Lecture 15: Authentication

Professor Adam Bates
CS 461 / ECE 422
Fall 2019

Goals for Today

- Learning Objectives:
  - Understand the three ways in which we verify identity
  - Describe the challenges with storing passwords and how they can be attacked
  - Provide examples of "what you have" and "what you are"
  - Know the limits (e.g. attacks) of biometric and token-based systems
  - Describe trends in identity and authentication
- Announcements etc:
  - **Midterm October 9th 7pm 1404 Siebel**
  - MP2 Checkpoint #2: **Due Oct 7 at 6pm**

Reminder: Please put away devices at the start of class

What is authentication?

- Authentication binds identity to a subject in a system
- Two step process
  - Identification - establish identity to system
  - Verification - process verifies and binds subject and identity

Verification Method

- Something you know…
  - e.g. *passwords*
- Something you have…
  - e.g. *token*
- Something you are…
  - e.g. *fingerprint*

Password Authentication

- User keeps a secret string (password)
- Something the user knows
- Advantages?
- Disadvantages?

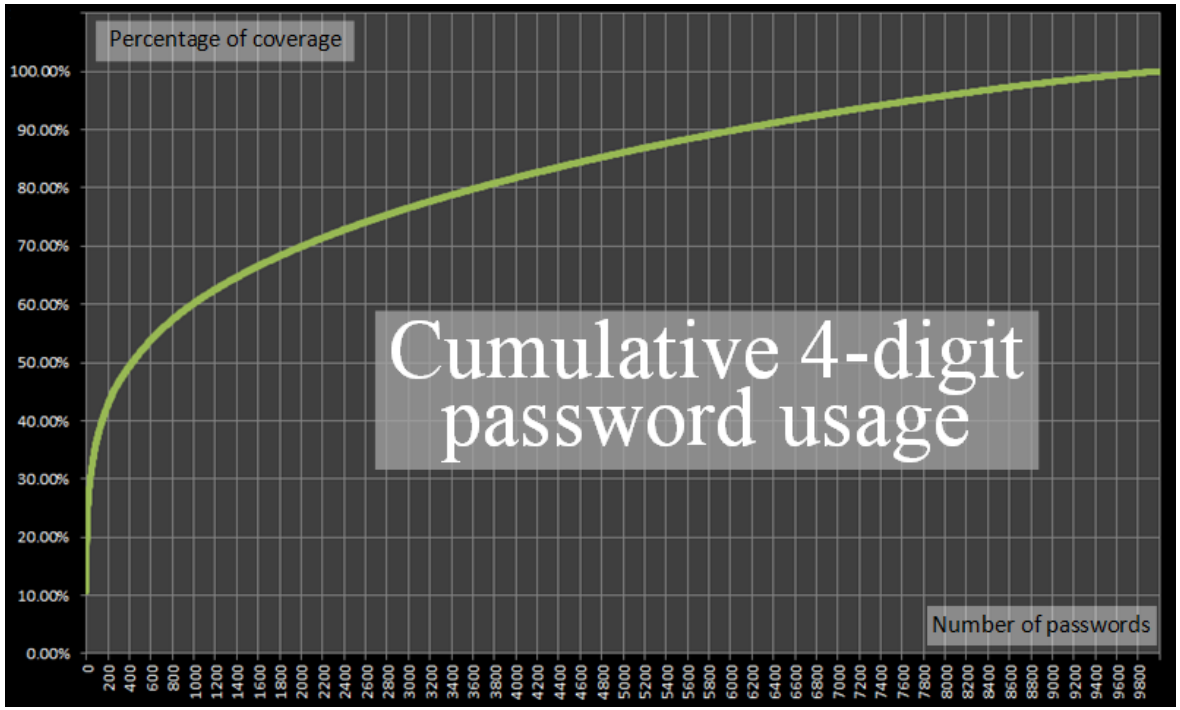
Attacks on Passwords

- Steal from the user
  - Install a keylogger (hardware or software)
  - Find it written down
  - Social engineering/Phishing
  - Intercept the password over network
  - Use a side channel
- Steal from the service
  - Install malware on the web server
  - Dump the password database with SQL injection
- Steal from a third party (password reuse)

Password Guessing

How good are users at picking passwords (or PINs)?

	PIN	Freq
#1	1234	10.713%
#2	1111	6.016%
#3	0000	1.881%
#4	1212	1.197%
#5	7777	0.745%
#6	1004	0.616%
#7	2000	0.613%
#8	4444	0.526%
#9	2222	0.516%
#10	6969	0.512%
#11	9999	0.451%
#12	3333	0.419%
#13	5555	0.395%
#14	6666	0.391%
#15	1122	0.366%
#16	1313	0.304%
#17	8888	0.303%
#18	4321	0.293%
#19	2001	0.290%
#20	1010	0.285%



