

Security

Wenyuan Xu
(reference: [Vitaly Shmatikov](#))

Department of Computer Science and Engineering
University of South Carolina

2008

CSCE790: Security and Privacy for Emerging Ubiquitous
Communication system

Outline

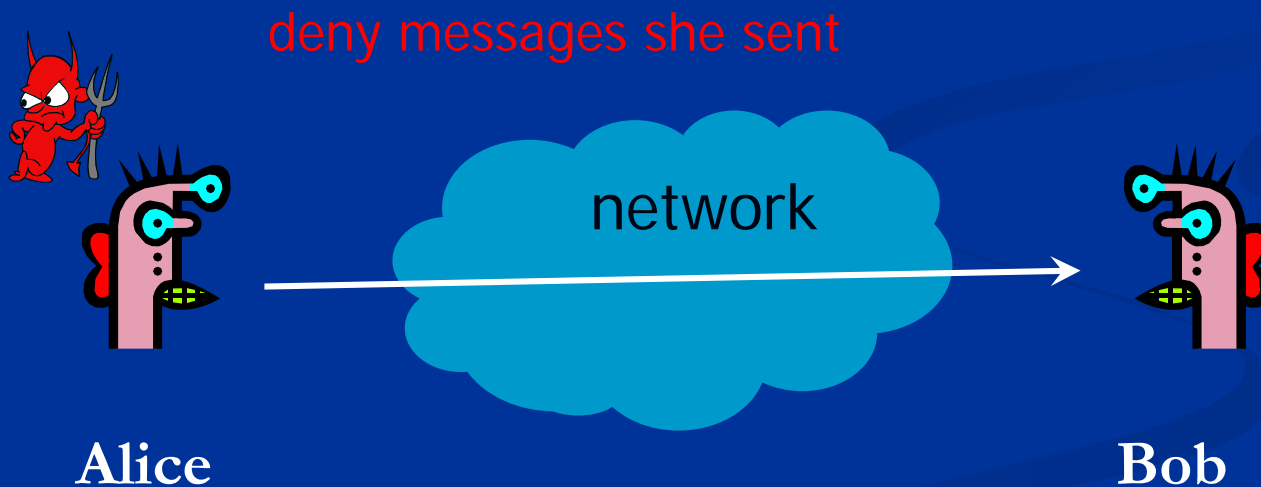
- Security objectives
- Basics of cryptography
 - Symmetric key algorithm
 - Public-key algorithms
 - Cryptographic hash functions

Security Objectives

- Confidentiality
- Integrity
- Authenticity
- Availability
- Accountability and non-repudiation
- Privacy of collected information

Accountability and non-repudiation

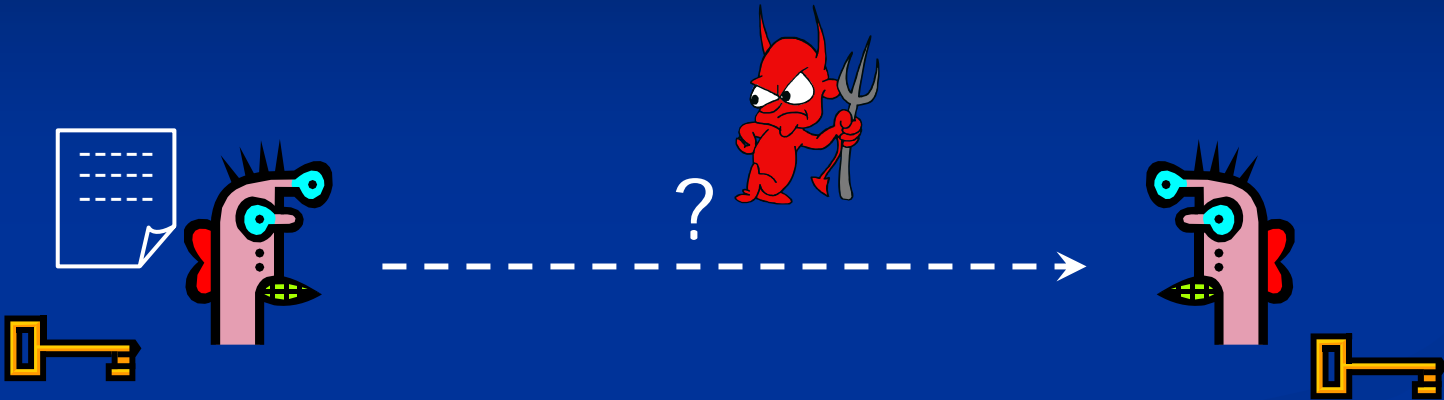
- Non-repudiation:
 - Alice cannot claim that she did not send the message.



