Northeastern University

Network Security: Use & Misuse of Cryptography -- Contemporary Tales

Guevara Noubir

Northeastern University
noubir@ccs.neu.edu

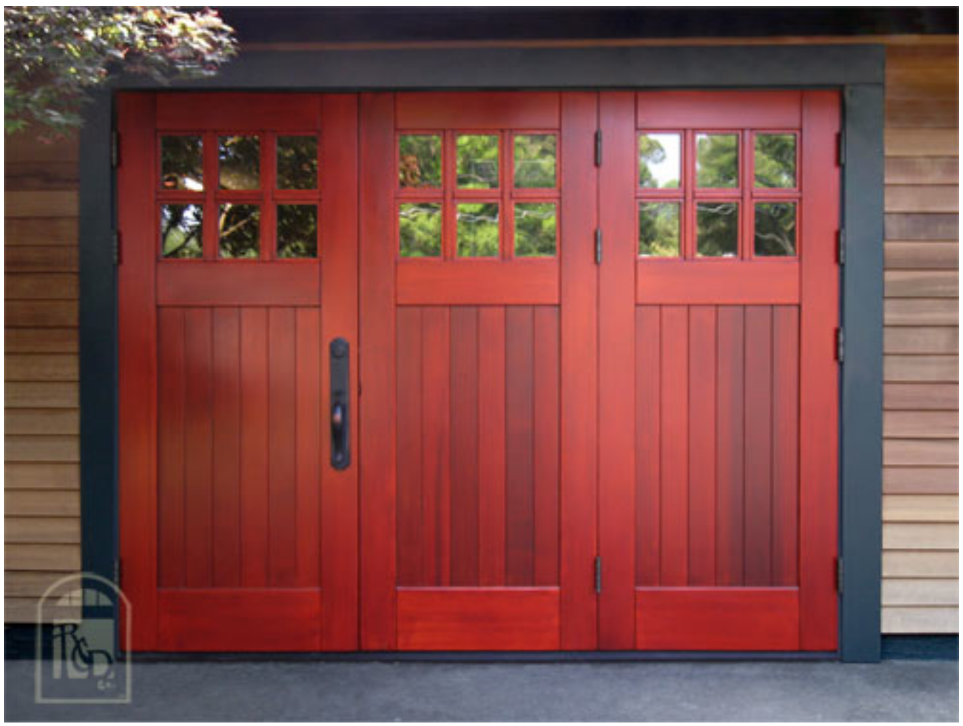
Network Security: the Evolution

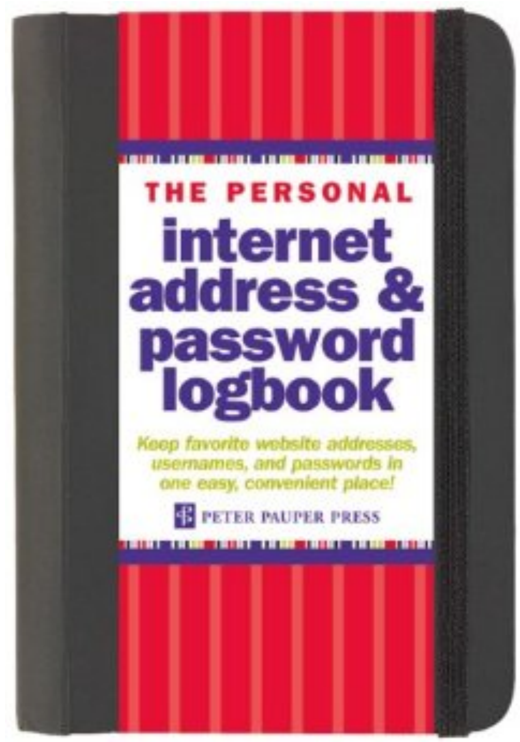
- The early days
  - Internet security
    - Ad hoc mechanisms, obfuscation, little cryptography, address based authentication, firewalls, proprietary protocols
    - Applications: telnet, rlogin (.rhosts), smtp, dns, tcp, arp
  - Cryptography
    - Specialized and sensitive applications, proprietary

- Evolution: cryptography became pervasive
  - TLS/SSL (Web, VPN, WiFi), IPSec, DNSSEC, PGP, DKIM, Kerberos, Tor/Hidden Services, Bitcoin
  - Malicious: FLAME, Cryptolocker, Silk road

Cryptography is not a Panacea

- Secure building block are essential but not sufficient: integration, usability challenges





Outline

- Basics of cryptography: basics & best practices
  - Secret Key Cryptography (symmetric crypto)
  - Modes of Operation of Encryption Algorithms
  - Hashing and Message Authentication Codes
  - Public Key Algorithms (asymmetric crypto)
  - Cryptographic Pseudo Random Numbers Generation
- Overview of applications across the network stack
- Recent misuse of the basics
  - Android Apps, Adobe passwords leaks, Blizzard, PGP
- Systems, Standards
  - TLS/SSL overview, vulnerabilities, and misuse (e.g., WPA-Enterprise)
- Emerging trend of malicious use of cryptography
  - Worms, Ransomware
- Privacy

Cryptography & Network Security

- Cryptography provides the key building blocks for many network security services
- Network Security services
  - Authentication, Confidentiality, Integrity, Access control, Non-Repudiation, Availability, Key Management, Audit
- Cryptographic algorithms (building blocks)
  - Encryption:
    - Symmetric Encryption (e.g., AES), Asymmetric Encryption (e.g., RSA, El-Gamal)
  - Hashing functions
  - Message Authentication Code (e.g., HMAC + SHA1)
  - Digital Signature functions (e.g., RSA, El-Gamal)
  - Cryptographic Pseudo Random Numbers Generation

Terminology & Services

Terminology

- Network security services
  - Authentication, confidentiality, integrity, access control, non-repudiation, availability, key management, auditing
- Security attacks
  - Passive, active
- Cryptography models
  - Symmetric (secret key), asymmetric (public key)
- Cryptanalysis
  - Ciphertext only, known plaintext, chosen plaintext, chosen ciphertext, chosen text

Network Security Services X.800, RFC 2828

- Authentication:
  - assures the recipient of a message the authenticity of the claimed source
- Confidentiality:
  - protects against unauthorized release of message content
- Integrity:
  - guarantees that a message is received as sent (modifications are detected)
- Access control:
  - limits the access to authorized users
- Non-repudiation:
  - protects against sender/receiver denying sending/receiving a message
- Availability:
  - guarantees that the system services are always available when needed
- Security audit:
  - keeps track of transactions for later use (diagnostic, alarms…)
- Key management:
  - allows to negotiate, setup and maintain keys between communicating entities

Network Security Attacks

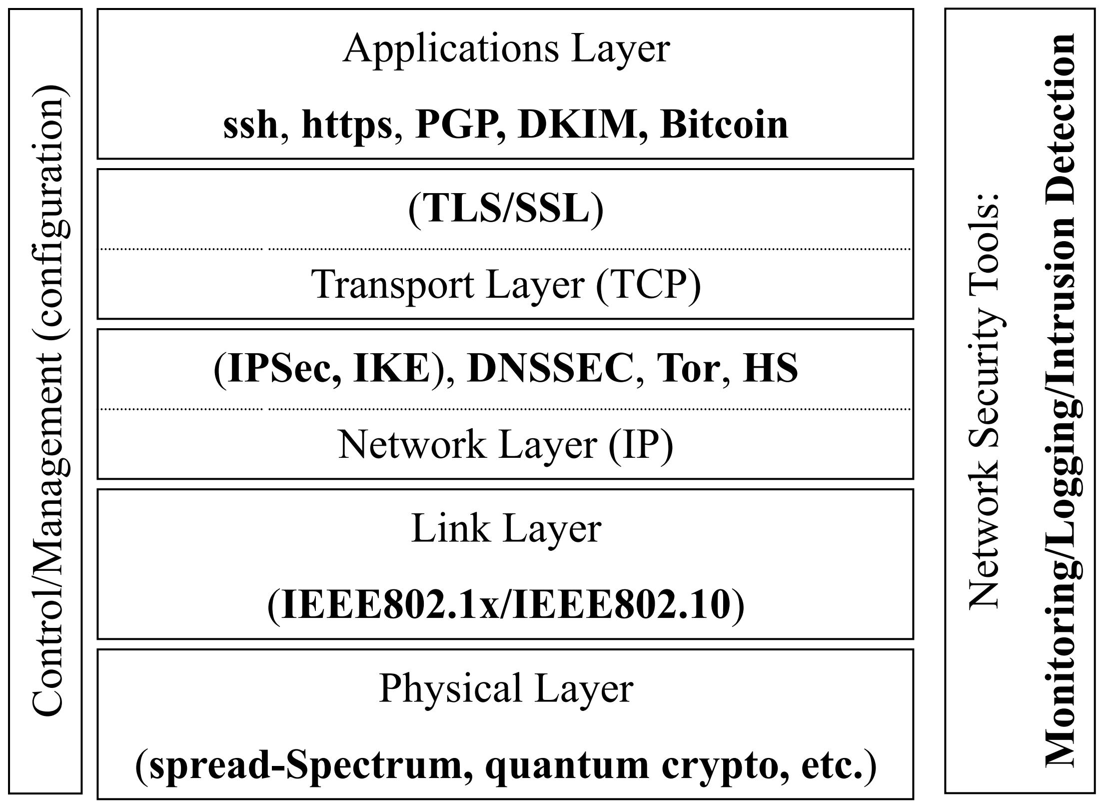
- Kent's classification
  - Passive attacks:
    - Release of message content
    - Traffic analysis
  - Active attacks:
    - Masquerade
    - Replay
    - Modification of message
    - Denial of service
- Security attacks
  - Interception (confidentiality)
  - Interruption (availability)
  - Modification (integrity)
  - Fabrication (authenticity)

Kerchoff's Principle

- The cipher should be secure even if the intruder knows all the details of the encryption process except for the secret key
- "No security by obscurity"
  - Examples of system that did not follow this rule and failed?

Securing Networks

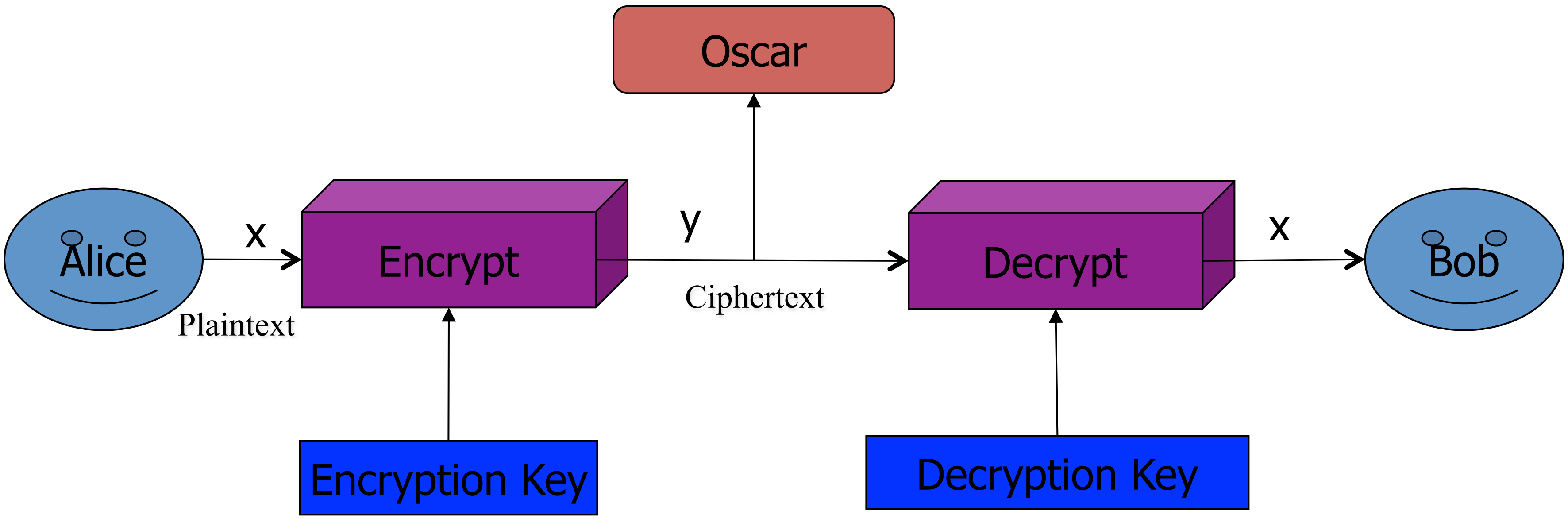
- Where to put the security in a protocol stack?
- Practical considerations:
  - End to end security
  - No modification to OS