http://www.datagenetics.com/blog/september32012/

Password Guessing

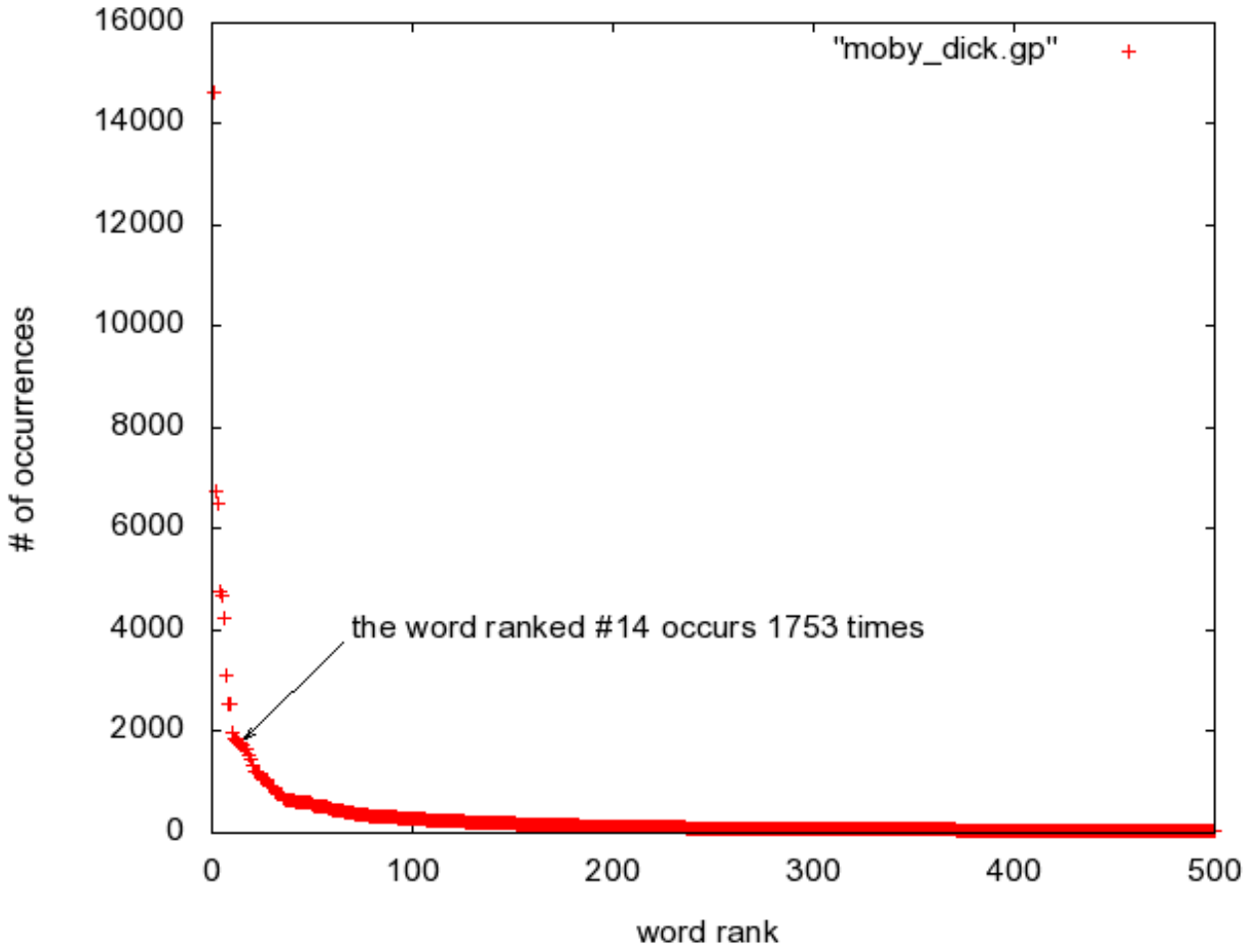
How good are users at picking passwords?

password	freq
password	32,027
123456	25,969
12345678	8,667
1234	5,786
qwerty	5,455
12345	4,523
dragon	4,321
pussy	3,945
baseball	3,739
football	3,682
letmein	3,536
monkey	3,487
696969	3,345
abc123	3,310
mustang	3,289
michael	3,249
shadow	3,209
master	3,182
jennifer	2,581
111111	2,570

https://xato.net/10-000-top-passwords-6d6380716fe0#.lo0geeq99

Password Guessing

How good are users at picking passwords?



http://www.philippeadjiman.com/blog/2009/10/26/drawing-the-long-tail-of-a-zipf-law-using-gnuplot-java-and-moby-dick/

Secure Passwords

- Uneven distribution makes guessing easier
- Passwords should be uniformly distributed
  - All characters in password chosen with equal probability
- Passwords should be long
  - Longer password = larger brute force search space
- Passwords should never be reused
- Passwords chosen randomly are difficult to remember
  - Tradeoff of security vs. convenience

Storing Passwords

Storing Passwords

- Password database is highly sensitive
- We should never store plaintext passwords
- Store something that lets user prove they know the password

Hash functions

- Input – data of an arbitrary size
- Output – fixed length
- Same input always produces the same output
- One way function – cannot deduce input from output
- A “fingerprint” for the input
- Examples: MD5, SHA-1, SHA-256, SHA3-512

- md5("welcome")= "M3ULPLtx$K6.aFwEvavGgNx8SGe9fq"

How can we use these to store passwords securely?

Password Hashes

- We store a database of password hashes
- Hash function is also called a key derivation function (KDF)
- e.g. /etc/shadow on UNIX:
  - `rcunnin2:$6$vb1tLY1qiY$M.1ZCqKtJBxBtZm1gRi8Bbkn39KU0YJW1cuMFzTRANcNKFKR4RmAQVk4rqQQCkaJT6wXqjUkFcA/qNxLyqW.U/:15405:0:99999:7::`

Password Hashes

- If we use strong passwords…
- … and those passwords are only stored as hashes…
- … our passwords are secure, right?