Basics of cryptography

Symmetric key algorithm

Symmetric Encryption



Given: both parties already know the same secret

Goal: send a message confidentially

How is this achieved in practice?

Any communication system that aims to guarantee confidentiality must solve this problem

Symmetric Encryption

- Classical Cryptosystems:
 - Shift Ciphers
 - Affine Ciphers
 - The Vigenere Cipher
 - One-time Pad
- Modern Symmetric key ciphers
 - DES: Data Encryption Standard
 - AES: Rijndael

Block Cipher Principles

- Operates on a single chunk (“block”) of plaintext
 - For example, 64 bits for DES, 128 bits for AES
 - Same key is reused for each block (can use short keys)
- Result should look like a random permutation
 - “As if” plaintext bits were randomly shuffled
 - Output should be “statistically” uncorrelated with the key and input message
- Only computational guarantee of secrecy
 - Not impossible to break, just very expensive
 - If there is no efficient algorithm (unproven assumption!), then can only break by brute-force, try-every-possible-key search
 - Time and cost of breaking the cipher exceed the value and/or useful lifetime of protected information

Permutation



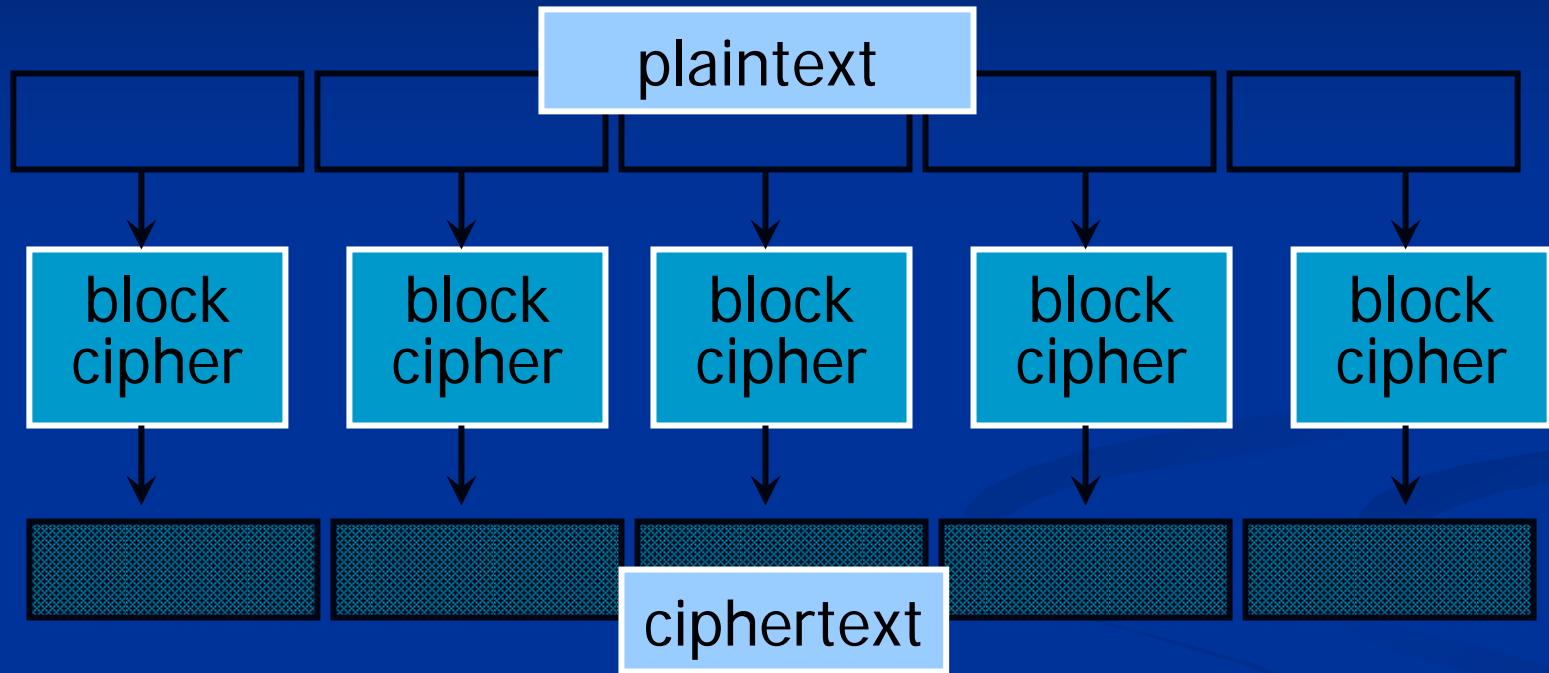
CODE becomes DCEO

- For N-bit input, $N!$ possible permutations
- Idea: split plaintext into blocks, for each block use secret key to pick a permutation, rinse and repeat
 - Without the key, permutation should “look random”

Encrypting a Large Message

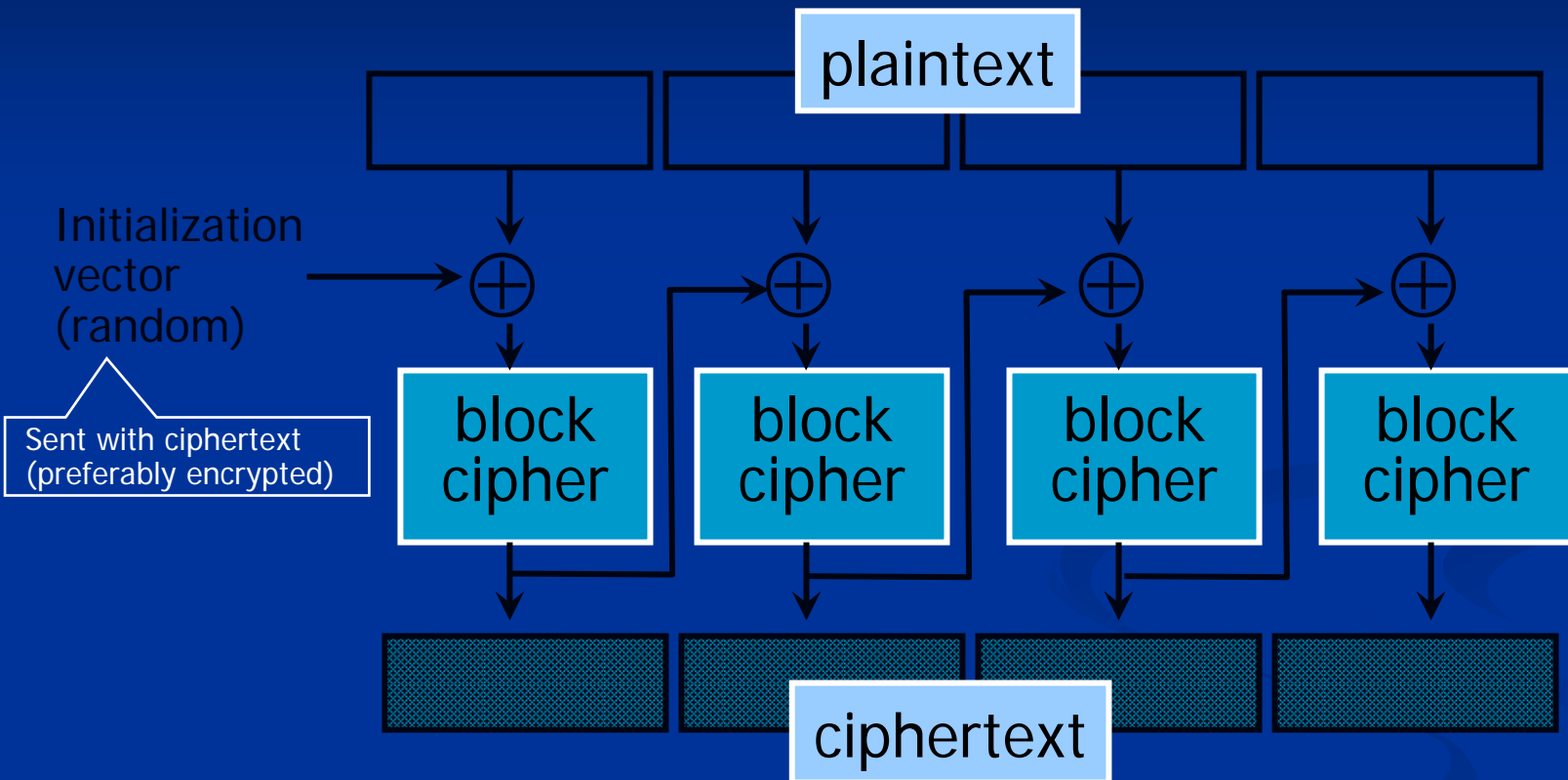
- So, we've got a good block cipher, but our plaintext is larger than 128-bit block size
- Electronic Code Book (ECB) mode
 - Split plaintext into blocks, encrypt each one separately using the block cipher
- Cipher Block Chaining (CBC) mode
 - Split plaintext into blocks, XOR each block with the result of encrypting previous blocks
- Also various counter modes, feedback modes, etc.