Encryption

Encrypted Communication

- Basic Goal:
  - Allow two entities (e.g., Alice, and Bob) to communicate over an insecure channel, such that an opponent (e.g., Oscar) cannot understand what is being communicated

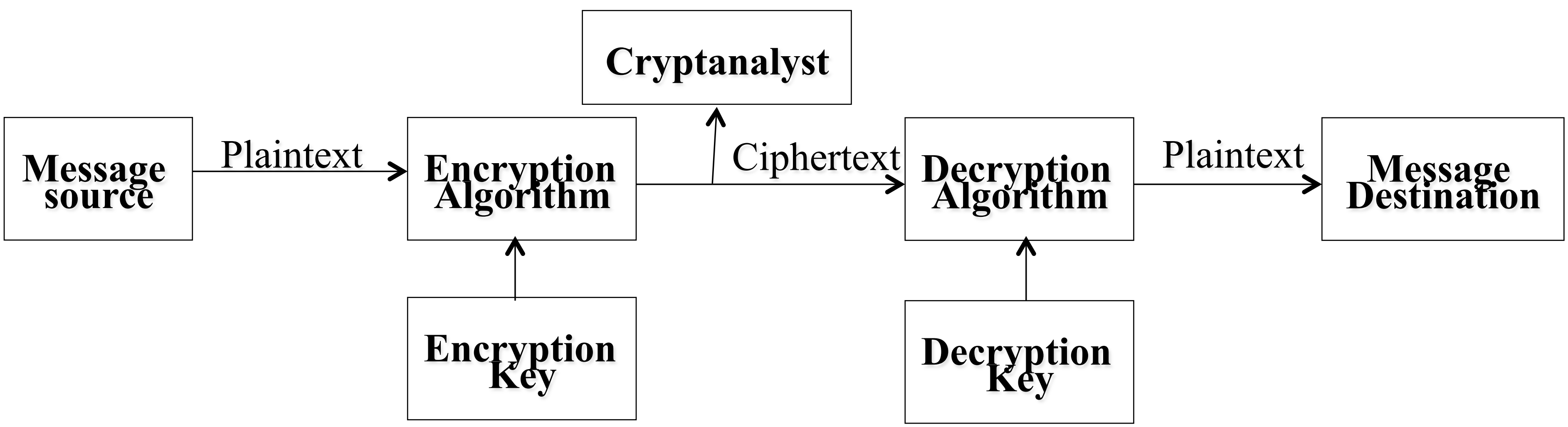


Encryption Algorithms Types

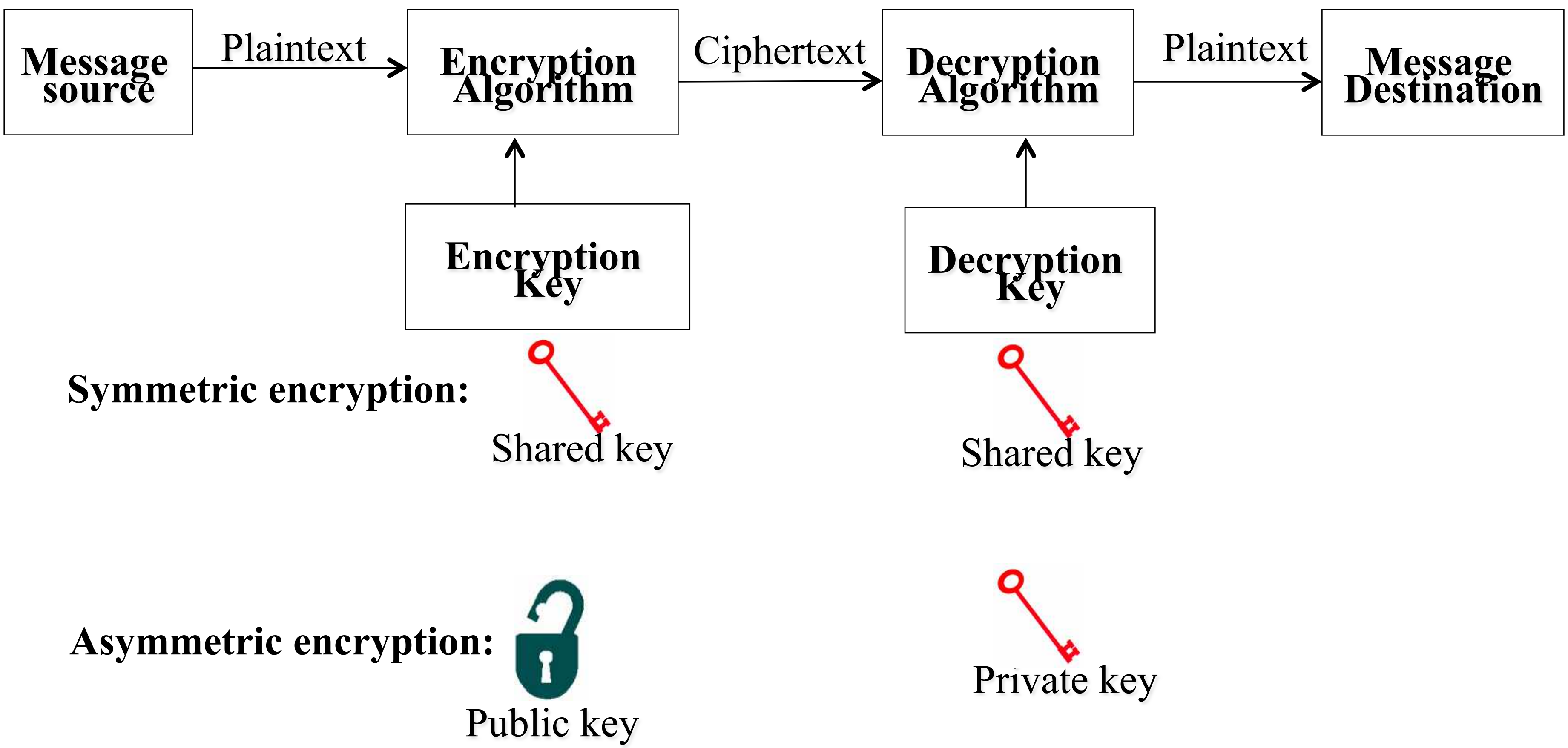
- Block vs. Stream ciphers
  - Block ciphers:
    - Input: block of *n* bits ; Output: block of *n* bits
    - Example: AES
  - Stream ciphers:
    - Input: stream of symbols ; Output: stream of symbols
    - Examples: RC4, GSM A5, SNOW 3G
  - Block ciphers can be used to build stream ciphers (under some assumptions)
    - Examples: AES-CBC

Encryption Models

- Symmetric encryption (conventional encryption)
  - Encryption Key = Decryption Key
  - i.e., Decryption key can be derived from encryption key
  - e.g., AES, DES, FEAL, IDEA, BLOWFISH
- Asymmetric encryption
  - Encryption Key ≠ Decryption Key
  - i.e., Decryption key cannot be derived from encryption key
  - e.g., RSA, Diffie-Hellman, ElGamal



Encryption Models



Symmetric vs. Asymmetric Algorithms

- Symmetric algorithms are much faster
  - In the order of a 1000 times faster
- Symmetric algorithms require a shared secret
  - Impractical if the communicating entities don't have another secure channel
- Both algorithms are combined to provide practical and efficient secure communication
  - E.g., establish a secret session key using asymmetric crypto and use symmetric crypto for encrypting the traffic PGP, TLS/SSL, IKE

Attacks on Encrypted Messages

- Ciphertext only:
  - encryption algorithm, ciphertext to be decoded
- Known plaintext:
  - encryption algorithm, ciphertext to be decoded, pairs of (plaintext, ciphertext)
- Chosen plaintext:
  - encryption algorithm, ciphertext to be decoded, plaintext (chosen by cryptanalyst) + corresponding ciphertext
- Chosen ciphertext:
  - encryption algorithm, ciphertext to be decoded, ciphertext (chosen by cryptanalyst) + corresponding plaintext
- Chosen text:
  - encryption algorithm, ciphertext to be decoded, plaintext + corresponding ciphertext (both can be chosen by attacker)

- Modern cryptography: better models (Game-based proofs)
  - IND-CPA, etc.