Password Cracking

- Attacker exfiltrates a hashed password database. Now what?
- GOAL: Find the passwords for the hashes
- METHOD: Guess and check
  - Guess a password
  - Hash it
  - Check if the hash is in our password database

Password Guessing

- Brute force search through all possible passwords in order
- Use a dictionary (/usr/share/dict/words)
- Use a dictionary of common passwords
- Combine dictionary with common passwords and heuristics (e.g. p@$$w0rd and password123)
- Use statistical models of known user passwords (e.g. Markov models)

Password Cracking

- Easy to parallelize
  - hash password guess, compare to entire hash database
- Commonly done with arrays of GPUs (or FPGAs)
- Same "work" as Bitcoin mining
  - Looking for input with desired hash output

Lookup Tables

- Guessing every password still takes time, and many passwords are common… can we just pre-compute?
  - an example of a time/space trade-off
- With small enough password space |A|
  - store all possible mappings between password and hash
  - 'reverse lookup' the hash to recover password
- For MD5 and 8-character alphanumeric passwords:

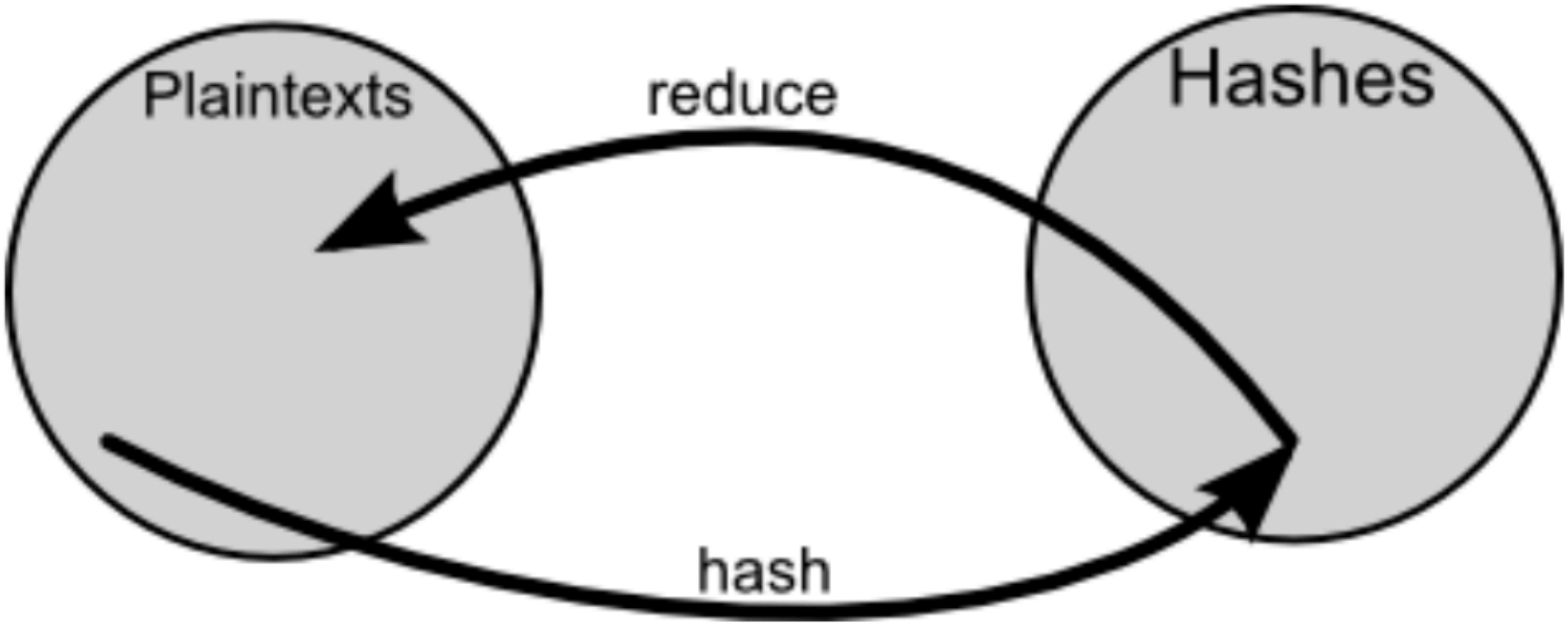
```
62^8= 218,340,105,584,896 entries
8 bytes+128 bits = 24 bytes/entry
```

Lookup table is 5.24 petabytes!

