3. Cryptography Review
ENEE 657

Prof. Tudor Dumitraș
Assistant Professor, ECE
University of Maryland, College Park

http://ter.ps/enee657

Today's Lecture

- Where we've been
  - Introduction to computer security
  - Memory corruption exploits
- Where we're going today
  - Cryptography review
- Where we're going next
  - **Homework 1 due on Friday!**
  - OS protection mechanisms

Reading: Low Level Security By Example

Discussion

3

A crash course in cryptography

4

Random Variables and Random Numbers

- A **random variable** X is a rule for assigning a number to each experimental outcome
  - Example: X is the number of heads after n coin tosses
  - We're often trying to determine the **probability distribution** of X
    - Assign probabilities $p_1, p_2, \ldots p_n$ to values $x_1, x_2, \ldots x_n$ of X
    - Uniform distribution: $p_1 = p_2 = \ldots = p_n$
- Computers produce **pseudo-random** numbers
  - **Sequence** of numbers that **appears** random
  - The numbers in the sequence follow certain mathematical properties, e.g. **uniform distribution**
  - Sequence of numbers (orbit) starting from a **seed** value
    - Same seed provided => same sequence generated

Common crypto misuse #1: **using static seeds**

6

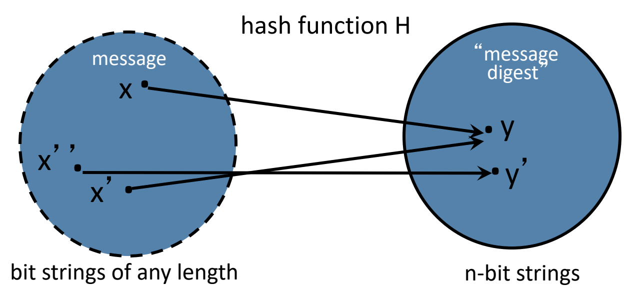
Crypto Primitives: Hashing vs. Encryption

- Given plaintext x
  - Hashing: function H(), such that H(x) is an n-bit string
  - Encryption: functions D() and E(), such that $D_{key'}(E_{key}(x)) = x$
- Hashing is one-way. There is no “uh-hashing”!
  - A ciphertext can be decrypted with a decryption key… hashes have no equivalent of decryption
- Hash(x) looks “random” but can be compared for equality with Hash(x’)
  - Hash the same input twice → same hash value
  - Encrypt the same input with different keys → different ciphertexts

Common crypto misuse #2: **using constant encryption keys**

7

Hash Functions: Main Idea



- Hash function H is a lossy compression function
  - **Collision**: H(x)=H(x') for some inputs x≠x'
- H(x) should look random
  - Every bit (almost) equally likely to be 0 or 1
- A **cryptographic hash function** must have certain properties

8

One-Way Functions

- Intuition: hash should be hard to invert
  - "Preimage resistance"
  - Given h() and y, it should be hard to find any x such that h(x)=y
    - y is an n-bit string randomly chosen from the output space of the hash function, ie, y=h(x') for some x'
- How hard?
  - Brute-force: try every possible x, see if h(x)=y
  - SHA256 (a common hash function, broken recently) has 256-bit output
    - Suppose we have HW that can compute 1B (≈ 2^{30}) hashes at once and do 2^{34} trials per second
    - Can try 2^{89} hashes per year
    - Will take 2^{167} years to invert SHA256 on a random image

9

Birthday Paradox

- T people
- Suppose each birthday is a random number taken from K days (K=365) – how many possibilities?
  - K^T - samples **with replacement**
- How many possibilities that are all different?
  - $(K)_T = K(K-1)\ldots(K-T+1)$ - samples **without replacement**
- Probability of no repetition?
  - $(K)_T/K^T \approx 1 - T(T-1)/2K$
- Probability of repetition?
  - $O(T^2)$

10

Collision Resistance

- Should be hard to find $x \neq x'$ such that $h(x)=h(x')$
- Birthday paradox
  - Let T be the number of values $x,x',x''\ldots$ we need to look at before finding the first pair $x \neq x'$ s.t. $h(x)=h(x')$
  - Assuming h is random, what is the probability that we find a repetition after looking at T values? $O(T^2)$
  - Total number of pairs?
    - n = number of bits in the output of hash function $O(2^n)$
  - Conclusion: $T \approx O(2^{n/2})$
- Brute-force collision search is $O(2^{n/2})$, not $O(2^n)$
  - For SHA256, this means $O(2^{128})$ vs. $O(2^{256})$

