# Key Establishment

Dhananjoy Dey

Indian Institute of Information Technology, Lucknow
ddey@iiitl.ac.in

December 23, 2022

# Disclaimers

1

All the pictures used in this presentation are taken from freely available websites.

2

If there is a reference on a slide all of the information on that slide is attributable to that source whether quotation marks are used or not.

# Outline

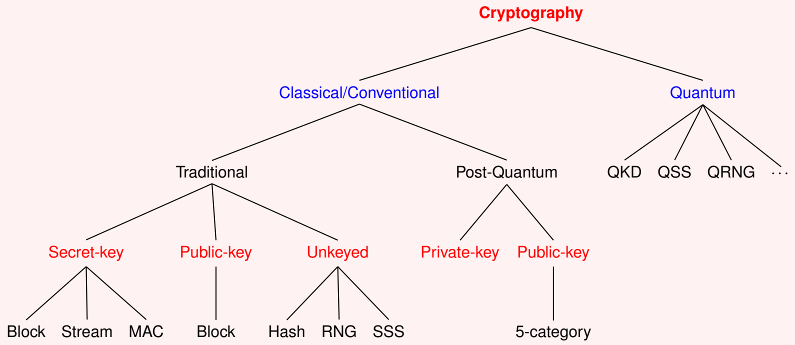
1. Introduction
2. Classification and Framework
3. Key Establishment Based on Symmetric Encryption
4. Key Establishment Based on Asymmetric Encryption
5. Secret Sharing

# Outline

1. Introduction
2. Classification and Framework
3. Key Establishment Based on Symmetric Encryption
4. Key Establishment Based on Asymmetric Encryption
5. Secret Sharing

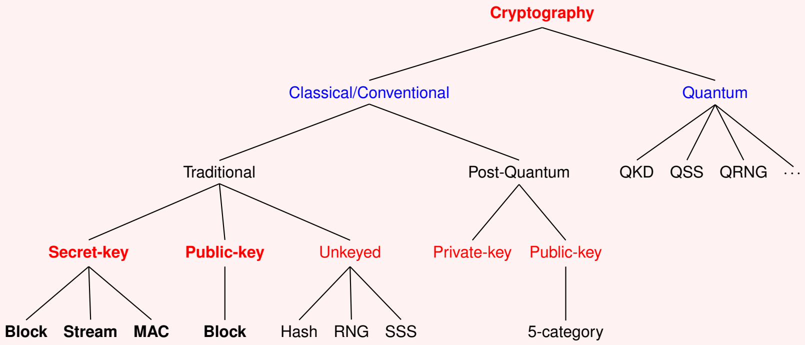
# Classification of Cryptography

## Classification



# Classification of Cryptography

## Classification



# Principle

The philosophy of modern cryptanalysis is embodied in the following principle formulated in 1883

- Basic assumptions:
  - The system is completely known to the attacker
  - Only the key is secret
  - That is, crypto algorithms are not secret
- This is known as *Kerckhoffs' Principle*

- Why do we make this assumption?
  - Easier to maintain secrecy of a short key rather than an algorithm
    - Algorithm parts may be leaked: insider or reverse engineering.
  - Key revocation/reissue is easier than algorithm revocation/reissue
  - Different people communication: different keys or different algorithms?

- All cryptographic mechanisms that we have learnt so far assume that keys are properly distributed between the parties involved.

- All cryptographic mechanisms that we have learnt so far assume that keys are properly distributed between the parties involved.
- The task of key establishment is in practice one of the most important and often also most difficult parts of a security system.
- We already learned some ways of **distributing keys**, in particular Diffie-Hellman key exchange.

# Outline

1 Introduction

2 Classification and Framework

3 Key Establishment Based on Symmetric Encryption

4 Key Establishment Based on Asymmetric Encryption

5 Secret Sharing

# Fundamental Concepts

## Key Establishment

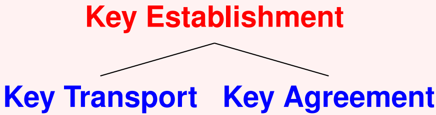
Key establishment is a **process** or **protocol** whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.

# Fundamental Concepts

## Key Establishment

Key establishment is a **process** or **protocol** whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.

## Key Establishment



**Key Establishment**

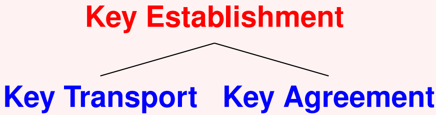
**Key Transport** **Key Agreement**

# Fundamental Concepts

## Key Establishment

Key establishment is a **process** or **protocol** whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.

## Key Establishment



- **Key Transport:** One party generates and distributes a secret key
- **Key Agreement:** Parties jointly generate a secret key

A **protocol** is a multi-party algorithm, defined by a sequence of steps precisely specifying the actions required of two or more parties in order to achieve a specified objective

# Fundamental Concepts

- Key establishment protocol itself is strongly related to **entity authentication/identification**.
- Key establishment protocols involving authentication typically require a set-up phase whereby authentic and possibly secret initial keying material is distributed.

# Fundamental Concepts

- Key establishment protocol itself is strongly related to **entity authentication/identification**.
- Key establishment protocols involving authentication typically require a set-up phase whereby authentic and possibly secret initial keying material is distributed.
- One may think of attacks by unauthorized users who join the key establishment protocol with the aim of masquerading as either Alice or Bob with the goal of establishing a secret key with the other party.
- To prevent such attacks, each party must be assured of the identity of the other entity.

