CSE509: (Intro to) Systems Security

Fall 2012

Radu Sion

Intro
Symmetric Key Encryption
Hash Functions

A message from our sponsors: What this class *is*

- **Fundamentals**
  - Systems Security, crypto
- **How** do things work
- **Why**
- How to **design** secure stuff
- Focus mostly on **systems**. But of course everything is networked today anyway

Another one from our sponsors: What this class is *not*

- How to **install** XXX
- **Command line options** of XXX
- Latest **iexplorer buffer overflow bug**
- Latest McAffee/XXX **products**
- Network **administration**
- How to break your gf/bf email account

Ground Rules

- Dates are listed online now
- Zero tolerance to academic dishonesty
- Informal class, ask questions anytime
- **Read your assigned readings !**
- Call me Radu
- Questions: office hours, or I can call you
- Email: cse509@cs
- Suggest cool alternatives to project
- Have fun !

Evaluation

- 3 Homeworks
- Midterm
- In class Pop quizzes
- Final
- 2 Projects (or you can suggest a security project you would like to do for credit and convince me it is worth doing)
- **http://www.cs.stonybrook.edu/~cse509**

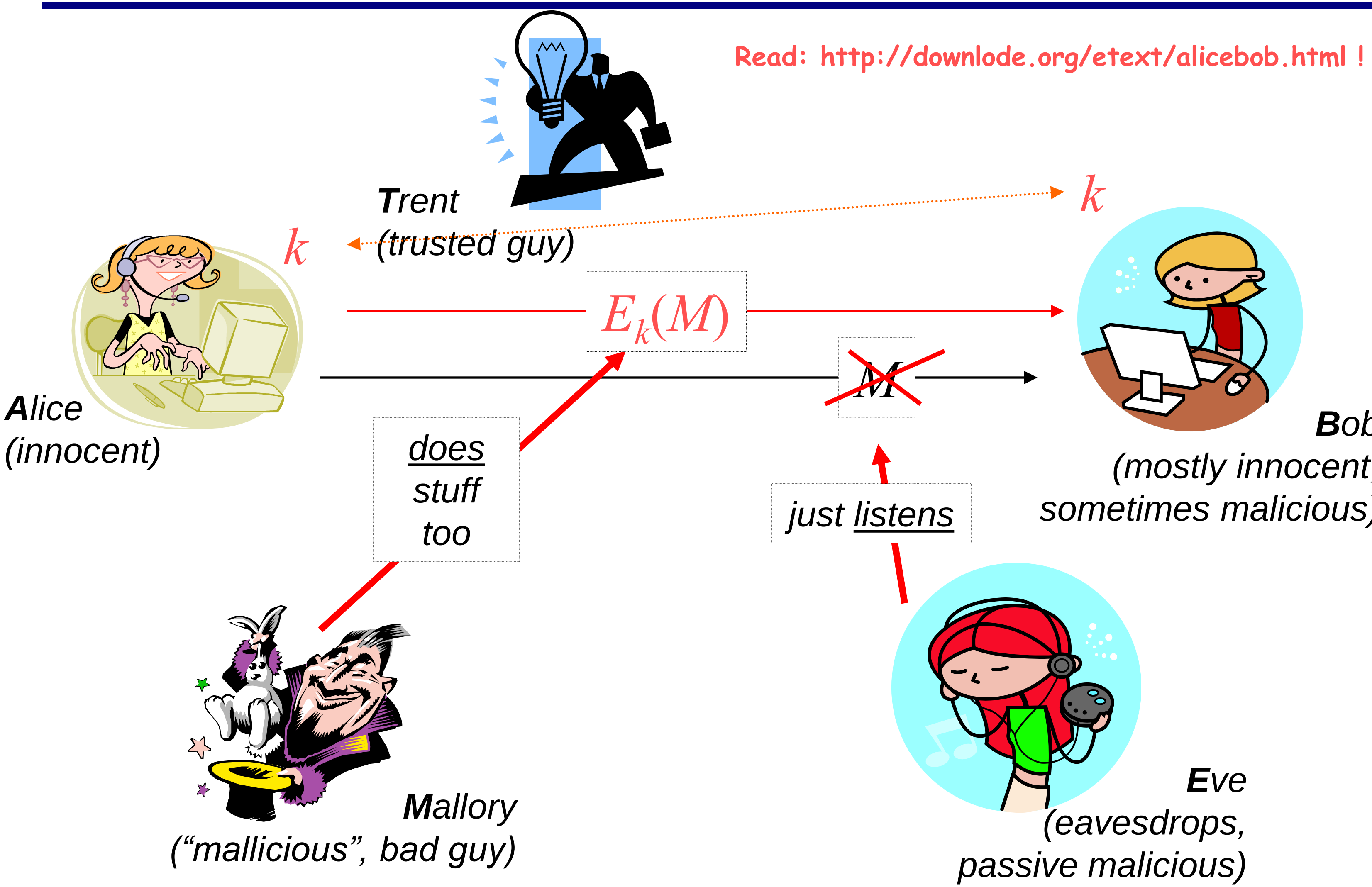
Your Background

- C programming
- Assembler programming (project 2)
  - You *may* learn this on the way
- Understanding of
  - TCP/IP and networking in general
  - Operating systems

Classical Cryptography

- Single/Symmetric Key Encryption
- Cryptographic Hash Functions

Basic Blocks: Meet the Cast



Basic Blocks: First questions !

- Where does *k* come from ? (key distribution – chicken and egg problem)
- Can Eve distinguish between $E_k(M_1)$ and $E_k(M_2)$ if she knows M_1 and M_2 ? Should not be able to !!!
