Week 5

<u>Required Additional Reading</u>: https://www.tag-cyber.com/advisory/annuals

Week 5

Case Study: Brute Force Cryptanalytic Attack

Classic Caesar Cipher
(Shift +2 in Alphabet)

Plaintext:
Loren ipsum is a pseudo Latin
text used in typography

→

Ciphertext:
Nqtgp kruwo ku c rugwfq Ncvjp
Vgzv wugf kp varqitcrjb

Week 5

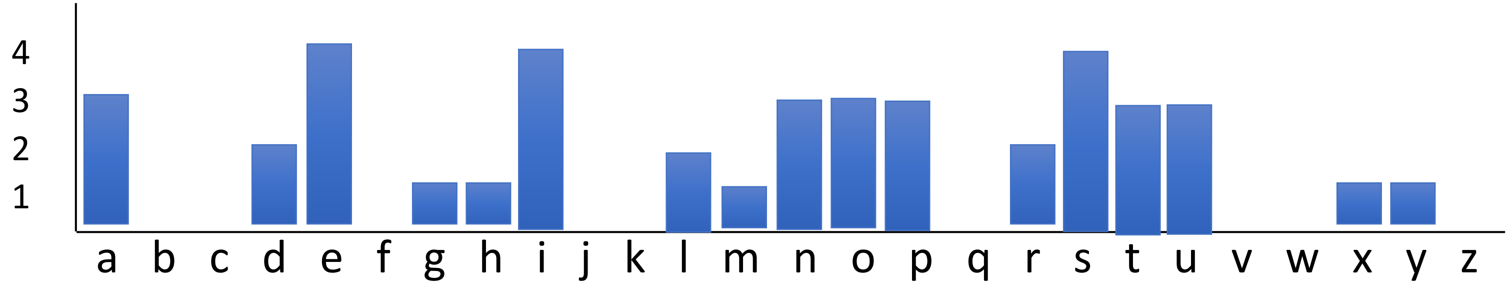
Case Study: Brute Force Cryptanalytic Attack

Classic Caesar Cipher
(Shift +2 in Alphabet)

Plaintext:
Loren ipsum is a pseudo Latin
text used in typography

Ciphertext:
Nqtgp kruwo ku c rugwfq Ncvjp
Vgzv wugf kp varqitcrjb

Plain Text Character Distribution:
(Approximates Frequency Distribution of English Language with More Data)



Week 5

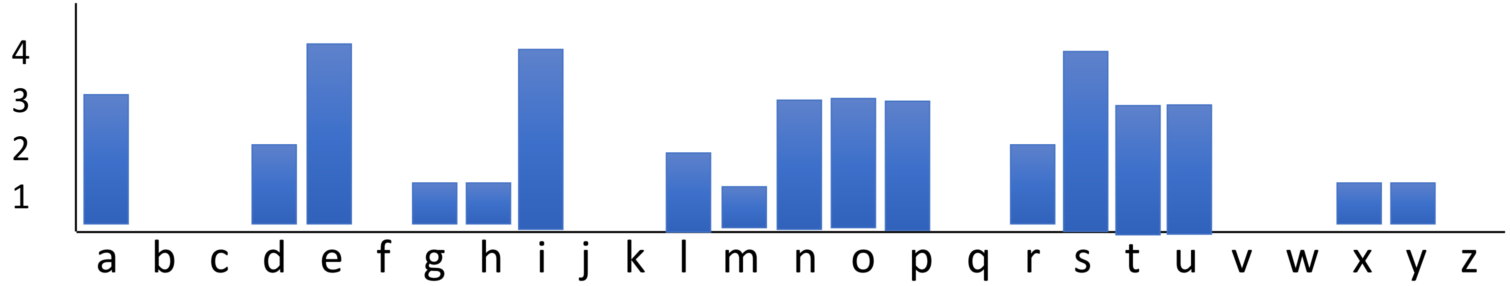
Case Study: Brute Force Cryptanalytic Attack

Plaintext:
Loren ipsum is a pseudo Latin text used in typography

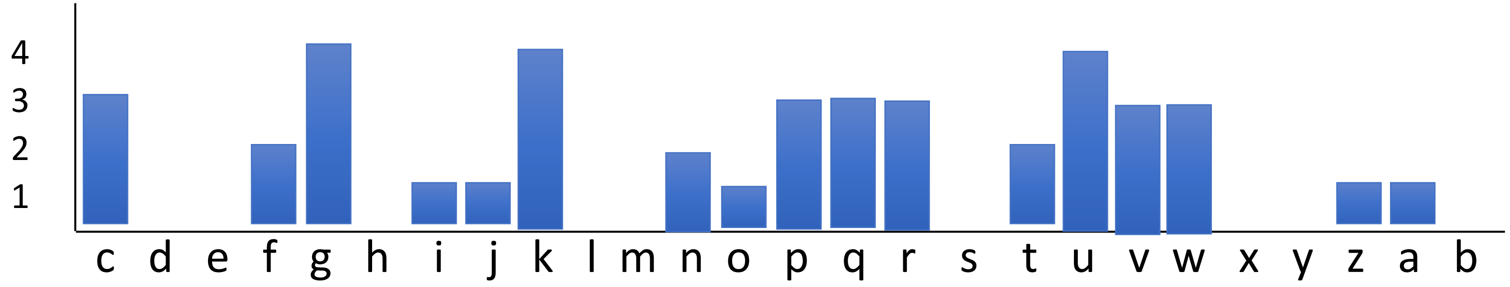
Classic Caesar Cipher
(Shift +2 in Alphabet)

Ciphertext:
Nqtgp kruwo ku c rugwfq Ncvjp Vgzv wugf kp varqitcrjb

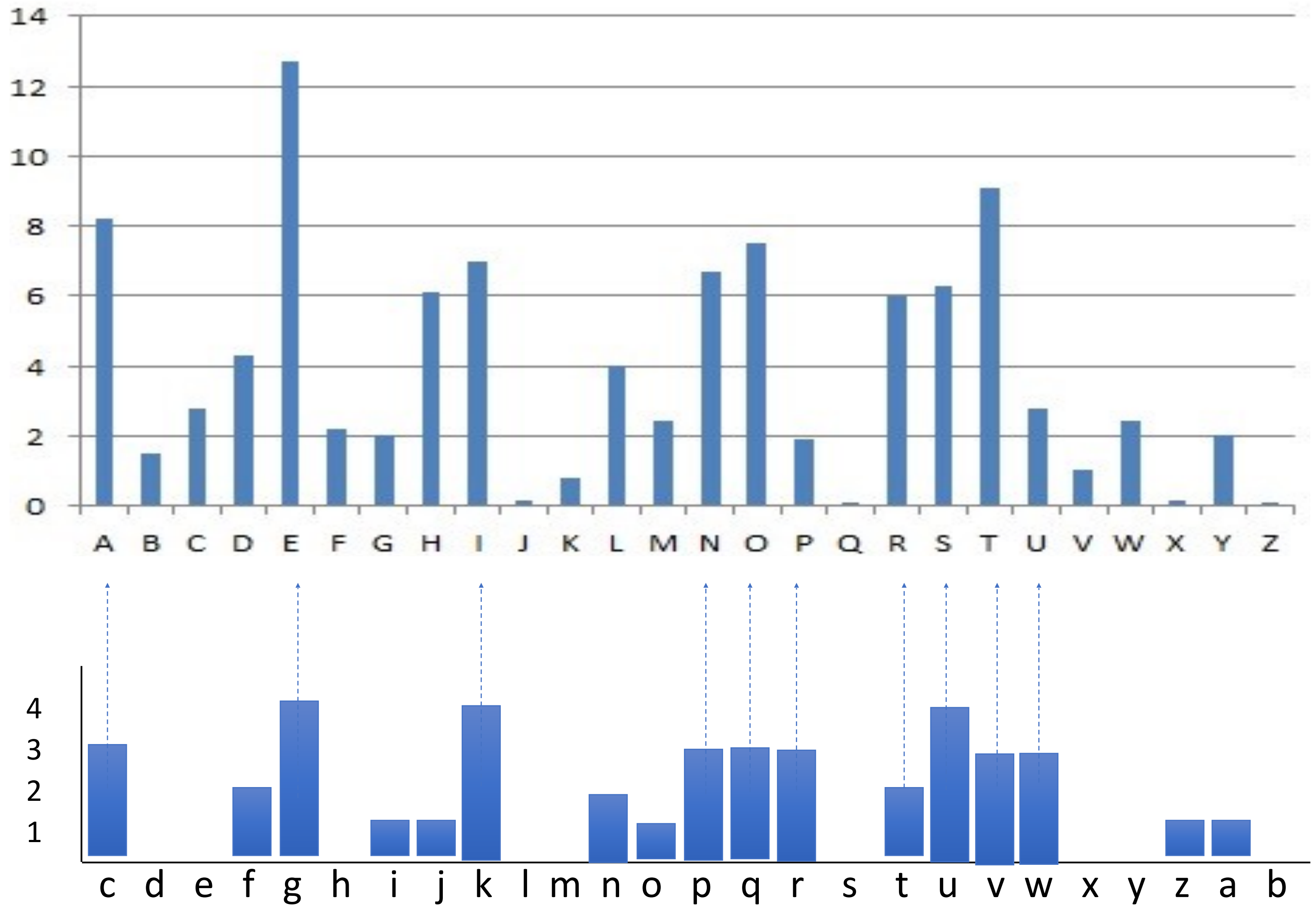
Plain Text Character Distribution:
(Approximates Frequency Distribution of English Language with More Data)



Ciphertext Character Distribution:
(Frequency Distribution Exposes Caesar Cipher Character Mapping)

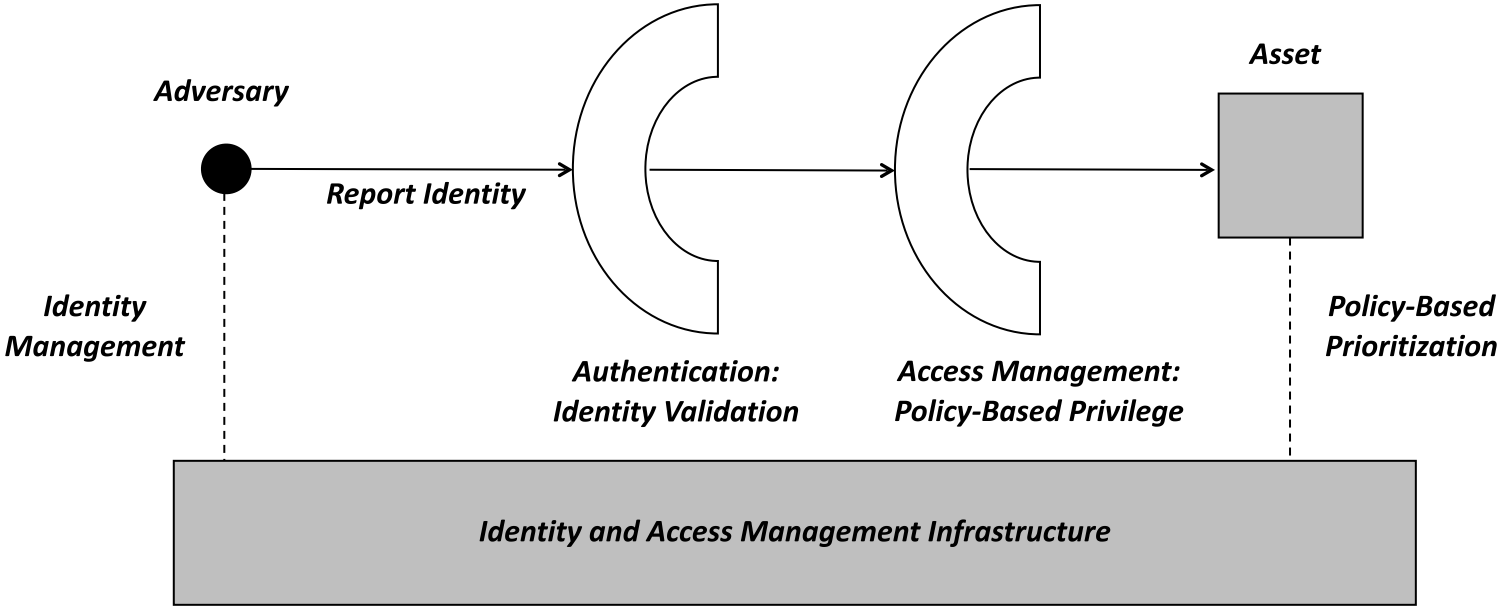


Case Study: Brute Force Cryptanalytic Attack



Actual Frequency Distribution

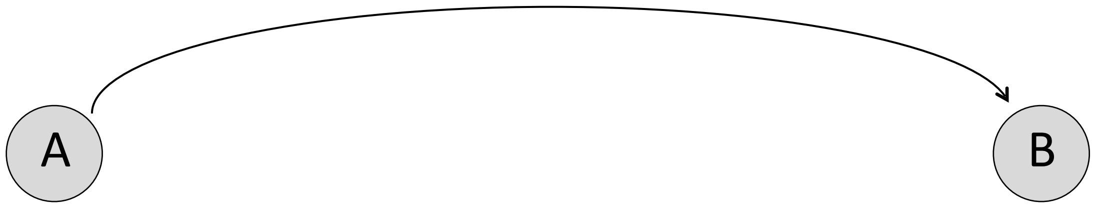
Safeguard: Authentication



Week 5

Authentication Schema

Step 1: Identification "I am Alice."



