

# Hashes, MACs & Passwords

Tom Chothia

Computer Security Lecture 5

# Today's Lecture

- Hash functions:
  - Generates a unique short code from a large file
  - Uses of hashes
  - MD5, SHA1, SHA2, SHA3
  - Message Authentication Codes
- Password protection
  - Password cracking.

# Hashes

- A hash of any message is a short string generated from that message .
- The hash of a message is always the same.
- Any small change makes the hash totally different.
- It is very hard to go from the hash to the message.
- It is very unlikely that any two different messages have the same hash.

# Uses of Hashing

- Download/Message verification
- Tying parts of a message together (hash the whole message)
- Hash message, then sign the hash.
- Protect Passwords
  - Store the hash, not the password

# Attacks on hashes

An attacker might like to:

- Find a message for a given hash: e.g. find a password: very hard.
- Pick a message with a given prefix and the same hash: “prefix collision attack”
- Find two “random” messages with the same hash: “Collision Attack”

# Birthday Paradox

- How many people do you need to ask before you find 2 that have the same birthday?
- 23 people, gives  $(23*22)/2 = 253$  pairs.
- Prob. that two people a different birthday is:  $364/365$
- $(364/365)^{(23*22/2)} = 0.4995$

# Collision attacks

- Collisions are surprising easy to find!
- If I find that `gdidhirndigighd.com` has the same hash as `CNN.com` then I pretend to be `CNN.com`
- Completely “random” messages that have the same hash are less easy to use. BUT, the hash scheme is considered broken.

# How the MD5 hash works

See notes.

- Careful analysis of these functions lead to:
  - A collision attack in  $2^{24.1}$  steps, (seconds)
  - chosen prefix attack in 2 hours
- MD5 was used in certificates so this attack could be used to fake SSL certificates.

# SHA1

- 1993, The US National Institute of Standards and Technology (NIST), developed a new hash SHA-0
- 1995, the NSA stepped in and “fixed” it: SHA-1.
- By far the most popular hash function
- 160-bit, hash.

# SHA1

- A birthday attack on SHA-1 should need  $2^{80}$  hash tests
- In 2005 a  $2^{63}$  attack was found.
- Not really practical, but no-one trusts SHA-1 any more.
- So ... SHA-2

# SHA2

- SHA2 is an improved version of SHA1 with a longer hash.
- 256 or 512 bits: also called SHA256, SHA512.
- Based on SHA-1 it has some of the same weaknesses. So, even though it seems secure the cryptographers aren't happy.

# The SHA-3 Competition

- Submissions opened on October 31, 2008,
- Round 1
  - 13 submissions rejected without comment.
  - 10 withdrawn by authors.
  - 16 rejected for design or performance.
    - Inc. Sony's
- Conference in Feb 2009. 14 scheme picked to go through to round 2.
  - Dropped schemes include
    - Ron Rivest's,
    - Lockheed Martin

# The SHA-3 Competition

- Last year the final round candidate were:
  - BLAKE
  - Grøstl
  - JH
  - Keccak, (Daemen et al. the AES guy)
  - Skein, (Schneier et al. the blog guy)
- Winner expected in 2012.

# Message Authentication Codes

- MACs are hashes with a key.
- You can only make or check the hash, if you know the key.
  - Stops guessing attacks.
- Splitting a key  $K$  into  $K_1$  and  $K_2$   
$$\text{MAC}_K(M) = \text{hash}(K_1, \text{hash}(K_2, M))$$