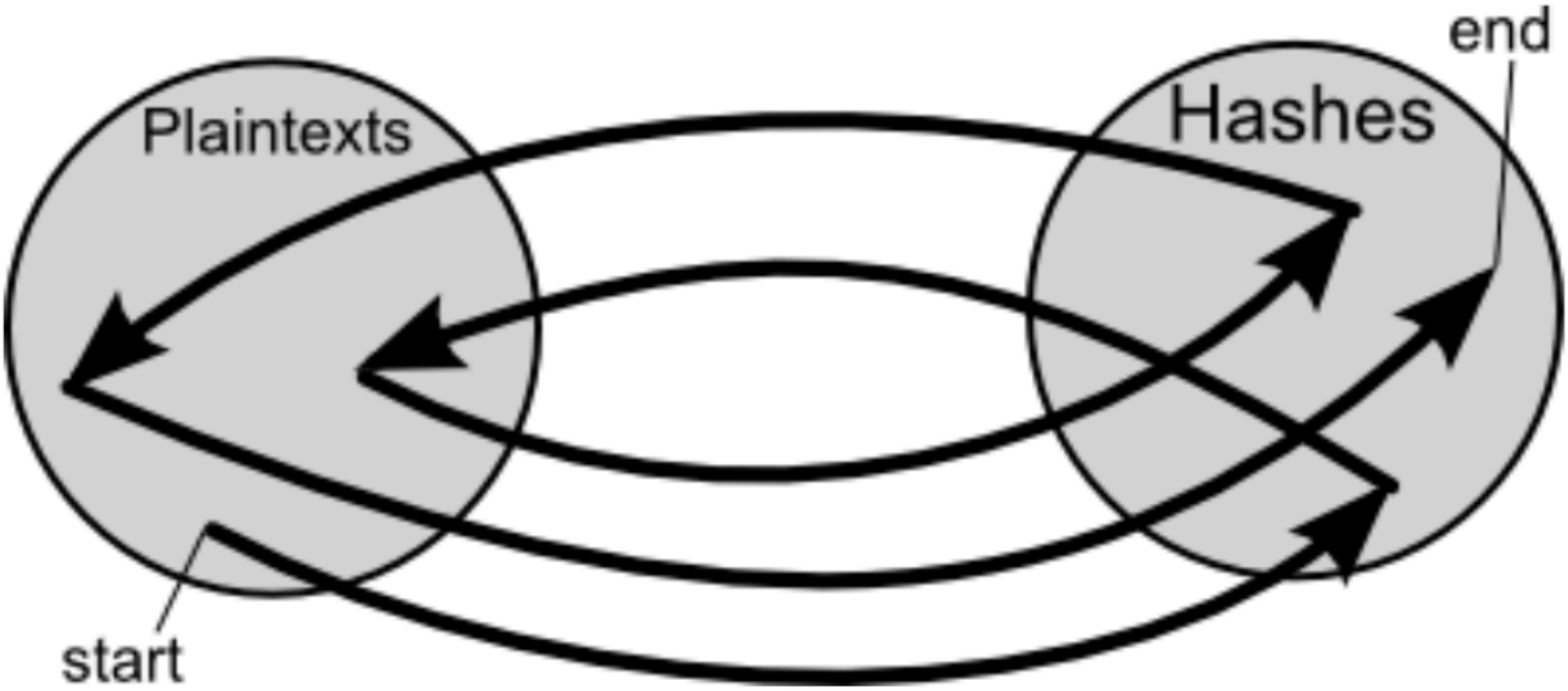
Rainbow Tables

- Huh. Our lookup table for |A| is too big to store.
- Workaround — is there a gradient between Max(Time Savings) and Max(Space Savings)?
- Rainbow Table:
  - Requires reduction functions R that map hashes to passwords
    - Given a hash in |C|, return a password in |A|
    - R = {r: C→ A}
  - R is **not** the inverse of the cryptographically-strong hash function!

Rainbow Tables



Rainbow Tables

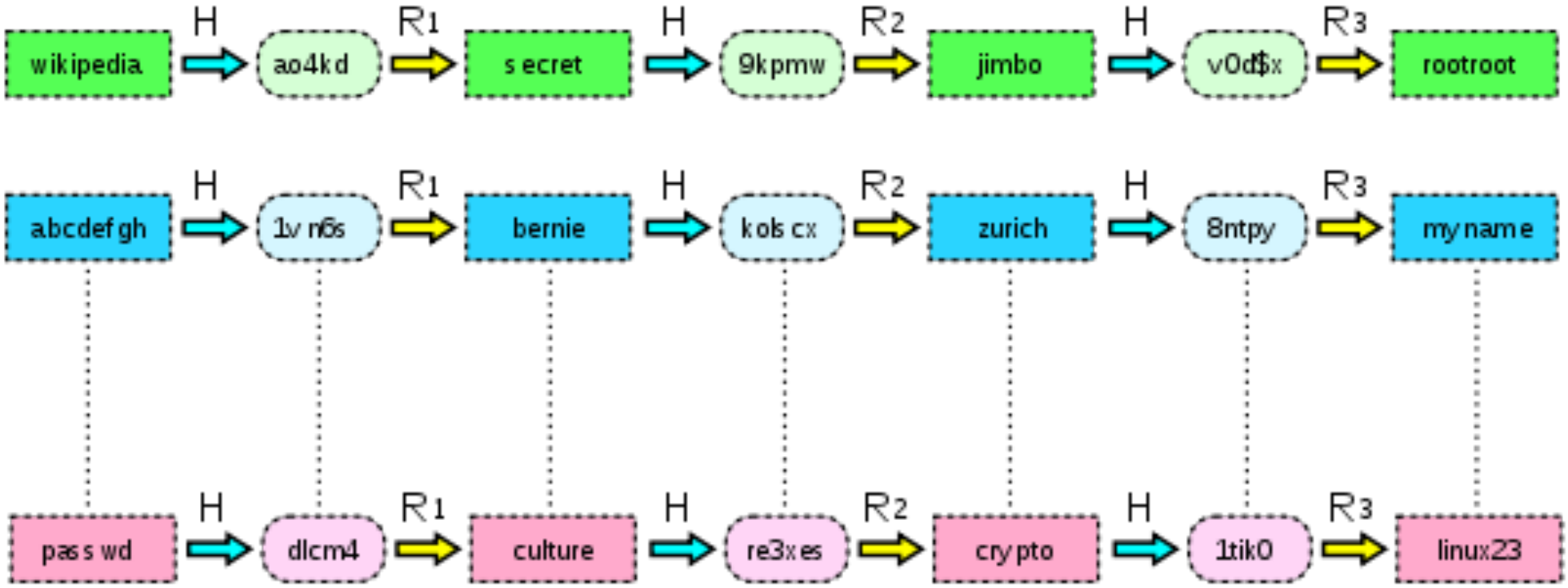


Generating Rainbow Tables

- Start with a random plaintext password
- Repeatedly hash and reduce to create a chain
- Record start and end of the chain
- GOTO 1

```
iaisudhiu -> 4259cc34599c530b1e4a8f225d665802
oxcvioix -> c744b1716cbf8d4dd0ff4ce31a177151
9da8dasf -> 3cd696a8571a843cda453a229d741843
[...]
sodifo8sf -> 7ad7d6fa6bb4fd28ab98b3dd33261e8f
```

Storing Rainbow Tables



Storing Rainbow Tables

Using Rainbow Tables

- Input: a rainbow table and hash we want to crack
  - Is the hash in the table? If so, break.
  - Reduce hash to plaintext. Hash new plaintext.
  - GOTO 1
- Match?
  - We have (maybe) identified the chain with our plaintext
  - Reproduce the chain to (maybe) identify the desired plaintext password

Is it worth it today?