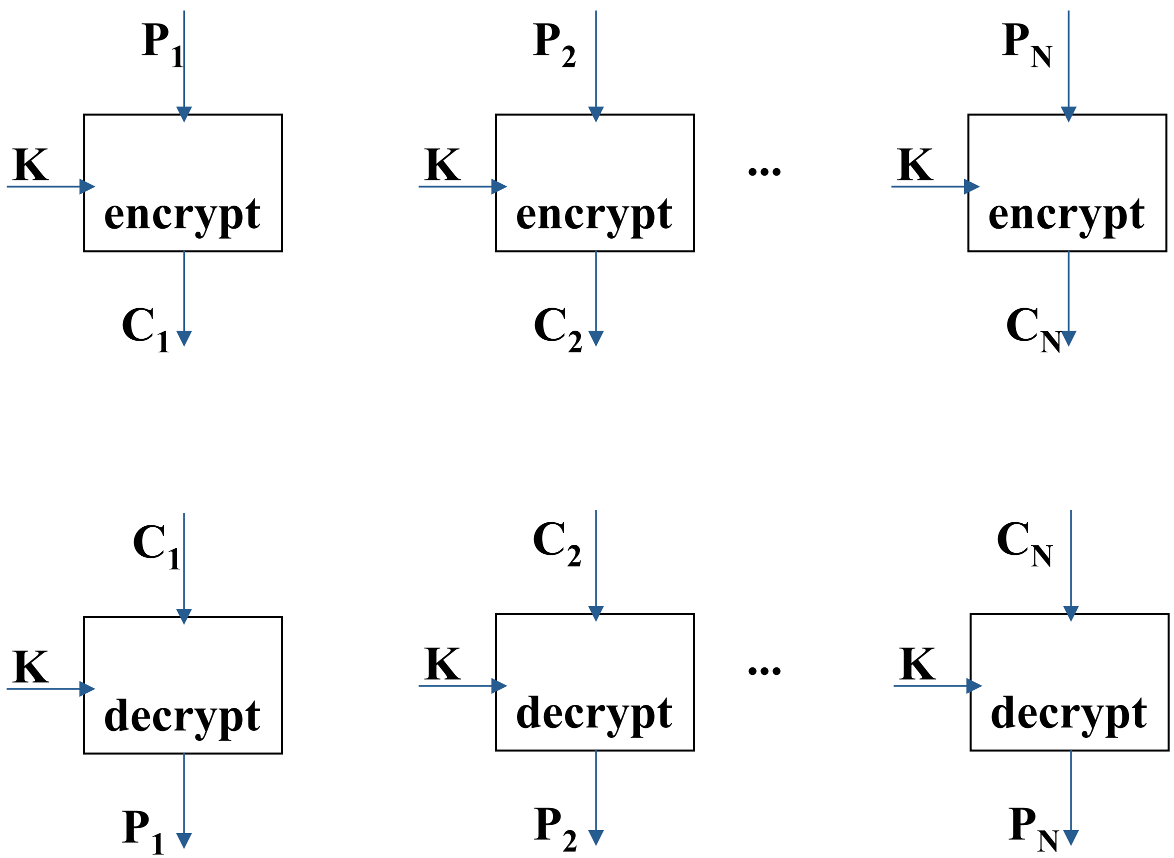
Secret Key Cryptography

Examples of Symmetric Encryption Algorithms

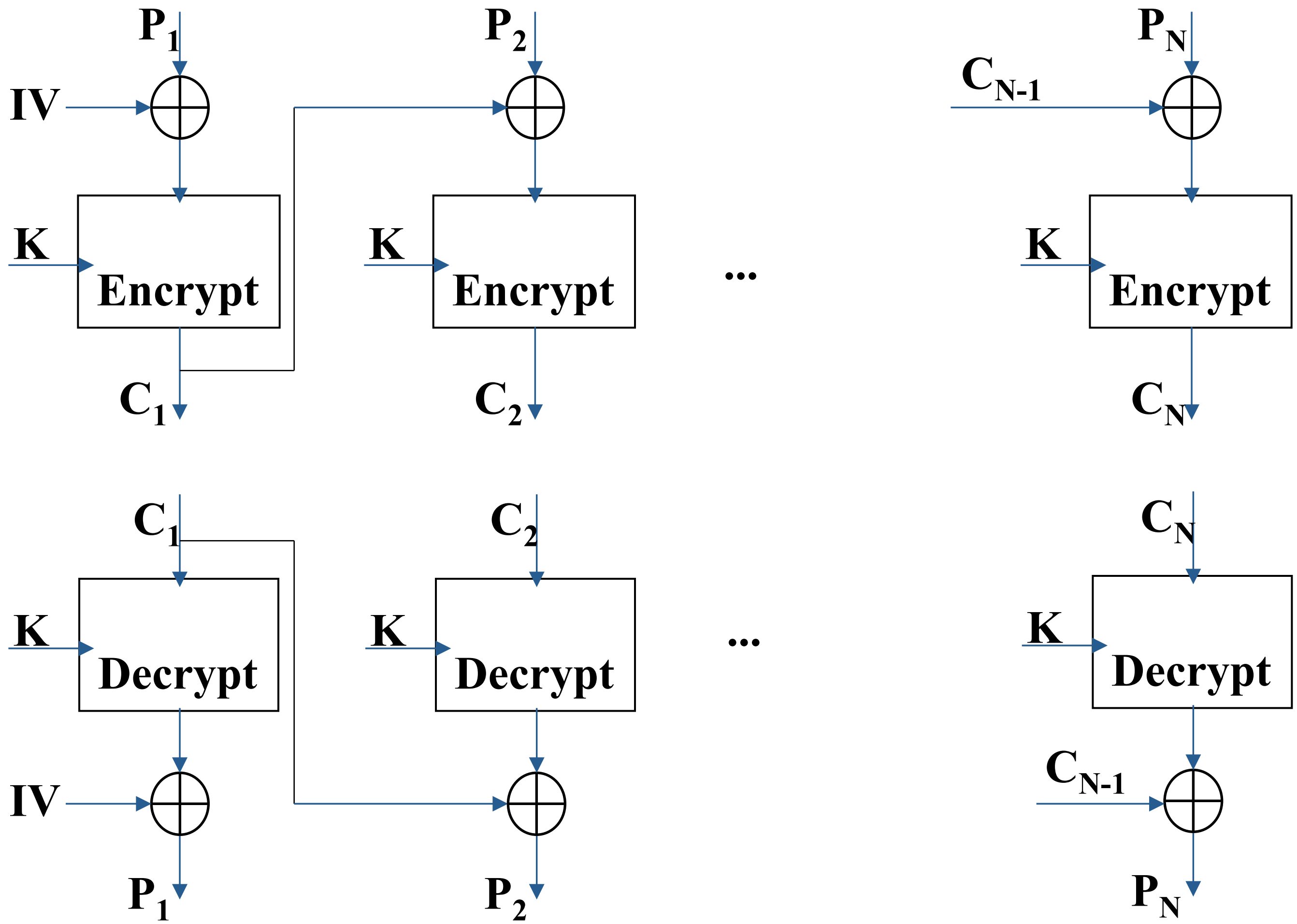
- Advances Encryption Algorithm (AES)
  - Block size: 128 bits
  - Key size:128/192/256

- Data Encryption Standard (DES) – not secure
  - Block size: 64 bits
  - Key size: 56 bits
- DES is not recommended (broken)

Encryption Modes
I. Electronic Codebook (ECB)

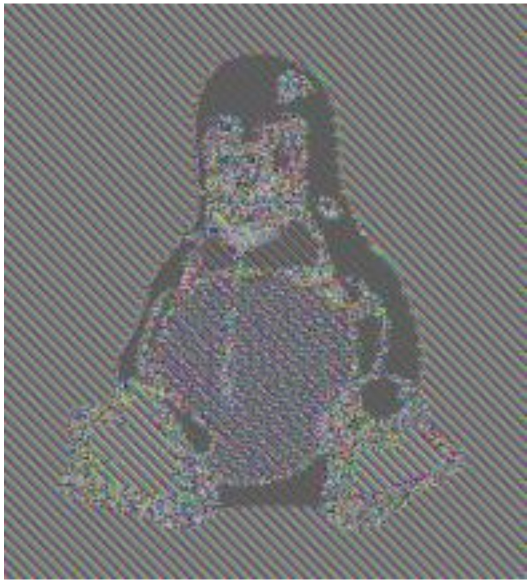


Encryption Modes:
II. Cipher Block Chaining (CBC)

