L3:
Password Cracking

Sudhir Aggarwal and Shiva Houshmand
Florida State University
Department of Computer Science
E-Crime Investigative Technologies Lab
Tallahassee, Florida 32306
August 5-7, 2015

Password Hashes

1. User creates password: “password123”
2. Computer hashes the password

   MD5(“password123”) = 2848805208b236de1b7caeb7fd0bb0a7
3. To log in the user types “password123”
4. The computer hashes “password123” and compares it against the hash it stored before

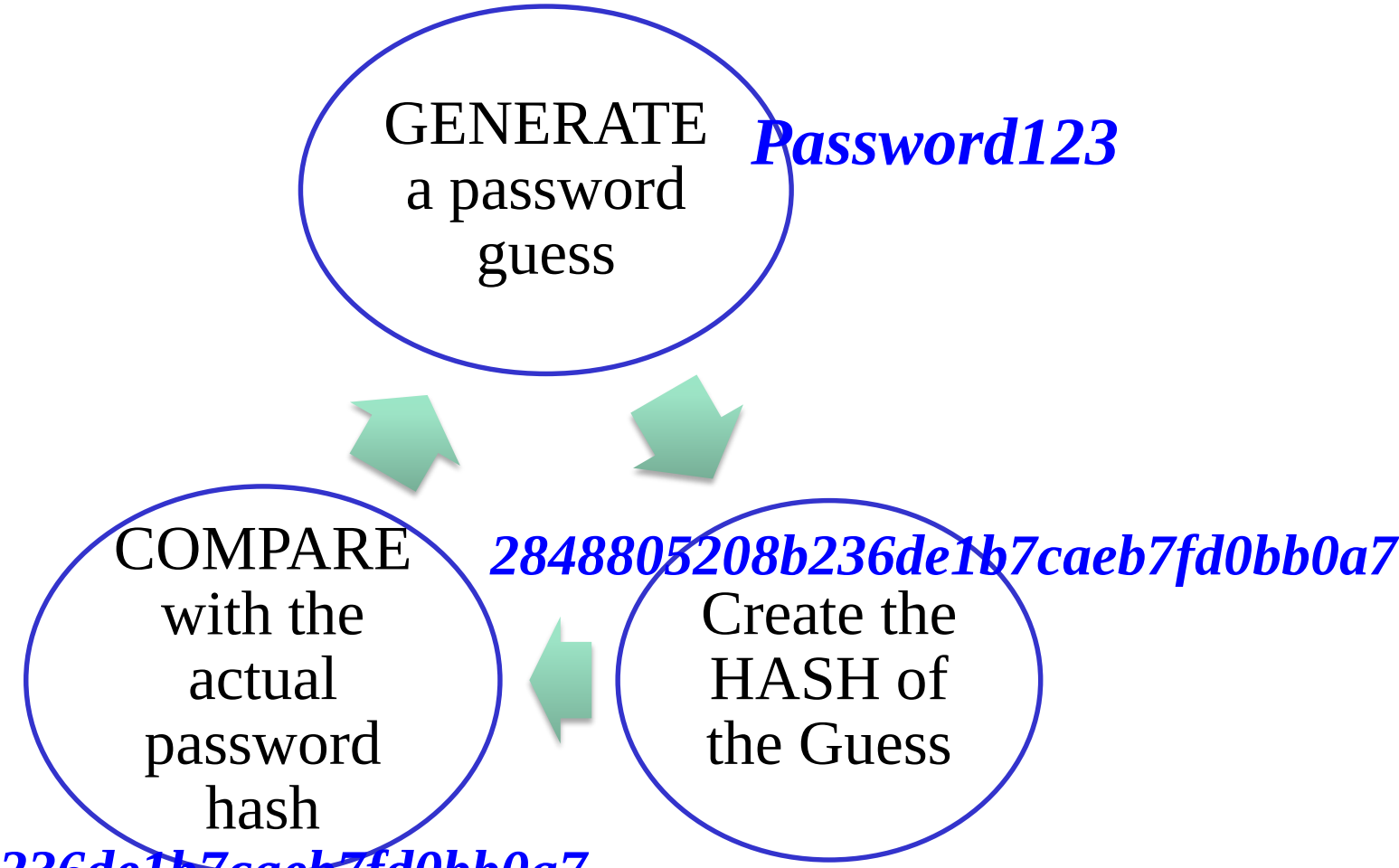
Two types of password cracking

- Online
  - Trying different passwords to log in
  - Can be slow and noisy
  - You may only be allowed a few guesses
- Offline
  - You already obtained the password file
  - You are only limited by your own resources

Offline Password Cracking?