Home | Hardware | Software | Engineering | Support | Why Sagitta? | Company

Brutalis

Highlights

1. World's fastest 8-GPU system -- 14% faster than 8x GTX Titan X OC!
2. First system to break 330 GH/s on NTLM -- will easily break 350 GH/s with OC!
3. First system to break 200 GH/s on MD5!
4. Driver 367.18 is still half-baked garbage, so we still can't do any overclocking >:/
5. GPU temperatures stable at < 73C in 23C ambient environment

Ba

The base
and comp
applicabl

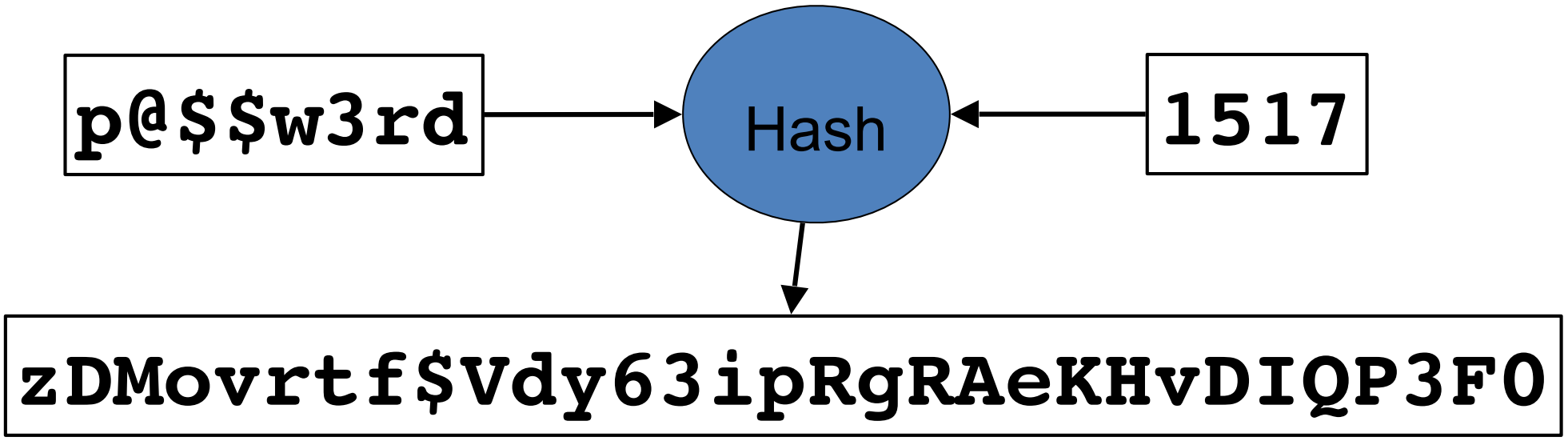
GPU	
Processor	2x Intel Xeon E5-2620 v4, 2.1 GHz Eight-core
Memory	64 GB DDR4 RDIMM ECC
Storage	2x 512 GB SATA/600 SSD in RAID-1
Operating System	Ubuntu Server 14.04 LTS
Software	Hashstack Pro
Price as Configured	21,169.00 USD

Storing Passwords

- Rainbow Tables (or blinding brute force guess speeds) make guessing user's passwords easy
- It seems like hashed passwords are (almost) as vulnerable as plaintext storage
- Are we doomed? How to fix?

Salting Password Database

- Generate and store a random number (nonce) for each password (salt)
- Concatenate password and salt to compute hash
- Effectively a unique hash function for each password



Salting Password Database

- Makes rainbow tables worthless
  - Table would be prohibitively large
- Protect commonly used/reused passwords
  - Each password must be cracked individually
  - h(p||s1) ≠h(p|| s2)
  - Effectively destroys value of dictionaries for guessing or RT's
- Note: salt works even if the attacker knows the value of the salt
- Concept of a secret nonce is called "pepper"
  - Not widely accepted
  - Salt and then encrypt the database

Storing Passwords

- Even with salted passwords, attacks can still issue lots of guesses.
- Could break lots of passwords eventually, one-at-a-time
- Is there anything else we can do to frustrate guesses?