Client A – Server B: "Client Authentication"

Client B – Server A: "Server Authentication"

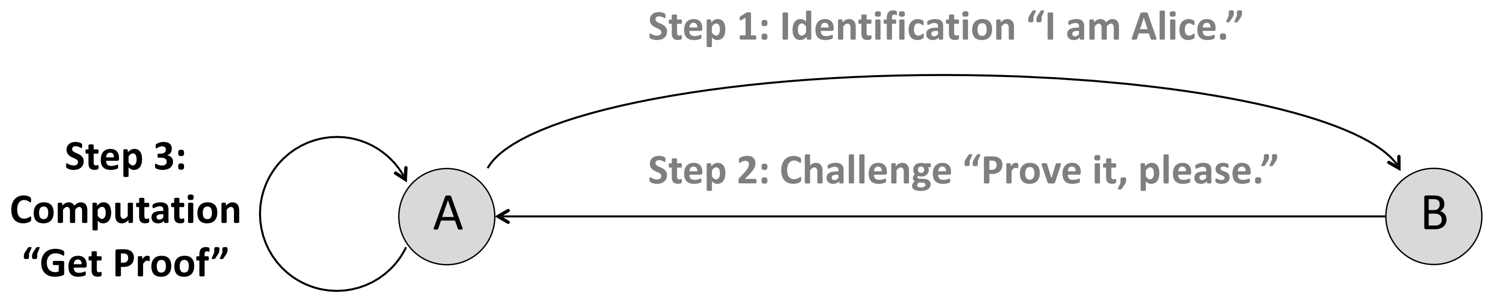
Week 5

Authentication Schema

Challenge includes tangible domain value – possible "known plaintext" attacks

Challenge includes no tangible domain value – likely to restrict to "ciphertext attacks"

Authentication Schema

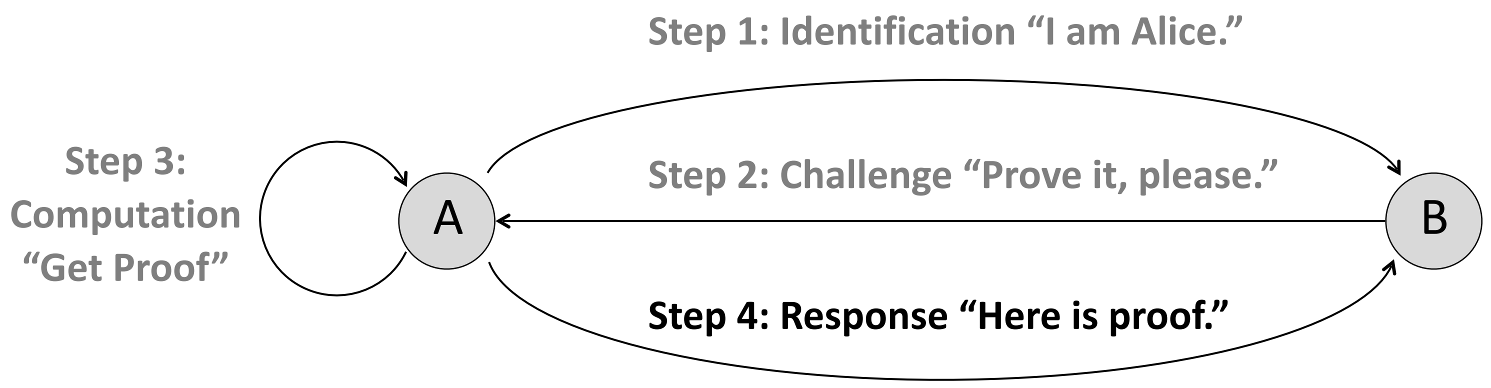


Computation might involve simple look-up/locate process (e.g., passwords)

Computation might be more deliberate mathematical operation on domain value

Week 5

Authentication Schema

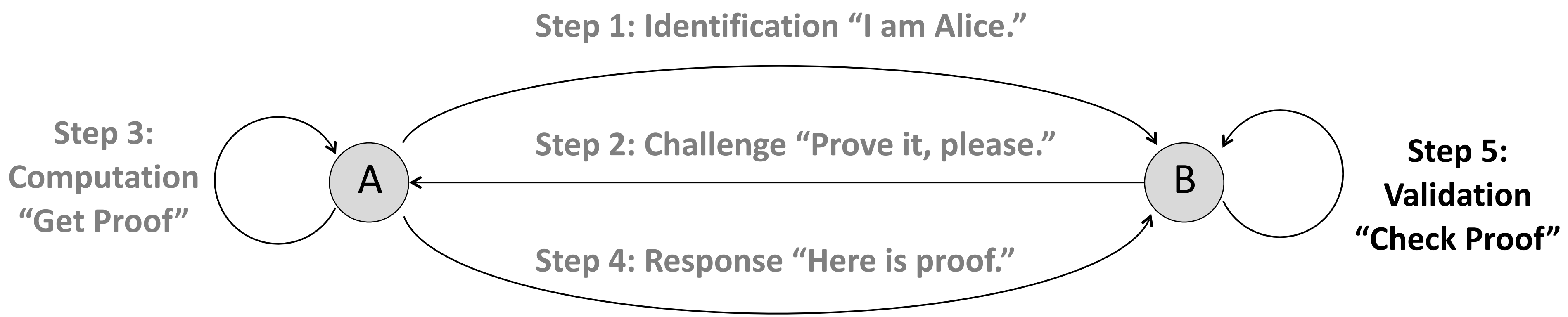


<u>Types of Proof</u>:

“Something You Know” – Passwords
“Something You Are” – Biometrics
“Something You Have” – Token
“Somewhere You Are” – Location

- ***Adaptive Authentication*** **considers context**
- ***Two-Factor Authentication*** **uses at least two types**

Authentication Schema

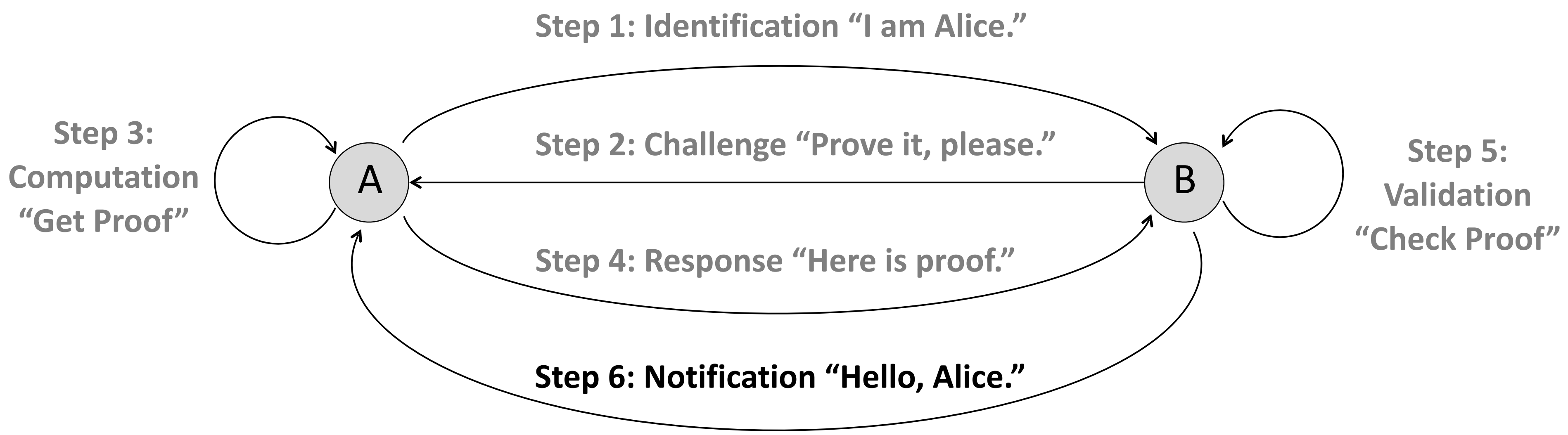


Validation might involve simple look-up/locate process (e.g., passwords)

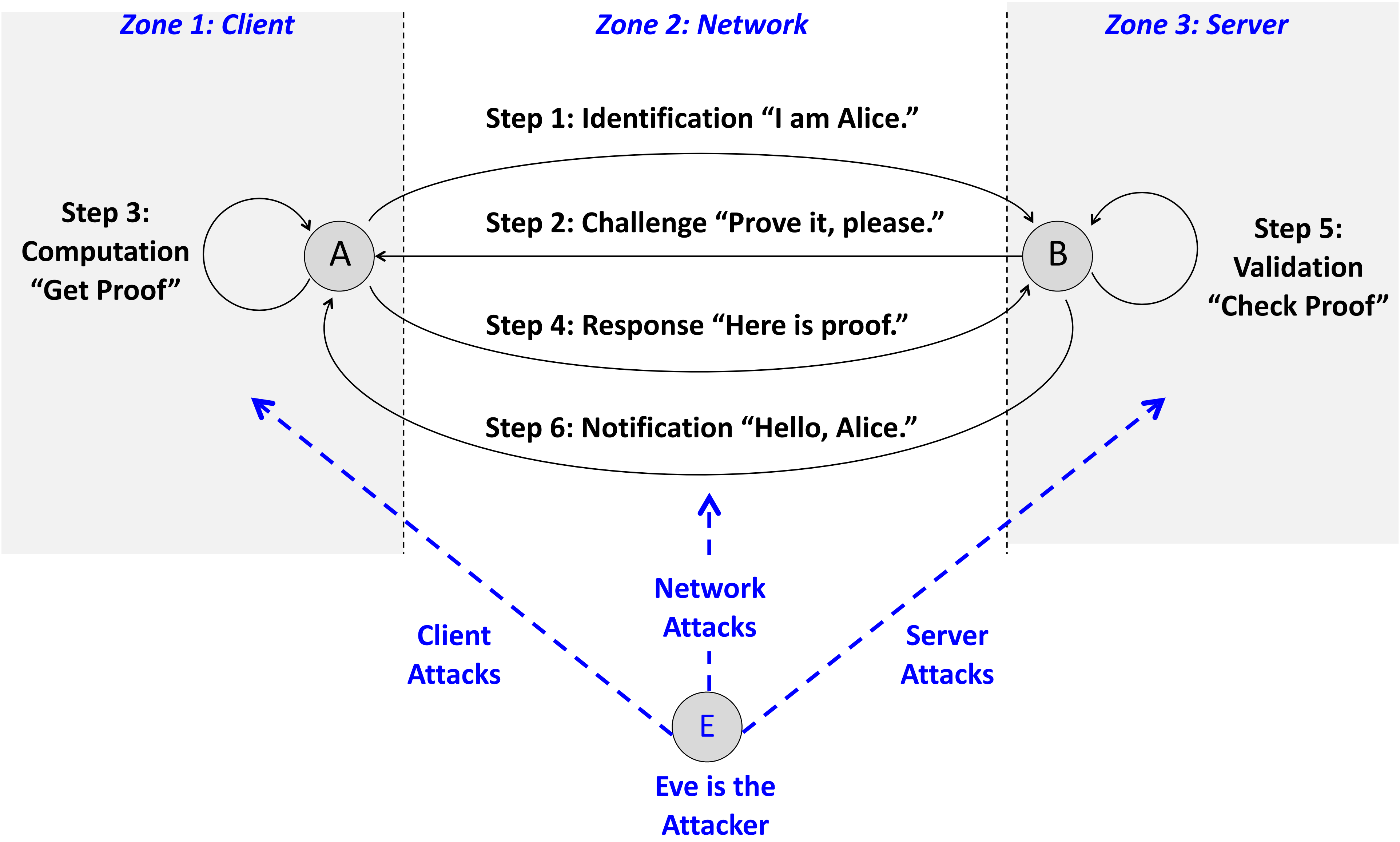
Validation might be more deliberate mathematical operation on domain value

