15-440 Distributed Systems

21 – Security Protocols - 1

Tuesday, Nov 16th, 2021

Logistical Updates

- P3 released – Due 11/23 (CHKP), 12/3 Final
  - Remember, you get 2 late days (no questions asked).
- HW4 – Release 11/17, Due 11/29
  - **NO LATE DAYS. Need to release solutions.**
- P3: 2 late day policy, please don't ask for more
  - Unless it is a major emergency (e.g. medical reasons)
- Midterm II – Thursday 12/2, 10:10am – 11:30am
  - In class. Please come 10mins early early to set up.
- ***Class webpage is most up to date for logistics***

Building User-Focused Sensing Systems

Spring 2022 I 17-422 / 17-722

Instructors: Mayank Goel, Yuvraj Agarwal

Sensors are at the core of most computers

Learn how these sensors work & build your own sensing systems

LEARN

Machine Learning | Signal Processing

Mobile Computing | Computer Vision

3D Printing and Milling | Embedded Computing

Website: https://www.synergylabs.org/courses/17-722

Prerequisites: Love programming, tinkering, and thinking out of the box!

Today's Lecture

- Internet security weaknesses
- Establishing secure channels (Crypto 101)
- Key distribution

What is "Internet Security" ?

Denial-of-Service

Password Cracking

Traffic modification

Worms & Viruses

Trojan Horse

DNS Poisoning

Phishing

Spyware

IP Spoofing

End-host impersonation

Route Hijacks

Traffic Eavesdropping

Spam

Internet Security: Prevent bad things from happening on the internet!

Internet Design Decisions: (ie: how did we get here?)

- Origin as a small and cooperative network
  - (➔ largely trusted infrastructure)
- Global Addressing
  - (➔every sociopath is your next-door neighbor)
- Connection-less datagram service
  - (➔can’t verify source, hard to protect bandwidth)

Internet Design Decisions: (ie: how did we get here?)

- Anyone can connect
  - (➔ ANYONE can connect)
- Millions of hosts run nearly identical software
  - (➔ single exploit can create epidemic)
- Most Internet users know about as much as Senator Stevens aka “the tubes guy”
  - (➔ God help us all…)

Our “Narrow” Focus

Yes:

1) Creating a “secure channel” for communication

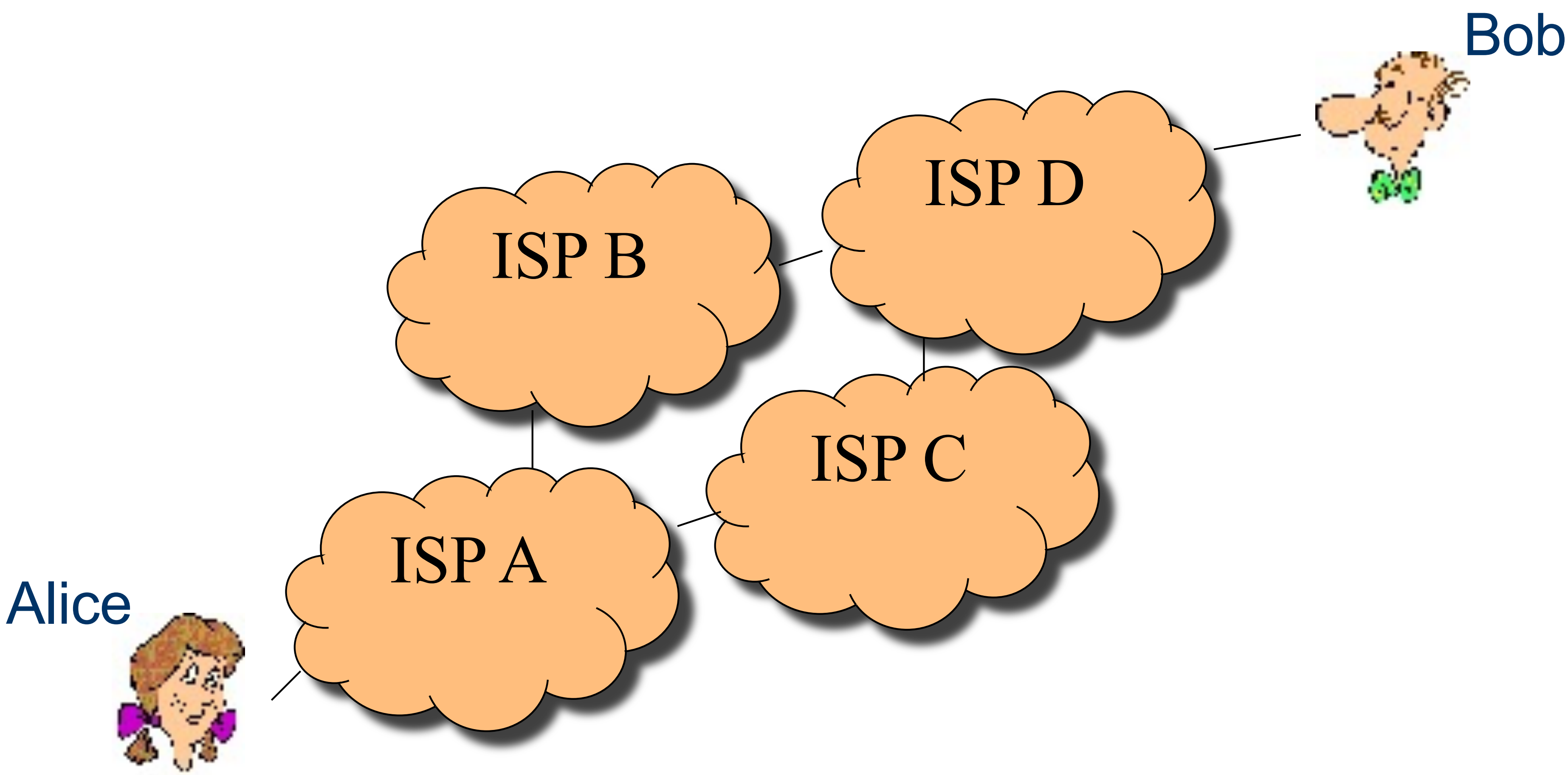
Some:

2) Protecting resources and limiting connectivity

No:

1) Preventing software vulnerabilities & malware, or “social engineering”.