# Fundamental Concepts

- **Key pre-distribution** schemes are key establishment protocols whereby the resulting established keys are completely determined a priori by initial keying material.

# Fundamental Concepts

- **Key pre-distribution** schemes are key establishment protocols whereby the resulting established keys are completely determined a priori by initial keying material.
- **Dynamic key establishment** schemes are those whereby the key established by a fixed pair (or group) of users varies on subsequent executions.

# Fundamental Concepts

- **Key pre-distribution** schemes are key establishment protocols whereby the resulting established keys are completely determined a priori by initial keying material.
- **Dynamic key establishment** schemes are those whereby the key established by a fixed pair (or group) of users varies on subsequent executions.
  - Dynamic key establishment is also referred to as **session key establishment**.
  - In this case the session keys are dynamic, and it is usually intended that the protocols are immune to known-key attacks.

# Trusted Third Party

- Many key establishment protocols involve a **centralized** or **trusted party**, for either or both initial system setup and on-line actions.

# Trusted Third Party

- Many key establishment protocols involve a **centralized** or **trusted party**, for either or both initial system setup and on-line actions.
- This party is referred as
  - trusted third party,
  - trusted server,
  - authentication server,
  - key distribution center (KDC),
  - key translation center (KTC), and
  - certification authority (CA).

# Fundamental Concepts

- **Key authentication** is the property whereby one party is assured that no other party aside from a specifically identified second party may gain access to a particular secret key.
- **Key confirmation** is the property whereby one party is assured that a second party actually has possession of a particular secret key.
    - The focus in key authentication is the *identity of the second party* rather than *the value of the key*, whereas in key confirmation the opposite is true.

# Fundamental Concepts

- **Key authentication** is the property whereby one party is assured that no other party aside from a specifically identified second party may gain access to a particular secret key.
- **Key confirmation** is the property whereby one party is assured that a second party actually has possession of a particular secret key.
    - The focus in key authentication is the *identity of the second party* rather than *the value of the key*, whereas in key confirmation the opposite is true.
- **Explicit key authentication** is the property obtained when both *key authentication* and *key confirmation* hold.

# Fundamental Concepts

- An **authenticated key establishment protocol** is a key establishment protocol which provides key authentication.
- A key establishment protocol is said to be **identity-based** if identity information of the party involved is used as the party's public key.

# Fundamental Concepts

- An **authenticated key establishment protocol** is a key establishment protocol which provides key authentication.
- A key establishment protocol is said to be **identity-based** if identity information of the party involved is used as the party's public key.
- **Objectives:**
  - **Authentication protocol** – to provide to one party some degree of assurance regarding the identity of another with which it is purportedly communicating;
  - **key establishment protocol** – to establish a shared secret;
  - **authenticated key establishment protocol** – to establish a shared secret with a party whose identity has been (or can be) corroborated.

# Motivation for Use of Session Keys

- Key establishment protocols result in shared secrets which is used to derive **session keys** or **ephemeral keys**.
- Motivation for ephemeral keys:

# Motivation for Use of Session Keys

- Key establishment protocols result in shared secrets which is used to derive **session keys** or **ephemeral keys**.
- Motivation for ephemeral keys:
    - to limit available ciphertext (under a fixed key) for cryptanalytic attack;
    - to limit exposure, with respect to both time period and quantity of data, in the event of (session) key compromise;
    - to avoid long-term storage of a large number of distinct secret keys by creating keys only when actually required;
    - to create independence across communications sessions or applications.

# Key Freshness and Key Derivation

- A key is fresh if it can be guaranteed to be new, as opposed to possibly an old key being reused through actions of either an adversary or authorized party.
- In key transport protocols, one party chooses a key value; whereas in key agreement, the key is derived from joint information, and it may be desirable that neither party be able to control or predict the value of the key.

# Key Freshness and Key Derivation

- A key is fresh if it can be guaranteed to be new, as opposed to possibly an old key being reused through actions of either an adversary or authorized party.
- In key transport protocols, one party chooses a key value; whereas in key agreement, the key is derived from joint information, and it may be desirable that neither party be able to control or predict the value of the key.
- How can key updates be realized?

# Key Freshness and Key Derivation

- A key is fresh if it can be guaranteed to be new, as opposed to possibly an old key being reused through actions of either an adversary or authorized party.
- In key transport protocols, one party chooses a key value; whereas in key agreement, the key is derived from joint information, and it may be desirable that neither party be able to control or predict the value of the key.
- How can key updates be realized?
  - Execute the key establishment protocols over and over again.

# Key Freshness and Key Derivation

- A key is fresh if it can be guaranteed to be new, as opposed to possibly an old key being reused through actions of either an adversary or authorized party.
- In key transport protocols, one party chooses a key value; whereas in key agreement, the key is derived from joint information, and it may be desirable that neither party be able to control or predict the value of the key.
- How can key updates be realized?
    - Execute the key establishment protocols over and over again.However, there are always certain costs associated with key establishment, typically wrt additional communication and computations.

