# CSE/EE 461
# Network Security

Chapter 8

# What do we mean by "Network Security"?

- Networks are fundamentally shared
  - Sharing a resource is "safe" if everybody behaves well.
  - It becomes unsafe if people badly
- Three ways to behave badly
  - Eavesdrop
  - Forge
  - Transform
- Leads to three desirable security properties
  - Privacy: messages can't be eavesdropped
  - Authenticity: messages were sent by the right party
  - Integrity: messages can't be tampered with
    - A & I are two sides of the same coin

# Network Security vs. Security

- Network Security
  - Messages are what they say they are
  - Messages say what they are only to the right parties
- Security
  - The system behaves as specified
  - Can have N.S. without S.
  - Example:
    - Secure log in lets me log in.
    - A setuid root program lets me crash the system
  - Not a problem with network security

# Why is security hard to achieve?

- It is an ill-defined goal
- It is hard to express goal
  - you can do X, but you can't do Y
  - What are X and Y?
- It's a negative goal
  - requires that you know there are no vulnerabilities
  - like proving there are no bugs
- It's a valuable goal to subvert

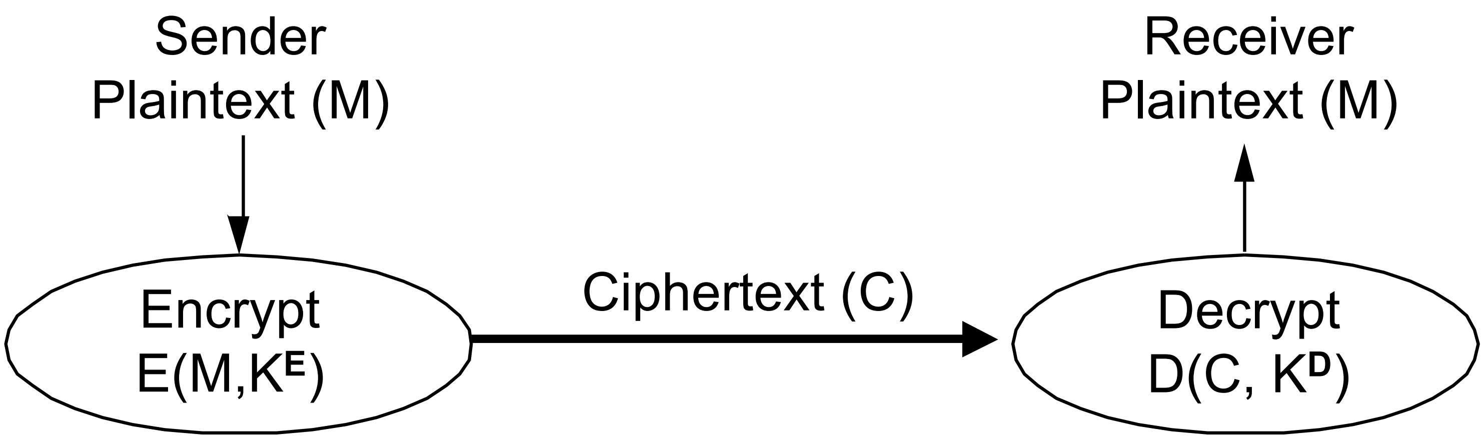
# Approaches at 10,000 ft

- **Locks**
  - Physical security
    - Tackle the problem of sharing directly
  - "Security through obscurity"
    - Hope no-one will find out what you're doing!
  - Throw math at the problem
    - Cryptography
- **Alarms**
  - Watch for the bad guys
    - Beware the false positives/negatives
- **Fingerprints**
  - Audit trails
    - Tracebacks
  - Hard not to get lost in a sea of data

# Achieving Network Security

- Use something secret to scramble the data
- Put the scrambled data on the wire
- Use something related to the secret to unscramble the data
- Two kinds of secrets
  - Symmetric
    - Sender/receiver share the same secret
  - Assymetric
    - Sender's and receiver's secret related computationally, but intractable to discover one from the other