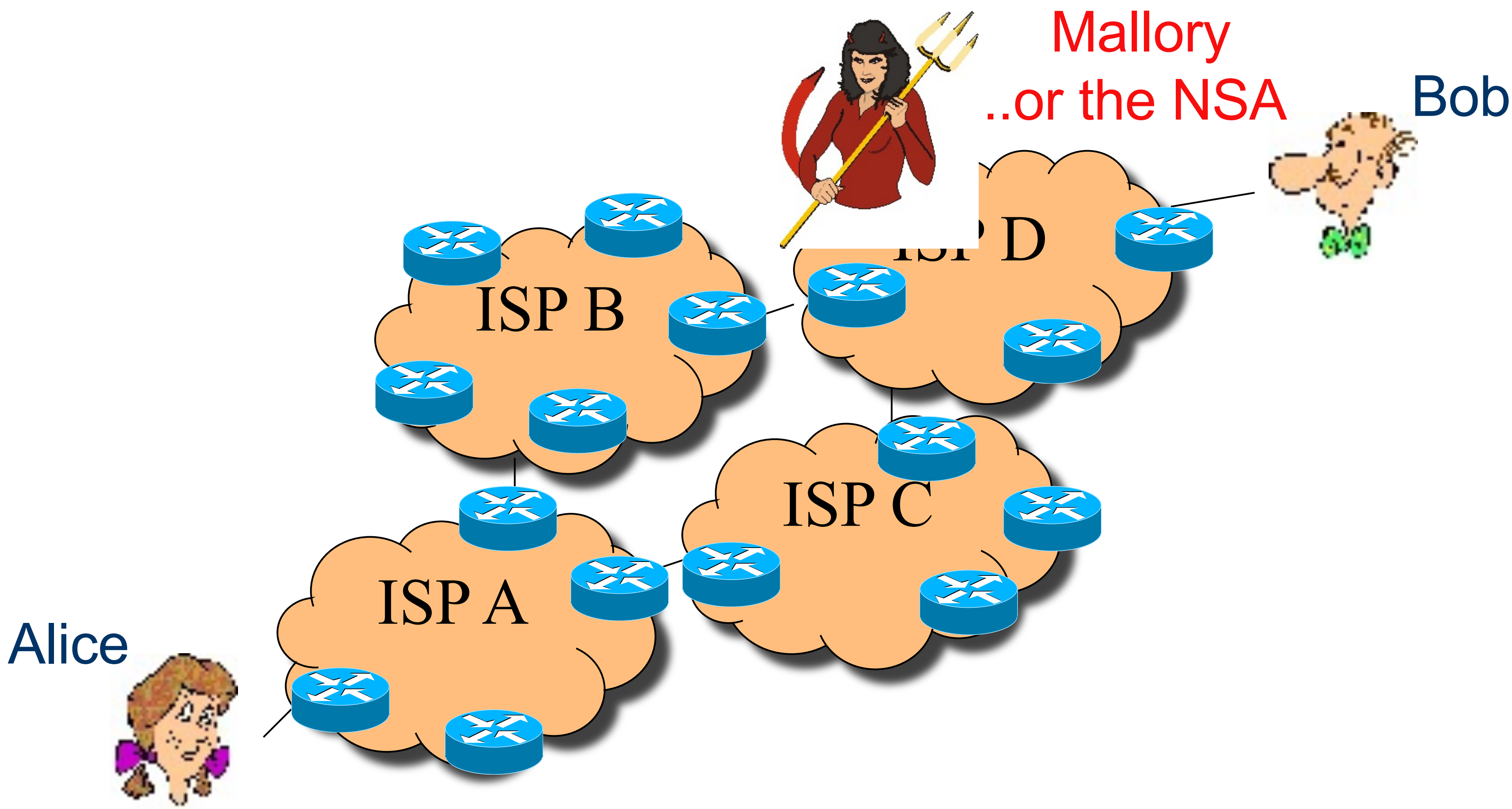
Secure Communication with an Untrusted Infrastructure



What do we need for a secure communication channel?

- Authentication (Who am I talking to?)
- Confidentiality (Is my data hidden?)
- Integrity (Has my data been modified?)
- Availability (Can I reach the destination?)

Example: Eavesdropping - Message Interception (Attack on Confidentiality)

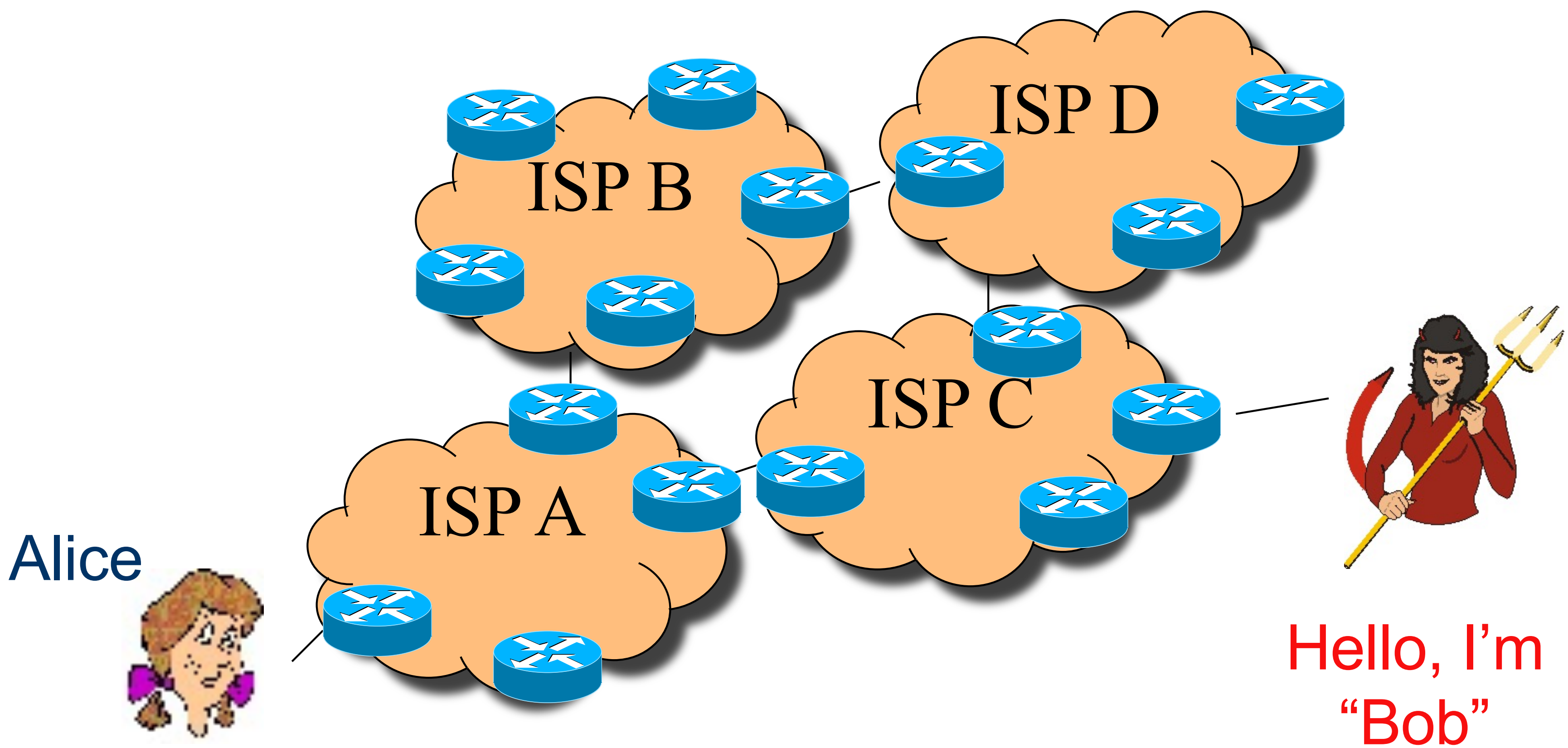


Eavesdropping Attack: Example

- tcpdump with promiscuous network interface
  - On a “**switched**” ethernet network, what can you see?
  - On the WiFi network in this room what can you see?