11

One-Way vs. Collision Resistance

- One-wayness does not imply collision resistance
  - Suppose g() is one-way
  - Define h(x) as g(x') where x' is x except the last bit
    - h is one-way (cannot invert h without inverting g)
    - Collisions for h are easy to find: for any x, h(x0)=h(x1)
- Collision resistance does not imply one-wayness
  - Suppose g() is collision-resistant
  - Define h(x) to be 0x if x is (n-1)-bit long, else 1g(x)
    - Collisions for h are hard to find: if y starts with 0, then there are no collisions; if y starts with 1, then must find collisions in g
    - h is not one way: half of all y's (those whose first bit is 0) are easy to invert (**how**?), thus random y is invertible with p ≥ ½

12

Weak Collision Resistance

- Given a randomly chosen x, hard to find x' ≠ x such that h(x)=h(x')
  - "Second pre-image resistance"
  - Attacker must find collision for a specific x… by contrast, to break collision resistance, enough to find any collision
  - Brute-force attack requires $O(2^n)$ time
- Weak collision resistance does not imply collision resistance

13

Application: Password Hashing

- Instead of user password, store hash(password)
- When user enters a password, compute its hash and compare with the entry in the password file
  - System does not store actual passwords!
  - Cannot go from hash to password!
- What attacks does this prevent?
  - Does hashing protect weak, easily guessable passwords?

14

Protecting Passwords Against Dictionary Attacks

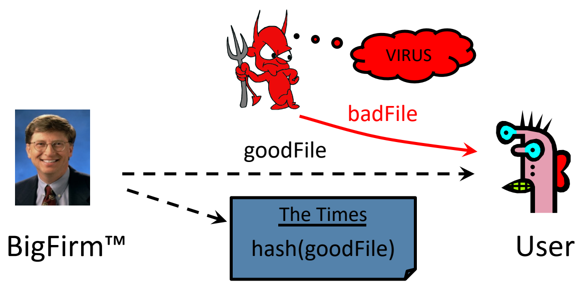
- People tend to chose common words for passwords
  - Attacker can pre-compute H(word) for every word in the dictionary – this only needs to be done once!
    (more on this later)
  - Defenses aim to increase the attacker's work factor
- Salting
  - Chose a salt: does not have to be secret, but must be random
  - Compute H(salt || pwd)
- Key derivation functions
  - Multiple iterations of the hash function: H(H (H (… (salt || pwd))))

Common crypto misuse #3: **hashing passwords with constant salts**

Common crypto misuse #4: **fewer than 1000 iterations of key deriv. function**

15

Application: Software 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 the 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)

16

Which Property Is Needed?

- Passwords stored as hash(password)
  - One-wayness: hard to recover entire password
  - Passwords are not random and thus guessable
- Integrity of software distribution
  - Weak collision resistance?
  - But software images are not random… maybe need full collision resistance
- Auctions: to bid B, send H(B), later reveal B
  - One-wayness… but does not protect B from guessing
  - Collision resistance: bidder should not be able to find two bids B and B’ such that H(B)=H(B’)

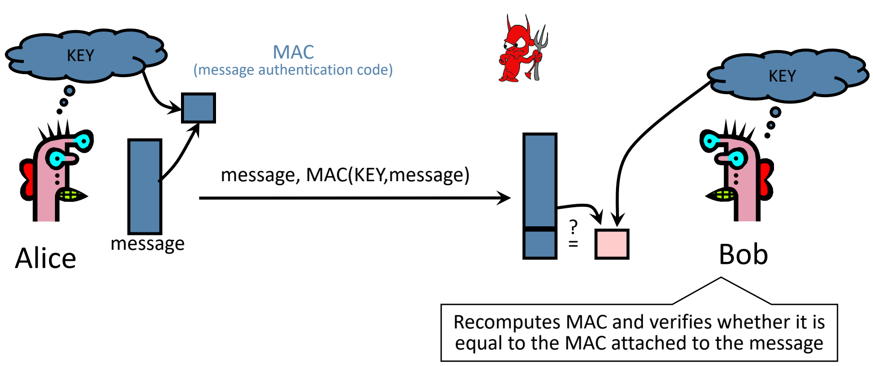
17

Common Hash Functions

- MD5
  - Collision attacks are practical
- RIPEMD-160
  - 160-bit variant of MD-5
- SHA-1 (Secure Hash Algorithm)
  - Widely used (but recently broken)
  - US government (NIST) standard as of 1993-95
    - Also the hash algorithm for Digital Signature Standard (DSS)
- SHA256

