Cyber-Physical Systems

Security

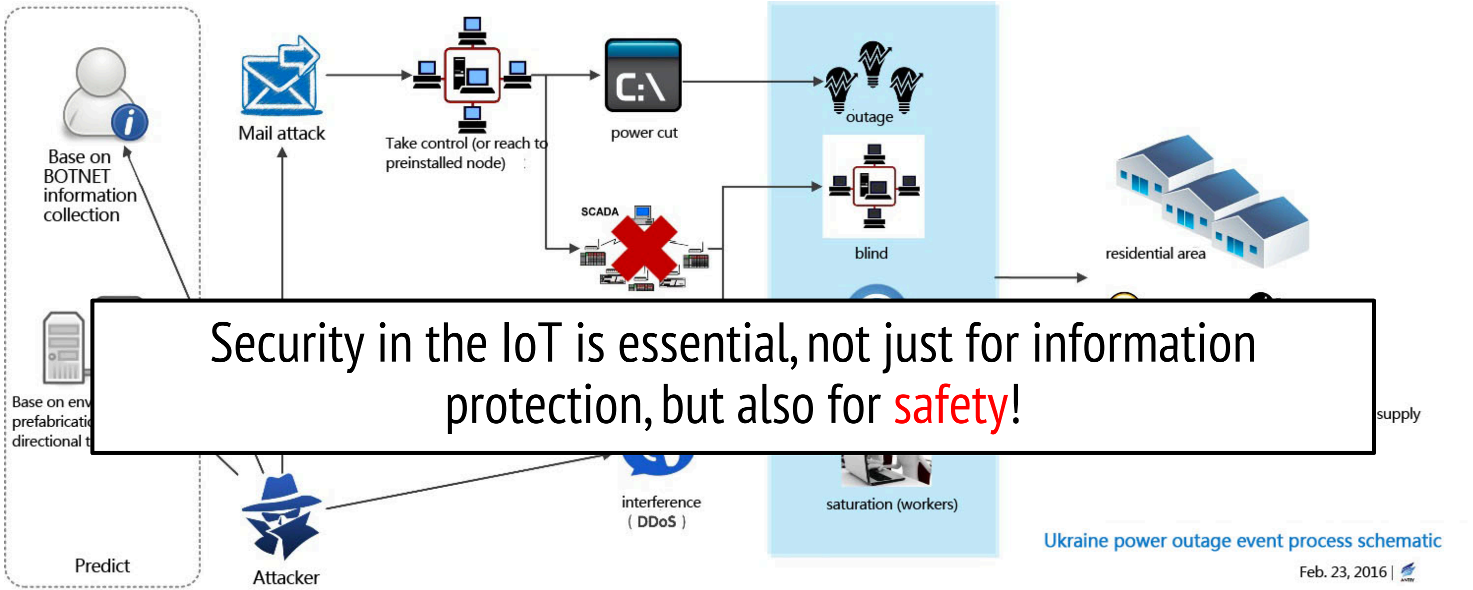
IECE 553/453– Fall 2020

Prof. Dola Saha

UNIVERSITY AT ALBANY
State University of New York

Security Threats in the IoT

- Cyber attack on the Ukrainian power grid
- Power outage caused by hackers



Security in the IoT is essential, not just for information protection, but also for safety!

Source: Comprehensive Analysis Report on Ukraine Power System Attacks
March 16, 2016 By Antiy Lab

IoT as a Huge Security Risk

828416577

The New York Times

© 2016 The New York Times Company NEW YORK, SATURDAY, OCTOBER 22, 2016

New Weapons Used in Attack On the Internet

By NICOLE PERLROTH

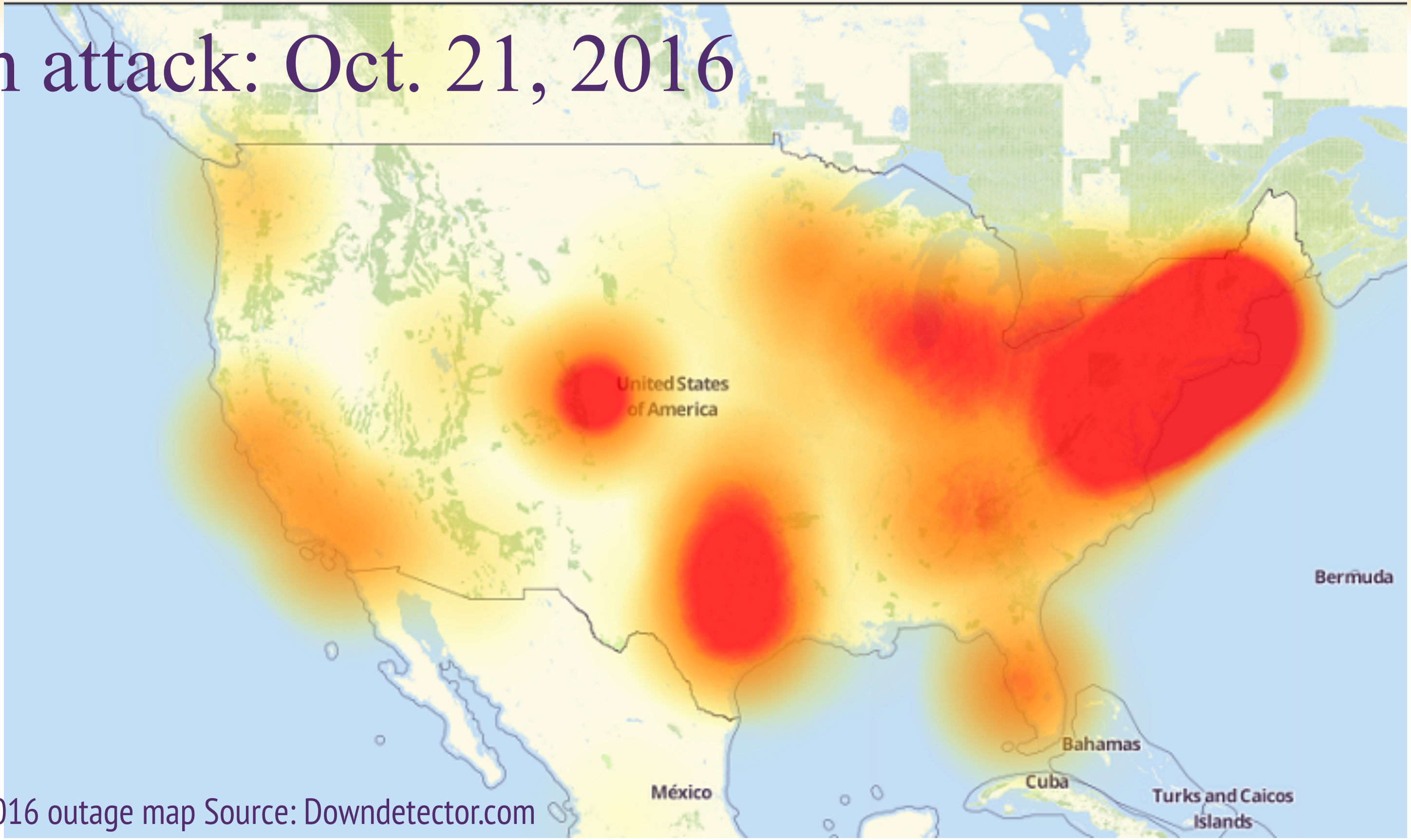
SAN FRANCISCO — Major websites were inaccessible to people across wide swaths of the United States on Friday after a company that manages crucial parts of the internet's infrastructure said it was under attack.

Users reported sporadic problems reaching several websites, including Twitter, Netflix, Spotify, Airbnb, Reddit, Etsy, SoundCloud and The New York Times.

The company, Dyn, whose servers monitor and reroute internet traffic, said it began experiencing what security experts called a distributed denial-of-service attack just after 7 a.m. Reports that many sites were inaccessible started on the East Coast, but spread westward in three waves as the day wore on and into the evening.

IoT vulnerabilities threaten the Internet

Dyn attack: Oct. 21, 2016



Oct. 21, 2016 outage map Source: Downdetector.com

Showcase vulnerabilities

➢ From Academic Community

Koscher, K., A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al., 2010: Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy (SP)*, IEEE, pp. 447–462.

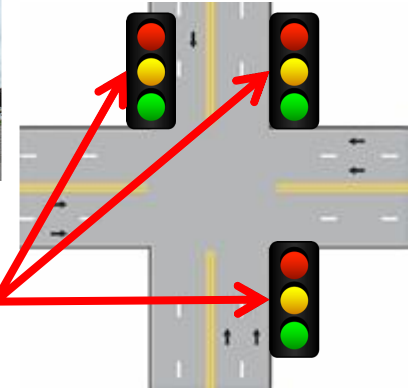
Halperin, D., T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, 2008: Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the 29th Annual IEEE Symposium on Security and Privacy*, pp. 129–142.

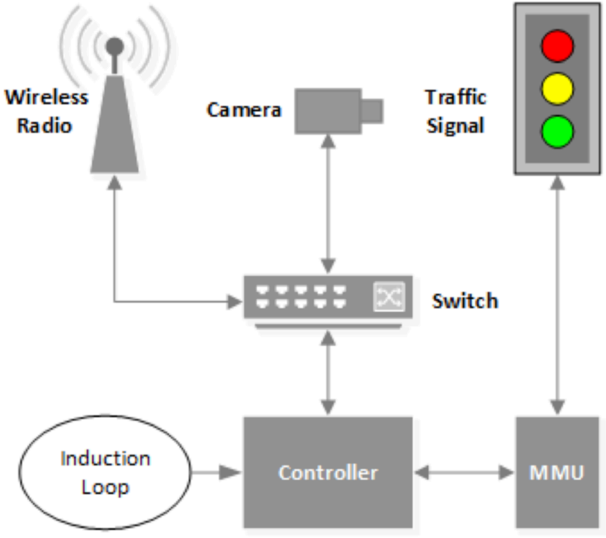
Ghena, B., W. Beyer, A. Hillaker, J. Pevarnek, and J. A. Halderman, 2014: Green lights forever: analyzing the security of traffic infrastructure. In *Proceedings of the 8th USENIX conference on Offensive Technologies*, USENIX Association, pp. 7–7.

Green Lights Forever

- Traffic lights in Ann Arbor (2014)
- Wireless traffic monitoring & mimicing

Traffic lights and controller in Ann Arbor, Michigan

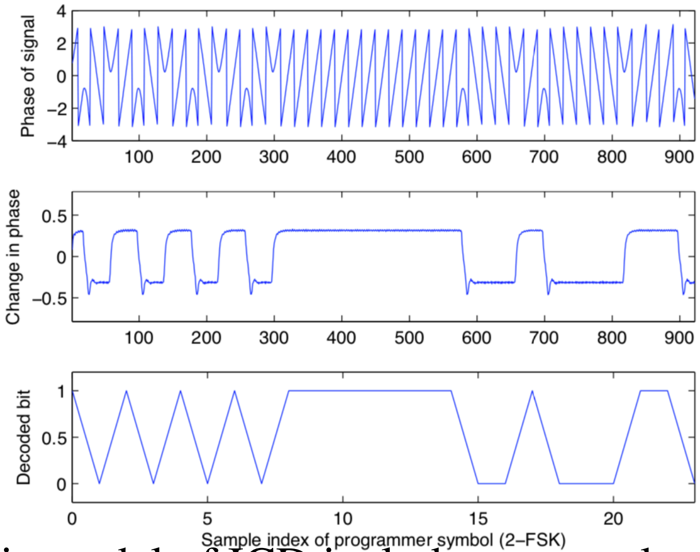


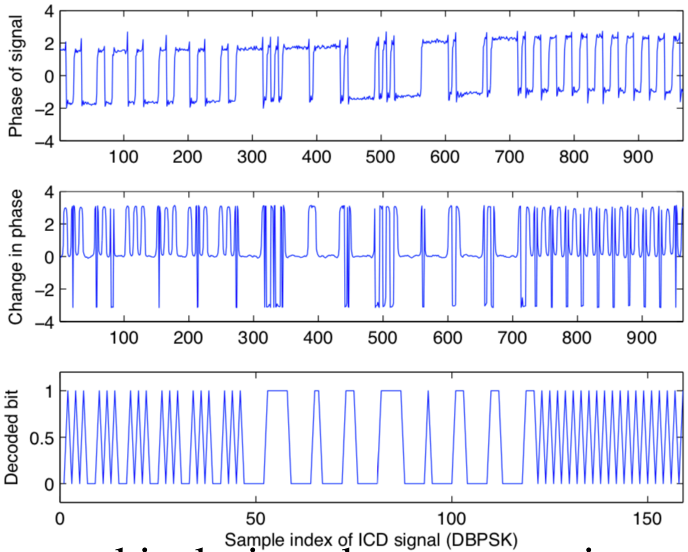


Ghena *et al.*, “Green Lights Forever: Analyzing Security of Traffic Infrastructure,” WOOT 2014.