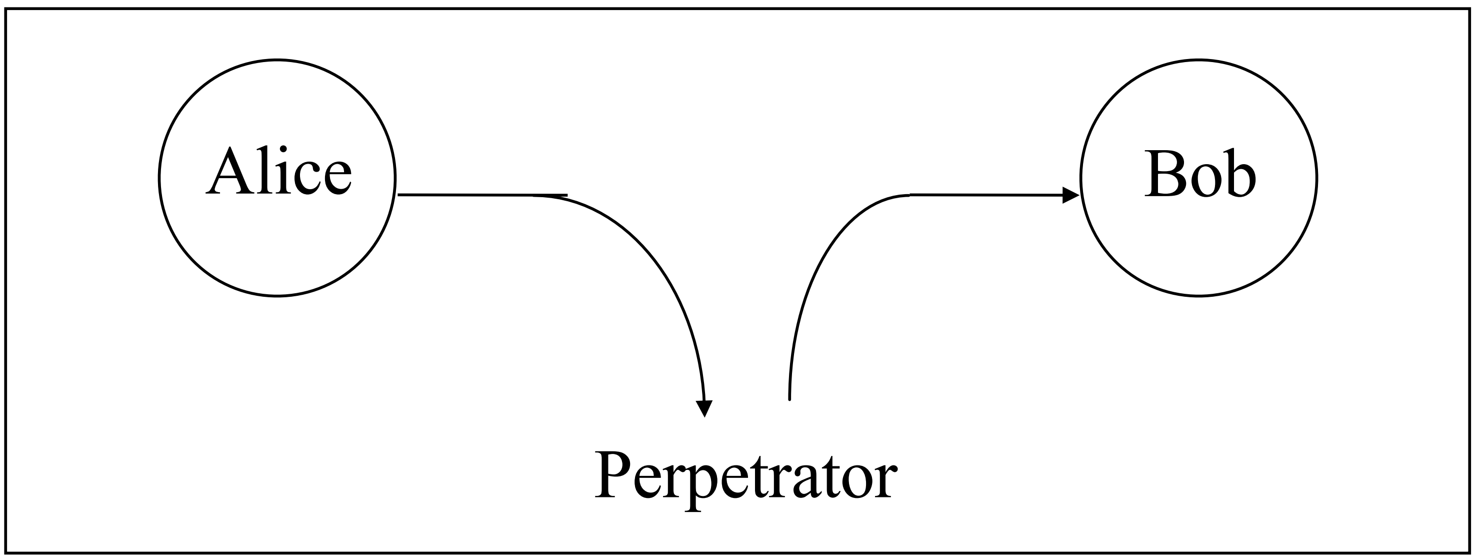
- What might the following traffic types reveal about communications?
  - Full IP packets with unencrypted data
  - Full IP packets with encrypted payloads
  - Just DNS lookups (and replies)

Authenticity Attack - Fabrication



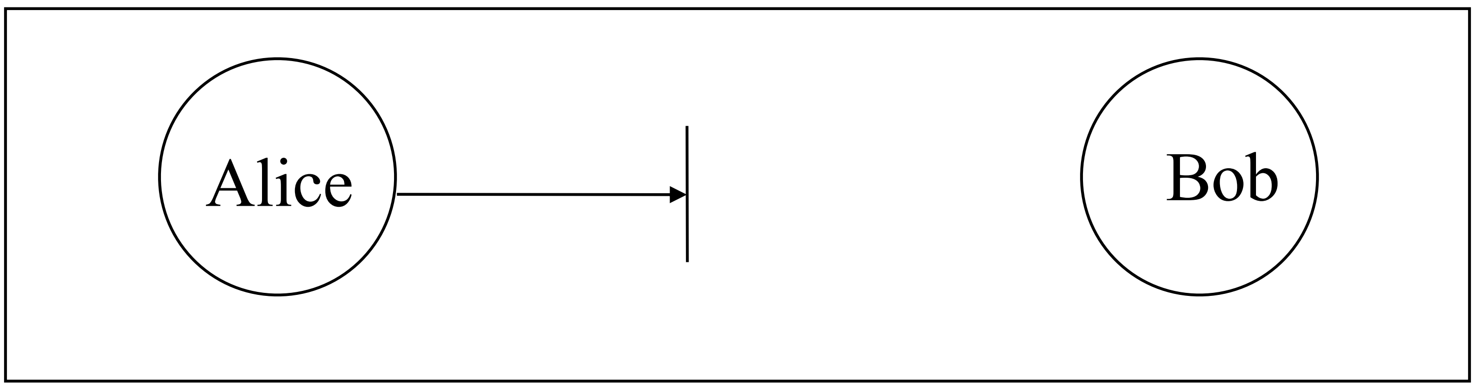
Integrity Attack - Tampering

- Stop the flow of the message
- Delay and optionally modify the message
- Release the message again



Attack on Availability

- Destroy hardware (cutting fiber) or software
- Modify software in a subtle way
- Corrupt packets in transit



- Blatant denial of service (DoS):
  - Crashing the server
  - Overwhelm the server (use up its resource)

Example: Web access

- Alice wants to connect to her bank to transfer some money...
- Alice wants to know ...
  - that she's really connected to her bank. — Authentication
  - That nobody can observe her financial data — Confidentiality
  - That nobody can modify her request — Integrity
  - That nobody can steal her money! — (A mix)
- The bank wants to know ...
  - That Alice is really Alice (or is authorized by Alice)
  - The same privacy things that Alice wants so they don't get sued or fined by the government.

Today's Lecture

- Internet security weaknesses
- Crypto 101
- Key distribution

Cryptography As a Tool

- Using cryptography securely is not simple
- Designing cryptographic schemes correctly is near impossible.

Today we want to give you an idea of what can be done with cryptography.

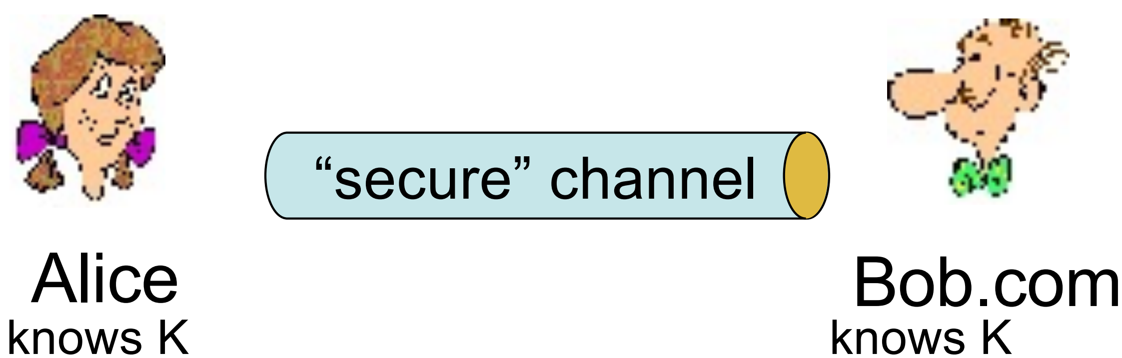
Take a security course if you think you may use it in the future (e.g. 18-487)

Well...

- What tools do we have at hand?
- Hashing
  - e.g., SHA-1
- Secret-key cryptography, aka symmetric key.
  - e.g., AES
- Public-key cryptography
  - e.g., RSA

Secret Key Cryptography

- Given a key k and a message m
  - Two functions: Encryption (E), decryption (D)
  - ciphertext c = E(k, m)
  - plaintext m = D(k, c)
  - Both use the same key k.



But... how does that help with authentication?

They both have to know a pre-shared key K before they start!

Symmetric Key: Confidentiality

Motivating Example:

You and a friend share a key K of L random bits, and a message M also L bits long.

Scheme:

You send her the *xor(M,K)* and then they "decrypt" using *xor(M,K)* again.