# Hashes in Java

```
try {
    MessageDigest sha00 =
        MessageDigest.getInstance("SHA1");
    byte[] sha1hash = new byte[40];
    sha00.update(text.getBytes(), 0,
                text.length());
    sha1hash = sha00.digest();
    return(Base64.encodeBytes(sha1hash));
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

# Password Protection

- People reuse passwords and sys admins, can be attackers.
- To protect passwords, only the hash is stored.
- When a user enters a password the system hashes this, and compares it to the true value.
- If an attacker gets the password file, they don't get the passwords.

# Password Cracking

- But: if an attacker gets the password shadow file
  - they can try to guess a password
  - and check if the hash of their guess is in the list.
- Truly random passwords are safe.
- Dictionary words are not.

# Linux\Unix password file

- Password hashes use to be in `/etc/passwd` and public readable!
- Now
  - `/etc/passwd` user info
  - `/etc/shadow` password hashes
- `shadow` can only be read with root level access.

# John The Ripper

- State of the art, free, password cracker.
- <http://www.openwall.com/john/>
- See demo.

# Pre-Computing the Hashes

Idea: computing all the hashes and store them on disk ... then to crack a password just look up the hash.

MD5 128 bit would need  $10^{26}$  terabytes!

Answer: just pre-compute likely passwords, e.g. 8 characters letters and numbers would need 200 terabytes.

# Time-Space Trade Off

Really clever idea: trade off some storage space for some time!

To do this we need to use a function  $R(\cdot)$  that maps hashes to possible passwords.

e.g. if we are looking at passwords that are 8 characters and letters, then

$R(0100102001\dots)$  could take the first 8 possible ASCII letters.

# Time-Space Trade Off

So we have 2 functions:

- $H(\text{password}) \rightarrow \text{hash}$
- $R(\text{hash}) \rightarrow \text{password}$

We can now form chains:

- $H(\text{password}) = 010010\dots$
- $R(010010\dots) = \text{shglafyw}$
- $H(\text{shglafyw}) = 10110\dots$
- $R(10110\dots) = \text{rdffgaai}$

# Time-Space Trade Off

The clever idea: split the hashes into chains, and only store the first and last member of the chain.

Pick  $M$  random passwords and for each one calculate  $K$  steps in the chain ( $M * K >$  no. of passwords).

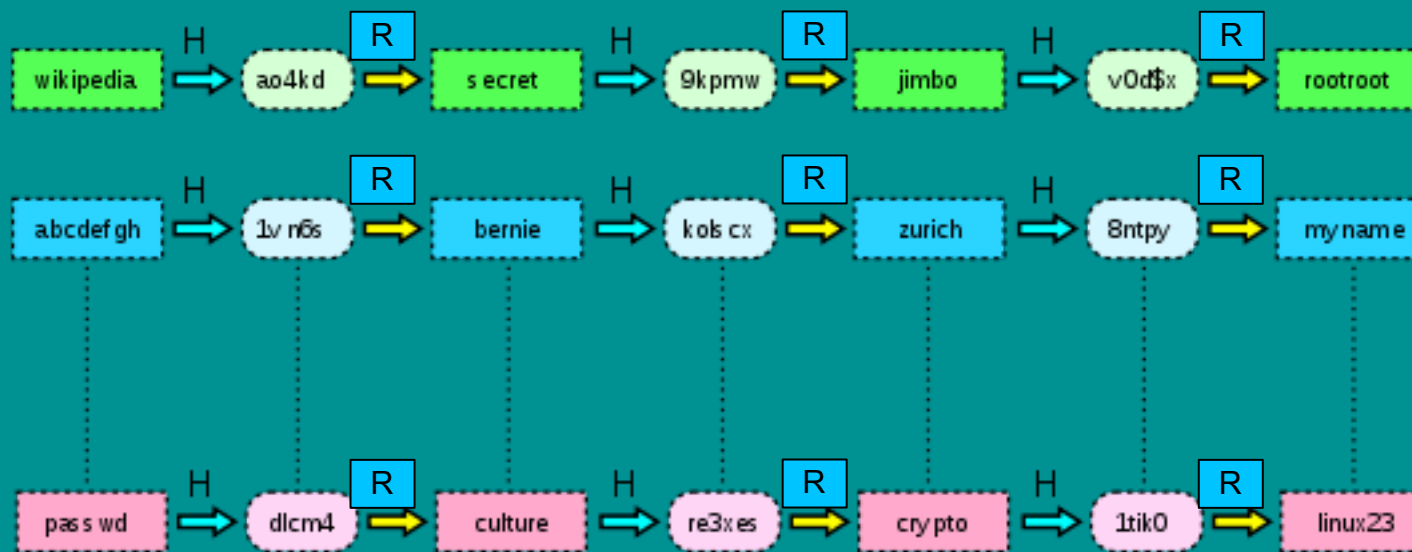
Store just the first and the last elements of the chain.

# Time-Space Trade Off

To find a password:

- Apply R and look for it in the last elements of the chain.
- If it's not there apply H & R and look again.
