15-441 15-641 Computer Networking

Lecture 20 – Security:
Threats and Crypto-based Solutions
Peter Steenkiste

Fall 2016

www.cs.cmu.edu/~prs/15-441-F16

With slides from: Debabrata Dash, Nick Feamster, Vyas Sekar, and others

Normal Mindset

- No user would do that
- The odds of a router being misconfigured that way is too small to worry about

Security Mindset

- The **adversary** will do anything it can to break your system
- It will study your system and purposefully do the worse thing it can
- Might even disregard its own well being
- Will attack your implementation and your assumptions
- Very different mindset – adversaries may not be rational!

Adversaries

- Possible adversaries include:
  - Competitors trying harm you
  - Governments trying to control you
  - Criminals who want to use your system
  - Disgruntled employees (the *insider threat*)
  - Hackers who find it fun to break stuff
  - Others we didn't even think of
- Assumptions about the adversary
- Security is very hard

Unlimited resources

Knows your source code

Destructive with no "real" goals

Internet design goals

1. Interconnection
2. Failure resilience
3. Multiple types of service
4. Variety of networks
5. Management of resources
6. Cost-effective
7. Low entry-cost
8. Accountability for resources

Where is security?

David D. Clark, "The Design Philosophy of the DARPA Internet Protocols", *Computer Communications Review* 18:4, August 1988, pp. 106–114

Why did they leave it out?

- Designed for connectivity
- Network designed with implicit trust
  - Started as a small and cooperative network
  - No "bad" guys (adversaries)
- Can't security be provided at the edge?
  - Encryption, Authentication etc
  - End-to-end arguments in system design

Saltzer, J. H., D. P. Reed, and D. D. Clark (1981) "End-to-End Arguments in System Design". In: Proceedings of the Second International Conference on Distributed Computing Systems. Paris, France. April 8–10, 1981. IEEE Computer Society, pp. 509-512

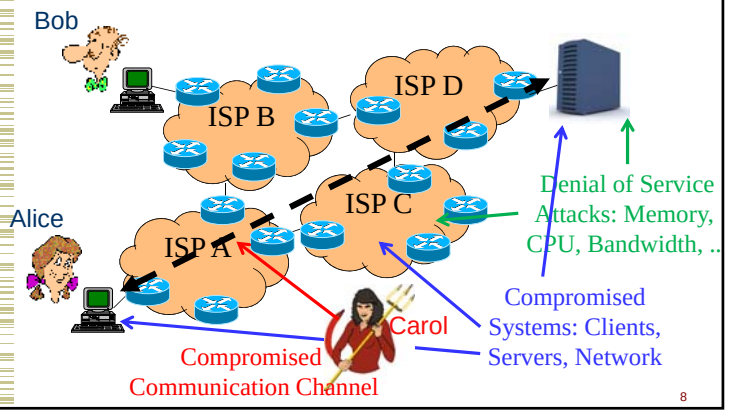
Internet Design and Usage versus Security

- Global Addressing
  (=> every sociopath is your next-door neighbor*)
- Connection-less datagram service
  (=> can't verify source, hard to protect bandwidth)
- Anyone can connect
  (=> ANYONE can connect)
- Millions of hosts run nearly identical software
  (=> single exploit can create epidemic)
- Most Internet users know nothing about the Internet
  (=> and you expect them to secure their system?)

* Dan Geer

7

What Can Wrong? A Lot!



8

Our “Narrow” Focus

- Yes:
  - Creating a “secure channel” for end-to-end communication (Today)
  - Protecting network resources and limiting connectivity (Next Lecture)
    - Accountability for resources (largely not end-to-end)
- No:
  - Preventing software vulnerabilities & malware, or “social engineering”.

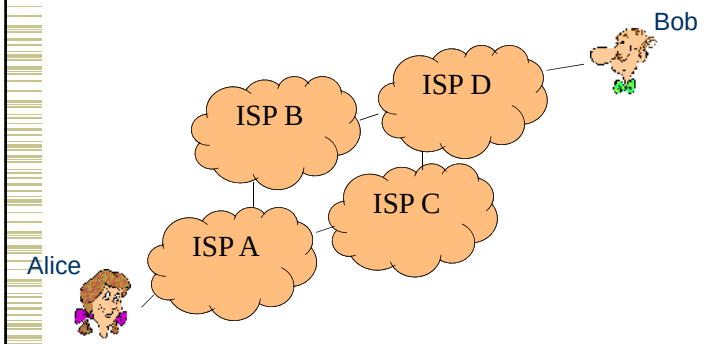
9

Outline – Creating a Secure Channel

- Security threats
- Cryptography overview
- Securing channels
- Key management
- TOR