Eavesdropping and Attack

- Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses

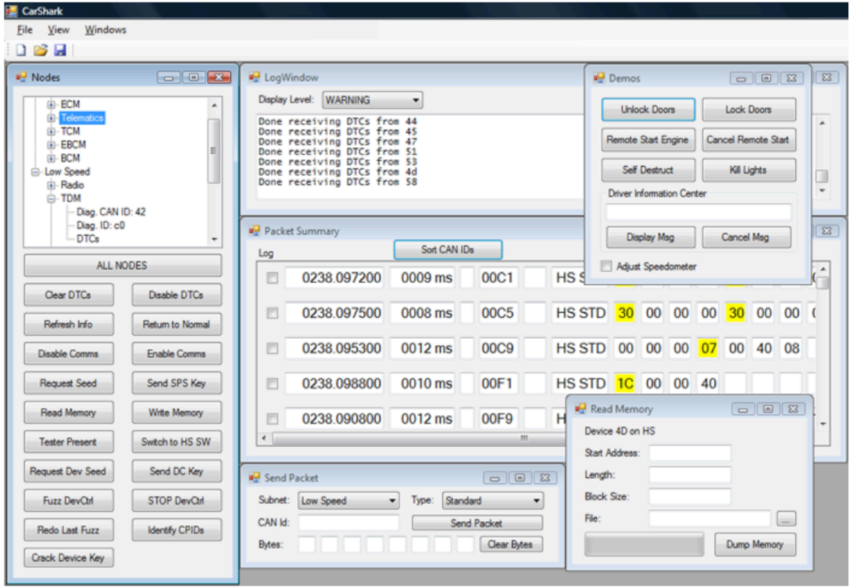


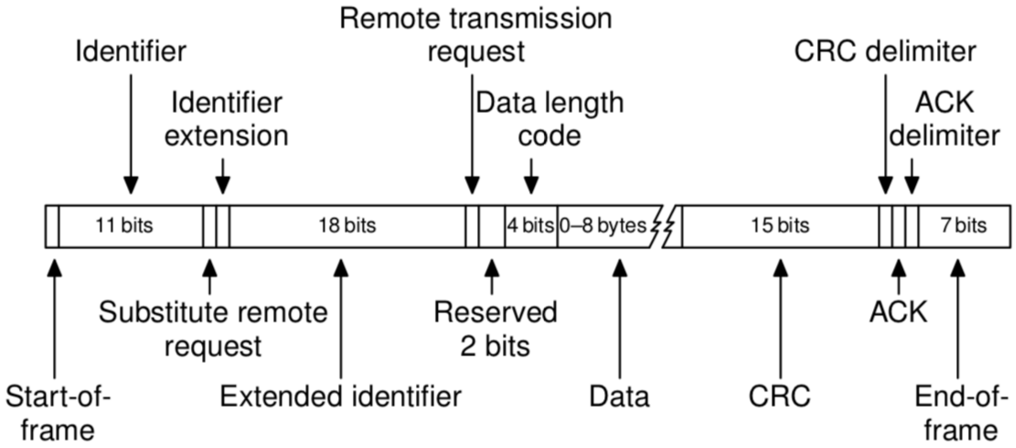


This model of ICD includes pacemaker technology and is designed to communicate wirelessly with a nearby external programmer in the 175 kHz frequency range.

Security Analysis of a Modern Vehicle

➢ Eavesdropping packets in CAN Bus



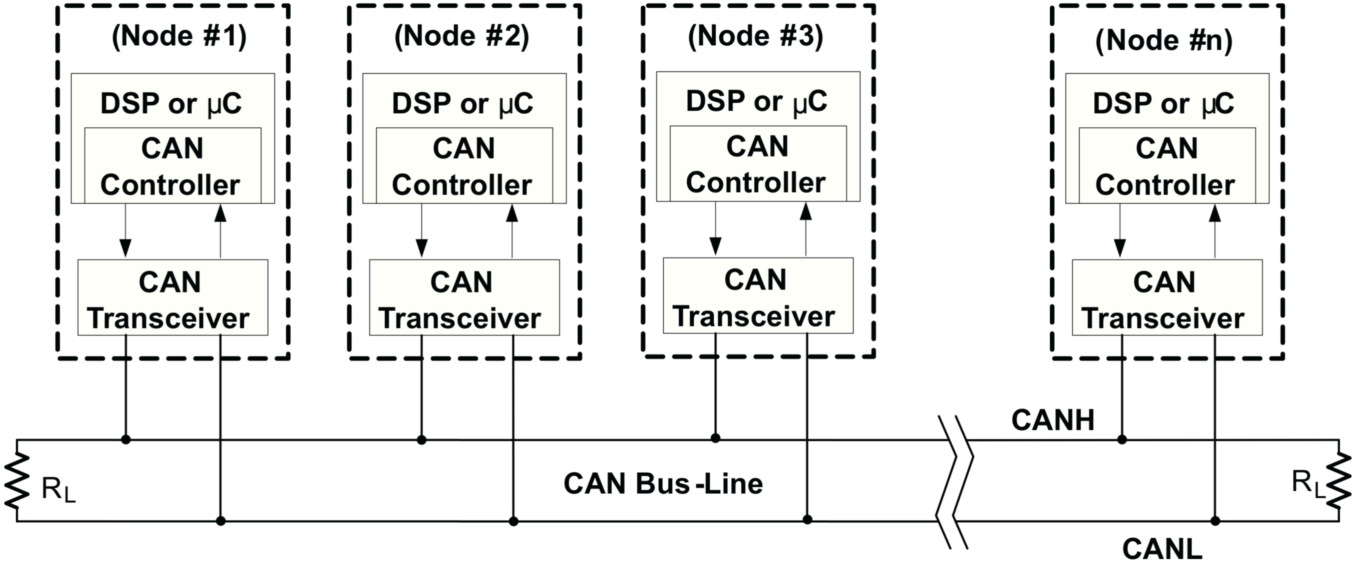


Wireless Carjackers

- https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
- Uconnect over Sprint Network

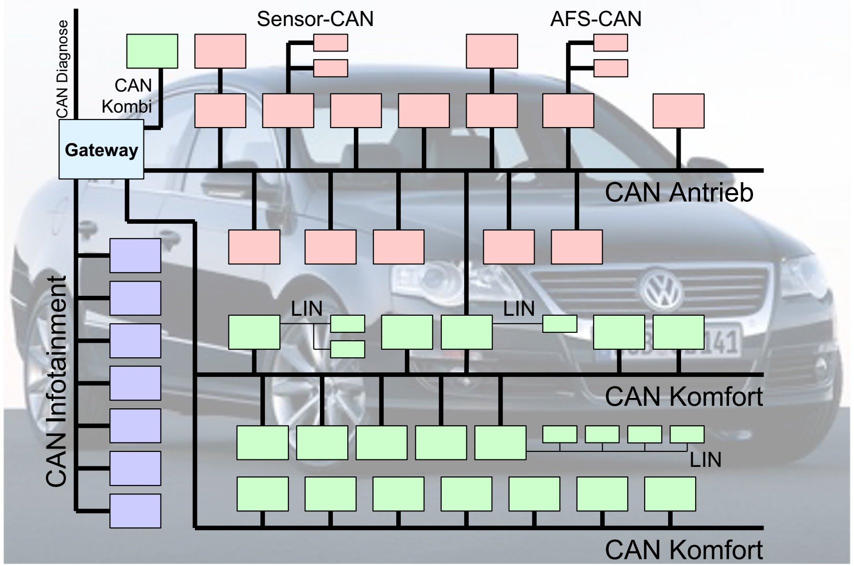
Controller Area Network (CAN)

- Developed by BOSCH as a multi-master, message broadcast system
- Many short messages are broadcast to the entire network, which provides for data consistency in every node of the system



Network architecture of a car

- Electronic Control Unit (ECU)
  - Sensors and actuators
  - Microcontroller
  - Software



- Bus
  - Connects individual ECUs
- Interconnect between buses

Example ECU (Freescale board EVB9512XF)

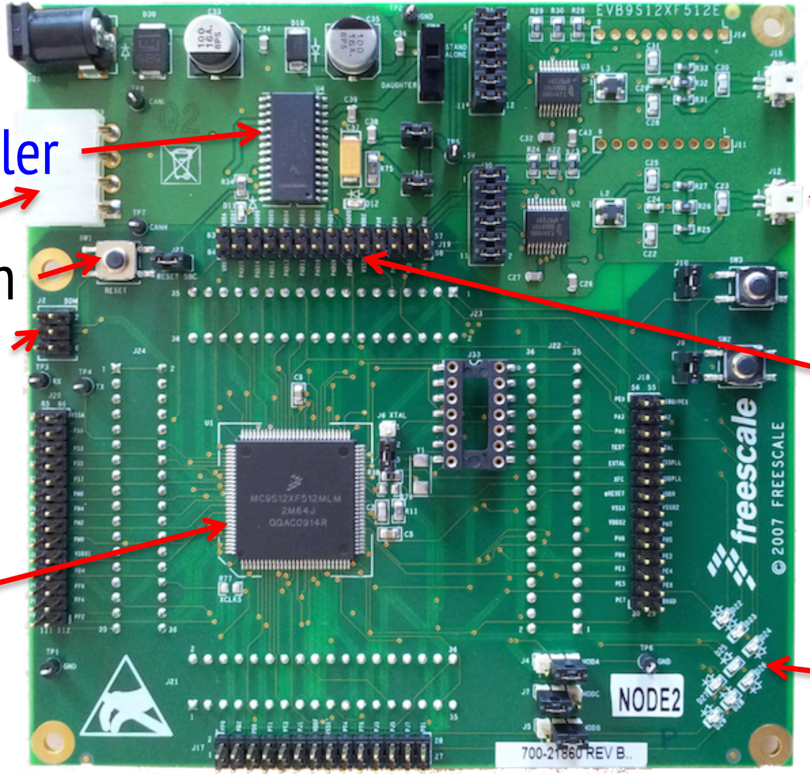
Power

CAN controller

CAN port

Reset button

Debug port



FlexRay port

Digital and Analog I/O ports

Microcontroller (CPU + memory)

LEDs

Properties and Threat Models

- Secrecy/Confidentiality
  - Can secret data be leaked to an attacker?
- Integrity
  - Can the system be modified by the attacker?
- Authenticity
  - Who is the system communicating/interacting with?
- Availability
  - Is the system always able to perform its function?
- Need to think about Threat (attacker) Models

What is network security?