18

Integrity and Authentication



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

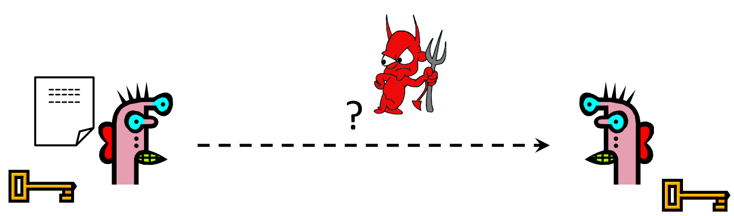
19

HMAC

- Construct MAC from a cryptographic hash function
  - Invented by Bellare, Canetti, and Krawczyk (1996)
  - Used in SSL/TLS, mandatory for Ipsec
- Why not encryption?
  - Hashing is faster than encryption
  - Library code for hash functions widely available
  - Can easily replace one hash function with another
  - There used to be US export restrictions on encryption

20

Symmetric Cryptography



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

21

Kerckhoffs's Principle

Full name:
Jean-Guillaume-Hubert-Victor-François-Alexandre-Auguste Kerckhoffs von Nieuwenhof

- An encryption scheme should be secure even if enemy knows everything about it **except the key**
  - Attacker knows all algorithms
  - Attacker does not know random numbers
- Do not rely on secrecy of the algorithms ("security by obscurity")

Common crypto misuse #5: **DIY crypto**

22

Common Symmetric Crypto Algorithms

- DES
  - 64-bit blocks (56-bit key + 8 bits for parity)
  - Outdated, but still in use (especially as 3DES)
    - 3DES: DES + inverse DES + DES (with 2 or 3 different keys)
- AES (Rijndael)
  - 128-bit blocks, keys can be 128, 192 or 256 bits
  - US federal standard as of 2001
- These are **block ciphers**
  - Operate on fixed-size blocks
  - As opposed to stream ciphers (key is as long as the plaintext)

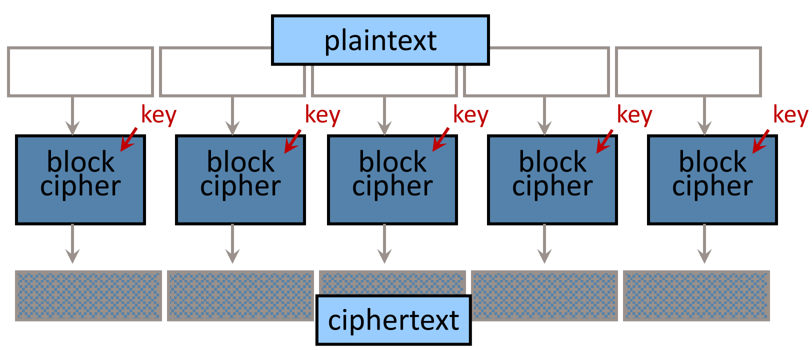
23

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.

24

ECB Mode

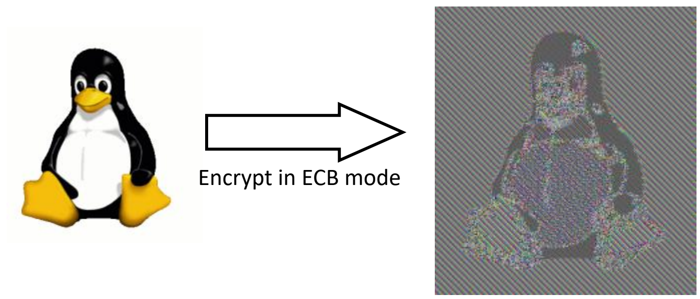


- Identical blocks of plaintext produce identical blocks of ciphertext
- Does not guarantee **integrity**

25

Information Leakage in ECB Mode

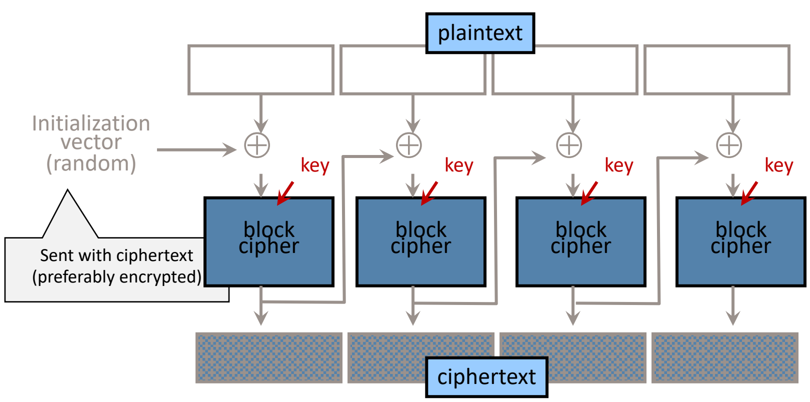
[Wikipedia]