  (indistinguishability under the choosen plain text attack – IND-CPA – see later)
- Make sure that $E_k(M_1) \neq E_k(M_2)$ if $M_1 \neq M_2$ (maybe not ?)
- Can Mallory modify $E_k(M)$ into an $E_k(M_{mallory})$?
  (non-malleability – see later)
- etc (! lots of stuff !) – danger: things seem trivial and they are not – result: super weak systems !

Example

- Example: Cæsar cipher
  - $\mathcal{M}$ = { sequences of letters }
  - $\mathcal{K}$ = { i | i is an integer and $0 \leq i \leq 25$ }
  - $\mathcal{E}$ = { E_k | $k \in \mathcal{K}$ and for all letters m,
    $E_k(m) = (m + k) \bmod 26$ }
  - $\mathcal{D}$ = { D_k | $k \in \mathcal{K}$ and for all letters c,
    $D_k(c) = (26 + c - k) \bmod 26$ }
  - $\mathcal{C} = \mathcal{M}$

Attacks

- Opponent whose goal is to break cryptosystem is the *adversary*
  - Assume adversary knows algorithm used, but not key
- Many types of attacks:
  - *ciphertext only*: adversary has only ciphertext; goal is to find plaintext, possibly key
  - *known plaintext*: adversary has ciphertext, corresponding plaintext; goal is to find key
  - *chosen plaintext*: adversary may supply plaintext and obtain corresponding ciphertext; goal is to find key
  - *chosen ciphertext*: adversary may supply ciphertext and obtain corresponding plaintext; goal is to find key
  - etc

Basis for Attacks

- Mathematical attacks
  - Based on analysis of underlying mathematics
- Statistical attacks
  - Make assumptions about the distribution of letters, pairs of letters (digrams), triplets of letters (trigrams), *etc*.
    - Called *models of the language*
  - Examine ciphertext, correlate properties with the assumptions.

