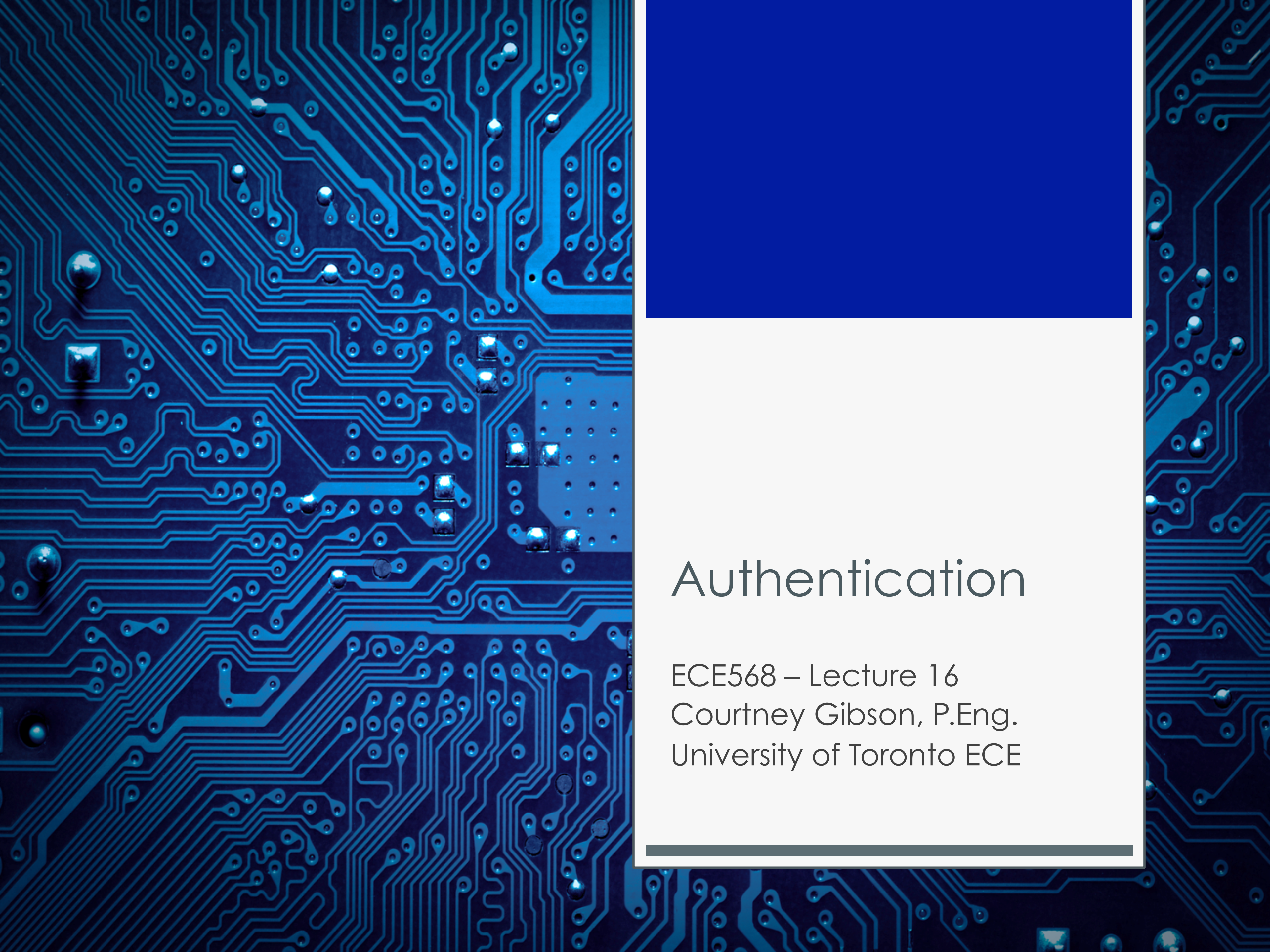


Authentication

ECE568 – Lecture 16
Courtney Gibson, P.Eng.
University of Toronto ECE

Outline

User Authentication

Passwords

- Password Storage
- Salts
- One-Time Passwords

Multi-Factor Authentication

User Authentication

Authentication

Authentication is used for two main purposes

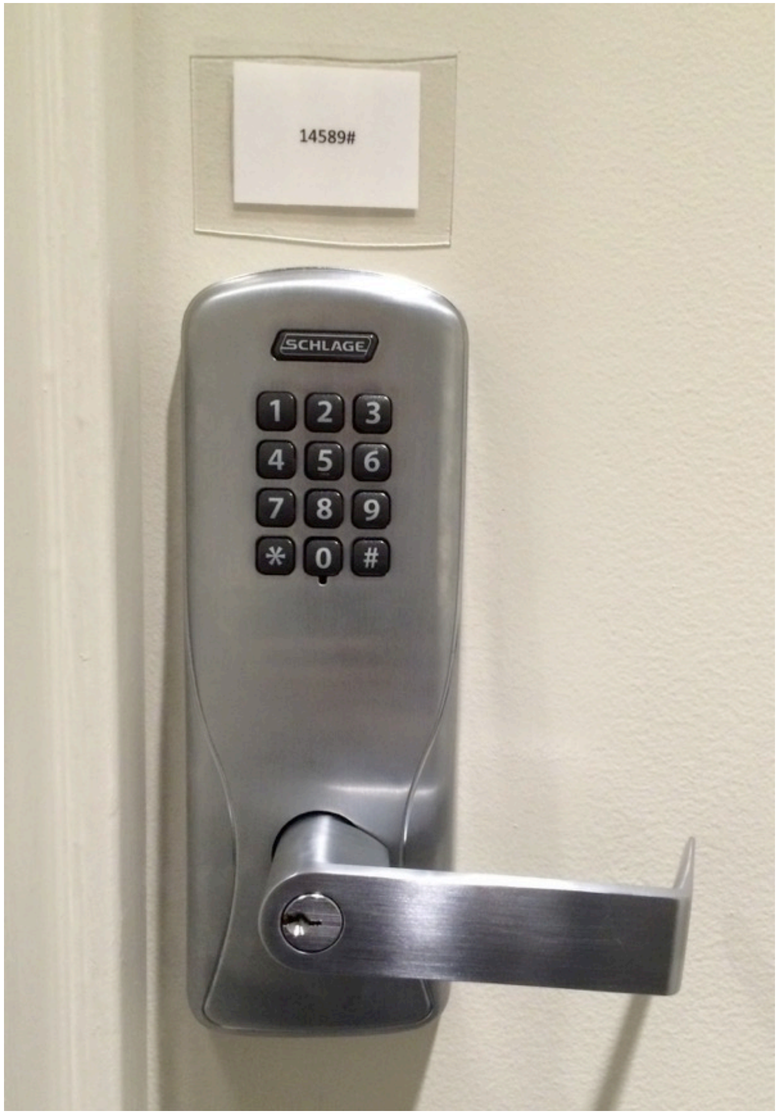
- **Data authentication**
  - Data integrity
    - Proves that data has not been modified
    - Uses what technique?
  - Data origin authentication
    - Proves the origin of some data
    - Uses what technique?
- **User authentication**
  - Allows a principal (user or machine) to prove their identity to a system

Passwords

Passwords

Passwords are commonly used for user authentication, but have several problems

- People can't remember them, so they pick **easy** passwords or write them down where attackers can find them
  - Attacker has a list of commonly used passwords and tries them (**dictionary attack**)
- People tend to use the **same** passwords for many systems
  - A weak system that gets compromised can lead to a strong system getting compromised
- The authentication is **not mutual**
  - When a user enters a password, they usually don't know if they are sending their password to the right system
  - Examples of problems?


14589#
SCHLAGE
1
2
3
4
5
6
7
8
9
*
0
#

Storage of Passwords

Systems usually don’t store passwords in the clear

- Why is this a good practice?
  - If an attacker somehow got the password file, they would know the passwords of all users
- The passwords are hashed using a one-way hash, and only the hash is stored

```
tom:$1$r4ySvmPT$uG3vAOKx9oe84UyaU6MEv0:13090:0:99999:7:::
```

  - If attacker gets the hash, she doesn’t know the password without guessing or reversing the hash
- Login procedure
  - User provides input password
  - System calculates the hash of the input password, and compares this value with the one in the password file

Storage of Passwords

/etc/passwd

```
cgibson:x:507:100:Courtney Gibson:/home/cgibson:/bin/bash
```

/etc/shadow

```
cgibson:qDjGf7984zb3.:12975:0:99999:7:::
```

Interesting note: the Unix `crypt` function, used for traditional password hashing, ignores everything after the first eight characters.

