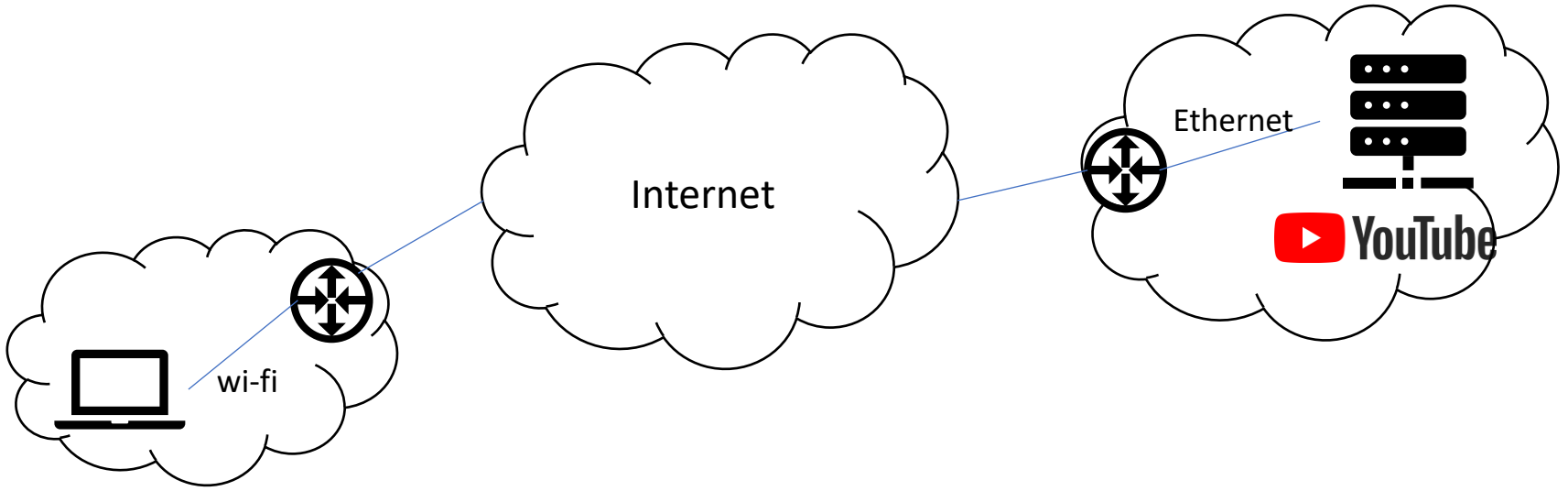
Problem / Overview

Course: Networking Fundamentals
Module: Network Security

University of Colorado **Boulder**

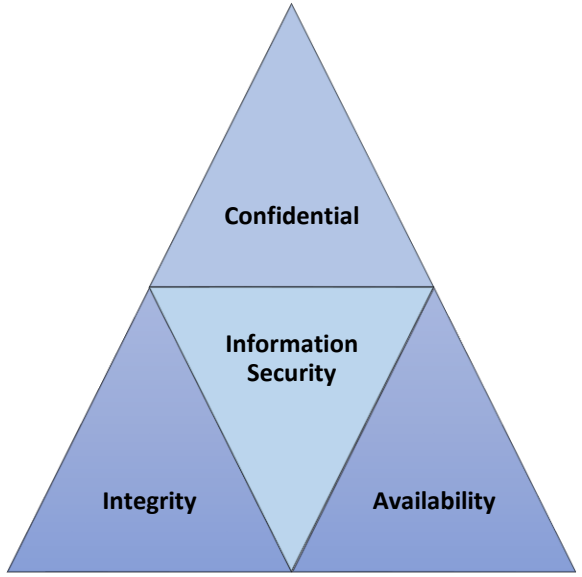
Network Security

- Communication is between parties across a medium that may or may not be under their control
- Malicious parties can attack the communication in various locations



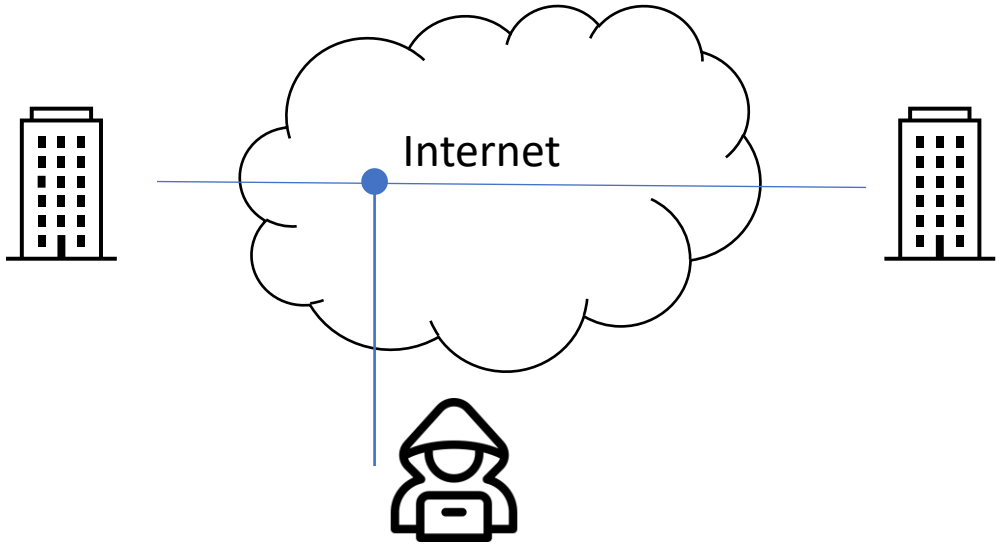
Properties

- Confidentiality – information only visible to those intended
- Integrity – data has not been modified / corrupted
- Availability – information can be accessed



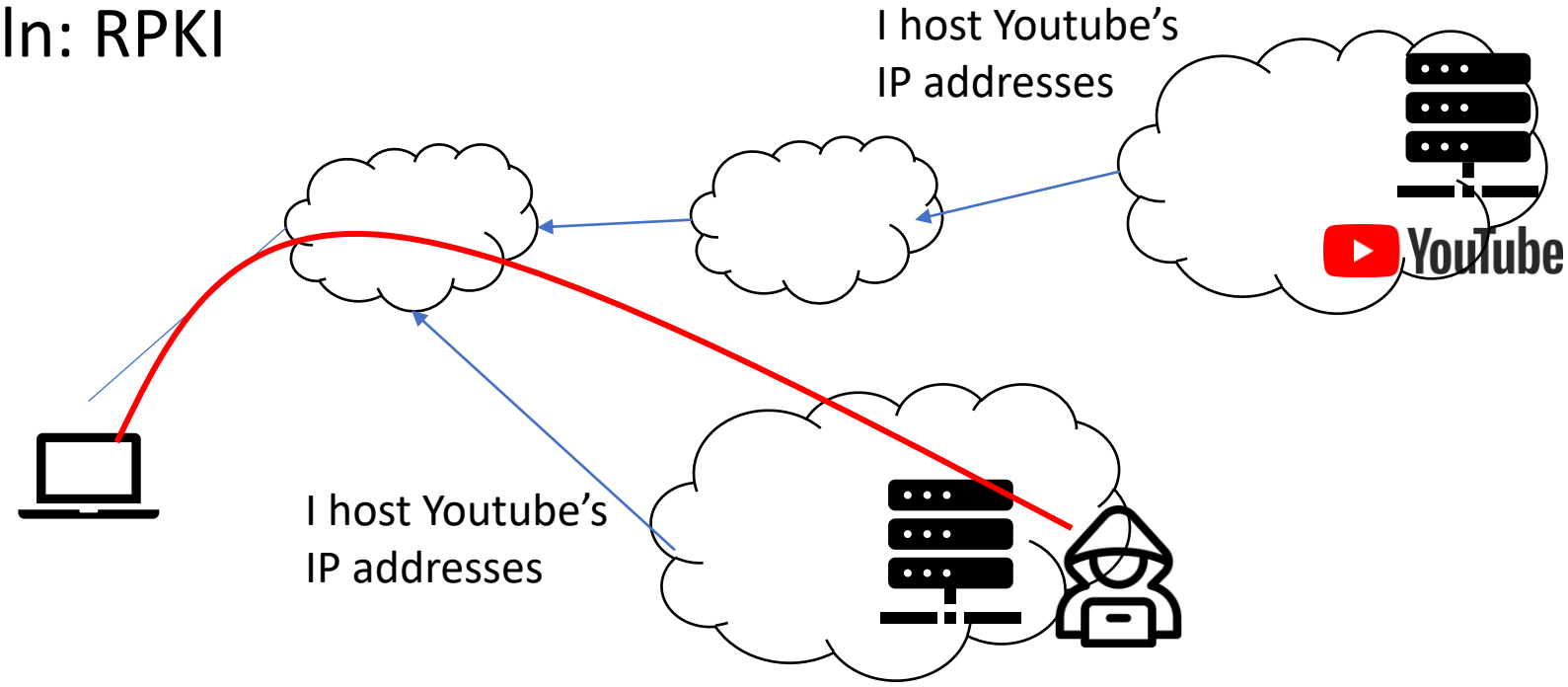
Example 1 – Network Data Plane

- Company has 2 sites and employees at different sites want to share information with each other
- If attacker can see the traffic, will get access to secret data.
- Soln: IPsec



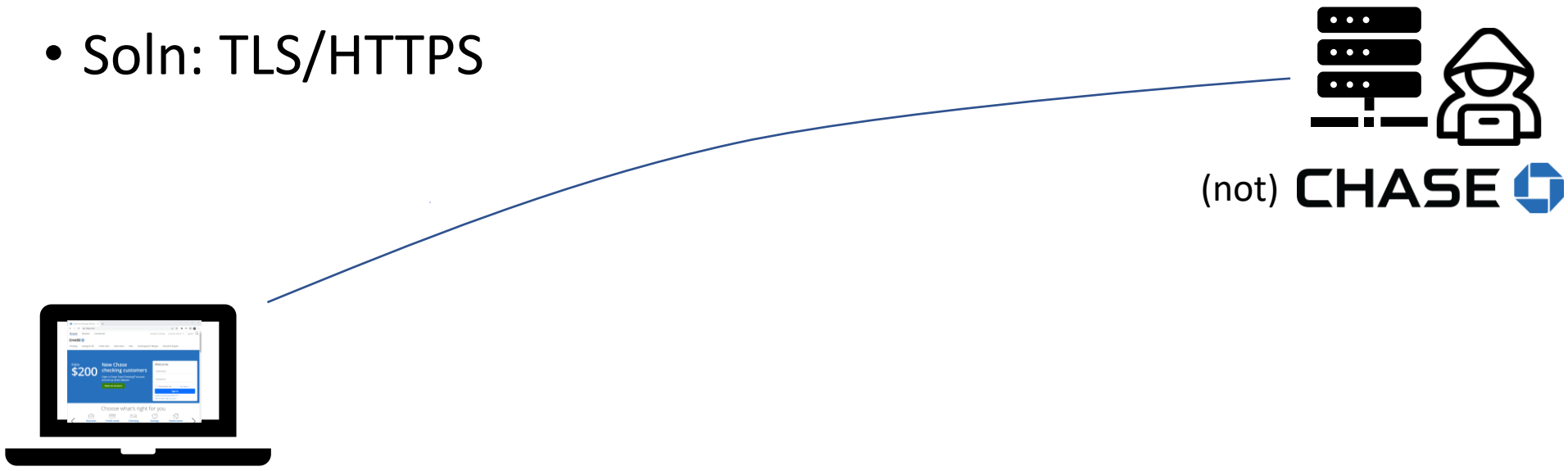
Example 2 – Network Control Plane

- BGP Communicates paths to reach IP prefixes
- If attacker can inject fake paths, it can get traffic directed to it (denying service, changing messages, inspecting traffic)
- Soln: RPKI



Example 3 – Application / Transport Layer

- TCP provides in-order reliable stream between two processes
- HTTP provides application protocol to get and put web objects
- Attacker could pretend to be the bank, or inspect / modify traffic (e.g., snooping on wi-fi in a public space)
- Soln: TLS/HTTPS



University of Colorado **Boulder**

Basics

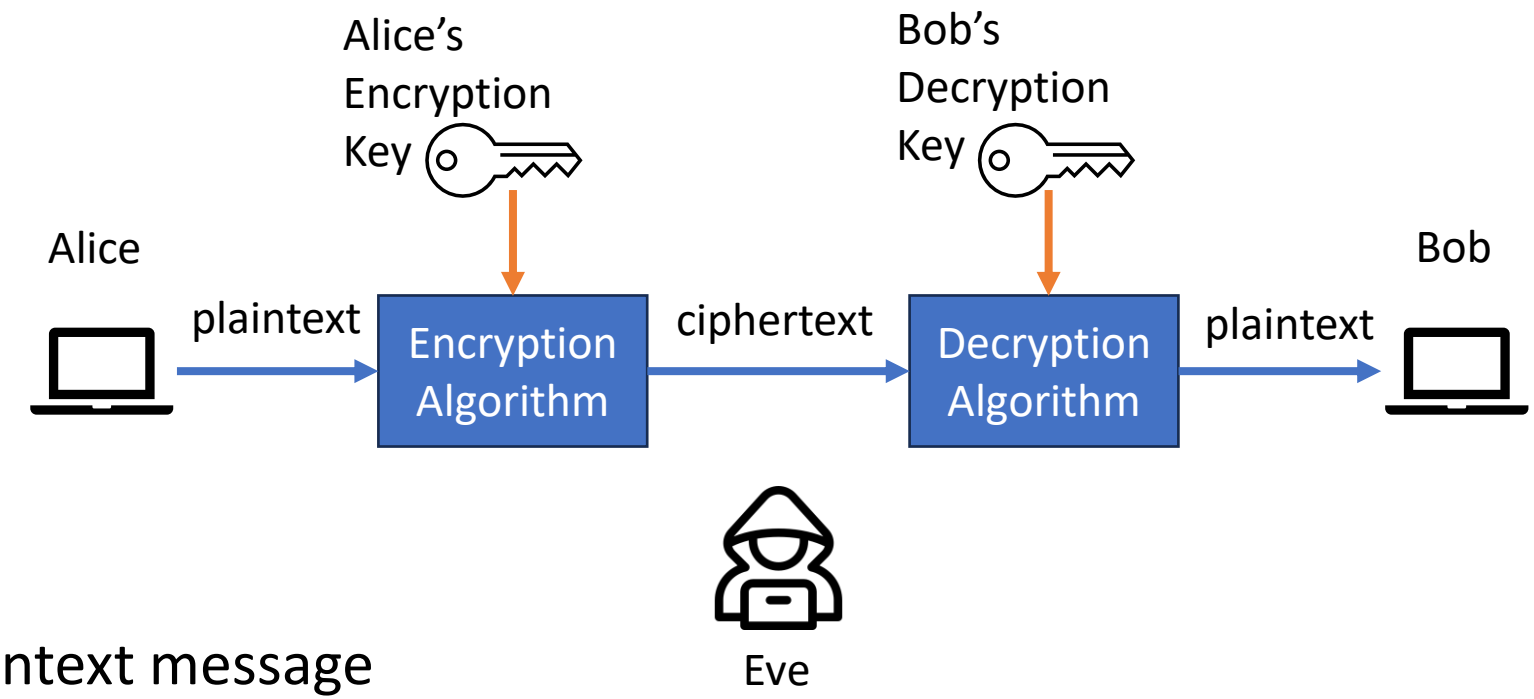
Course: Networking Fundamentals
Module: Network Security

University of Colorado **Boulder**

Confidentiality

- Want to send a message and keep it secret from anyone other than the recipient

Cryptography Terminology Overview



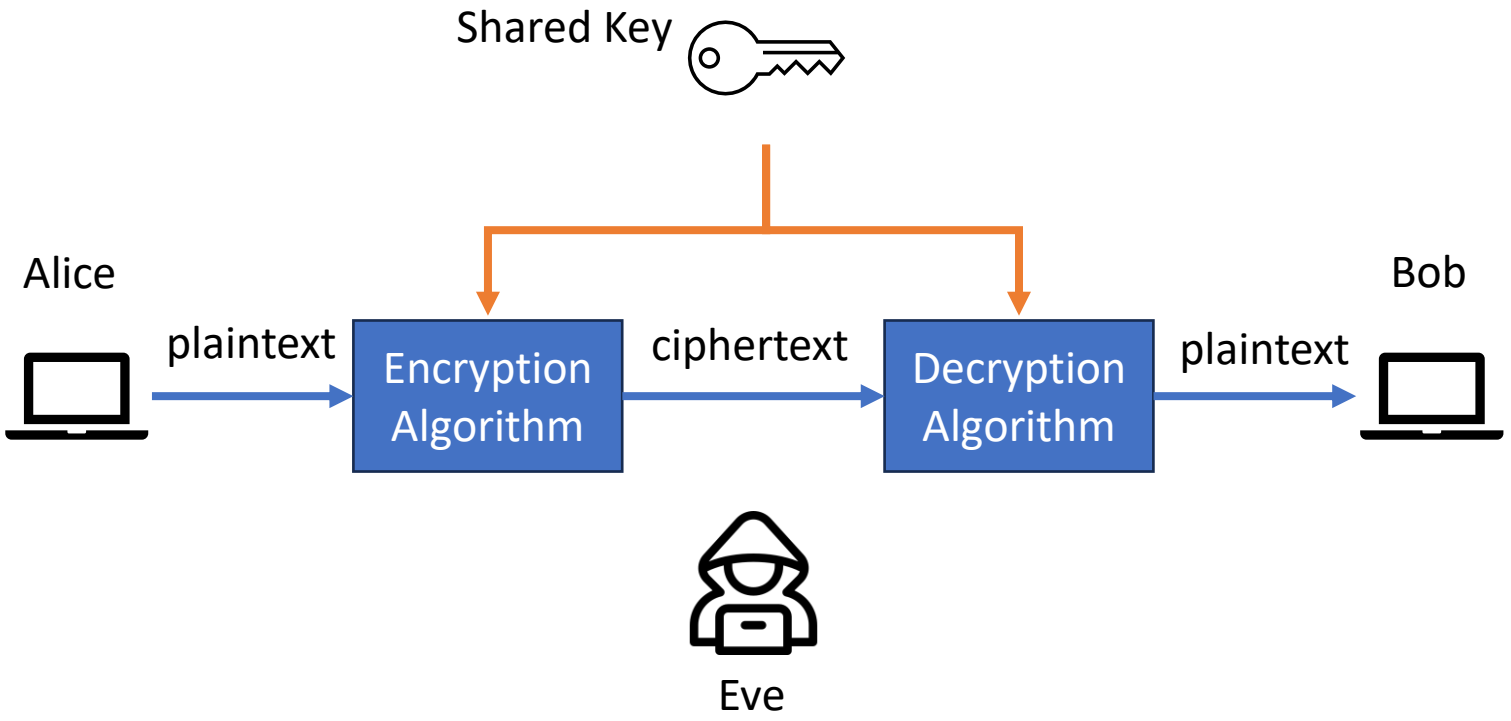
m: plaintext message

$K_A(m)$: ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

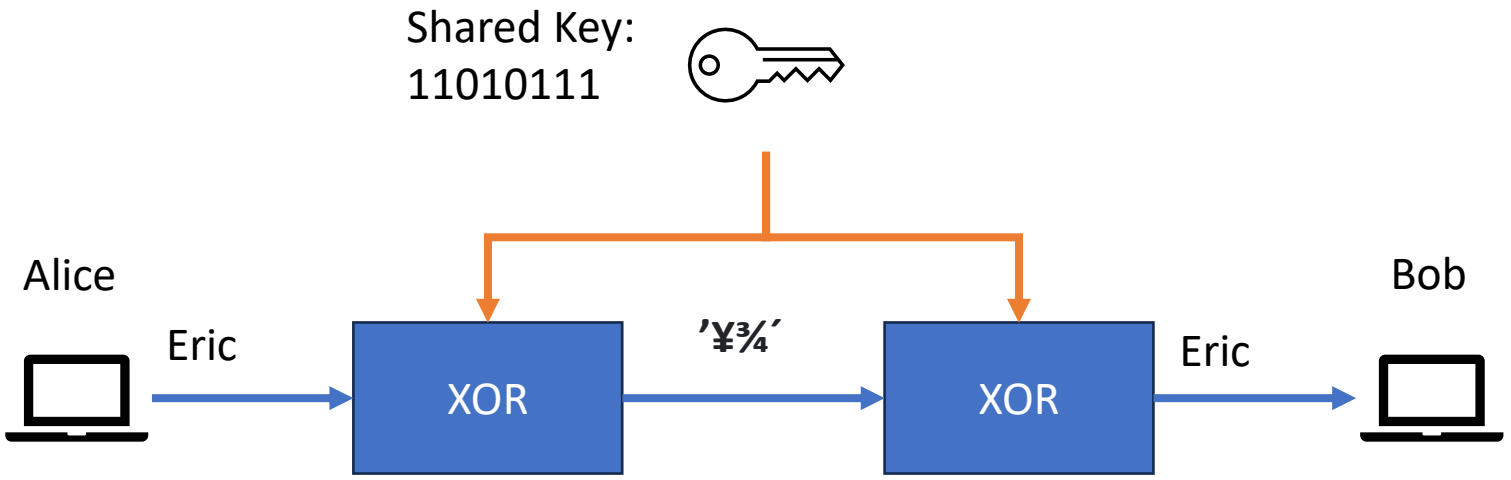
* Common names for attackers you may see: Eve, Mallory, Trudy