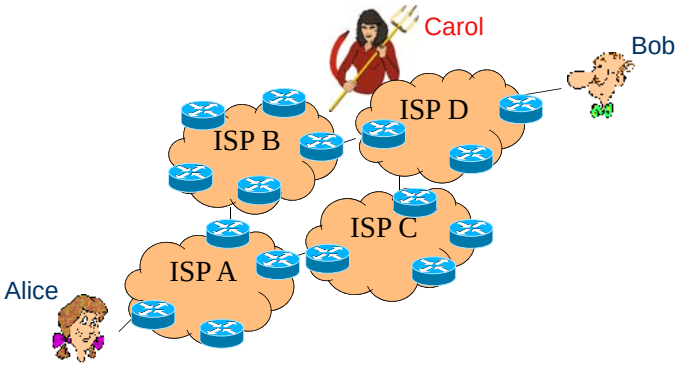
10

Secure Communication with an Untrusted Infrastructure



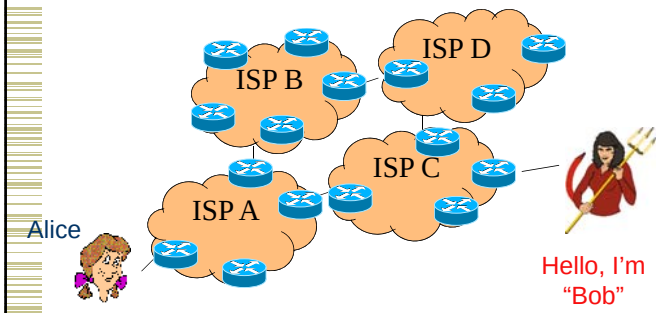
11

Secure Communication with an Untrusted Infrastructure



12

Secure Communication with an Untrusted Infrastructure



13

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?)
- Non-repudiation (Proof of communication)

14

What is cryptography?

"cryptography is about communication in the presence of adversaries."

- Ron Rivest

"cryptography is using math and other crazy tricks to approximate magic"

- Unknown 441 TA

15

What is cryptography?

Tools to help us build secure communication channels that provide:

1) Authentication
2) Integrity
3) Confidentiality

16

Cryptography As a Tool

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

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

Take a security course if you think you may want to use it (correctly) in the future

17

The Great Divide

	Symmetric Crypto (Private key) (E.g., AES)	Asymmetric Crypto (Public key) (E.g., RSA)
Shared secret between parties?	Yes	No
Speed of crypto operations	Fast	Slow

18

Symmetric Key Cryptography: Confidentiality

Motivating Example:

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

Scheme:

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

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

2) Can an adversary recover the message M?

3) Can adversary recover the key K?

19

Symmetric Key: Example

- One-time Pad (OTP) is proven "information-theoretically secure" (Claude Shannon, 1949)
  - No information provided about the message other than its length
- Impressive?
- Assumptions:
  - Perfectly random one-time pads
  - One-time pad at least the length of the message
  - Never can reuse a one-time pad
  - Adversary can never know the one-time pad

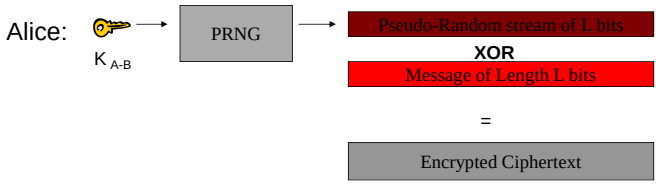
20

Symmetric Key: Reality

- All ciphers suffer from assumptions, but one-time pad's are impractical to maintain
  - Key is as long at the message
  - Keys cannot be reused
- In practice, two types of ciphers are used that require constant length keys:
  - **Stream Ciphers**
    Ex: RC4, A5
  - **Block Ciphers**
    Ex: DES, AES, Blowfish

Symmetric Key: Stream Ciphers

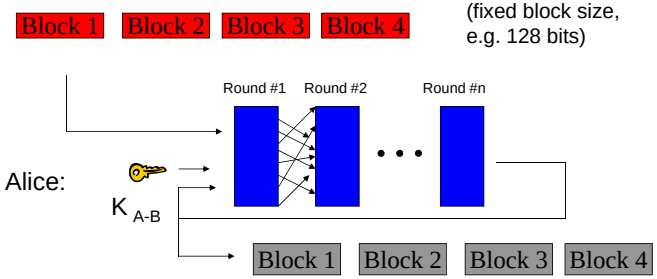
- Example: RC4



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

Symmetric Key: Block Ciphers

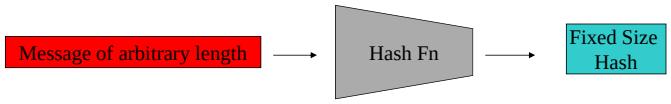
- Example: AES



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

Cryptographic Hash Functions

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

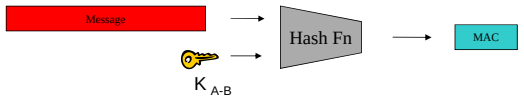


Symmetric Key: Integrity

- Hash Message Authentication Code (HMAC)