ECB Mode



- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

CBC Mode: Encryption

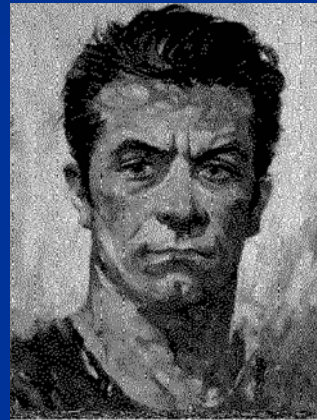


- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
 - Still does not guarantee integrity

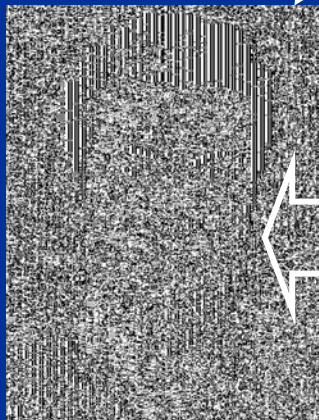
ECB vs. CBC

[Picture due to Bart Preneel]

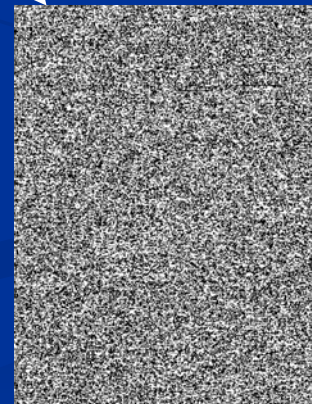
AES in ECB mode



AES in CBC mode



Similar plaintext blocks produce similar ciphertext blocks (not good!)



Applications of symmetric crypto

- Confidentiality?
- Authenticity?
- Availability?
- Accountability and non-repudiation?
- Integrity?