- Passwords are usually stored as hash values.
- Note that typically the hash function is a 1-way function and is not invertible.
- In order to find the password, the attacker needs to repeatedly make a guess, apply the same hash function and compare the hashes until a match is found.

Password Cracking



Passwords, passwords, passwords top 11 list (2012)

- password
- 123456
- 12345678
- abc123
- qwerty
- monkey
- letmein
- dragon
- 111111
- baseball
- iloveyou

Assumptions in the Previous? Not always easy to satisfy

- You have the actual hash somehow! From:
  - On the disk / in a file
  - On some other device / paper
  - You know what part is the hash value
- You know the type of hash!
- You know the hash algorithm!
- Or, you can verify the password (TrueCrypt)

Typical Cracking Approaches

1. Dictionary Based Attacks

- Uses a file containing words, phrases, common passwords, etc. called dictionary.

- Sometimes further processing is applied on each word for example, replacing words with their LeetSpeak equivalent. “move” → “m0v3”

- Each entry from the dictionary is a password guess!

1. Dictionary Based Attacks

- Sometimes mangling rules are used along with a dictionary.

- Mangling rules: Append 123
  Capitalize the word
  Append !!
  Append 1234
  Append dates (1997-2014)

1. Dictionary Based Attacks

- Cain and Abel (Windows, open source)
- John the Ripper (open source)
- Brutus (not updated for some time)
- L0phtCrack (Windows, open source)
- HashCat (open source)
- Many others …
- Commercial: Guidance (Encase), AccessData (FTK)

Typical Cracking Approaches

2. Brute Force Attack

- Try every possible combination of characters up to a given length.
- Eventually will find the password
- May not be very efficient in terms of hashes cracked / number of guesses made
- How do you specify the order of “all possible combinations.”

2.Brute Force Attack

- Trying: aaaa
  aaab
  aaac
  …..
  …..
  gaaa
  gaab
  gaac
  …..
- Clearly need a way to ensure that everything is theoretically tried!

Typical Cracking Approaches

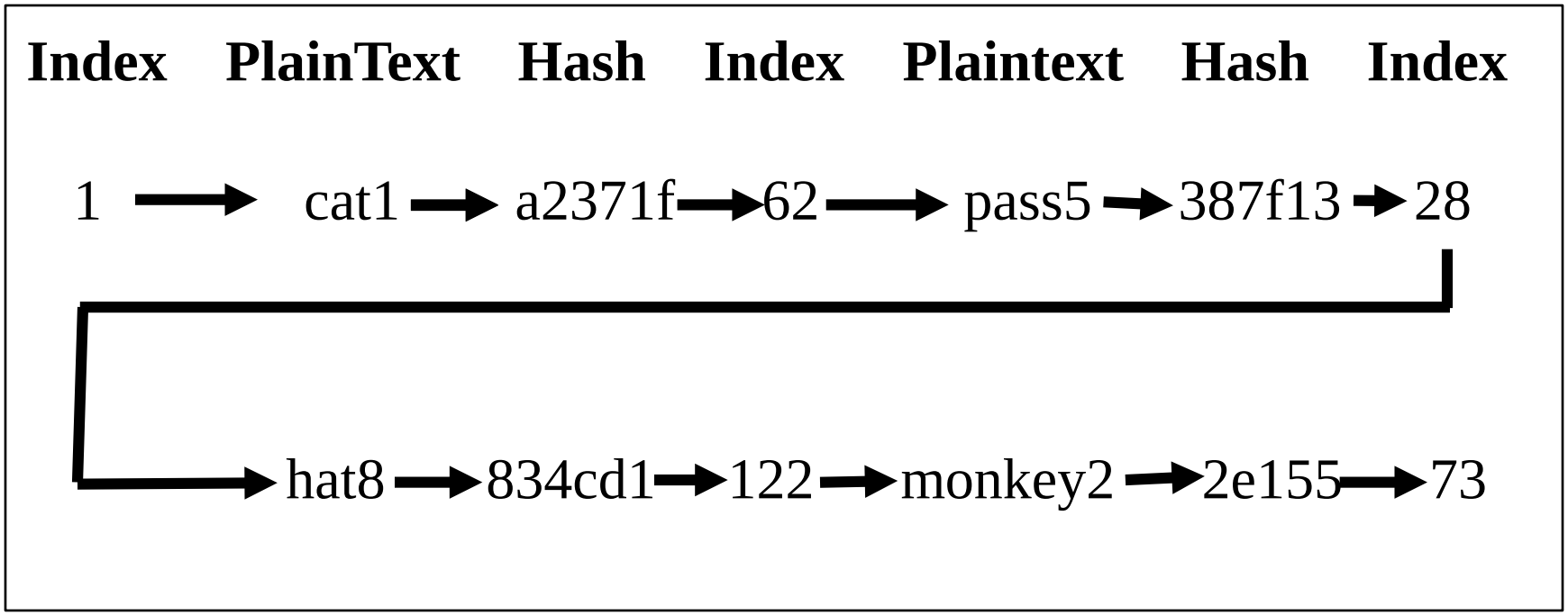
3. Look up Table / Rainbow Table

- Look up tables are effective methods for password cracking.
- The table contains pre-computed hashes of passwords and their corresponding passwords.
- The main weakness is their large file size.
- Rainbow Tables use a time-memory trade off technique and require less storage and more processing time than simple look up tables.