Common crypto misuse #6: **using ECB mode**

26

CBC Mode: Encryption

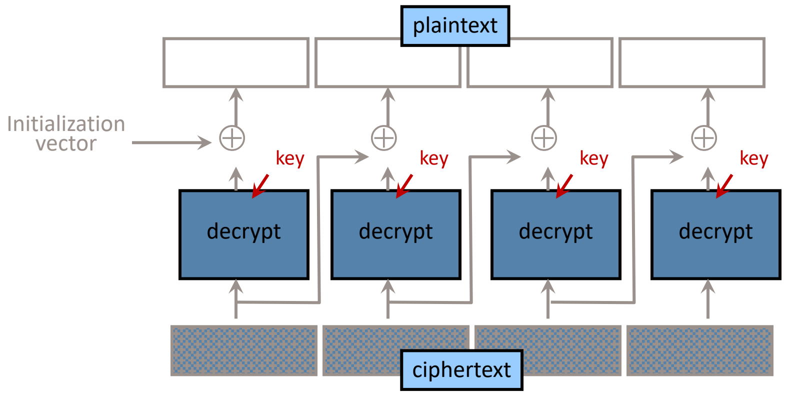


- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext

Common crypto misuse #7: **using a constant IV in CBC mode**

27

CBC Mode: Decryption

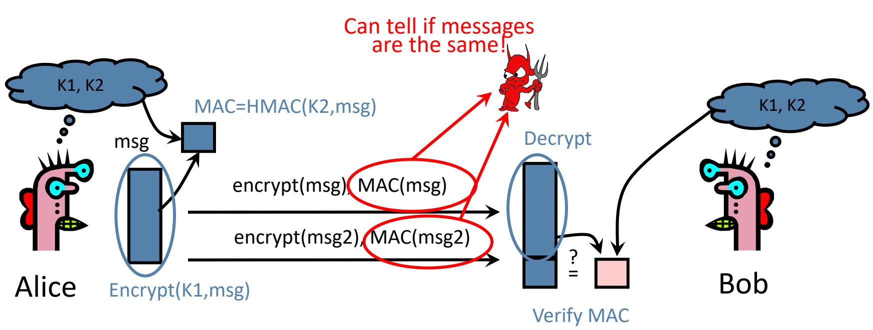


- Still does not guarantee integrity

28

Encrypt + MAC

Goal: confidentiality + integrity + authentication

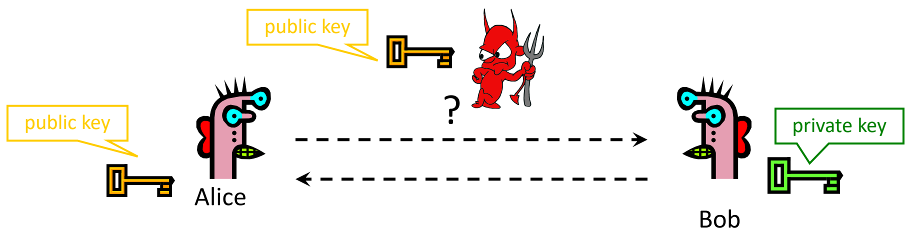


MAC is deterministic: messages are equal ⇒ their MACs are equal

Solution: Encrypt, <u>then</u> MAC (or MAC, then encrypt)

32

Public-Key Cryptography



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 message that only Bob can read

2. Bob wants to send a message that only Bob could have written

33

Applications of Public-Key Crypto

- Encryption for confidentiality
  - Anyone can encrypt a message
    - With symmetric crypto, must know the secret key to encrypt
  - Only someone who knows the private key can decrypt
  - Secret keys are only stored in one place
- Digital signatures for integrity
  - Only someone who knows the private key can sign
  - For authentication: must know to whom the private key belongs
- Session key establishment
  - Exchange messages to create a secret session key
  - Then switch to symmetric cryptography (why?)