- *confidentiality:* only sender, intended receiver should "understand" message contents
  - Method – encrypt at sender, decrypt at receiver
  - A protocol that prevents an adversary from understanding the message contents is said to provide *confidentiality*.
  - Concealing the quantity or destination of communication is called *traffic confidentiality*.
- *message integrity:* sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
  - A protocol that detects message tampering provides *data integrity*.
  - The adversary could alternatively transmit an extra copy of your message in a *replay attack*.
  - A protocol that detects message tampering provides *originality*.
  - A protocol that detects delaying tactics provides *timeliness*.

What is network security?

- *authentication:* sender, receiver want to confirm identity of each other
  - A protocol that ensures that you really are talking to whom you think you're talking is said to provide *authentication*.
  - Example: DNS Attack [correct URL gets converted to malicious IP]
- *access and availability*: services must be accessible and available to users
  - A protocol that ensures a degree of access is called *availability*.
  - Denial of Service (DoS) Attack
  - Example: SYN Flood attack (Client not transmitting 3rd message in TCP 3-way handshake, thus consuming server's resource)
  - Example: Ping Flood (attacker transmits ICMP Echo Request packets)

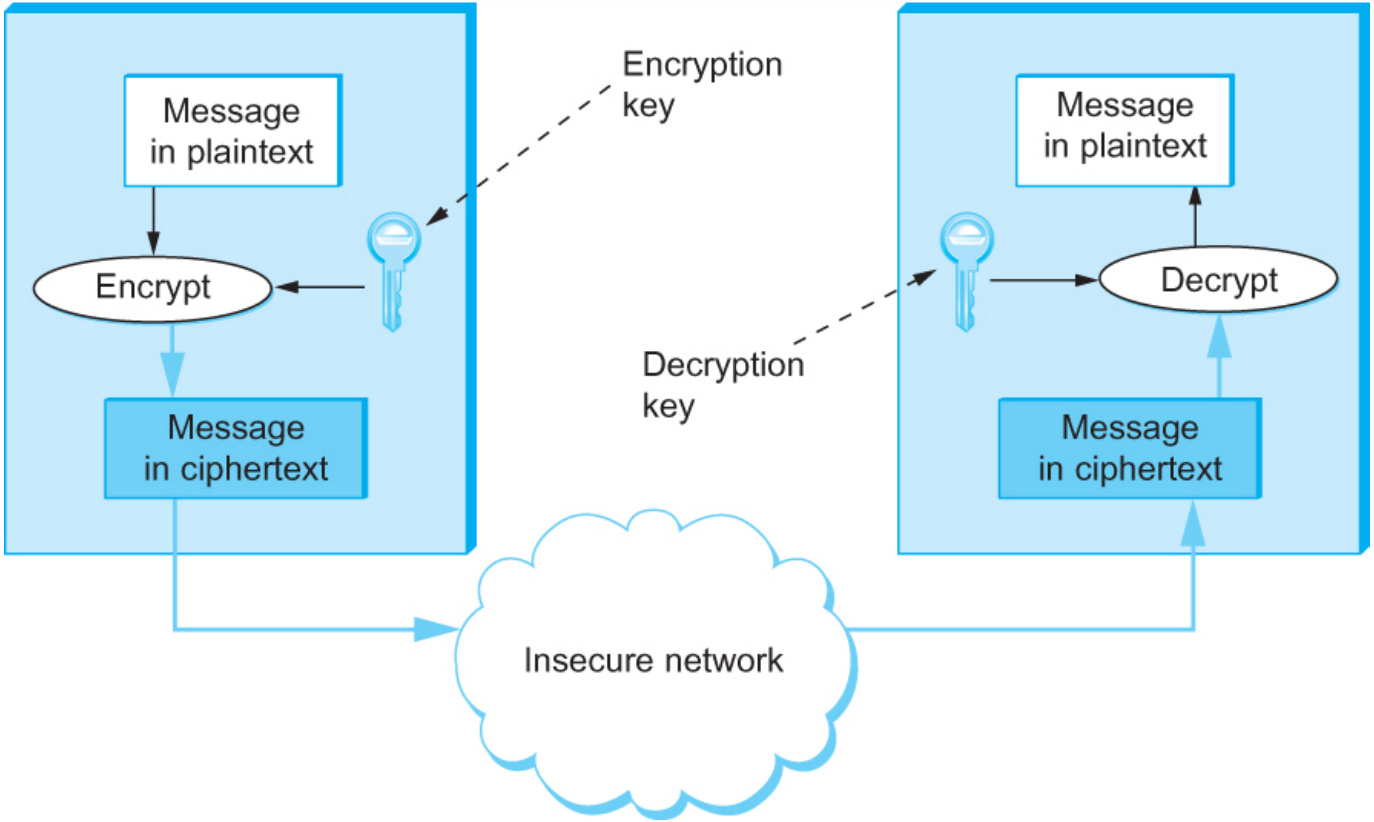
There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

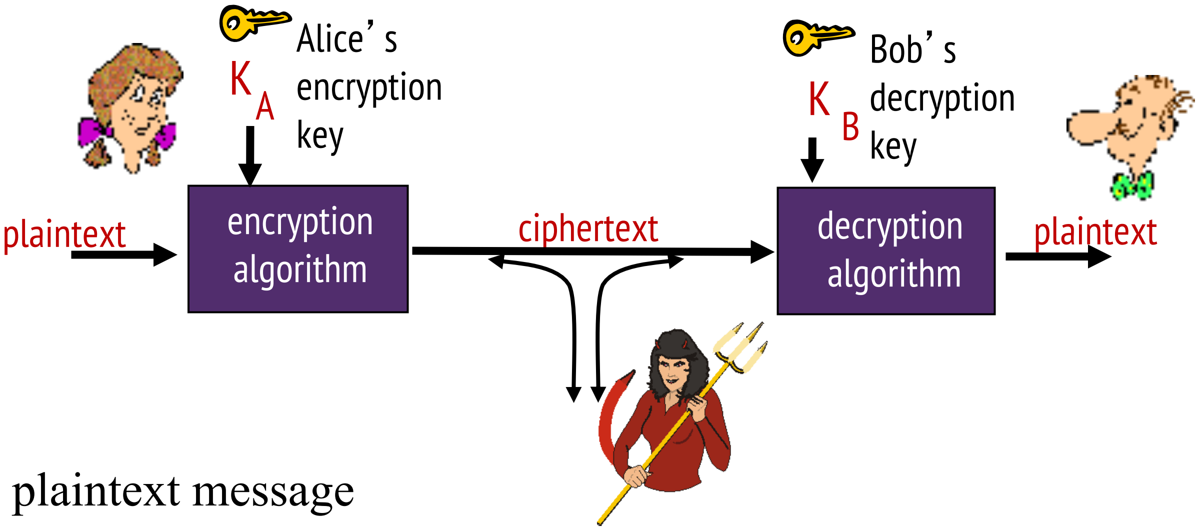
A: A lot!

- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

Cryptography in Insecure Network



The language of cryptography



m plaintext message

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

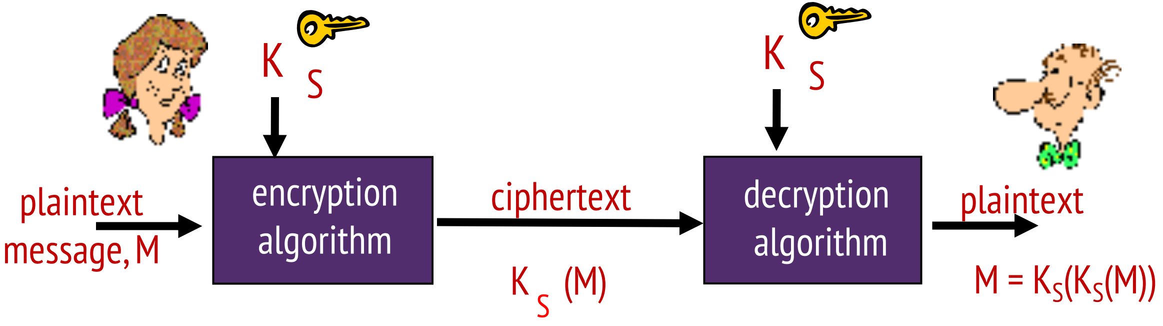
$m = K_B(K_A(m))$

Kerckhoff's Principle

- A cryptographic algorithm should be secure even if everything about the system, except the key, is public knowledge.
- Even if adversary knows the algorithm, he should be unable to recover the plaintext as long as he does not know the key.

Symmetric key cryptography

n-bit plaintext message, $M = m_1 m_2 m_3 \ldots m_n \in \{0, 1\}^n$



symmetric key crypto: Bob and Alice share same (symmetric) key: K_s

Two properties:

- Bob should be able to easily recover M from C
- Any adversary who does not know K should not, by observing C, be able to gain any more information about M

One-time Pad

Alice and Bob share an n-bit secret key $K = k_1k_2k_3 \ldots k_n \in \{0, 1\}^n$, where the n bits are chosen independently at random. K is known as the one-time pad.

$$C = M \oplus K. \qquad \text{Bit-wise XOR}$$

To decode C,

$$C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 0 = M.$$

This uses the facts that exclusive OR ($\oplus$) is associative and commutative, that $B \oplus B = 0$ for any B, and that $B \oplus 0 = B$ for any B.

How is One-Time Pad Secure?

- Assumptions:
  - Eve observes C.
  - Fixed plaintext message M (Eve does not know).

- Every unique ciphertext $C \in \{0, 1\}^n$ can be obtained from M with a corresponding unique choice of key K
  - Set $K = C \oplus M$ where C is the desired ciphertext
  - $C = M \oplus K = M \oplus (C \oplus M) = C \oplus (M \oplus M) = C$

- A uniformly random bit-string $K \in \{0, 1\}^n$ generates a uniformly random ciphertext $C \in \{0, 1\}^n$.
- Thus, with known C, Eve can do no better than guessing at the value of K uniformly at random.

Use the key more than once?

- Eve has access to two ciphertexts
  - $C_1 = M_1 \oplus K$ and $C_2 = M_2 \oplus K$

- Eve computes $C_1 \oplus C_2$
  - $C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K) = (M_1 \oplus M_2)$

- Eve has partial knowledge of M

- If Eve knows one of the messages
  - It can decode other M
  - It can decode Key K

Simple encryption scheme

substitution cipher: substituting one thing for another

- *monoalphabetic* cipher: substitute one letter for another

plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq

e.g.:

Plaintext: bob. i love you. alice

ciphertext: nkn. s gktc wky. mgsbc

Encryption key: mapping from set of 26 letters to set of 26 letters

Breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis

A chosen-plaintext attack is more powerful than known-plaintext attack

- known-plaintext attack: Trudy has plaintext corresponding to ciphertext [when an intruder knows some of the (plain, cipher) pairings]
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext
  - If Trudy could get Alice to send encrypted message, "The quick brown fox jumps over the lazy dog", then the encryption is broken.