Blowfish

- Block cipher designed by Bruce Schneier in '93
- Requires long preprocessing setup step for each key
  - key scheduling to prep substitution boxes
  - processes about 4kb of data
- Not widely used for encryption, but…

bcrypt

- A hash function for passwords
- Designed based on Blowfish, but with a more expensive setup function
- Runs Blowfish multiple rounds
- Not designed to be cryptographically secure, designed to require work
- Required work is configurable

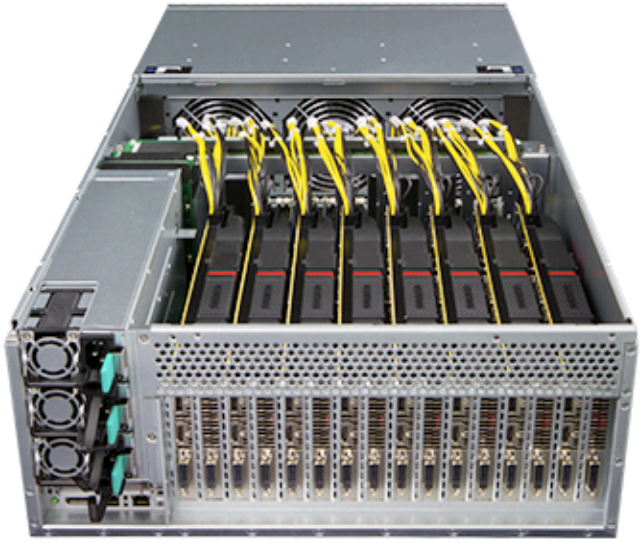
But guessing is still fast…

Home | Hardware | Software | Engineering | Support | Why Sagitta? | Company

Brutalis

Brutalis is an eight-GPU monster, clawing its way through hashes at unprecedented speeds. Providing up to eight Nvidia GTX GPUs, two Intel Xeon E5-2600 v4 CPUs, up to 3TB ECC memory, and up to 18TB of SSD storage, the Brutalis is the fastest, meanest, most hardcore password cracker money can buy. Ships with a 3-year warranty and full commercial support.

Base configuration price: 21,169.00 USD



… and can be made faster.

High-speed implementation of bcrypt password search using special-purpose hardware

2 Author(s) Friedrich Wiemer ; Ralf Zimmermann **View All Authors**

2 Paper Citations

334 Full Text Views

Abstract

Document Sections

I. Introduction

II. Background

III. Implementing Bcrypt on Fpgas

IV. Results

V. Conclusion and Future Work

Abstract:
Using passwords for user authentication is still the most common method for many internet services and attacks on the password databases pose a severe threat. To reduce this risk, servers store password hashes, which were generated using special password-hashing functions, to slow down guessing attacks. The most frequently used functions of this type are PBKDF2, bcrypt and scrypt. In this paper, we present a novel, flexible, high-speed implementation of a bcrypt password search system on a low-power Xilinx Zynq 7020 FPGA. The design consists of 40 parallel bcrypt cores running at 100 MHz. Our implementation outperforms all currently available implementations and improves password attacks on the same platform by at least 42%, computing 6,511 passwords per second for a cost parameter of 5.

Published in: 2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)

Date of Conference: 8-10 Dec. 2014

INSPEC Accession Number: 14916762

Date Added to IEEE *Xplore*: 09 February 2015

DOI: 10.1109/ReConFig.2014.7032529

scrypt

- Another hash function for passwords
- Designed to require a lot of memory to compute
  - Must generate a large random bit vector and randomly access it during computation
- Also used as a proof of work hash for Litecoin and Dogecoin

Argon2

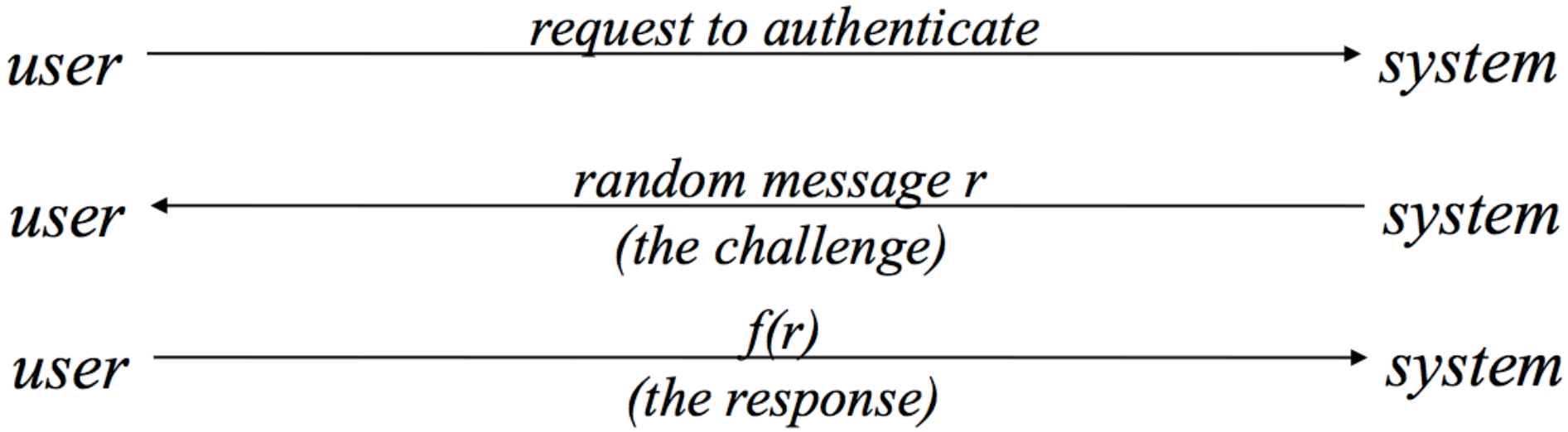
- Also a password hashing algorithm
- Won the Password Hashing Competition in 2015
- Can configure the amount of time, memory, and parallelism required to compute Argon2

Token-Based Authentication

- Something the user **has**
- Static memory cards
  - Read only
  - e.g. ATM card/Credit Card
  - Vulnerable to replay attack

- Smart card
  - Storage and computation
  - Enables challenge-response or one-time password
  - Protects against replay attack
- Hardware tokens
  - One time password

Challenge-Response



One-time-password

- Smart card can also implement one-time password scheme
  - S/Key is one such scheme:
    - Start with a random seed
    - Hash the current seed to produce the next
  - Basically, share a pseudorandom number generator with shared state
  - Use the hash outputs in reverse order
- Hardware Token
  - Time-based One-time Password algorithm (TOTP) is an extension of the HMAC-based One-time Password algorithm (HOTP)
  - More on hashing (i.e., HMACs) later

Disadvantages

- Token can be lost, stolen, or counterfeited
- Requires an individual physical token
- Requires an extra step (inconvenient)
- Hardware can be expensive
- Steal the seed!