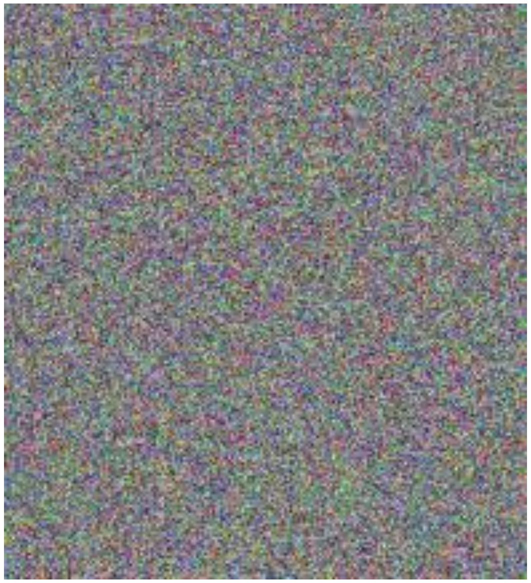


ECB vs. CBC

Plaintext



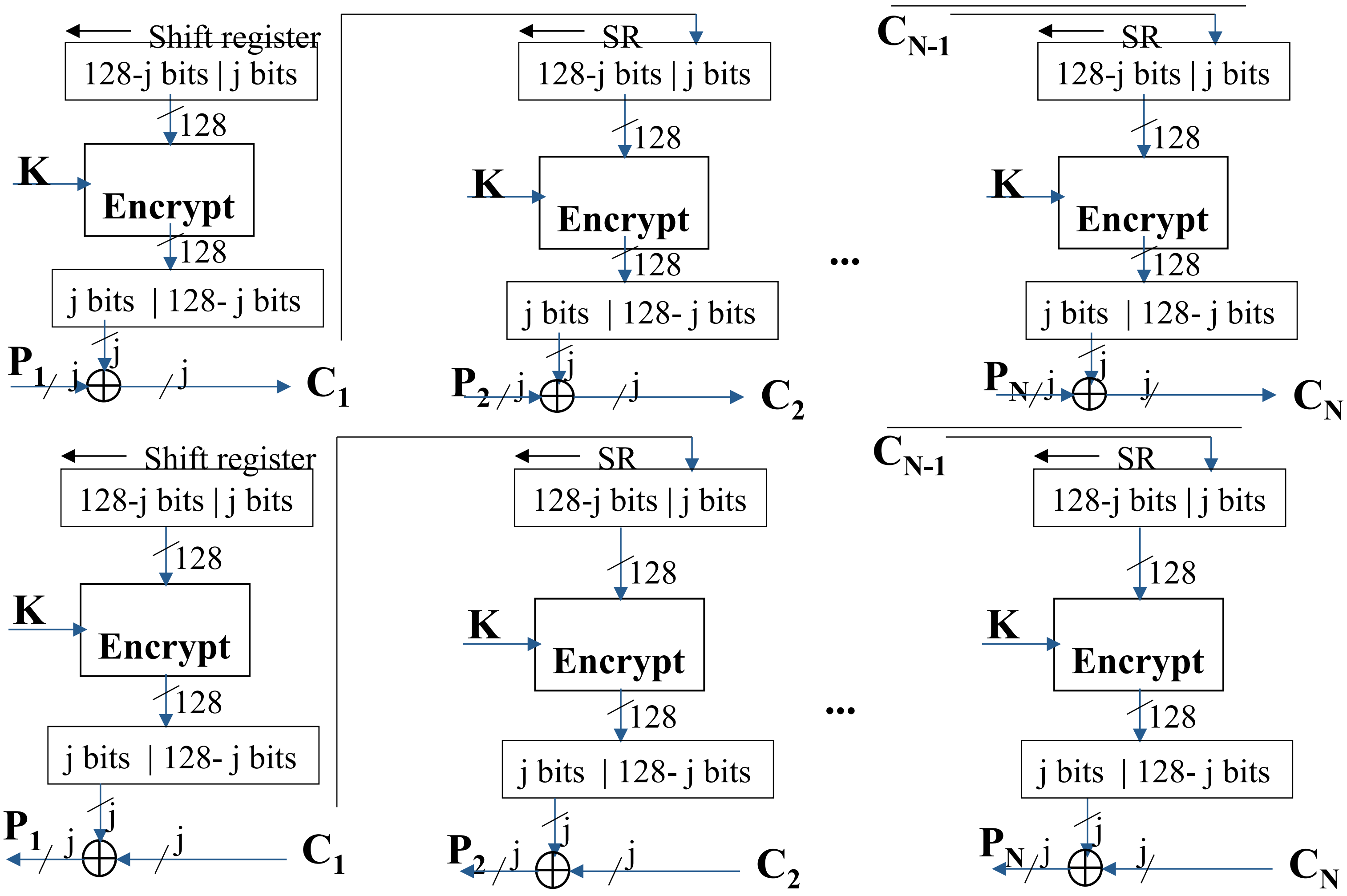
ECB Mode Encryption



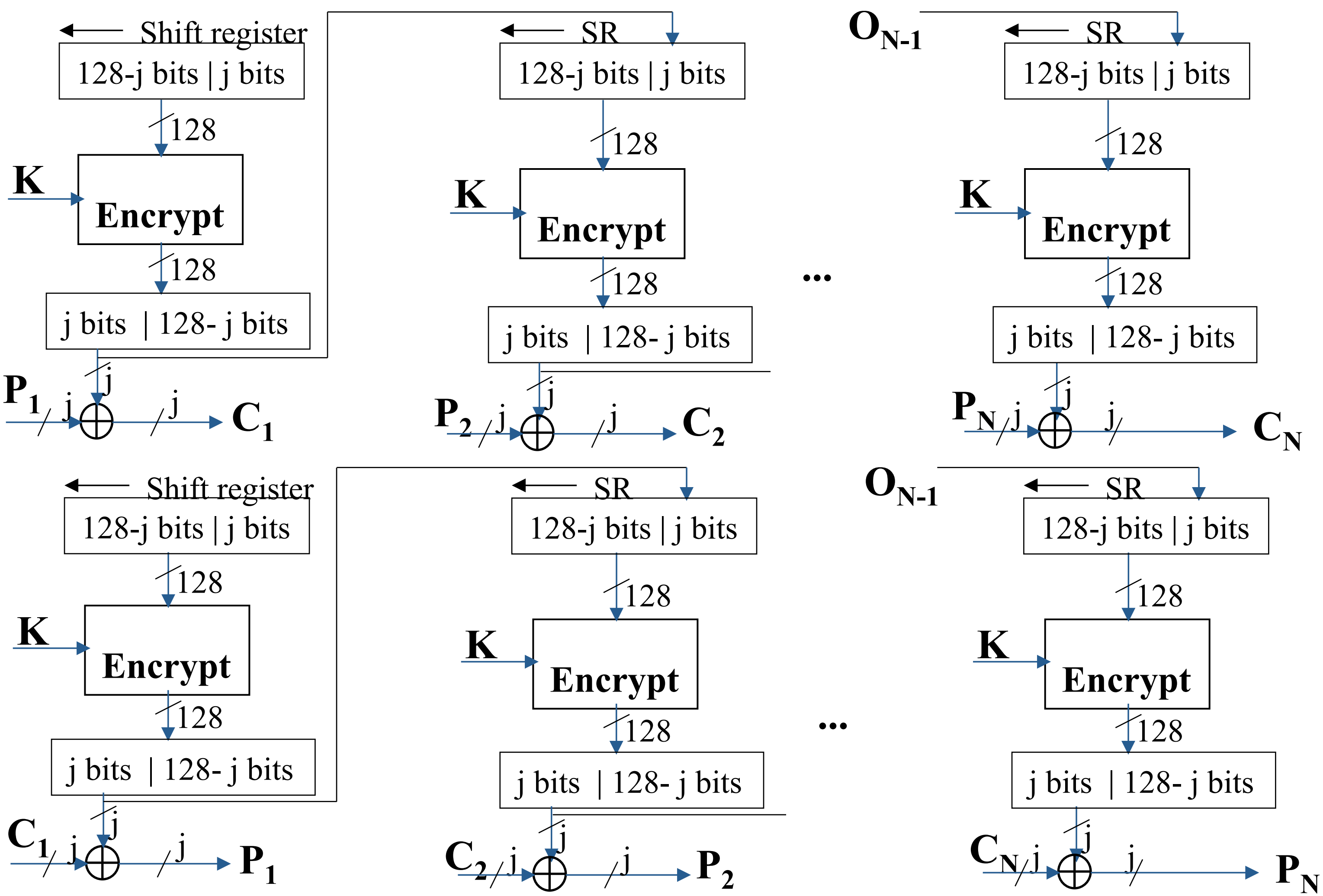
CBC Mode Encryption

Source: wikipedia

Encryption Modes:
III. Cipher Feedback (CFB)



Encryption Modes: IV. Output Feedback (OFB)



Encryption Modes:
V. Counter (CTR)

- Similar to OFB but encrypts counter value rather than any feedback value
- Must have a different key & counter value for every plaintext block (never reused)

  $O_i = \mathrm{Encrypt}_{K1}(i)$

  $C_i = P_i \text{ XOR } O_i$
- Uses: high-speed network encryptions, random access to files

Hashing Functions

Hashing Functions and Message Digests

- Goal:
  - Input: long message
  - Output: short block (called *hash* or *message digest*)
  - Desired properties:
    - Pre-image: Given a hash *h* it is computationally infeasible to find a message that produces *h*
    - Second preimage
    - Collisions
- Examples: http://www.slavasoft.com/quickhash/links.htm
  - Recommended Hash Algorithm (SHA-2, SHA-3) by NIST
  - MD2, MD4, and MD5 by Ron Rivest [RFC1319, 1320, 1321]
  - SHA-1: output 160 bits being phased out
  - SHA-2: output 224-256-384-512 believed more secure than others
  - SHA-3: winner selected official standard to be published
    http://csrc.nist.gov/groups/ST/hash/timeline.html

Birthday Attacks

- Is a 64-bit hash secure?
  - Brute force: 1ns per hash => 10^{13} seconds over 300 thousand years
- But by **Birthday Paradox** it is not
- Example: what is the probability that at least two people out of 23 have the same birthday? P > 0.5
- **Birthday attack technique**
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- Need to use larger MACs

Message Digest 5 (MD5) by R. Rivest [RFC1321]

- Input: message of arbitrary length
- Output: 128-bit hash
- Message is processed in blocks of 512 bits (padding if necessary)
- Security: not recommended
  - Designed to resist to the Birthday attack
  - Collisions where found in MD5, SHA-0, and almost found for SHA-1
  - Near-Collisions of SHA-0, Eli Biham, Rafi Chen, Proceedings of Crypto 2004, http://www.cs.technion.ac.il/~biham/publications.html
  - Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu, http://eprint.iacr.org/2004/199.pdf
  - MD5 considered harmful today: creating a rogue CA certificate, Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, December 30, 2008
  - Same attack as part of Flame malware 2012

Applications of Hashing Functions

- Authentication
- Encryption
- Message Authentication Codes

Message Authentication Code (MAC) Using an Encryption Algorithm

- Also called Message Integrity Code (MIC)
- Goal:
  - Detect any modification or forgery of the content by an attacker
- Some techniques:
  - Simple techniques have flaws
  - Use CBC mode, send only the last block (residue) along with the plaintext message
  - For confidentiality + integrity:
    - Use two keys (one for CBC encryption and one for CBC residue computation)
    - Append a cryptographic hash to the message before CBC encryption
  - Best practice technique: use a Nested MAC technique such as HMAC

HMAC

- $HMAC_K(x) = \text{SHA-1}((K \oplus \mathit{opad}) \mid \text{SHA-1}((K \oplus \mathit{ipad}) \mid x))$
  - *ipad* = 3636…36; *opad* = 5C5C…5C

- HMAC can be combined with any hashing function
- Proven to be secure under some assumptions…

Public Key Systems

Asymmetric cryptosystems

- Invented by Diffie and Hellman [DH76], and Merkle
  - When DES was proposed for standardization
- Asymmetric systems are much slower than the symmetric ones (~1000 times)
- Advantages:
  - does not require a shared key
  - simpler security architecture (no-need to a trusted third party)

Modular Arithmetic

- Modular addition:
  - E.g., $3 + 5 = 1 \bmod 7$
- Modular multiplication:
  - E.g., $3 * 4 = 5 \bmod 7$
- Modular exponentiation:
  - E.g., $3^3 = 6 \bmod 7$

- Group, Rings, Finite/Galois Fields …

Basic RSA Cryptosystem [RSA78]

- $E(M) = M^e \bmod n = C$ **(Encryption)**
- $D(C) = C^d \bmod n = M$ **(Decryption)**

- RSA parameters and basic (not secure) operations:
  - **p, q, two big prime numbers** **(private, chosen)**
  - **$n = pq$, $\phi(n) = (p-1)(q-1)$** **(public, calculated)**
  - **e, with $\gcd(\phi(n), e) = 1$, $1<e<\phi(n)$** **(public, chosen)**
  - **$d = e^{-1} \bmod \phi(n)$** **(private, calculated)**

- $D(E(M)) = M^{ed} \bmod n = M^{k\phi(n)+1} = M$ **(Euler's theorem)**

Example of RSA

- Keys generation:
  - $p = 5; q = 11 \Rightarrow n =$
  - $e = 3 \Rightarrow d = 27$
    - Because $ed = 1 \bmod (p-1)(q-1)$
  - Public key: (e, n); Private Key: (d, n)
- Encryption
  - M = 2
  - Encryption(M) = $M^e \bmod n = 8$
  - Decryption(8) = $8^d \bmod n = 2$
- Typical value $e = 2^{16}+1$, p & q 1000 bits

Prime Numbers Generation

- Density of primes (prime number theorem):
  - $\pi(x) \sim x/\ln(x)$
- Sieve of Erathostène
  - Try if any number less than SQRT(n) divides n
- Based on Fermat's Little Theorem but does not detect Carmichael numbers
  - $b^{n-1} = 1 \bmod n$ [if there exists b s.t. gcd(b, n) = 1 and $b^{n-1} \neq 1 \bmod n$ then n does not pass Fermat's test for half b's relatively prime with n]
- Solovay-Strassen primality test
  - If n is not prime at least 50% of b fail to satisfy the following:
    - $b^{(n-1)/2} = J(b, n) \bmod n$
- Rabin-Miller primality test
  - If n is not prime then it is not pseudoprime to at least 75% of $b<n$:
    - Pseudoprime: n-1 = $2^s t$, $b^t = \pm 1 \bmod n$ **OR** $b^{t2^r} = -1 \bmod n$ for some r<s
  - Probabilistic test, deterministic if the Generalized Riemann Hypothesis is true
- Deterministic polynomial time primality test [Agrawal, Kayal, Saxena'2002]

Use of RSA

- Encryption (A wants to send a message to B):
  - *A* uses the public key of *B* and encrypts *M* (i.e., $E_B(M)$)
  - Since only *B* has the private key, only *B* can decrypt M (i.e., $M = D_B(M)$)

- Digital signature (A want to send a signed message to B):
  - Based on the fact that $E_A(D_A(M)) = D_A(E_A(M))$
  - *A* encrypts *M* using its private key (i.e., $D_A(M)$) and sends it to *B*
  - *B* can check that $E_A(D_A(M)) = M$
  - Since only *A* has the decryption key, only can generate this message

Flaws in using Textbook RSA

- If message has low entropy
  - e.g., $M \in \{0, 1\} \Rightarrow$ easy to guess
  - Even if M is a random 64 bit *whp* $M = M_1 \text{ x } M_2$ *the* adversary can do a sort of meet in the middle attack

- Such potential misuse provides the rational for the design of standards for best practices in using RSA and cryptography in general

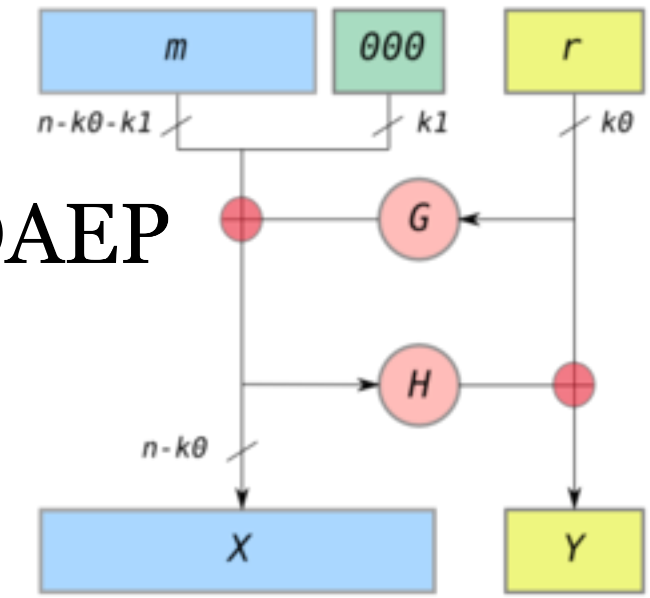
Ciphertext Indistinguishability

- Indistinguishable Chosen Plaintext Attack (IND-CPA)
  - Probabilistic asymmetric key encryption algorithm
  - Computational security
  - Adversary: probabilistic polynomial time Turing machine
- Game
  - Challenger generates a key pair *PK*, *SK* based on some security parameter k (e.g., a key size), publishes *PK*. The challenger retains *SK*
  - Adversary performs a polynomially bounded number of encryptions/operations
  - Eventually, the adversary submits two chosen plaintexts M_0, M_1 to challenger
  - Challenger selects a bit b uniformly random, and sends $C = E(PK, M_b)$ to adversary
  - The adversary is free to perform additional computations or encryptions.
  - Finally, it outputs a guess for the value of b.
- Scheme is IND-CPA secure if $|\text{Prob[guessing } b] - ½| < \varepsilon(k)$ [negligible]

- Similar definition for symmetric key encryption algorithms using oracles

Optimal Asymmetric Encryption Padding (OAEP)

- Use of RSA is standardized by several PKCS public key crypto standards
- PKCS #1 v2 (RFC2437) uses OAEP



When combined with secure trapdoor one-way permutation is proven semantically secure under IND-CPA in Random Oracle model

Keys Establishment

Diffie-Hellman Key Exchange

Private: A	Public	Private: B
x	p: prime number, g: primitive element of GF(p)	y
compute: $g^x \bmod p$		compute: $g^y \bmod p$
receive: $g^y \bmod p$		receive: $g^x \bmod p$
Compute shared key: $(g^y)^x \bmod p$	$K = g^{xy} \bmod p$	Compute shared key: $(g^x)^y \bmod p$