Polyalphabetic Cipher

Plaintext letter:	a b c d e f g h i j k l m n o p q r s t u v w x y z
$C_1(k = 5)$:	f g h i j k l m n o p q r s t u v w x y z a b c d e
$C_2(k = 19)$:	t u v w x y z a b c d e f g h i j k l m n o p q r s

- n substitution ciphers, $C_1, C_2, \ldots, C_n$
- cycling pattern:
  - e.g., n=4 [C_1-C_4], k=key length=5: C_1,C_3,C_4,C_3,C_2; C_1,C_3,C_4,C_3,C_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from C_1, o from C_3, g from C_4

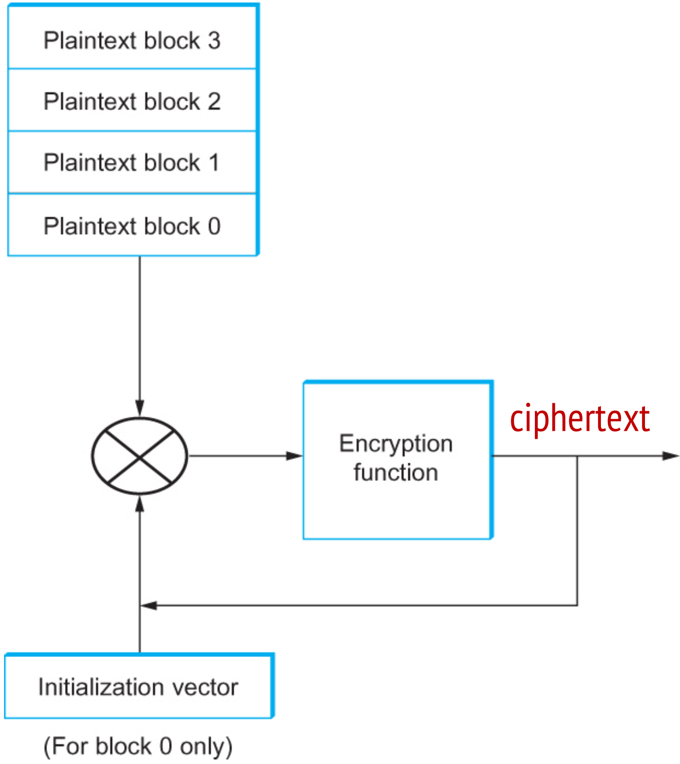
  Encryption key: n substitution ciphers, and cyclic pattern
  - key need not be just n-bit pattern

Block vs Stream Cipher

- Block ciphers process messages into blocks, each of which is then en/decrypted
  - 64-bits or more
  - Example: DES, AES
- Stream ciphers process messages a bit or byte at a time when en/decrypting
  - Example: WEP (used in 802.11)
- Brute Force attack is possible if few number of bits are chosen

Cipher Block Chaining

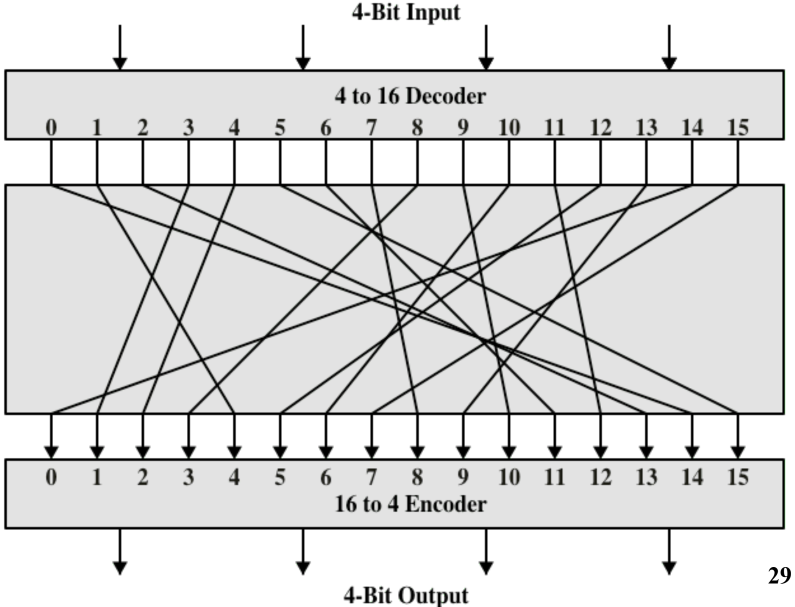
- Plaintext block is XORed with the previous block's ciphertext before being encrypted.
  - Each block's ciphertext depends on the preceding blocks
  - First plaintext block is XORed with a random number.
    - ✓ That random number, called an *initialization vector (IV), is included with the series of ciphertext blocks so that the first ciphertext* block can be decrypted.
- Provides better efficiency for brute force attack



Block Cipher (Basics)

- Operates on a plaintext block of n bits to produce a ciphertext block of n bits.
- There are 2^n possible different plaintext blocks
- For the encryption to be reversible, each must produce a unique ciphertext block.
- Such a transformation is called reversible, or nonsingular.

A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits.



Ideal Block Cipher

- Feistel refers to this as the *ideal block cipher*
  - it allows for the maximum number of possible encryption mappings from the plaintext block
- Practical Problem
  - Small block size degenerates to substitution cipher
  - Note: not a problem of block cipher, but choice of n

Key length (Ideal Block Cipher)

- Mapping is the key
  - the key that determines the specific mapping from among all possible mappings
- the required key length is (4 bits) x (16 rows) = 64 bits
- The length of the key is n x 2^n bits
- For a 64-bit block the required key length is 64 x 2^{64} ~ 10^{21} bits

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions and permutations

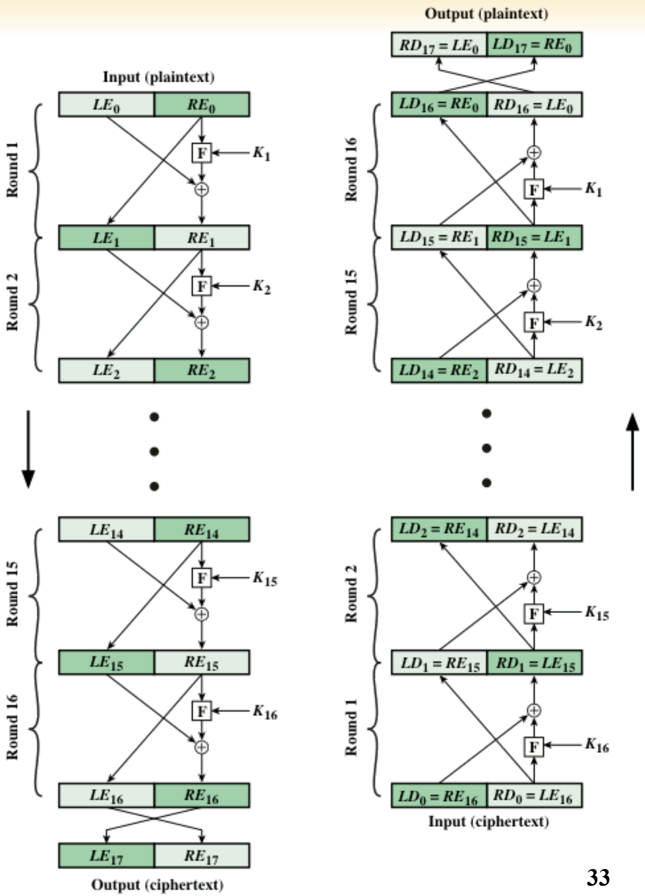
Substitutions
- Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements

Permutation
- No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

- Is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions
- Is the structure used by many significant symmetric block ciphers currently in use

Feistel Cipher

- Block size and Key Size
  - Larger block/key sizes → greater security
  - Larger block/key sizes → reduced encryption/decryption speed
- Number of rounds
  - a single round offers inadequate security but that multiple rounds offer increasing security
- Subkey generation algorithm
  - Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis



Symmetric key crypto: DES

DES: Data Encryption Standard

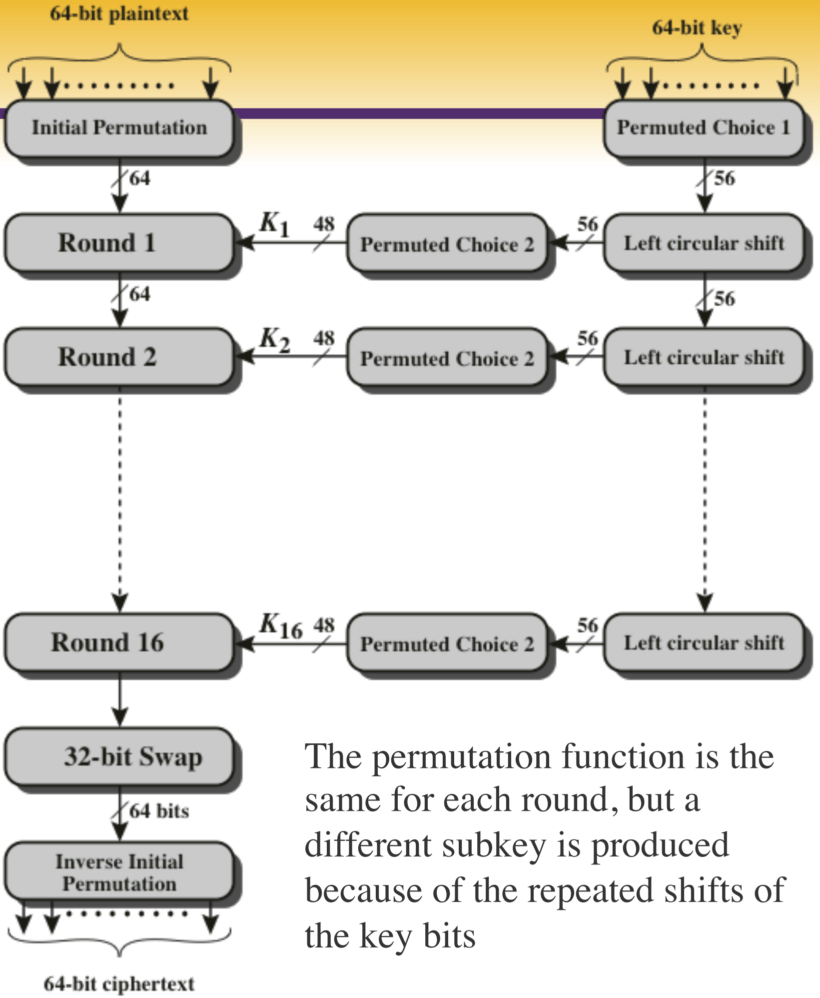
- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase, decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

DES

- initial permutation (on 64 bits)
- 16 identical “rounds” of function application
  - each using different 48 bits of key
  - a subkey (K_i) is produced by the combination of a left circular shift and a permutation
  - rightmost 32 bits are moved to leftmost 32 bits
- final permutation (on 64 bits)

Kaufman, Schneier, 1995

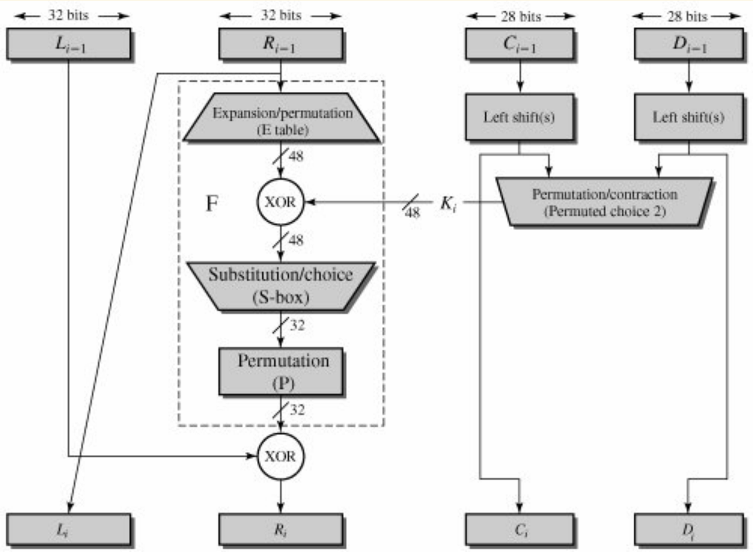
With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher



The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits

Each round of DES

- K_i is 48 bits, R input is 32 bits.
- R is first expanded to 48 bits
  - a table defines a permutation plus an expansion that involves duplication of 16 of the R bits
- Resulting 48 bits are XORed with Ki
- This 48-bit result passes through a substitution function (S box) that produces a 32-bit output
- This is permuted