THE WALL STREET JOURNAL.

TECHNOLOGY

Security 'Tokens' Take Hit

RSA Offers to Replace Its SecurIDs or Provide Monitoring for Nearly All Customers

By Siobhan Gorman And Shara Tibken

June 7, 2011

Biometric Authentication

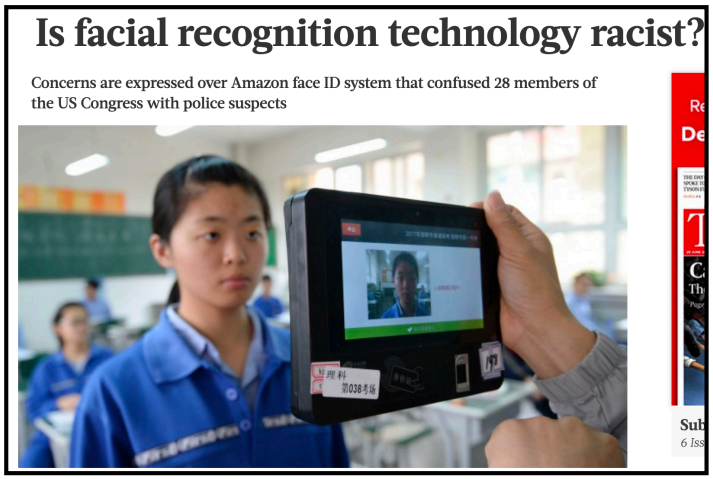
- Something the user **is** or **does**
- Derive a signature from biological features of user
  - Voice, fingerprint, face, retina, handwriting, gait
- Advantages?
- Disadvantages?

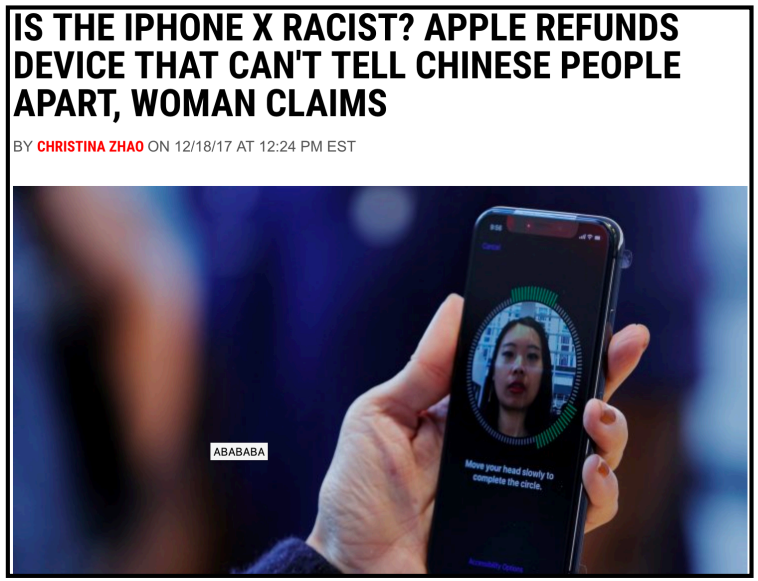
Disadvantages

- Imprecise measurements require approximate matching
  - Essentially a machine learning task… biased data creates biased authentication outcomes
  - False negatives and false positives have a cost

Disadvantages

- Imprecise measurements require approximate matching
  - Essentially a machine learning task… biased data creates biased authentication outcomes
  - False negatives and false positives have a cost
  - E.G., your biometric authentication mechanism may be racist, sexist, ablist, agist…