Week 5

Authentication Schema



Authentication Schema



Week 5

Handheld Authentication Device

A

B

Embedded Function
f: integer -> integer

User	*Function*
A	**f**
C	**f '**
G	**f ''**
. . .	**. . .**

Week 5

Handheld Authentication Device

Protocol Implementation

A

B

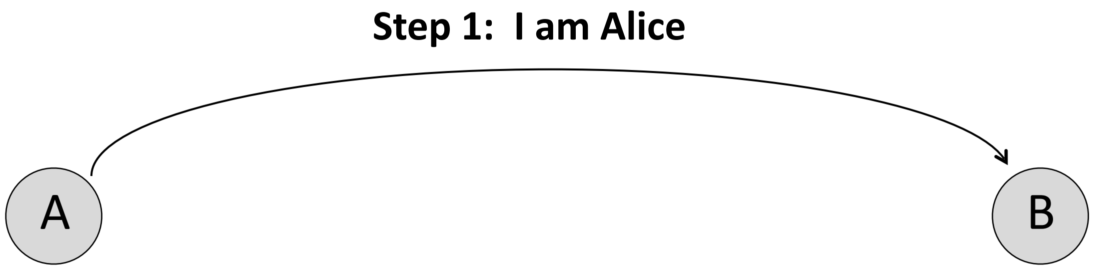
Protocol Infrastructure

Embedded Function
f: integer -> integer

User	*Function*
A	f
C	f ’
G	f ”
. . .	. . .

Week 5

Handheld Authentication Device



Embedded Function
f: integer -> integer

User	*Function*
A	**f**
C	**f ’**
G	**f ”**
. . .	. . .

Week 5

Handheld Authentication Device

Step 1: I am Alice

Step 2: λ = 237

A

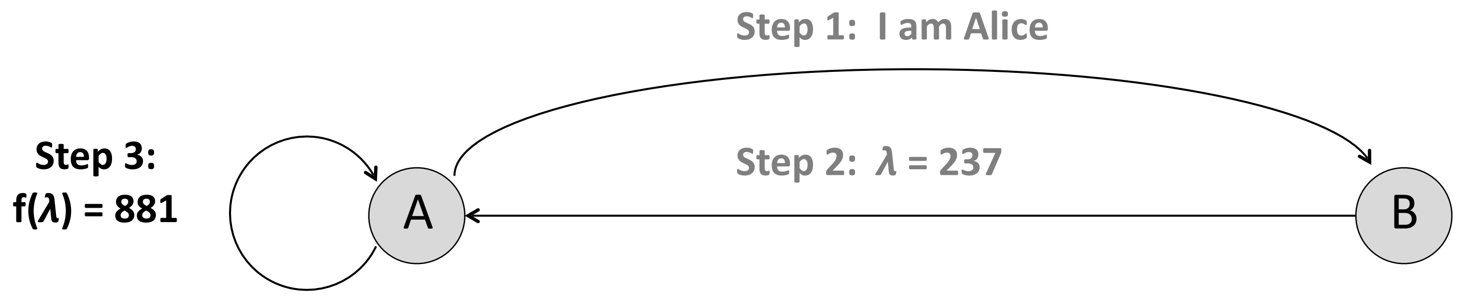
B

Randomly selected integer λ

Embedded Function
f: integer -> integer

User	*Function*
A	**f**
C	**f ’**
G	**f ’’**
. . .	. . .

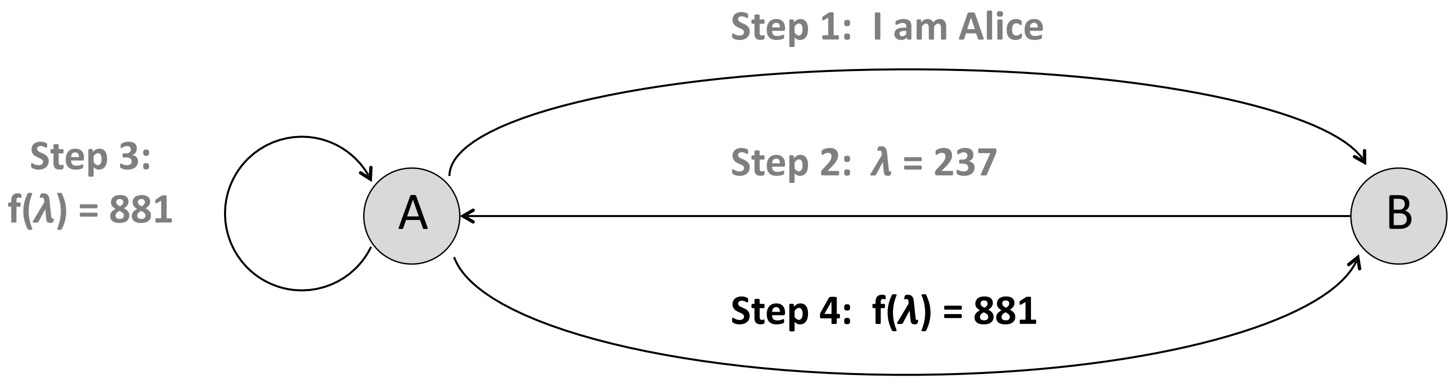
Handheld Authentication Device



Embedded Function
f: integer -> integer

User	*Function*
A	f
C	f ’
G	f ”
. . .	. . .

Handheld Authentication Device

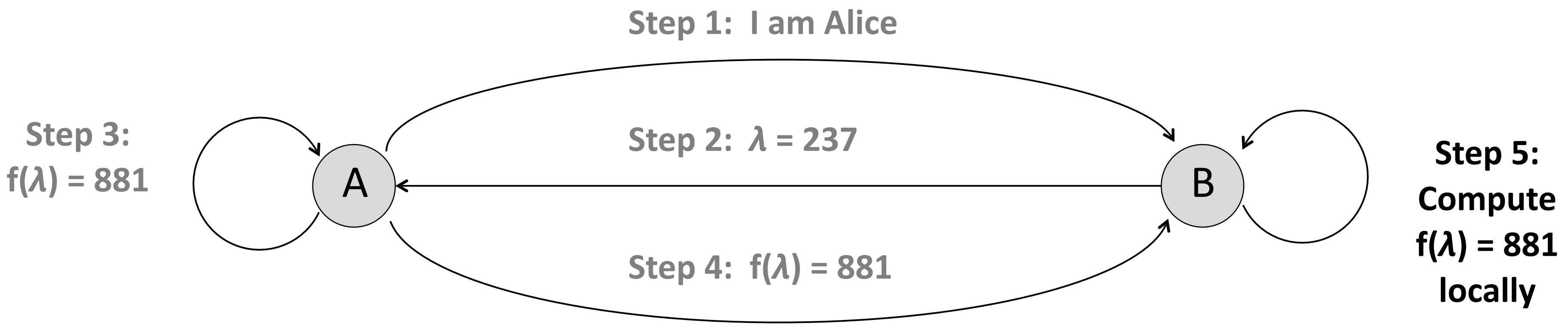


Embedded Function
f: integer -> integer

User	*Function*
A	f
C	f ’
G	f ’’
. . .	. . .