For example, the string "Wiki" (01010111 01101001 01101011 01101001 in 8-bit ASCII) can be encrypted with the repeating key 11110011 as follows:

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
  -----------------------------------
= 10100100 10011010 10011000 10011010
```

And conversely, for decryption:

```
  10100100 10011010 10011000 10011010
⊕ 11110011 11110011 11110011 11110011
  -----------------------------------
= 01010111 01101001 01101011 01101001
```

1) Do you get the right message to your friend?

2) Can an adversary recover the message M?

Symmetric Key: Confidentiality

- One-time Pad (OTP) is secure but usually impractical
  - Key is as long at the message
  - Keys cannot be reused (why?)

In practice, two types of ciphers are used that require only constant key length:

Stream Ciphers:

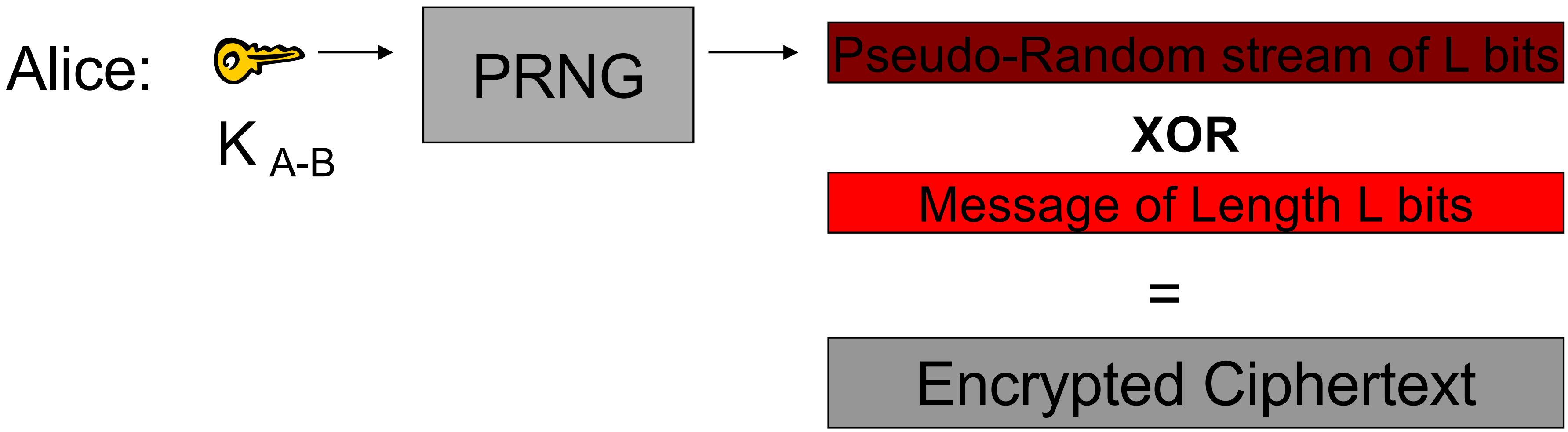
Ex: RC4, A5

Block Ciphers:

Ex: DES, AES, Blowfish

Symmetric Key: Confidentiality

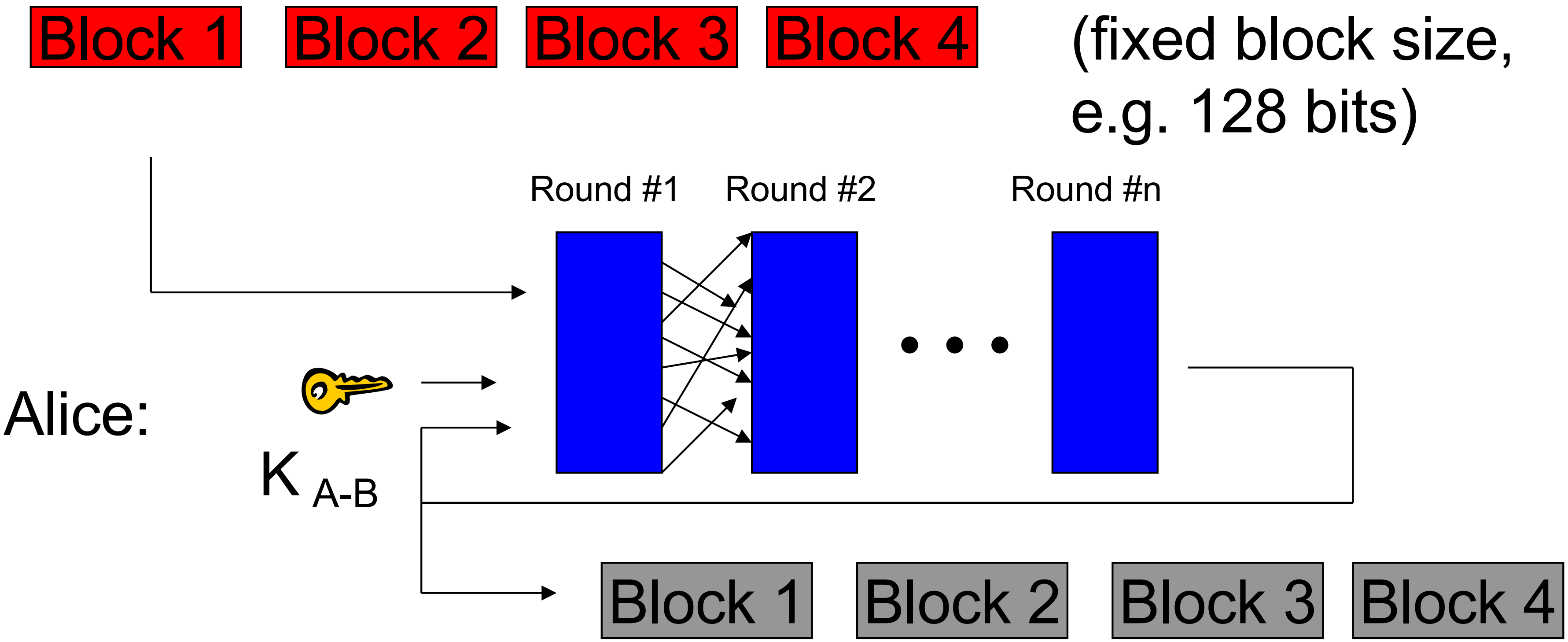
- Stream Ciphers (ex: RC4)



Bob uses $K_{A\text{-}B}$ as PRNG seed, and XORs encrypted text to get the message back (just like an OTP).

Symmetric Key: Confidentiality

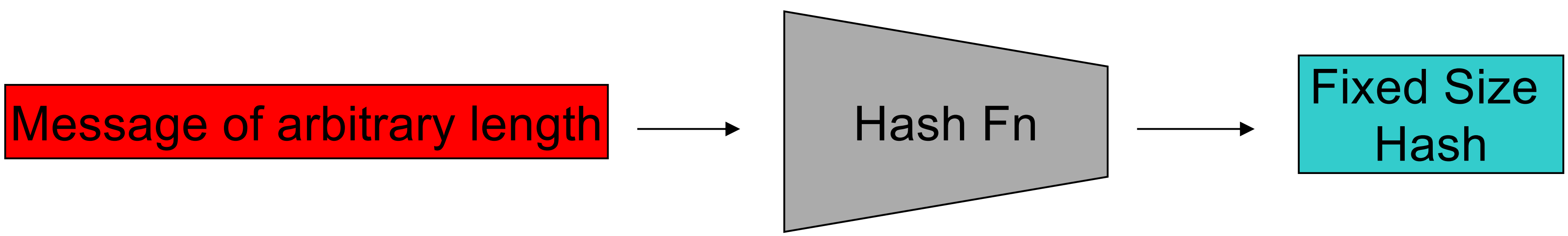
- Block Ciphers (ex: AES)