# Use Encryption for Privacy

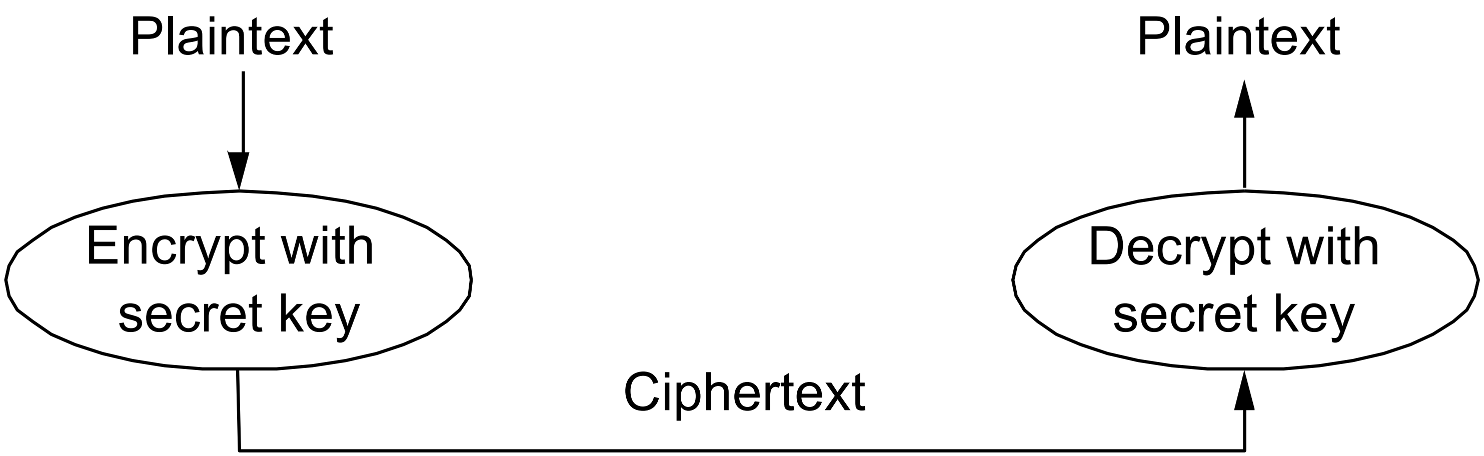



- *Cryptographer* chooses functions E, D and keys $K^E$, $K^D$
- *Cryptanalyst* tries to "break" the system
  - Depends on what is known: E and D, M and C?

# Two Basic Encryption Strategies

- Symmetric: Secret Key
  - Bob and Alice each share a secret (K)
  - The secret is used to encrypt communication between Bob and Alice.
    - D(E(M,K),K) = M
  - DES
- Assymetric: Public Key (RSA)
  - Bob has a secret key (K) and a matching public key (K')
    - D(E(M, K'), K) = M
    - D(E(M, K), K') = M

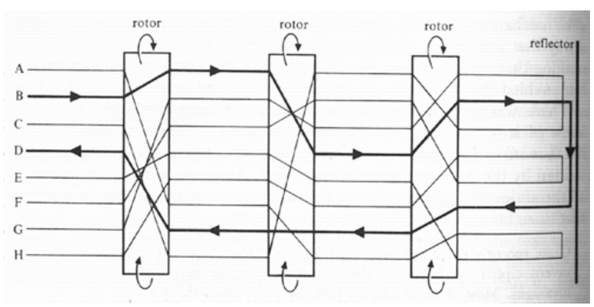
# Secret Key Functions (DES)