Symmetric Encryption (shared key)



- Alice and Bob share a key, which is used in both the encryption and decryption
- Fast but need a way to share a key

Trivial Example: Xor Symmetric Encryption



- Block by Block (block = 8 bit, so key is 8 bits). Algorithm is xor

Encrypt

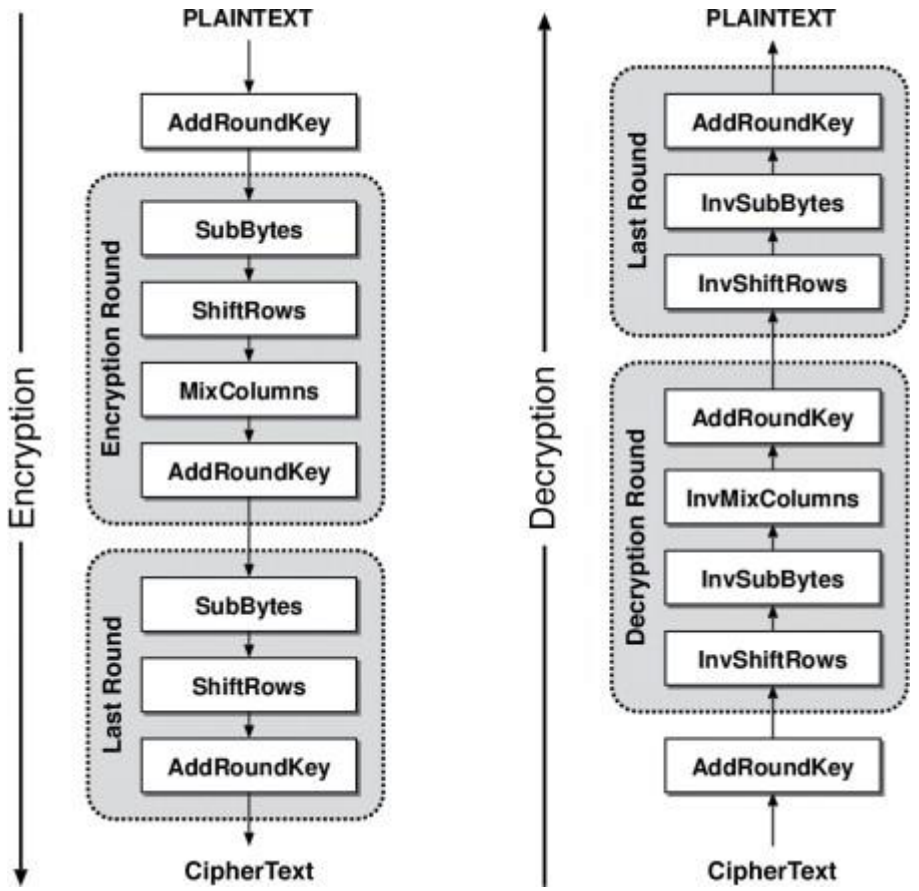
```
  4      5  = E
0100 0101  (plaintext byte)
xor 1101 0111  (key)
-------------
1001 0010  (ciphertext byte)
  9      2  = ‘
```

Decrypt

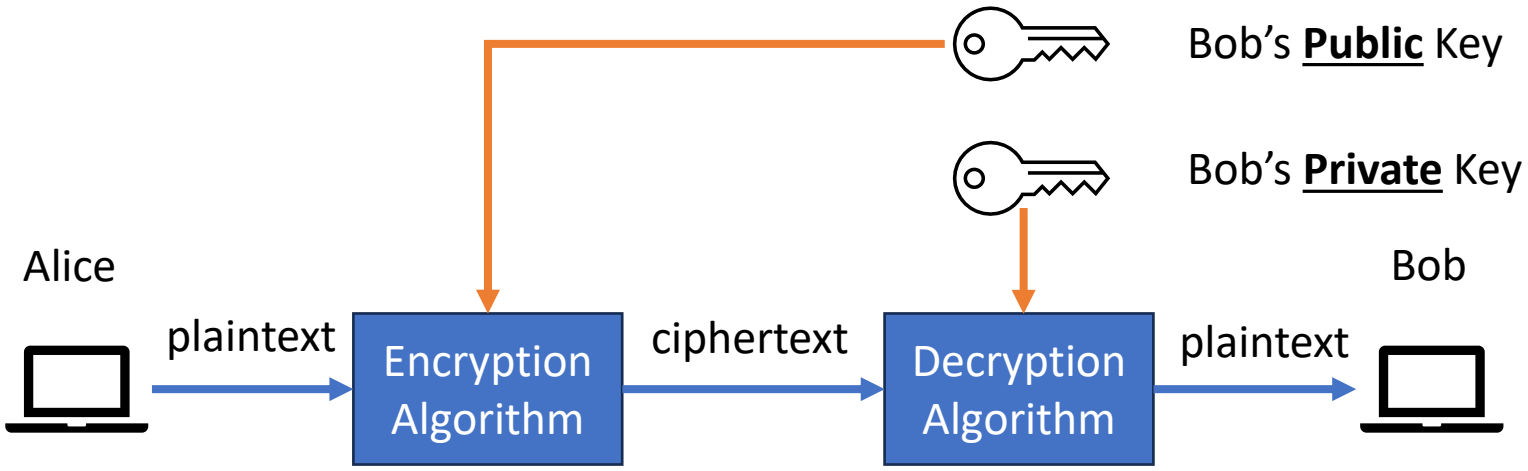
```
  9      2  = ‘
1001 0010  (ciphertext byte)
xor 1101 0111  (key)
-------------
0100 0101  (plaintext byte)
  4      5  = E
```

Better Example: AES

- NIST Standard in 2001
- 128, 192, or 256 bit keys



Asymmetric Encyption



- Bob generates a key pair – one is private, one can be shared with all
- Slow, but simple to share the key (since it doesn't matter who has it)

RSA: encryption, decryption

1. given Public Key (*n,e*) and Private Key (*n,d*)

2. Message *m* is just a bit pattern interpreted as an integer

3. to encrypt message *m (<n)*, compute

$c = m^e \bmod n$

4. to decrypt received bit pattern, *c*, compute

$m = c^d \bmod n$

Works because of properties of mod and how n, e, and d are chosen

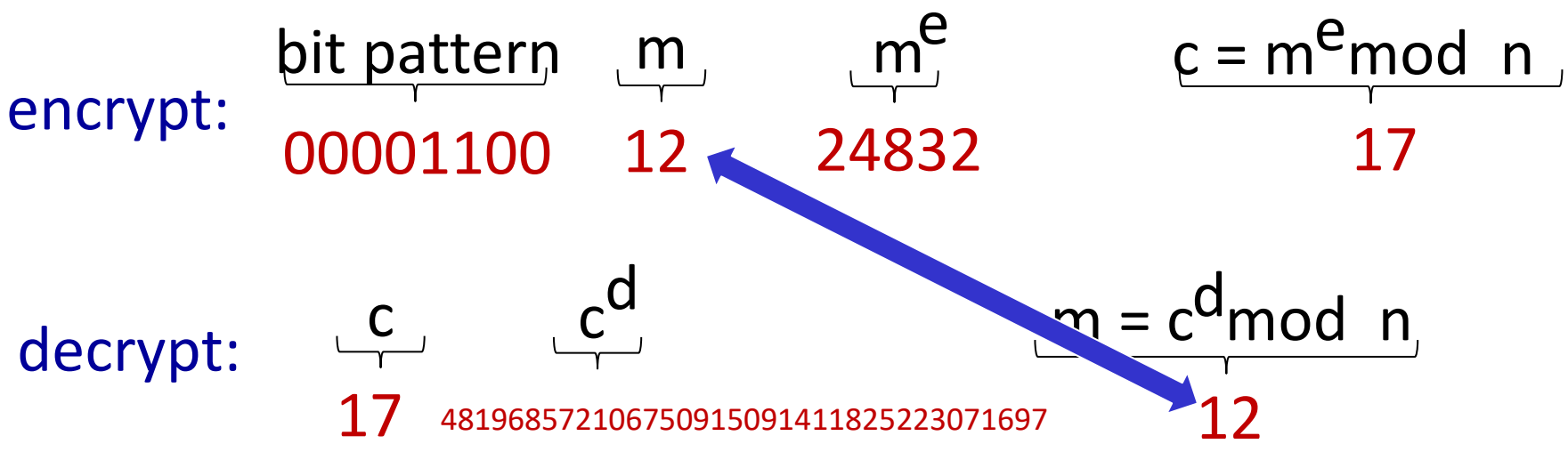
RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=(p-1)(q-1)=24*.

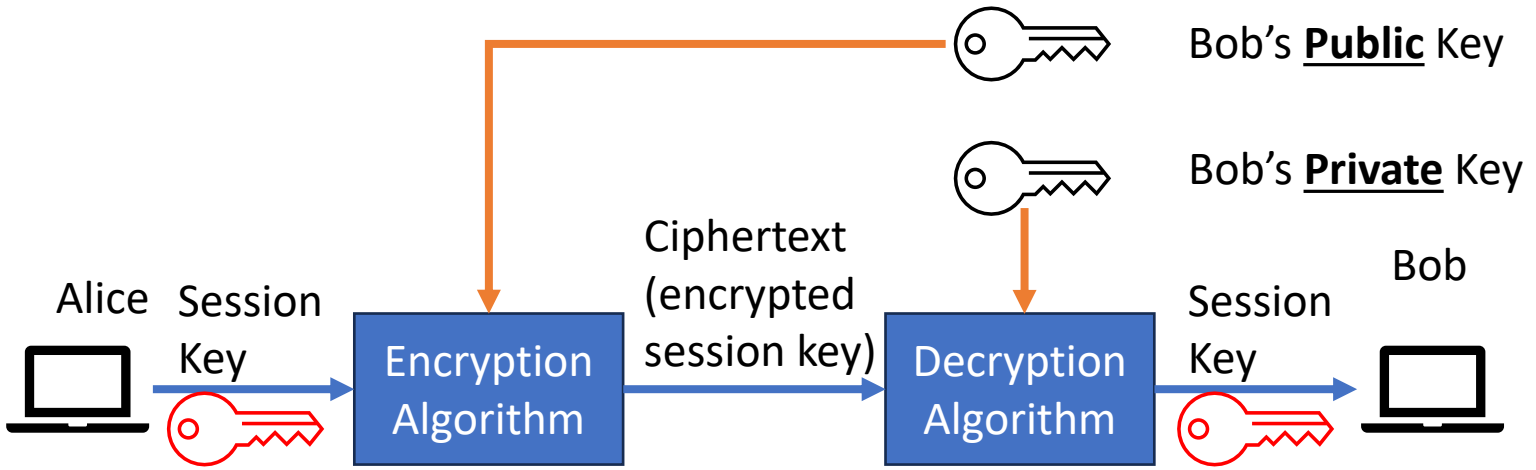
e=5 (so *e, z* relatively prime).

d=29 (so *ed-1* exactly divisible by z).

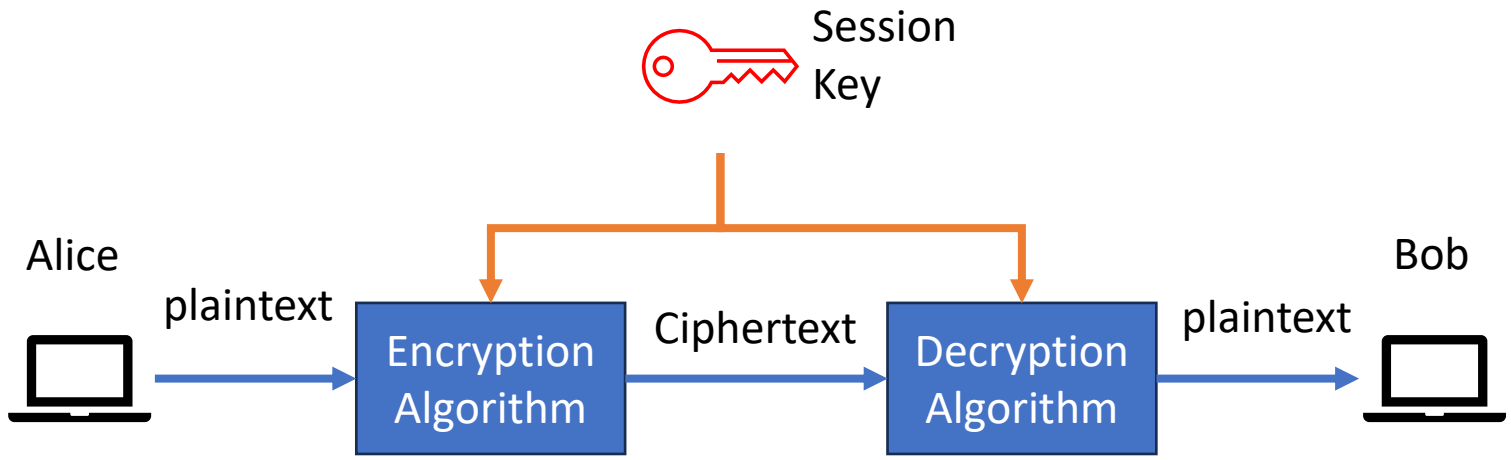
encrypting 8-bit messages.



Using Public Key to exchange Shared Key



Part 1: Alice picks session key, uses Bob's public key to encrypt and send to Bob. Bob can decrypt with his private key.



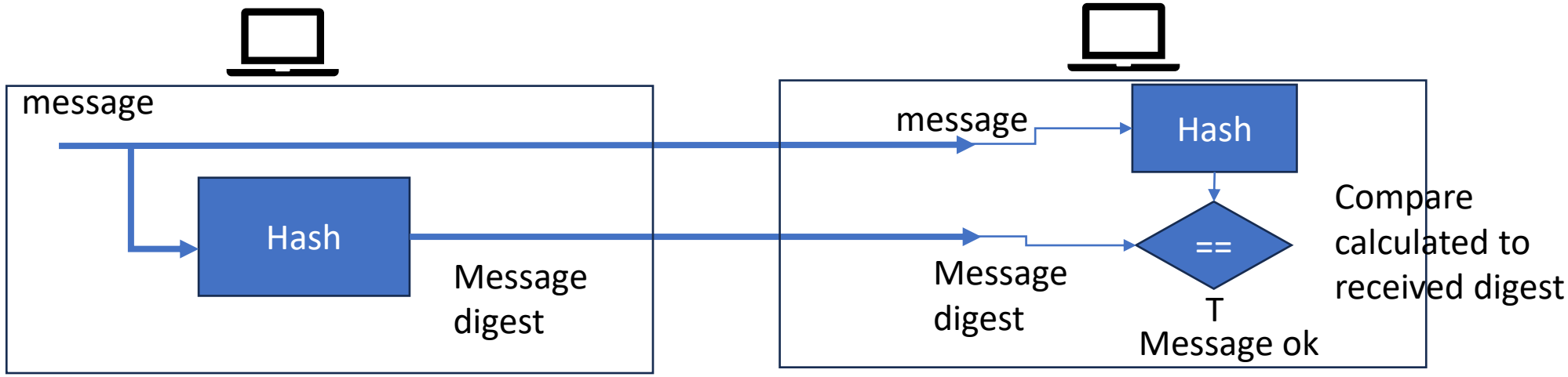
Part 2: Since Alice and Bob now share a secret key that only they know about, they can use that to encrypt/decrypt longer messages

Message Integrity

- Don’t want messages to be modified

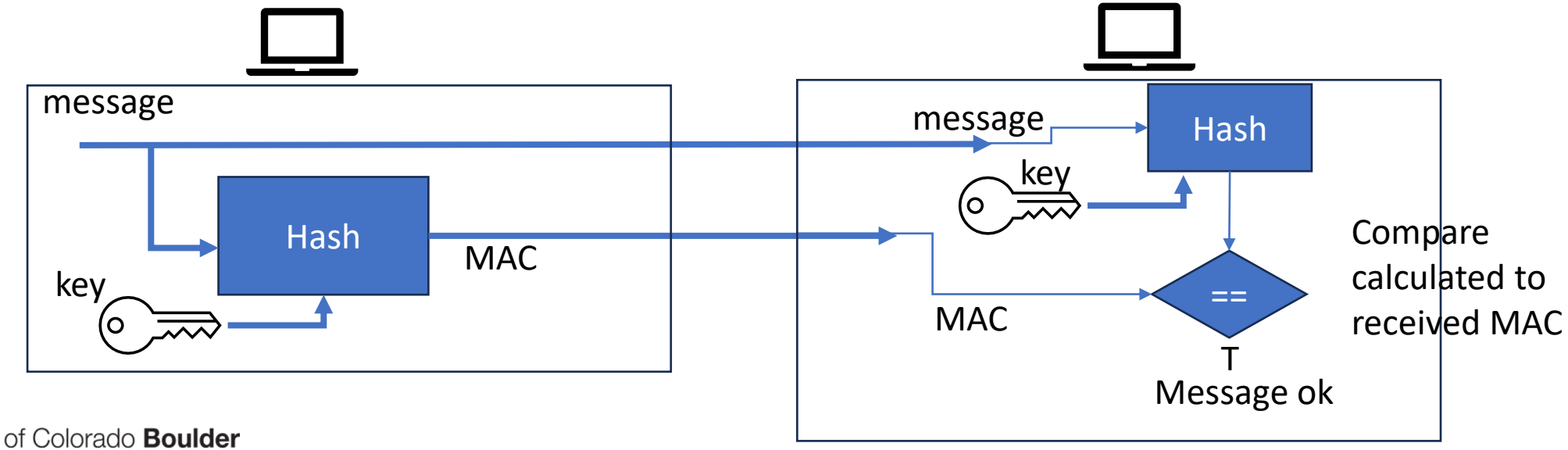
Hash

- Hash function – calc a fixed length digest from a variable length message
- Desired property – computationally infeasible to find another message that would result in the same message digest
- Assumes digest unmodified – used in integrity checking software downloads