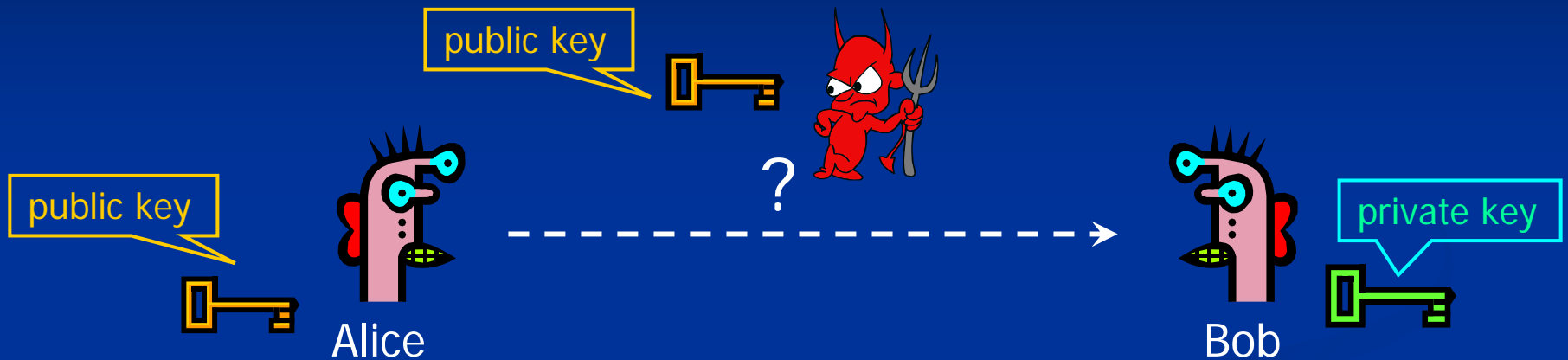
Integrity vs. Confidentiality

- **Integrity:** attacker cannot tamper with message
- Encryption may not guarantee integrity!
 - Intuition: attacker may be able to modify message under encryption without learning what it is
 - Given one-time key K , encrypt M as $M \oplus K$... Perfect secrecy, but can easily change M under encryption to $M \oplus M'$ for any M'
 - Online auction: halve competitor's bid without learning its value (How?)
 - Some encryption schemes provide confidentiality AND integrity

Basics of cryptography

Asymmetric key algorithm

Asymmetric Cryptograph



Given: Everybody knows Bob's **public key**

- How is this achieved in practice?

Only Bob knows the corresponding **private key**

- Goals:
1. Alice wants to send a secret message to Bob
 2. Bob wants to authenticate himself

Applications of Public-Key Crypto

- Encryption for confidentiality
 - Anyone can encrypt a message
 - With symmetric crypto, must know secret key to encrypt
 - Only someone who knows private key can decrypt
 - Key management is simpler (maybe)
 - Secret is stored only at one site: good for open environments
- Digital signatures for authentication
 - Can “sign” a message with your private key
- Session key establishment
 - Exchange messages to create a secret session key
 - Then switch to symmetric cryptography (why?)

Public-Key Encryption

- Key generation: computationally easy to generate a pair (public key PK, private key SK)
 - Computationally infeasible to determine private key SK given only public key PK
- Encryption: given plaintext M and public key PK, easy to compute ciphertext $C = E_{PK}(M)$
- Decryption: given ciphertext $C = E_{PK}(M)$ and private key SK, easy to compute plaintext M
 - Infeasible to compute M from C without SK
 - Trapdoor function: $Decrypt(SK, Encrypt(PK, M)) = M$

Some Number Theory Facts

- Euler totient function $\varphi(n)$ where $n \geq 1$ is the number of integers in the $[1, n]$ interval that are relatively prime to n
 - Two numbers are relatively prime if their greatest common divisor (gcd) is 1
- Euler's theorem:
if $a \in \mathbb{Z}_n^*$, then $a^{\varphi(n)} = 1 \pmod n$
- Special case: Fermat's Little Theorem
if p is prime and $\gcd(a, p) = 1$, then $a^{p-1} = 1 \pmod p$

RSA Cryptosystem

[Rivest, Shamir, Adleman 1977]

- Key generation:
 - Generate large primes p, q
 - Say, 1024 bits each (need primality testing, too)
 - Compute $n=pq$ and $\phi(n)=(p-1)(q-1)$
 - Choose small e , relatively prime to $\phi(n)$
 - Typically, $e=3$ (may be vulnerable) or $e=2^{16}+1=65537$ (why?)
 - Compute unique d such that $ed = 1 \pmod{\phi(n)}$
 - Public key = (e,n) ; private key = d
- Encryption of m : $c = m^e \pmod n$
 - Modular exponentiation by repeated squaring
- Decryption of c : $c^d \pmod n = (m^e)^d \pmod n = m$

RSA Example

1. Select primes: $p=885320963$ & $q=238855417$
2. Compute $n = pq = 211463707796206571$
3. Compute $\phi(n) = (p-1)(q-1)$
4. Select $e : \gcd(e, \phi(n)) = 1$; choose $e=9007$
5. Determine $d: de=1 \pmod{\phi(n)}$ and $d < \phi(n)$.
Value is $d=116402471153538991$

Example encryption

1. Suppose $m=30120$
2. Ciphertext $c=m^e \pmod{n} = (30120)^{9007} = 113535859035722866$
3. Reconstruct plaintext: $m=c^d \pmod{n}$
 $= 113535859035722866^{116402471153538991}$
 $= 30120$

