ECE 443/518 – Computer Cyber Security
Lecture 12 Key Establishment with Symmetric Cryptography

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

September 30, 2024

Outline

Midterm Exam

- Lecture 1 $\sim$ Lecture 13, see Homework 1 and 2 for sample.
  - Points may be deducted if key steps are missing.
- Students registered for main campus section: Wed. 10/9, 11:25 AM – 12:40 PM, in class.
  - A physical calculator is allowed. Laptop or any other electronic device or calculator apps running on them are not allowed.
  - Closed book/notes. A letter-size page of cheat sheet is allowed.
- Online students may take the exam as above, or follow the instruction in the email "ECE 443/513 Exams for Online Section Students" from Charles Scott (scott@iit.edu).
  - Contact Scott and me if you cannot find the email.
  - No make-up exam will be offered if you fail to do so.
- ADA Accommodations: contact Center for Disability Resource (disabilities@iit.edu)
- Emergency/extraordinary reasons for make-up midterm exams are accepted only with documented proof like docter's notes.

Reading Assignment

- This lecture: UC 13
- Next lecture: UC 13

Outline

Key Establishment

- To establishing a shared secret between two or more parties.
  - Which could be used later for secure communication via symmetric cryptography.
- Key transport: one party securely transfers a secret value to others
- Key agreement: two or more parties derive the shared secret
  - Ideally, none of the parties can control what the secret will be.
- Key establishment assumes identification.
  - What about the $O(n^2)$ keys needed to support pair-wise communication among n parties if we use symmetric cryptography?
  - What about the Man-in-the-Middle attack if we use public-key cryptography?

Key Freshness

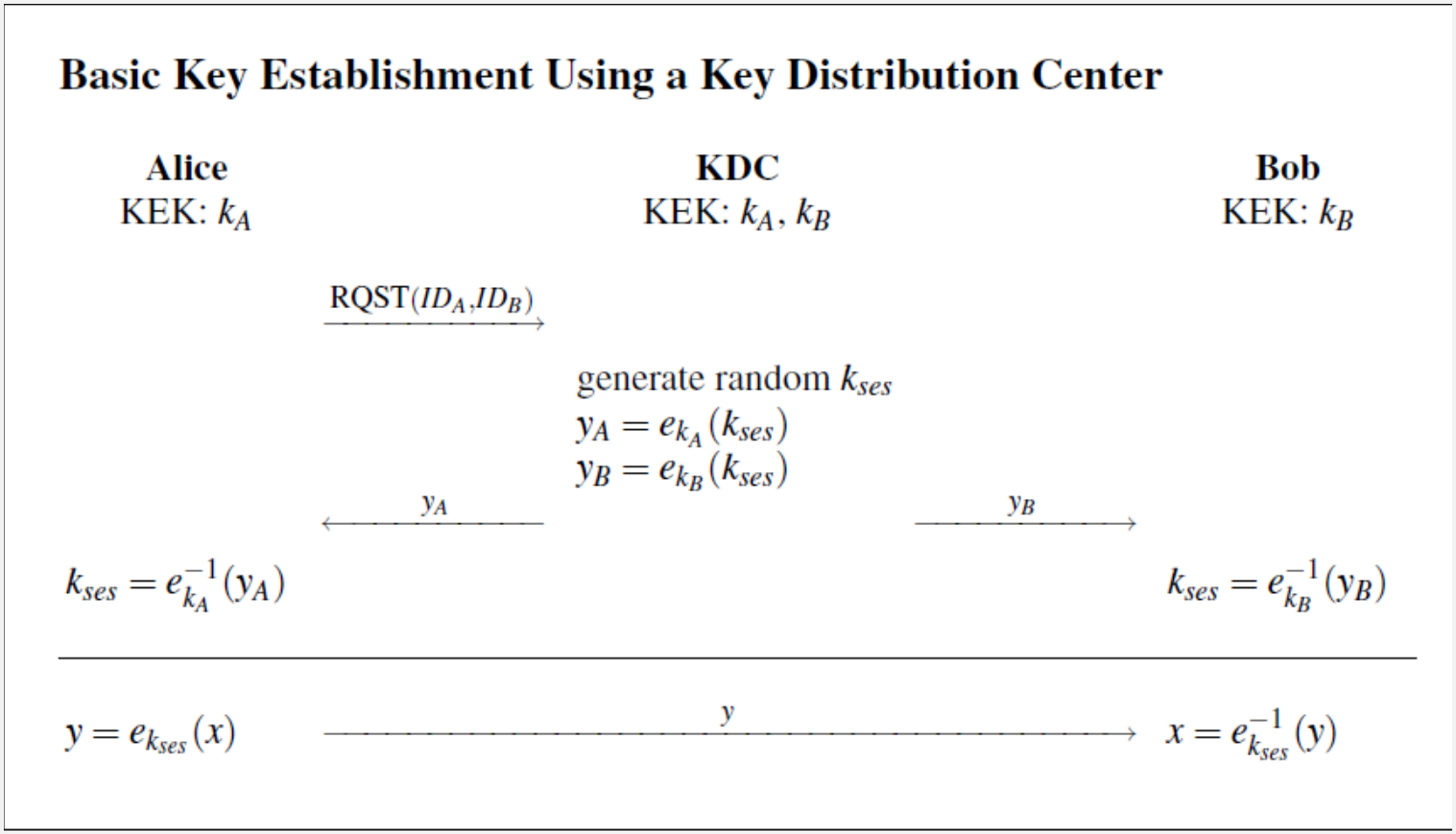
- Many security systems prefer to use one secret key only for a limited period of time.
  - Less damage if the key is exposed.
  - Less ciphertexts under the same key are available for attackers to analysis.
  - More works for attackers to decrypt same amount of ciphertexts.
- Session keys or ephemeral keys.
  - New keys are generated for each Internet connection, or within a matter of minutes, or sometimes even seconds.
- But how?
  - Need to be efficient in both computation and communication.