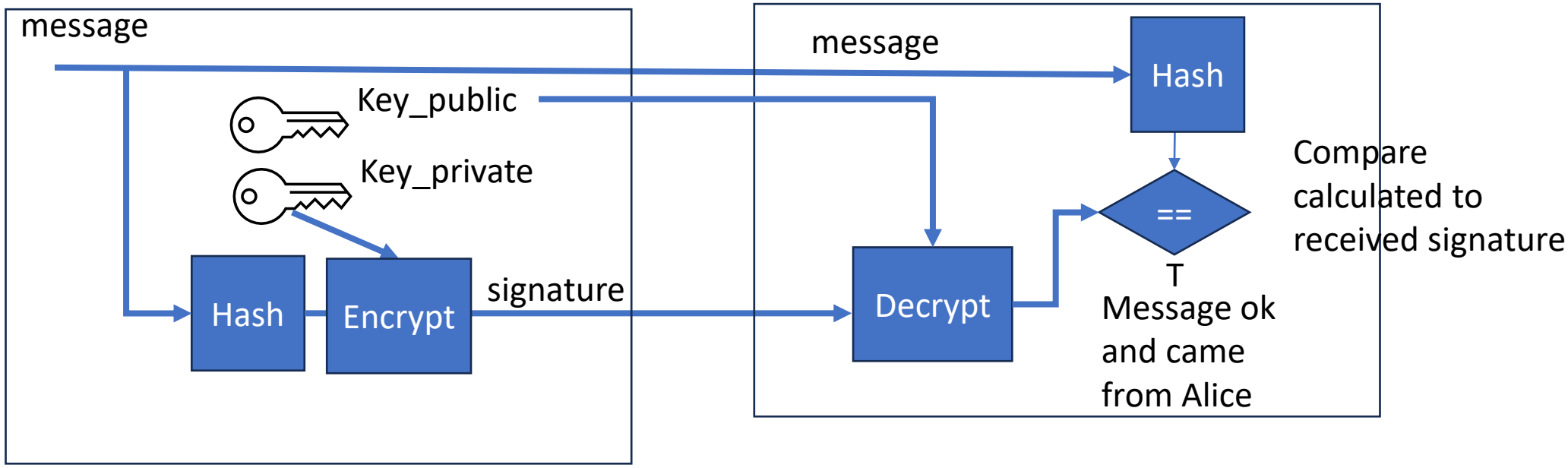
Hash-keyed Message Authentication Code (HMAC)

- Removes assumption of digest known / unmodified
- Sender and receiver share a secret key
- Append key and message as follows:
  H(*key* ‖ **H**(*key* ‖ *message*))



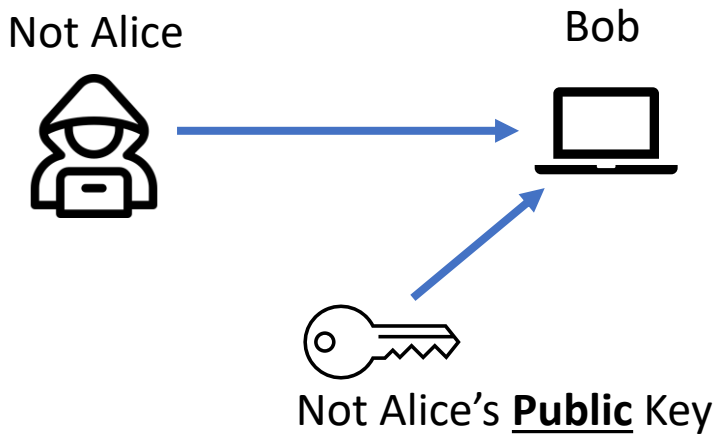
Digital Signatures

- Uses asymmetric keys to enable checking 1) the message hasn't changed, and 2) it came from Alice
- Alice uses private key to "sign", Bob uses Alice's public key to verify



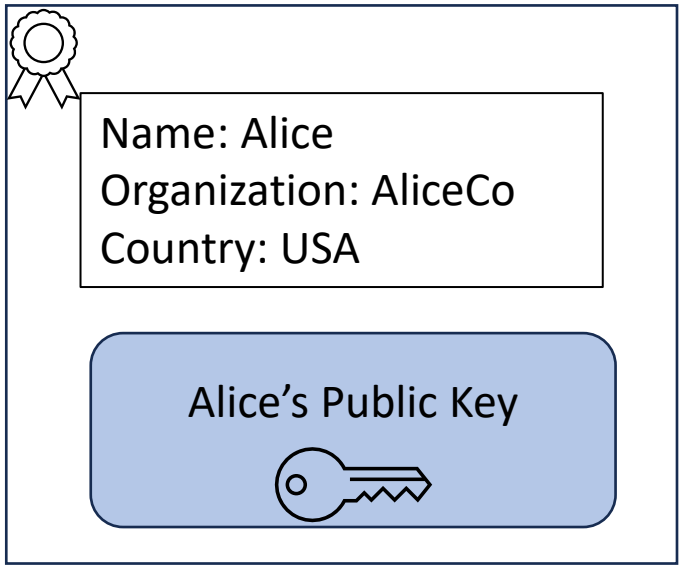
Authentication

- Want to ensure you are communicating with who you expect to be communicating with
  (e.g., is this Public key really Alice's?)



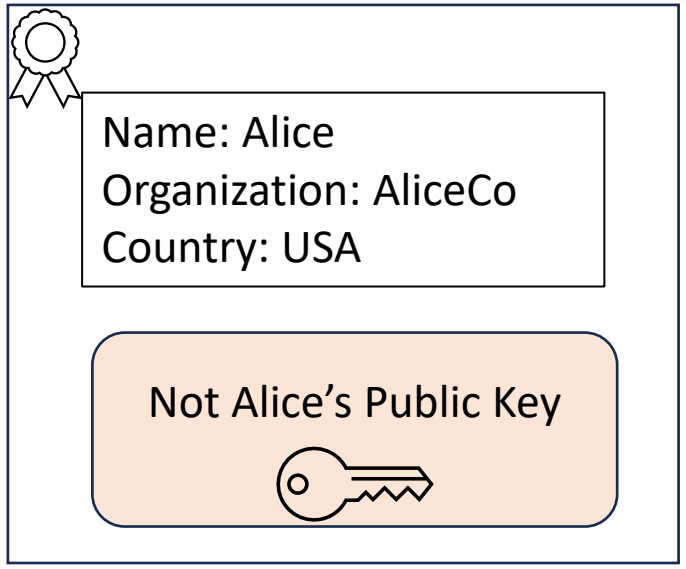
Digital Certificates

- Binds public key and identity
  (definition of identity depends on the context)
- Bob can validate that this key is Alice's



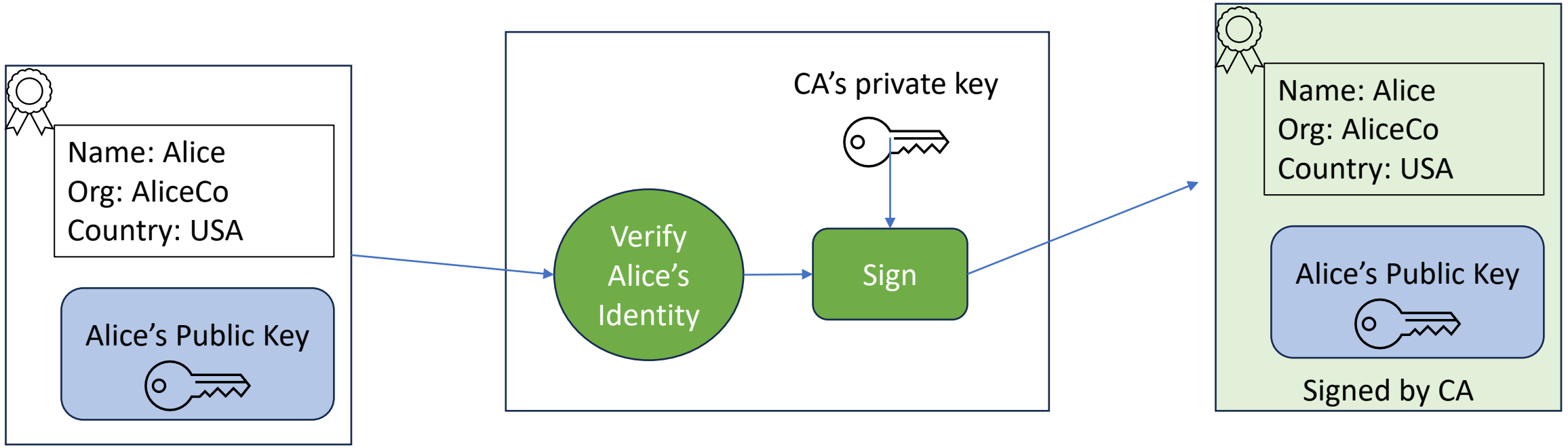
Problem – Integrity of Certificate

- Attacker could modify the certificate



Certificate Authority

- Some trusted authority can verify Alice's identity and use its private key to sign
- Anybody can verify the digital certificate using CA's public key
- CA's public key has to be trusted (e.g., it's pre-installed in a browser)



University of Colorado **Boulder**

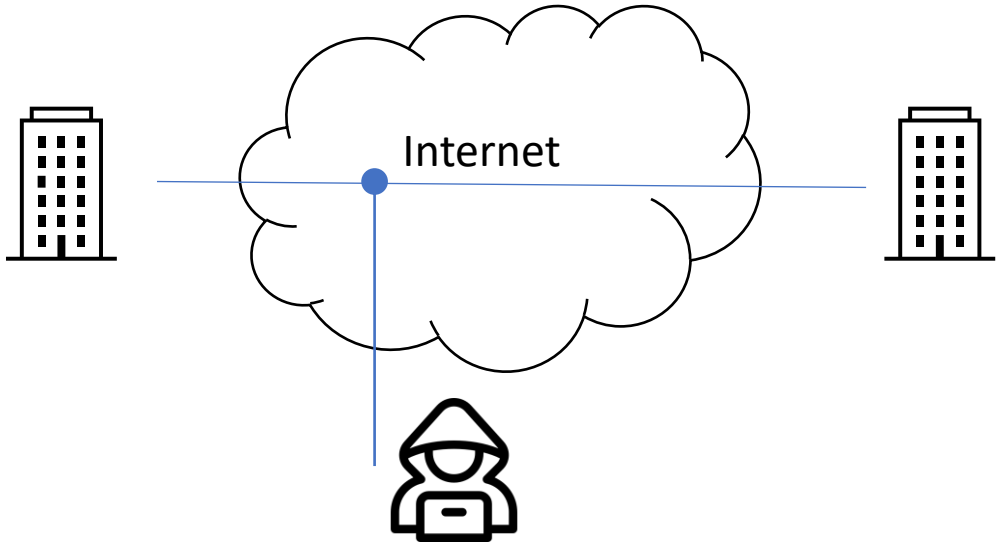
IPSec

Course: Networking Fundamentals
Module: Network Security

University of Colorado **Boulder**

Example 1 – Network Data Plane

- Company has 2 sites and employees at different sites want to share information with each other
- If attacker can see the traffic, will get access to secret data.
- Soln: IPsec

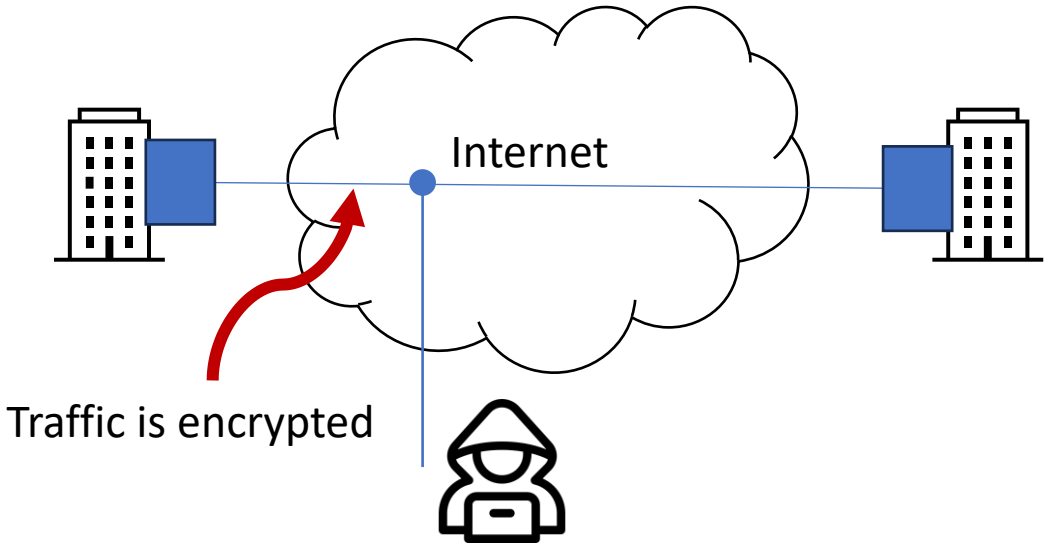


How Would Someone Eavesdrop?

- Wi-fi is broadcast communication
- Network provider compelled by government agency or has a rogue employee or a compromised router
- Misconfiguration in routing redirects traffic to the wrong place
- Route hijacking to intentionally redirect traffic

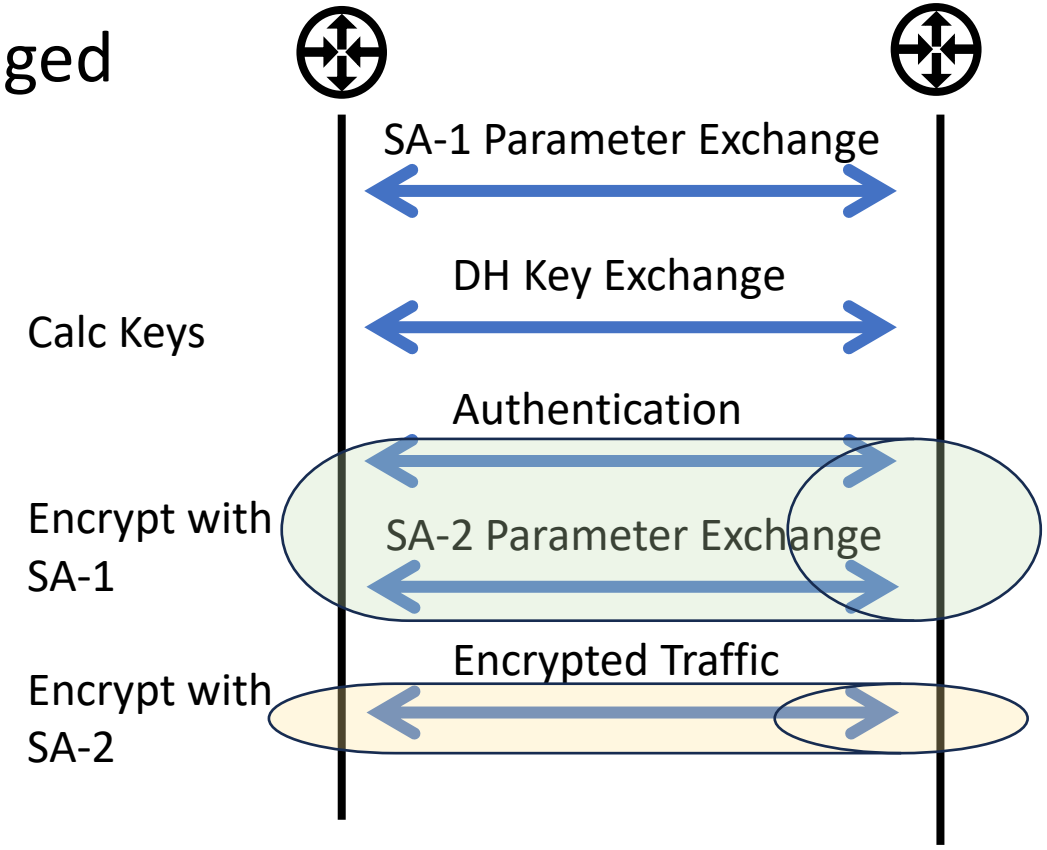
IPsec

- Create an encrypted tunnel between two end points
  - Can be for whole site or one machine
- Commonly used in VPNs (virtual private networks)



IPsec Overview

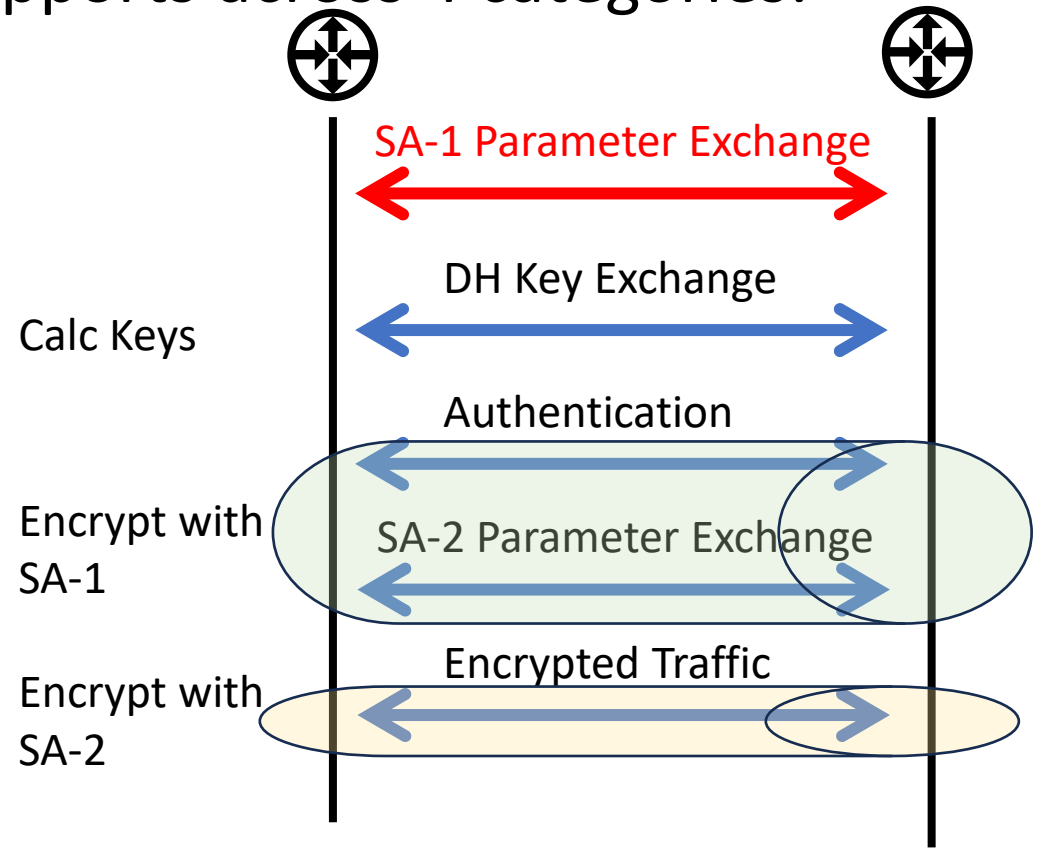
- To set up an IPsec tunnel, a protocol (ISAKMP/IKE) is used to set up the parameters and keys used, and authenticate each side
- Then traffic can be exchanged