Why RSA Decryption Works

- $e \cdot d = 1 \pmod{\phi(n)}$
- Thus $e \cdot d = 1 + k \cdot \phi(n) = 1 + k(p-1)(q-1)$ for some k
- If $\gcd(m, p) = 1$, then $m^{ed} = m \pmod{p}$
 - By Fermat's Little Theorem, $m^{p-1} = 1 \pmod{p}$
 - Raise both sides to the power $k(q-1)$ and multiply by m
 - $m^{1+k(p-1)(q-1)} = m \pmod{p}$, thus $m^{ed} = m \pmod{p}$
 - By the same argument, $m^{ed} = m \pmod{q}$
- Since p and q are distinct primes and $p \cdot q = n$,
 $m^{ed} = m \pmod{n}$

Why Is RSA Secure?

- **RSA problem:** given $n=pq$, e such that $\gcd(e, (p-1)(q-1))=1$ and c , find m such that $m^e = c \pmod n$
 - i.e., recover m from ciphertext c and public key (n, e) by taking e^{th} root of c
 - There is no known efficient algorithm for doing this
- **Factoring problem:** given positive integer n , find primes p_1, \dots, p_k such that $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$
- If factoring is easy, then RSA problem is easy, but there is no known reduction from factoring to RSA
 - It may be possible to break RSA without factoring n

Applications of asymmetric crypto

- Confidentiality?
- Authenticity?
- Availability?
- Accountability and non-repudiation?
- Integrity?

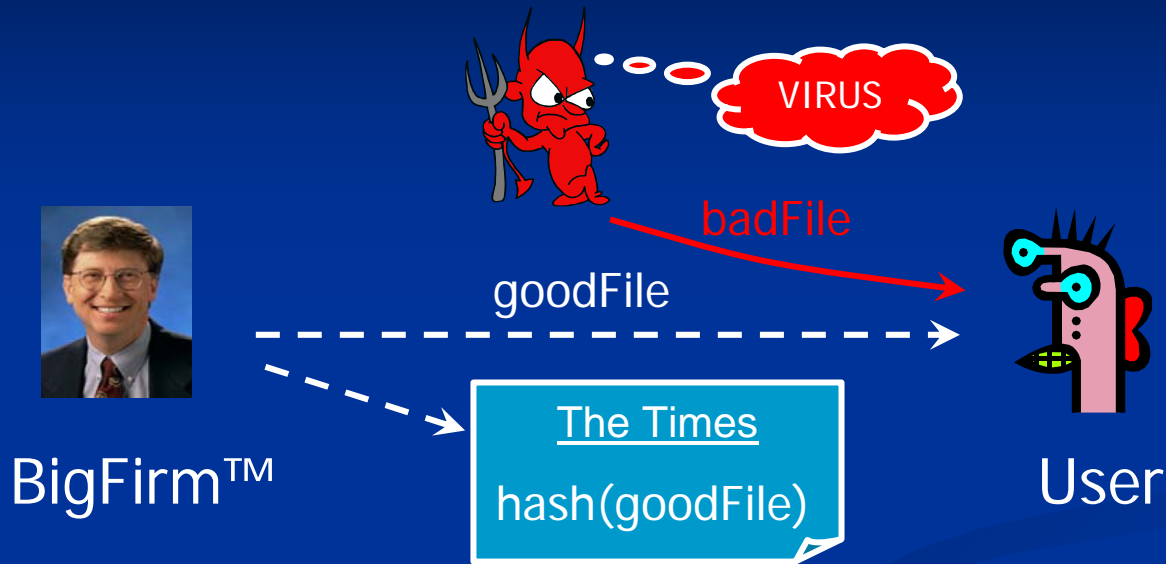
Integrity in RSA Encryption

- Plain RSA does not provide integrity
 - Given encryptions of m_1 and m_2 , attacker can create encryption of $m_1 \cdot m_2$
 - $(m_1^e) \cdot (m_2^e) \bmod n = (m_1 \cdot m_2)^e \bmod n$
 - Attacker can convert m into m^k without decrypting
 - $(m^e)^k \bmod n = (m^k)^e \bmod n$
- In practice, OAEP (Optimal Asymmetric Encryption Padding) is used: instead of encrypting M , encrypt $M \oplus G(r) ; r \oplus H(M \oplus G(r))$
 - r is random and fresh, G and H are hash functions
 - Resulting encryption is plaintext-aware: infeasible to compute a valid encryption without knowing plaintext
 - ... if hash functions are “good” and RSA problem is hard