Statistical Attack: e.g., for known language

- Compute frequency of each letter in ciphertext:

G	0.1	H	0.1	K	0.1	O	0.3
R	0.2	U	0.1	Z	0.1		

- Apply 1-gram model of English
- Correlate and invert encryption

Cæsar’s Problem

- Key is too short
  - Can be found by exhaustive search
  - Statistical frequencies not concealed well
    - They look too much like regular English letters
- So make it longer
  - Multiple letters in key
  - Idea is to smooth the statistical frequencies to make cryptanalysis harder

Vigènere Cipher

- Like Cæsar cipher, but use a phrase
- Documented by Blaise de Vigenere (court of Henry III of France) in Paris, 1586 – actually a variant of a cipher by a J.B. Porter
- Example
  - Message `THE BOY HAS THE BALL`
  - Key `VIG`
  - Encipher using Cæsar cipher for each letter:

```
key    VIGVIGVIGVIGVIGV
plain  THEBOYHASTHEBALL
cipher OPKWWECIYOPKWIRG
```

Holy Grail: The One-Time Pad

- A Vigenère cipher with a random key at least as long as the message
  - Provably unbreakable
  - Why? Look at ciphertext `DXQR`. Equally likely to correspond to plaintext `DOIT` (key `AJIY`) and to plaintext `DONT` (key `AJDY`) and any other 4 letters
  - Warning: keys *must* be random, or you can attack the cipher by trying to regenerate the key
    - Approximations, such as using pseudorandom number generators to generate keys, are *not* random

Cryptographic Hash Functions

- Mathematical function to generate a set of k bits from a set of n bits (where $k \leq n$).
  - k is usually smaller then n
- Example: ASCII parity bit
  - ASCII has 7 bits; 8th bit is "parity"
  - Even parity: even number of 1 bits
  - Odd parity: odd number of 1 bits

Example Use

- Bob receives "10111101" as bits.
  - Sender is using even parity; 6 1 bits, so character was received correctly
    - Note: could be garbled, but 2 bits would need to have been changed to preserve parity
  - Sender is using odd parity; even number of 1 bits, so character was not received correctly

Definition

Cryptographic hash h: $A \rightarrow B$:

1. For any $x \in A$, $h(x)$ is **easy to compute**
2. *h(x)* is of fixed length for any x (**compression**)
3. For any $y \in B$, it is computationally infeasible to find $x \in A$ such that $h(x) = y$. (**pre-image resistance**)
4. It is computationally infeasible to find <u>any</u> two inputs $x, x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$ (**collision resistance**)
5. Alternate form of 3 (stronger): Given any $x \in A$, it is computationally infeasible to find a different $x' \in A$ such that $h(x) = h(x')$. (**second pre-image resistance**)

Collisions

- If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*
  - Pigeonhole principle: if there are n containers for n+1 objects, then at least one container will have 2 objects in it.
  - Application: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