Disadvantages

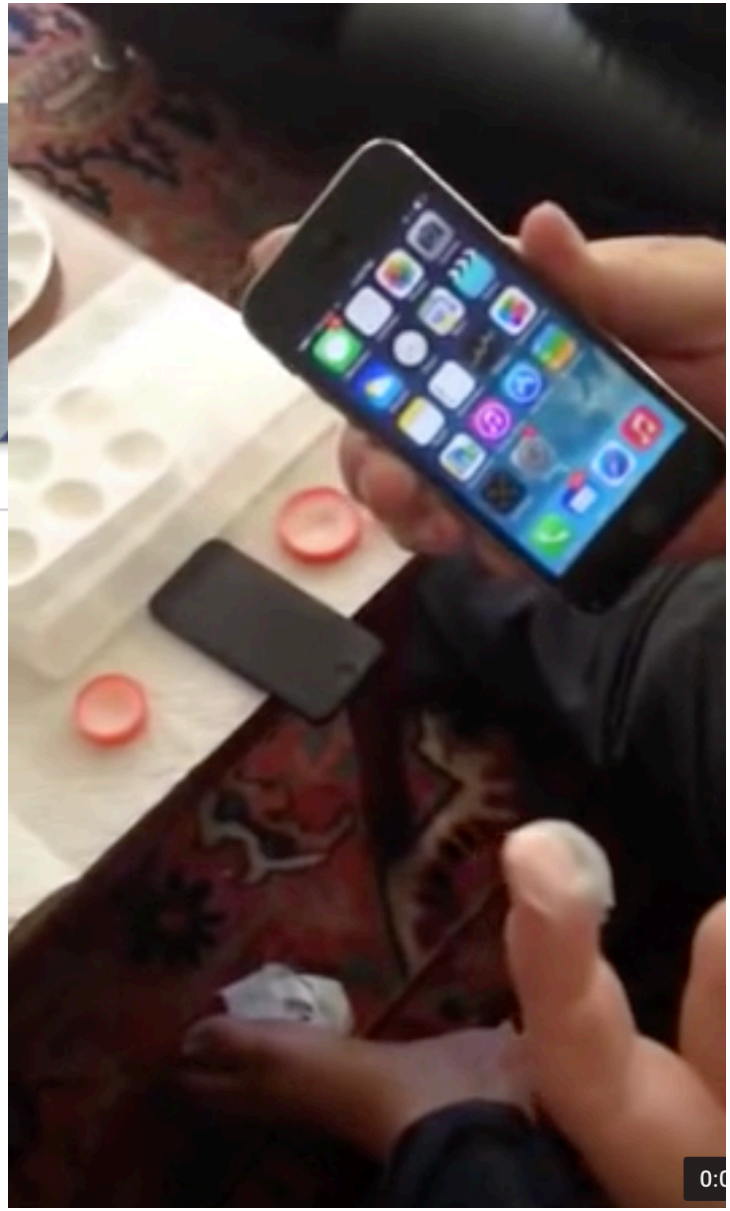
- Imprecise measurements require approximate matching
  - Essentially a machine learning task… biased data creates biased authentication outcomes
  - False negatives and false positives have a cost
- Measurements change over time
- Poor accessibility
- Cannot be replaced or concealed
- Replay attacks/spoofing possible
- Can be legally compelled to provide biometrics

OPM Breach

KrebsonSecurity
In-depth security news and investigation

Congressional Report Slams OPM on Data Breach

nassive data breach at the **U.S. Office of Personnel Management** (OPM) that ed background investigations and fingerprint data on millions of Americans was the of a cascading series of cybersecurity blunders from the agency's senior leadership on to the outdated technology used to secure the sensitive data, according to a lengthy t released today by a key government oversight panel.



Facial Recognition

Browse Journals & Magazines > IEEE Transactions on Informat... > Volume: 9 Issue: 7

Spoofing Face Recognition With 3D Masks

Purchase or Sign In to View Full Text

14 Paper Citations

1588 Full Text Views

Related Articles

- Face Verification With Local Sparse Representation
- 3D Assisted Face Recognition: Dealing With Expres...
- Depth Estimation of Face Images Based on the Cons...

2 Author(s)

Nesli Erdogmus ; Sébastien Marcel

View All Authors

Abstract | Authors | Figures | Reference

Abstract:

Spoofing is the act of masquerading as a valid user by falsify
attacks (presentation attacks) is still an open security issue i
threat, since it is particularly easy to access and reproduce.
various algorithms have been proposed to detect them. Mair
videos on mobile devices, a significant portion of these studi
However, with the advancements in 3D reconstruction and p
aim to inspect the spoofing potential of subject-specific 3D fa
more complex attack type. In order to assess the spoofing pe
various texture-based countermeasures using both 2D and 2
sets: the Morpho database which is not publicly available an

2 Factor Authentication (2FA)

- Something you **have** AND something you **know**
- Either factor is useless without the other
- Chip and PIN
- Commonly implemented in mobile phones via SMS
  - Disadvantages:
    - ONE device (if hacked)
    - SMS is easy to redirect
    - ONE point of failure for SE (phone company)

Multifactor Authentication

- Next level 2FA
- Combination of biometrics, knowledge, and possession

Behavior Profiling

- Track access behavior of users
  - Systems used
  - Times and locations when active
  - Typical usage
- Look for anomalous or fraudulent behavior
- "Why is this guy who was in Iowa 2 minutes ago logging in from Nigeria?"
- Used in fraud prevention

Password Managers

- Application that maintains (and generates) passwords
- Examples: LastPass, KeePass, DashLane, 1Password
- Advantages?
  - Can handle random passwords
  - Can create unique passwords for every website and service
- Disadvantages?
  - One point of failure
  - Requires a strong password (could be snooped)
  - Could be hacked (only as secure as the password manager)
  - Inconvenient (doesn't work for some sites, set up time, etc.)

Password Managers

one point of failure…

Trend Micro password manager had remote command execution holes and dumped data to anyone: Project Zero

Google's Project Zero discovered multiple trivial remote code execution vulnerabilities sitting within a password manager installed by Trend Micro as default alongside its AntiVirus product.

By Chris Duckett | January 12, 2016 -- 01:32 GMT (17:32 PST) | Topic: Security

in 101

A password management tool installed by default alongside Trend Micro AntiVirus was

RELATED STORIES

Security

ClixSense data breach exposes personal information of million of subscribers

https://www.zdnet.com/article/trend-micro-password-manager-had-remote-command-execution-holes-and-dumped-data-to-anyone-project/

Single Sign-On (SSO)

- Login to trusted 3rd party, who vouches for user identity
- Examples: Facebook Connect, OAuth, OpenID
- Pros and cons similar to Password Managers
- Third party can track users…

To Learn More …

- Books
  - Stalling and Brown, Chapter 3
  - Pfleeger and Pfleeger, Chapter 2
  - Bishop, Chapter 11
  - Anderson, Chapter 2
  - Easttom, Chapter 9
  - Vacca, Chapter 37
- Papers
  - Analysis of Credential Stealing Attacks in an Open Networked Environment - Sharma*