Basics of cryptography

Hash Function

More on Integrity



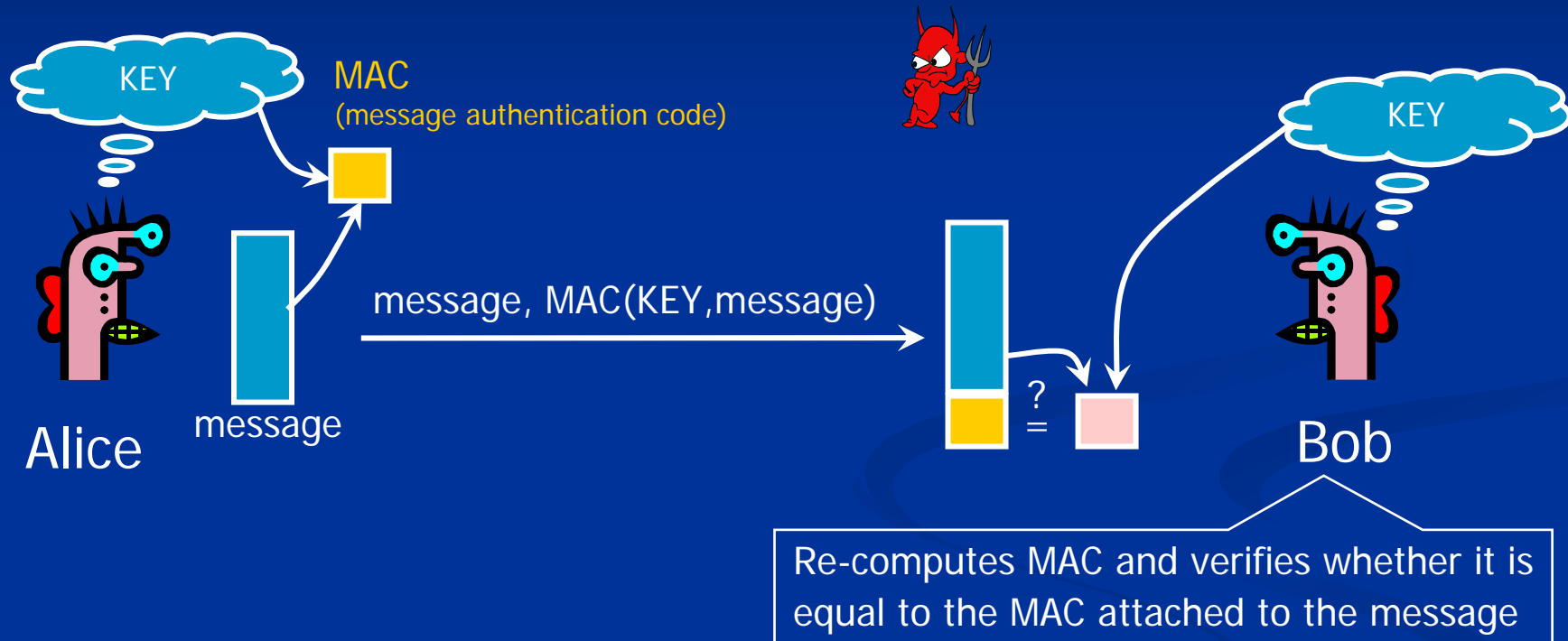
Software manufacturer wants to ensure that the executable file is received by users without modification...