$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \times F(R_{i-1}, K_i)$$

AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Public Key Cryptography

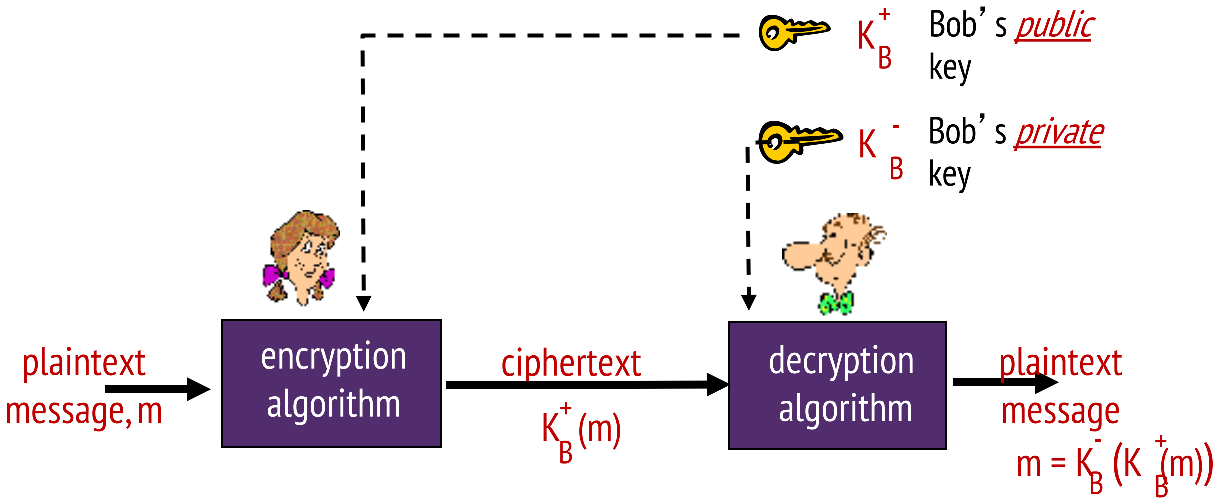
symmetric key crypto

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key crypto

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public key cryptography



Public key encryption algorithms

RSA: Rivest, Shamir, Adelson algorithm [1999]

requirements:

1. need $K_B^+(\ )$ and $K_B^-(\ )$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key K_B^+, it should be impossible to compute private key K_B^-

RSA's security relies on the difficulty of finding p and q knowing only n (the "factorization problem").

Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:

  [(a mod n) + (b mod n)] mod n = (a+b) mod n

  [(a mod n) - (b mod n)] mod n = (a-b) mod n

  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus

  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: x=14, n=10, d=2:

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number

example:

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

1. choose two large prime numbers *p, q.* (e.g., 1024 bits each)

2. compute $n = pq$, $z = (p-1)(q-1)$

3. choose *e* (with $e<n$) that has no common factors with z (*e, z* are "relatively prime").

4. choose *d* such that $ed-1$ is exactly divisible by *z.* (in other words: $ed \bmod z = 1$).

5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above

1. to encrypt message m ($<n$), compute

   $$c = m^e \bmod n$$

2. to decrypt received bit pattern, c, compute

   $$m = c^d \bmod n$$

$$m = \underbrace{(m^e \bmod n)}_{c}{}^{d} \bmod n$$

RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

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

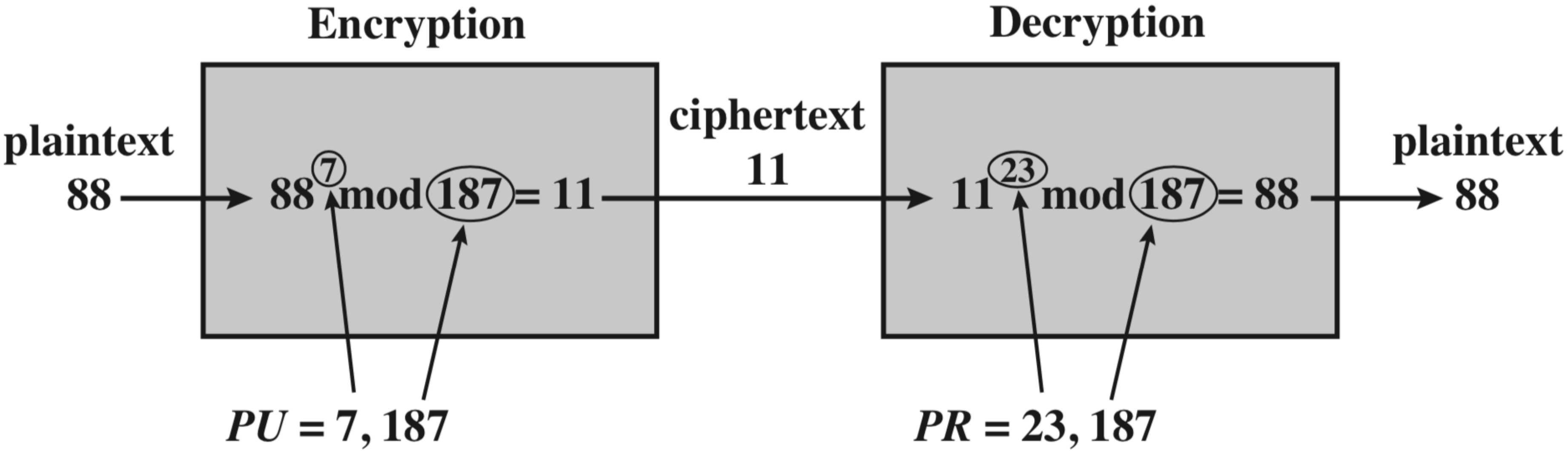
d=29 (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.

	bit pattern	m	m^e	$c = m^e \bmod n$
encrypt:	00001000	12	24832	17

	c	c^d	$m = c^d \bmod n$
decrypt:	17	481968572106750915091411825223071697	12

RSA Example


Encryption
Decryption
plaintext
88
88 7 mod 187 = 11
ciphertext
11
11 23 mod 187 = 88
plaintext
88
PU = 7, 187
PR = 23, 187

Why does RSA work?

- must show that $c^d \bmod n = m$ where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where $n= pq$ and $z = (p-1)(q-1)$
- thus,

$$
\begin{aligned}
c^d \bmod n &= (m^e \bmod n)^d \bmod n \\
&= m^{ed} \bmod n \\
&= m^{(ed \bmod z)} \bmod n \\
&= m^1 \bmod n \\
&= m
\end{aligned}
$$

RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

use public key first, followed by private key

use private key first, followed by public key

result is the same!

How is it possible?

follows directly from modular arithmetic:

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$
$$= m^{de} \bmod n$$
$$= (m^d \bmod n)^e \bmod n$$

Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

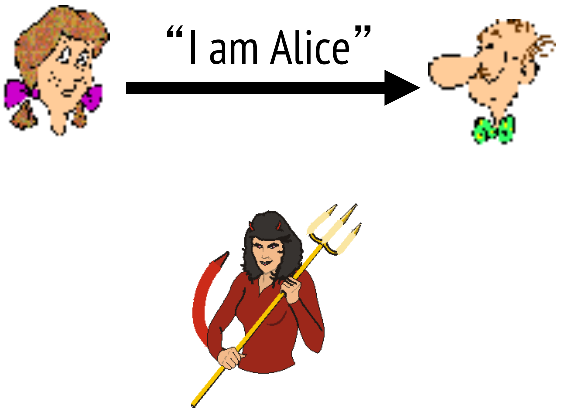
session key, K_S

- Bob and Alice use RSA to exchange a symmetric key K_S
- once both have K_S, they use symmetric key cryptography

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"

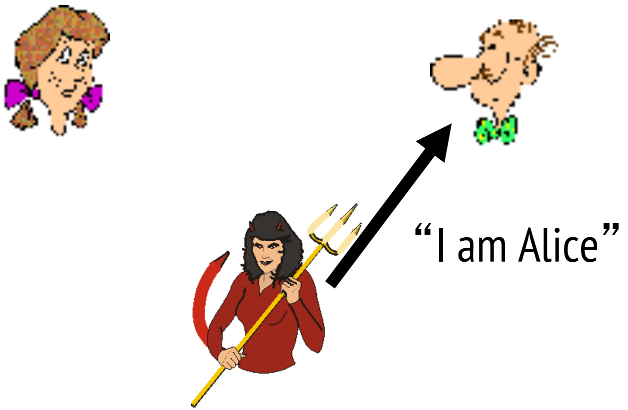


Failure scenario??

Authentication

Goal: Bob wants Alice to "prove" her identity to him

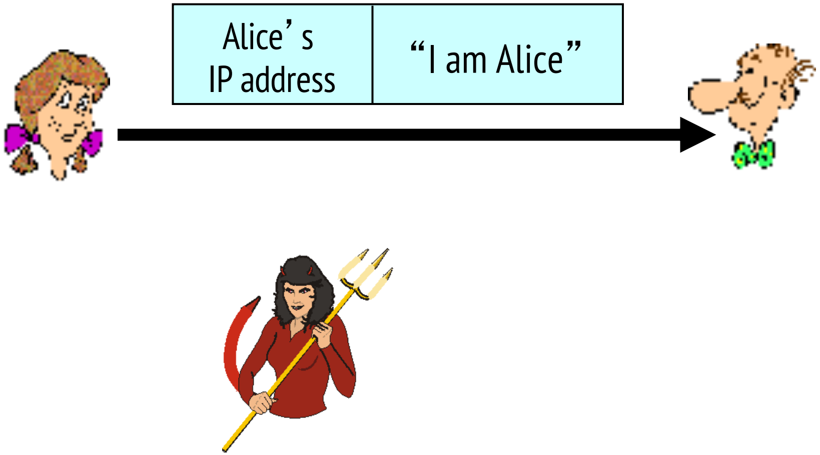
Protocol ap1.0: Alice says "I am Alice"



in a network,
Bob can not "see" Alice, so
Trudy simply declares
herself to be Alice

Authentication: another try

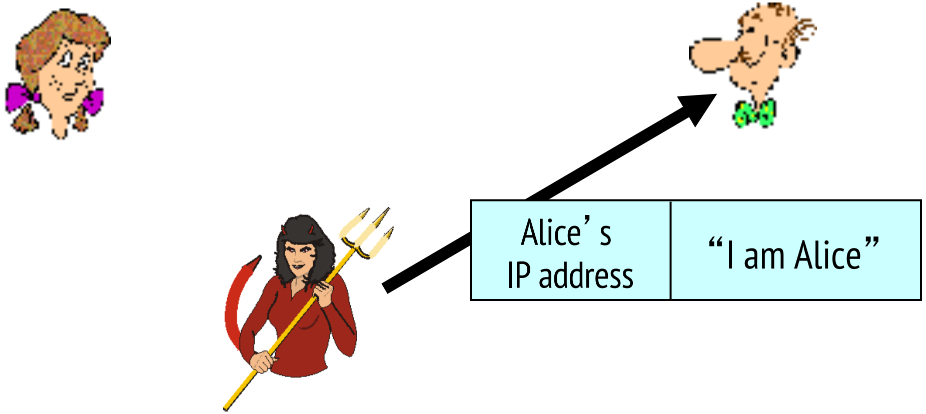
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Failure scenario??