ISAKMP - Internet Security Association and Key Management Protocol
IKE – Internet Key Exchange
SA – Security Association

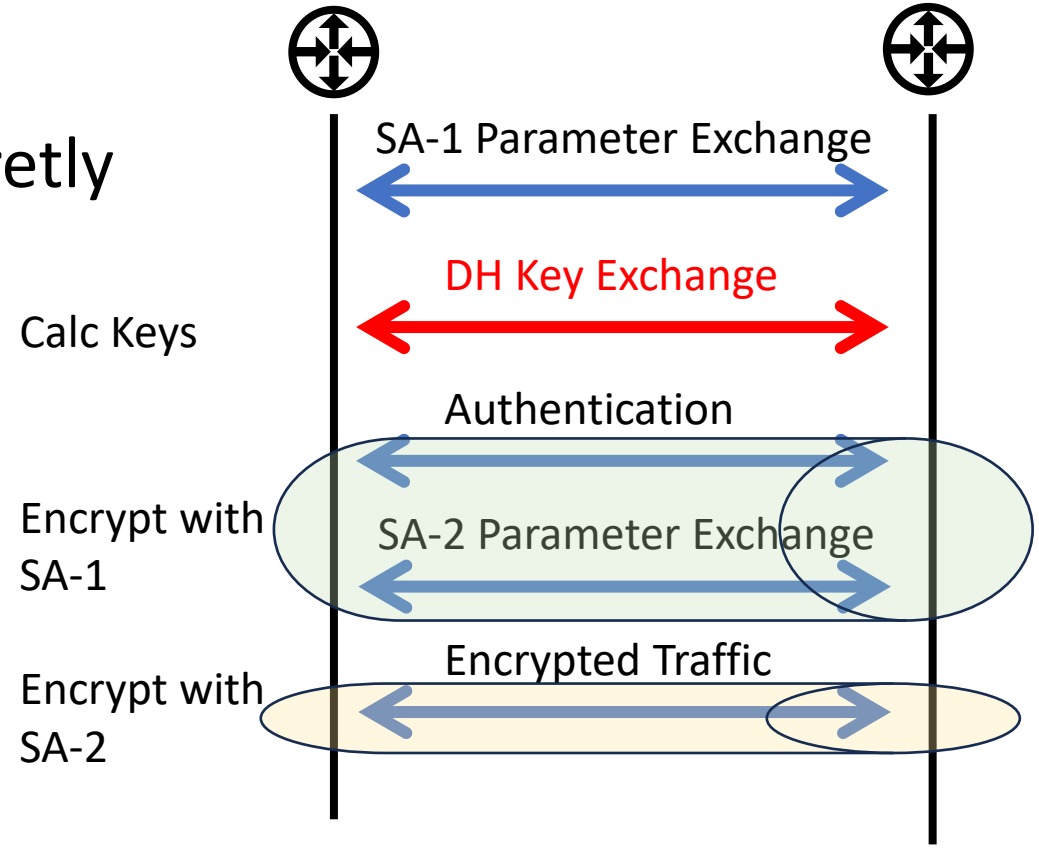
SA-1 Parameter Exchange

- Initiator says what algorithms it supports across 4 categories:
  - Encryption – DES, AES…
  - Hash – SHA, MD5…
  - DH Group – 1,2,5…
  - Auth – shared, certificate…
- Responder chooses

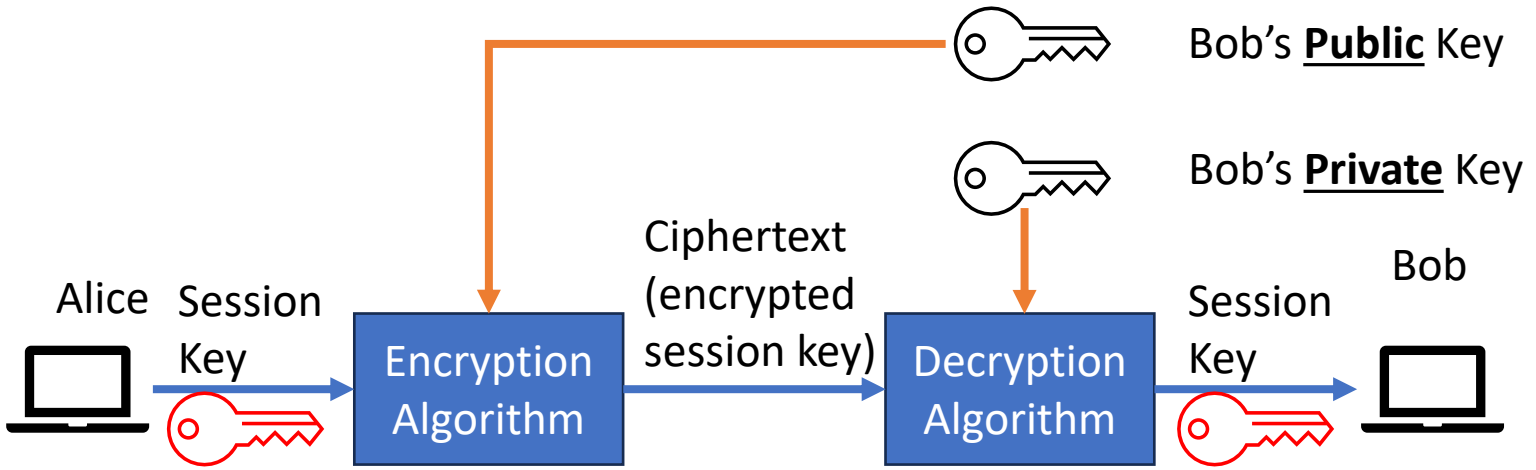


Diffie Hellman Key Exchange

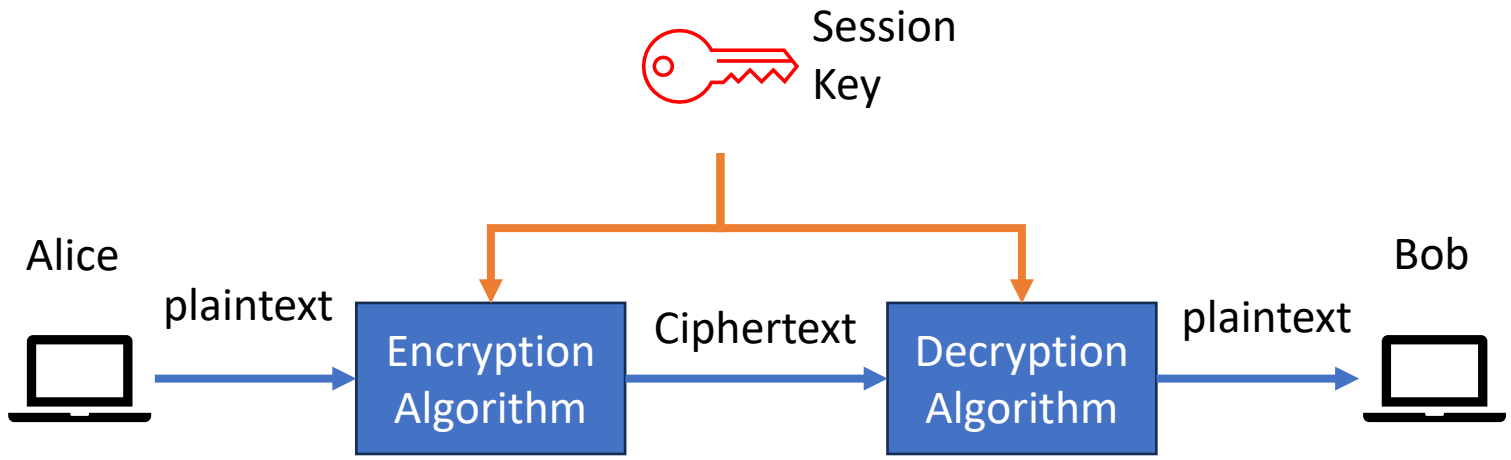
- IPsec uses shared keys for encryption and integrity
- Diffie Hellman is a protocol to secretly create and share private keys



Recall: Using Public Key to exchange Shared Key



Part 1: Alice picks session key, uses Bob's public key to encrypt and send to Bob. Bob can decrypt with his private key.



Part 2: Since Alice and Bob now share a secret key that only they know about, they can use that to encrypt/decrypt longer messages

Diffie Hellman

allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel.

Pick prime p and base g

p, g →

Pick secret x
Calc h_A = g^x mod p

h_A →

Pick secret y
Calc h_B = g^y mod p

Shared key
K = $(h_B)^x \ mod \ p$

← h_B

Shared key
K = $(h_A)^y \ mod \ p$

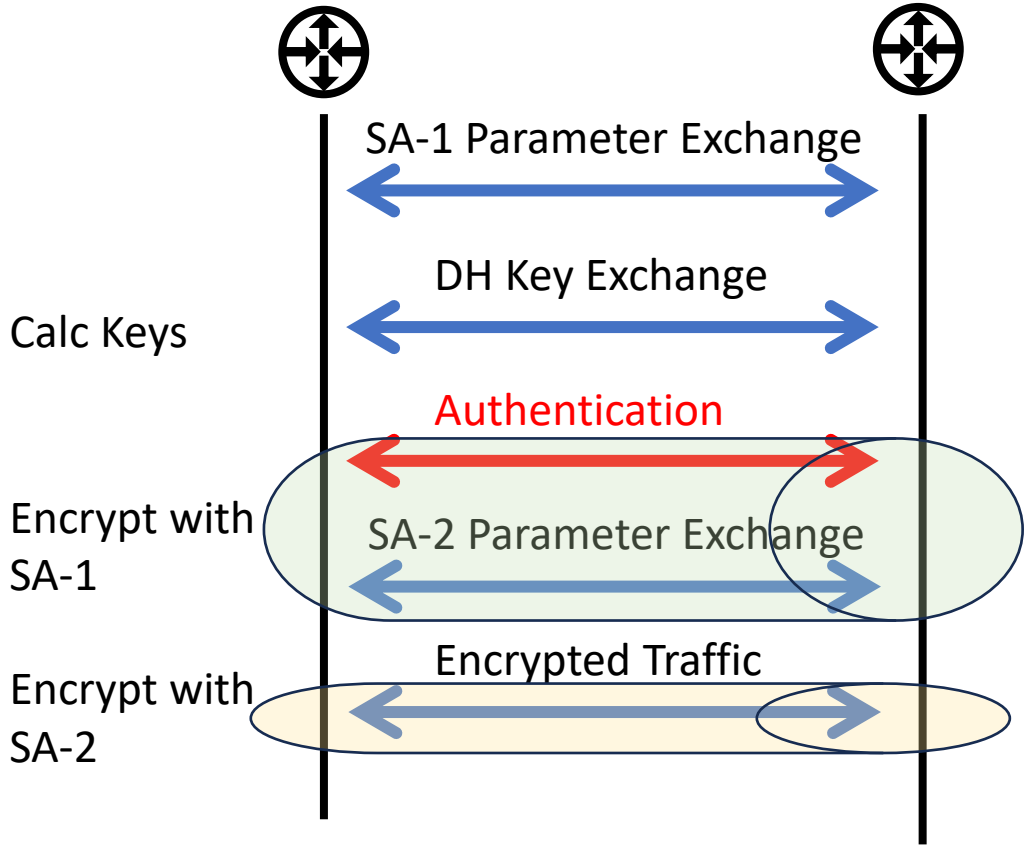
Given p, g, h_A , and h_B
it is hard to calculate K

Why K is same:
= (g^y mod p)^{x}mod p
= g^{xy} mod p

Why K is same:
= (g^x mod p)^{y}mod p
= g^{xy} mod p

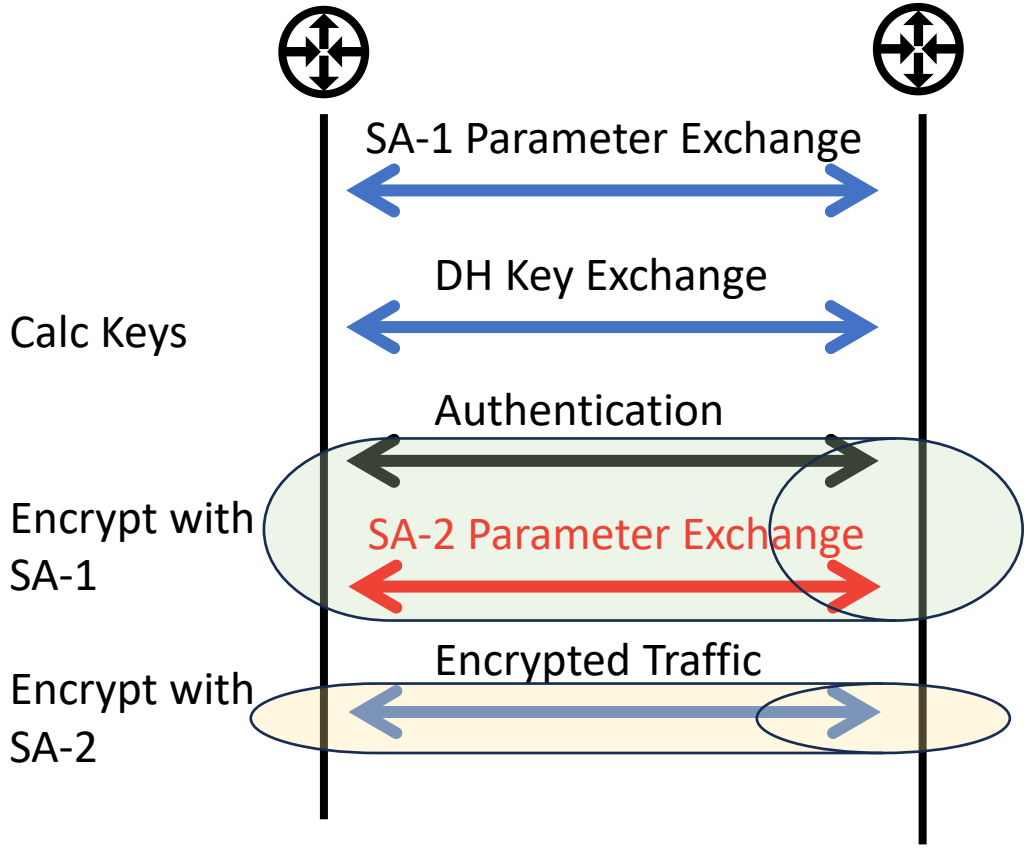
Authentication

- To verify each side is communicating with who they think they are
- Option decided in SA-1 Param exchange:
  - Shared Private Key
  - Digital Certificate
- This step uses keys determined with the DH Key Exchange step



SA-2 Parameter Exchange

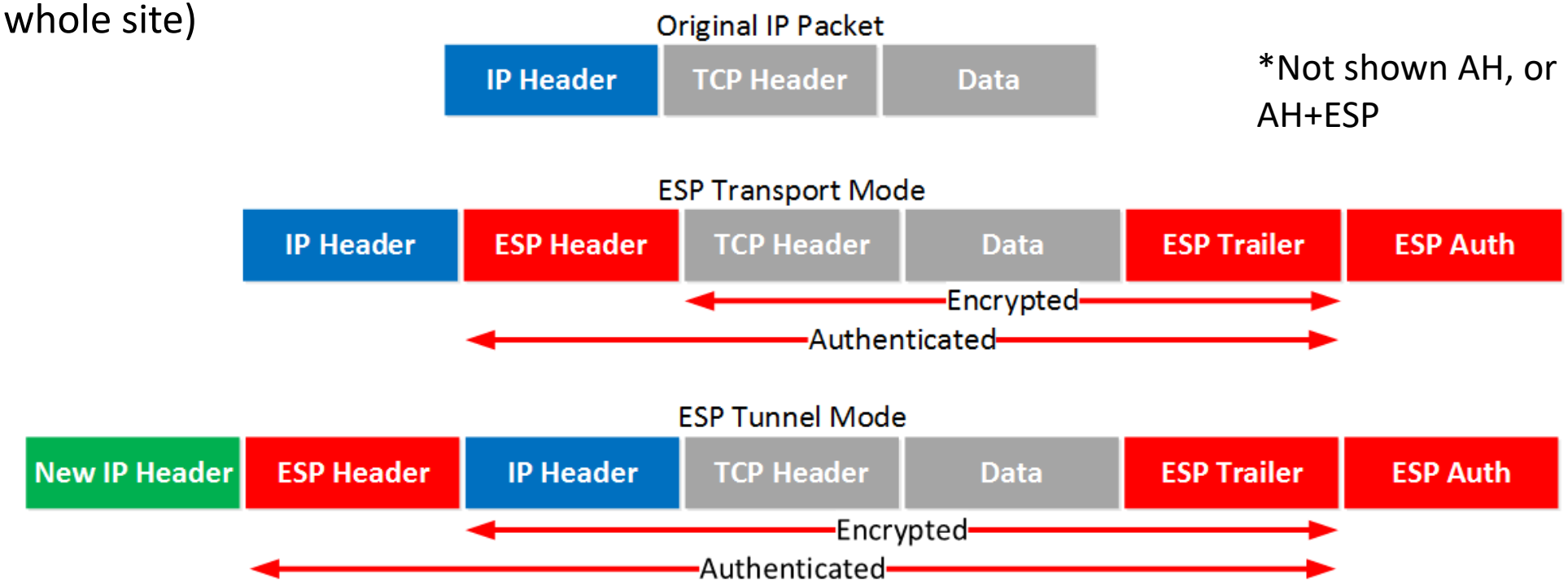
- Sets up security association for traffic exchange:
  - in case different parameters are desired
- Also, used to set up traffic selectors which determine what traffic is allowed through the tunnel (5-tuple of IP src/dest, protocol, transport src/dest)



Traffic Exchange

Supports several modes:

- AH (Authenticating Header – provides just integrity) or
  ESP (Encapsulating Security Payload – provides integrity and confidentiality)
- Transport (mostly for single endpoint) or
  tunnel (mostly for whole site)



*Not shown AH, or AH+ESP

https://networklessons.com/cisco/ccie-routing-switching/ipsec-internet-protocol-security

Software

- From Vendors on most routers / firewalls – Juniper, Cisco, Palo Alto
- In cloud – AWS, Azure, GCP's VPN services are IPsec
- Open Source – Strong Swan

Summary

IPsec leveraged many of the techniques discussed in the Basics lesson

- Message integrity through HMAC
- Confidentiality with encryption using a shared private key
- Exchanging shared keys using Diffie Hellman
- Authentication with public keys or pre-shared private key

