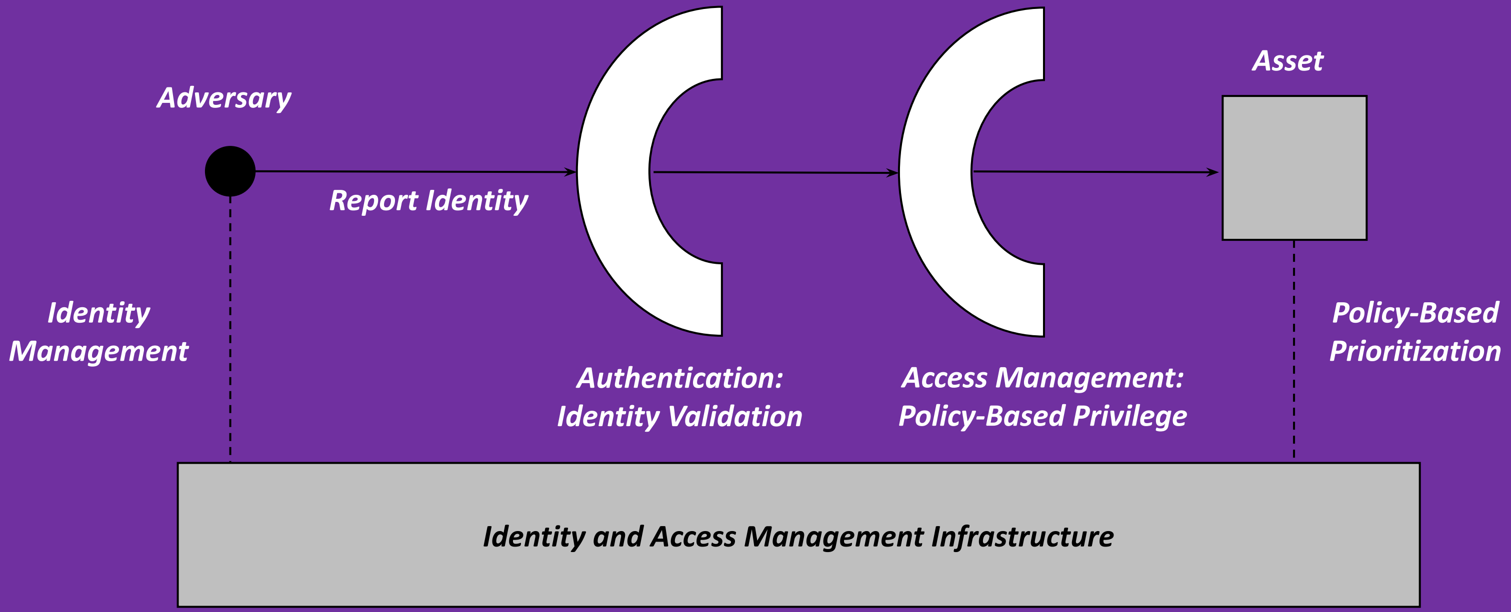
Week 7: Authentication and Identity/Access Management

[*] Slides based upon materials maintained by Dr. Edward G. Amoroso (eamoroso@tag-cyber.com)

Week 7

Safeguard: Authentication



NYU

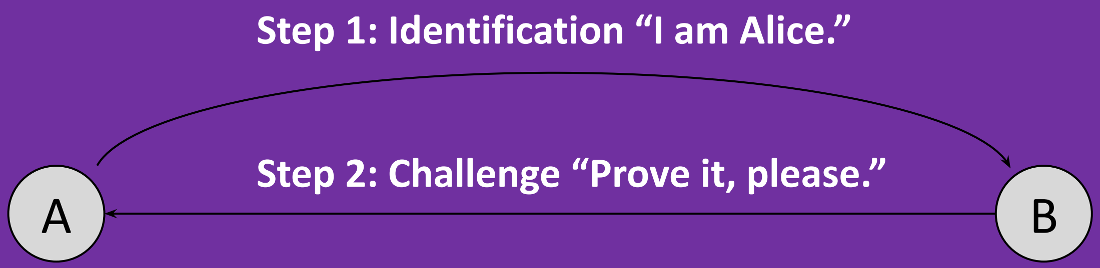
Week 7

Authentication Schema

Client A – Server B: “Client Authentication”

Client B – Server A: “Server Authentication”

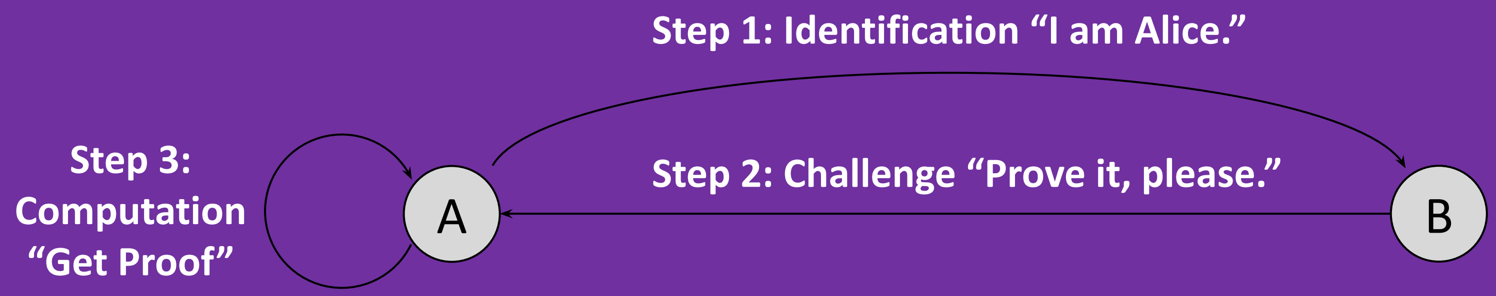
Authentication Schema



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

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

Authentication Schema

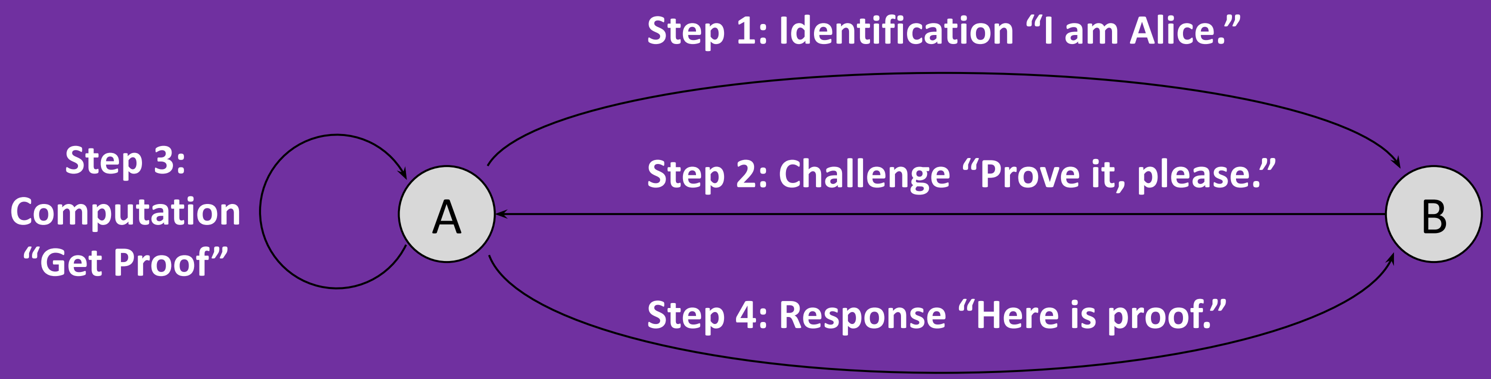


Computation might involve simple look-up/locate process (e.g., passwords)

Computation might be more deliberate mathematical operation on domain value

Week 7

Authentication Schema



Types of Proof:

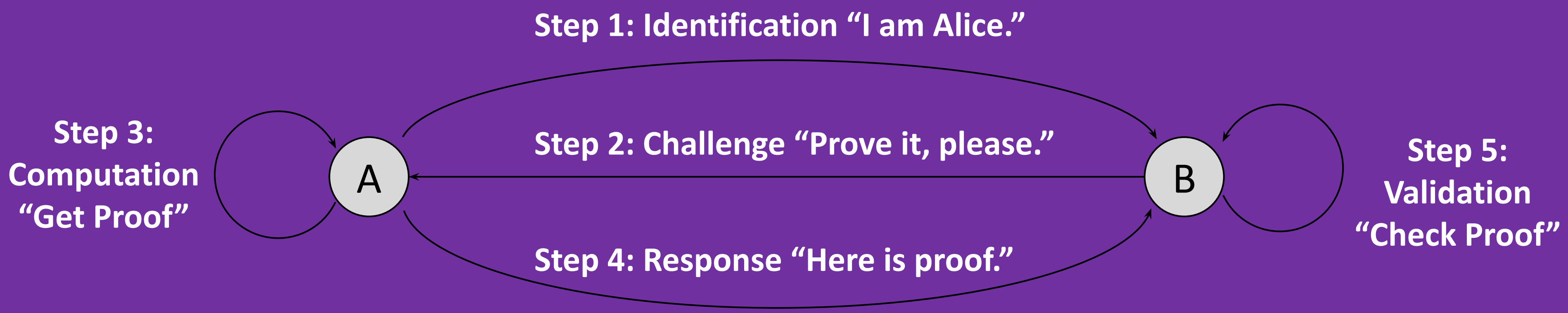
"Something You Know" – Passwords
"Something You Are" – Biometrics
"Something You Have" – Token
"Somewhere You Are" – Location

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

NYU

Week 7

Authentication Schema



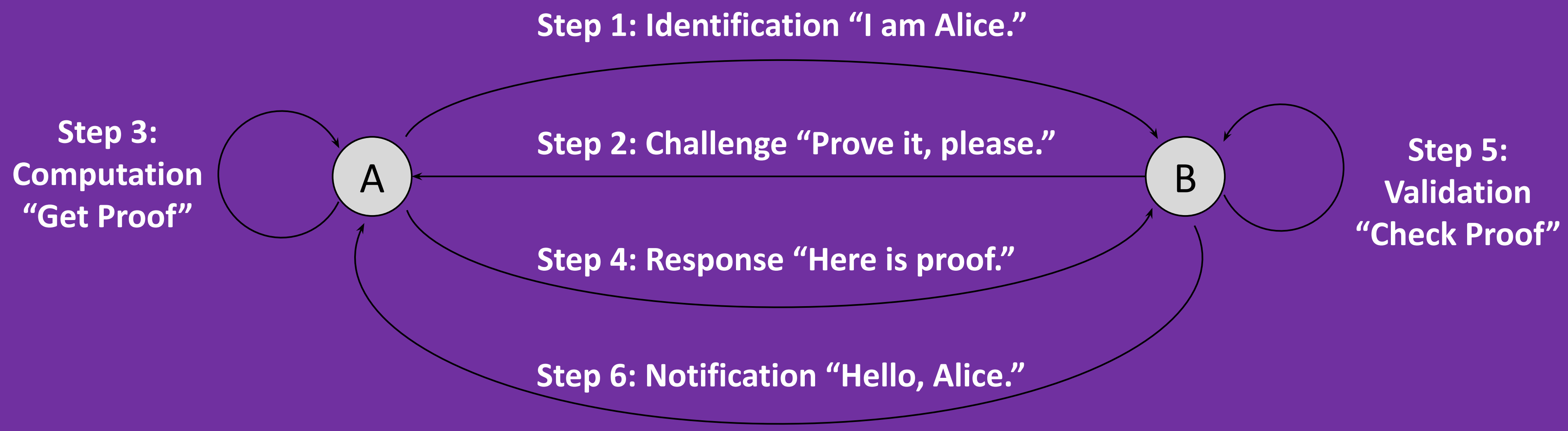
Validation might involve simple look-up/locate process (e.g., passwords)