# Key Freshness and Key Derivation

- A key is fresh if it can be guaranteed to be new, as opposed to possibly an old key being reused through actions of either an adversary or authorized party.
- In key transport protocols, one party chooses a key value; whereas in key agreement, the key is derived from joint information, and it may be desirable that neither party be able to control or predict the value of the key.
- How can key updates be realized?
  - Execute the key establishment protocols over and over again.However, there are always certain costs associated with key establishment, typically wrt additional communication and computations.
  - Use a key derivation function (KDF)

# The $n^2$ Key Distribution Problem

- Assume a network having $n$ users, where every party is capable of communicating with every other one in a secure fashion.

# The $n^2$ Key Distribution Problem

- Assume a network having $n$ users, where every party is capable of communicating with every other one in a secure fashion.
  - Each user must store $n - 1$ keys.
  - There is a total of $n(n - 1) \approx n^2$ keys in the network.
  - A total of $\binom{n}{2} = \frac{n(n-1)}{2}$ symmetric key pairs are in the network.
  - If a new user joins the network, a secure channel must be established with every other user in order to upload new keys.

# The $n^2$ Key Distribution Problem

- Assume a network having $n$ users, where every party is capable of communicating with every other one in a secure fashion.
    - Each user must store $n - 1$ keys.
    - There is a total of $n(n - 1) \approx n^2$ keys in the network.
    - A total of $\binom{n}{2} = \frac{n(n-1)}{2}$ symmetric key pairs are in the network.
    - If a new user joins the network, a secure channel must be established with every other user in order to upload new keys.
- The consequences of these observations are not very favorable if the number of users increases.
- All these keys must be generated securely at one location, which is typically some type of trusted authority.

# Adversaries in Key Establishment Protocols

- Communicating parties or entities in key establishment protocols are formally called principals.
- In addition to legitimate parties, the presence of an unauthorized 'third' party is hypothesized, which is given many names as:
  - adversary,
  - intruder,
  - opponent,
  - enemy,
  - attacker,
  - eavesdropper, or
  - impersonator.

# Adversaries in Key Establishment Protocols

- When examining the security of protocols, it is assumed that the underlying cryptographic mechanisms used are secure.
- Otherwise, there is no hope of a secure protocol.
- An adversary is hypothesized to be **not a cryptanalyst** attacking the underlying mechanisms directly, but rather one attempting to subvert the protocol objectives by defeating the manner in which such mechanisms are combined, i.e., attacking the protocol itself.

# Attacks in Key Establishment Protocols

- A **passive attack** involves an adversary who attempts to defeat a cryptographic technique by simply recording data and thereafter analyzing it (e.g., in key establishment, to determine the session key).
- An **active attack** involves an adversary who modifies or injects messages.

# Attacks in Key Establishment Protocols

- An adversary in a key establishment protocol may pursue many strategies, including attempting to:
  1. deduce a session key using information gained by eavesdropping;
  2. participate covertly in a protocol initiated by one party with another, and influence it, e.g., by altering messages so as to be able to deduce the key;
  3. initiate one or more protocol executions, and combine messages from one with another, so as to masquerade as some party or carry out one of the above attacks;
  4. without being able to deduce the session key itself, deceive a legitimate party regarding the identity of the party with which it shares a key

# Attacks in Key Establishment Protocols

- In analyzing key establishment protocols, the potential impact of compromise of various types of keying material should be considered, even if such compromise is not normally expected. In particular, the effect of the following is often considered:

  1. compromise of long-term secret (symmetric or asymmetric) keys, if any;
  2. compromise of past session keys.

# Attacks in Key Establishment Protocols

- A protocol is said to have **perfect forward secrecy/break-backward protection** if compromise of long-term keys does not compromise past session keys.

# Attacks in Key Establishment Protocols

- A protocol is said to have **perfect forward secrecy/break-backward protection** if compromise of long-term keys does not compromise past session keys.

- A protocol is said to be vulnerable to a **known-key attack** if compromise of past session keys allows either a passive adversary to compromise future session keys, or impersonation by an active adversary in the future

# Outline

1 Introduction

2 Classification and Framework

3 Key Establishment Based on Symmetric Encryption

4 Key Establishment Based on Asymmetric Encryption

5 Secret Sharing

# Basic Protocol

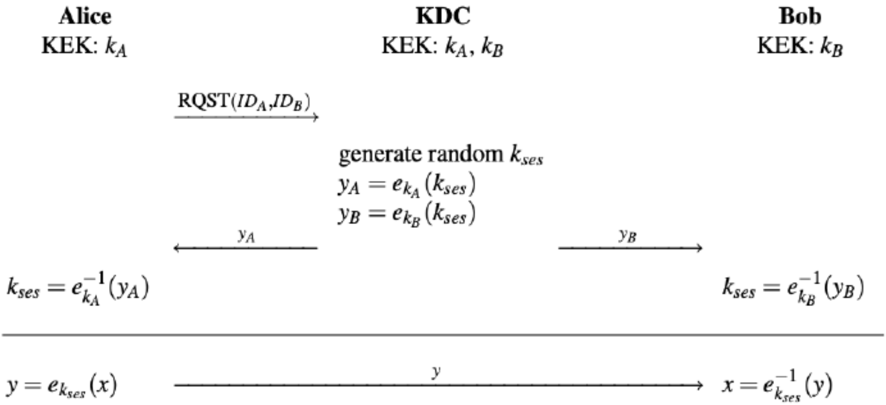
- It developed based on a **Key Distribution Center** (KDC).
- This is a server that is fully trusted by all users and that shares a secret key with each user.
- **Key Encryption Key** (KEK), is used to securely transmit session keys to users.