University of Colorado **Boulder**

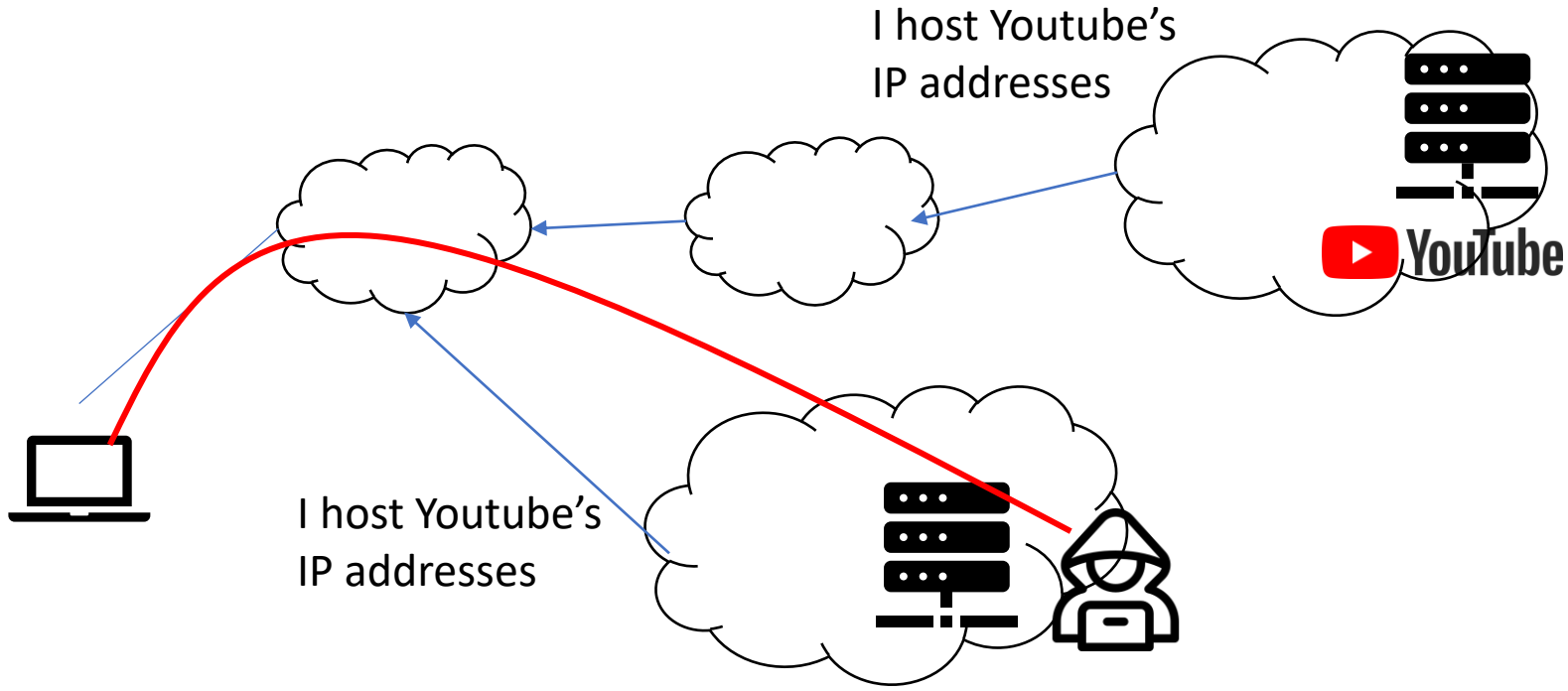
RPKI

Course: Networking Fundamentals
Module: Network Security

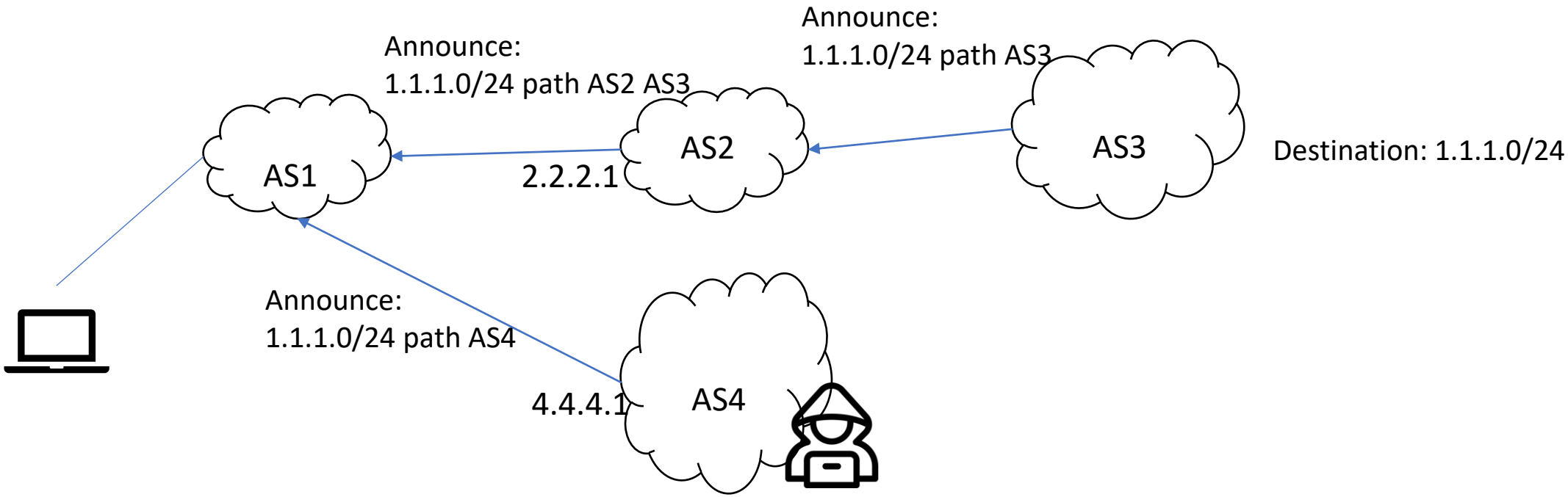
University of Colorado **Boulder**

Example 2 – Network Control Plane

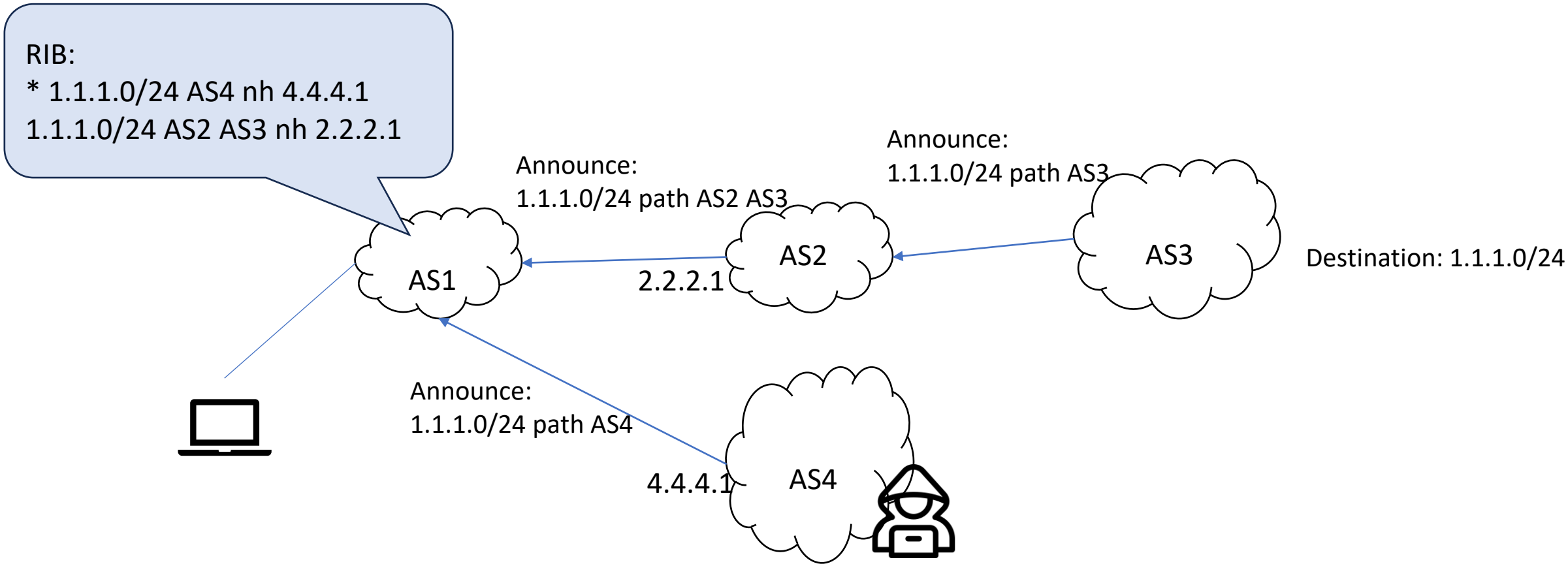
- BGP Communicates paths to reach IP prefixes
- If attacker can inject fake paths, it can get traffic directed to it (denying service, changing messages, inspecting traffic)



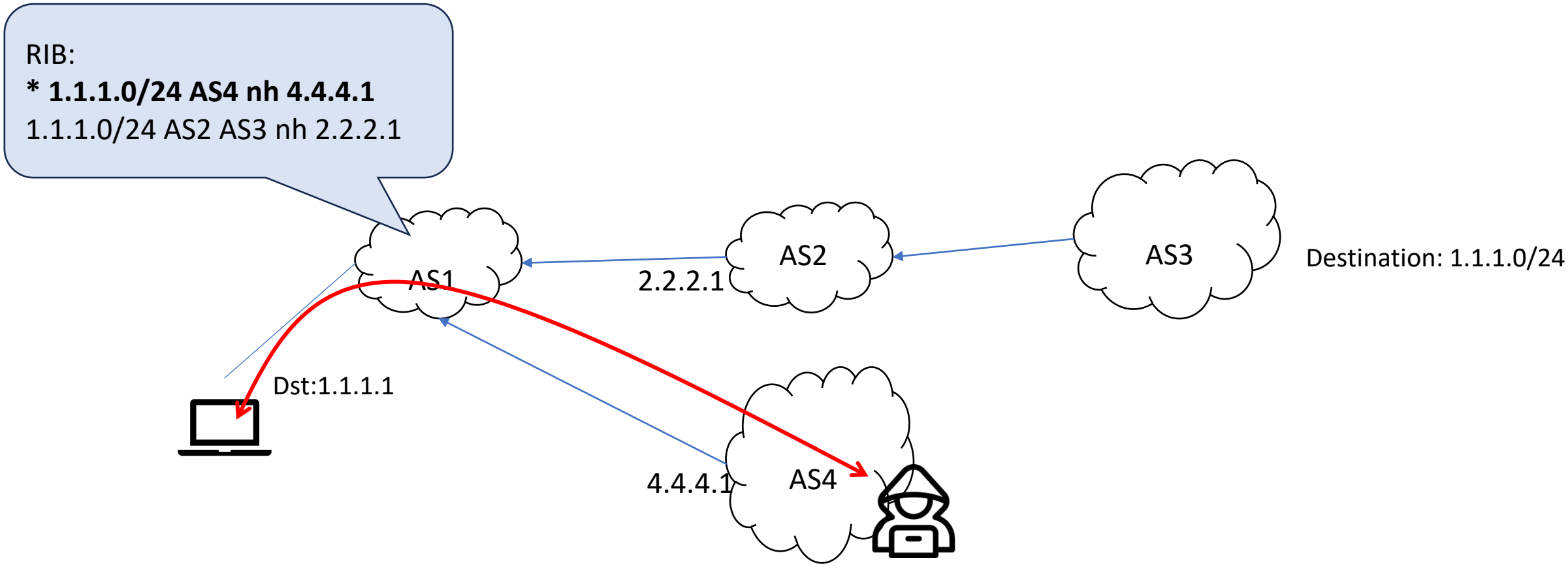
Concrete Example



Concrete Example

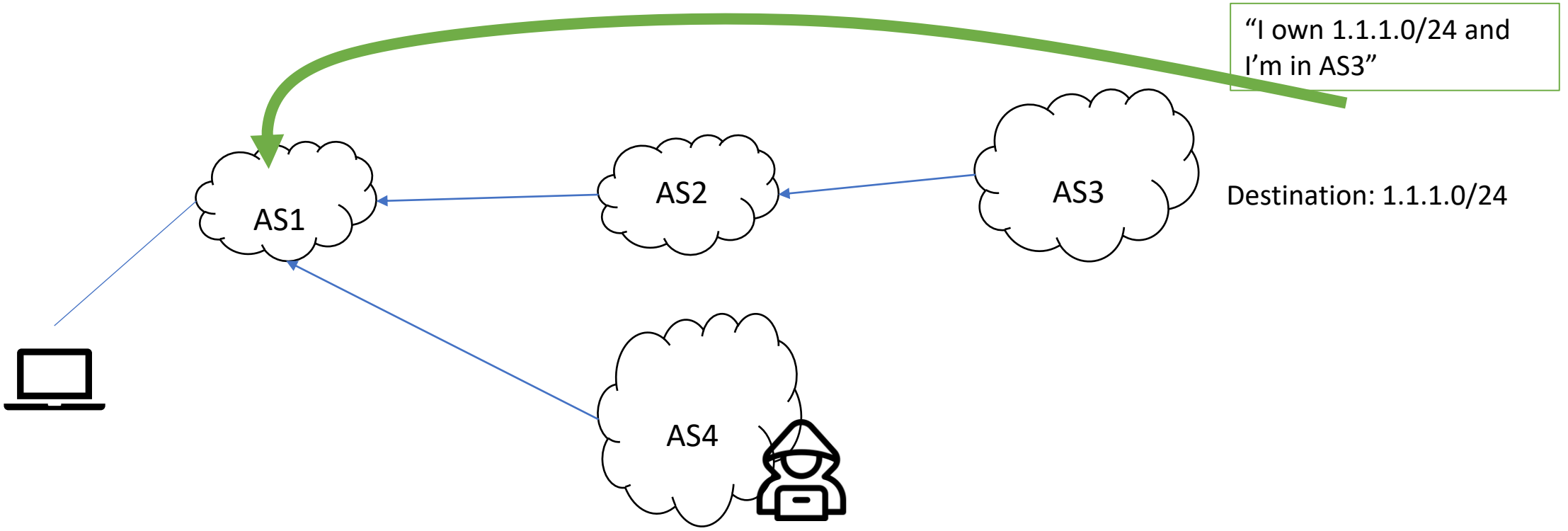


Concrete Example



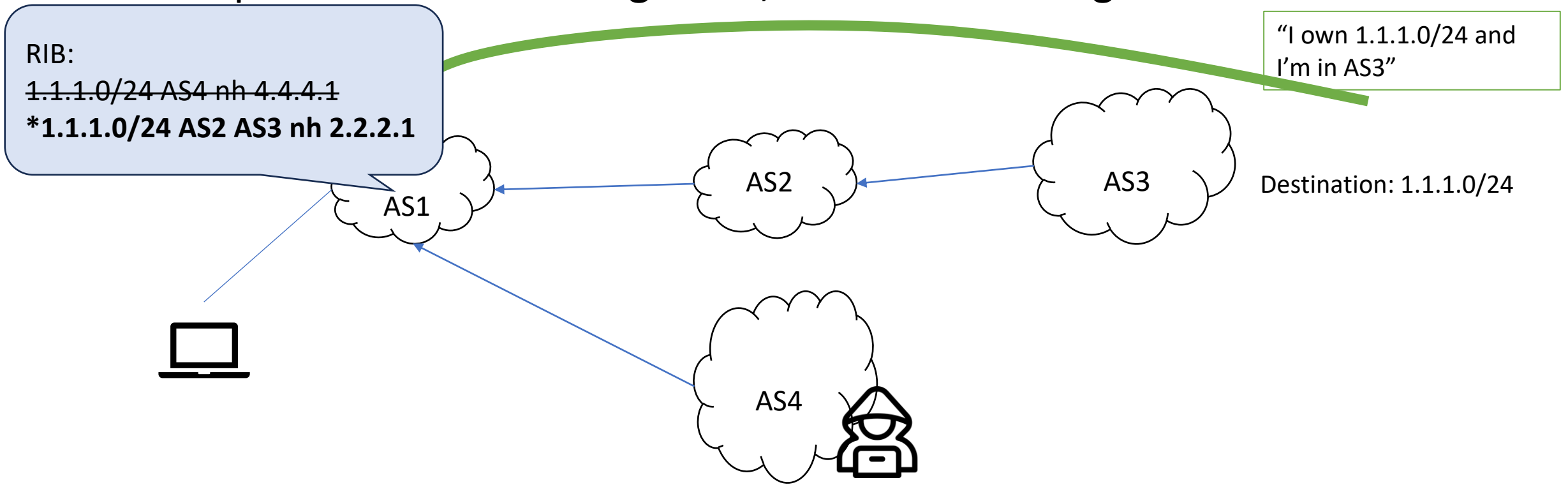
Idealized Solution

- Prefix owner tells ASes what AS it is in.
- The path is chosen through BGP, but needs to originate in that AS



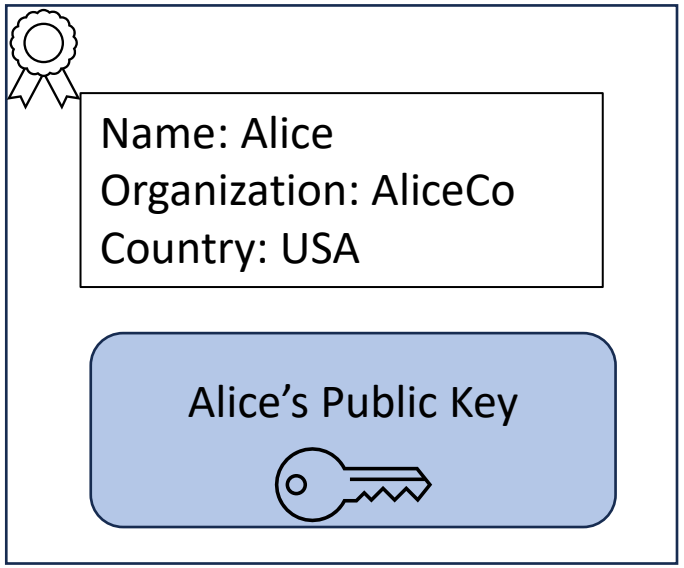
Idealized Solution

- Prefix owner tells ASes what AS it is in.
- The path is chosen through BGP, but needs to originate in that AS



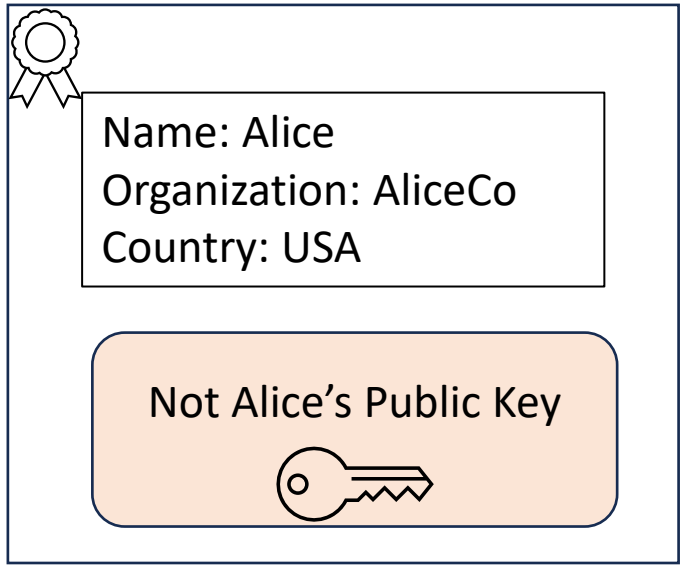
Recall: Digital Certificates

- Binds public key and identity
  (definition of identity depends on the context)
- Bob can validate that this key is Alice's