Validation might be more deliberate mathematical operation on domain value

NYU

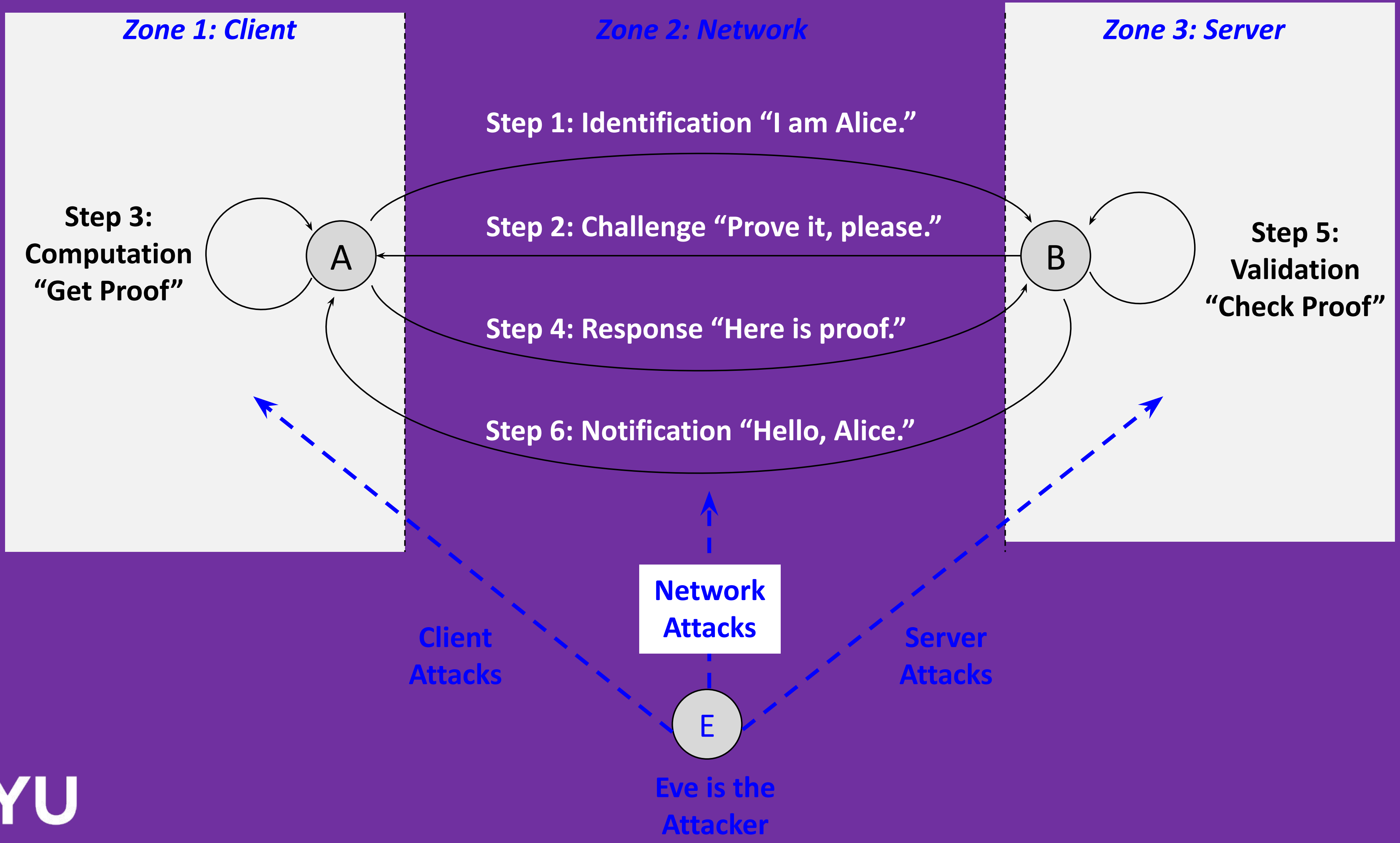
Week 7

Authentication Schema



NYU

Authentication Schema



Week 7

Handheld Authentication Device

A

B

Embedded Function
f: integer -> integer

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

NYU

Week 7

Handheld Authentication Device

Protocol Implementation

A

B

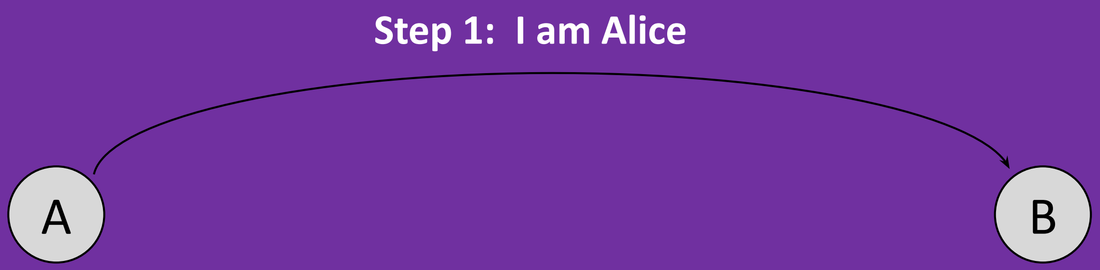
Protocol Infrastructure

Embedded Function
f: integer -> integer

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

Week 7

Handheld Authentication Device



Embedded Function
f: integer -> integer

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

Week 7

Handheld Authentication Device

Step 1: I am Alice

Step 2: λ = 237

A

B

Randomly selected integer λ

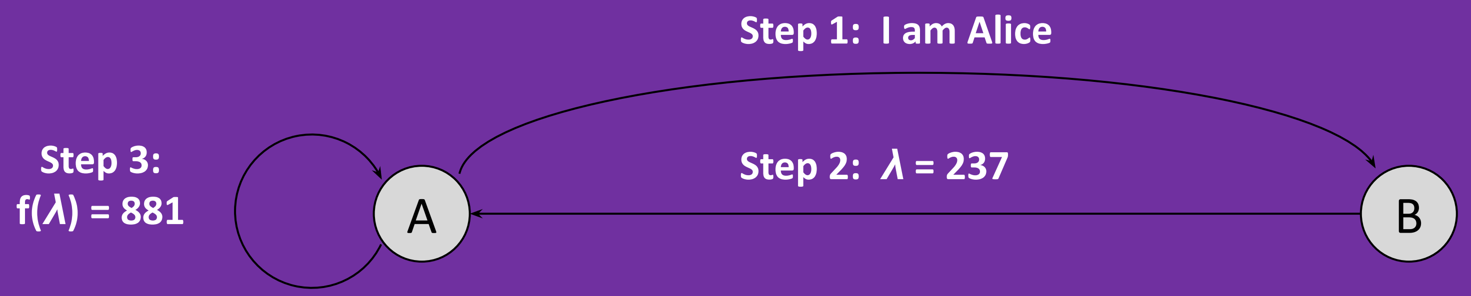
Embedded Function
f: integer -> integer

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

NYU

Week 7

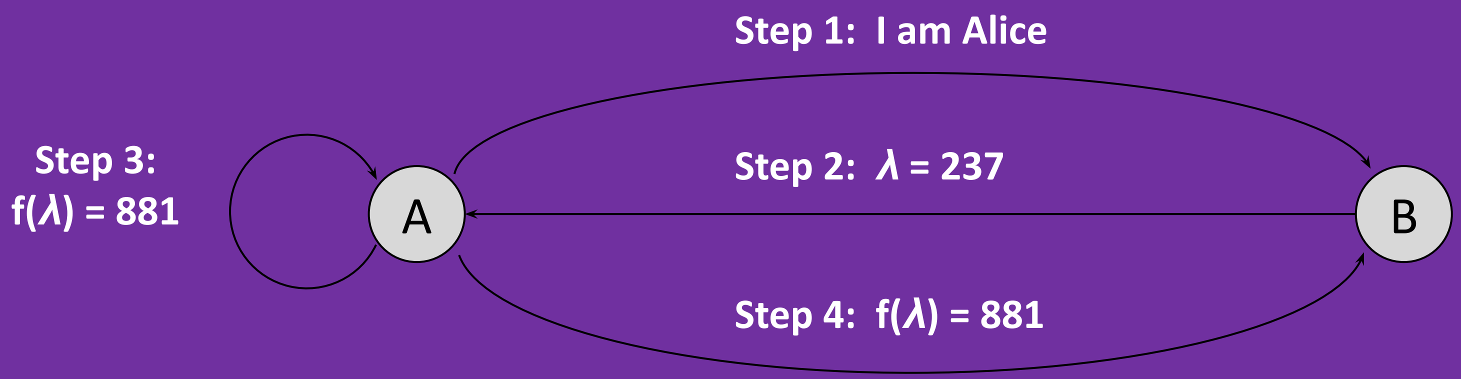
Handheld Authentication Device



Embedded Function
f: integer -> integer

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

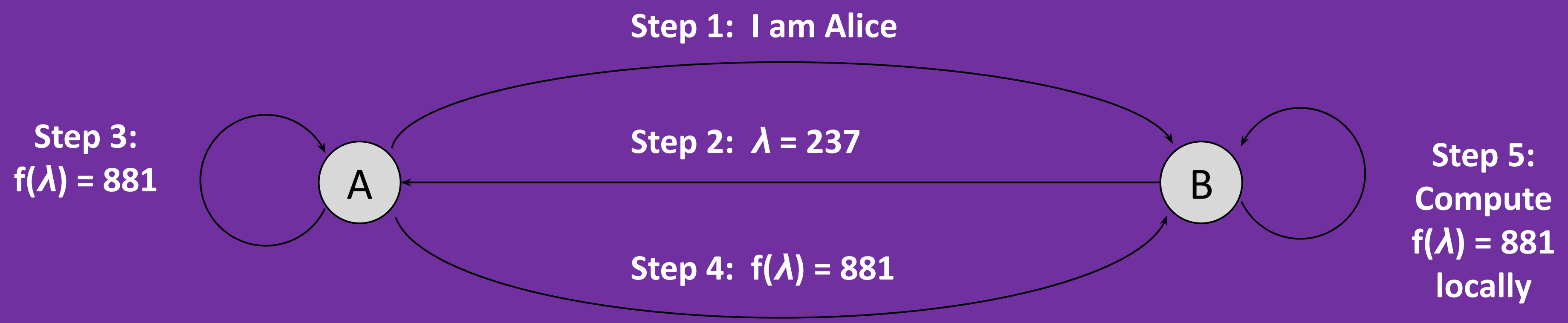
Handheld Authentication Device



Embedded Function
f: integer -> integer