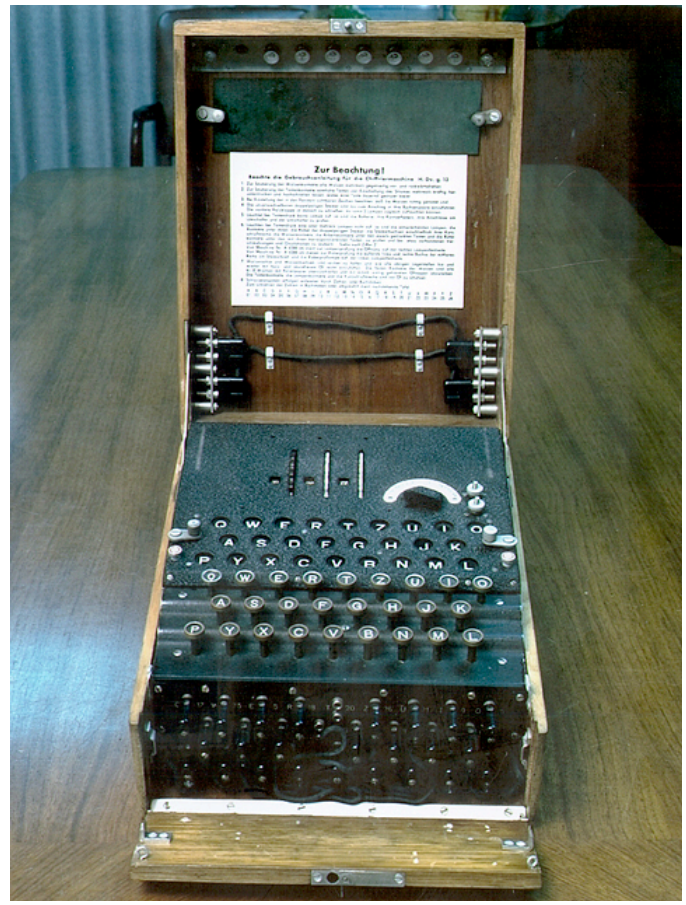



- Single key (symmetric) is shared between parties
  - Often chosen randomly, but must be communicated
  - Turns out to be a hard problem (key distribution)

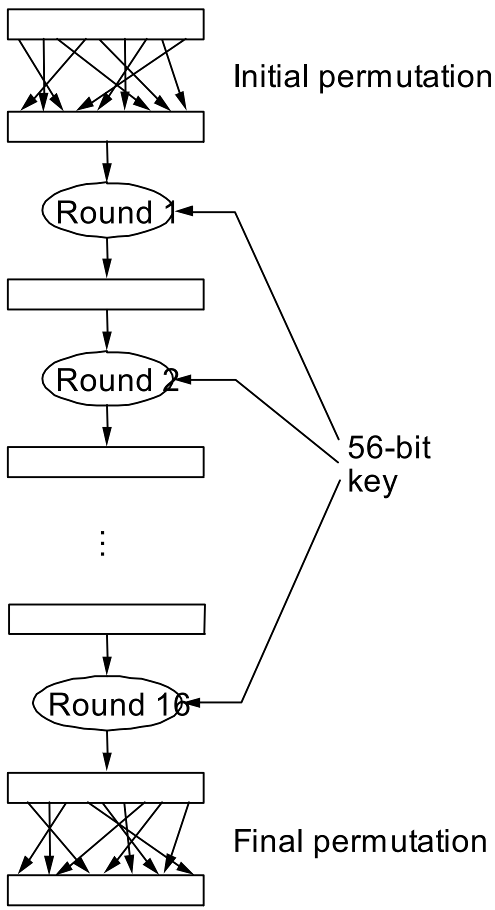
# DES Is a Bit Scrambler

- The bits go in one way and come out another according to some scrambling rules
- Unscrambling runs it backwards
  - Not unlike the WW2 German Cyphers (ENIGMA)