Sends out the file to users and publishes its hash in NY Times

The goal is integrity, not secrecy

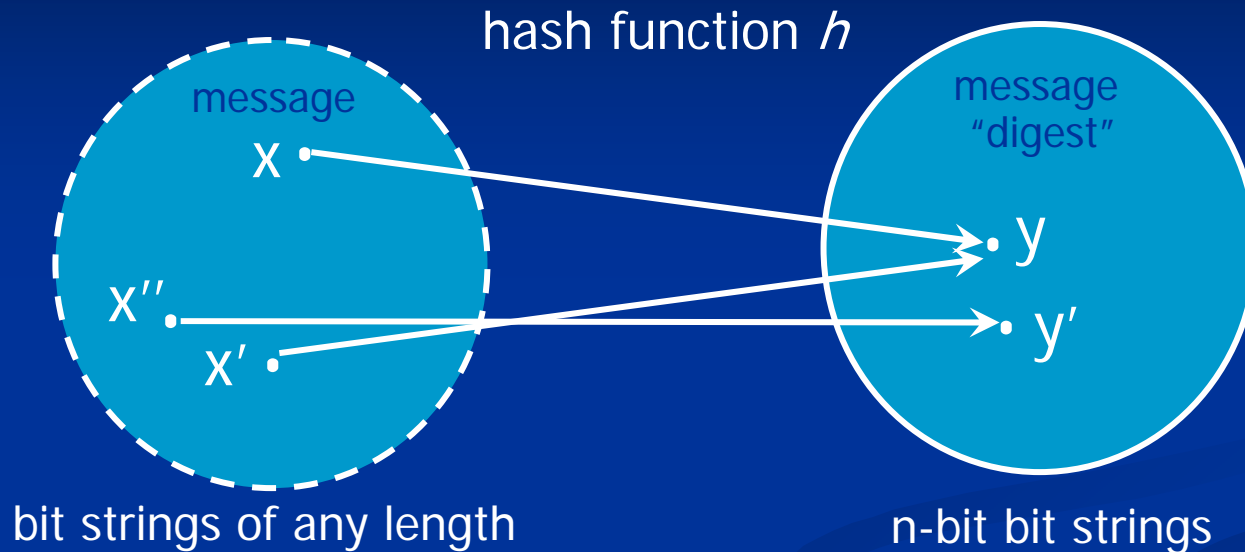
Idea: given goodFile and hash(goodFile),
very hard to find badFile such that hash(goodFile)=hash(badFile)

Authentication Without Encryption



Integrity and authentication: only someone who knows KEY can compute MAC for a given message

Hash Functions: Main Idea



- h is a lossy compression function
 - **Collisions:** $h(x)=h(x')$ for some inputs x, x'
 - Result of hashing should "look random" (make this precise later)
- $h(x)$ is relatively easy to compute for any given x
- Example: $h(m) = m \pmod{n}$
 - Is this a good cryptographic hash function?
- **Cryptographic hash function** needs a few properties...

One-Way

- Intuition: hash should be hard to invert
 - “Preimage resistance”
 - Let $h(x')=y \in \{0,1\}^n$ for a random x'
 - Given y , it should be hard to find any x such that $h(x)=y$
- Why?
 - Let $C = H(key || M)$, if Eve can invert hash function to get $key || M = H^{-1}(C)$, then Eve can recover key .

Collision Resistance

- Strong collision resistance
 - Should be hard to find any pair (x, x') such that $h(x)=h(x')$
 - Why? → Example of digital signature:
 - Eve wants to add Alice's signature to a message m'
 - Eve finds pair (m, m') such that, $h(m) = h(m')$
 - Eve requests Alice to sign message m
 - Eve get $sig(h(m))$ from Alice, where $sig(h(m)) = sig(h(m'))$
 - Eve can create a signed message $(m', sig(h(m')))$
 - Vulnerable to Birthday paradox
 - Let t be the number of values $x, x', x'' \dots$ we need to look at before finding the first pair x, x' s.t. $h(x)=h(x')$
 - What is probability of collision for each pair x, x' ?
 - How many pairs would we need to look at before finding the first collision?

Collision Resistance

- Weak collision resistance:
 - Given randomly chosen x , hard to find x' such that $h(x)=h(x')$
 - Why?
 - If it's easy to find x' , such that $h(x)=h(x')$ attackers can modify the executable file ...
 - Attacker must find collision for a specific x . By contrast, to break collision resistance, enough to find any collision.
 - Weak collision resistance does not imply strong collision resistance (why?)

Common Hash Functions

- MD5
 - 128-bit output
 - Designed by Ron Rivest, used very widely
 - Collision-resistance broken (summer of 2004)
- RIPEMD-160
 - 160-bit variant of MD-5
 - Collision-resistance broken (August 2004)
- SHA-1 (Secure Hash Algorithm)
 - 160-bit output
 - US government (NIST) standard as of 1993-95
 - Also the hash algorithm for Digital Signature Standard (DSS)
 - Collision-resistance broken (summer of 2005)