3. Rainbow Table Construction

- Index Value – the index is from 0 to the maximum # of plaintext passwords stored -1. Thus each index value simply corresponds to a unique password.
- Functions used in creation and application:
  - IndextoPlain: Given an index value, returns the plaintext password value. Note that this must be fast and "obvious" to find the plaintext given index.
  - PlaintoHash: Given a password, computes the hash value.
  - HashtoIndex: Given a hash, computes an index value using a reduction function Note that for rainbow tables the reduction function is **different at each position** in the chain.

3. Rainbow Table Construction: Chains



We store only the initial and final indices of a chain. Chains can store thousands of passwords. How do you find a password given the hash?

3. Rainbow Table: finding password given hash

Target Hash	Index	Plain Text	Hash	Index	Plain Text	Hash	Index	Plain Text	Hash	Index
a2371f	102									
a2371f	37	rat2	64257f	97						
a2371f	34	bat2	834cd1	107	stuff1	735821	81			
a2371f	62	pass	387f13	28	hat8	834cd1	122	monkey2	2ee125	73

Typical Cracking Approaches

3. Look up Table / Rainbow Table

- Example: https://crackstation.net/

e26fcf419830f5a3611d7fa36f558594
4076e7493e3dc14d844a801d8d7db114
b243071d14cd440adbee55756d747d53
B506fb0cabb60e9f1132daa72df4d567

- Try a password!! http://www.miraclesalad.com/webtools/md5.php

Typical Cracking Approaches

- Defense Against Rainbow Tables: Large random SALT value
  - The salt value is not secret and can be a random value stored with the password hash
  - Saltedhash(password) = hash(password + salt)
  - Two users with the same password will have different hash values.
  - The attacker needs to create the rainbow table for each salt value.

- Rainbow Tables are used less frequently nowadays for various other reasons: Massive storage is a problem, GPUs are available, not good for cracking lots of hashes at a time.

Key Stretching

- Using salt does not prevent an attacker from dictionary-based or brute force attacks.
- Techniques are used to increase the computation time required to hash each passwords, by repeating a hash function multiple times.
- Standard algorithms such as PBKDF2 or bcrypt

PBKDF2

DK = PBKFD2(PRF, Password, salt, c, dkLen)

DK: Derived key

PRF: The Pseudo random function (hash algorithm)

output length *hLen*

C: iteration count (1000 at least)

dkLen: the length of the derived key in bytes

PBKDF2

$DK = T_1 \| T_2 \| \ldots T_{dkLen/hlen}$

$T_i = F(password, salt, c, i)$

$F(password, salt, c, i) = U_1 \wedge U_2 \wedge \ldots \wedge U_c$

$U_1 = PRF(password, Salt \| INT\_32\_BE(i))$

$U_2 = PRF(password, U_1)$

..

$U_c = PRF(Password, U_{c-1})$

Typical Cracking Approaches

4. Markov models

- Markov chains: a sequence of random variables $\{X_i\}$ indexed by the integers (also called time). Issues of time invariant etc.

- Markov chain of order *n:*
  $P(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_{i-n})$

- Can be applied to strings or other components. For strings $x_1x_2\ldots x_m$ we typically use a first order Markov chain where:
  $P(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i \mid x_{i-1})$

- Thus we have:
  $P(x_1x_2\ldots x_m) = P(x_1 \mid x_0)\, P(x_2 \mid x_1)\, P(x_3 \mid x_2) \ldots P(x_m \mid x_{m-1})$

4. Markov Models

- John the Ripper has a Markov mode
- It is based on work by Narayan and Shmatikov
  - N&S use a “Markovian filter” that filters out strings of some low probability θ.
  - They show how to define an index function that returns a string from the space of all allowed strings. This was based on quantizing string by length and value.
  - They also used an finite state machine to further restrict allowable strings.

Typical Cracking Approaches

5. Grammar Based Probabilistic Cracking

   – Learning a context-free grammar

   – Cracking using the grammar

   – Not openly available yet

   – Will discuss this later in much greater detail