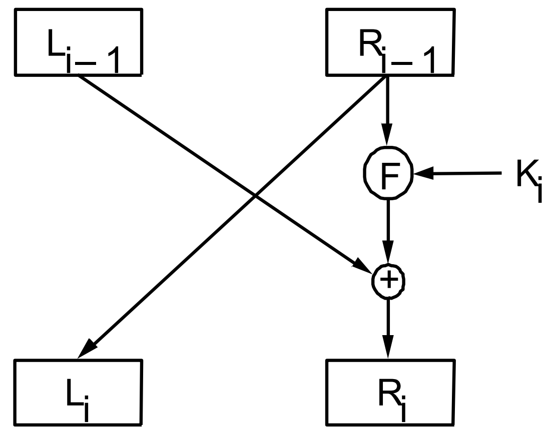


# DES as a Digital Scrambler




Each Round:




DES uses a 64 bit key (56 + 8)
Message encrypted 64 bits at a time
16 rounds in the encryption
Each round scrambles 64 bits

# On the Hardness of DES

- Exhaustive search is the only known attack
  - Not much is known about the unknown attacks
- Size of key space determines cost of attack
  - Key space needs to track Moore's law just to stay even (future proof keys)
    - a key that's just barely long enough today won't be long enough in a few years
      - today's 52 bit DES key is "equivalent" to a 40 bit key from 20 years ago
  - Easy to parallelize

exhaustive key searches .

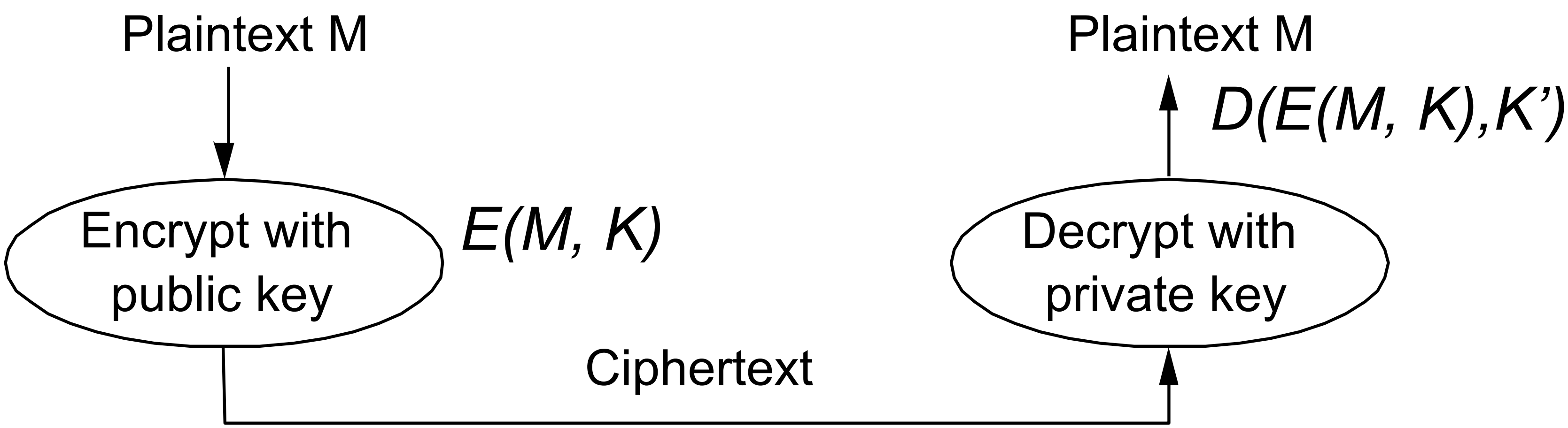
| Key Size (bits) | Number of Alternative Keys | Time Required at 1 Decryption / µs | Time Required at $10^6$ Decryptions / µs |
|---|---|---|---|
| 32 | $2^{32} = 4.3 \times 10^9$ | $2^{31}$ µs = 35.8 minutes | 2.15 milliseconds |
| 56 | $2^{56} = 7.2 \times 10^{16}$ | $2^{55}$ µs = 1142 years | 10 hours |
| 128 | $2^{128} = 3.4 \times 10^{38}$ | $2^{127}$ µs = $5.4 \times 10^{24}$ years | $5.4 \times 10^{18}$ years |
| 168 | $2^{168} = 3.7 \times 10^{50}$ | $2^{167}$ µs = $5.9 \times 10^{36}$ years | $5.9 \times 10^{30}$ years |