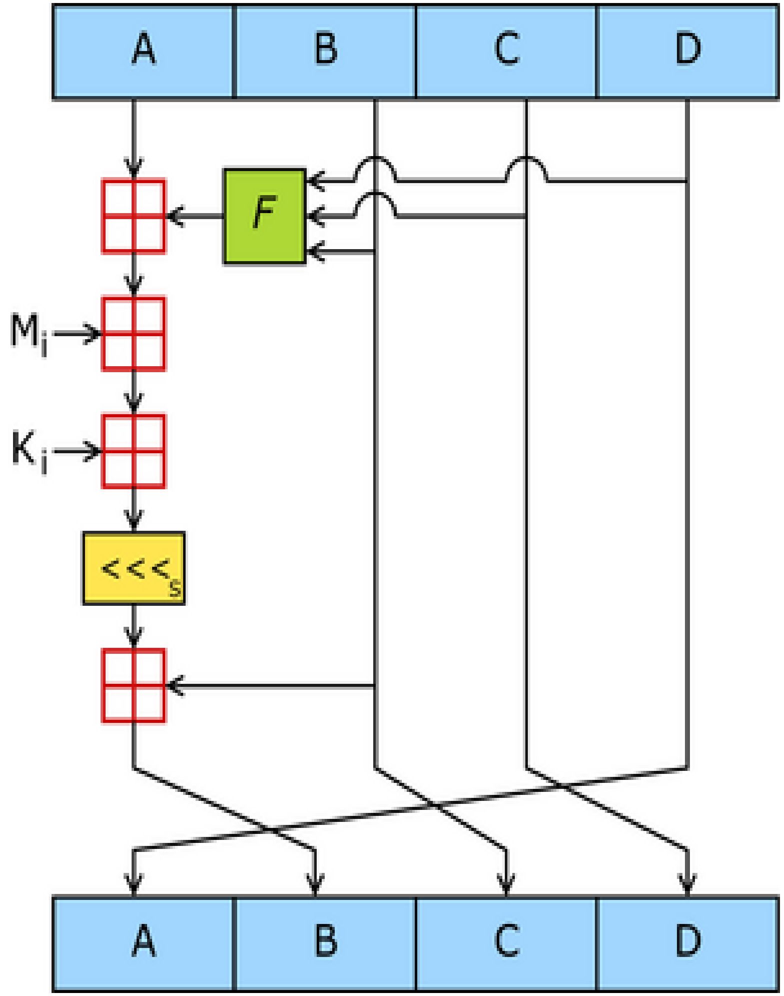
Intuition

- A hash is a **one-way, non-invertible** function of that produces **unique** (with *high likely-hood*), **fixed-size** outputs for different inputs.
- The probability of any bit flipping in the output bit-string should be always ½ for any change (even one bit) in the input (“randomness”).

MD5

- Basic idea: Continuously update hash value with 512 bit blocks of message
  - 128 bit initial value for hash
  - Bit operations to "compress"
- Compression function: Update 128 bit hash with 512 bit block
  - Pass 1: Based on bits in first word, select bits in second or third word
  - Pass 2: Repeat, selecting based on last word
  - Pass 3: xor bits in words
  - Pass 4: $y \oplus (x \text{ or } \sim z)$

MD5



Example: MD5

md5_digest("The quick brown fox jumps over the lazy dog")
= **9e107d9d372bb6826bd81d3542a419d6**

md5_digest("The quick brown fox jumps over the lazy cog")
= **1055d3e698d289f2af8663725127bd4b**

Weaknesses

- Length Extension
  - h(m||X) can be built out of h(m) and X !!!
- Partial Message Collision
  - if we find **m'≠m** such that **h(m')=h(m)** then **h(m||X)=h(m'||X)** because **h(m||X) ≈ h(h(m)||X)**

Fixes for weak hashes

- Slow (claim full n-bit security)
  - **slow_coolhash(m)=h(h(m)||m)**
- Faster (but claim only n/2- bit security !)
  - **faster_coolhash(m)=h(h(m))**

Hashes to (not) use

- **MD5**
  - Output 128-bit
  - Designed by Ron Rivest, 1991
  - Wang et. al.: collision in 1 hr using cluster (2004)
  - Klima: collision with 1 min on laptop (2006)

- **SHA-1**
  - Output 160-bit
  - Designed by NSA
  - ”broken” by Wang et. al. – attack requires $< 2^{69}$ ops to find collision (exhaustive would take 2^{80}) (2005)

Hashes to (not) use

- Do not use at all the following:
  - **MD5, SHA-0/1, any other obscure "secret" ones**
- For use in civilian/.com setting (until 2010/15):
  - **SHA-256/512**

Cryptographic One-Way Hash Functions ?! Why ?

- Unique identifiers
  - Handy because small
- Used in more complex protocols
  - Pre-commitment (because one-way)
- Cool result: "*pseudo-random number generators exist iff. one-way functions exist*"

Keyed Hashes

- MAC(msg)=H(H(key,msg,key),msg)
- Usage: append this to message to allow authentication

Why Keyed Hashes ?

- Want to enable only a certain party to verify authenticity of data for which it has a MAC (for example).
- Want to prevent Mallory to alter message and simply replace MAC (cannot do it now – doesn't know the secret key)