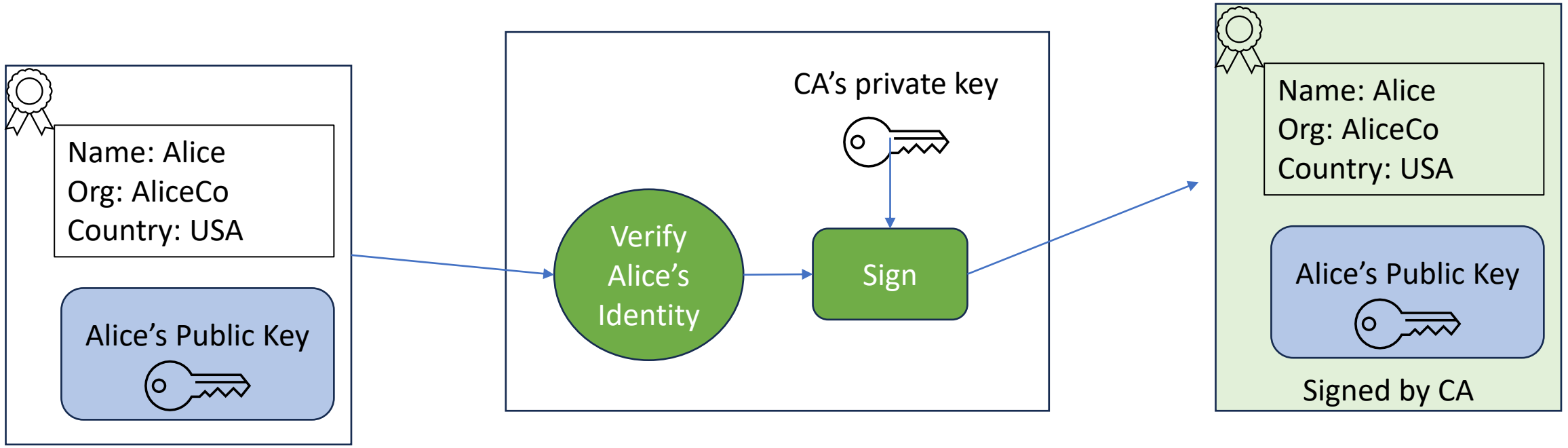
Recall: Problem – Integrity of Certificate

- Attacker could modify the certificate



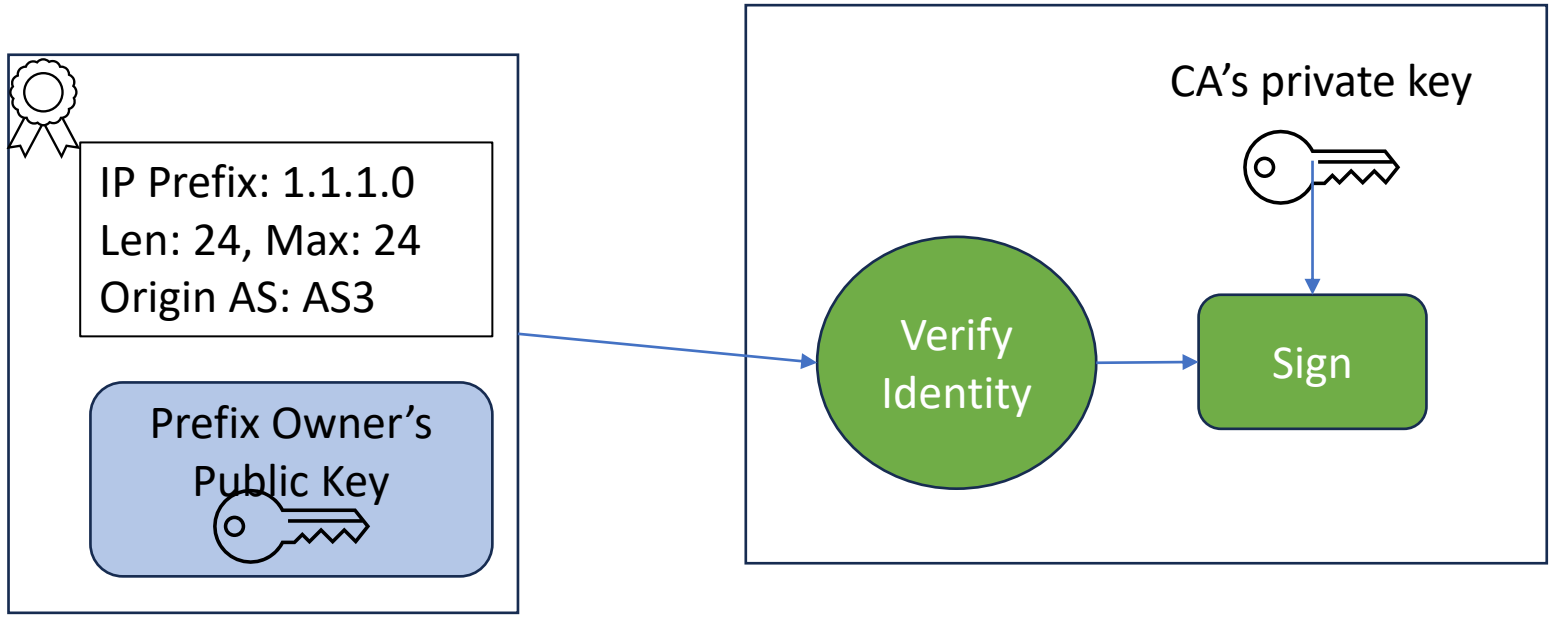
Recall: Certificate Authority

- Some trusted authority can verify Alice's identity and use its private key to sign
- Anybody can verify the digital certificate using CA's public key
- CA's public key has to be trusted (e.g., it's pre-installed in a browser)

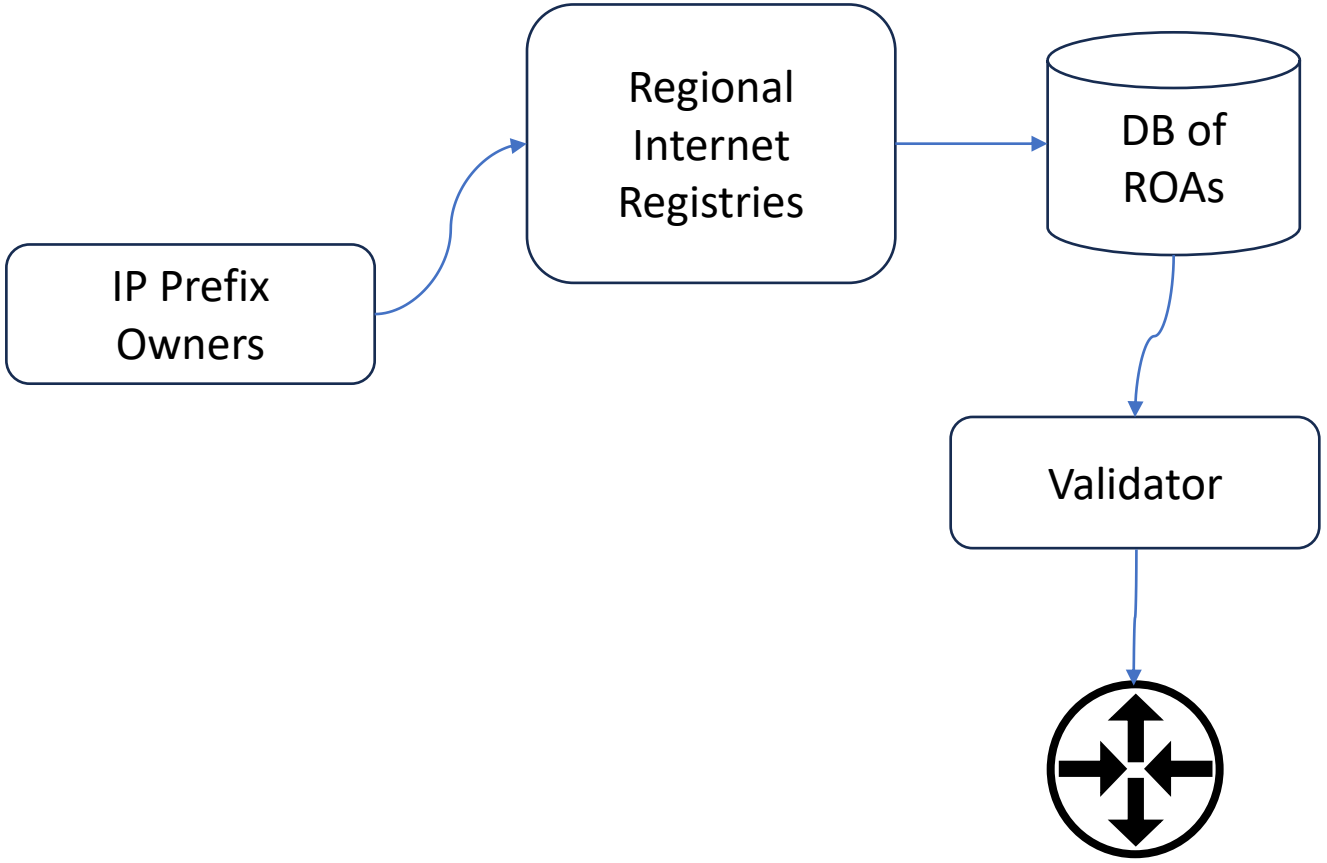


RPKI – Resource Public Key Infrastructure

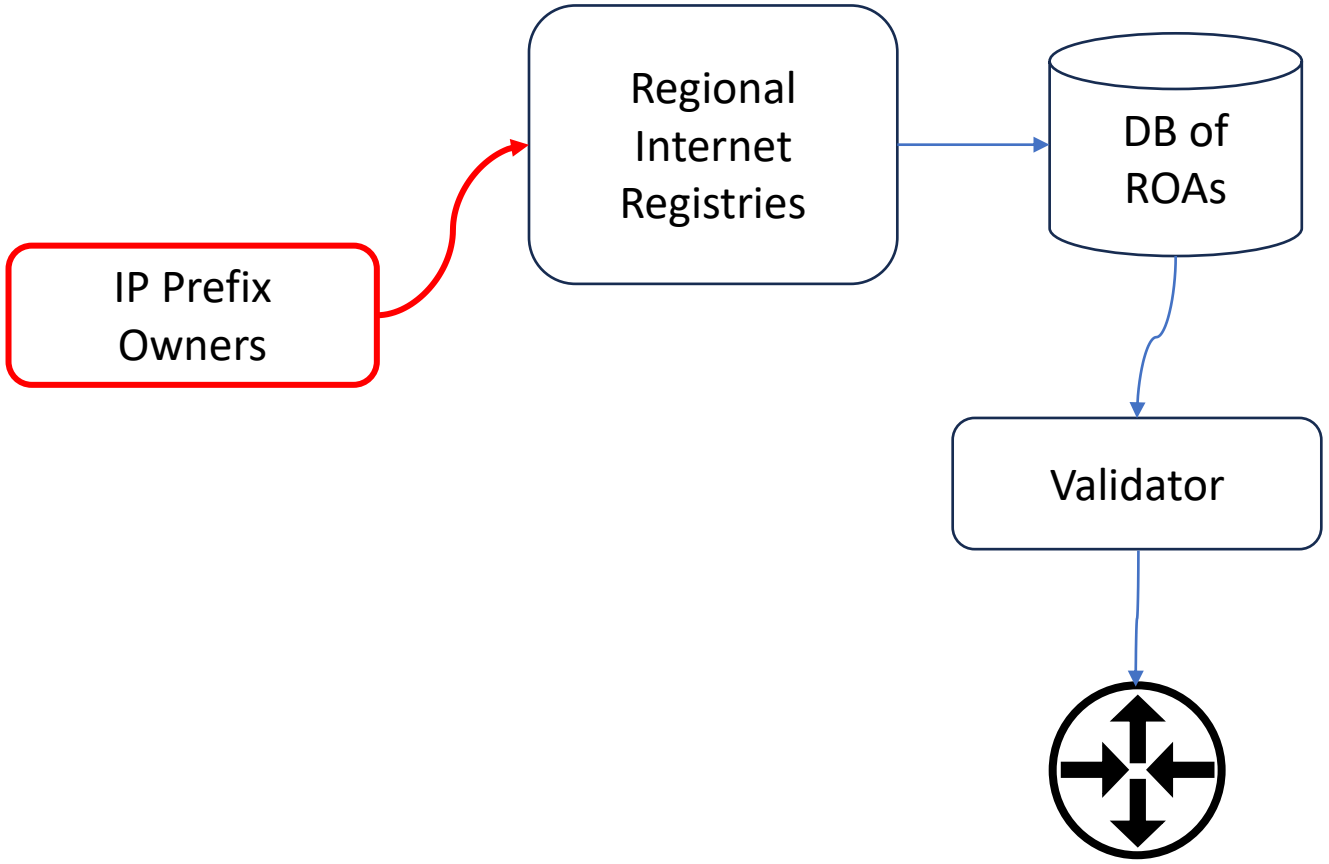
- Regional Internet Registries (RIRs) established as CAs
  - They are the ones that allocate IP address space – so, there is already a business relationship so they can verify the Identity
- Digital Certificates contain IP prefix, AS Origin, and tie that to owner



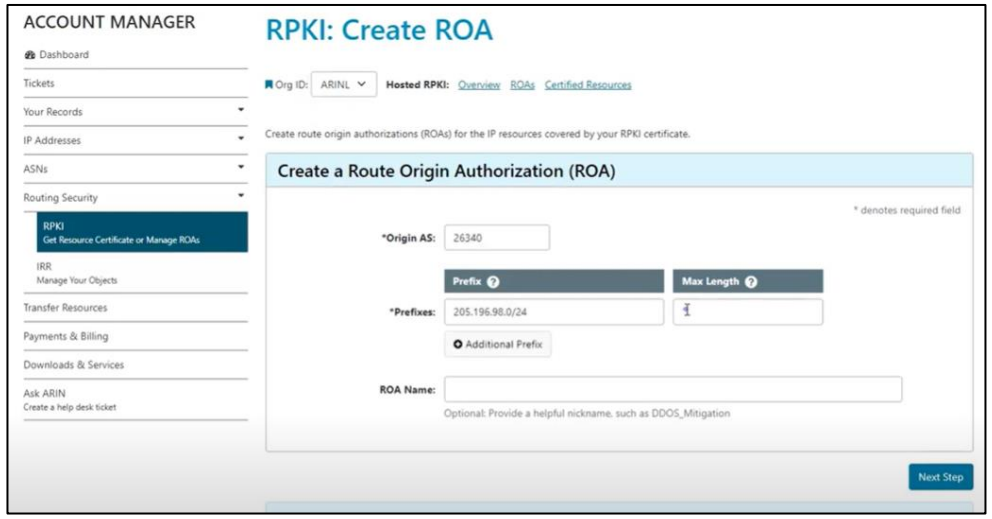
RPKI Architecture



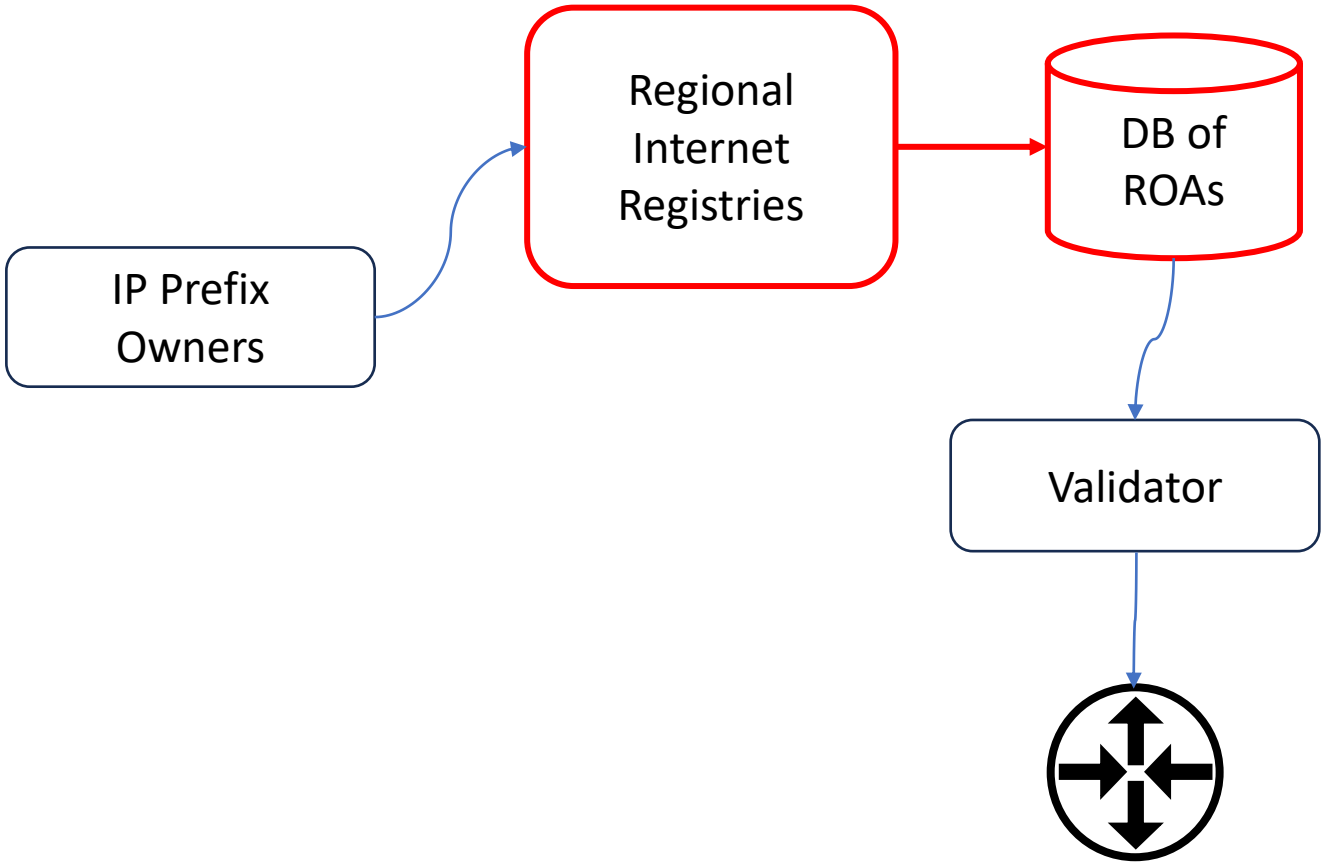
RPKI Architecture



IP Owners create a public/private key pair, Create digital certificate request and sends to the Regional Internet Registry