IDEA
DES3/3

# But more fundamentally

- Secret key systems are vulnerable because it's hard to keep a secret.
  - **you've got to tell somebody your secret to use it.**
    - **There's no protection from blabbermouths.**
  - Also, key needs to be kept somewhere in order to use it.
    - user can type it in
      - but the keys won't be very long
    - keep it in a file?
      - that won't work unless the file is encrypted
    - keep it on a removable device
      - smartcard, PCMCIA
- Needed is a strategy that doesn't require me to tell you my secret and expect it to remain a secret forever.
- Assymetric

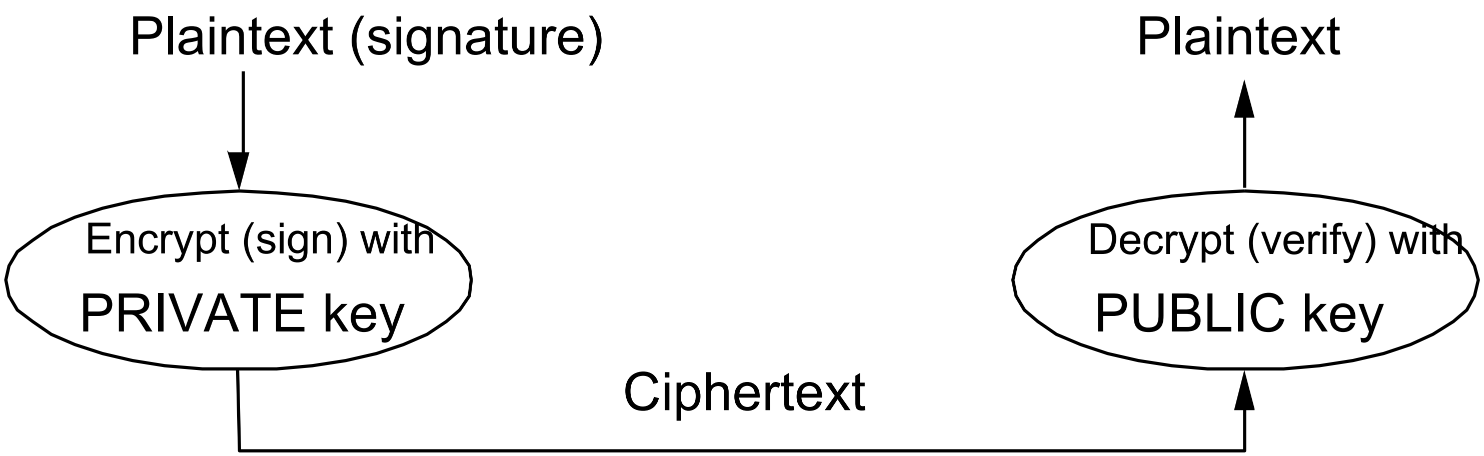
# Public Key Functions (RSA)




**E(M, K)**

- Only holder of appropriate private key can read message
- Public and private key related mathematically
  - Public key can be published; private is a secret
  - Very Hard to deduce private key from public key
    - Equivalent to factoring very large numbers
      - For details, take a crypto class