- Once you find it, go to the start of the chain and apply H & R until you find the password.

# Example

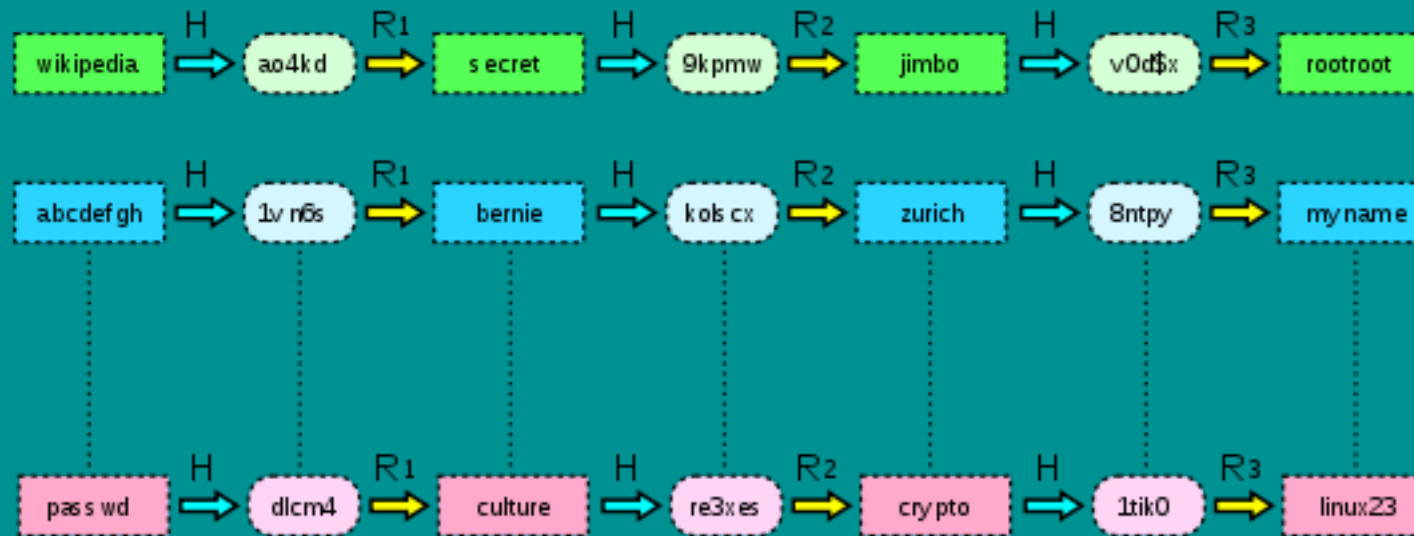


From wikipedia

# Rainbow Tables

- Problem with this method:
  - The R function might map the two hashes to the same password.
  - This leads to repeat values and wastes space
  - It also increases the chance of not finding an answer.
- Answer: Rainbow Tables
  - Use a different R function for each step.
  - Collisions are only a problem if they happen at the same step, which is less likely.

# Example: Rainbow Tables



N.B. different R (functions to map hashes to passwords) used in each step.

# Set papers

*“Making a Faster Cryptanalytic Time-Memory Trade-Off”*

by Philippe Oechslin, introduces rainbow tables.

*“A cryptanalytic time-memory trade off”*

by Martin Hellman.

An earlier paper that describes the more basic, less efficient, time-memory trade off.

# Windows Password Hashes

- Windows stores its password hashes in:  
system32/config/SAM

This file requires Admin level to read.

It is locked and encrypted with a key,  
based on other key values.

– This adds no real security

# Ophacrack

- State of the art, free, rainbow table software
- <http://ophcrack.sourceforge.net/>
- See demo

# Password Injection

- Want access to the system without cracking the password?
- Have access to the hard disk?
- Add your own account, or replace the hash with one you know.

# Better Security: BIOS

- Set a password in the BIOS to stop the computer booting from anything but the hard disk.
- It's very hard to brute force, the BIOS.
- Work around: remove the hard disk from the computer.

# Better Security: Salt

- Add some random public data to the hash.
- Result: pre-computed tables don't help.
- Stops rainbow tables, does nothing to stop brute force attacks.

# Better Security

- Whole disk encryption
  - Encrypt the whole hard drive
  - Key can be brute forced
  - Not safe if the computer is in sleep mode
- “Trusted Computing”
  - Can help, but has its problems
  - See trusted computer lectures

# Some Other Password Cracking Tools

- Cain & Able
  - Windows only,
  - Can sniff passwords from wi-fi
- THC Hydra
  - Can brute force across a network,
  - This will be detected by the network admin
- L0phtcrack (LC6)
  - Like Ophacrack, but not free

There are many others.

# Moral of the story:

- Passwords are a weak point. If you have physical access to a computer you can probably crack the password.
- Use the strongest possible hash for passwords, enforce strong passwords.
- Salt the password list
- Use access control to protect the password list from remote attackers.