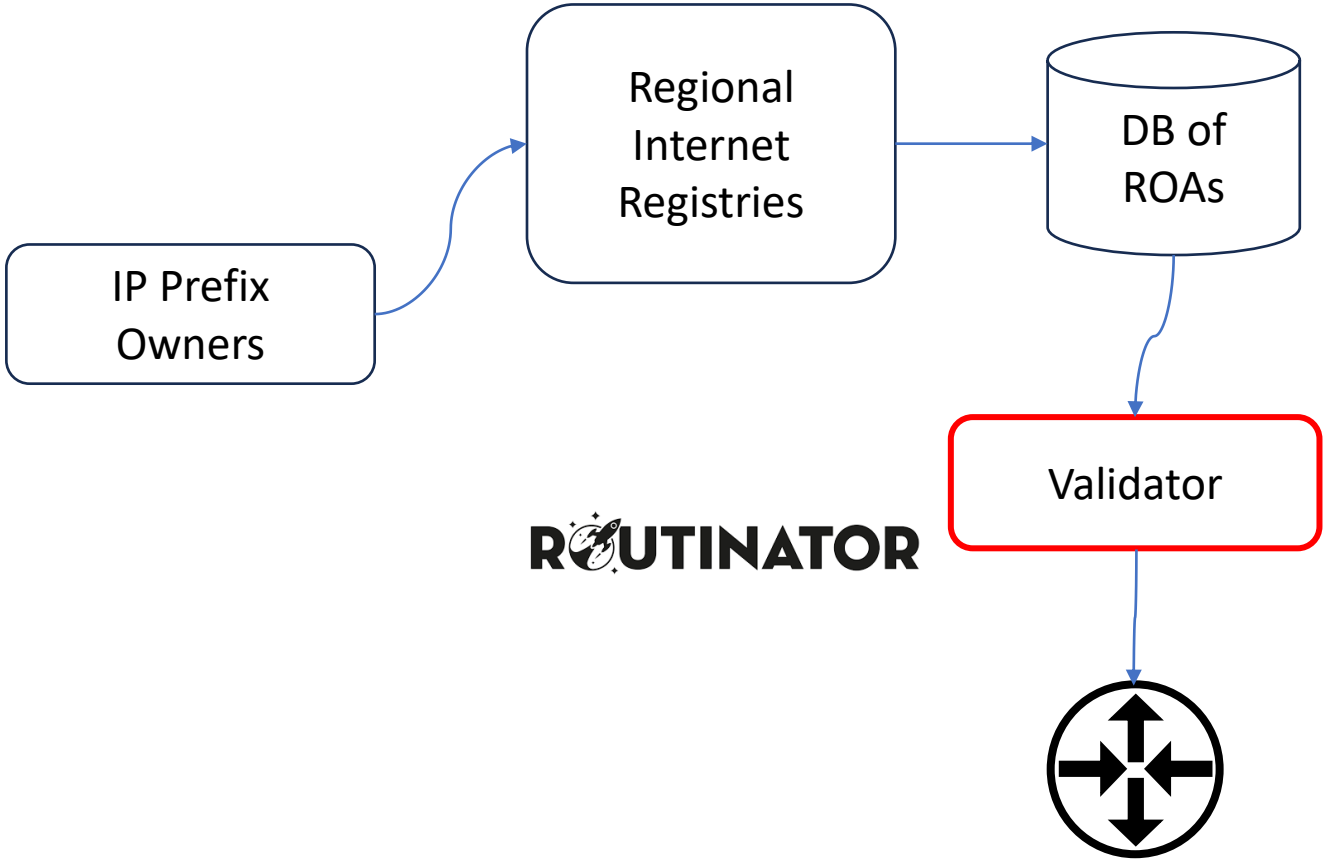


RPKI Architecture



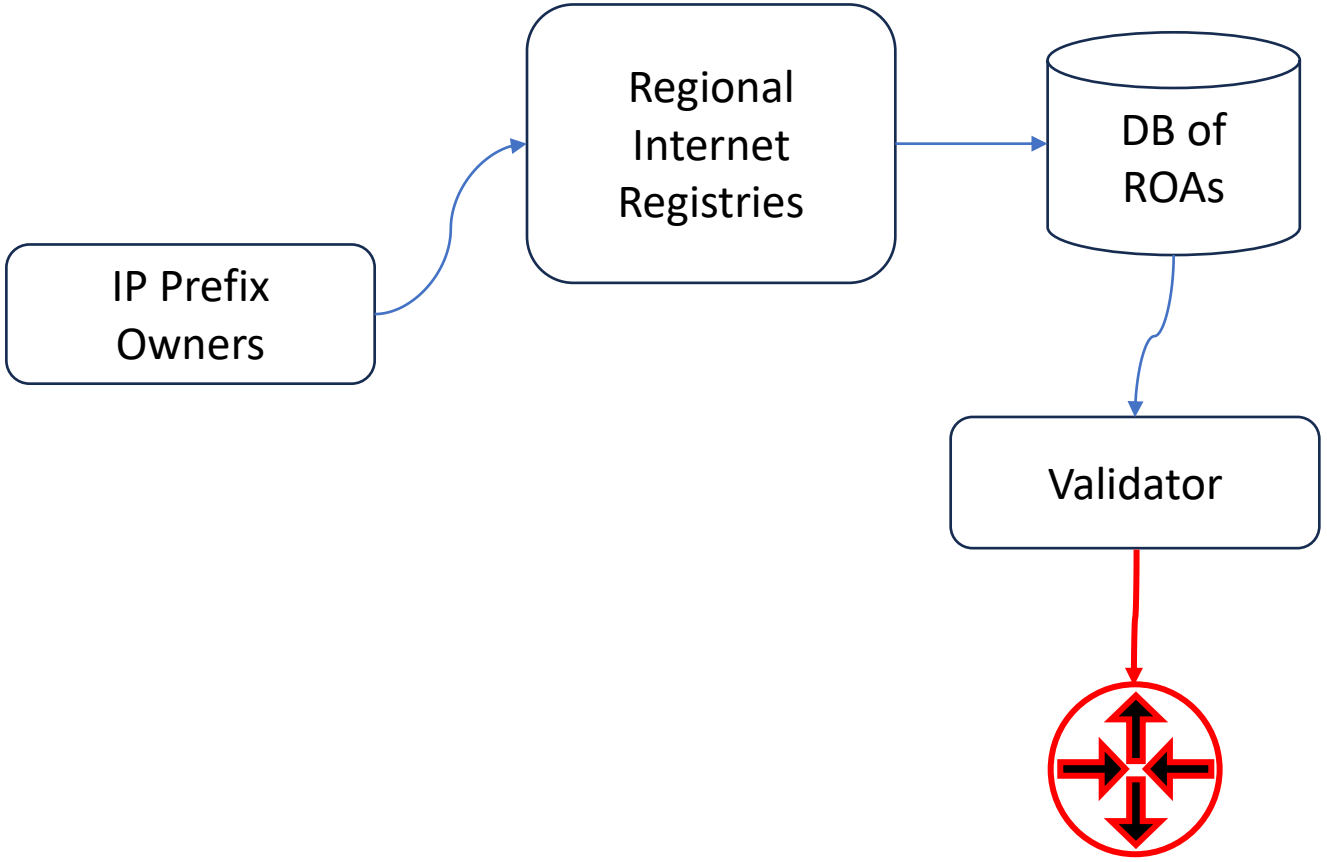
The RIR validates the identity and ownership, and stores a Resource Ownership Authorization (ROA) in a Database

RPKI Architecture



Software (called a validator) that runs in an AS deploying RPKI downloads the ROAs and verifies the signatures

RPKI Architecture



The validator installs a cache of the verified prefixes / route origins into the routers.

The router, upon receiving a BGP update, checks the prefix state (Valid, Invalid, Unknow) and accepts/rejects it accordingly

University of Colorado **Boulder**

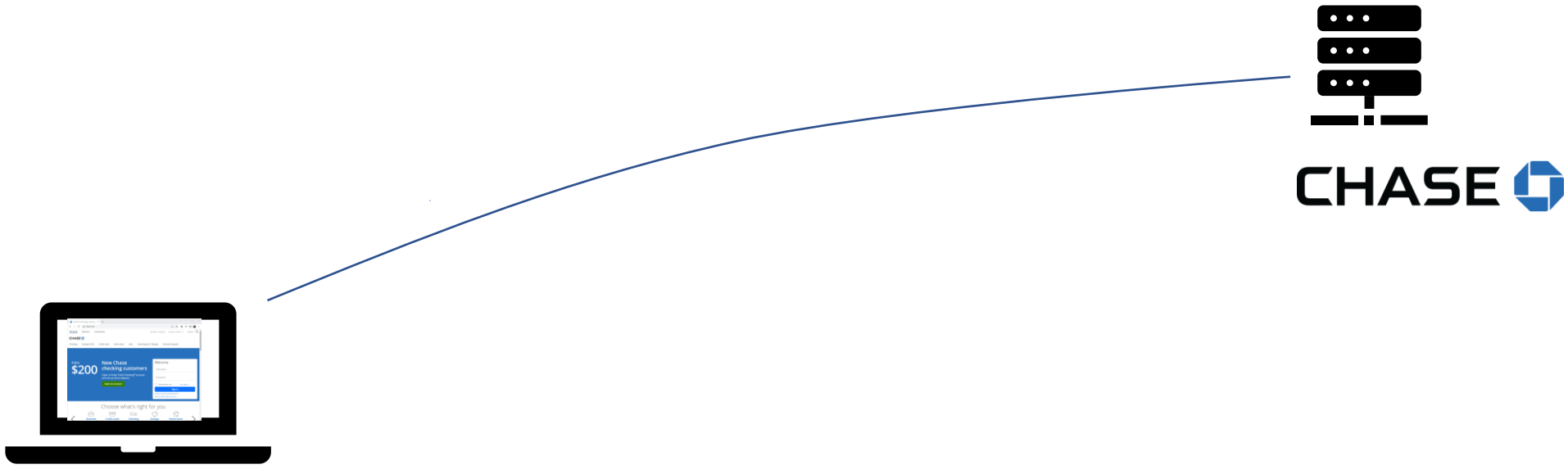
TLS / HTTPS

Course: Networking Fundamentals
Module: Network Security

University of Colorado **Boulder**

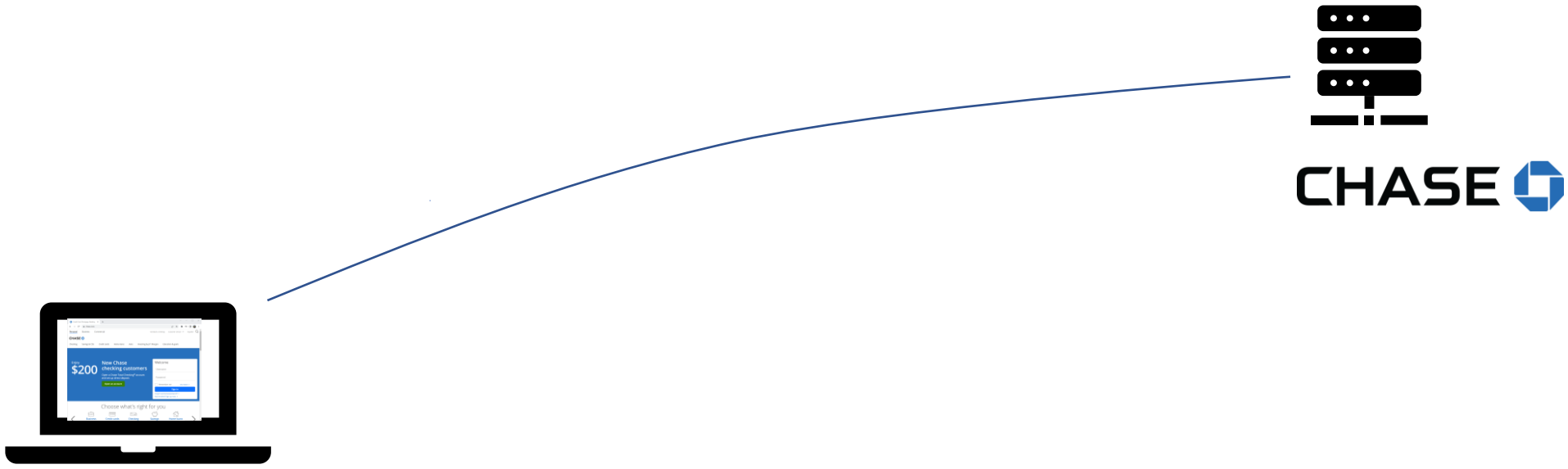
Example 3 – Application / Transport Layer

- TCP provides in-order reliable stream between two processes
- HTTP provides application protocol to get and put web objects
- Attacker could pretend to be the bank, or inspect / modify traffic (e.g., snooping on wi-fi in a public space)



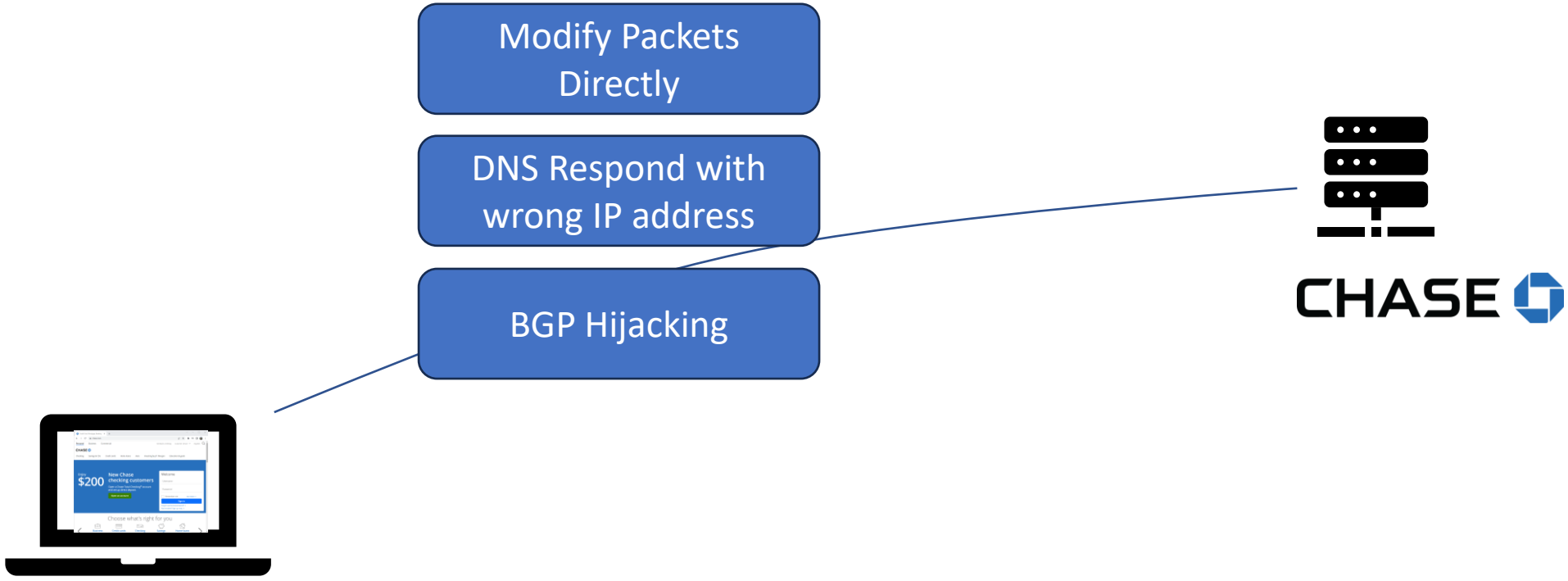
Is this threat real?

- DNS Query to lookup chase.com and return IP address
- Message(s) to/from IP address returned from DNS



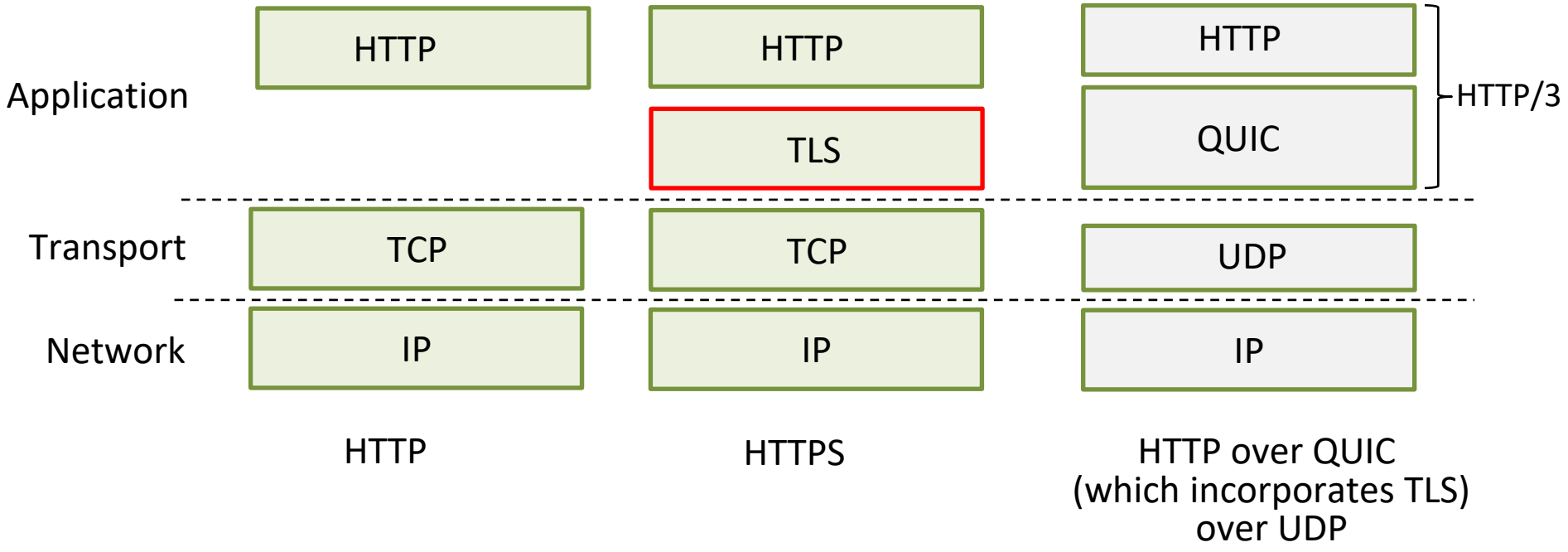
Is this threat real?