# Authenticates as well as Encrypts



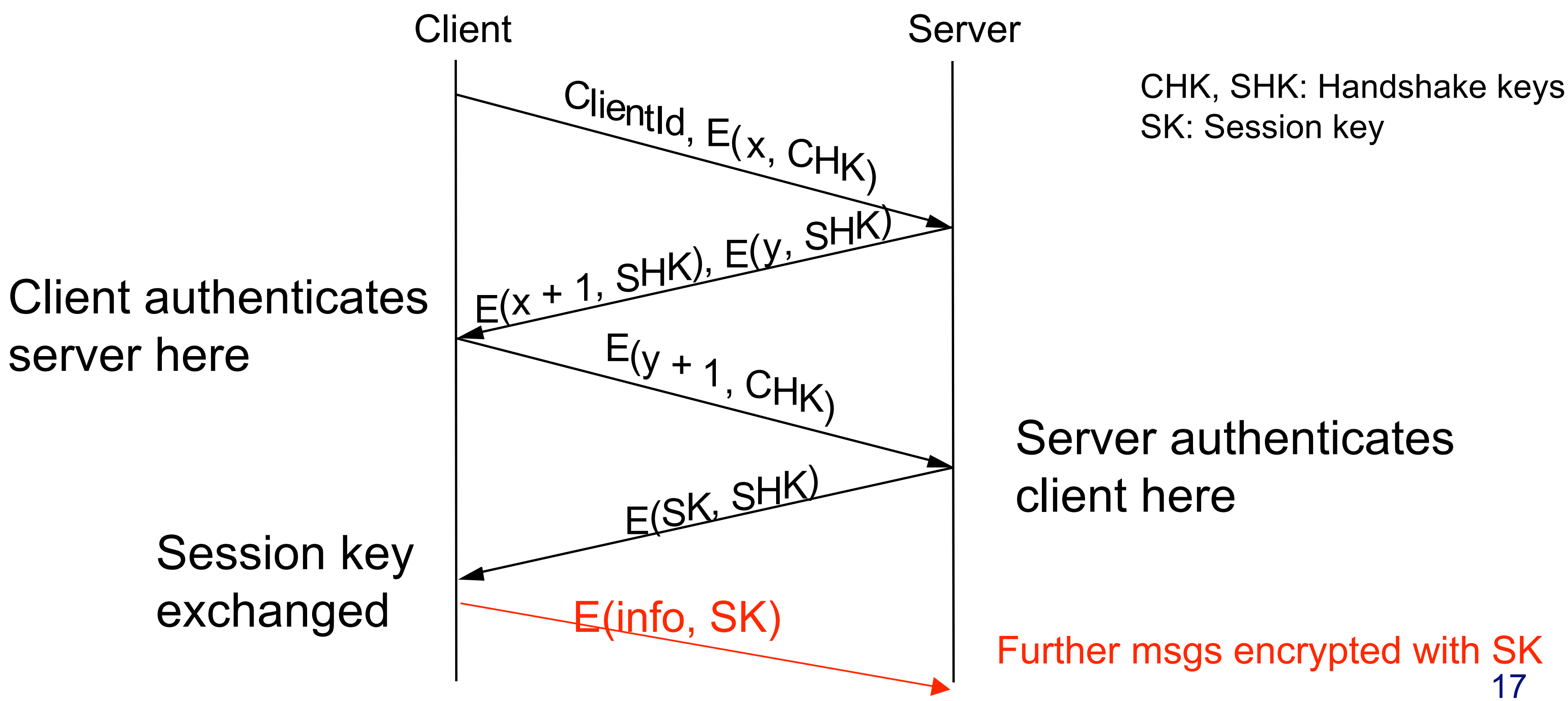

- Digital Signature

# Algorithms vs. Protocols

- Algorithms let you encode the bits
- Protocols tell you how to decide if the bits are valid
- Looks pretty easy
- But in practice, it's pretty hard
  - what assumptions do we make?
- Most failures come from attacks on the protocol and not the algorithm