Bob breaks the ciphertext into blocks, feeds it through decryption engine using $K_{A\text{-}B}$ to recover the message.

Symmetric Key: Integrity

- Background: Hash Function Properties
  - Consistent: hash(X) always yields same result
  - One-way: given X, can't find Y s.t. hash(Y) = X
  - Collision resistant:
    given hash(W) = Z, can't find X such that hash(X) = Z

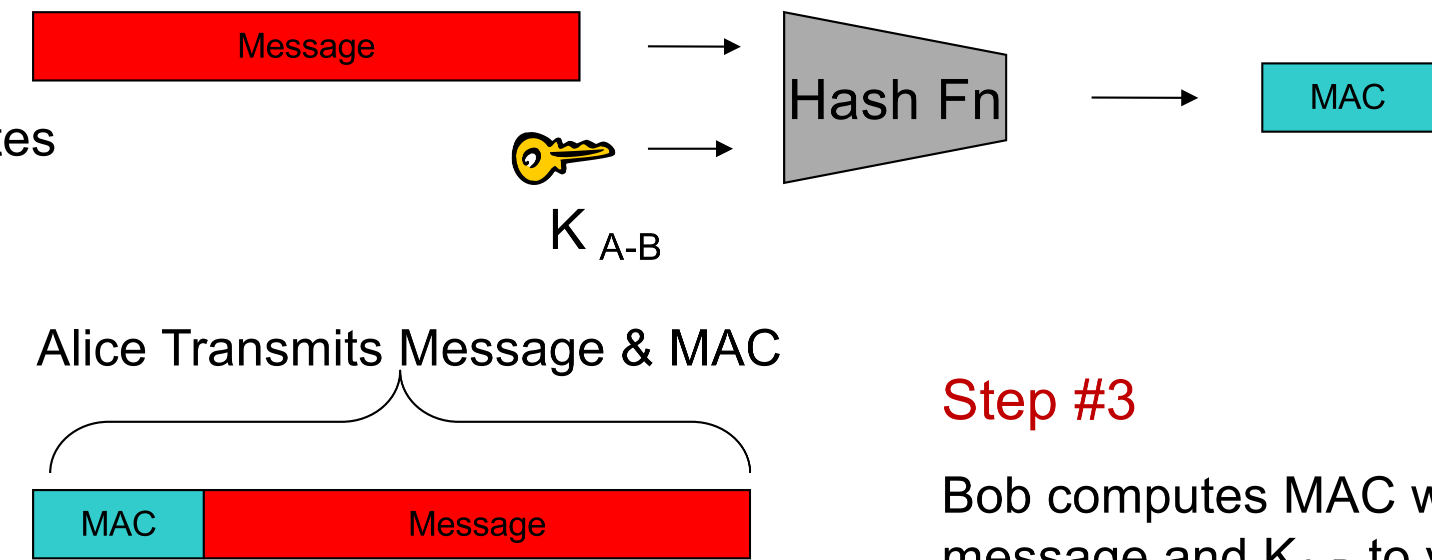


Symmetric Key: Integrity

- Hash Message Authentication Code (HMAC)

Step #1:

Alice creates MAC



Step #2

Step #3

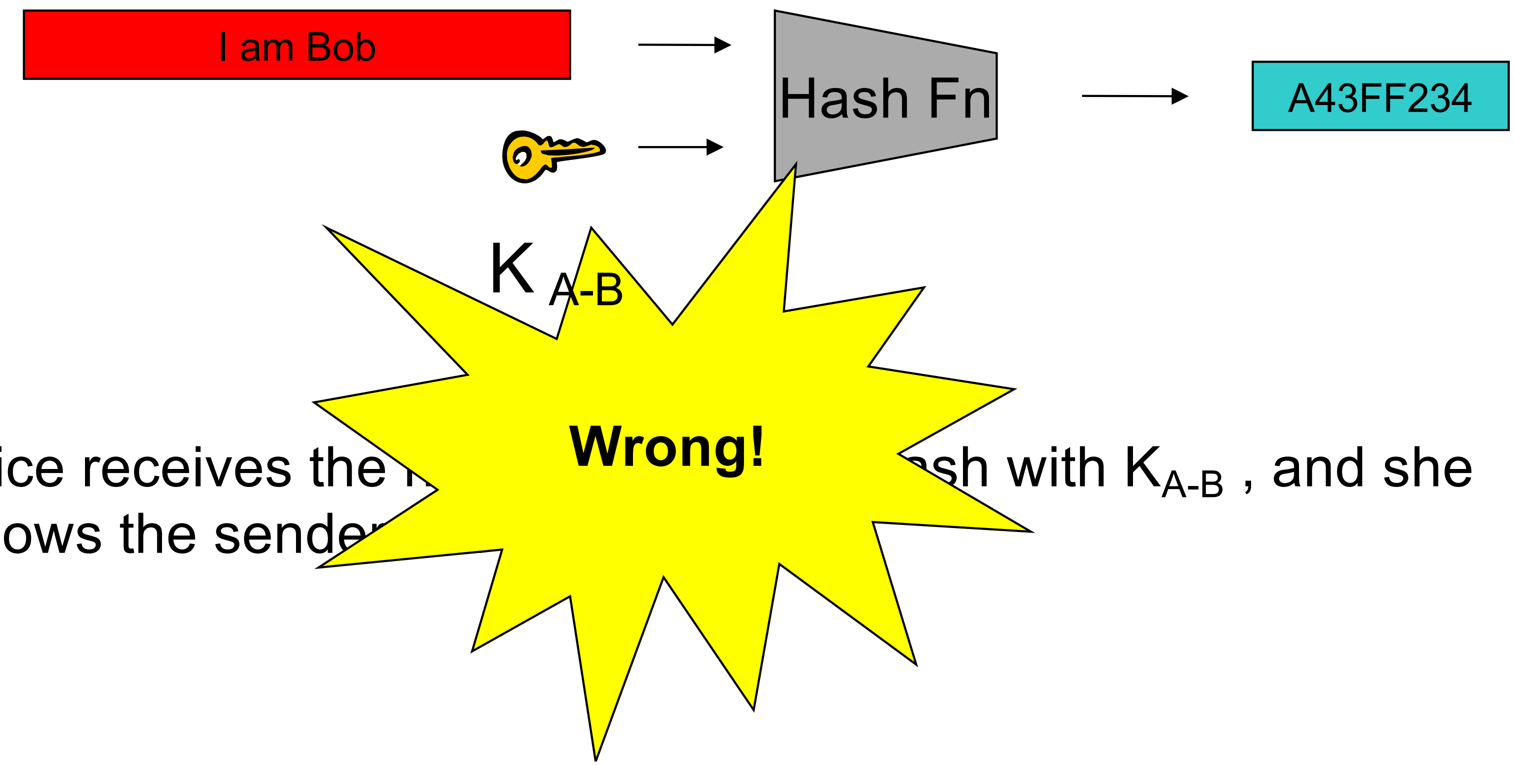
Bob computes MAC with message and $K_{A\text{-}B}$ to verify.

Why is this secure from a message integrity perspective?
How do properties of a hash function help us?

Symmetric Key: Authentication

- You already know how to do this!

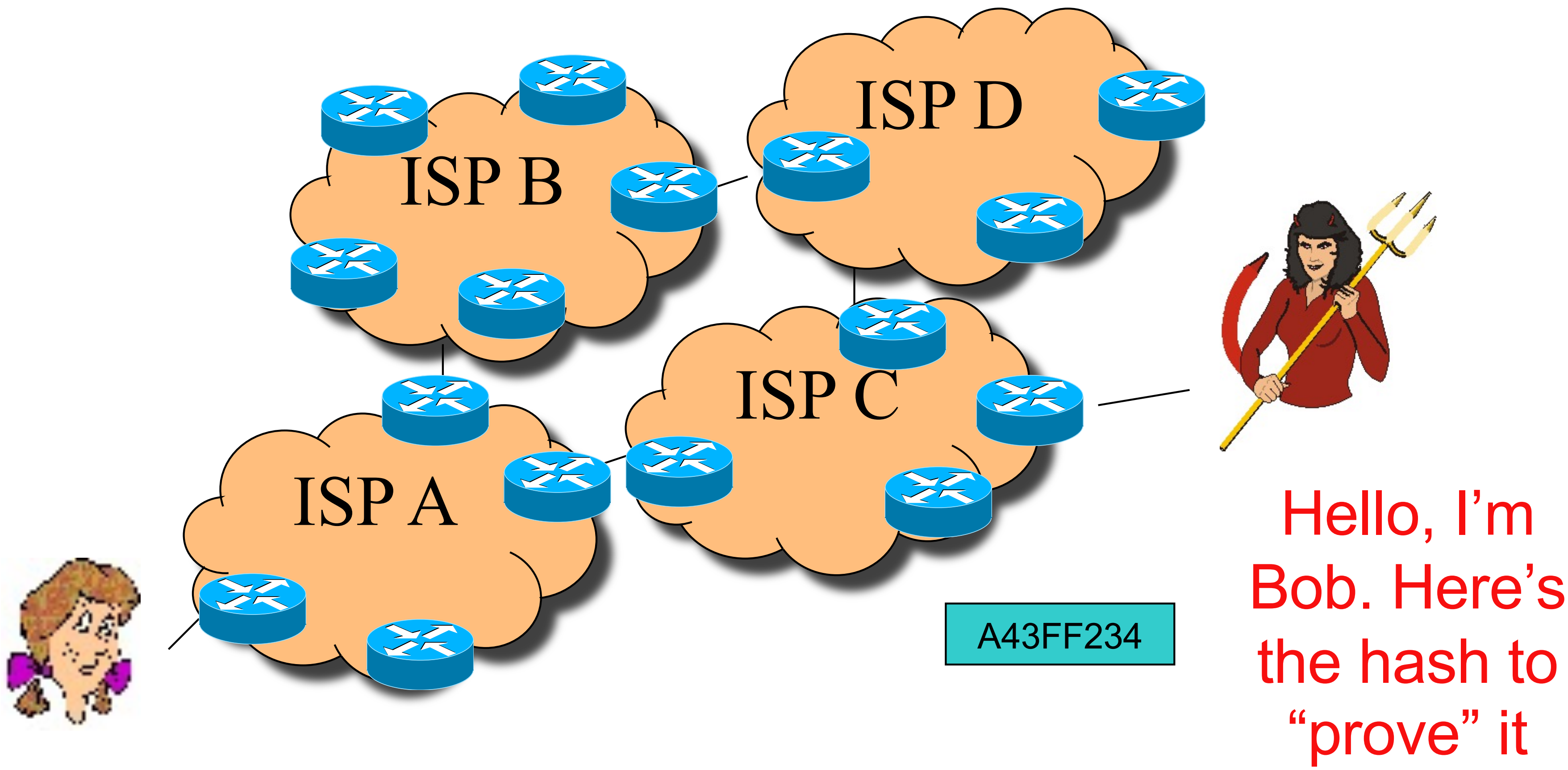
  (hint: think about how we showed integrity)



Alice receives the ... sh with $K_{A\text{-}B}$, and she knows the sende...

Symmetric Key: Authentication

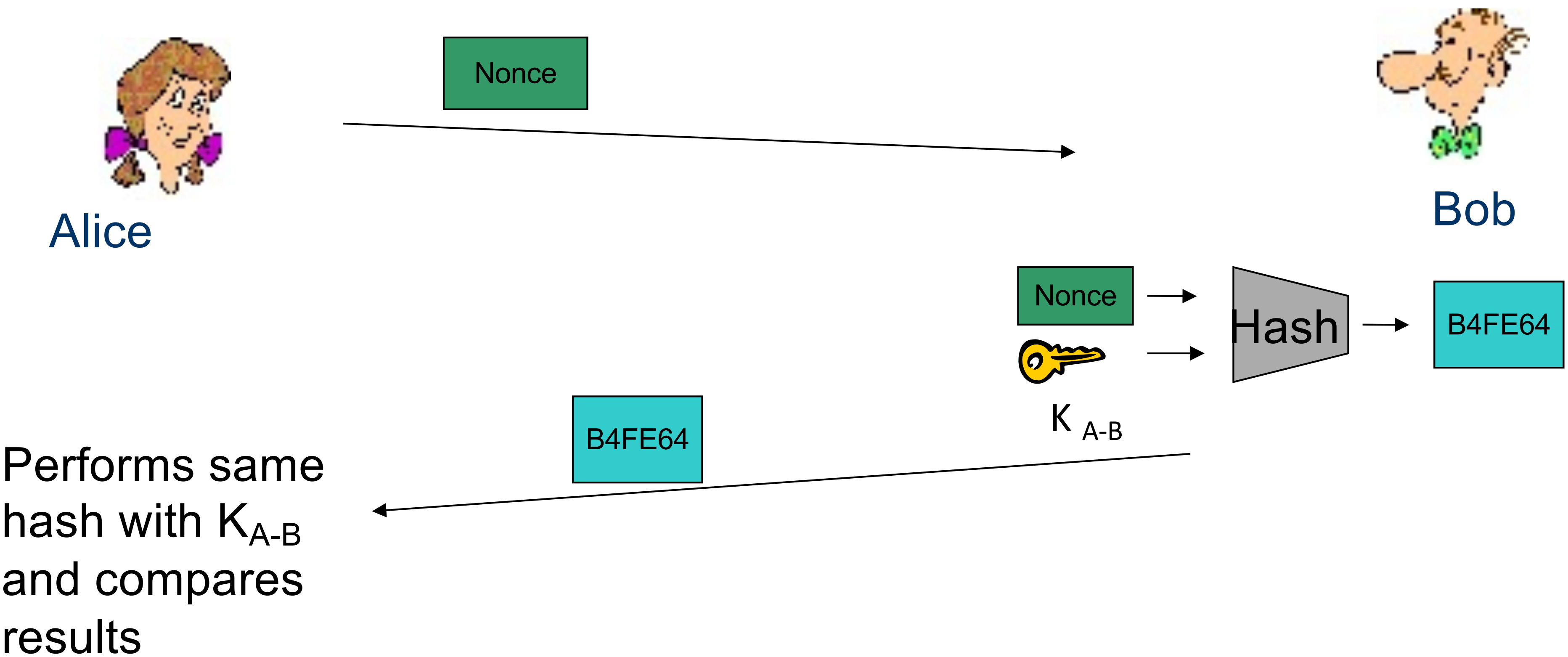
What if Mallory overhears the hash sent by Bob, and then "replays" it later?



Huh, how can we solve this?

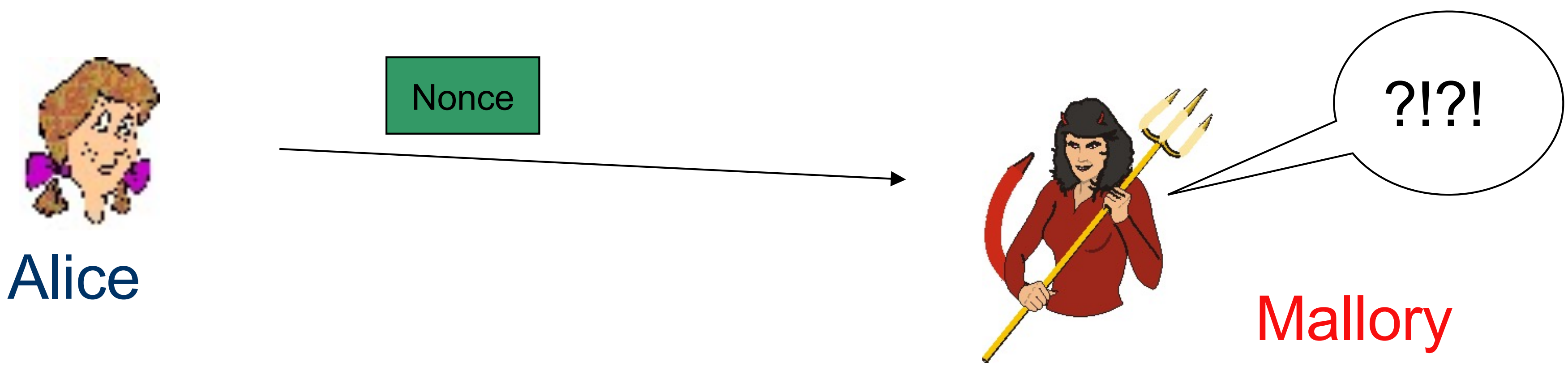
Symmetric Key: Authentication

- A “Nonce”