Outline

Kerberos

- Based on symmetric cryptography.
- Developed by MIT in 80's.
- Standardized as RFC 1510 in 1993, currently RFC 4120.

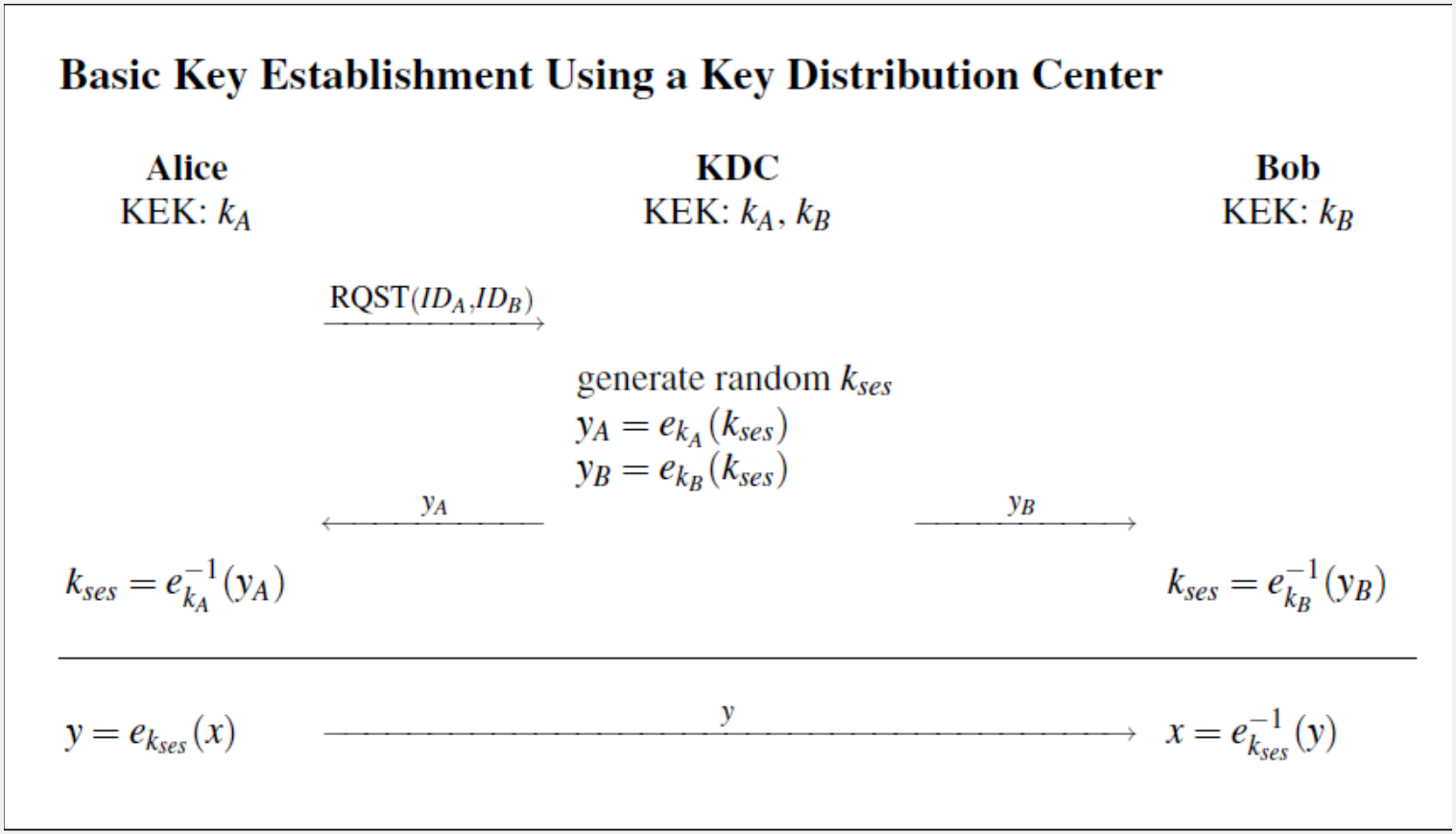
Key Distribution Center (KDC)