## Basic Protocol

- A **necessary prerequisite** is that each user $U$ shares a unique secret key KEK $k_U$ with the KDC which predistributed through a secure channel.

# Basic Protocol

- A **necessary prerequisite** is that each user $U$ shares a unique secret key KEK $k_U$ with the KDC which predistributed through a secure channel.



**Alice**
KEK: $k_A$

**KDC**
KEK: $k_A, k_B$

**Bob**
KEK: $k_B$

$\xrightarrow{\text{RQST}(ID_A, ID_B)}$

generate random $k_{ses}$
$y_A = e_{k_A}(k_{ses})$
$y_B = e_{k_B}(k_{ses})$

$\xleftarrow{y_A}$ $\xrightarrow{y_B}$

$k_{ses} = e_{k_A}^{-1}(y_A)$

$k_{ses} = e_{k_B}^{-1}(y_B)$

$y = e_{k_{ses}}(x)$ $\xrightarrow{y}$ $x = e_{k_{ses}}^{-1}(y)$

# Basic Protocol

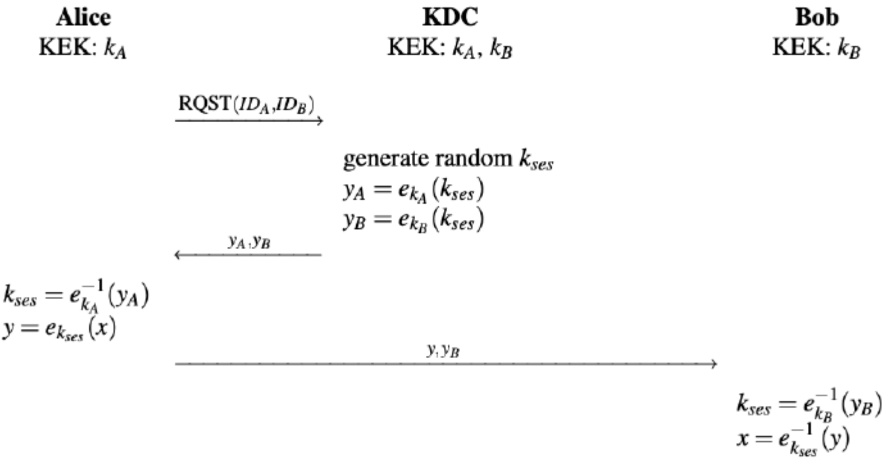
- It is important to note that **two types of keys** are involved in the protocol.
- The KEKs $k_A$ and $k_B$ are **long-term keys** that do not change.
- The session key $k_{ses}$ is an **ephemeral key** that changes frequently.
- Since the KEKs are long-term keys, whereas the session keys have typically a much shorter lifetime, in practice sometimes **different encryption algorithms** are used with both.

# Basic Protocol

- It is easy to modify the above protocol s/t we save one communication session.

# Basic Protocol

- It is easy to modify the above protocol s/t we save one communication session.



**Alice**
KEK: $k_A$

**KDC**
KEK: $k_A$, $k_B$

**Bob**
KEK: $k_B$

$\xrightarrow{\mathrm{RQST}(ID_A, ID_B)}$

generate random $k_{ses}$
$y_A = e_{k_A}(k_{ses})$
$y_B = e_{k_B}(k_{ses})$

$\xleftarrow{y_A, y_B}$

$k_{ses} = e^{-1}_{k_A}(y_A)$
$y = e_{k_{ses}}(x)$

$\xrightarrow{y, y_B}$

$k_{ses} = e^{-1}_{k_B}(y_B)$
$x = e^{-1}_{k_{ses}}(y)$

# Basic Protocol

- Both of the KDC-based protocols have the advantage.
- There are only $n$ long-term symmetric key pairs in the system.
- The first naive scheme that we discussed, where $\approx \frac{n^2}{2}$ key pairs were required.
- The $n$ long-term KEKs only need to be stored by the KDC, while each user only stores his/her own KEK.
- Most importantly, if a new user $N$ joins the network, a secure channel only needs to be established once between the KDC and $N$ to distribute the KEK $k_N$.

# Analysis of Basic Protocol

- The two protocols protect against a **passive attacker**.

# Analysis of Basic Protocol

- The two protocols protect against a **passive attacker**.
- **Replay Attack**
  - This attack makes use of the fact that neither Alice nor Bob know whether the encrypted session key they receive is actually a new one.
  - If an old one is reused, key freshness is violated.
  - This can be a particularly serious issue if an old session key has become compromised.
  - This could happen if an old key is leaked, e.g., through a hacker, or if the encryption algorithm used with an old key has become insecure due to cryptanalytical advances.