- Based on the difficulty of computing discrete logarithms
- Works also in extension Galois fields: $GF(p^q)$

Attack on Diffie-Hellman Scheme: Public Key Integrity

Man-in-the-Middle Attack

A **x** — **I (intruder)** **z** — **B** **y**

A → I: g^x

I → B: g^z

I → A: g^z

B → I: g^y

Shared key: $K_{AI} = g^{xz}$

Shared key: $K_{BI} = g^{yz}$

Message encrypted using K_{AI} (A → I)

Decrypt using K_{AI} **+Decrypt using** K_{BI} (I → B)

- Need for a mean to verify the public information: certification

Random Number Generation (RNG)

- RNG is a critical building block of security services
- Cryptographic RNG need to be computationally unguessable by an adversary and are quite different from RNG for simulations
- Blum Blum Shub 1986
  - $x_{n+1} = x_n^2 \bmod M$ where $M = pq$ the product of 2 large primes both congruent to 3 mod 4
  - x_0 co-prime with M
  - $r_i = \text{LSB}(x_i)$
  - Computationally reduces to the quadratic residue problem
  - Cons: too slow
- Rivest RNG
  - $r_i = \text{LSB}(\text{SHA-256}(\textit{secret-seed} \mid i))$

Building Network Security Services

- Confidentiality:
  - Use an encryption algorithm
  - Generally an symmetric algorithm for a stream of data
- Integrity:
  - MAC algorithm
- Access control:
  - Use access control tables
- Authentication
  - Use authentication protocols
- Non-repudiation
  - Digital signatures

Some Examples

- Email
  - PGP or S/MIME: basic use of crypto
    - Beware your mail client might be storing drafts on the server!
  - Anti-spam: Hashcash, DKIM
- DNSSEC, SSH
- Cryptocurrency: Bitcoin
- TLS/SSL
  - https, VPN, WPA-Enterprise, Tor, Hidden Services

Anti-Spam

- Current solutions:
  - Black/white listing IP addresses (e.g., zombie computers, addresses that sent spam to honeypots, ISP willingly hosting spammers)
  - Signatures/content matching rules
  - Distributed Checksum Clearinghouse: message fuzzy checksum is sent to DCC to check how many times it appeared
  - Sender Policy Framework: specify who can send email from a domain (relies on TXT/SPF DNS record)

    dig @8.8.8.8 neu.edu ANY

  - HashCash: add header

    Example: X-Hashcash: 1:20:101130:noubir@ccs.neu.edu::HdG5s/(oiuU7Ht7b:ePa+tr5

    The counter ePa+tr5 is found such that the hash of the X-Hashcash header has its first 20 bits = 0

    This information is found using brute force

    X-Hashcash constrains the destination email address and date => proof of work protects against spam replays

    ver:*bits*:*date*:*resource*:[*ext*]:*rand*:*counter*

    - *ver* = 1
    - *bits* = how many bits of partial-preimage the stamp is claimed to have
    - *date* = YYMMDD[hhmm[ss]]
    - *resource* = resource string (eg IP address, email address)
    - *ext* = extension -- ignored in the current version
  - Example of software combining these techniques: spamassassin