Authentication: another try

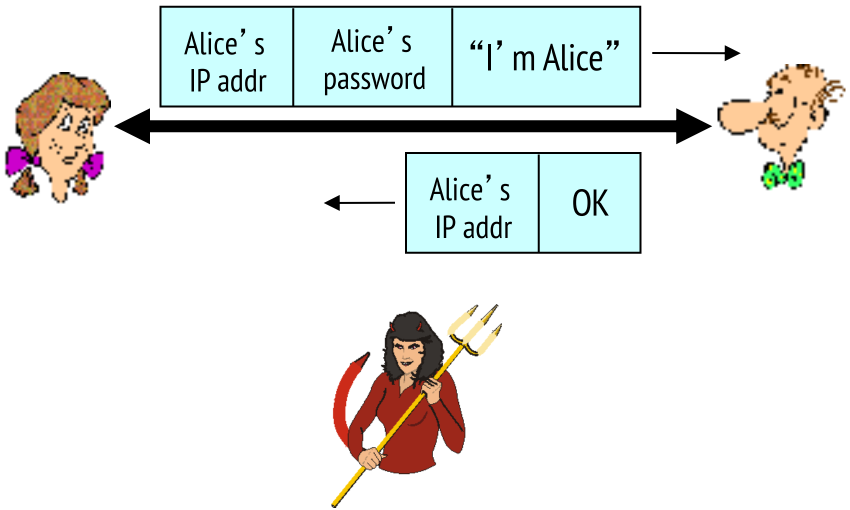
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Trudy can create a packet "spoofing" Alice's address

Authentication: another try

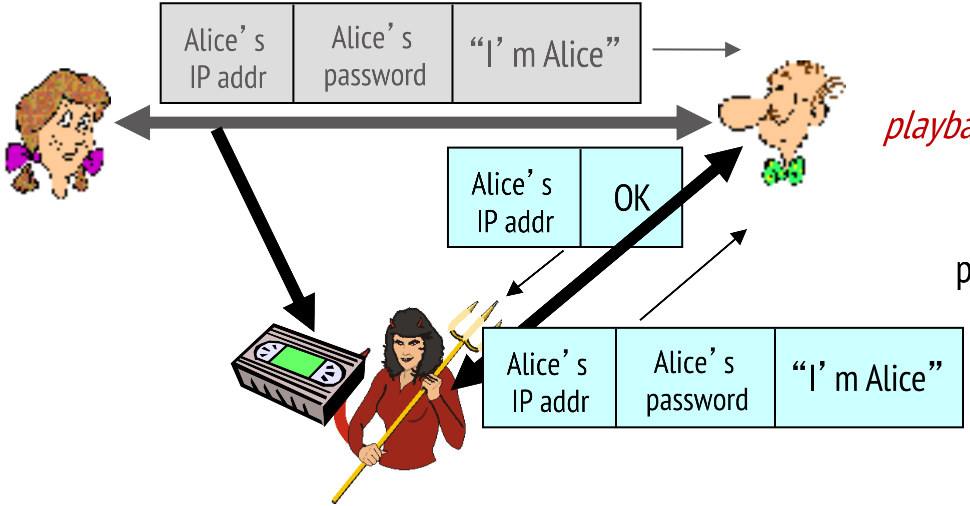
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

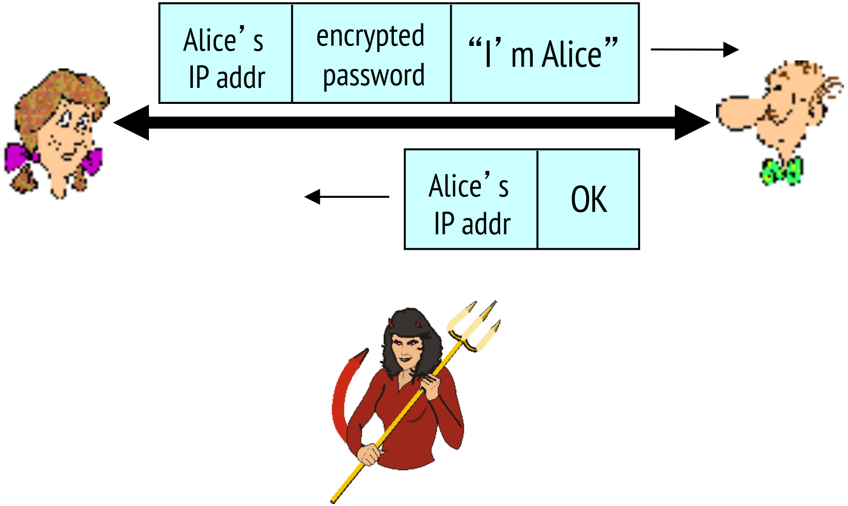
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: yet another try

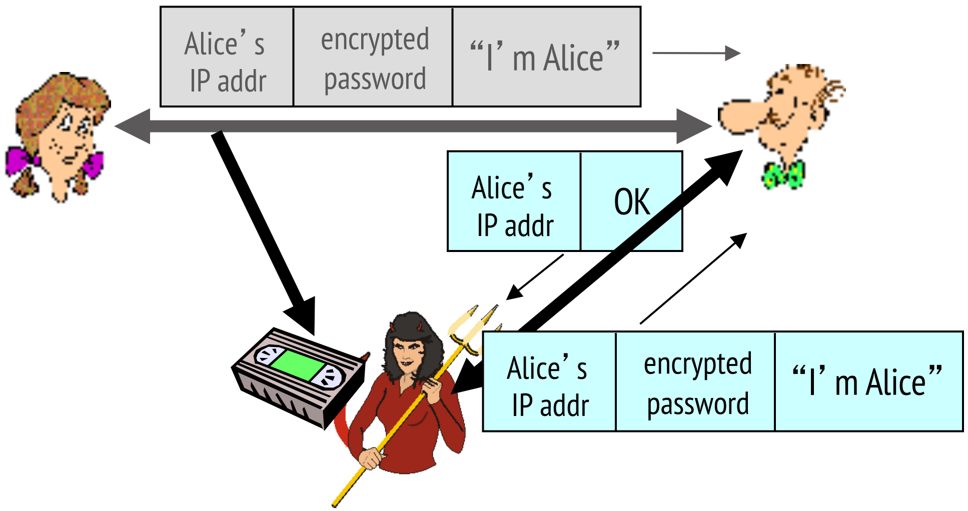
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



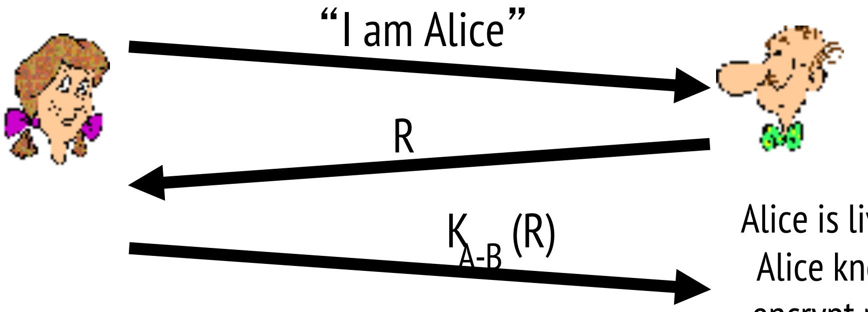
record
and
playback
still works!

Authentication: yet another try

Goal: avoid playback attack

nonce: number (R) used only *once-in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key

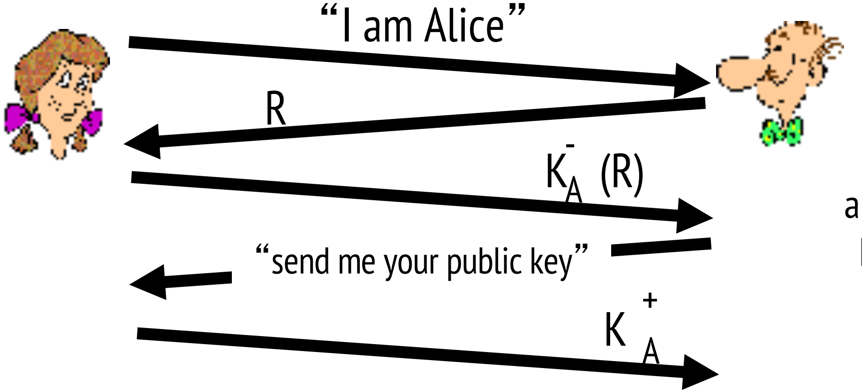


Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Failures, drawbacks?

Authentication: ap5.0

- ap4.0 requires shared symmetric key
- can we authenticate using public key techniques?
- ap5.0: use nonce, public key cryptography



Bob computes

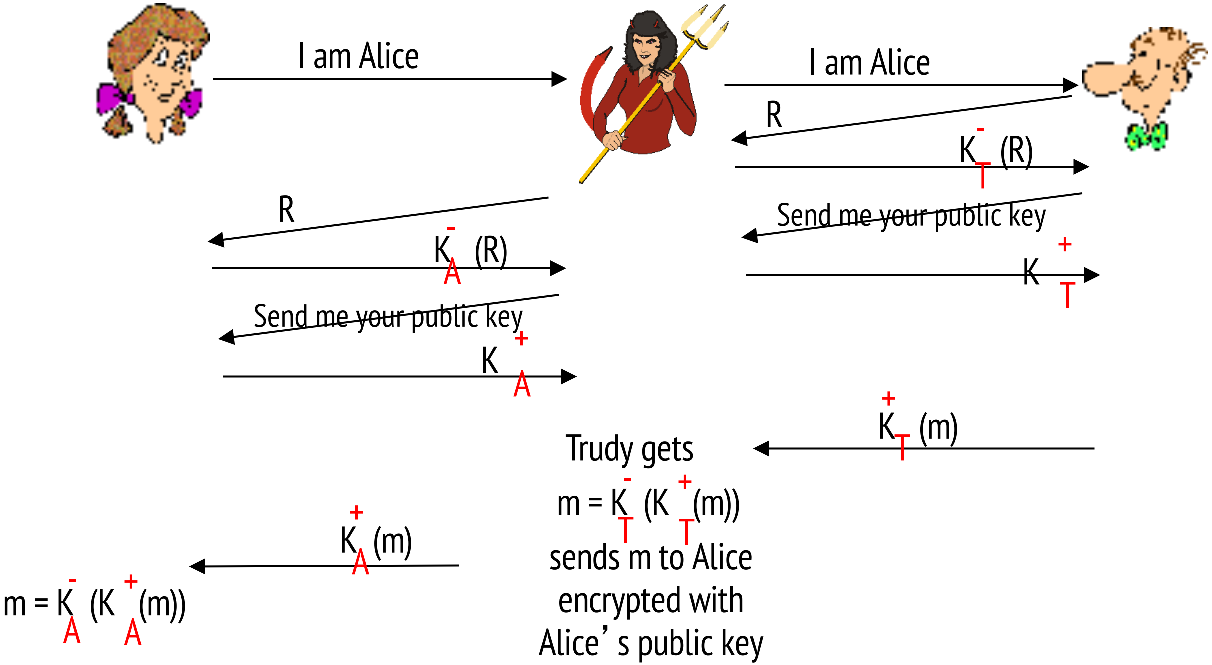
$K_A^+(K_A^-(R)) = R$

and knows only Alice could have the private key, that encrypted R such that

$K_A^+(K_A^-(R)) = R$

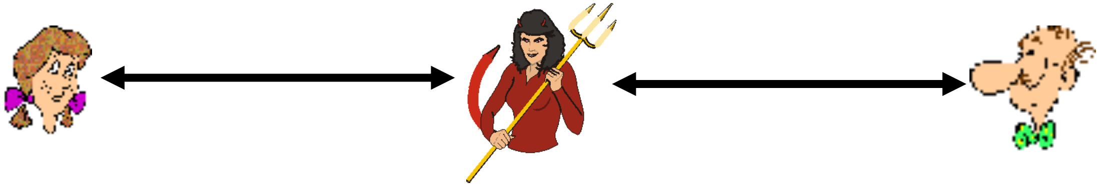
ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Trudy receives all messages as well!