(page 337, Paar and Pelzl)

- A trusted third-party.
  - Able to identify each party.
  - Won't leak secret and will follow protocol faithfully.
- KDC identifies each party with its Key Encryption Key (KEK).

Basic Key Establishment using KDC



(page 337, Paar and Pelzl)

- Basic key establishment protocol
  - Alice requests to establish communication with Bob.
  - KDC generates k_{ses} and distributes to Alice (y_A) and Bob (y_B).
  - KDC may ask Alice to distribute y_B to Bob.
- Does any channel need to be secure or authentic?

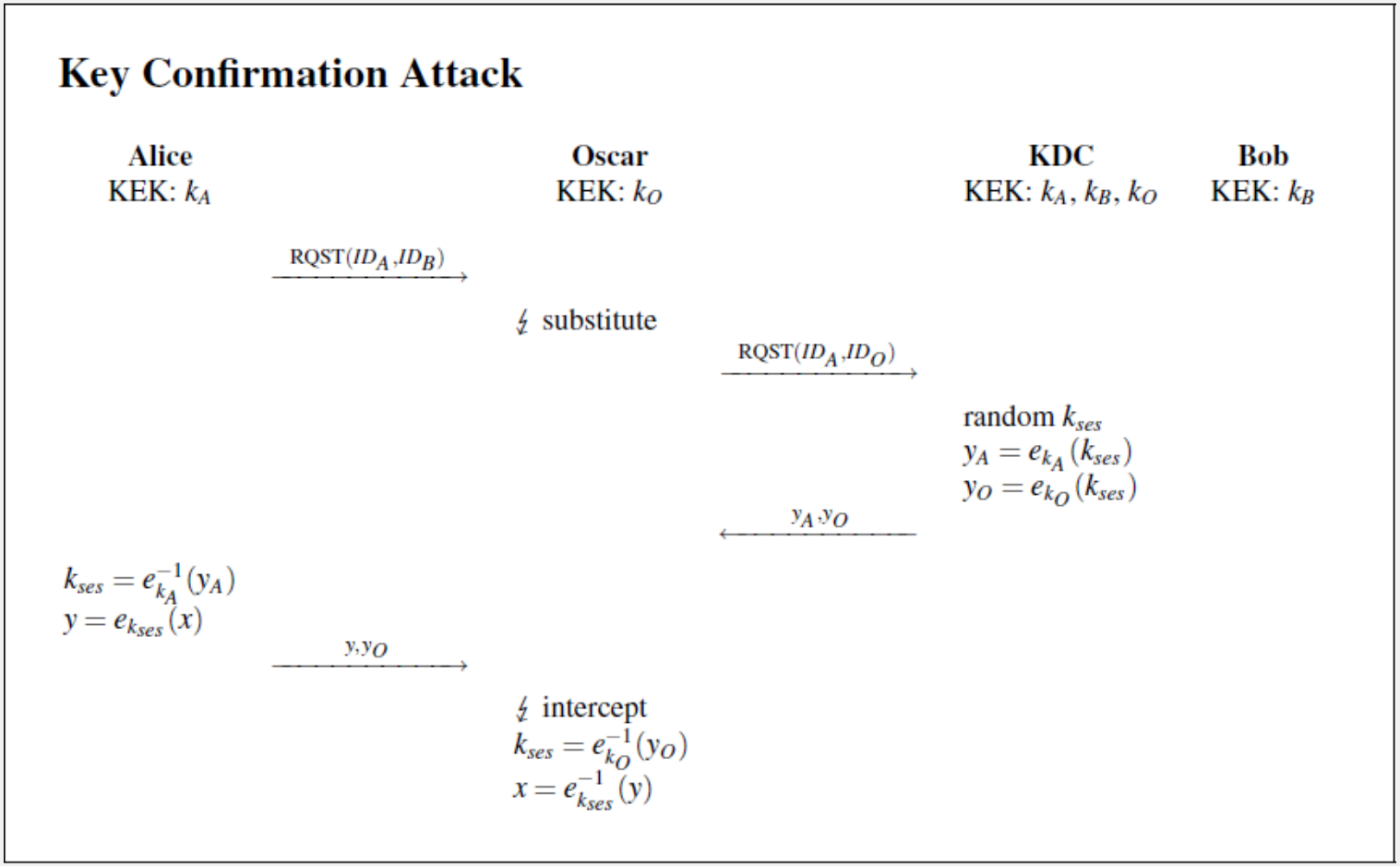
Advantages of KDC

- System wide, only KEKs need to be stored in long term.
  - $O(1)$ storage per party.
  - $O(n)$ storage per KDC
- The $O(n^2)$ keys needed to support pair-wise communication are all generated on-the-fly and ephemeral.
- Party updates are all handled by KDC.
  - Leave
  - Join
  - KEK update

Attacks

- Replay attack: Oscar may replace y_A and y_B from KDC.
  - With y'_A and y'_B that correspond to a previously compromised session key.
- Key confirmation attack: what is protected by KEKs?
  - The basic key establishment protocol: k_{ses} only in y_A and y_B.
  - Imply a valid session but not necessarily a session between Alice and Bob – how Bob confirms messages encrypted by k_{ses} from Alice?
  - Give Oscar, a legitimate but malicious user, opportunities to attack the system.