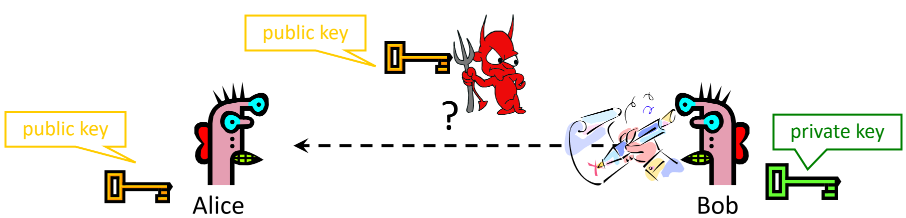
34

Public-Key Encryption

- Key generation: computationally easy to generate a pair (public key PK, private key SK)
- 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 learn anything about M from C without SK
  - Trapdoor function: Decrypt(SK,Encrypt(PK,M))=M
- Popular algorithm: RSA

35

Digital Signatures: Basic Idea



Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

Goal: Bob sends a "digitally signed" message
1. To compute a signature, must know the private key
2. To verify a signature, only the public key is needed

36

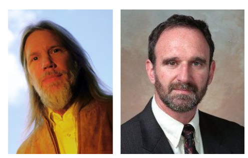
Popular Digital Signature Schemes

- RSA signatures
  - Signing and decryption are the same mathematical operation
  - Verification and encryption are the same mathematical operation
  - Message must be hashed and padded

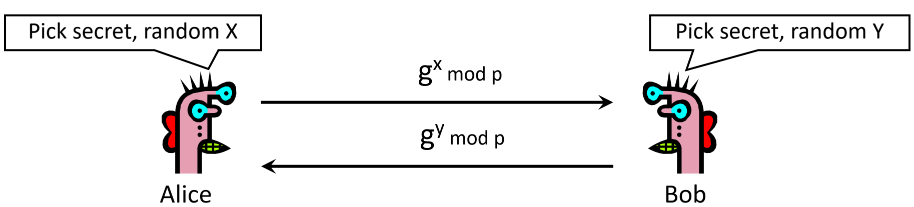
- DSA (digital signature algorithm) signatures
  - U.S. government standard (1991-94)
  - Modification of the ElGamal signature scheme (1985)
  - Security of DSA requires hardness of discrete log problem
    - Hard to extract x (private key) from g^x mod p (public key)
  - If the same message is signed twice, signatures are different
    - Each signature is based in part on random secret k
    - Secret k must be different for each signature!

37

Diffie-Hellman Protocol



- Alice and Bob never met and share no secrets
- Public info: p and g
  - Hard to extract x (private key) from g^x mod p (public key)



Compute $k=(g^y)^x=g^{xy}$ mod p

Compute $k=(g^x)^y=g^{xy}$ mod p

Common crypto misuse #8: **exchanging keys w/o authenticating the endpoint**

38

Properties of Diffie-Hellman

- Assuming the discrete logarithm problem is hard, Diffie-Hellman is a secure key establishment protocol against passive attackers
  - Eavesdropper can’t tell the difference between the established key and a random value
  - Can use the new key for symmetric cryptography
- Basic Diffie-Hellman protocol does not provide authentication
  - IPsec combines Diffie-Hellman with signatures, anti-DoS cookies, etc.

39

Disadvantages of Public-Key Crypto

- Calculations are 2-3 orders of magnitude slower
  - Modular exponentiation is an expensive computation
  - Typical usage: use public-key cryptography to establish a shared secret, then switch to symmetric crypto
    - SSL, IPsec, most other systems based on public crypto
- Keys are longer
  - 2048 bits (RSA) rather than 128 bits (AES)
- Relies on unproven number-theoretic assumptions
  - Factoring, RSA problem, discrete logarithm problem, decisional Diffie-Hellman problem…

40

Advantages of Public-Key Crypto

- Confidentiality without shared secrets
  - Very useful in open environments
  - Can use this for key establishment, avoiding the "chicken-or-egg" problem
    - With symmetric crypto, two parties must share a secret before they can exchange secret messages
- Authentication without shared secrets
- Encryption keys are public, but must be sure that Alice's public key is really her public key
  - **How?**

41

Additional References

- Jonathan Katz's Coursera class: https://www.coursera.org/course/cryptography
- KPS chapters 2–6

44

Assignment for Wednesday

- Using http://keybase.io for proving your identity online
  1. Create an account on Keybase. Use your hacker handle.
  2. Use Keybase to prove your identity on an online application (Twitter, Reddit, Github, etc.)
  3. Post a security analysis of Keybase on our Piazza forum
     - What security properties does Keybase provide?
     - What assumptions does it make about the adversary?
     - What are some potential weaknesses of the system? How would you attack it?
  4. Read posts from your classmates and discuss – do you disagree with their analysis?

45

Review of Lecture

- What did we learn?
  - Hash functions: one-way, collision resistant, weakly collision resistant
  - Message authentication codes
  - Security properties: confidentiality, integrity, authentication
  - Symmetric crypto
  - Public key crypto
  - Common ways to misuse crypto APIs
- Sources
  - Vitaly Shmatikov
- What's next?
  - OS protection mechanisms

46