Digital signatures

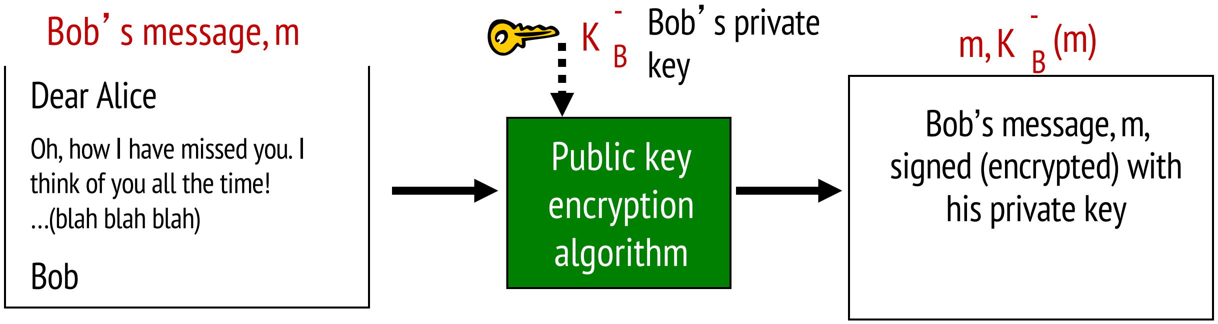
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- *verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital signatures

simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B, creating "signed" message, $K_B(m)$



Digital signatures

- suppose Alice receives msg m, with signature: m, $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

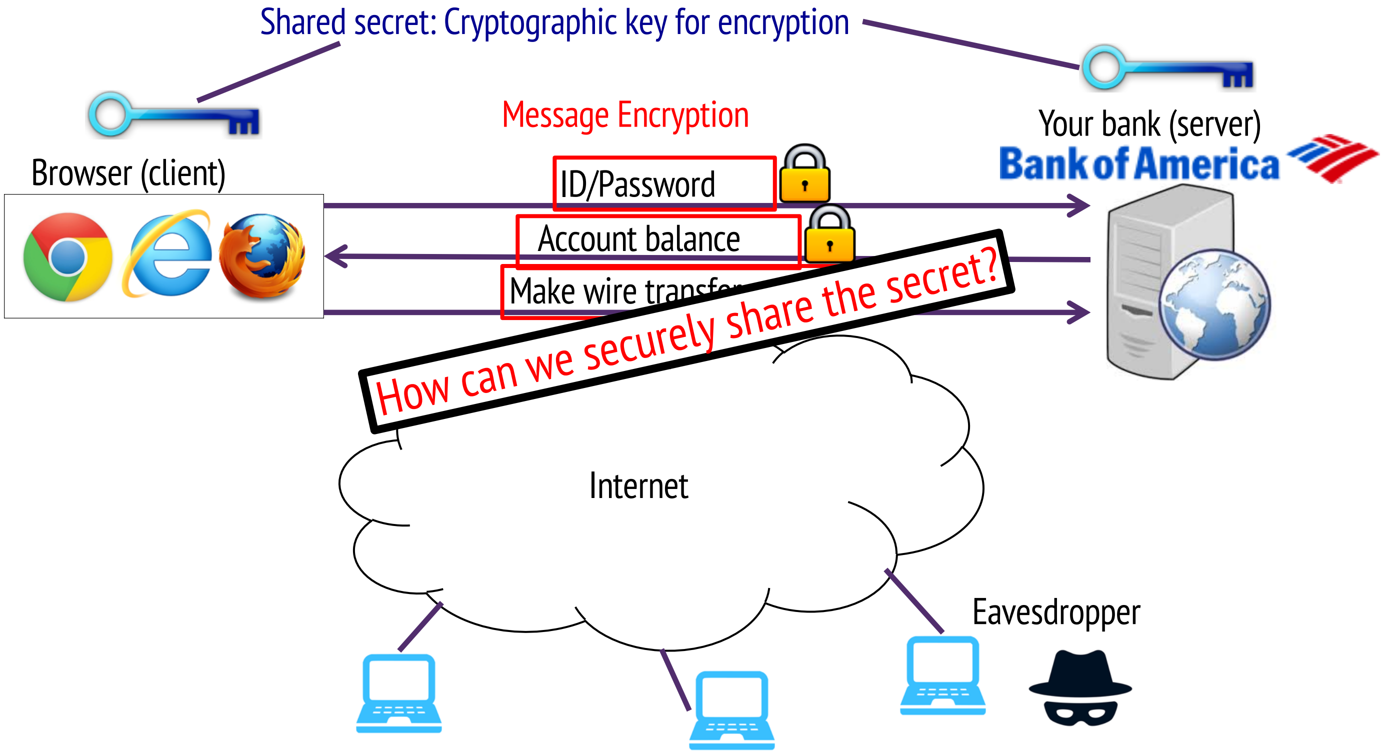
Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

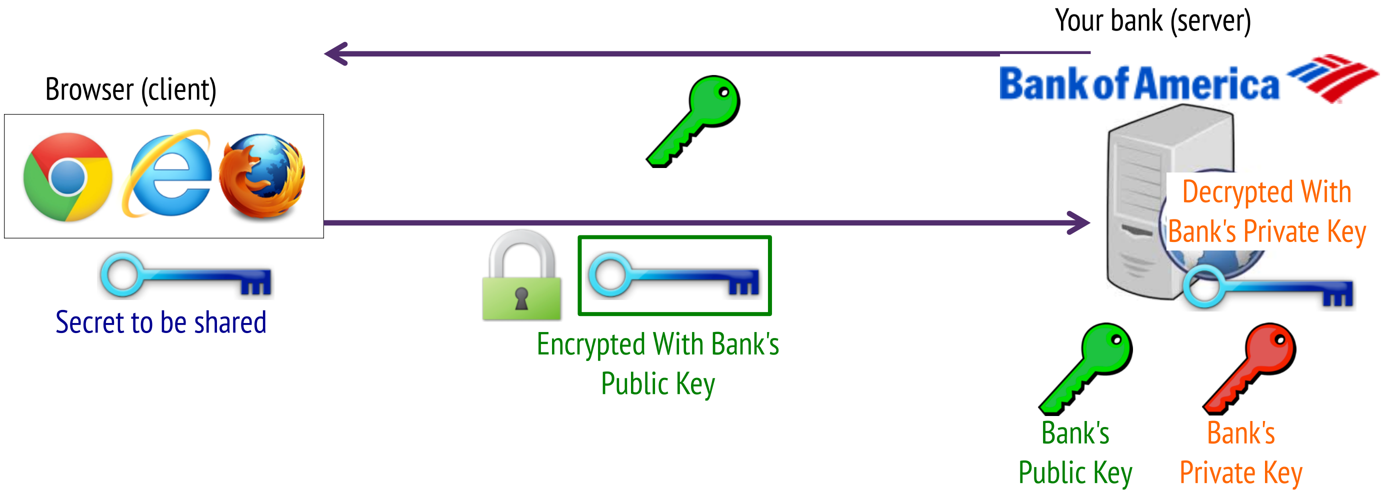
✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

Intro to SSL/TLS Based on Certificates



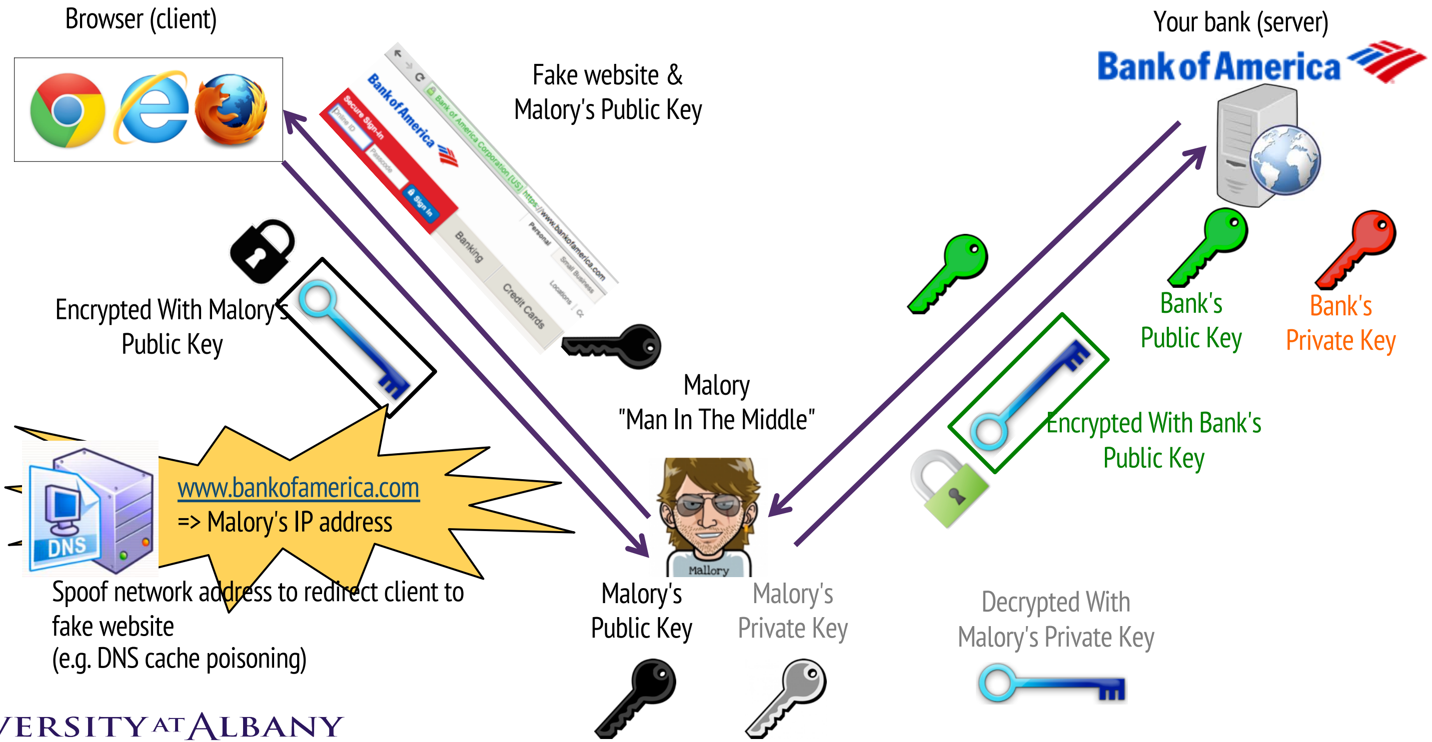
Intro to SSL/TLS Based on Certificates

- Public key cryptography (e.g., RSA)



Intro to SSL/TLS Based on Certificates

➢ However, even with public key cryptography…



Signing a Message

- Each participant has two keys, a public and a private one.

- A message is encrypted with the *private* key and both the message and its encryption are sent.

- The encrypted part can be decrypted with the *public* key. If it matches the plaintext message, the signature is valid.

Intro to SSL/TLS Based on Certificates

A (Digital) Certificate (Proof of Public Key's Authenticity)

- Name of certificate authority (CA)

  Symantec. digicert

- www.bankofamerica.com

  Bank of America

- Bank's public key

```
00:d6: e:3c:c3:b7:b0:0e:c2:a3:0b:a8:7e:d6:
72: e4:29:d0:2b:9b:30:ec:cc:d2:05:c1:0e:17:
41:e0:e3:58:42:98:95:73:06:13:8a:41:99:fb:69:
79:70:95:3e:77:69:c0:70:31:01:6f:fa:22:02:8e:
```

- Additional Information: validity period, etc.