  - A random bitstring used only once. Alice sends nonce to Bob as a “challenge”. Bob Replies with “fresh” MAC result.



Symmetric Key: Authentication

- A “Nonce”
  - A random bitstring used only once. Alice sends nonce to Bob as a “challenge”. Bob Replies with “fresh” MAC result.



If Alice sends Mallory a nonce, she cannot compute the corresponding MAC without $K_{A\text{-}B}$

Symmetric Key Crypto Review

- Confidentiality: Stream & Block Ciphers
- Integrity: HMAC
- Authentication: HMAC and Nonce

Questions??

Are we done? Not Really:

1) Number of keys scales as $O(n^2)$

2) How to securely share keys in the first place?

Asymmetric Key Crypto:

- Instead of shared keys, each person has a "key pair"

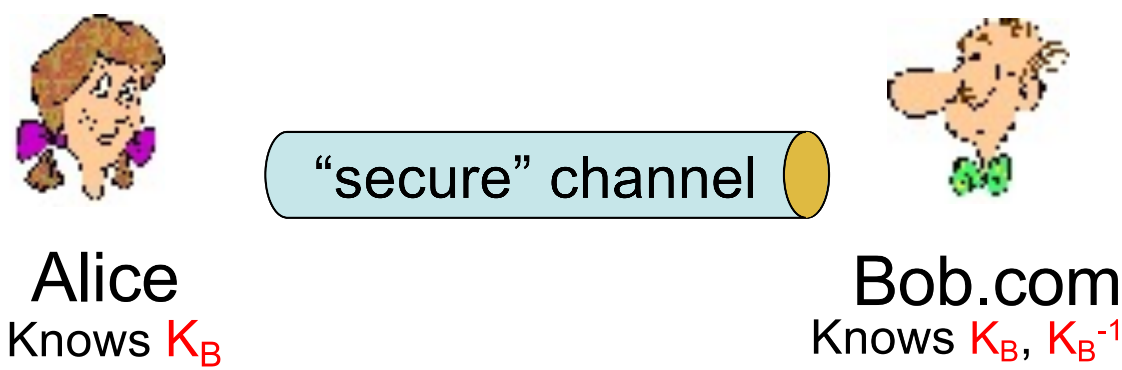
K_B Bob's <u>public</u> key

K_B^{-1} Bob's <u>private</u> key

- The keys are inverses, so: $K_B^{-1}(K_B(m)) = m$

Asymmetric/Public Key Crypto:

- Given a key *k* and a message *m*
  - Two functions: Encryption (E), decryption (D)
  - ciphertext $c = E(K_B, m)$
  - plaintext $m = D(K_B^{-1}, c)$
  - Encryption and decryption use *different* keys!



But how does Alice know that K_B means "Bob"?

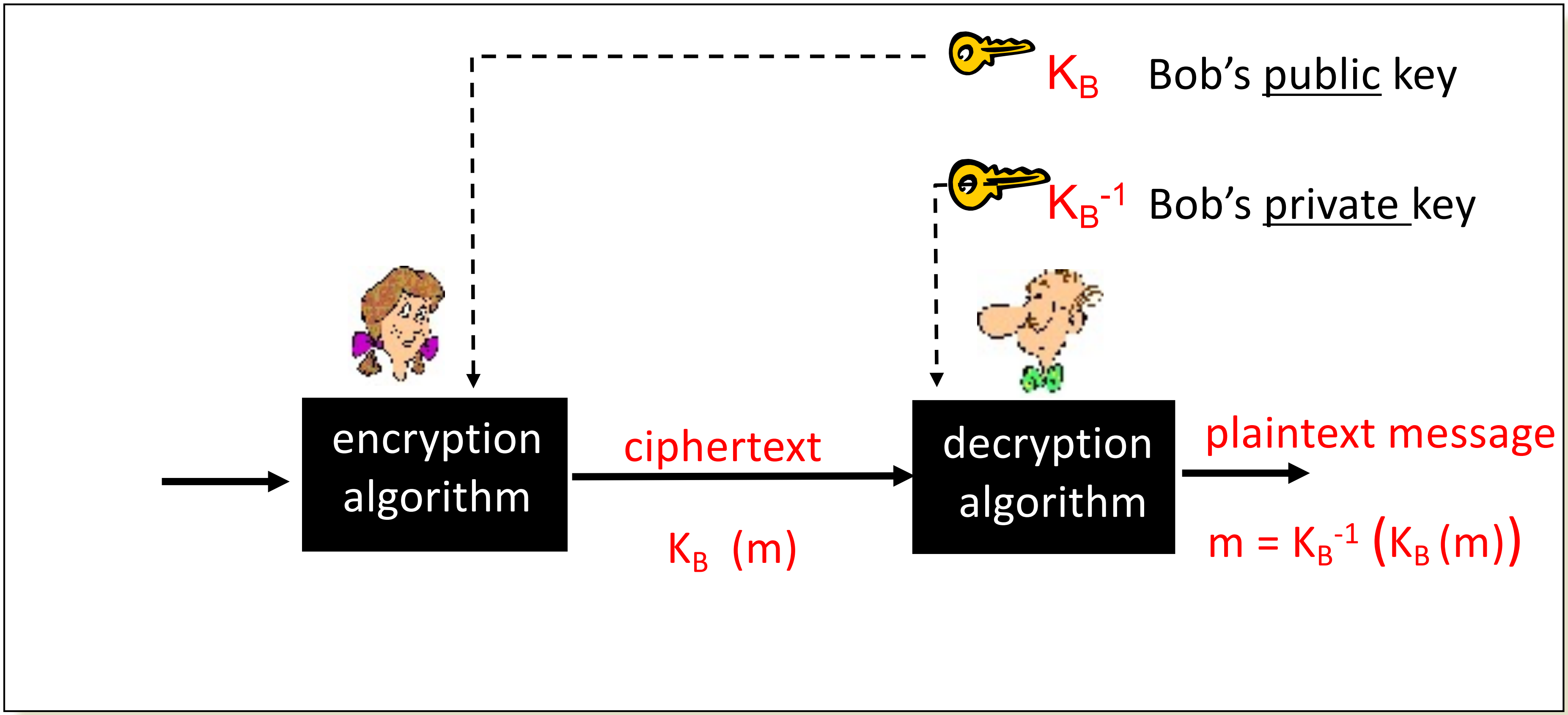
Asymmetric Key Crypto:

- It is believed to be computationally unfeasible to derive K_B^{-1} from K_B or to find any way to get M from $K_B(M)$ other than using K_B^{-1} .

➔K_B can safely be made public.

Note: We will not detail the computation that $K_B(m)$ entails, but rather treat these functions as black boxes with the desired properties. (more details in the book).

Asymmetric Key: Confidentiality



Asymmetric Key: Sign & Verify

- If we are given a message M, and a value S such that $K_B(S) = M$, what can we conclude?
  - The message must be from Bob, because it must be the case that $S = K_B^{-1}(M)$, and only Bob has K_B^{-1} !
- This gives us two primitives:
  - Sign (M) = $K_B^{-1}(M)$ = Signature S
  - Verify (S, M) = test($K_B(S)$ == M)

Asymmetric Key: Integrity & Authentication

- We can use Sign() and Verify() in a similar manner as our HMAC in symmetric schemes.

Integrity:

S = Sign(M)	Message M

Receiver must only check Verify(M, S)

Authentication:

Nonce → S = Sign(Nonce)

Verify(Nonce, S) ←

Asymmetric Key Review:

- Confidentiality: Encrypt with Public Key of Receiver
- Integrity: Sign message with private key of the sender
- Authentication: Entity being authenticated signs a nonce with private key, signature is then verified with the public key