- DNS Query to lookup chase.com and return IP address
- Message(s) to/from IP address returned from DNS



HTTPS / TLS Overview

- TLS is a layer that provides security (confidentiality, integrity, authentication)



Problem of HTTP / TCP summarized

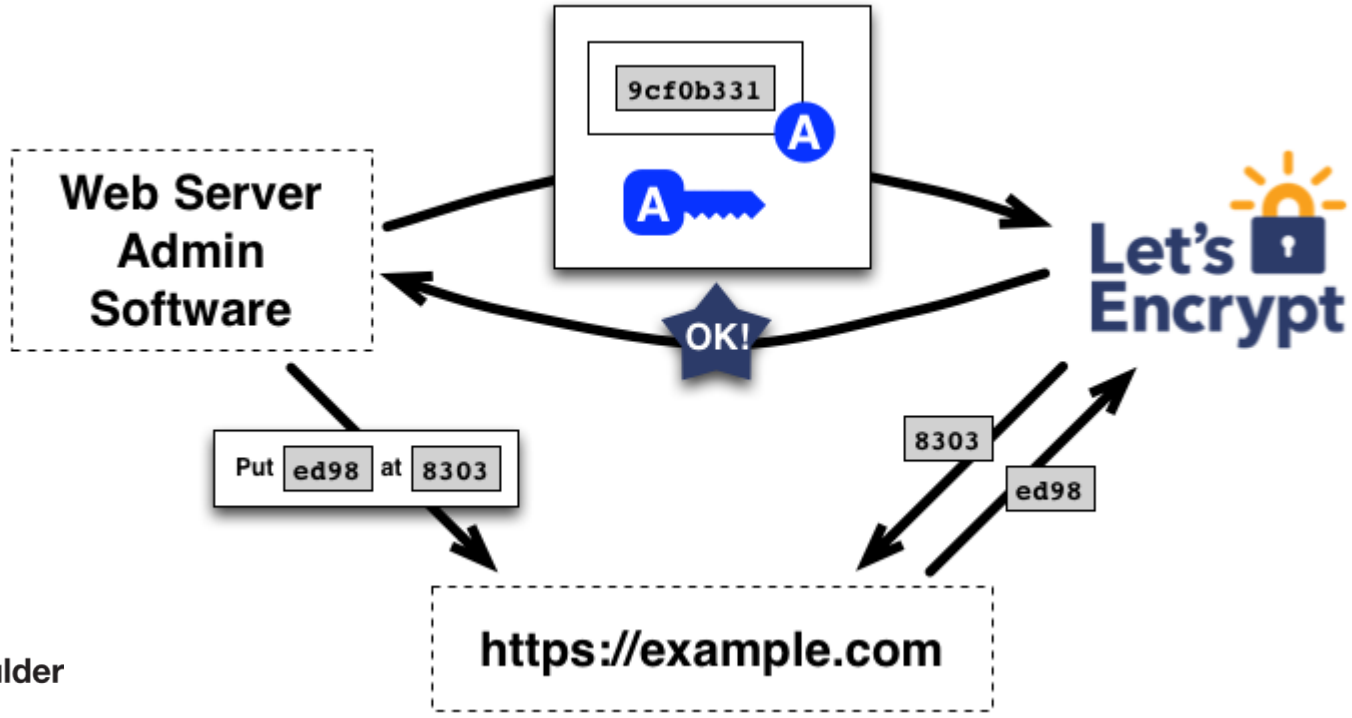
- Messages sent in plaintext
- Message integrity not ensured
- Not guaranteed communicating with desired server

Solutions

- Symmetric Encryption and Message Authentication Codes – with shared secret key (provides confidentiality and integrity)
  - But: How do we share a private key?
- Asymmetric encryption to exchange key (encrypt shared key with public key of server and server decrypts with their private key)
  - But: How do we trust the public key of the server?
- Digital Certificates and Certificate Authorities – bind identity to public key, and signed by a trusted source
  (also provides authentication)

Let’s Encrypt - Verifying Identity

- It verifies Identity (ownership of domain) through a challenge
  - Provisioning a DNS record under example.com, or
  - Provisioning an HTTP resource under a well-known URI

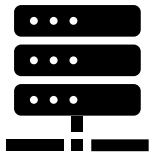


TLS Handshake

- Happens after the TCP Connection is established (3 way handshake)
- Sets up agreement on parameters, authenticates the server, and sets up a private key
- Then, encrypted traffic can be exchanged

TLS 1.2 Handshake

Client Hello



Tells server of what the client can support, and what it prefers.
e.g., ciphers supported (DES, AES), data integrity algorithms (SHA1, MD5)

TLS 1.2 Handshake

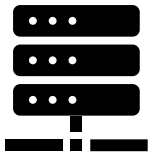
Server picks the parameters and the client what it chose.

TLS 1.2 Handshake

Client Hello

Server Hello

Server Certificate



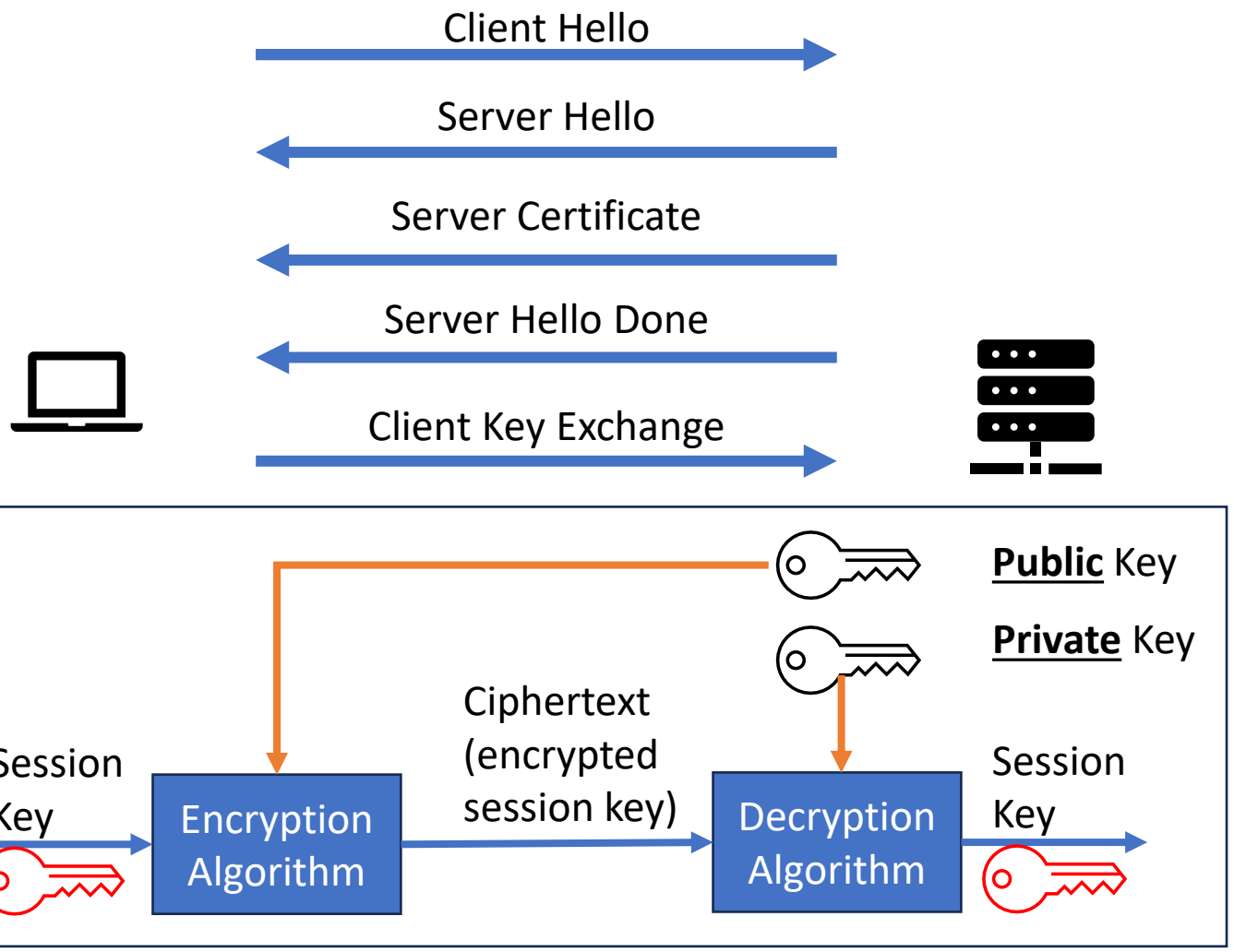
Server sends its digital certificate signed by a CA. This binds a domain name to a public key of the server.

The client will have root certificates installed, and verifies the server's certificate

TLS 1.2 Handshake

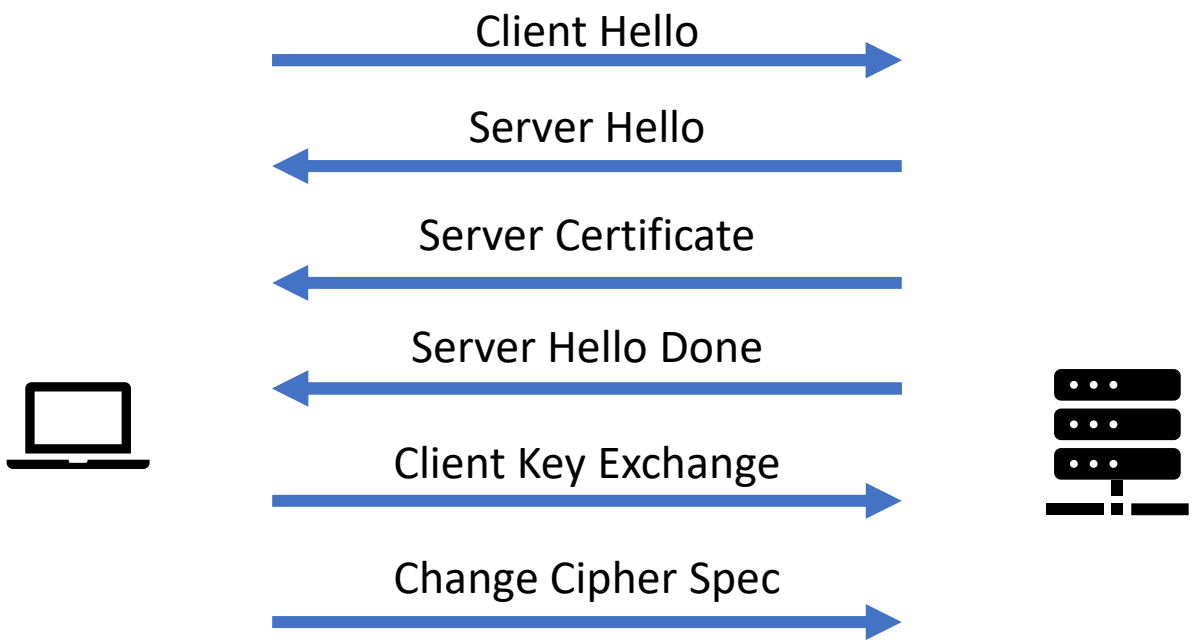
Server indicates it is done with its handshake negotiation

TLS 1.2 Handshake



Server indicates it is done with its handshake negotiation

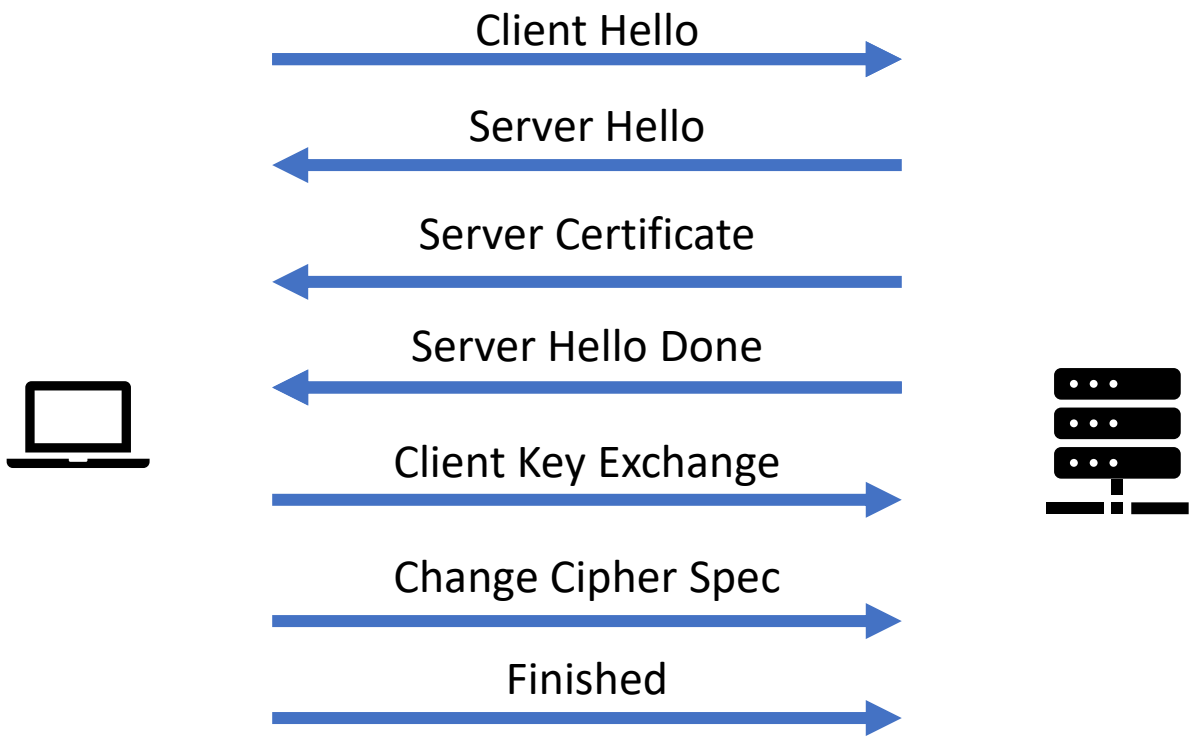
TLS 1.2 Handshake



Client indicates that it now believes parameters are agreed upon.

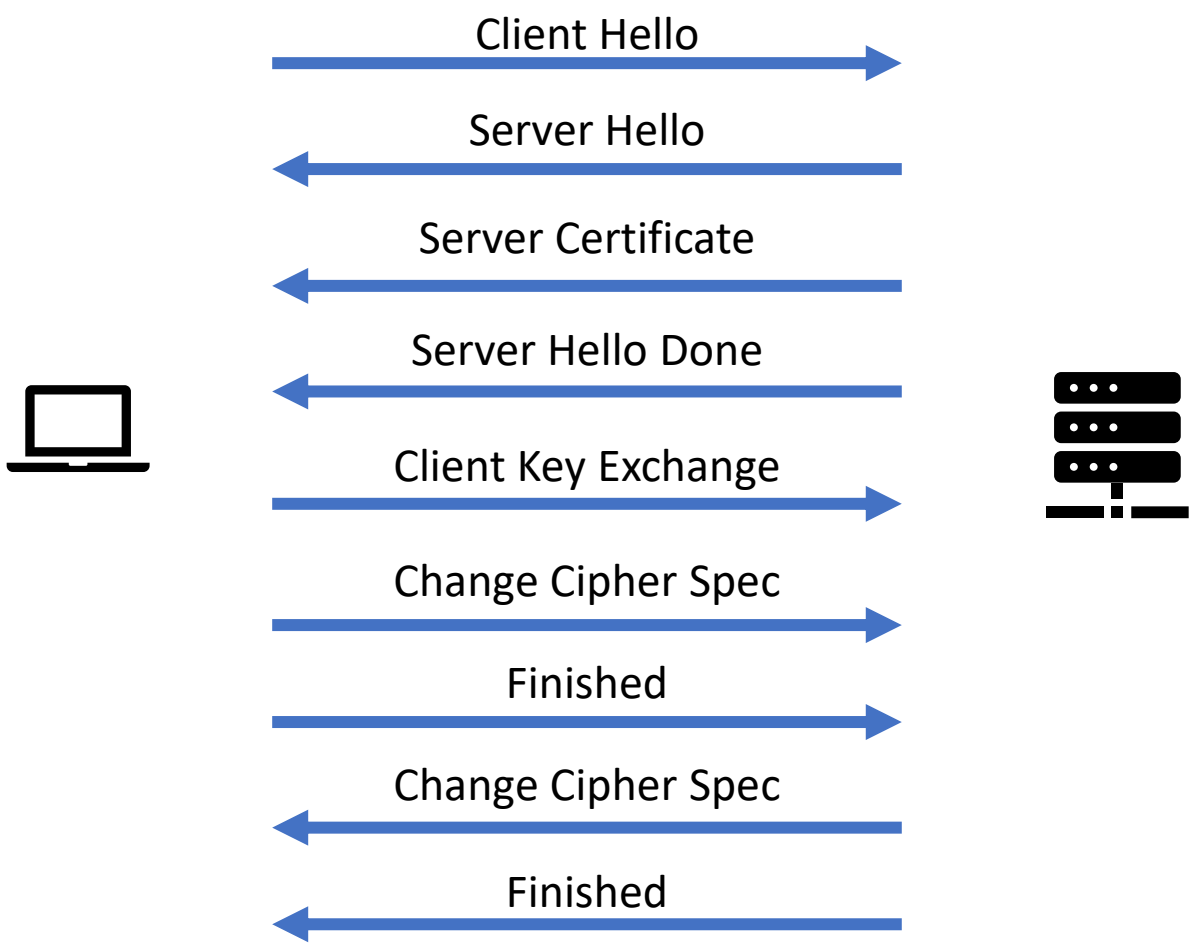
Next message will be encrypted

TLS 1.2 Handshake



Client then sends a summary of all messages to this point, ensuring integrity and that none have been tampered with

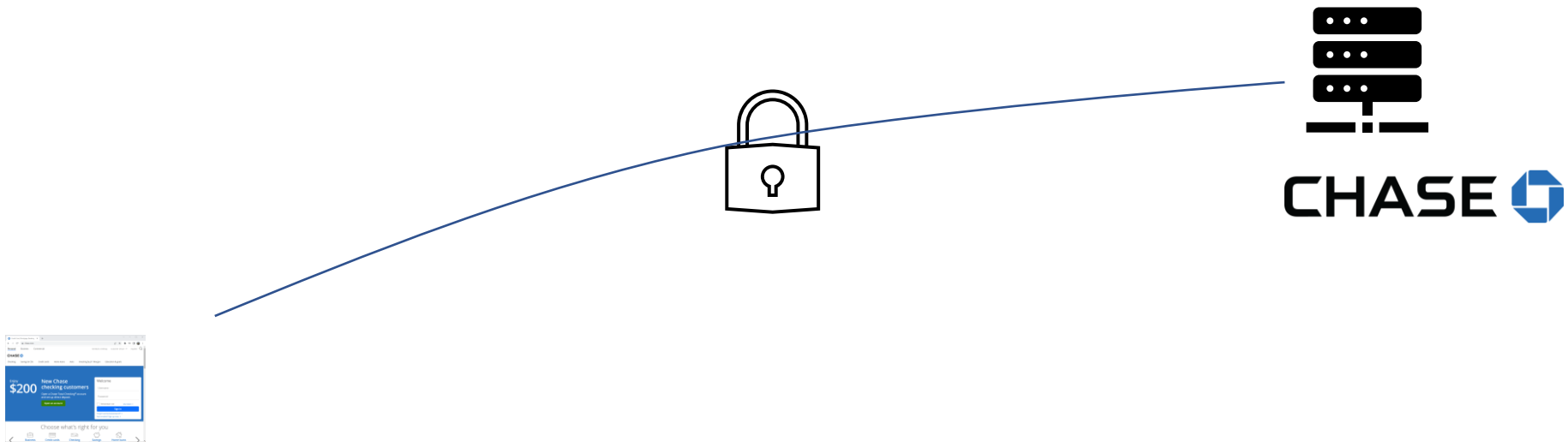
TLS 1.2 Handshake



Server then also sends:

1) Message indicating that it has everything
2) Summary of all messages to this point

Now – Communication is Secured



Mutual TLS (mTLS)

- As described, client verifying Identity of server is enough (that the domain of the public key matches website client visiting)
- Server may want to verify identity of clients as well
  - E.g., Access control for API end points
- Browers (in client) would look at domain in certificate and match to what's being accessed (i.e., implicit accept all)
- Servers would likely have an explicit access control list

University of Colorado **Boulder**