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

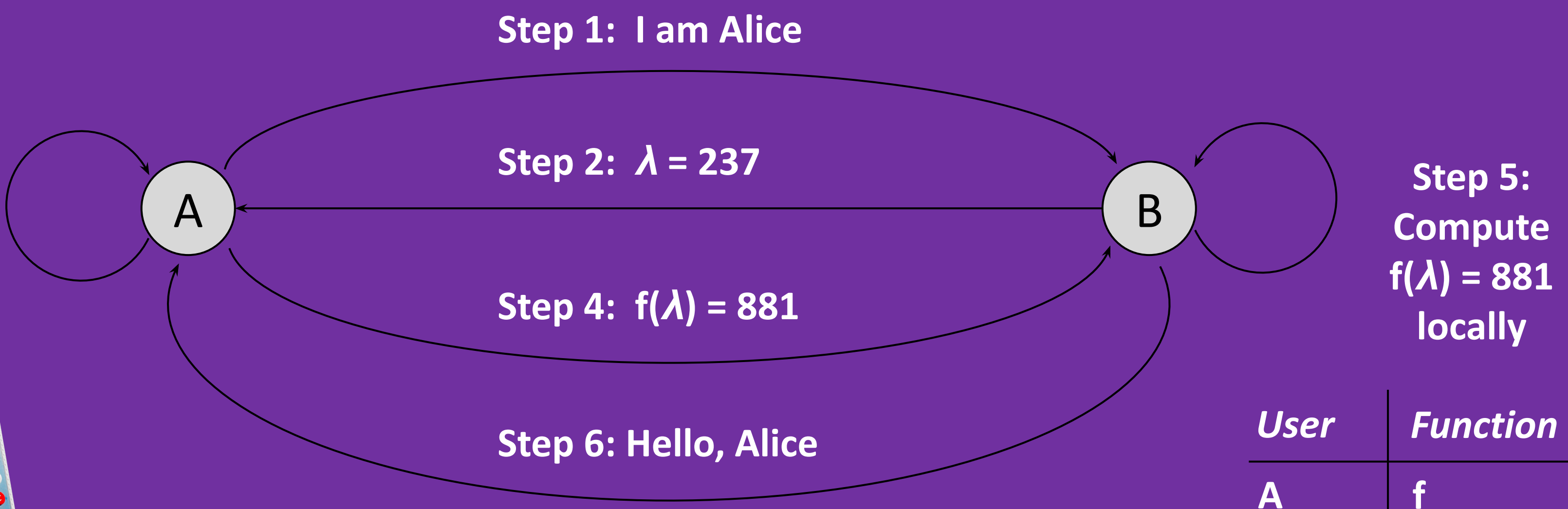
Handheld Authentication Device



Embedded Function
f: integer -> integer

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

Handheld Authentication Device

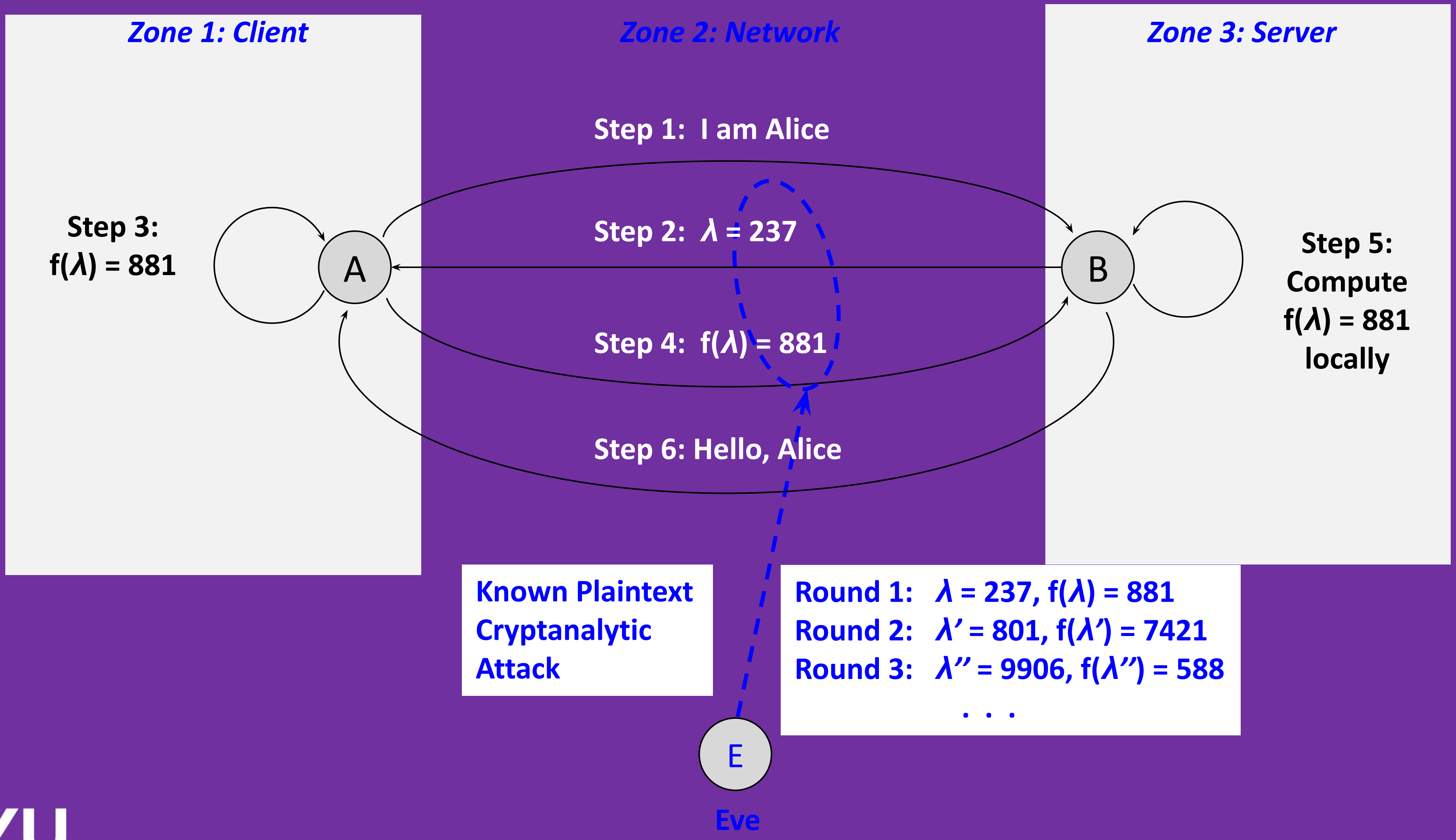


Embedded Function
f: integer -> integer

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

Week 7

Handheld Authentication Device Protocol



NYU

Week 7

RSA SecurID One-Time Password (OTP) Algorithm

f: integer -> integer

λ: integer seed

t_0: initial time

t_C: current time

Δt: time interval

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

Week 7

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

NYU

Week 7

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

OTP = $f^1(\lambda)$

t_0 = 0 sec

n = 0

t_1 = 15 sec

n = 1

t_C = 15 sec

Δt = 15 sec

NYU

Week 7

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

Encrypt seed twice using f

OTP = $f^2(\lambda)$

t_2 = 30 sec

n = 2

t_C = 30 sec

Δt = 15 sec

NYU

Week 7

RSA SecurID Protocol

Step 1: I am Alice

A $\rightarrow$ B

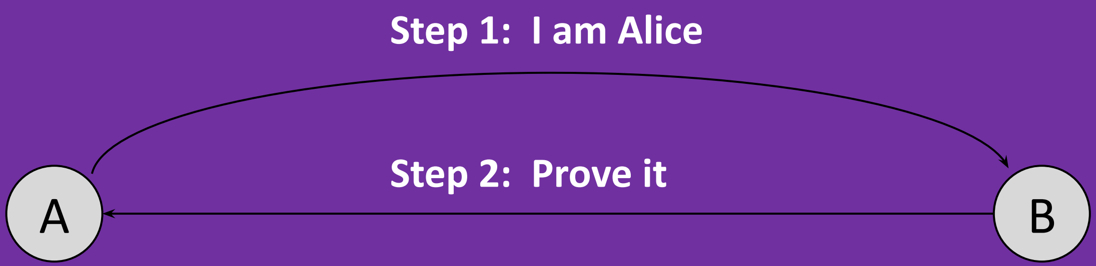
f: integer -> integer
λ: integer seed
t_0: initial time
t_C: current time
Δt: time interval
$n = (t_C - t_0) / \Delta t$