Week 5

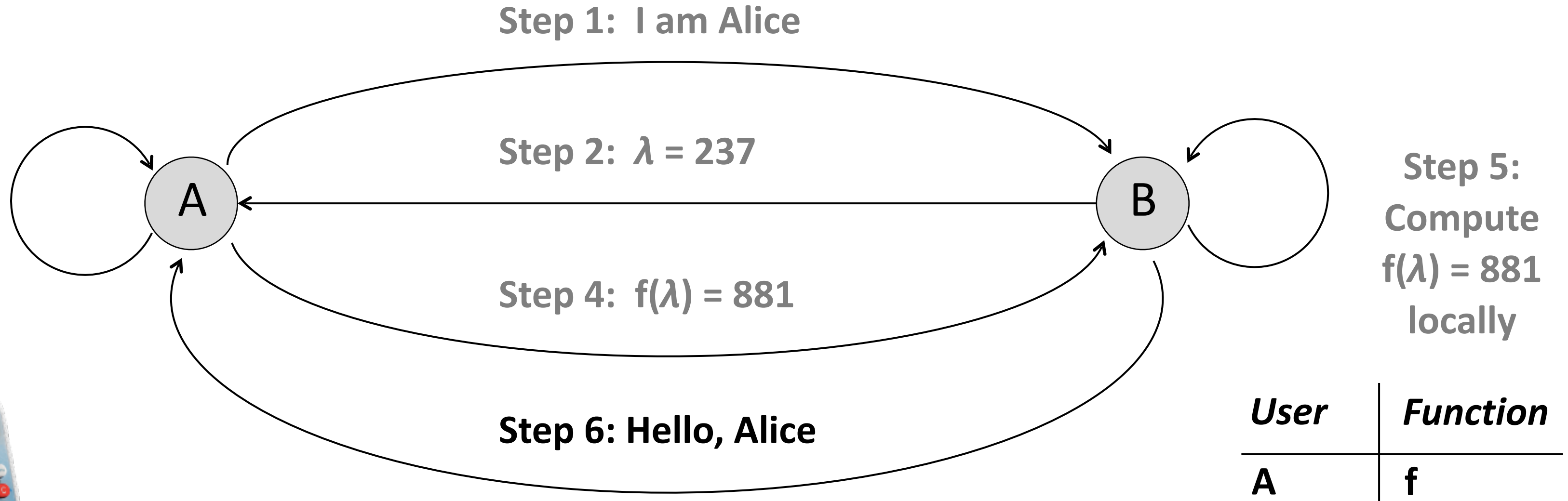
Handheld Authentication Device



Embedded Function
f: integer -> integer

User	*Function*
A	**f**
C	**f ’**
G	**f ’’**
. . .	**. . .**

Handheld Authentication Device

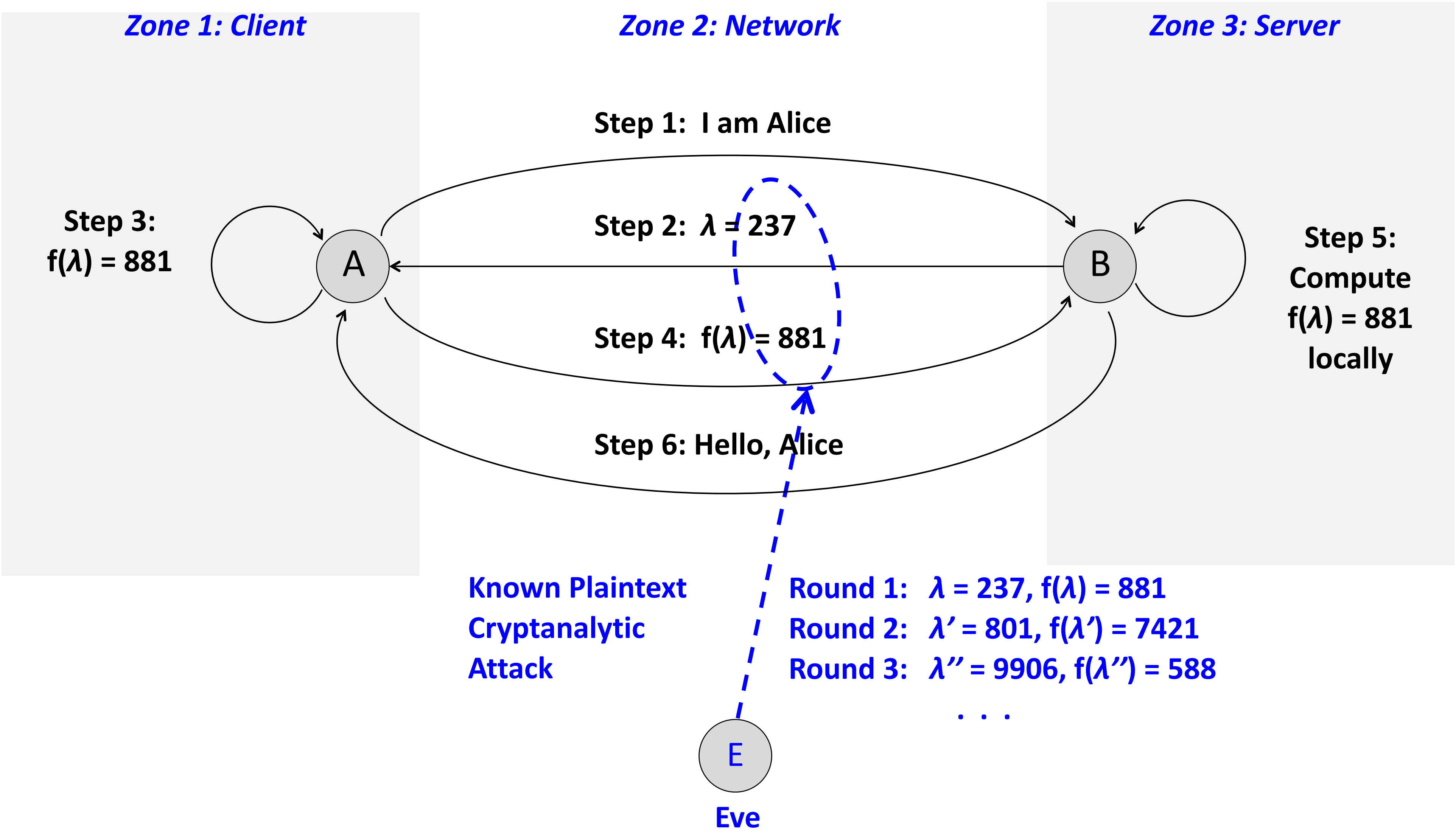


Embedded Function
f: integer -> integer

User	*Function*
A	**f**
C	**f ’**
G	**f ”**
. . .	. . .

Week 5

Handheld Authentication Device Protocol



Week 5

RSA SecurID One-Time Password (OTP) Algorithm

f: integer -> integer

λ: integer seed

t_0: initial time

t_C: current time

Δt: time interval

$n = (t_C – t_0) / Δt$

Week 5

RSA SecurID One-Time Password (OTP) Algorithm

f: integer -> integer

λ: integer seed

t_0: initial time

t_C: current time

Δt: time interval

$n = (t_C - t_0) / \Delta t$

Unique seed for each user

seed = λ

$t_0 = 0$ sec

$n = 0$

$t_C = 0$ sec

Week 5

RSA SecurID One-Time Password (OTP) Algorithm

f: integer -> integer

λ: integer seed

t_0: initial time

t_C: current time

Δt: time interval

$n = (t_C - t_0) / \Delta t$

Unique seed for each user

Encrypt seed once using f

seed = λ

$t_0 = 0$ sec

$n = 0$

OTP = $f^1(\lambda)$

$t_1 = 15$ sec

$n = 1$

$t_C = 15$ sec

$\Delta t = 15$ sec

Week 5

RSA SecurID One-Time Password (OTP) Algorithm

f: integer -> integer

λ: integer seed

t_0: initial time

t_C: current time

Δt: time interval

$n = (t_C - t_0) / \Delta t$

Unique seed for each user

seed = λ

$t_0 = 0$ sec

n = 0

Encrypt seed once using f

$OTP = f^1(\lambda)$

$t_1 = 15$ sec

n = 1

$\underbrace{\qquad\qquad}_{\Delta t = 15 \text{ sec}}$

Encrypt seed twice using f

$OTP = f^2(\lambda)$

$t_2 = 30$ sec

n = 2

$t_C = 30$ sec

Week 5

RSA SecurID Protocol

Step 1: I am Alice

A → B

f: integer -> integer

λ: integer seed

t_0: initial time

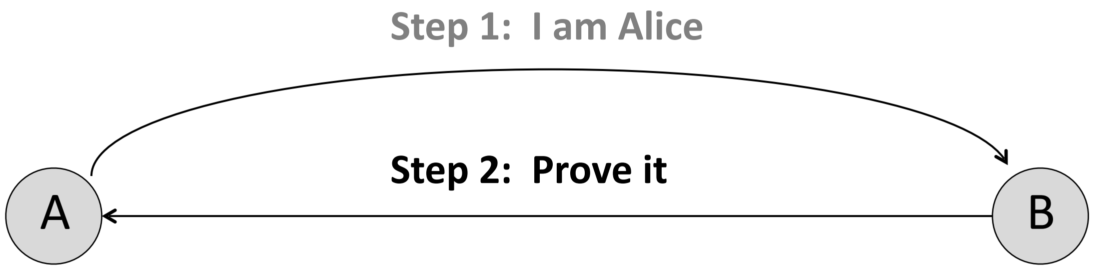
t_C: current time

Δt: time interval

$n = (t_C - t_0) / \Delta t$

User	*Information*
A	f: integer -> integer λ: integer seed t_0: initial time t_C: current time Δt: time interval $n = (t_C - t_0) / \Delta t$

RSA SecurID Protocol



f: integer -> integer

λ: integer seed

t_0: initial time

t_C: current time

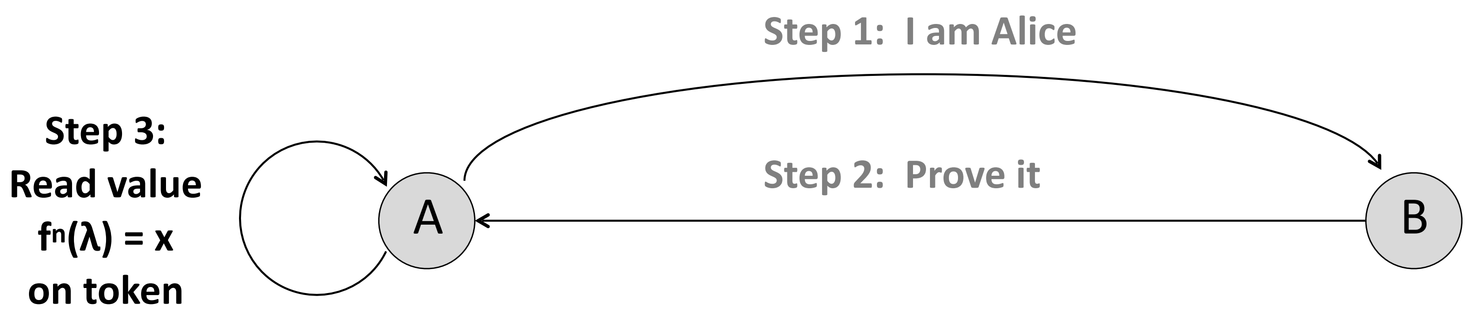
Δt: time interval

$n = (t_C – t_0) / \Delta t$

User	*Information*
A	f: integer -> integer λ: integer seed t_0: initial time t_C: current time Δt: time interval $n = (t_C – t_0) / \Delta t$

Week 5

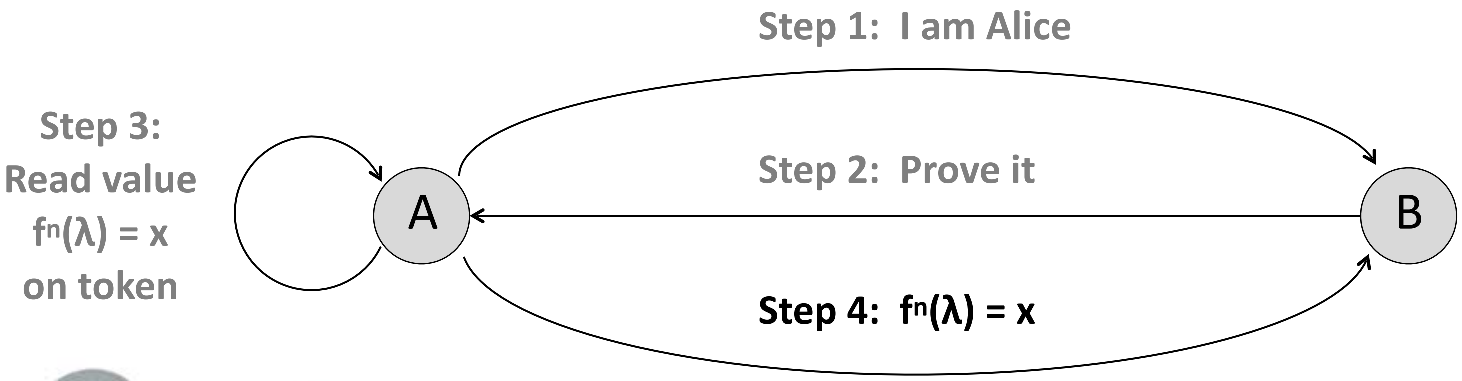
RSA SecurID Protocol



f: integer -> integer
λ: integer seed
t_0: initial time
t_C: current time
Δt: time interval
$n = (t_C - t_0) / \Delta t$

User	*Information*
A	f: integer -> integer λ: integer seed t_0: initial time t_C: current time Δt: time interval $n = (t_C - t_0) / \Delta t$

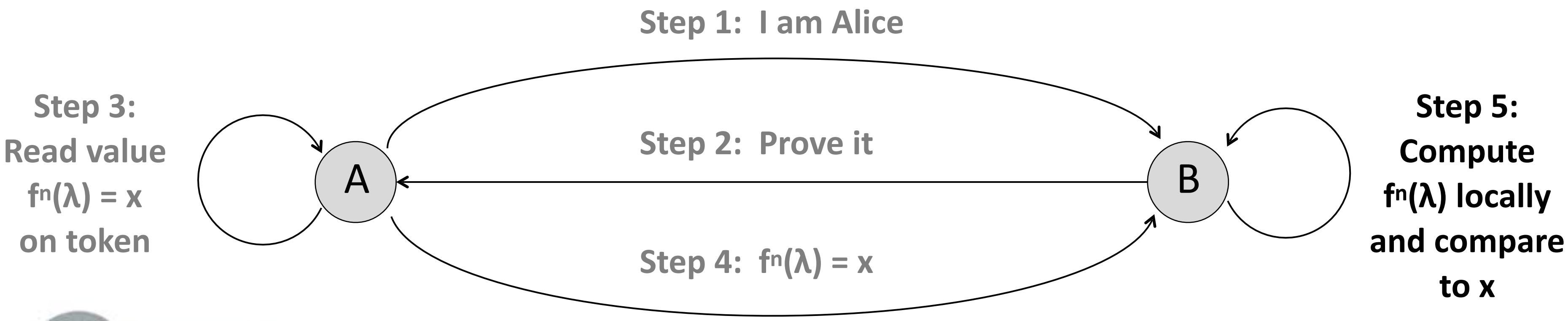
RSA SecurID Protocol



f: integer -> integer
λ: integer seed
t_0: initial time
t_C: current time
Δt: time interval
$n = (t_C – t_0) / \Delta t$

User	***Information***
A	f: integer -> integer
	λ: integer seed
	t_0: initial time
	t_C: current time
	Δt: time interval
	$n = (t_C – t_0) / \Delta t$