Password Salts

Suppose an attacker steals the password file and breaks one password (*i.e.*, attacker finds a password **p**, given **H(p)**)

- Then the attacker has found the password for all users using the password **p**, since **H(p)** will match
- This problem can be prevented by adding a random value called a **salt** (different for every user) to the password, before hashing:
  - User 1: crypt("password" + "**aa**") = `jfMkNH1hTm2`
  - User 2: crypt("password" + "**bb**") = `v4NvUbQpMC2`
- Salt value is stored in the password file
  - Do salts make it harder to break a single password?

One-Time Passwords

Static passwords can be broken given enough time

- A **one-time password** changes every time it is used, reducing this risk greatly
- One-time passwords are often implemented using **challenge-response authentication**
  - One party presents a question (“challenge”)
  - Other party must provide a valid answer (“response”)
- Example:
  - Server encrypts ‘n’ using shared key, sends E(n) to client
  - Client decrypts ‘n’, adds 1, and send E(n+1) to server
  - Client is authenticated if server is able to decrypt the message and get ‘n+1’

Multi-Factor Authentication

Multi-factor Authentication

Authentication, in general, uses a piece of information about the user, called an **authentication factor**

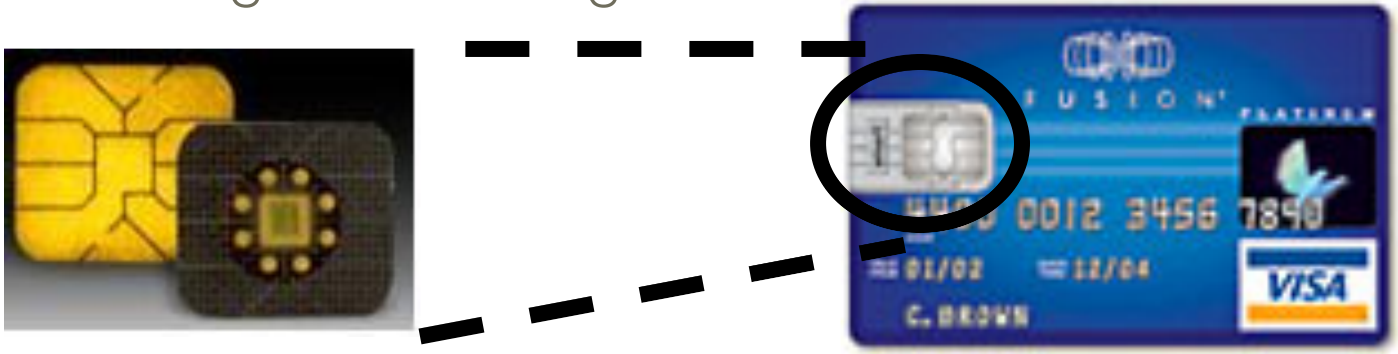
- An authentication factor can be
  - Something the user **knows** (*i.e.*, password)
  - Something the user **has**
  - Something the user **is**
  - Something the user **can do**
- **Multi-factor authentication** provides better security by using multiple authentication factors

Something the User Has: Smart Cards

Smart cards are a good option for a security token

  - Smart card contains secure microcontroller that is hardened against tampering
  - Contains keys and performs cryptographic operations

- How can the card be used for authentication?
- Have the card sign a randomly generated string
  - Why random?
  - Who generates string?



Something the User Is: Biometrics

Biometrics involves identifying a person by measuring some physical aspect of that person

- Common biometrics techniques are based on using:
  - **Fingerprints:** Every person’s fingerprint is fairly unique, a product of genetics and environment
  - **Iris:** Scans the pattern of a person’s iris (pretty accurate)
  - **Retina:** Scans the pattern of blood vessels at the back of the eye (pretty accurate)
  - **Hand Geometry:** Matches the shape of your hand
  - **Gait Recognition:** Recognize people by the way they walk (non-intrusive)
  - **Facial Recognition:** Recognize faces (non-intrusive)
  - **Speech Recognition:** Recognize people by their voices and the way the say things