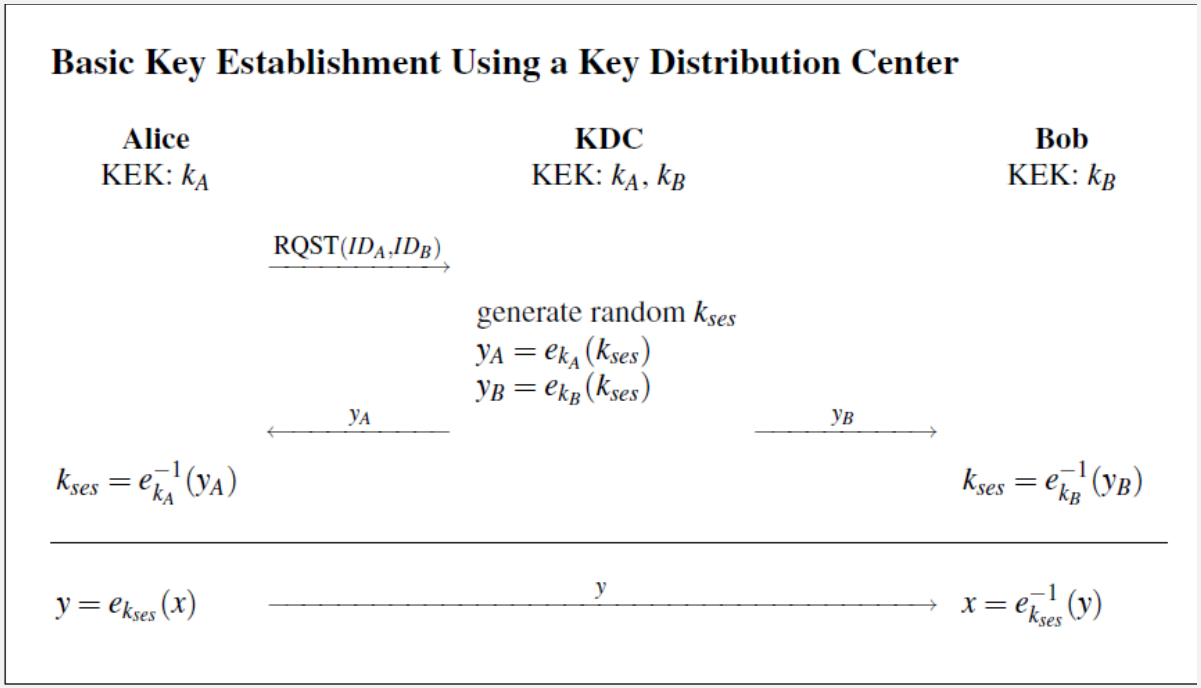
Key Confirmation Attack



(page 339, Paar and Pelzl)

- Oscar could further establish a session with Bob and forward x to Bob in order to impersonate Alice to Bob.
  - Consider this as Man-in-the-Middle attacks for symmetric cryptography when more than two parties are involved!