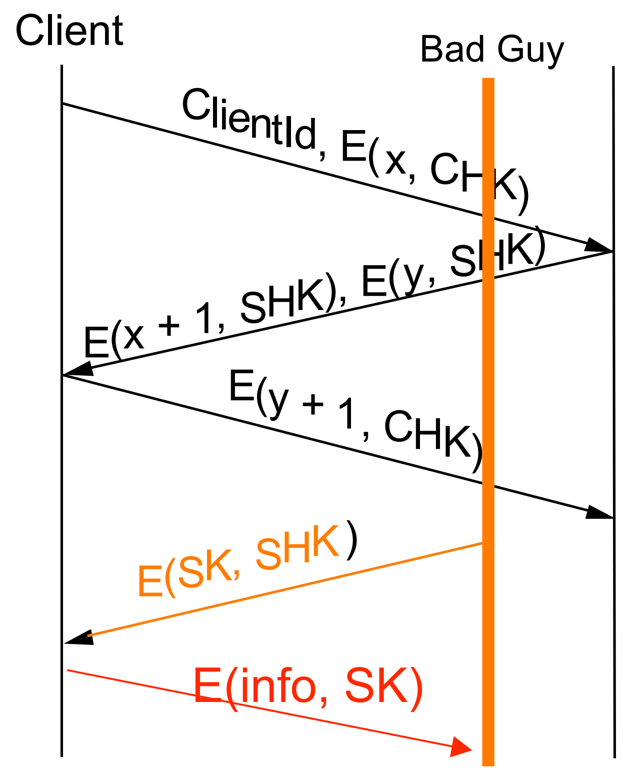
# Example Authentication Protocol

- Three-way handshake for mutual authentication
  - Client and server share secrets, e.g., login password.
  - Use to construct *session key*



# Simple Assumptions Can Lead to Weaknesses

- No protection against replay
  - Client assumes that it is receiving a *fresh* key
- This doesn't mean the protocol is broken, only that it makes certain assumptions.




Bad Guy gets the "info"

# Why did this weakness slip by?

- Eyeball verification is not very effective
- Assumptions are often not explicit
  - eg, the key is fresh
- An attacker will leverage these assumptions to break the protocol
- What's needed is a way to reason about authentication

# A Logic of Authentication

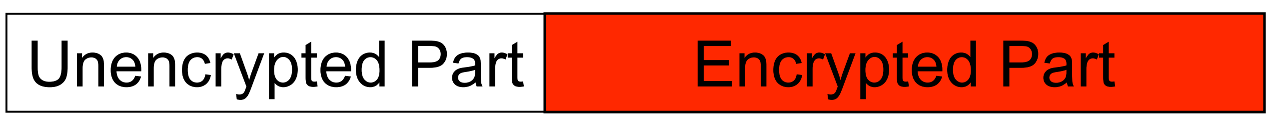
- Seminal paper published in 1991 SOSP by Burrows, Abadi and Needham
  - BAN Logic
- Simple idea
  - make explicit assumptions in an authentication protocol
  - describe protocol by formal algebra
    - make explicit initial states
    - derive belief relationships through state transitions
    - final state tells us what we can know

# Example Questions

- What does this protocol achieve?
- Does this protocol need more assumptions than another one?
- Does this protocol do anything unnecessary that could be left out without weakening it?
- Does this protocol encrypt something that could be sent in the clear
- See BAN paper if you want to know more

# Cryptography in Protocols

- These techniques can be applied at different levels:
  - NETWORK: IP packets (IPSEC)
  - TRANSPORT: STCP
  - SESSION: SSLTLS, Secure HTTP
  - APPLICATION: Email (PGP)



# A Faster "RSA Signature"

- Suppose you have a very big "message" that you want to authenticate
  - Encryption can be expensive, e.g., RSA ~1Mbps
- To speed up, let's sign just the checksum instead!
  - Check that the encrypted bit is a signature of the checksum
  - Problem: Easy to alter data without altering checksum
- Answer: Cryptographically strong "checksums" called message digests
  - computationally difficult to choose data with a given checksum
  - But they still run much more quickly than encryption
  - MD5 (128 bits) is the most common example

# Message Digests (MD5, SHA)

- Act as a cryptographic checksum or hash
  - Typically small compared to message (MD5 128 bits)
  - "One-way": infeasible to find two messages with same digest