Using Fingerprints

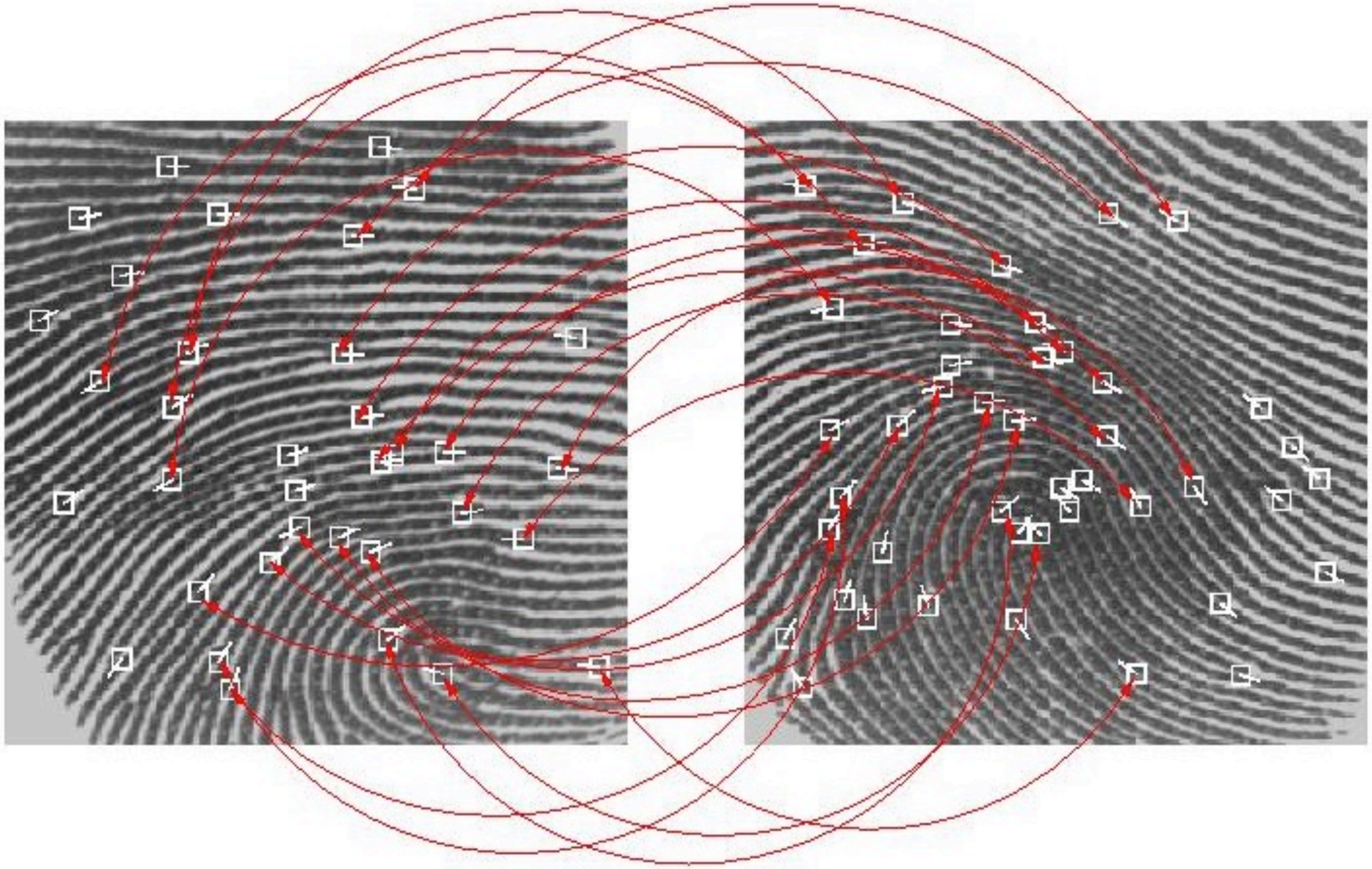
Challenges in using fingerprints are:

- Getting a high quality image to compare
- Accurately comparing the image

Two general algorithms used for comparisons are:

- **Minutiae-based**
  - The algorithm picks out minutiae points, which can be ridge endings or ridge bifurcation (splitting)
  - The relative positions of these are compared
  - Does not take into account global fingerprint features
  - Requires high quality image to extract minutiae points well
- **Correlation**
  - The entire fingerprint is compared
  - This algorithm suffers inaccuracies arising from rotation or translation

Finding and Matching Minutiae



Something the User Can Do: Turing Test

Often a remote site wants to check whether there is a human on the other end

- In 1950, Alan Turing postulated a test that could differentiate between a machine and a human
  - Given a means of communicating only through written language (*i.e.*, a terminal, chat client, etc.), can a human tell if the other end is a human or a computer?
- This test was a way to decide whether machines can think…
- While Turing's paper had much larger implications, the concept of a **test** to tell if something is a human or a computer still applies