# Analysis of Basic Protocol

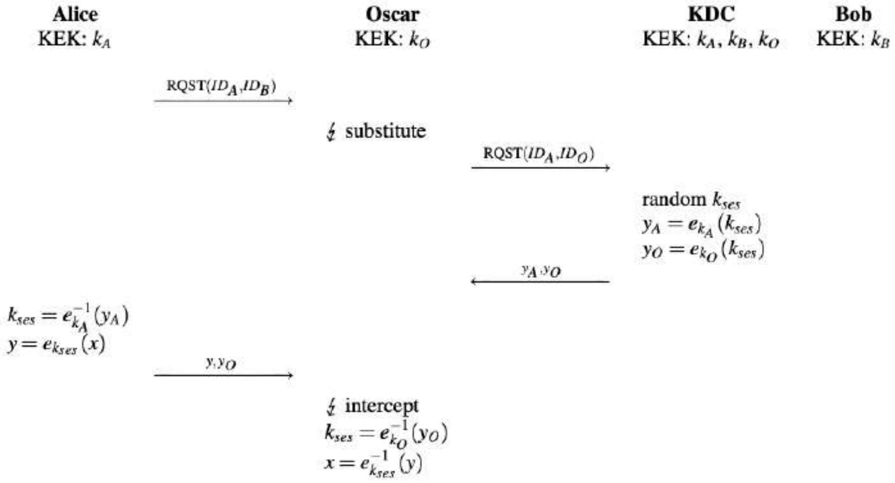
- If Oscar gets hold of a previous session key, he can impersonate the KDC and resend old messages $y_A$ and $y_B$ to Alice and Bob.
- Since Oscar knows the session key, he can decipher the plaintext that will be encrypted by Alice or Bob.

# Analysis of Basic Protocol

- If Oscar gets hold of a previous session key, he can impersonate the KDC and resend old messages $y_A$ and $y_B$ to Alice and Bob.
- Since Oscar knows the session key, he can decipher the plaintext that will be encrypted by Alice or Bob.

- **Key Confirmation Attack**
  - Another weakness of the above protocol is that Alice is not assured that the key material she receives from the KDC is actually for a session between her and Bob.
  - This attack assumes that Oscar is also a legitimate (but malicious) user.
  - By changing the session-request message Oscar can trick the KDC and Alice to set up session between him and Alice as opposed to between Alice and Bob.

# Key Confirmation Attack



**Alice**
KEK: $k_A$

**Oscar**
KEK: $k_O$

**KDC**
KEK: $k_A, k_B, k_O$

**Bob**
KEK: $k_B$

$\xrightarrow{\text{RQST}(ID_A, ID_B)}$

↯ substitute

$\xrightarrow{\text{RQST}(ID_A, ID_O)}$

random $k_{ses}$
$y_A = e_{k_A}(k_{ses})$
$y_O = e_{k_O}(k_{ses})$

$\xleftarrow{y_A, y_O}$

$k_{ses} = e^{-1}_{k_A}(y_A)$
$y = e_{k_{ses}}(x)$

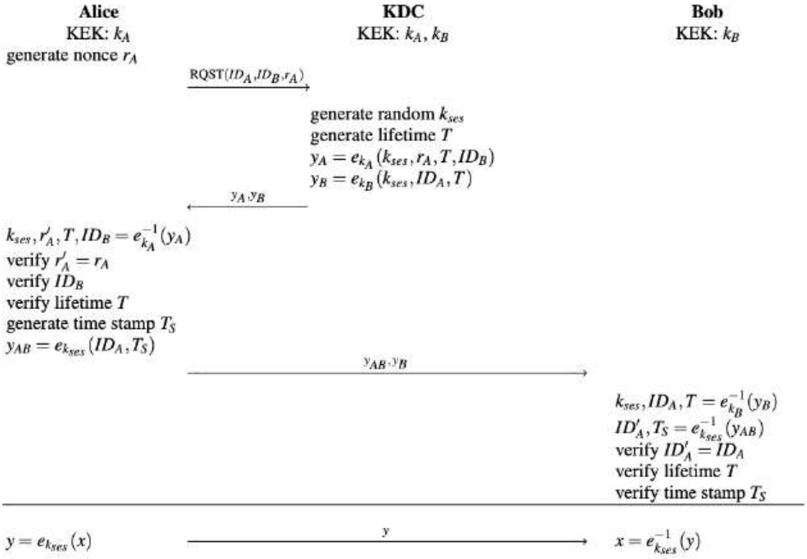
$\xrightarrow{y, y_O}$

↯ intercept
$k_{ses} = e^{-1}_{k_O}(y_O)$
$x = e^{-1}_{k_{ses}}(y)$

# Kerberos

- Kerberos protects against both replay and key confirmation attacks.
- It is more than a mere key distribution protocol.
- Its main purpose is to provide user authentication in computer networks.
- Kerberos was standardized as an RFC 1510 in 1993 and is in widespread use.

# Kerberos



**Alice**
KEK: $k_A$
generate nonce $r_A$

**KDC**
KEK: $k_A, k_B$

**Bob**
KEK: $k_B$

$\xrightarrow{\text{RQST}(ID_A, ID_B, r_A)}$

generate random $k_{ses}$
generate lifetime $T$
$y_A = e_{k_A}(k_{ses}, r_A, T, ID_B)$
$y_B = e_{k_B}(k_{ses}, ID_A, T)$

$\xleftarrow{y_A, y_B}$

$k_{ses}, r'_A, T, ID_B = e^{-1}_{k_A}(y_A)$
verify $r'_A = r_A$
verify $ID_B$
verify lifetime $T$
generate time stamp $T_S$
$y_{AB} = e_{k_{ses}}(ID_A, T_S)$

$\xrightarrow{y_{AB}, y_B}$

$k_{ses}, ID_A, T = e^{-1}_{k_B}(y_B)$
$ID'_A, T_S = e^{-1}_{k_{ses}}(y_{AB})$
verify $ID'_A = ID_A$
verify lifetime $T$
verify time stamp $T_S$

$y = e_{k_{ses}}(x)$ $\xrightarrow{y}$ $x = e^{-1}_{k_{ses}}(y)$

# Kerberos

- Kerberos assures the timeliness of the protocol through two measures.
  1. First, the KDC specifies a lifetime $T$ for the session key.
  2. Second, Alice uses a time stamp $T_S$, through which Bob can be assured that Alice's messages are recent and are not the result of a replay attack.
- The usage of the lifetime parameter $T$ and the time stamp $T_S$ prevent replay attacks by Oscar.