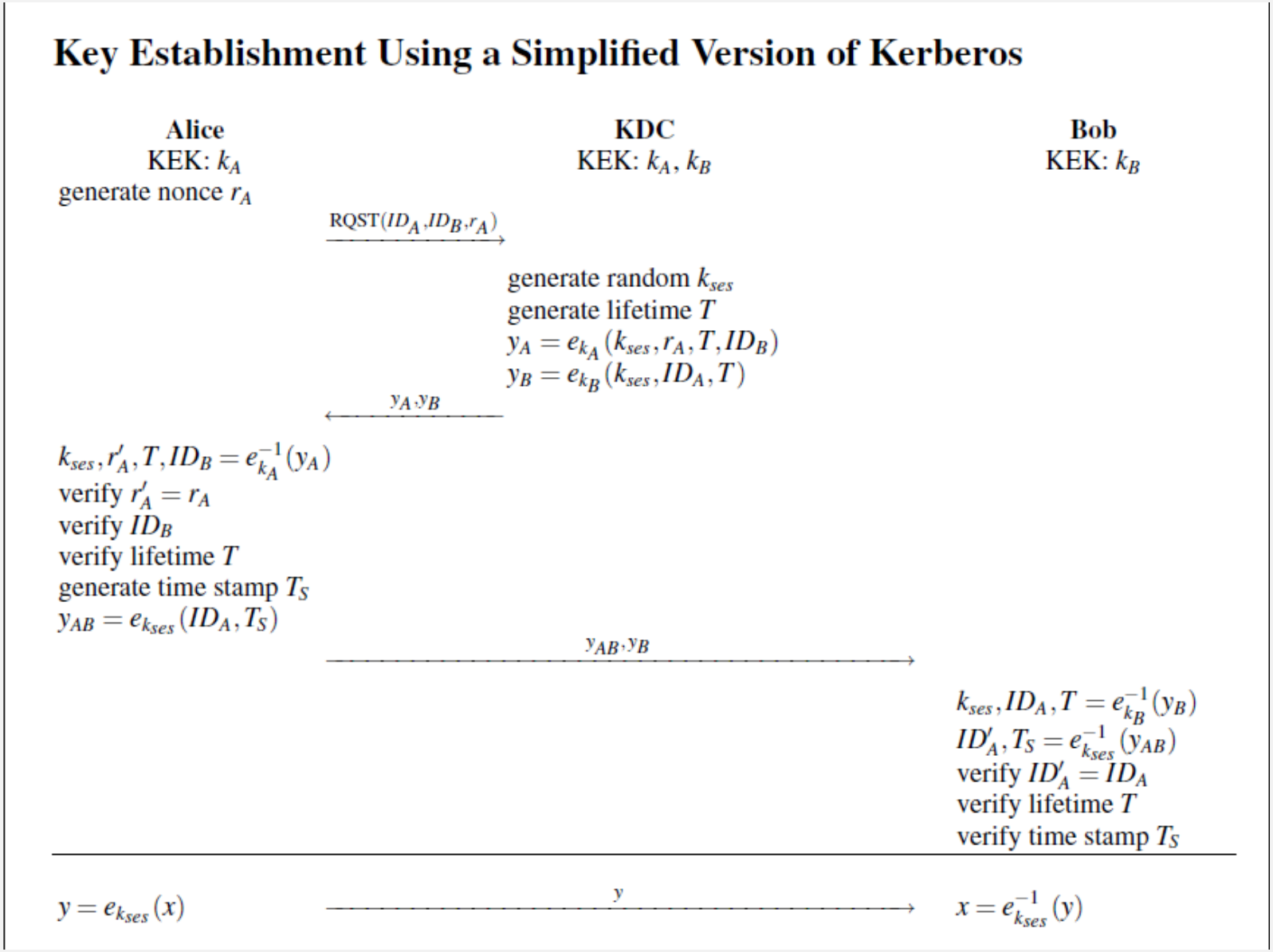
Improving Basic Key Establishment



(page 337, Paar and Pelzl)

- Need to include session information in y_A and y_B.
  - Challenge-response: no replay attack on Alice.
  - Participating parties: who are you talking to.
  - Time: no replay attack on Bob.

Kerberos



(page 340, Paar and Pelzl)

Remaining Issues

- KEK setup and update require secure channel to KDC.
  - As implied by symmetric cryptography.
- Communication requirements: KDC need to be online.
  - Performance concerns: need to response to every session.
  - Reliability concerns: no more sessions if KDC fails.
  - Should we add a secondary KDC?
- Single point of failure: security disaster.
  - A compromised KDC reveal all KEKs and thus all future communications.
  - And thus all past session keys if Oscar has recorded all sessions.
  - And this is highly possible since attackers know this weakness!
- Perfect forward secrecy (PFS): can we protect past session keys if KEKs are compromised?
  - KEKs are used to authenticate parties and to exchange session keys. Can we seperate these two purposes?

Outline

Password Authentication

- A cryptography system based on symmetric cryptography, e.g. Kerberos, inevitably depends on shared secrets between the system and its users.
  - Password: a string that could be memorized by human beings.
- Setup: Alice comes up with a password, and shares it with Bob via a secure channel.
  - This is a secret that none of Alice and Bob should disclose.
- Authentication: Bob asks whoever claims to be Alice to show knowledge of the password.
  - Via challenge-response and authenticated encryption.
  - Or on a secure channel created by public-key cryptography.
  - What if Bob is not Bob and Alice sends the password directly?

Why Password Authentication?

- Apparent "advantages"
  - Simple to implement for Bob: compare strings.