Something the User Can Do: Turing Test

Previously, setting up an account required a user to physically meet a sys admin

- Today, that is often not feasible
- Common solution is to use a **CAPTCHA** (**C**ompletely **A**utomated **P**ublic **T**uring Test to tell **C**omputers and **H**umans **A**part)

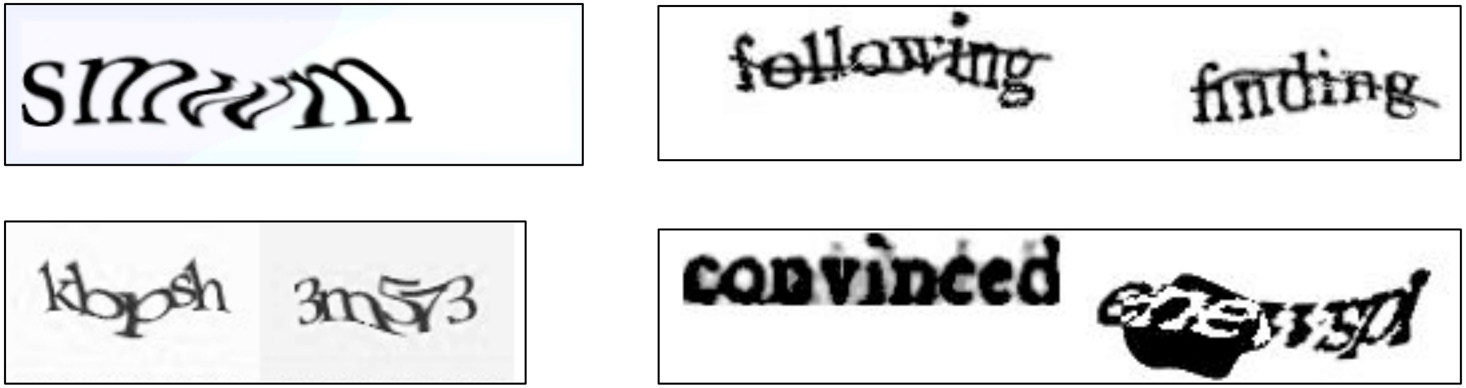


Image Source: Wikimedia Commons

reCAPTCHA

200 million CAPTCHAs are solved per day: equivalent to 150,000 hours of work

- The reCAPTCHA project provides a free CAPTCHA engine: aims to improve the accuracy of scanned historical texts
- Each new word that cannot be read correctly by OCR is given to a user in conjunction with another word for which the answer is already known. User is asked to read both words: if they solve the one for which the answer is known, the system assumes their answer is correct for the new one. The system gives the new image to a number of other people to determine, with higher confidence, whether the original answer was correct.