Signed (encrypted)* with issuer (CA)'s Private key

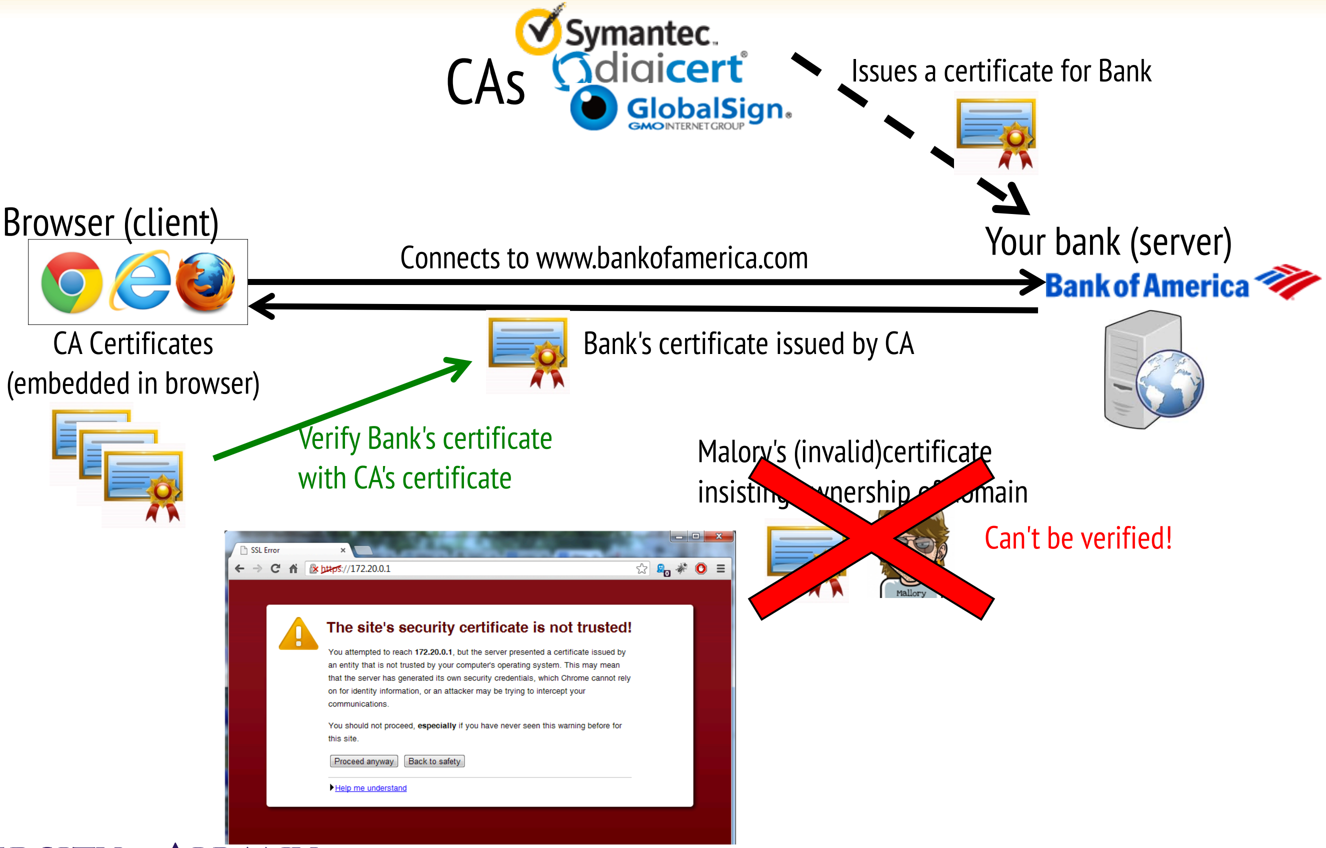
- Digital Signature

```
7c:29:18:b4:57:8e:f5:bf:ec:bc:14:ea:ac:b9:3e:ad:13:dc:
3a:77:e7:7a:ab:3b:23:46:46:4a:2d:ee:7a:d0:
43:d6:17:d1:c6:86:ff:a0:b5:33:c4:de:ec:d8:
a6:cb:0f:02:a8:22:c7:fb:98:b1:75:61:b1:9d:
```

Can only be decrypted (verified) with issuer (CA)'s matching public key!

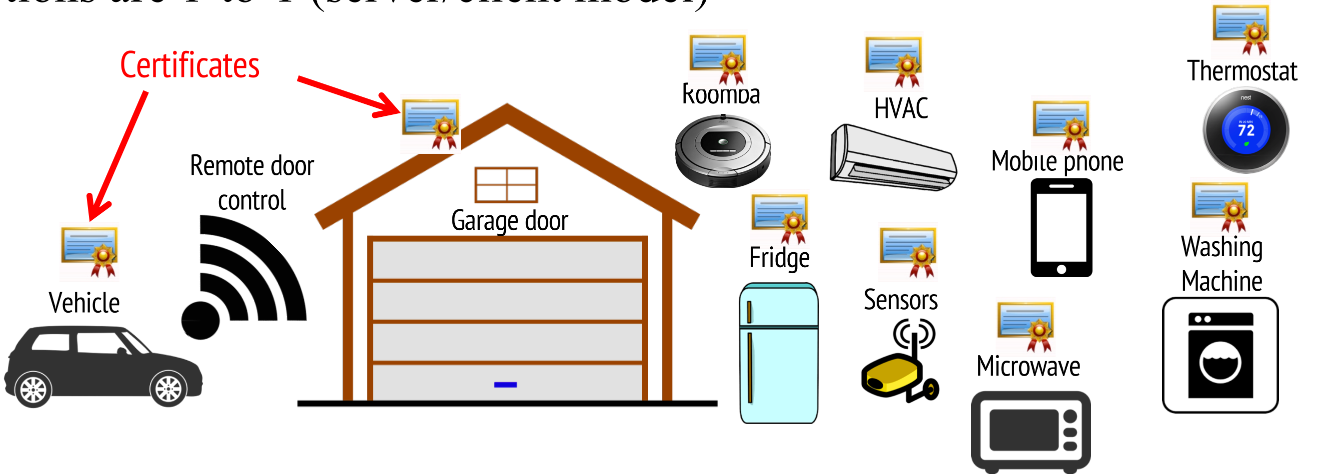
Actually the hash of data is encrypted (signed), and the result of decryption is also hash

Intro to SSL/TLS Based on Certificates



Issues with Using SSL/TLS for IoT

- Overhead for resource-constrained devices
  - Energy/computation overhead for public key crypto, communication bandwidth, memory, etc.
- Limited support one-to-many communication
  - Connections are 1-to-1 (server/client model)



Security: Exploiting Locality

Best Paper Award
IoTDI 2017 (IoT Design a
Implementation)

Locally Centralized, Globally Distributed Authentication and Authorization for the Internet of Things

Hokeun Kim and Edward A. Lee, *University of California, Berkeley*

Abstract— Authentication and authorization are essential parts of basic security processes and are sorely needed in the Internet of Things (IoT). The emergence of edge and fog computing creates new opportunities for security and trust management in the IoT. In this paper, we discuss some existing solutions to establish and manage trust in networked systems and argue that these solutions face daunting challenges when scaled to the IoT. We give a vision of efficient and scalable trust management for the IoT based on locally centralized, globally distributed trust management using an open-source infrastructure with local authentication and authorization entities to be deployed on edge devices.

IT Professional
2017

A Toolkit for Construction of Authorization Service Infrastructure for the Internet of Things

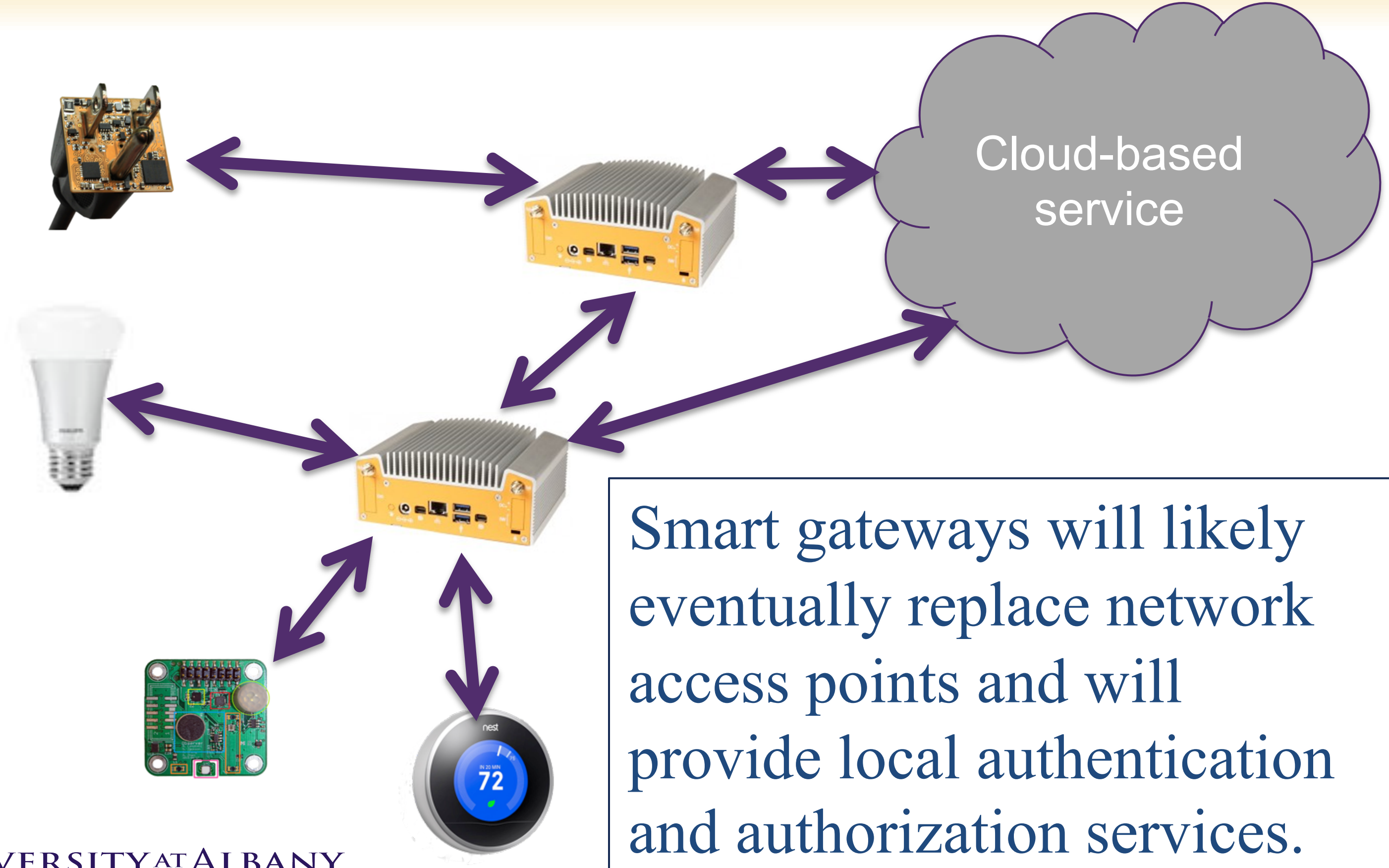
Hokeun Kim
University of California, Berkeley
hokeunkim@eecs.berkeley.edu

Eunsuk Kang
University of California, Berkeley
eunsuk@berkeley.edu

Edward A. Lee
University of California, Berkeley
eal@eecs.berkeley.edu

David Broman
KTH Royal Institute of Technology
dbro@kth.se

Smart Gateways: Exploiting Locality



Smart gateways will likely eventually replace network access points and will provide local authentication and authorization services.

Future of CPS Design

- Rising trend: combine model-based design with data-driven methods (learning from data)

- This course discussed how design is done today, but you can be sure that the technology will change!

- The goal of this course has been to give you what you need to think critically about the technology.