  - No additional hardware for Alice: memorize strings.
  - Provide mutual authentication between Alice and Bob.
- Disadvantage
  - Without proper protection, Oscar may obtain Alice's password from where Bob stores passwords.
  - Alice may need to authenticate to many Bob's.
    - Easy-to-remember passwords are easy for Oscar to guess.
    - Use a password manager that depends on Internet or a device.
    - How about write passwords down on sticky notes?
  - Nonrepudiation does not hold and auditing requires additional evidences.

Password Storage

- Assume Bob need to store many passwords for his customers.
- What if Oscar stole the file containing these many passwords?
- Hash and salting
  - Instead of storing *password* directly in a file, Bob stores both a random *salt* and $MAC_{salt}(password)$.
  - So Oscar cannot recover *password* from Bob's password file easily.
  - If Bob does not use *salt*, Oscar may precompute hashes for popular passwords and then easily identify them from the file.
- This is a standard practice for over 40 years, but guess how many websites still store your passwords in plaintext!

Password Policy

- Bob may apply password policy to require his customers to use better passwords.
- Rules: length restrictions, no dictionary word, must contain uppercase/lowercase/digits/symbols, etc.
- Aging: require to replace passwords half year, one year, etc.
- But they do impact usability.
  - How about write passwords down on sticky notes?

Multi-Factor Authentication

- Use multiple methods like phone numbers, emails, devices, biometrics, and location to determine the identity
- Trade-off between usability, privacy, and security.
- The process to reset authentication could be the weakest link!

Summary

- Kerberos: key establishment based on symmetric cryptography may work, but has a lot of potential issues.
- It seems trivial to make passwords more secure but it isn't.