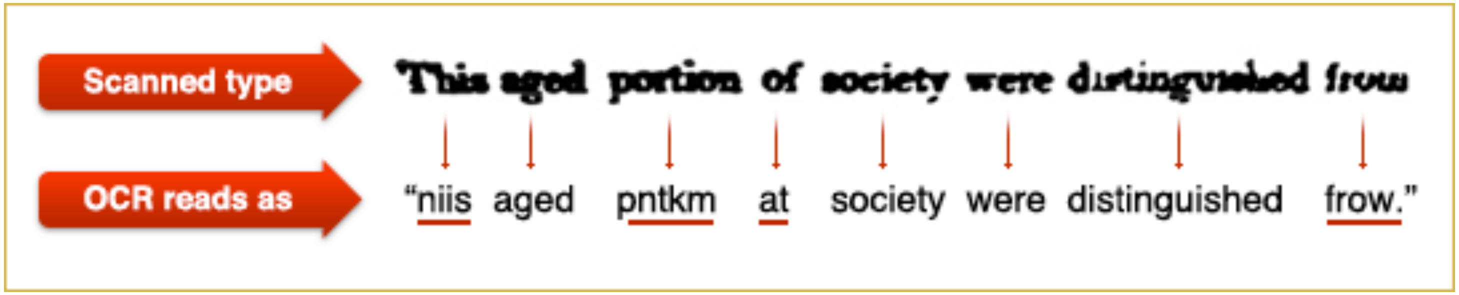
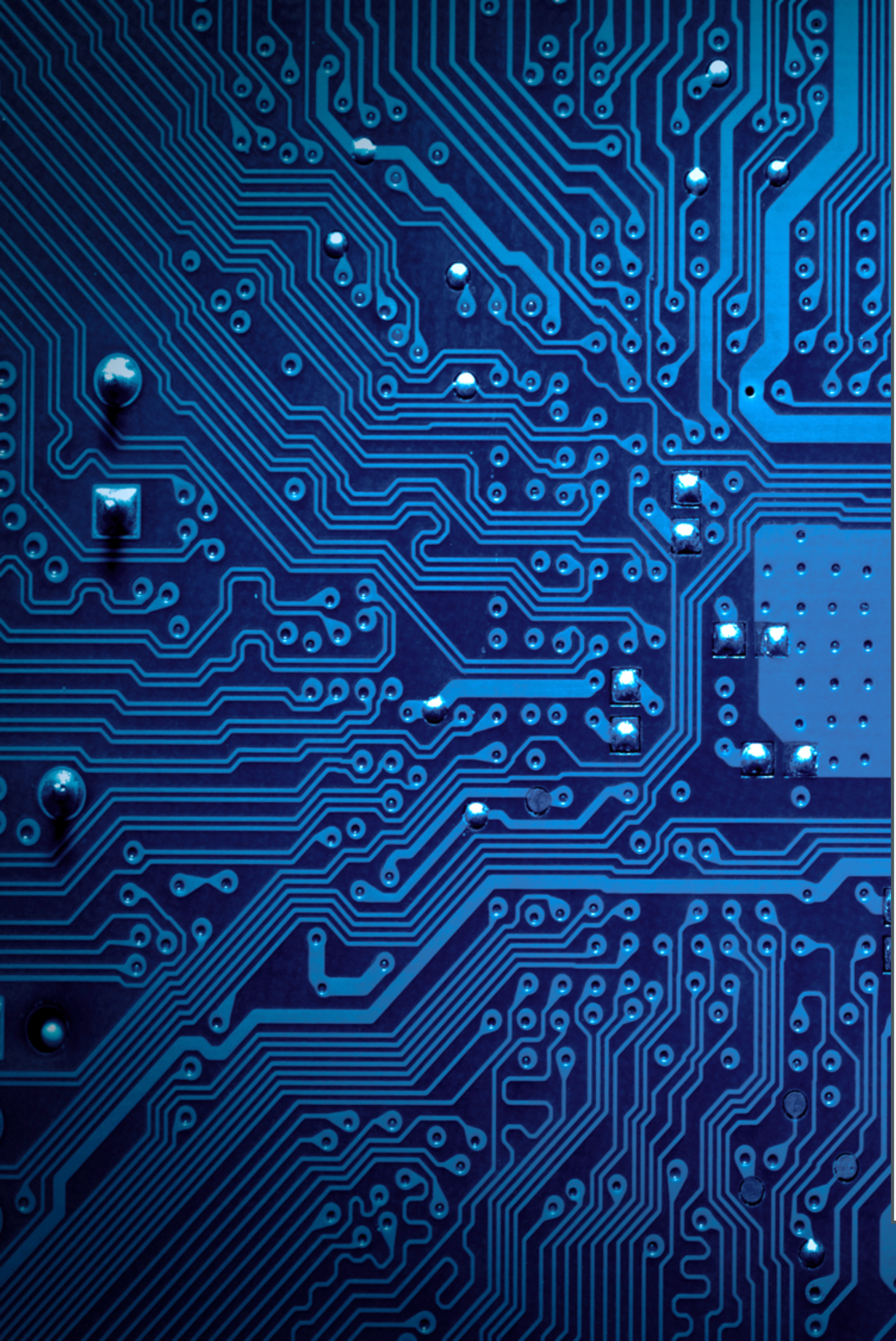

Scanned type
This aged portion of society were distinguished from
OCR reads as
"niis aged pntkm at society were distinguished frow."

Image Source: Google.com

Strength of CAPTCHA's

- CAPTCHA's have many nice properties
  - No human tester is needed, since a computer can test if the other end is a computer or not quite reliably
  - Relatively easy to use (though getting worse)
- Unfortunately, CAPTCHA's have been getting weaker and alternatives will likely have to be found
  - Computer vision researchers have improved algorithms to the point that they are nearly on-par with humans
    - Analysis is slow, but is successful
  - One of the largest botnets tracked in 2008 consisted of 1.5 million infected computers being used to automate breaking CAPTCHAs
  - Can also trick humans into posting answers to CAPTCHA by copying the graphics to another page



Questions?