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

NYU

Week 7

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

NYU

RSA SecurID Protocol

Step 1: I am Alice

Step 2: Prove it

Step 3:
Read value
$f^n(\lambda) = x$
on token

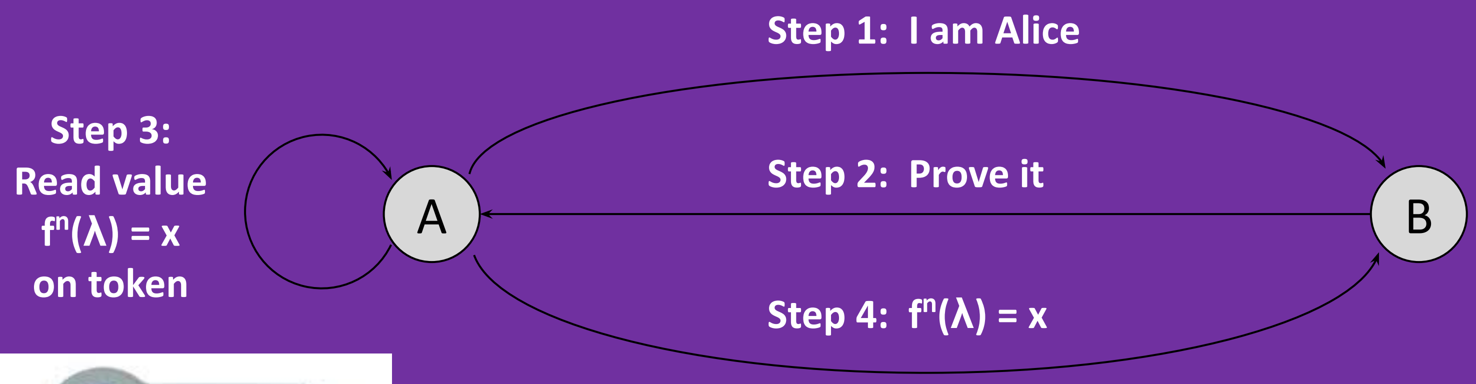
A

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

NYU

Week 7

RSA SecurID Protocol

Step 1: I am Alice

Step 2: Prove it

Step 3: Read value $f^n(\lambda) = x$ on token

Step 4: $f^n(\lambda) = x$

Step 5: Compute $f^n(\lambda)$ locally and compare to x

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

NYU

Week 7

RSA SecurID Protocol

Step 1: I am Alice

Step 2: Prove it

Step 3: Read value $f^n(\lambda) = x$ on token

Step 4: $f^n(\lambda) = x$

Step 5: Compute $f^n(\lambda)$ locally and compare to x

Step 6: Hello, Alice

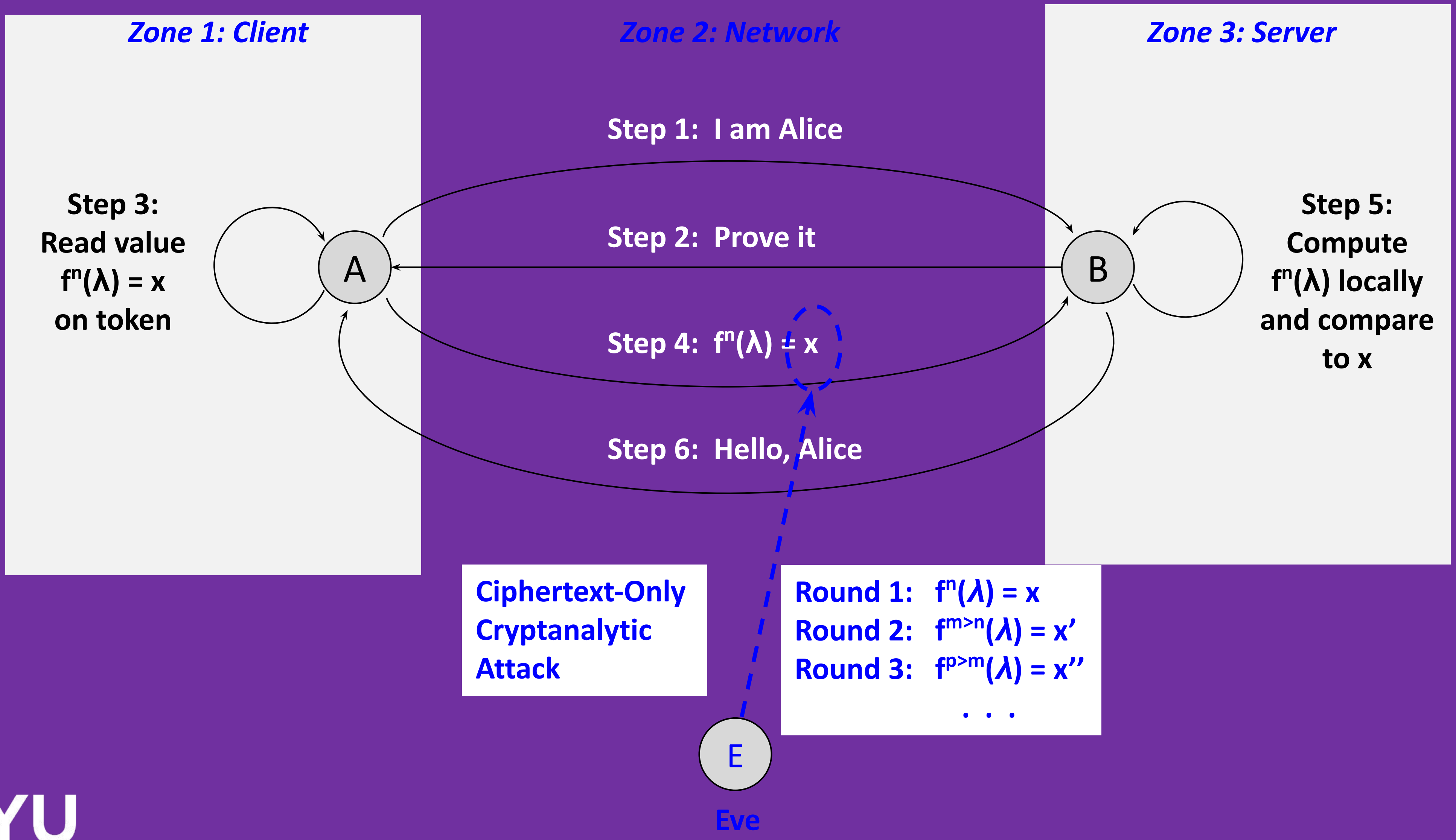
f: integer -> integer
λ: integer seed
t_0: initial time
t_C: current time
Δt: time interval
$n = (t_C - t_0) / \Delta t$

