recover the key for a challenge session**

Key Agreement Schemes

- **Consider the authenticated Diffie-Hellman protocol**
  - **Mallory observes values g^a and g^b (and signatures)**
  - **Mallory is also allowed to ask for $k = g^{ab}$**
  - **we allow Mallory to ask for a key even if she cheats in a protocol**
    - **suppose Mallory is engaging in a key exchange with Bob**
    - **Mallory picks a random h sends it to Bob (i.e., $h = g^x$ s.t. Mallory doesn't know x)**
    - **Bob sends g^b back (and they send signatures)**
    - **Mallory is still allowed to ask for the key $k = h^b$**

Key Agreement Schemes

- **Known session key attack on authenticated Diffie-Hellman**
  - this key exchange protocol is **secure against the known session key attack**
  - intuition:
    - the values g^a, g^b are chosen anew for each session
    - they are not related to previous sessions or the long-term keys of the participants
  - it is computationally infeasible, given g^a and g^b, to compute any information about g^{ab}

Key Agreement Schemes

- **Perfect forward secrecy**
  - this property means that compromise of long-term key does not compromise past session keys
  - suppose Mallory records sessions between Alice and Bob and somehow gets ahold of Alice's secret signing key
  - this property requires that Mallory cannot recover session keys for Alice's expired session
    - an expired session is a session for which Alice erased all information used to generate the session key k
    - what is this information in authenticated Diffie-Hellman?

Key Agreement Schemes

- **Perfect forward secrecy (cont.)**
  - **where do we stand with respect to authenticated Diffie-Hellman key exchange?**
  - **in authenticated Diffie-Hellman protocol, session keys are independent of long-term keys**
  - **it achieves perfect forward secrecy**
- **We arrive at the following conclusion:**
  - **authenticated Diffie-Hellman key agreement scheme is an authenticated key agreement scheme secure against known session key attacks and achieving perfect forward secrecy**
  - **now this is the standard security requirement for key exchange protocols**

Key Agreement Schemes

- There are different versions of authenticated DH key exchange
- We'll study SIGMA next
  - SIGMA is signature-based authenticated key exchange
  - it stands for SIGn-and-MAc
  - it has been formally analyzed and proven secure
  - it has been standardized as the main protocol in Internet Key Exchange (IKE) version 1 and 2 (RFCs 2409 and 4306, respectively)
- As before, assume that Alice and Bob want to agree on a session key
- Each of them hold a private signing and a public verification key

SIGMA Key Exchange

- **SIGMA key exchange**

Alice $\qquad\qquad\qquad\qquad$ **Bob**

$$\xrightarrow{\quad g^a \quad}$$

$$\xleftarrow{\quad g^b \quad}$$

$$\xrightarrow{A,\ \sigma_A(g^b,\ g^a),\ \mathsf{MAC}_{K_m}(0||A)}$$

$$\xleftarrow{B,\ \sigma_B(g^a,\ g^b),\ \mathsf{MAC}_{K_m}(1||B)}$$

  - **here $K_m = h(g^{ab})$ is a hash of g^{ab}**
  - **the sender includes 0 in the MAC, and the responder includes 1**
  - **the purpose of the MAC is to prevent the identity misbinding attack**
  - **also notice that the identity of the peer is never signed**

SIGMA Key Exchange

- **There is a 3-message variant of the protocol**
  - **the 4-message SIGMA is called SIGMA-R and the 3-message variant is called SIGMA-I**
  - **SIGMA-I can be obtained by reverting the order of the 3rd and 4th messages**

Alice **Bob**

$$g^a \quad \longrightarrow$$

$$\longleftarrow \quad g^b,\ B,\ \sigma_B(g^a,\ g^b),\ \mathsf{MAC}_{K_m}(1||B)$$

$$A,\ \sigma_A(g^b,\ g^a),\ \mathsf{MAC}_{K_m}(0||A) \quad \longrightarrow$$

  - **this has advantage of identity protection if the last two messages are encrypted**
    - g^a **and** g^b **are then used to compute such an encryption key**

Key Agreement Schemes

- **Another rather new standardized key exchange protocol is SKEME**
  - **it is based on public-key encryption instead of signatures**
  - **it also uses MAC**
  - **it was introduced because of its deniability property**
- **Deniability provides a way to deny participation in a key exchange (and the consecutive encrypted conversation)**
  - **authenticated Diffie-Hellman is not deniable**
  - **SIGMA provides limited deniability**
  - **SKEME is fully deniable**

Key Agreement Schemes

- **All protocols so far relied on the use of public keys and certificates**
- **What happens if there is no public-key infrastructure and instead two users share a password?**
  - **a password can often be shared between a user and a server**
  - **the password is likely to be too short to be used as a good cryptographic key**
- **How can we establish a session key then?**
  - **one suggestion is to encrypt the session key with the password**
  - **i.e., Alice chooses a new key k and sends $\mathsf{Enc}_{pwd}(k)$ to Bob**
  - **Bob decrypts and they start sending messages encrypted with k**

Key Agreement Schemes

- **Password-based key establishment**
  - unfortunately, since the password is short, Mallory can try all possibilities
  - Mallory saves $x = \mathsf{Enc}_{pwd}(k)$ and $y = \mathsf{Enc}_k(m)$
  - she computes $k' = \mathsf{Dec}_{pwd}(x)$ and $m' = \mathsf{Dec}_{k'}(y)$ for each possible password pwd
  - since m normally contains redundancy, Mallory will be able to tell when a match is found
  - Mallory now can impersonate the user or read all communication
- It is still possible to securely encrypt data during the key agreement
  - such schemes are called **Encrypted Key Exchange** (EKE)

Key Agreement Schemes

- **We'll look at the simplified Bellovin-Merritt protocol obtained from DH key exchange**
- **Bellovin-Merritt EKE2**
  - **public parameters consist of a group G and element $g \in G$**
  - **Alice and Bob share a secret password pwd**
  - **Alice picks a and Bob picks b, and the session key is $k = g^{ab}$**
  - **the difference from previous solutions is that values g^a and g^b are encrypted using the password during the transmission**

Bellovin-Merritt EKE

- **Bellovin-Merritt EKE2**

Alice		**Bob**
choose a	$A,\ \mathsf{Enc}_{pwd}(g^a)$ $\longrightarrow$	**choose** b
	$B,\ \mathsf{Enc}_{pwd}(g^b)$ $\longleftarrow$	

  - **each of them decrypt the messages received and compute the shared key** $k = g^{ab}$
  - **authentication is not used, but encryption prevents an adversary from carrying out a successful attack**
    - **Alice knows that knowledge of** g^a **is required to construct the key**
    - **the only person who knows the decryption key is Bob**

Bellovin-Merritt EKE

- **Bellovin-Merritt EKE2**
  - the above analysis assumes that the password is not known to other parties
  - it is also assumed that an adversary cannot compute any information about the password
  - consider the previous brute force search attack
    - before attacker could test all possible passwords because he would know when a match occurred
    - now the password is used to encrypt g^a and g^b, while a different value g^{ab} is used for encryption of messages themselves
  - even if the value of a past session key is known to the attacker, the password remains secure

Summary

- There are many key exchange protocols, many of which are based off of the Diffie-Hellman key exchange
- The properties that are essential
  - secure mutual authentication
  - secure key computation
  - resilience to known session key attack
  - perfect forward secrecy
- Deniability can be important as well