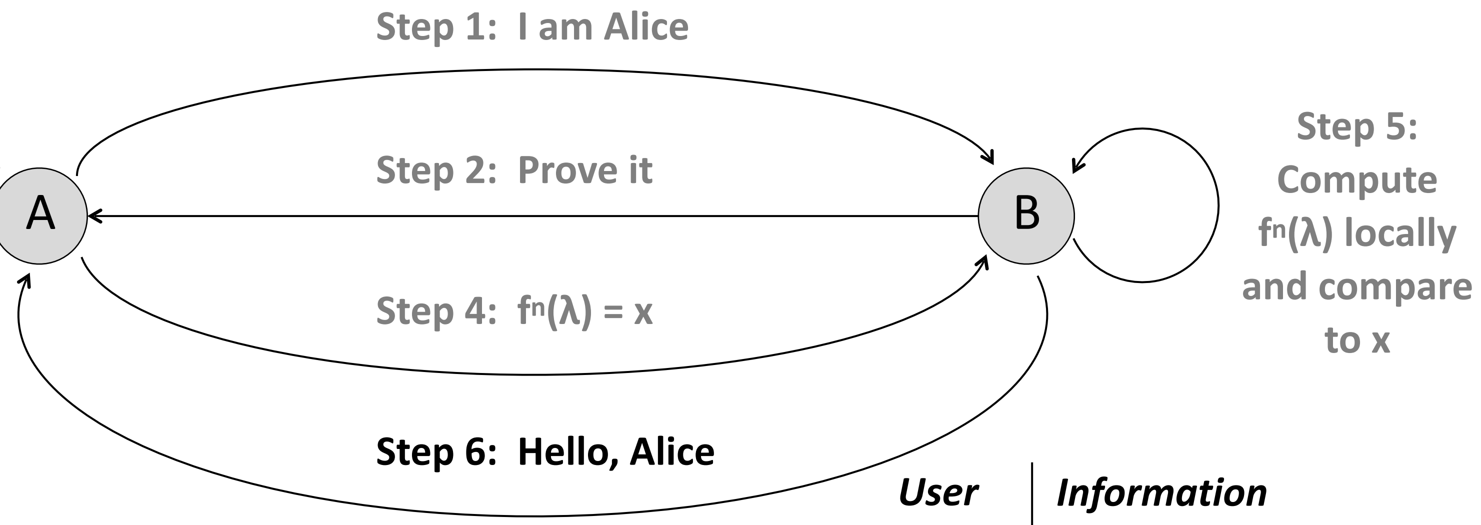
RSA SecurID Protocol



f: integer -> integer
λ: integer seed
t_0: initial time
t_C: current time
Δt: time interval
$n = (t_C - t_0) / \Delta t$

User	*Information*
A	f: integer -> integer λ: integer seed t_0: initial time t_C: current time Δt: time interval $n = (t_C - t_0) / \Delta t$

RSA SecurID Protocol

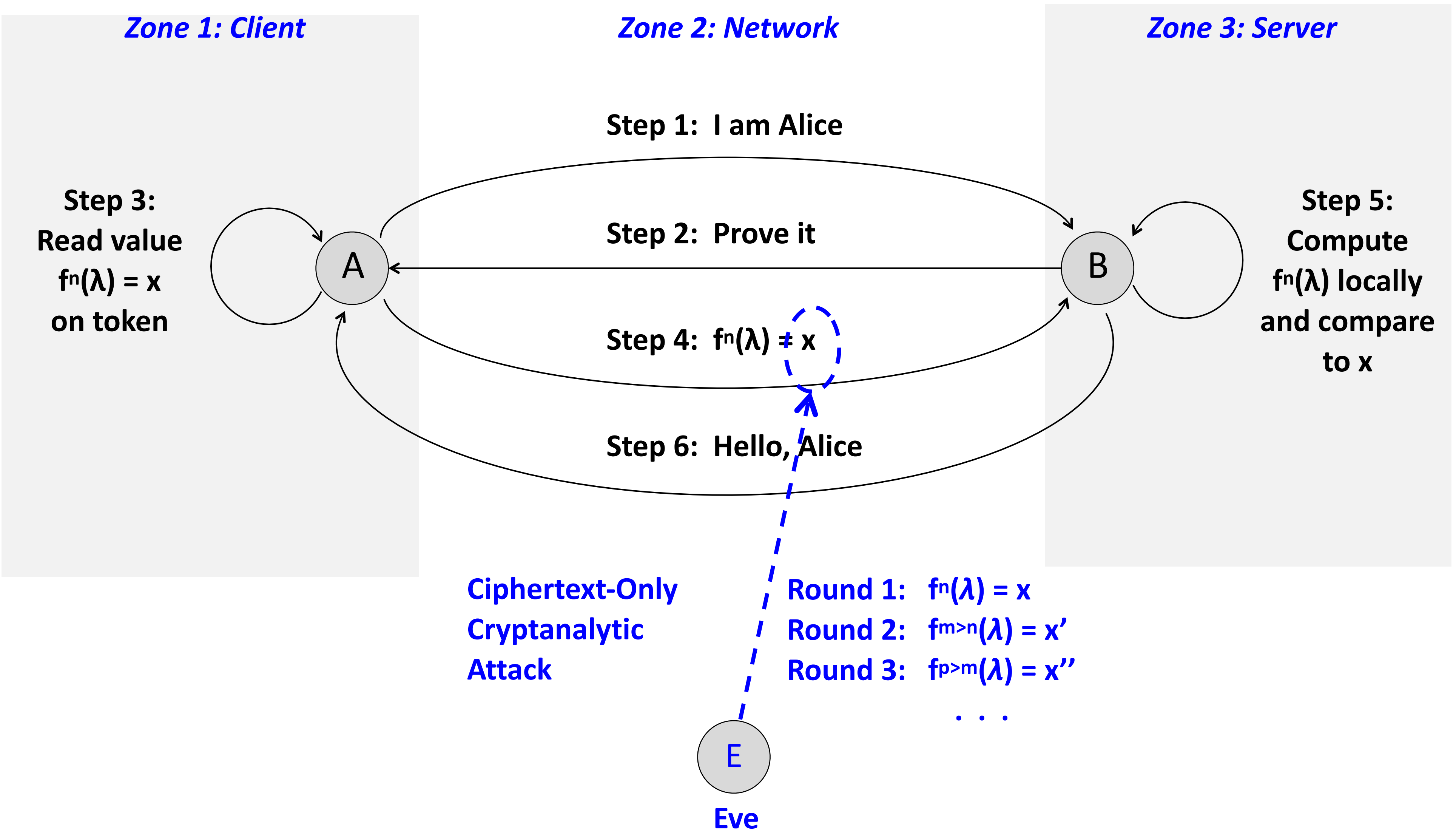


f: integer -> integer
λ: integer seed
t_0: initial time
t_C: current time
Δt: time interval
$n = (t_C - t_0) / \Delta t$

User	*Information*
A	f: integer -> integer λ: integer seed t_0: initial time t_C: current time Δt: time interval $n = (t_C - t_0) / \Delta t$

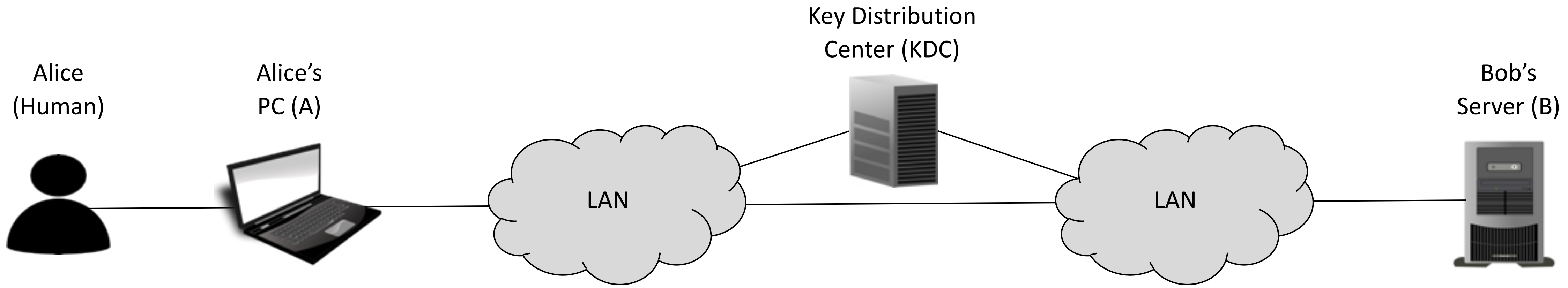
Week 5

RSA SecurID Protocol



Week 5

Kerberos: A Complex Solution to a Simple Password Problem



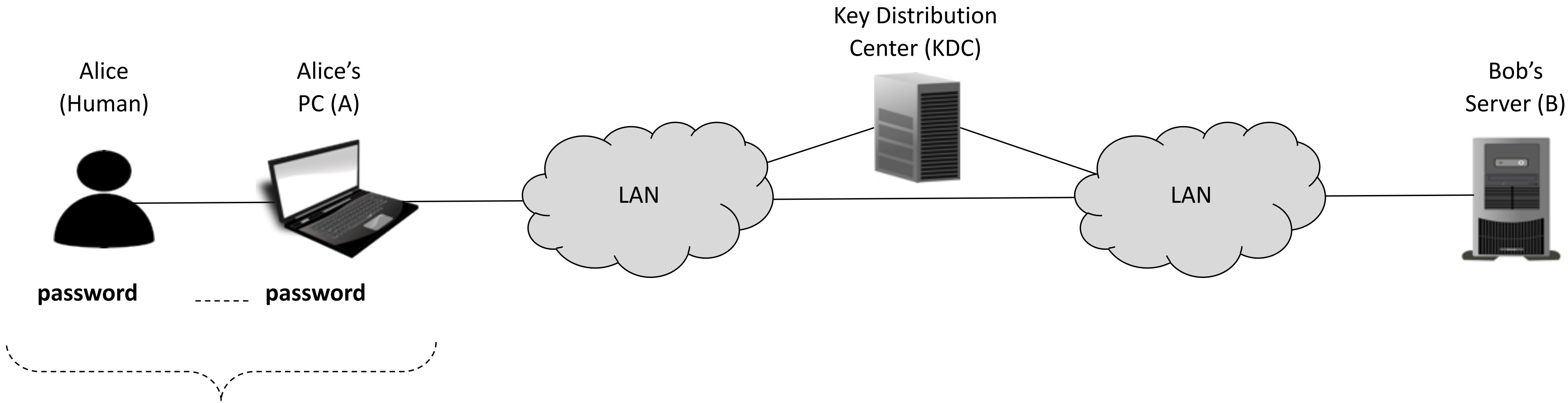
Basic Kerberos Concept:

- Invented at MIT in 1980's as part of Project Athena
- Goal is that Alice (client) can authenticate to Bob (server) without using a password on the local area network (LAN);
- Key Distribution Center (KDC) enables this process using conventional cryptography (i.e., no public key technology)

Week 5

Kerberos – Preconditions

Infrastructure Preconditions:



Kerberos password set up for Alice to log into her PC (Never used over any LAN)

Week 5

Kerberos – Keys

Infrastructure Preconditions:

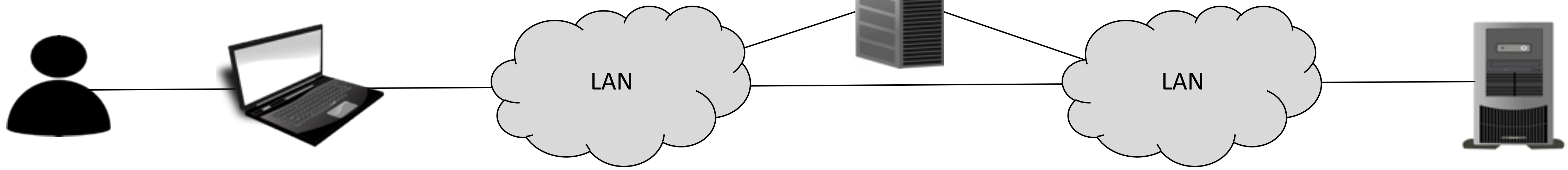
K_{KDC} : KDC's Encryption Key (Created by KDC)

Key Distribution Center (KDC)

Alice (Human)

Alice's PC (A)

Bob's Server (B)



password ------ **password**

Key (Issued by KDC)
K_A : Alice's Encryption

K_B : Bob's Encryption Key (Issued by KDC)

Kerberos KDC creates and issues three cryptographic keys: $\mathbf{K_A}$, $\mathbf{K_B}$, $\mathbf{K_{KDC}}$