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

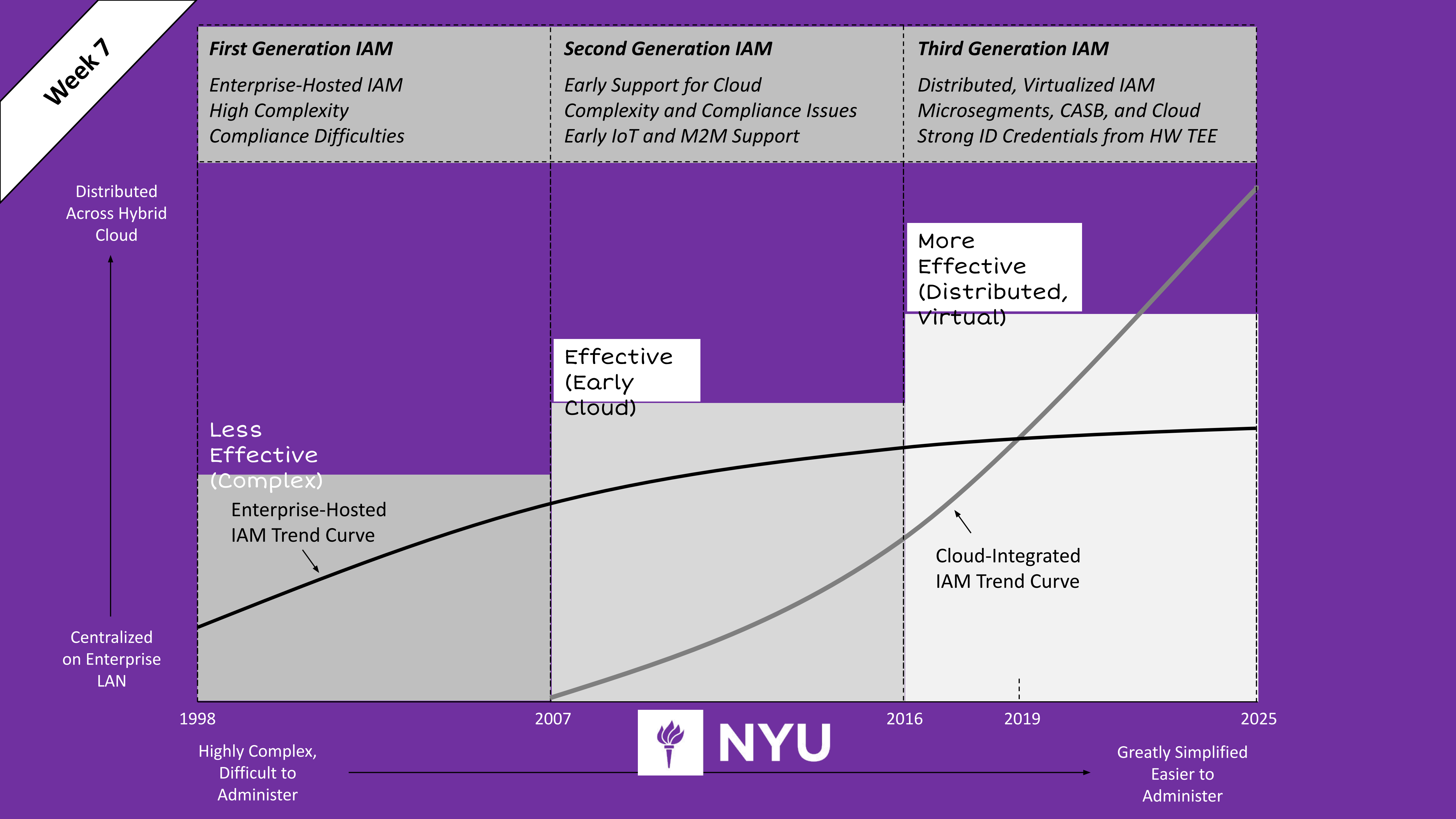
NYU

Week 7

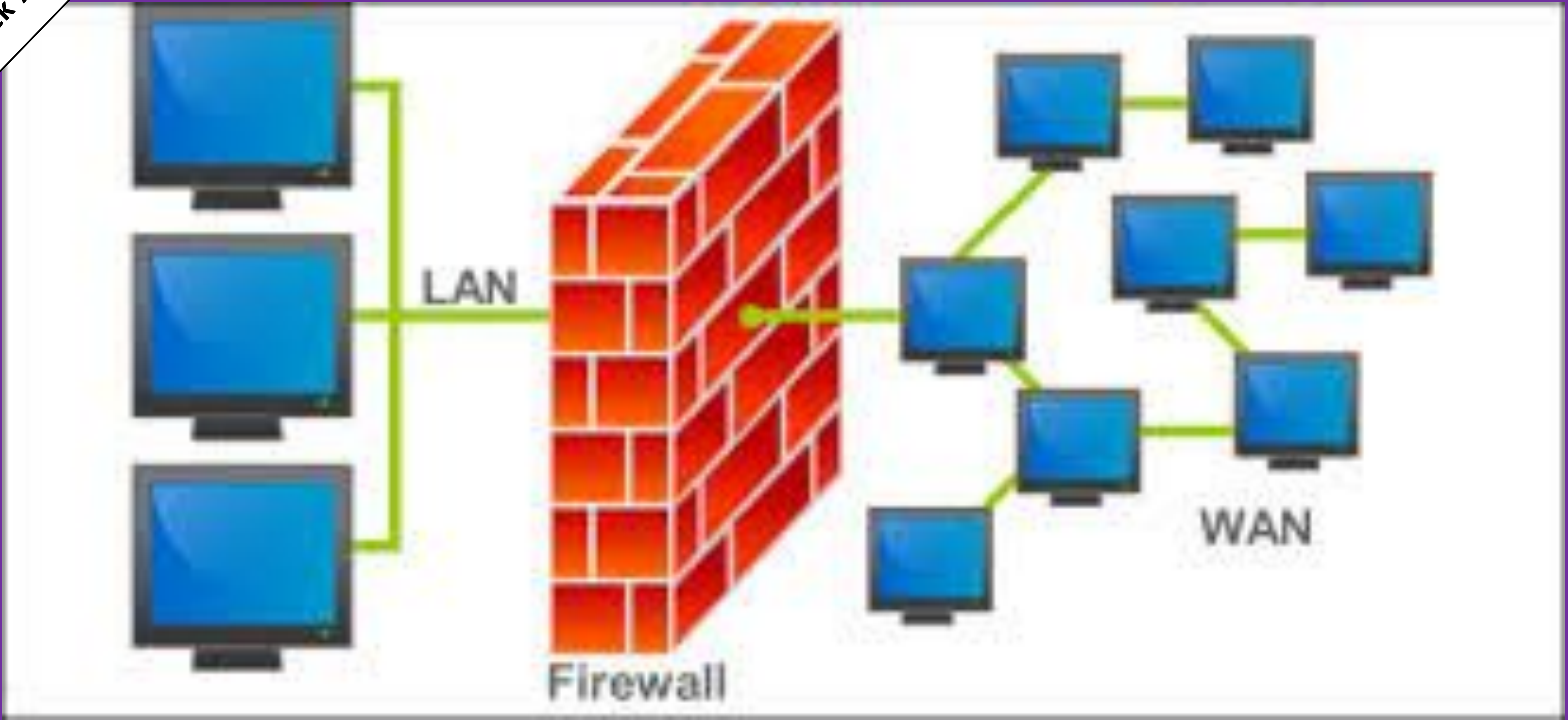
RSA SecurID Protocol



NYU


Week 7
First Generation IAM
Enterprise-Hosted IAM
High Complexity
Compliance Difficulties
Second Generation IAM
Early Support for Cloud
Complexity and Compliance Issues
Early IoT and M2M Support
Third Generation IAM
Distributed, Virtualized IAM
Microsegments, CASB, and Cloud
Strong ID Credentials from HW TEE
Distributed Across Hybrid Cloud
Centralized on Enterprise LAN
More Effective (Distributed, Virtual)
Effective (Early Cloud)
Less Effective (Complex)
Enterprise-Hosted IAM Trend Curve
Cloud-Integrated IAM Trend Curve
1998
2007
2016
2019
2025
Highly Complex, Difficult to Administer
Greatly Simplified Easier to Administer
NYU