Sender MTA Authentication

- DomainKeys Identified Mail (DKIM RFC 4871, 2007 – RFC 6376, 2011)
  - DomainKeys initiated by Yahoo!, today a IETF standard DKIM
- The sending MTA adds a signature to the message
  - MIME header
  - Public key can be retrieved through DNS system
    dig @8.8.8.8 s1024._domainkey.yahoo.com any
    dig @8.8.8.8 gamma._domainkey.gmail.com any
- Example:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=gmail.com; s=gamma;
        h=domainkey-signature:mime-version:received:received:date:message-id
         :subject:from:to:content-type;
        bh=cvC34ODyPB/uEHubbDQQmwxZfqZboGjW5gpY4W6DuzE=;
        b=ASsElEtXCmM/x3aL38Efnvi9xDrBdleaaBqd24f7XS49pRzhXK/7Vak9+LyLLcN89e
         GZ7SZi7swY2xIlt3zJTiGrGif0bfQdf7LvlP12g53nczhBBRa8McBVtdK9+ImAZByg8o
         oEM4INNjMvdhXi9MVXtntkvmsTmWitAJxZgQQ=
DomainKey-Signature: a=rsa-sha1; c=nofws;
        d=gmail.com; s=gamma;
        h=mime-version:date:message-id:subject:from:to:content-type;
        b=JFWiE0YlmWxu+Sq4OJ9Ef5k3rjbZQ51dGEyaFyvKJYR8NkoGrNoPIUq5f29ld8P0AD
         Lg058evTVeuWxvfPQfa7K65J9AjEQt5U8d9zBKFfxRAz1h5nr7k2kCLRMnhbqVTkiOIS
         OUfxIQeMfgbYz0ydCgerEnfGreKMQIYax+dpo=
```

Misuse of the Basics

- Crypto libraries are widely available
- Developers still lack knowledge of crypto basics
- Default black-box use leads to vulnerabilities

Analysis of Android Apps

- Android SSL support can lead to the following
  - Trusting all certificates no matter who signed them
  - Accepting a certificate for an arbitrary different domain
  - 1,074 potentially vulnerable apps to MITM
  - 41 out 100 selected for manual verification are vulnerable: 39M – 185M users

[FHMSBF'12] "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security" CCS'2012.

- Misuse of Android Crypto Service Providers (15K Apps)
  - 5,656: ECB (BouncyCastle default)
  - 3,644: Constant symmetric key
  - 2,000: ECB (Explicit use)
  - 1,932: Uses constant IV
  - 1,636: Used iteration count < 1,000 for PBE
  - 1,629: Seeds SecureRandom with static data
  - 1,574: Uses static salt for PBE

[EBFK CCS'13] "An Empirical Study of Cryptographic Misuse in Android Applications" CCS'2013.

Adobe Breach (October 2013)

- Passwords encrypted with 64 bits 3DES in ECB
  - Not hashed, not salted, not in CBC, not AES

Password data (hex)				Password hint
0b4c27d8f75cc41a			->	Same old, same old
e826ef87cc7a3029	e2a311ba09ab4707		->	You'll never guess
0842ccb7edf3e343	e2a311ba09ab4707		->	
92663700893c3f27	a667d747891a8255		->	Dog + digit
88fc540356d561ec			->	Dog
fb0a9047a5dd5ef8	f3c512b0e38a5392	a3f492fbd917f632	->	Virtuously long
92bb535704f0ae7f			->	Geburtestag

Pwd data length	Count (logarithmic scale)
8	461016
16	538396
24	526
32	53
40	6
48	3

Source: Naked Security

Adobe Breach (October 2013)

- ECB, no salting

⇒ same password results in the same hash

⇒ combining the hints makes he guesses easy

Adobe password data		Password hint	
110edf2294fb8bf4	->	numbers 123456	❶ 123456
110edf2294fb8bf4	->	==123456	
110edf2294fb8bf4	->	c'est "123456"	
8fda7e1f0b56593f e2a311ba09ab4707	->	numbers	❷ 12345678
8fda7e1f0b56593f e2a311ba09ab4707	->	1-8	
8fda7e1f0b56593f e2a311ba09ab4707	->	8digit	
2fca9b003de39778 e2a311ba09ab4707	->	the password is password	❸ password
2fca9b003de39778 e2a311ba09ab4707	->	password	
2fca9b003de39778 e2a311ba09ab4707	->	rhymes with assword	
e5d8efed9088db0b	->	q w e r t y	❹ qwerty
e5d8efed9088db0b	->	ytrewq tagurpidi	
e5d8efed9088db0b	->	6 long qwert	
ecba98cca55eabc2	->	sixxone	❺ 111111
ecba98cca55eabc2	->	1*6	
ecba98cca55eabc2	->	sixones	

Weak Pseudo-Random Number Generators

- Out or 4.7 million distinct 1024-bit RSA 12,720 have a shared prime

- Many embedded devices

[LHABK] "Ron was wrong, Whit is right", IACR, 2012.

TLS/SSL

- A closer look at the popular TLS/SSL
- Overview
- Vulnerabilities
  - Design, integration, implementation

General Description of SSL/TLS

- Terminology:
  - SSL: Secure Socket Layer
  - TLS: Transport Layer Security
- Concept: secure connections on top of TCP
  - OS independent
  - TCP instead of UDP
    - Cons: Rogue packet problem
    - Pro: SSL/TLS doesn't have to deal with packet retransmission
- History:
  - SSLv2 proposed and deployed in Netscape 1.1 (1995)
  - PCT (Private Communications Technology) by Microsoft
  - SSLv3: (1995)
  - TLS proposed by the IETF based on SSLv3 but not compatible (1996)
    - Uses patent free DH and DSS instead of RSA which patent didn't expire yet
  - TLS 1.2 (2008)
    - Updated in 2011 does not allow SSLv2

SSL Architecture

- There is a **Client** and a **Server**
- **SSL session**
  - An association between client & server
  - Created by the Handshake Protocol
  - Defines a set of cryptographic parameters
  - May be shared by multiple SSL connections
- **SSL connection**
  - A transient, peer-to-peer, communications link
  - Associated with 1 SSL session

SSL/TLS Basic Protocol

- Basic Protocol:
  - A -> B: I want to talk, ciphers I support, R_A
  - B -> A: certificates, cipher I choose, R_B
  - A -> B: $\{S\}_B$, {keyed hash of handshake msgs}
  - B -> A: {keyed hash of handshake msgs}
  - A <-> B: data encrypted and integrity checked with keys derived from K
  - Keyed hashes use $K = f(S, R_A, R_B)$

- SSL/TLS partitions TCP byte stream into records:
  - A record has: header, cryptographic protection => provides a reliable encrypted, and integrity protected stream of octet
  - Record types:
    - Handshake messages
    - Change cipher spec
    - Application data
    - Alerts: error messages or notification of connection closure

SSL/TLS Basic Protocol (Cont'd)

- How do you make sure that keyed hash in message 3 is different from *B*'s response?
  - Include a constant *CLNT/client finished* (in SSL/TLS) for *A* and *SRVR/server finished* for *B*
- Keyed hash is sent encrypted and integrity protected
  - Not necessary
- Keys: derived by hashing K and R_A and R_B
  - 3 keys in each direction: encryption, integrity and IV
  - Write keys (to send: encrypt, integrity protect)
  - Read keys (to receive: decrypt, integrity check)

What's still missing?

- SSL/TLS allowed to authenticate the server
- How would the server authenticate the user?
  - SSL/TLS allows clients to authenticate using certificates:
    - *B* requests a certificate in message 2
    - *A* sends: certificate, signature of hash of the handshake messages

Session Resumption

- Many secure connections can be derived from the session
  - Cheap: how?
- Session initiation: modify message 2
  - *B* -> *A*: session_id, certificate, cipher, R_B
- *A* and *B* remember: (session_id, master key)
- To resume a session: *A* presents the session_id in message 1
  - *A* -> *B*: session_id, ciphers I support, R_A
  - *B* -> *A*: session_id, cipher I choose, R_B, {keyed hash of handshake msgs}
  - *A* -> *B*: {keyed hash of handshake msgs}
  - *A* <-> *B*: data encrypted and integrity checked with keys derived from *K*

Computing the Keys

- S: pre-master secret (forget it after establishing K)
- $K = f(S, R_A, R_B)$
- 6 keys = $g_i(K, R_A, R_B)$
- Rs: 32 bytes (usually the first 4 bytes are Unix time)

PKI in SSL

- Client comes configured with a list of “trusted organizations”: CA
- What happens when the server sends its certificate?
- When the server whishes to authenticate the client
  – Server sends a list of CA it trusts and types of keys it can handle
- In SSLv3 and TLS a chain of certificates can be sent

Negotiating Cipher Suites

- A cipher suite is a complete package:
  - (encryption algorithm, key length, integrity checksum algorithm, etc.)
- Cipher suites are predefined:
  - Each assigned a unique value (contrast with IKE)
  - SSLv2: 3 bytes, SSLv3: 2 bytes => upto 65000 combinations
    - 30 defined,
    - 256 reserved for private use: FFxx (risk of non-interoperability)
- Selection decision:
  - In v3 A proposes, B chooses
  - In v2 A proposes, B returns acceptable choices, and A chooses
- Suite names examples:
  - SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
  - SSL2_RC4_128_WITH_MD5

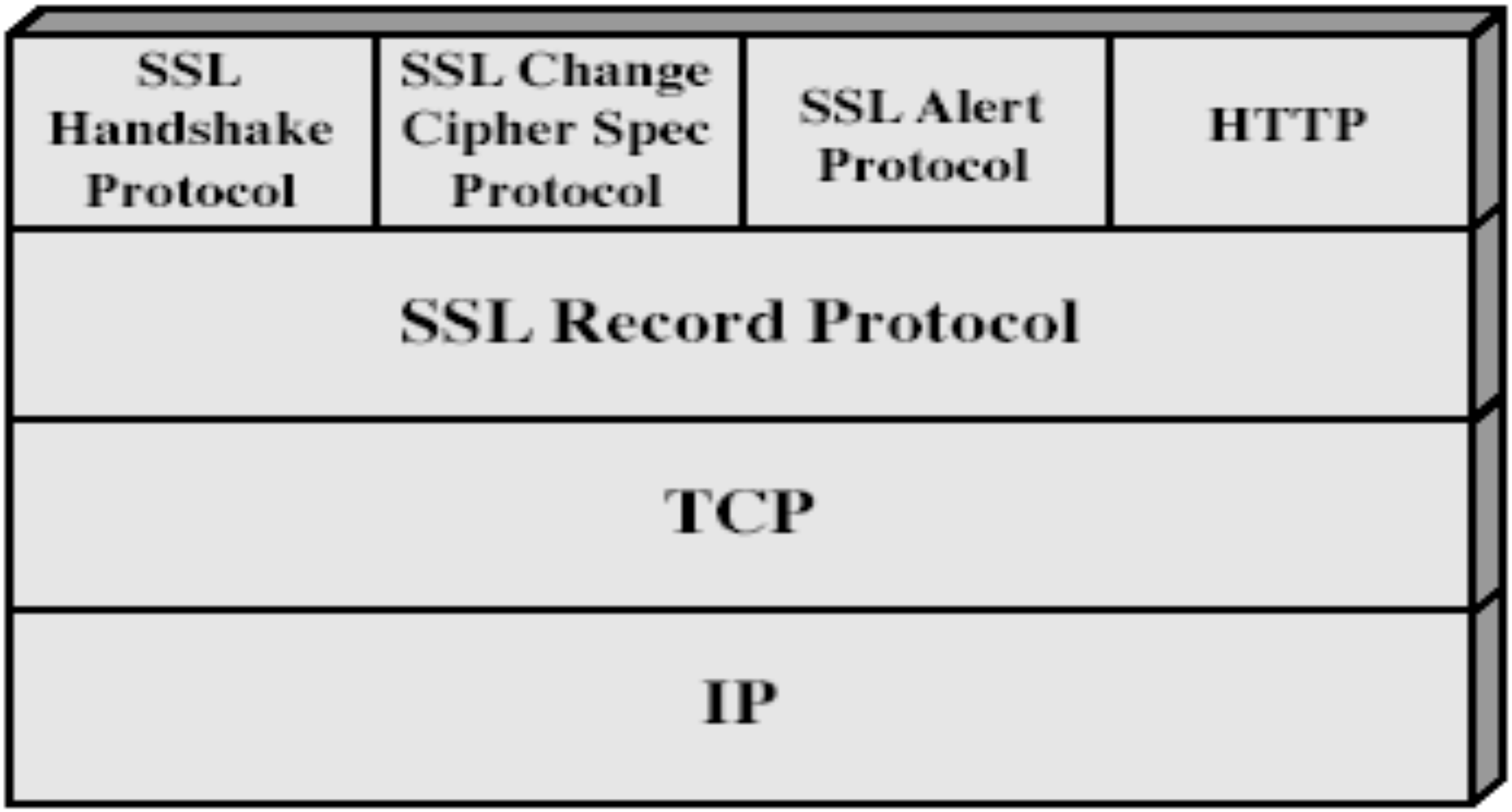
Attacks fixed in v3

- Downgrade attack:
  - In SSLv2 there is no integrity protection for the initial handshake
  - Active attacker can remove strong crypto algorithm from proposed cipher suite by *A* => forcing *A* and *B* to agree on a weak cipher
  - Fixed by adding a *finished* message containing a hash of previous messages

- Truncation attack:
  - Without the *finished* message an attacker can send a TCP FIN message and close the connection without communicating nodes detecting it

- Attacks not fixed: session renegotiation, BEAST, CRIME/ BREACH…

SSL/TLS Detailed Protocol SSL Stack



SSL Record Protocol

- SSL Record Protocol defines these two services for SSL connections:
  - **Confidentiality**
    - Using symmetric encryption with a shared secret key defined by Handshake Protocol
    - AES, IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
    - CBC mode (except for RC4)
    - Message is compressed before encryption
  - **Message integrity**
    - Using a MAC with shared secret key
    - Based on HMAC and MD5 or SHA (with a padding difference due to a typo in an early draft of HMAC RFC2104)
- Records sent after *ChangeCipherSpec* record are cryptographically protected
- Record header:
  - [record type, version number, length]
    - ChangeCipherSpec = 20, Alert = 21, Handshake = 22, Application_data = 23

SSL Change Cipher Spec Protocol

- One of 3 SSL-specific protocols which use the SSL Record Protocol
- Single message
  - Causes pending state to become current

  $\Rightarrow$ all records following this will be protected with the ciphers agreed upon

SSL Alert Protocol

- Conveys SSL-related alerts to peer entity
- Severity
    - warning or fatal
- Specific alerts
    - Unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
    - Close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- Compressed & encrypted

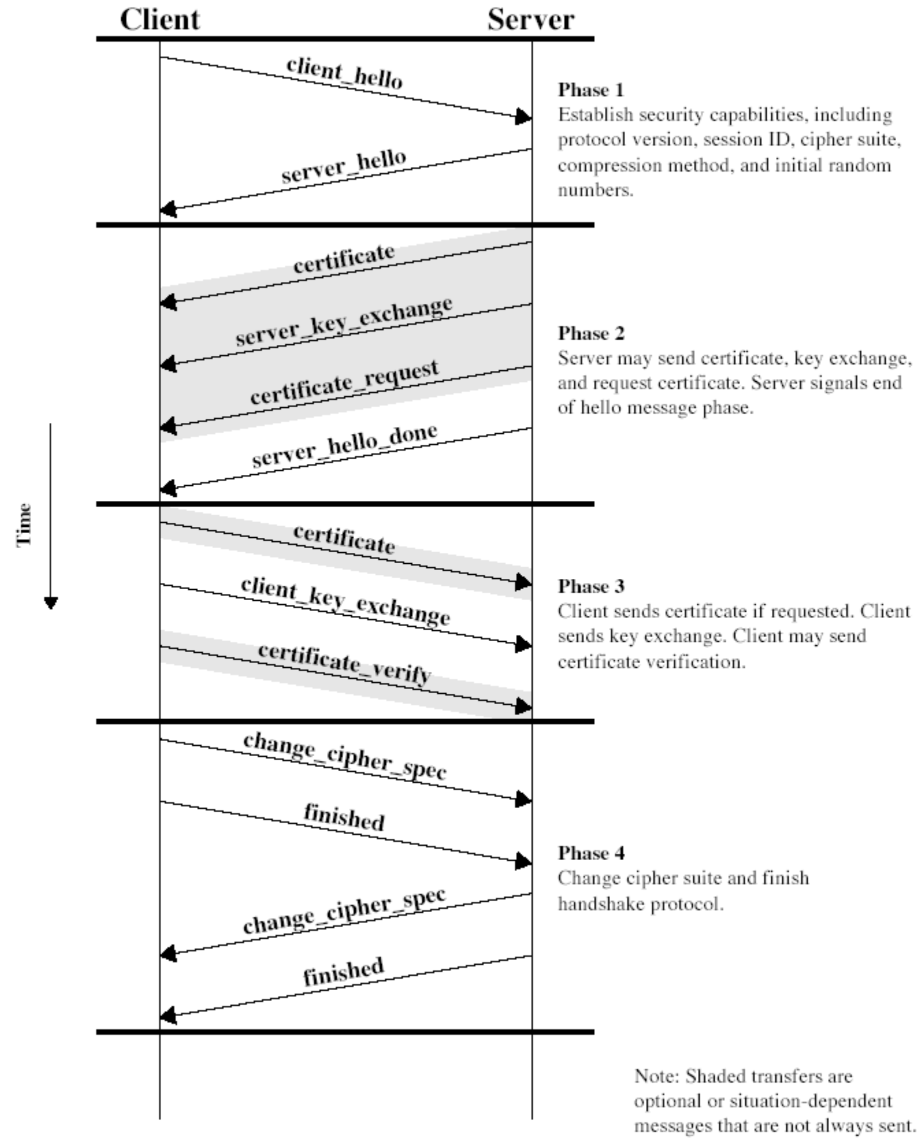
SSL Handshake Protocol

- Allows server & client to:
  - Authenticate each other
  - Negotiate encryption & MAC algorithms
  - Negotiate cryptographic keys to be used
- Comprises a series of messages in phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish

Handshake Messages

- ***ClientHello* message:**
  - [type=1, length, version number, R_A, length of session_id, session_id, length of cipher suite list, sequence of cipher suites, list of compression methods]
- ***ServerHello*:** [type=2, length, version number, R_B, length of session_id, session_id, chosen cipher, chosen compression method]
- ***Certificate*:** [type=11, length, length of first certificate, first certificate, …]
- ***ServerKeyExchange*:** (for export: ephemeral public key)
  - [type=12, length, length of modulus, modulus, length of exponent, exponent]
- ***CertificateRequest*:** [type=13, length, length of key type list, list of types of keys, length of CA name list, length of first CA name, 1stCA name, …]
- ***ServerHelloDone*:** [type=14, length=0]
- ***ClientKeyExchange*:** [type=16, length, encrypted pre-master secret]
- ***CertificateVerify***:[type=15, length, length of signature, signature]
- ***HandshakeFinished***:[type=20, length=36 (SSL) or 12 (TLS), digest]

SSL Handshake Protocol



Exportability Issues

- Exportable suites in SSLv2:
  - 40 secret bits out of 128 in symmetric keys
  - 512-bits RSA keys
- Exportability in SSLv3:
  - Integrity keys computed the same way
  - Encryption keys: 40 bits secret
  - IV non-secret
  - When a domestic server (e.g., 1024-bit RSA key) communicates with an external client the server creates an ephemeral key of 512-bits and signs it with it's 1024-bit key

TLS (Transport Layer Security)

- TLS is and IETF standard similar to SSLv3
  - RFC 2246, RFC 4346, and RFC 5246
- Minor differences
  - Record format version number
  - HMAC for MAC
  - Pseudo-random function to expand the secrets
  - Additional alert codes
  - Changes in supported ciphers
  - Changes in certificate negotiations
  - Changes in use of padding

Session Renegotiation Flaw/Attack (2009)

- The adversary carries a MITM

```
Client                        Attacker                          Server
------                        --------                          ------
                                  <----------- Handshake ------------>
                                  <======= Initial Traffic ==========>
<-------------------------- Handshake ================================>
<======================== Client Traffic ============================>
```

- Initial traffic:

```
GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1
X-Ignore-This:
```

  Note no: CR LF

- Client traffic

```
GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1
Cookie: victimscookie
```

- Server sees:

```
GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1
X-Ignore-This: GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1
Cookie: victimscookie
```

OS X (2014)

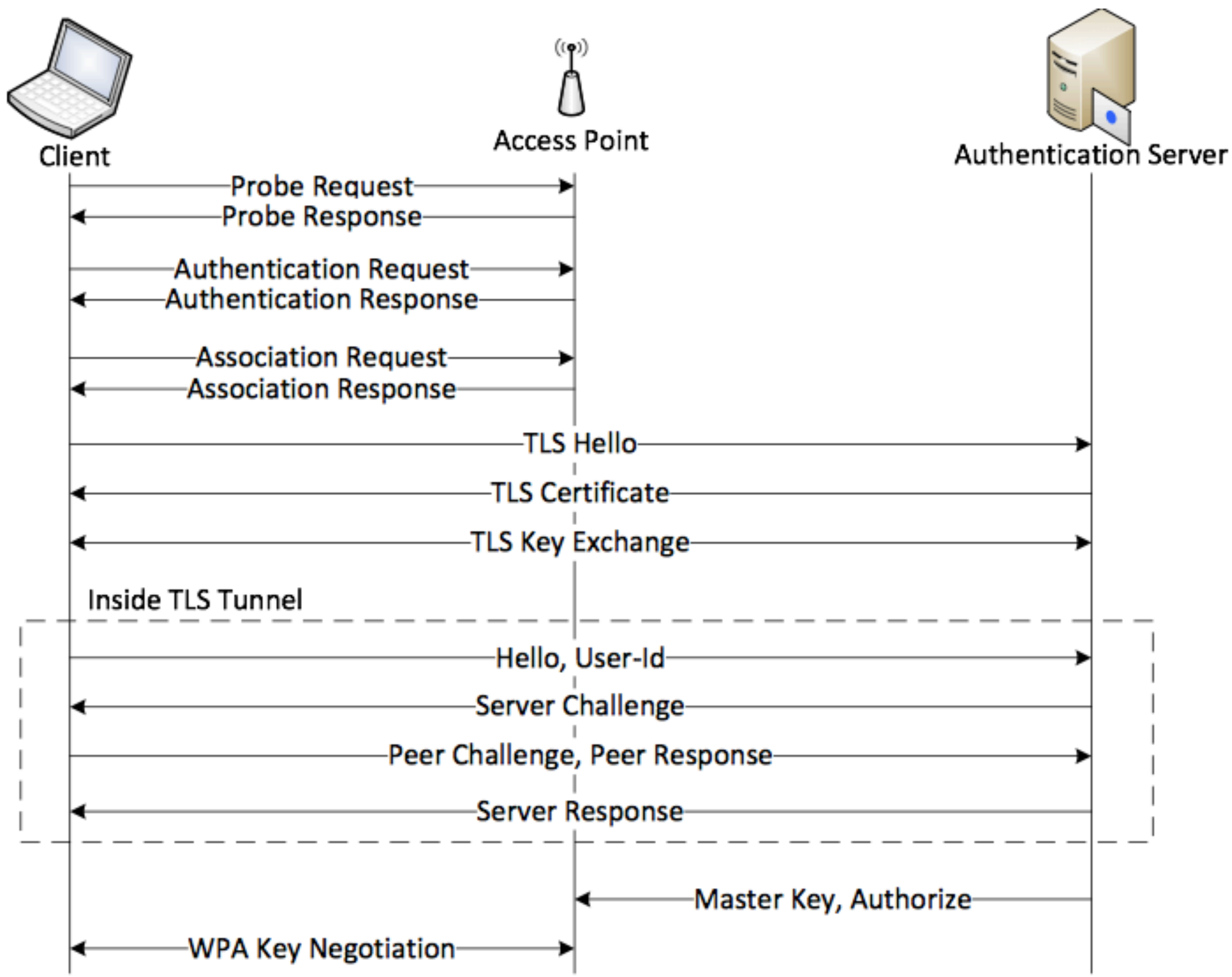
```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
(…)
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

  err = sslRawVerify(ctx,
                       ctx->peerPubKey,
                       dataToSign,        /* plaintext */
                       dataToSignLen,     /* plaintext length */
                       signature,
                       signatureLen);
  if(err) {
    sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                "returned %d\n", (int)err);
    goto fail;
  }

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

Other Attacks

- BEAST (2011)
  - Attack on CBC mode by re-injecting IVs…
- CRIME/BREACH
  - Attack on compression when combined with
- Require attacker to be on the routing path
  - e.g., controls Access Point
- Heartbleed (2014)
  - Implementation

- Check:
  https://www.trustworthyinternet.org/ssl-pulse/

WPA-Enterprise Attacks [CKRN'12]

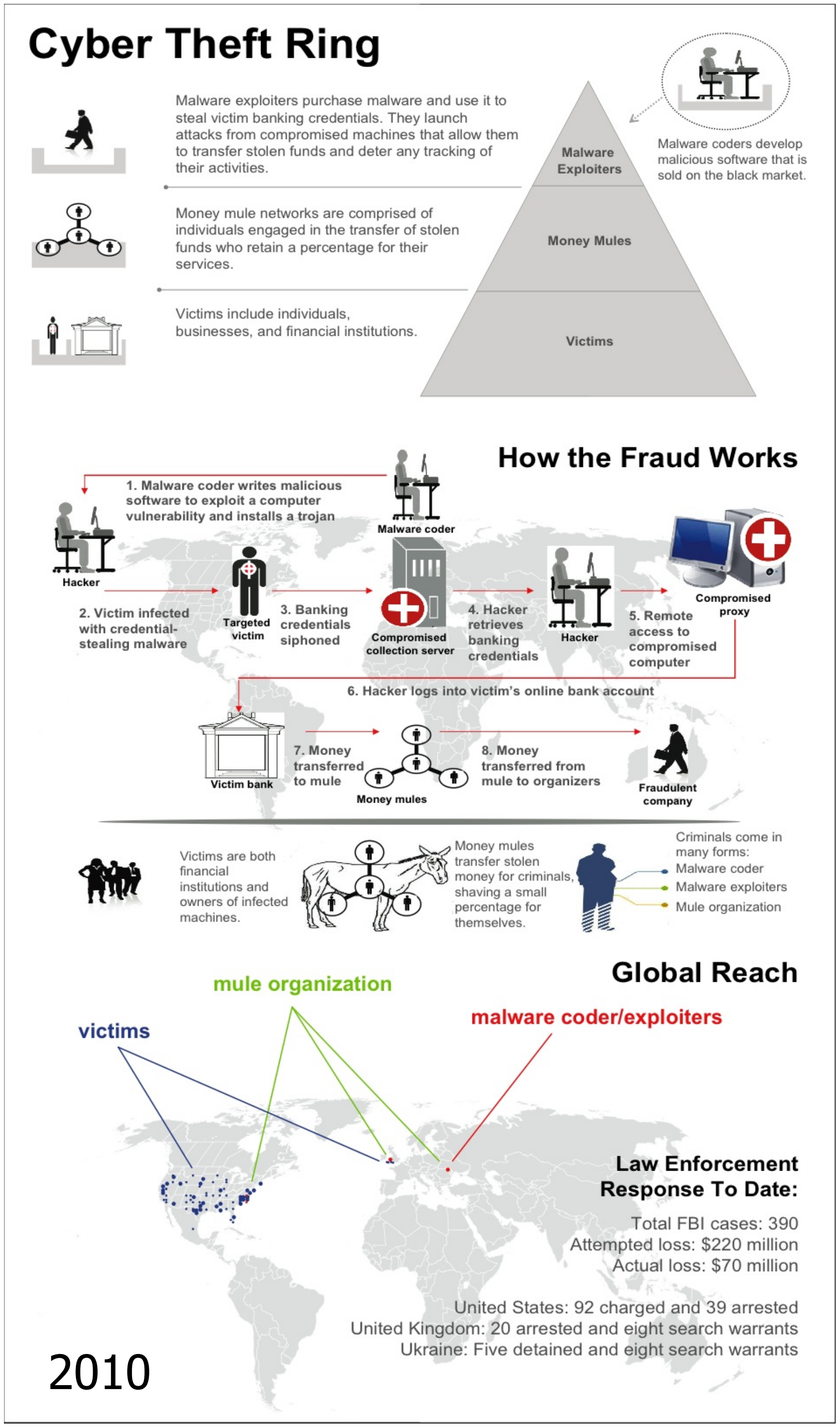


Worms: Buffer Overflow to Crypto-Based

- Popularized by R. Morris 1988, re-emerged in late 90s - ~2003 mostly DoS
  - Code Red CRv1 (7/13/2001), Code Red CRv2 (7/19/2001), Code Red II (8/4/2001), Nimbda (9/18/2001), …
- MS SQL Slammer
  - Date January 25, 2003
  - Buffer overflow in MS SQL Server
  - Doubled every 8.5 seconds until network collapse
  - 90% of vulnerable hosts infected in 10 minutes (75,000)
- Helpful worms: Welchia/Nachia worm (installs patches)
- Check: http://en.wikipedia.org/wiki/Timeline_of_notable_computer_viruses_and_worms
- Where did all the worms go?
  - Stealthy, instrumented for financial benefits, cyber-crime, cyber-warfare targeted attacks
  - Conficker A, B, C, D, E: since November 2008 infected 9-15 million hosts
  - In 2009, PandaLabs analyzed 2M machines and found 6% infected
  - Stuxnet, FLAME (2009 – 2012 see next slides)
  - In 2013: Cryptolocker encrypts the files on a user's hard drive, and asks for a ransom

Zeus

- Trojan horse (2007 -)
  - Steals banking information
  - Man-in-the-browser keystroke logging and Form Grabbing
  - Spreads through drive-by downloads, phishing
  - 3.6M infected in the US
- Used sophisticated scheme to funnel stolen money to exploiters through mules
  - More recently: Bitcoin, MoneyPak
- New versions using Tor HS



Stuxnet

- Stuxnet is a computer worm with unique characteristics
  - Time frame 2009-2010?
- Targets specific SCADA systems
  - Supervisory Control and Data Acquisition systems
  - Control industrial systems such as power plants
- Stuxnets spreads slowly searching for specific SCADA systems and reprograms their PLC

How does it operate?

- Stuxnet uses 4 zero-day attacks as infection vectors + other bugs
  - USB drive, print spooler, two elevation of privilege bugs
- Spreads slowly (to max three nodes)
- When spreading over the network remains local to the company
- Looks for a MS Windows machine with
  - WinCC/PCS 7 Siemens Software that controls PLC
  - Checks for Variable Frequency Drives (AC rotational speed controllers)
  - Focuses on two vendors (Vacon & Fararo Paya)
  - Attacks systems that run between 807-1210Hz
  - Modifies the output frequency for a short interval of time to 1410Hz and then to 2Hz and then to 1064Hz
- Tries default/hardcoded passwords
- Hides existence by installing malicious drivers signed using two stolen keys (Realtek, JMicron)
- 60% damage believed to be in Iran
- Variants: Duqu similar to Stuxnet but with different purpose

- Seems there was another variant that started in 2007 (stealthier, replays recorded physical process, propagates through contractors)

FLAME

- Perceived goal: cyber-espionage in middle east
  - Time frame 2010 – 2012?
  - Targets MS Windows: screenshots, network traffic, records audio/keyboard, skype calls, bluetooth beaconing
  - http://www.crysys.hu/skywiper/skywiper.pdf

- Similar to stuxnet but more sophisticated
  - Size: 20MB
  - Propagates through LAN or USB stick
  - Stealthy: identifies which anti-virus is used and avoids it e.g., changing files extensions
  - 5 encryption algorithms
  - Used a fraudulent MD5-based certificate similar to rogue CA technique

Remarks

- Security is about the whole system
- Software vulnerabilities are still a major issue
- Crypto-based solutions are replacing ad hoc solutions
- Public Key Infrastructure and deployment is weak
- Network architecture not designed with sufficient security
- Human factor, users, passwords, policies
- SCADA system are vulnerable and critical
- Attacks are becoming more sophisticated and targeted

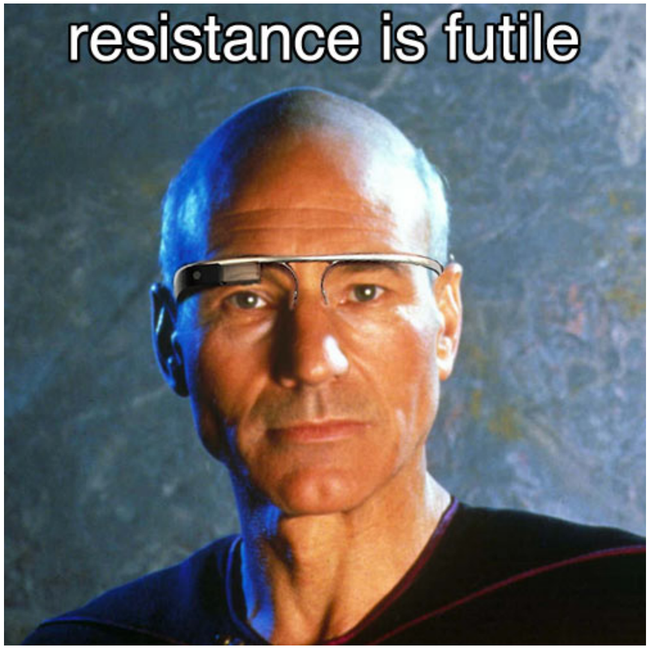
Privacy in the Age of Big Data

Ubiquitous Computing for Free?