Step #1:

Alice creates MAC



Step #2

Alice Transmits Message & MAC

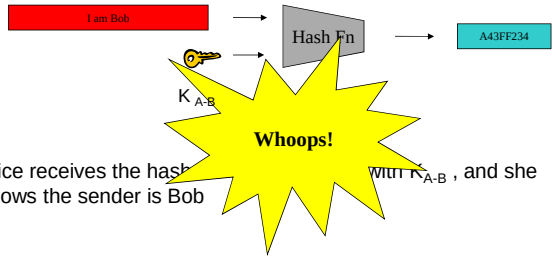
Step #3

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

Why is this secure?
How do properties of a hash function help us?

Symmetric Key: Authentication

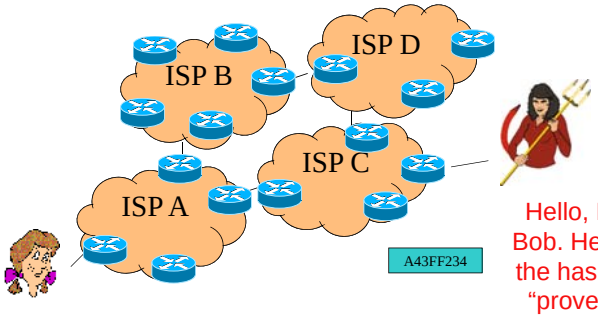
- You already know how to do this!
  (hint: think about how we showed integrity)



Alice receives the hash with K_{A-B}, and she knows the sender is Bob

Symmetric Key: Authentication

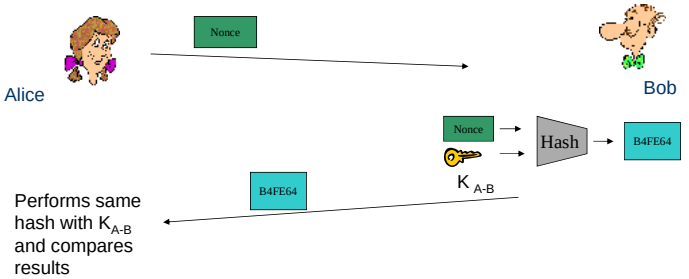
What if Mallory overhears the hash sent by Bob, and then "replays" it later?



Hello, I'm Bob. Here's the hash to "prove" it

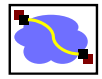
Symmetric Key: Authentication

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



Performs same hash with K_{A-B} and compares results

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.

Alice

Nonce

?!?!

Mallory

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

Symmetric Key Crypto Review

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

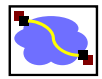
Questions??

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 public key

K_B^{-1} Bob's private key

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

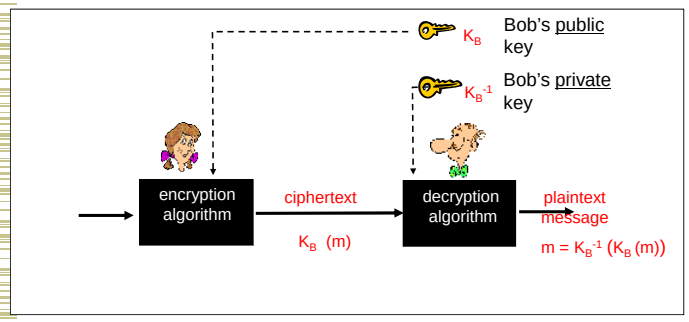
Asymmetric Key Crypto:

- 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 explain the computation that $K_B(m)$ entails, but rather treat these functions as black boxes with the desired properties.

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
  - $Verify(S, M) = test(K_B(S) == M)$

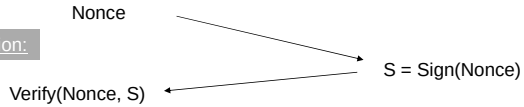
Asymmetric Key: Integrity & Authentication

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

Integrity:

Receiver must only check Verify(M, S)

Authentication:



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*

Outline – Creating a Secure Channel

- Security threats
- Cryptography overview
- Securing channels
- Key management
- TOR

37

Resources

- Textbook: 8.1 – 8.3
- Wikipedia for overview of Symmetric/Asymmetric primitives and Hash functions.
- OpenSSL (www.openssl.org): top-rate open source code for SSL and primitive functions.
- “Handbook of Applied Cryptography” available free online: www.cacr.math.uwaterloo.ca/hac/

38