Week 7

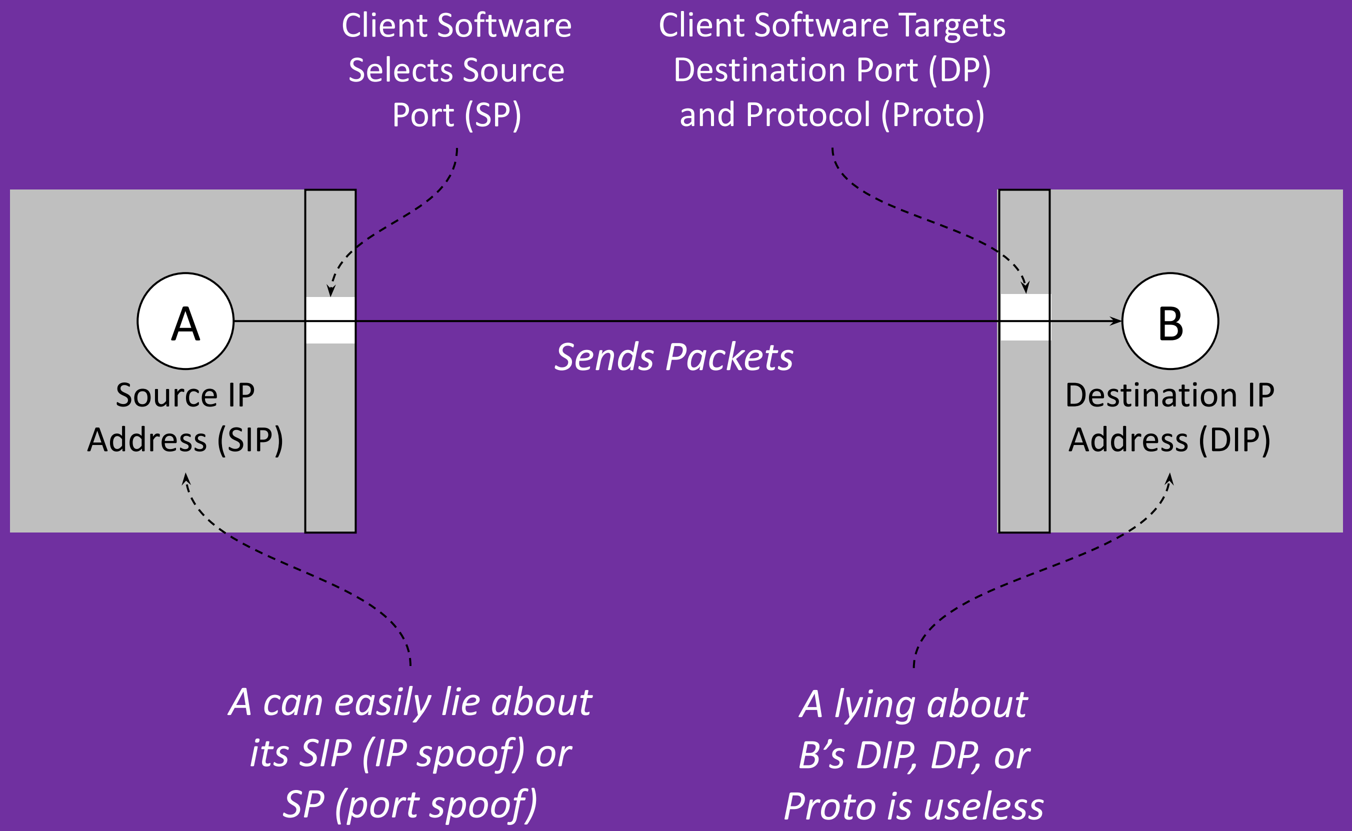


Access Controls and Firewalls

NYU

Week 7

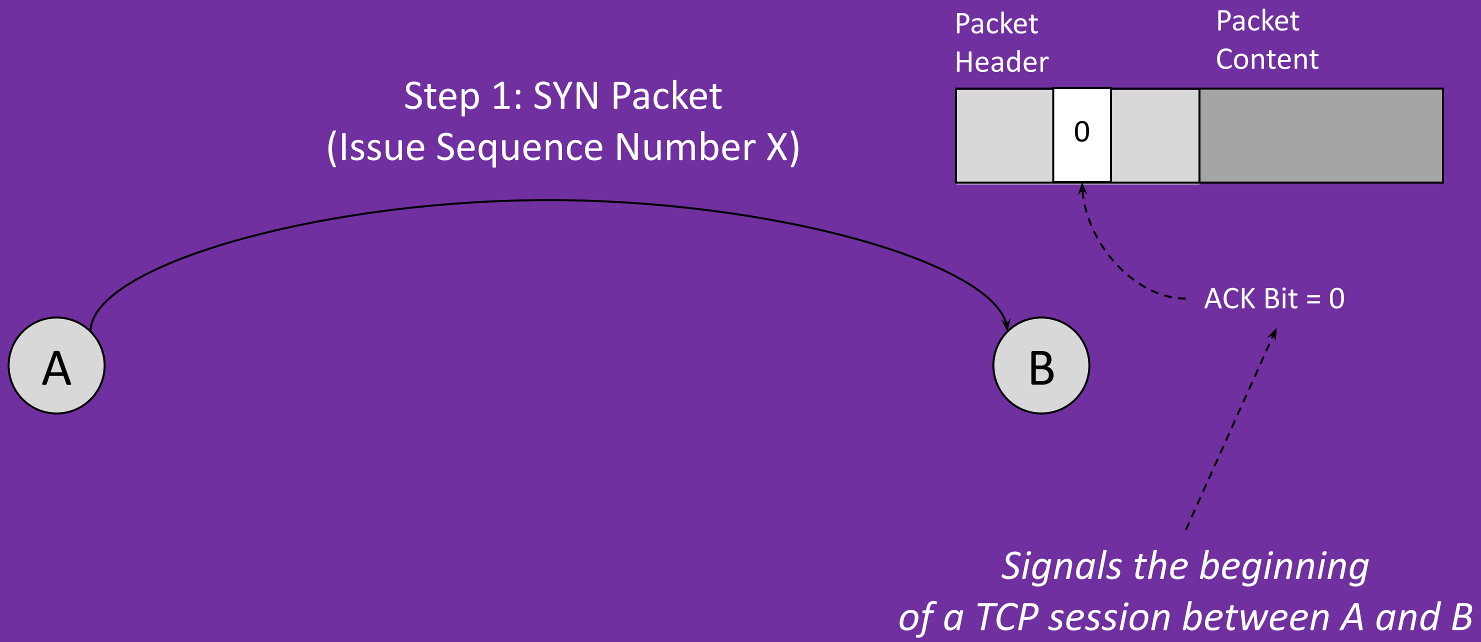
TCP/IP Basics



NYU

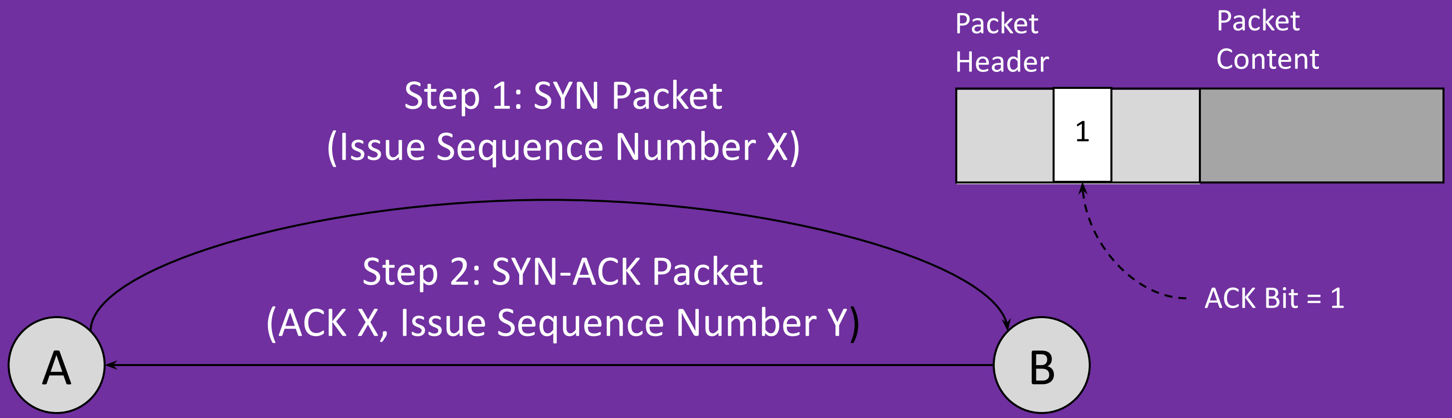
Week 7

Three-Step TCP Handshake



NYU

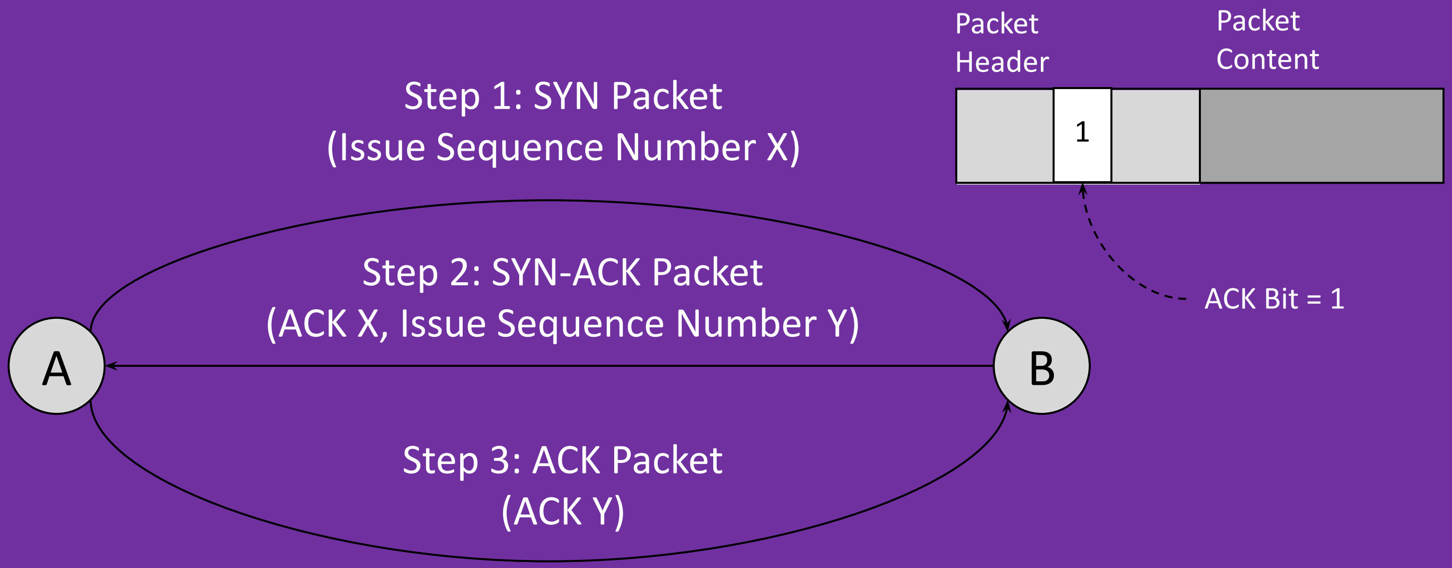
Three-Step TCP Handshake



NYU

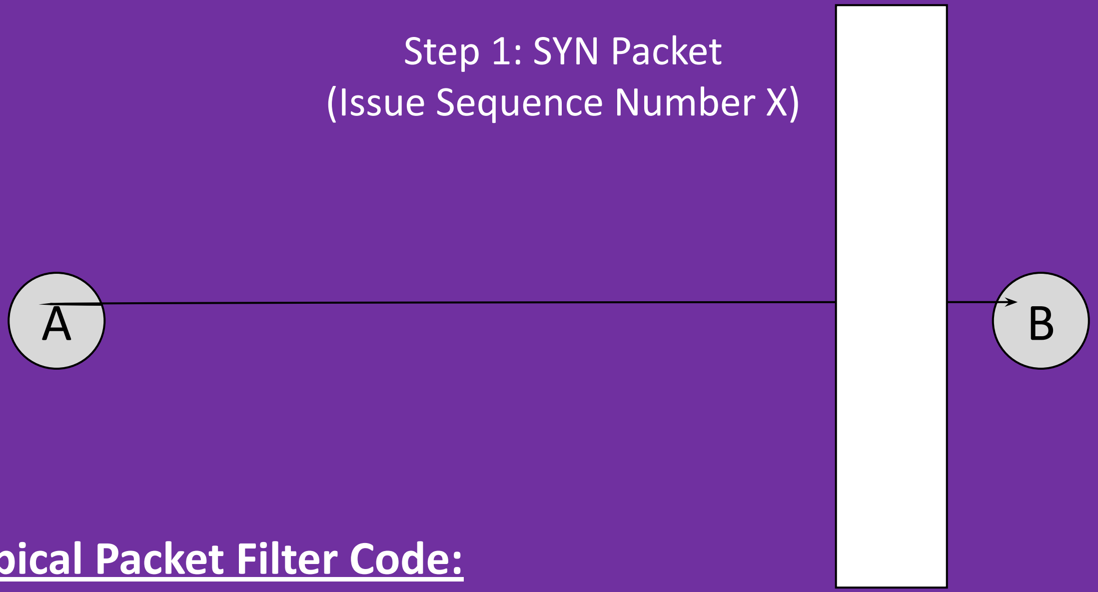
Week 7

Three-Step TCP Handshake



Week 7

Basis for Packet Filtering Firewall



Typical Packet Filter Code:

```
if packet header ACK bit = 0 then
      examine SIP, SP, DIP, and DP
      and determine if allow or block
else
      allow packet (ACK bit = 1)
fi
```

Might make network vulnerable to scanning

Week 7

Packet Filtering Firewall Graphical User Interface

Rule	SIP	SP	DIP	DP	Prot	ACK	Dir	Action
Name of the firewall rule	Source IP address of initiator	Source port of initiator	Destination IP address of initiator	Destination port of initiator	Protocol used by initiator	Value of the ACK bit for TCP only	Physical direction of packet	Block, allow, or divert

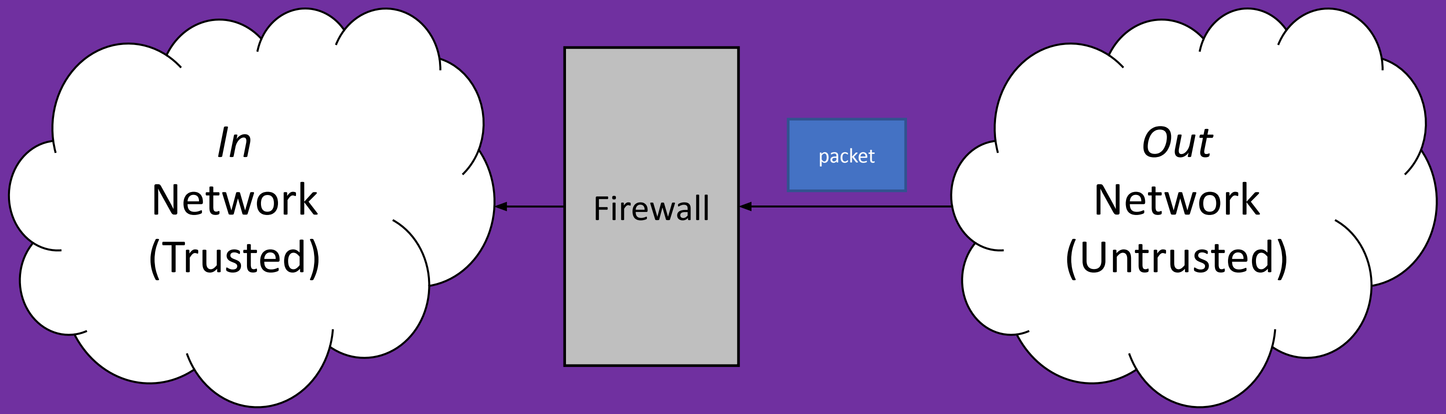
In
Network
(Trusted)

Out
Network
(Untrusted)

NYU

Packet Filtering Firewall Graphical User Interface

Rule	SIP	SP	DIP	DP	Prot	ACK	Dir	Action
Spoof Block (inbound)	In	*	In	*	TCP	0 (first TCP Packet)	Inbound	Block (Makes no sense)

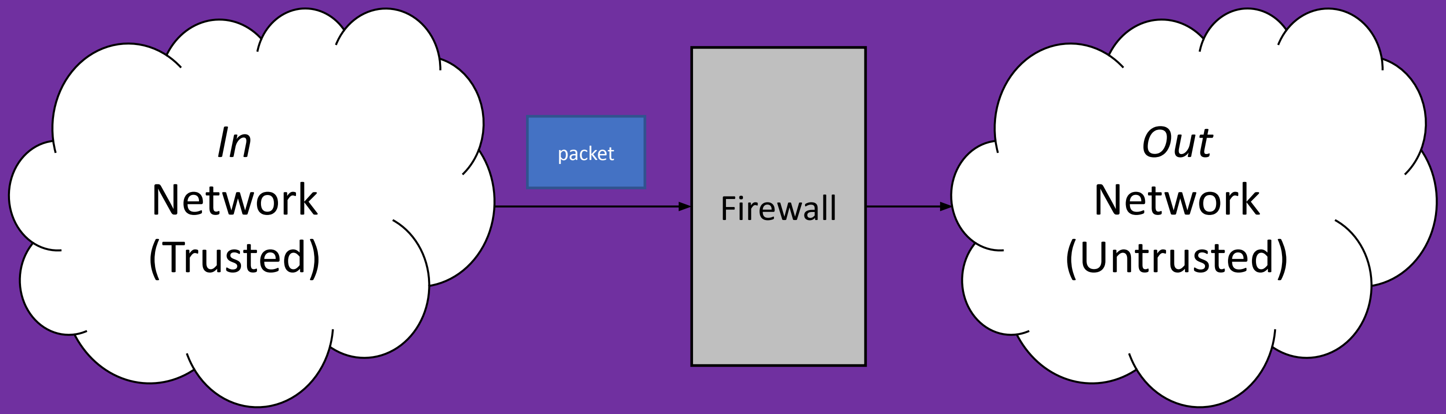


NYU

Week 7

Packet Filtering Firewall Graphical User Interface

Rule	SIP	SP	DIP	DP	Prot	ACK	Dir	Action
Spoof Block (outbound)	out	*	Out	*	TCP	0 (first TCP Packet)	Outbound	Block (Makes no sense)



NYU

Week 7

TLS 1.2: Client Makes two round trips

1. Client sends ClientHello-list of cipher suites plus random number (nonce)