- Ubiquitous computing is a reality beyond expectations
  - Access information anywhere, anytime, any type
- Sensing devices
  - Phones, cameras, routers, access points, ebook readers, set top boxes, CCTV, game consoles, websites
- Platforms for computing, storage, and communication, seemingly for FREE
  - email, search, instant messaging, telephony, social networking, documents, spreadsheets, rich content sharing
  - Unprecedented ease of use (confusing privacy implications)

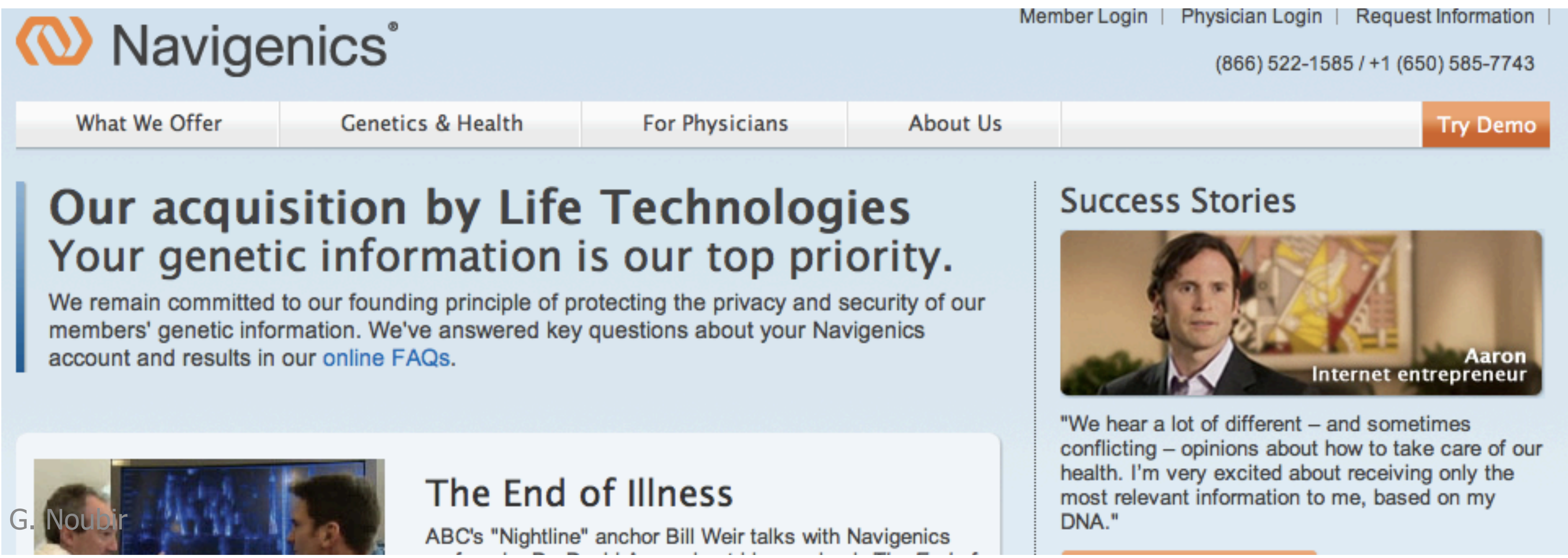
How Pervasive?

- Everything read, write, see, eat, where, when, to, from, how long
  - Project glass periodic pictures capture



How Pervasive?

- Who you are -- free whole genome



Big Data

- Ever decreasing cost of storage

- Storing massive amounts of users information cross-analyzing
  - Implicit logging/tracking
  - Explicit pictures/movies uploads
  - E.g., on March 1st 2012, google started aggregating information about users across all its platforms

Free Services?

- “Free” services are paid for someway
- Targeted advertisement
- Brokering for services
- Temptation is high to ignore users privacy concerns
  - Skype, Google, Facebook

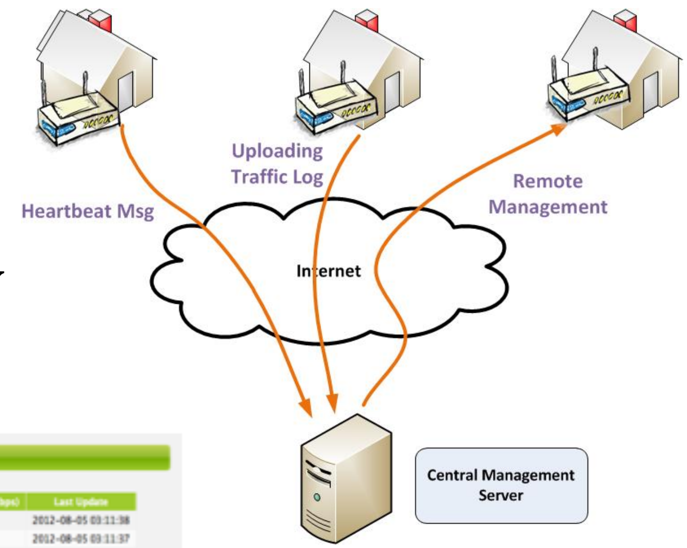
Scalable and Secure Wireless and Data Access

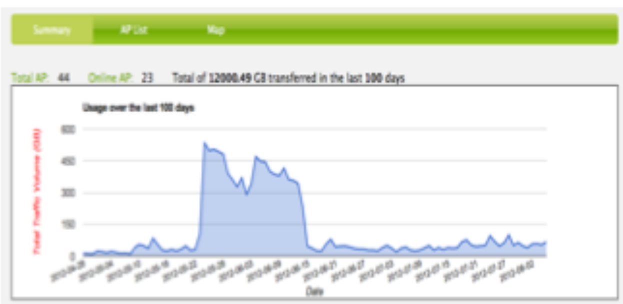
- Leverage the high density of APs to provide
  - Scalable ubiquitous access: WiZi-cloud, BaPu, SNEAP
  - Scalable storage

- Challenges
  - Privacy: anonymity, unlinkability
  - Robustness: DoS, blackmailing
  - Incentives: payment, credit

Open Infrastructure Testbed

- 30 home WiFi APs in Boston (since 02/2011)
- Customized OpenWRT firmware
- 16GB USB Flash
- A suite of management tools
- Traffic monitoring at 10sec granularity
- 1.3TB full data trace (6 month)



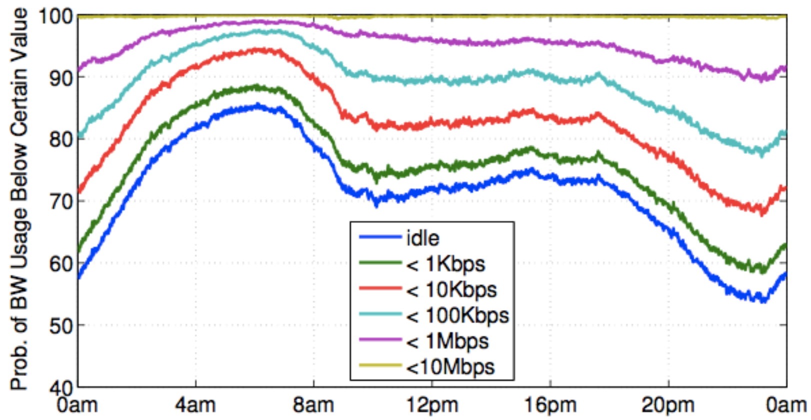


Summary | AP List | Map

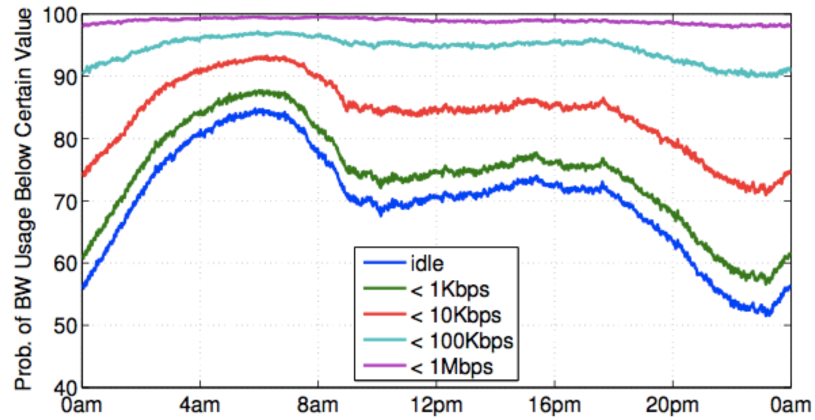
	version	IP	uptime (hr)	WiFi ESSID	PriBW (Kbps)	GuestBW (Kbps)	Last Update
•	0.63	129.10.115.200	3028.82		0.66	0.00	2012-08-05 03:11:38
•	0.63	65.96.165.130	1946.94		0.59	0.00	2012-08-05 03:11:37
•	0.63	71.232.32.247	1.22		10.49	0.00	2012-08-05 03:11:41
	0.61	129.10.115.200	0.04		0.00	0.00	2012-07-19 18:20:25
•	0.63	24.63.24.189	4117.74		0.59	0.00	2012-08-05 03:11:37
•	0.61	174.62.207.20	471.97		0.23	0.00	2012-08-05 03:11:39
	0.6	209.6.232.79	47.44		0.00	0.00	2012-04-12 19:41:07
•	0.63	76.175.169.116	773.54		10.30	0.00	2012-08-05 03:11:34
•	0.63	24.34.221.134	1434.77		0.80	0.00	2012-08-05 03:11:39
	0.63	24.147.69.225	4523.30		2086.77	0.00	2012-05-27 09:24:04
•	0.63	75.67.17.113	777.22		0.47	0.00	2012-08-05 03:11:42
	0.6	24.218.216.22	0.24		0.00	0.00	2012-02-26 16:12:48

OpenWrt Wireless Freedom

Testbed Measurement Findings



(a) Downlink Bandwidth Usage During 24 Hours



(b) Uplink Bandwidth Usage During 24 Hours

- Residential broadband is mostly under utilized
  - Over 90% chance, DL bw. < 1Mbps, UL bw. < 100Kbps
- WiFi APs generally have good connectivity to Internet
  - inter-ISP, intra-ISP, ISP to major public servers
  - Latency: 24ms
- Wardriving in Boston (Dec. 2011) to verify our findings in a large scale
  - 26K APs
  - Instrumented latency measurements with hundreds of them

Conclusions

- Cryptographic provides powerful mechanisms and is becoming ubiquitous in systems and Apps
- Misuse Challenges
  - Lack of basic understanding of building blocks
  - Unsafe defaults
  - Security libraries should be better scrutinized
- Crypto an enabled of future cybercrime
  - Tor/HS + Bitcoin: Cryptolocker, silk road
  - How to prevent criminal misuse?
- Privacy in the Era of Big Data
  - Cryptography can play a key role: privacy-preserving services