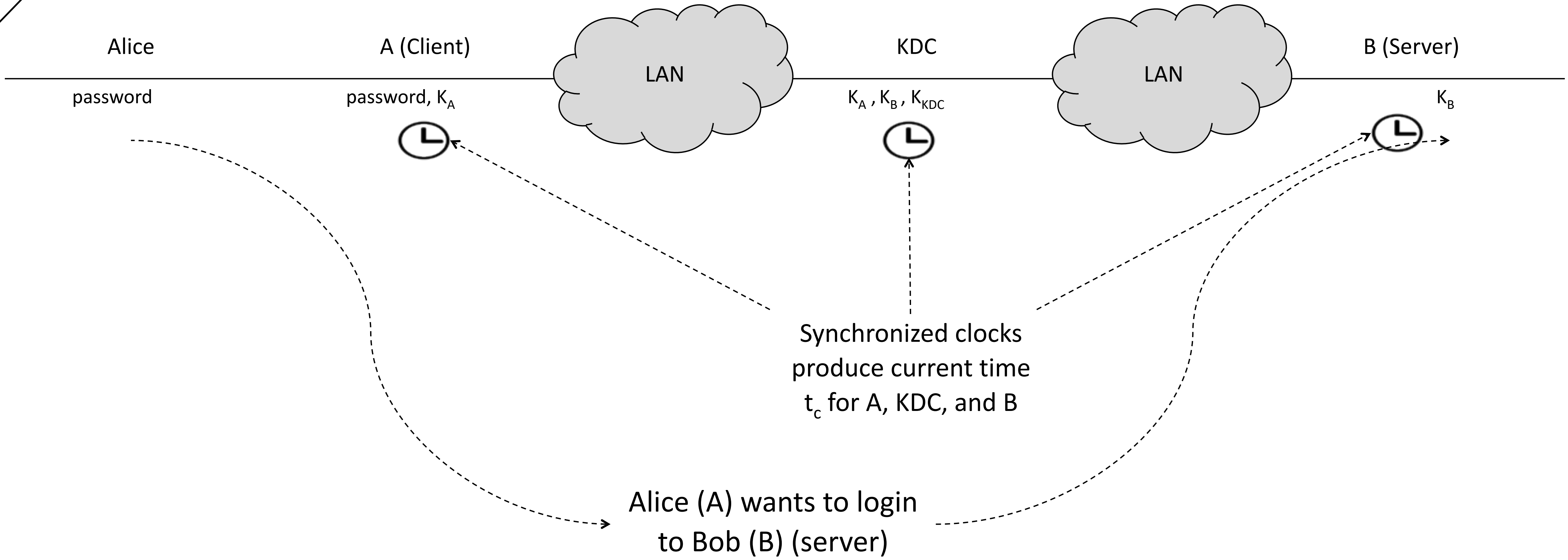
$$\{\{ m \}_{K_A}\}_{K_A} = m \qquad \{\{ m \}_{K_B}\}_{K_B} = m \qquad \{\{ m \}_{K_{KDC}}\}_{K_{KDC}} = m$$