2. Server sends ServerHello-chooses a cipher suite and sends a random number (nonce) and its certificate, optional DH ServerKeyExchange, and ServerHelloDone
3. Client verifies the certificate, extracts the public key, and sends a ClientKeyExchange premaster secret encrypted with the server's public key
4. Client and server both generate a key using a key-derivation function that takes in the two nonces and the PMS and outputs a symmetric key
5. Client sends ChangeCipherSpec to switch to symmetric encryption and an encrypted Finished message containing a MAC (hash of key + hash of all messages) to prevent MITM attack where cipher suite is downgraded
6. Server sends ChangeCipherSpec to switch to symmetric encryption and an encrypted Finished message containing a MAC (hash of key + hash of all messages) to prevent MITM attack where cipher suite is downgraded

Week 7

TLS 1.2

Step	Client	Direction	Message	Direction	Server
1			Client Hello	→	
2		←	Server Hello		
3		←	Certificate		
4		←	Server Key Exchange		
5		←	Server Hello Done		
6			Client Key Exchange	→	
7			Change Cipher Spec	→	
8			Finished	→	
9		←	Change Cipher Spec		
10		←	Finished		

NYU

Week 7

TLS 1.3: Client makes one round trip

1. Client sends ClientHello and Picks a Cipher Suite and sends DH Key Exchange
2. Server sends ServerHello and DH Key Exchange and uses that to encrypt and send Finished message
3. Client verifies certificate, extracts public key and generates a key using a key-derivation function that takes in a PMS and outputs a symmetric key

NYU

TLS 1.3

Step	Client	Direction	Message	Direction	Server
1			Client Hello Supported Cipher Suites Guesses Key Agreement Protocol Key Share	→	
2		←	Server Hello Key Agreement Protocol Key Share Server Finished		
3			Checks Certificate Generates Keys Client Finished	→	