But, these operations are computationally expensive*

The Great Divide

	Symmetric Crypto: (Private key) Example: AES	Asymmetric Crypto: (Public key) Example: RSA
Requires a pre-shared secret between communicating parties?	Yes	No
Overall speed of cryptographic operations	Fast	Slow

One last "little detail"…

How do I get these keys in the first place??

Remember:

- Symmetric key primitives assumed Alice and Bob had already shared a key.
- Asymmetric key primitives assumed Alice knew Bob's public key.

This may work with friends, but when was the last time you saw Amazon.com walking down the street?

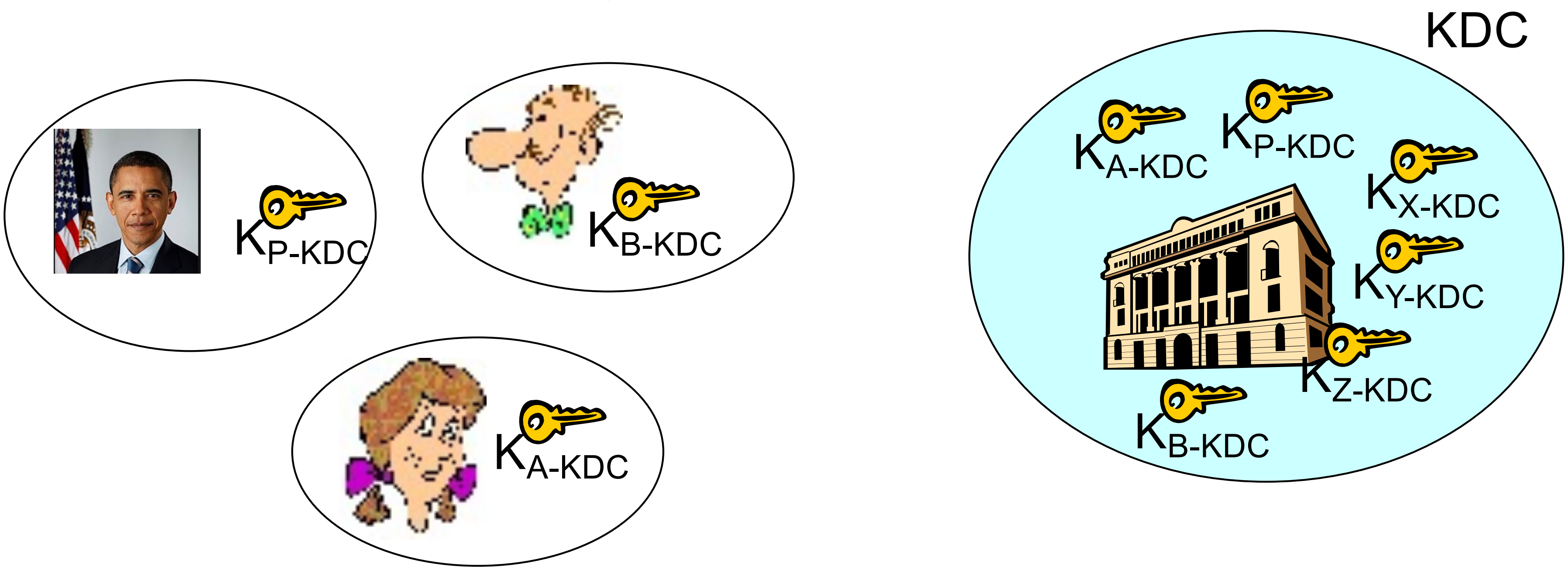
Symmetric Key Distribution

- How does Andrew do this?

Andrew Uses Kerberos, which relies on a Key Distribution Center (KDC) to establish shared symmetric keys.

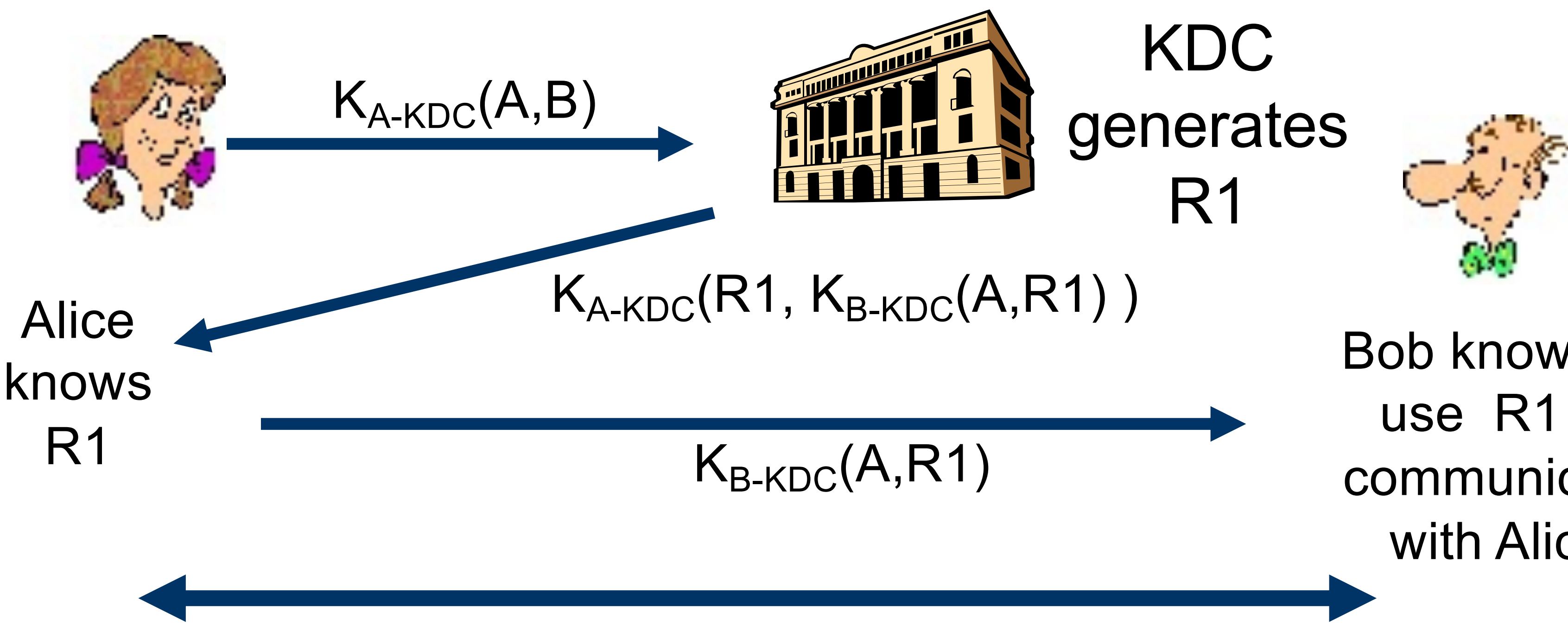
Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

How Useful is a KDC?

- Must always be online to support secure communication
- KDC can expose our session keys to others!
- Centralized trust and point of failure.

In practice, the KDC model is mostly used within single organizations (e.g. Kerberos) but not more widely.

Today's Lecture

- Internet security weaknesses
- Crypto 101
- Key distribution (Kerberos)