Encrypt

Decrypt

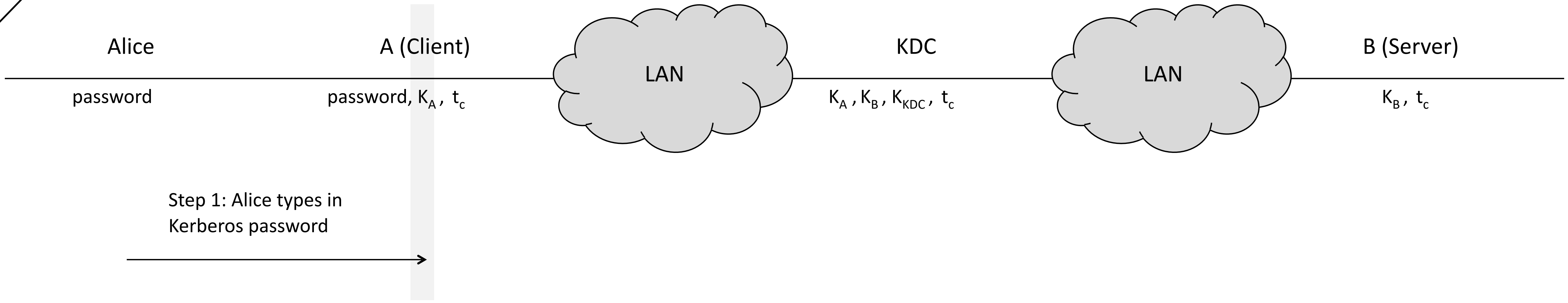
Week 5

Kerberos – Clocks



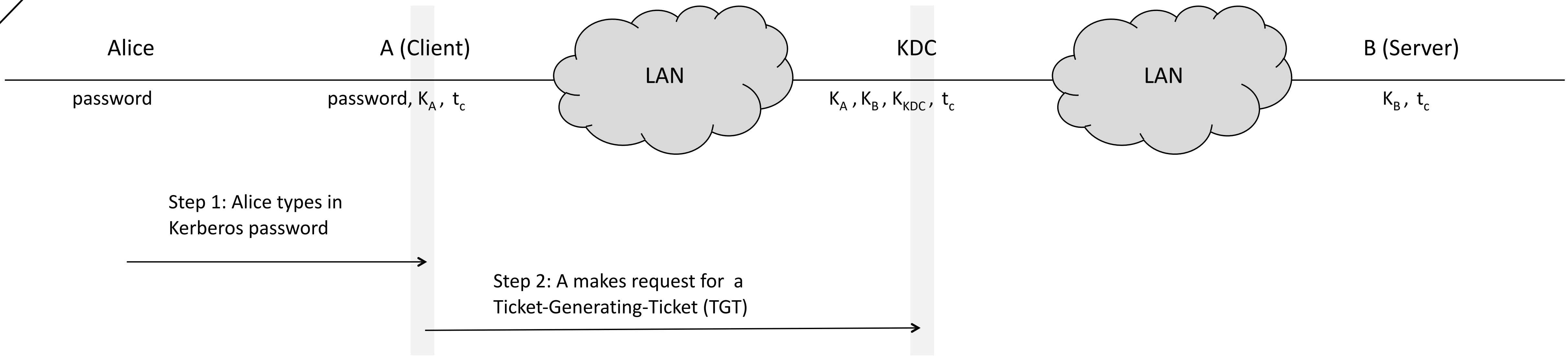
Week 5

Kerberos Step 1: Type in Local Password



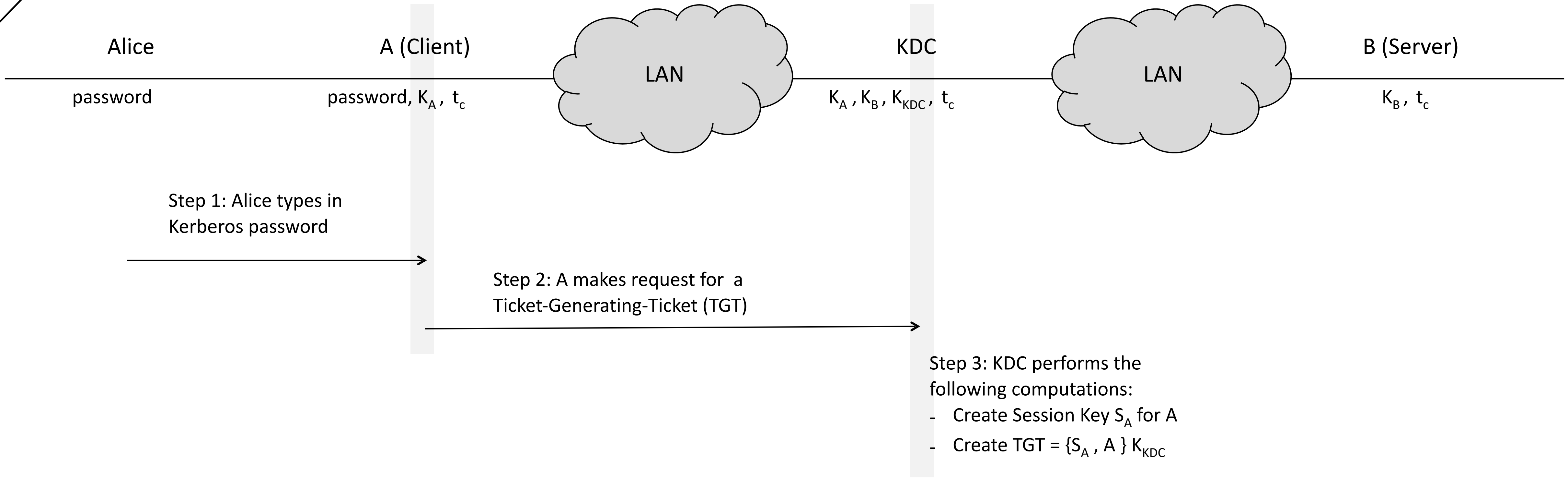
Week 5

Kerberos Step 2: Request TGT and Session Key



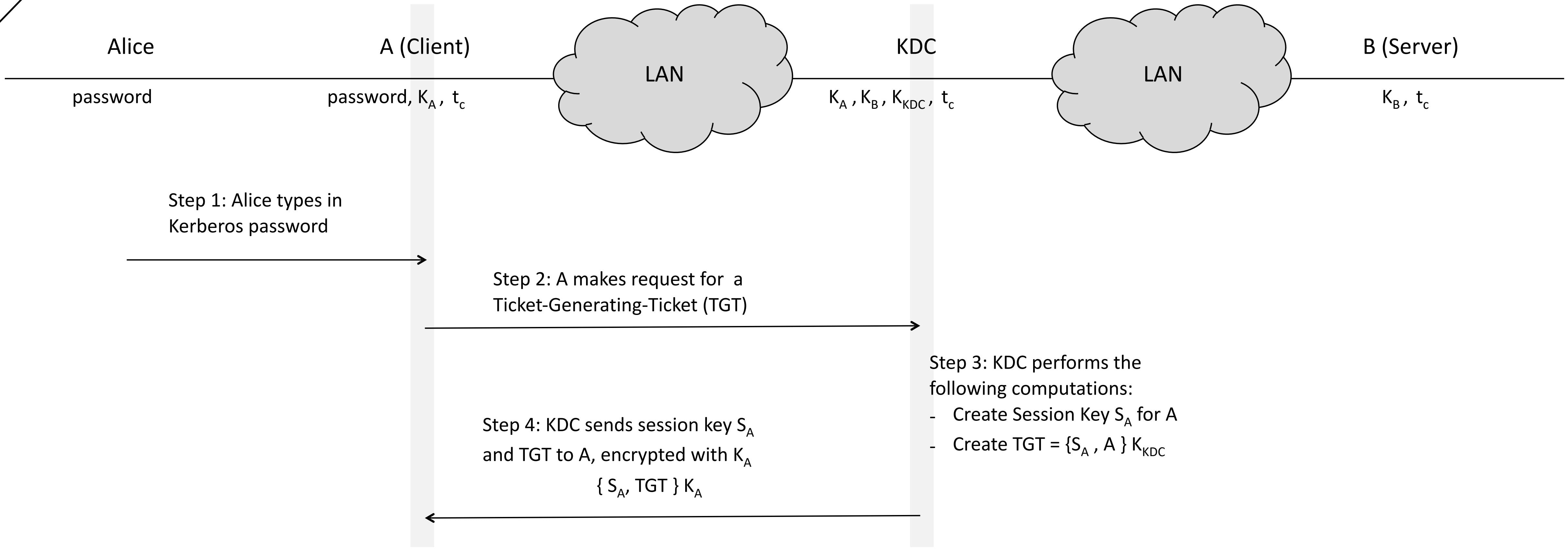
Week 5

Kerberos Step 3: Create Session Key and TGT



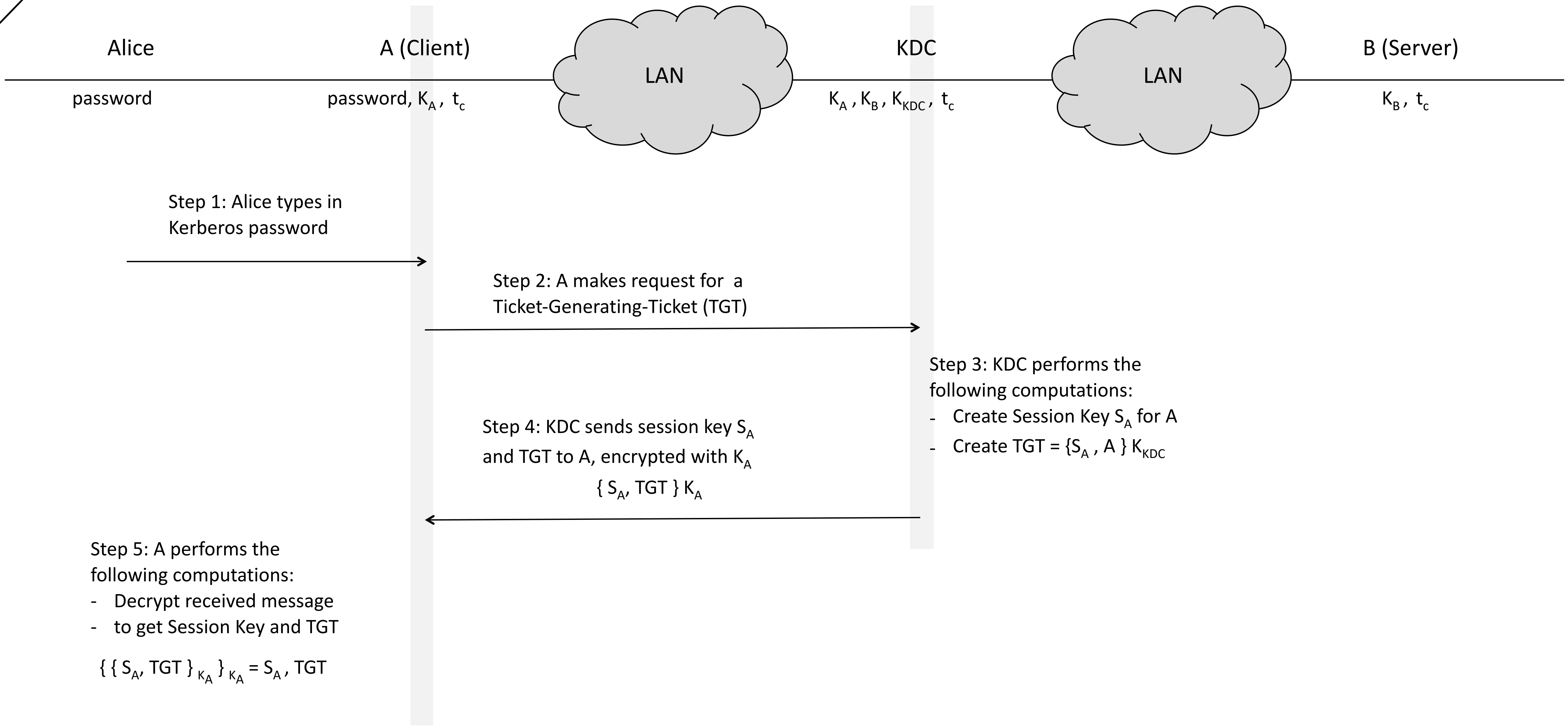
Week 5

Kerberos Step 4: Provide Session Key and TGT



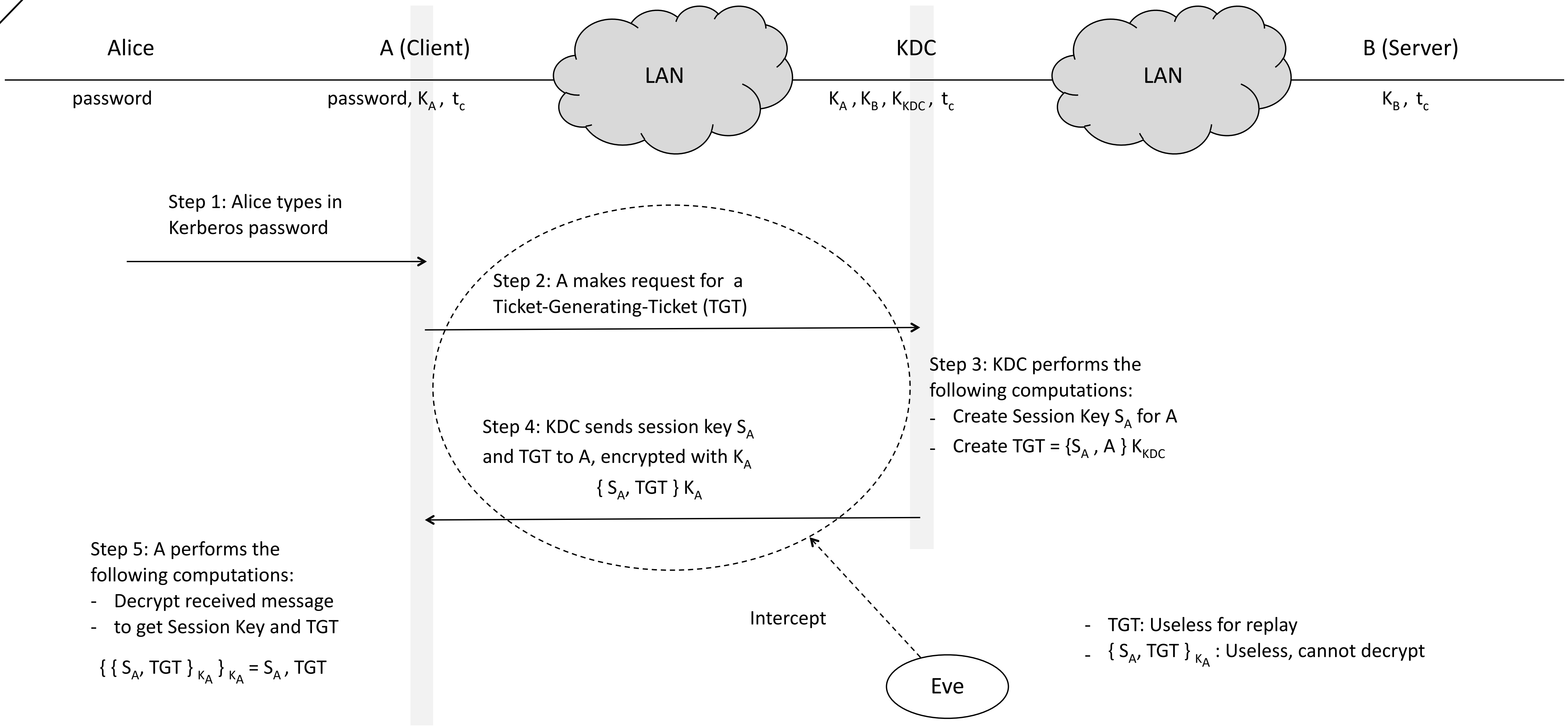
Week 5

Kerberos Step 5: Decrypt Session Key and Store TGT



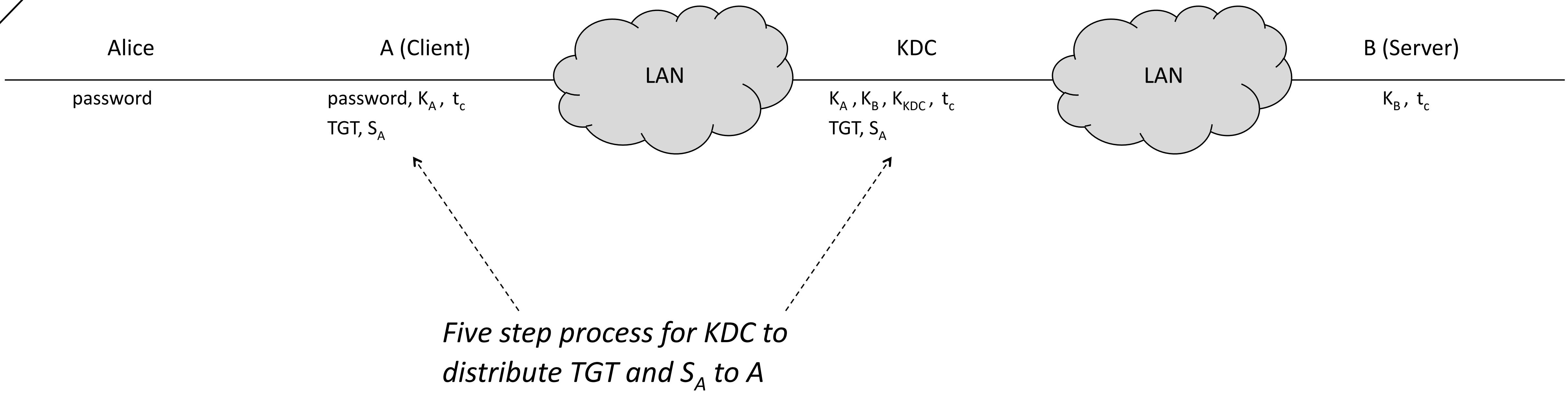
Week 5

Kerberos – Through Five Steps: Eve Cannot Hack

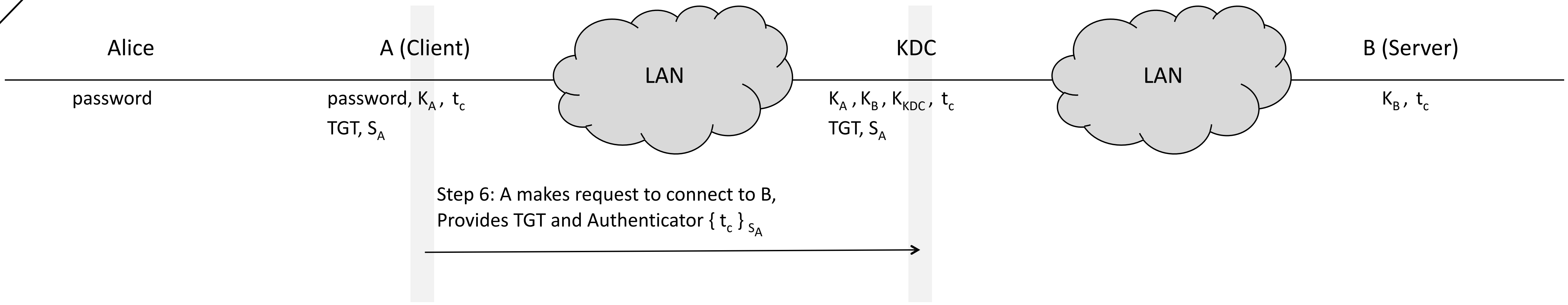


Week 5

Kerberos – Result of Five Steps

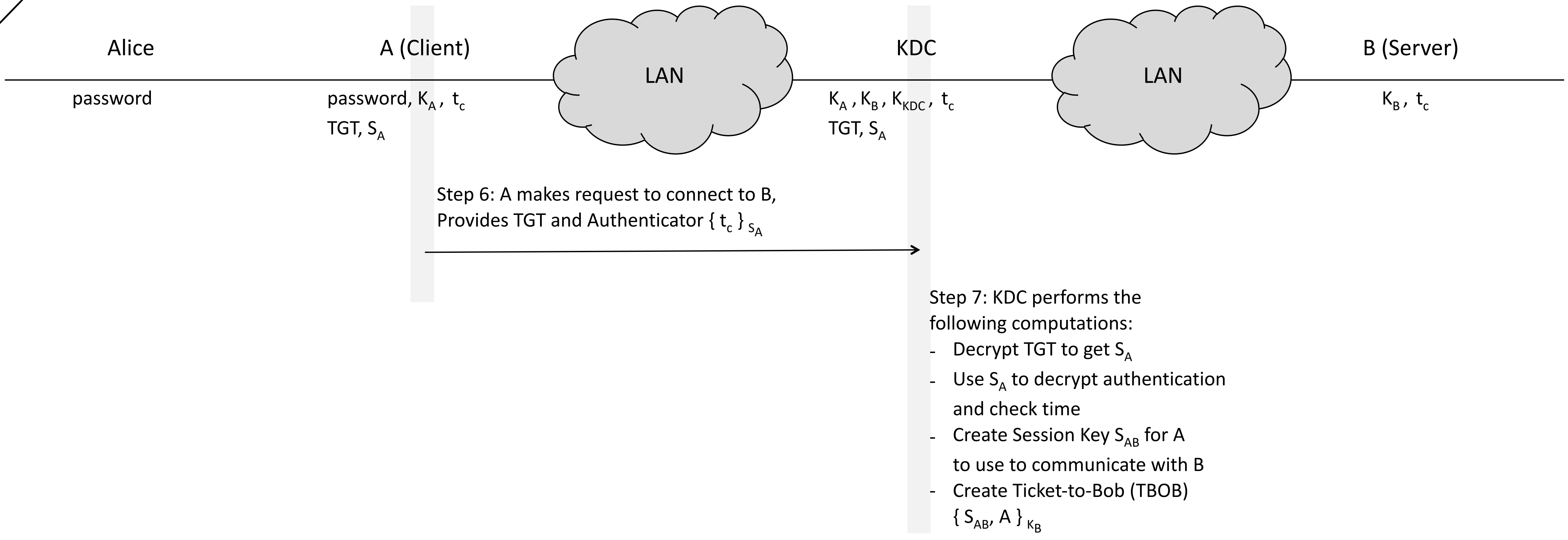


Kerberos Step 6: Request Login to B



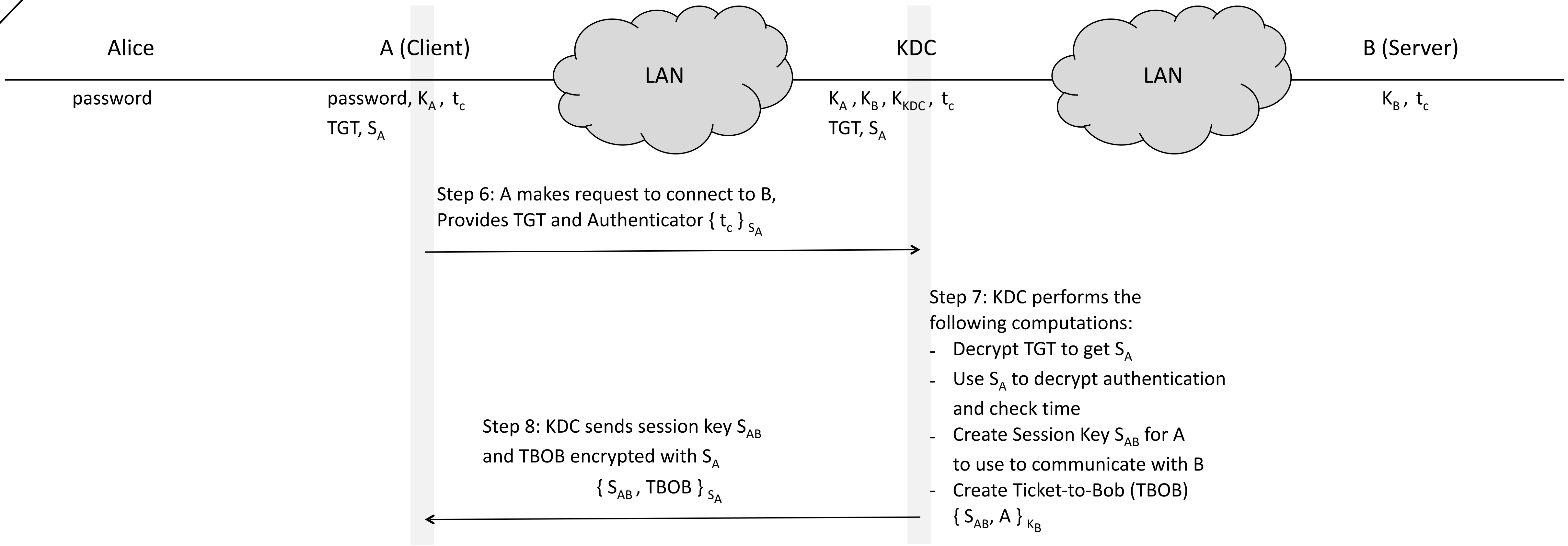
Week 5

Kerberos Step 7: Create Session Key and Ticket to Bob



Week 5

Kerberos Step 8: Send Session Key and Ticket to Bob



Week 5

Kerberos Step 9: Decrypt Session Key and Store Ticket to Bob

Alice

password

A (Client)

password, K_A , t_c

TGT, S_A

LAN

KDC

K_A , K_B , K_{KDC} , t_c

TGT, S_A

LAN

B (Server)

K_B , t_c

Step 6: A makes request to connect to B,
Provides TGT and Authenticator $\{ t_c \}_{S_A}$

Step 7: KDC performs the following computations:

- Decrypt TGT to get S_A
- Use S_A to decrypt authentication and check time
- Create Session Key S_{AB} for A to use to communicate with B
- Create Ticket-to-Bob (TBOB) $\{ S_{AB}, A \}_{K_B}$

Step 8: KDC sends session key S_{AB} and TBOB encrypted with S_A

$\{ S_{AB} , \text{TBOB} \}_{S_A}$

Step 9: A performs the following computations:

- Decrypt received message