# Kerberos

- It provides key confirmation and user authentication.
- In the beginning, Alice sends a random nonce $r_A$ to the KDC.
- This can be considered as a challenge because she challenges the KDC to encrypt it with their joint KEK $k_A$.
- If the returned challenge $r'_A$ matches the sent one, Alice is assured that the message $y_A$ was actually sent by the KDC.
- This method to authenticate users is known as challenge-response protocol and is widely used, e.g., for authentication of smart cards.

# Kerberos

- Through the inclusion of Bob's identity $ID_B$ in $y_A$, Alice is assured that the session key is actually meant for a session between herself and Bob.
- With the inclusion of Alice's identity $ID_A$ in both $y_B$ and $y_{AB}$, Bob can verify that
  1. the KDC included a session key for a connection between him and Alice and
  2. that he is currently actually talking to Alice.

# Problems with Symmetric-Key Distribution

- **Communication requirements:**
  - One problem in practice is that the KDC needs to be contacted if a new secure session is to be initiated between any two parties in the network.

# Problems with Symmetric-Key Distribution

- **Communication requirements:**
  - One problem in practice is that the KDC needs to be contacted if a new secure session is to be initiated between any two parties in the network.
  - In Kerberos, one can alleviate this potential problem by increasing the lifetime $T$ of the key.
  - Kerberos can run with tens of thousands of users but not for all Internet users.

# Problems with Symmetric-Key Distribution

- **Communication requirements:**
  - One problem in practice is that the KDC needs to be contacted if a new secure session is to be initiated between any two parties in the network.
  - In Kerberos, one can alleviate this potential problem by increasing the lifetime $T$ of the key.
  - Kerberos can run with tens of thousands of users but not for all Internet users.
- **Secure channel during initialization**
- **Single point of failure**
  - The database that contains the key encryption keys, the KEKs.
  - If the KDC becomes compromised
- **No perfect forward secrecy**

# Outline

1 Introduction

2 Classification and Framework

3 Key Establishment Based on Symmetric Encryption

4 Key Establishment Based on Asymmetric Encryption

5 Secret Sharing

# Advantages

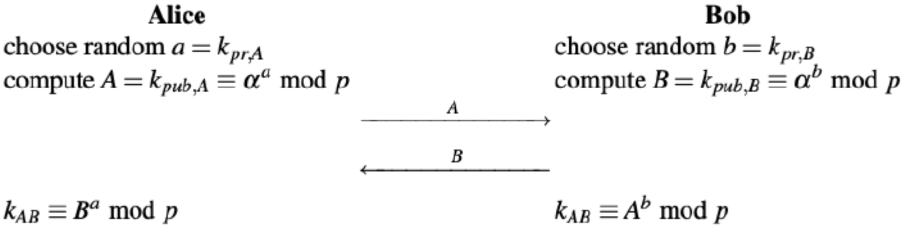
- In key establishment, *perfect forward secrecy (PFS)* can be achieved using asymmetric encryption.
- It overcomes most of the drawbacks that we observed in symmetric key approaches.
- They can be used for both key transport and key agreement.
- For key agreement, we used Diffie-Hellman, Elliptic Curve Diffie-Hellman key exchange.
- For key transport, any of the public-key encryption schemes like RSA or ElGamal, can be used.

# Advantages

- In key establishment, *perfect forward secrecy (PFS)* can be achieved using asymmetric encryption.
- It overcomes most of the drawbacks that we observed in symmetric key approaches.
- They can be used for both key transport and key agreement.
- For key agreement, we used Diffie-Hellman, Elliptic Curve Diffie-Hellman key exchange.
- For key transport, any of the public-key encryption schemes like RSA or ElGamal, can be used.

**Drawback:** It requires **an authenticated channel** to distribute the public keys.

# Diffie-Hellman Key Exchange



**Alice**

choose random $a = k_{pr,A}$

compute $A = k_{pub,A} \equiv \alpha^a \bmod p$

**Bob**

choose random $b = k_{pr,B}$

compute $B = k_{pub,B} \equiv \alpha^b \bmod p$

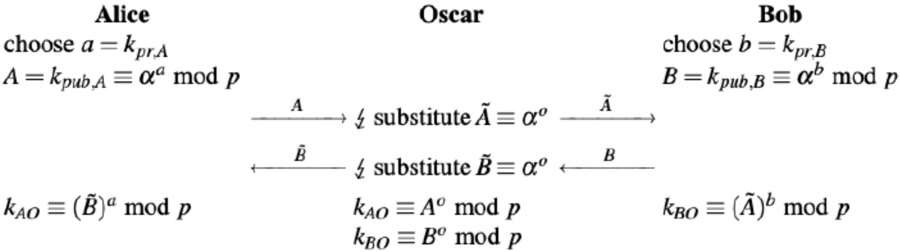
$\xrightarrow{A}$

$\xleftarrow{B}$

$k_{AB} \equiv B^a \bmod p$

$k_{AB} \equiv A^b \bmod p$

# MITM Againt Diffie-Hellman Key Exchange



**Alice**

choose $a = k_{pr,A}$

$A = k_{pub,A} \equiv \alpha^a \bmod p$

**Oscar**

$\xrightarrow{A}$ ↯ substitute $\tilde{A} \equiv \alpha^o$ $\xrightarrow{\tilde{A}}$

$\xleftarrow{\tilde{B}}$ ↯ substitute $\tilde{B} \equiv \alpha^o$ $\xleftarrow{B}$

$k_{AO} \equiv A^o \bmod p$

$k_{BO} \equiv B^o \bmod p$

**Bob**

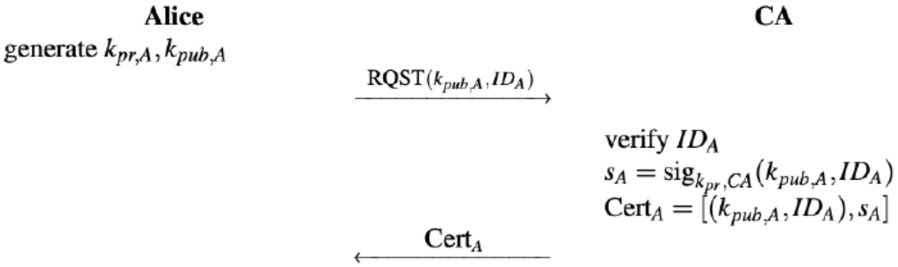
choose $b = k_{pr,B}$

$B = k_{pub,B} \equiv \alpha^b \bmod p$

$k_{AO} \equiv (\tilde{B})^a \bmod p$

$k_{BO} \equiv (\tilde{A})^b \bmod p$

# Certificate Generation with User-Provided Keys



**Alice**

generate $k_{pr,A}, k_{pub,A}$

$\xrightarrow{\text{RQST}(k_{pub,A}, ID_A)}$

**CA**

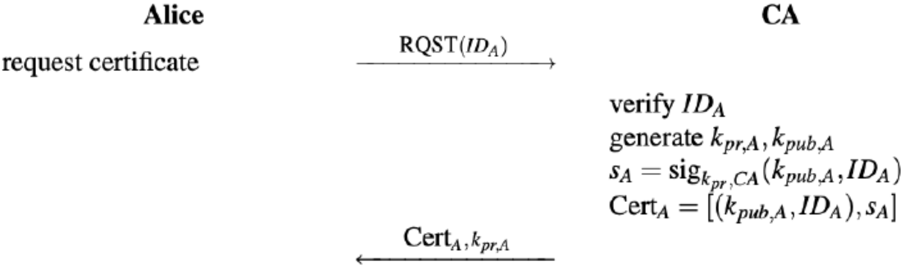
verify $ID_A$

$s_A = \text{sig}_{k_{pr,CA}}(k_{pub,A}, ID_A)$

$\text{Cert}_A = [(k_{pub,A}, ID_A), s_A]$

$\xleftarrow{\text{Cert}_A}$

# Certificate Generation with CA-Generated Keys



**Alice** | **CA**

request certificate

$\xrightarrow{\text{RQST}(ID_A)}$

verify $ID_A$

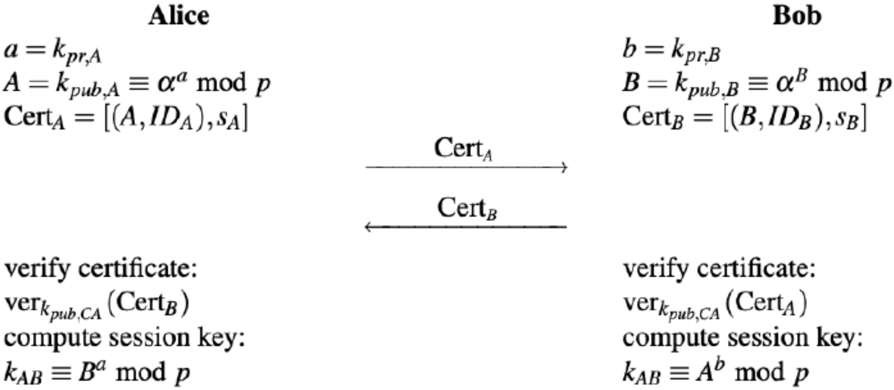
generate $k_{pr,A}, k_{pub,A}$

$s_A = \text{sig}_{k_{pr,CA}}(k_{pub,A}, ID_A)$

$\text{Cert}_A = [(k_{pub,A}, ID_A), s_A]$

$\xleftarrow{\text{Cert}_A, k_{pr,A}}$

# Diffie-Hellman Key Exchange with Certificates



**Alice**

$a = k_{pr,A}$

$A = k_{pub,A} \equiv \alpha^a \bmod p$

$\mathrm{Cert}_A = [(A, ID_A), s_A]$

**Bob**

$b = k_{pr,B}$

$B = k_{pub,B} \equiv \alpha^B \bmod p$

$\mathrm{Cert}_B = [(B, ID_B), s_B]$

$\xrightarrow{\mathrm{Cert}_A}$

$\xleftarrow{\mathrm{Cert}_B}$

verify certificate:

$\mathrm{ver}_{k_{pub,CA}}(\mathrm{Cert}_B)$

compute session key:

$k_{AB} \equiv B^a \bmod p$

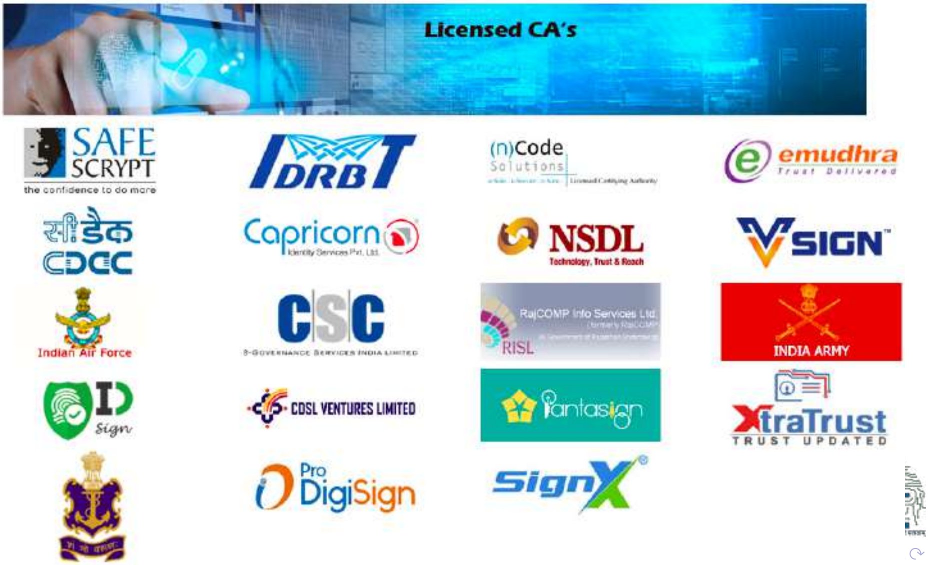
verify certificate:

$\mathrm{ver}_{k_{pub,CA}}(\mathrm{Cert}_A)$

compute session key:

$k_{AB} \equiv A^b \bmod p$

# Licensed CAs



# Outline

1 Introduction

2 Classification and Framework

3 Key Establishment Based on Symmetric Encryption

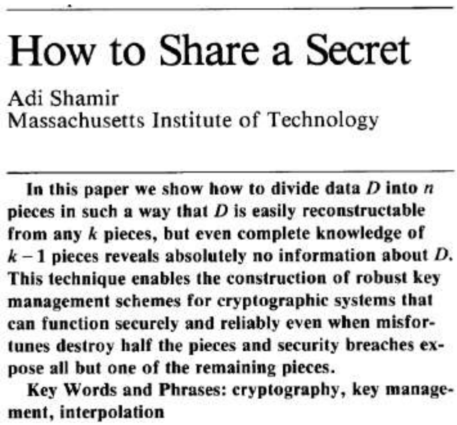
4 Key Establishment Based on Asymmetric Encryption

5 Secret Sharing

*"Three may keep a secret, if two of them are dead."*

– Benjamin Franklin

# Shamir’s Secret Sharing Scheme



How to Share a Secret

Adi Shamir
Massachusetts Institute of Technology

In this paper we show how to divide data $D$ into $n$ pieces in such a way that $D$ is easily reconstructable from any $k$ pieces, but even complete knowledge of $k - 1$ pieces reveals absolutely no information about $D$. This technique enables the construction of robust key management schemes for cryptographic systems that can function securely and reliably even when misfortunes destroy half the pieces and security breaches expose all but one of the remaining pieces.

Key Words and Phrases: cryptography, key management, interpolation

A. Shamir„
*How to Share a Secret*, Communications of the ACM, 22: pages 612 - 613, 1979.

# A $(t, n)$-Threshold Scheme

- A $(t, n)$-threshold scheme based on polynomial interpolation over $\mathbb{Z}_p$, where $p$ is prime.
- $p \geq n + 1$ be a prime
- **Initialization phase:**
    - The TA selects $n$ distinct elements $x_1, x_2, \cdots, x_n$ from $\mathbb{Z}_p^*$
    - The TA gives $x_i$ to $P_i \; \forall \; 1 \leq i \leq n$
    - The $x_i$'s are public information

# Share Generation for the Shamir's Scheme

- Given a secret $K \in \mathbb{Z}_p$, the share generation algorithm is as follows:

  1. The TA chooses $a_1, \cdots, a_{t-1}$ independently and uniformly at random from $\mathbb{Z}_p$
  2. The TA defines

$$a(x) = K + \sum_{j=1}^{t-1} a_j x^j$$

  3. The TA constructs the share

$$s_i = a(x_i) \ \forall \ 1 \leq i \leq n$$

# References

- Alfred J. Menezes, Paul C. van Oorschot & Scott A. Vanstone,
  *Handbook of Applied Cryptography*, CRC Press, 1996.

  https://cacr.uwaterloo.ca/hac/

- C. Paar & J. Pelzl,
  *Understanding Cryptography*, Springer, 2010.

# The End

**Thanks a lot for your attention!**