- to get Session Key and TBOB

$\{ \{ S_{AB}, \text{TBOB} \}_{S_A} \}_{S_A} = S_{AB} , \text{TBOB}$

Kerberos – Through Nine Steps: Eve Cannot Hack

Alice — A (Client) — LAN — KDC — LAN — B (Server)

Alice: password

A (Client): password, K_A, t_c
TGT, S_A

KDC: K_A, K_B, K_{KDC}, t_c
TGT, S_A

B (Server): K_B, t_c

Step 6: A makes request to connect to B,
Provides TGT and Authenticator $\{ t_c \}_{S_A}$

Step 7: KDC performs the following computations:

- Decrypt TGT to get S_A
- Use S_A to decrypt authentication and check time
- Create Session Key S_{AB} for A to use to communicate with B
- Create Ticket-to-Bob (TBOB) $\{ S_{AB}, A \}_{K_B}$

Step 8: KDC sends session key S_{AB} and TBOB encrypted with S_A

$\{ S_{AB}, TBOB \}_{S_A}$

Step 9: A performs the following computations:

- Decrypt received message
- to get Session Key and TBOB

$\{ \{ S_{AB}, TBOB \}_{S_A} \}_{S_A} = S_{AB}, TBOB$

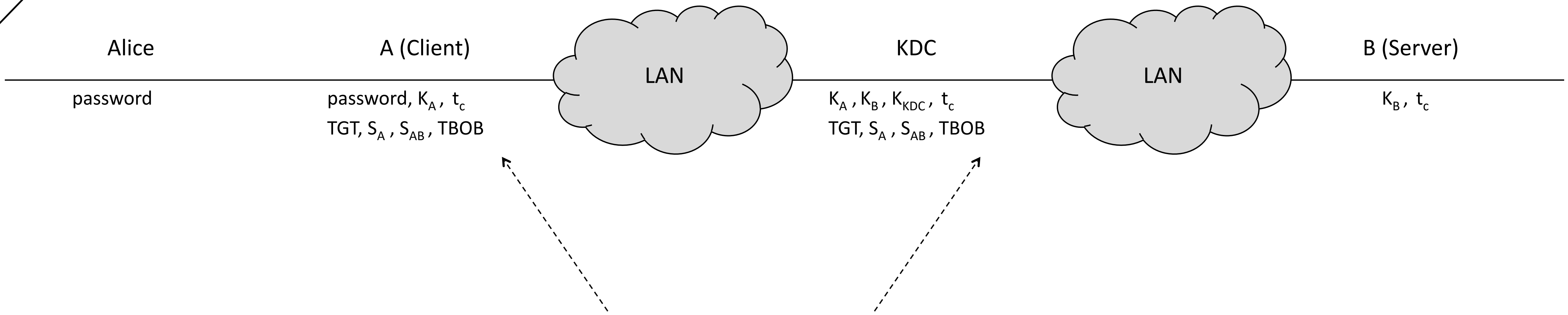
Intercept

Eve

- TGT: Useless for replay
- Authenticator cannot be replayed (time staleness)
- $\{ S_{AB}, TBOB \}_{S_A}$: Useless, cannot decrypt

Week 5

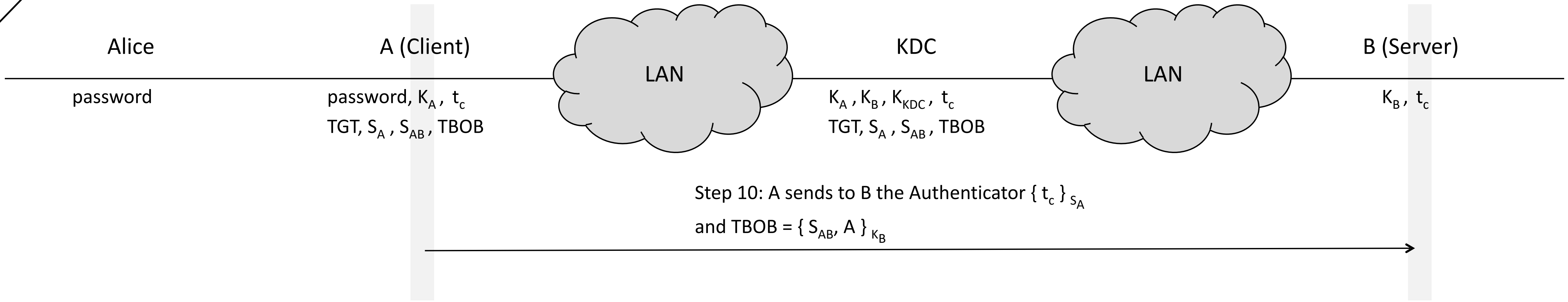
Kerberos – Result of Nine Steps



Four step process for KDC to distribute TBOB and S_{AB} to A

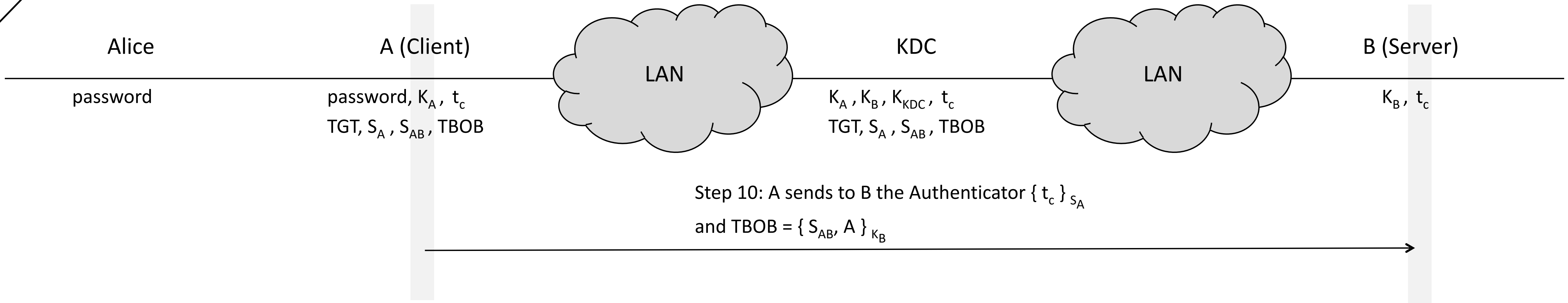
Week 5

Kerberos Step 10: Send Bob the Ticket to Bob



Week 5

Kerberos Step 11: Decrypt Ticket to Bob and Check Time

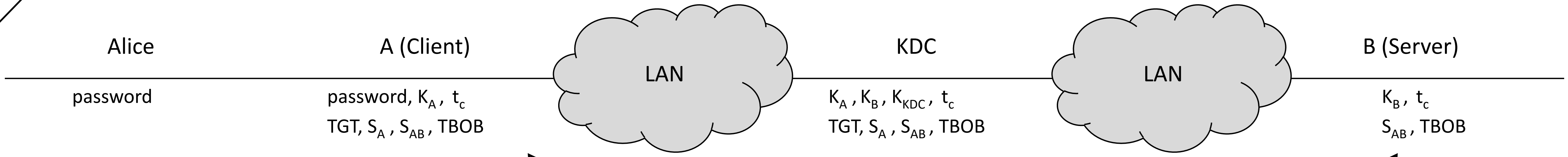


Step 11: B performs the following computations:

- Decrypt TBOB to get S_{AB}
- Use S_{AB} to decrypt authentication and check time

Week 5

Kerberos – Result of Eleven Steps



Two step process (plus nonce messages) for A to use TBOB to get S_{AB} to B

Week 5

Kerberos – Realms