CIS 4930: Secure IoT

Prof. Kaushal Kafle

Lecture 21

Derived from slides by Adwait Nadkarni, William Enck, Micah Sherr and Patrick McDaniel

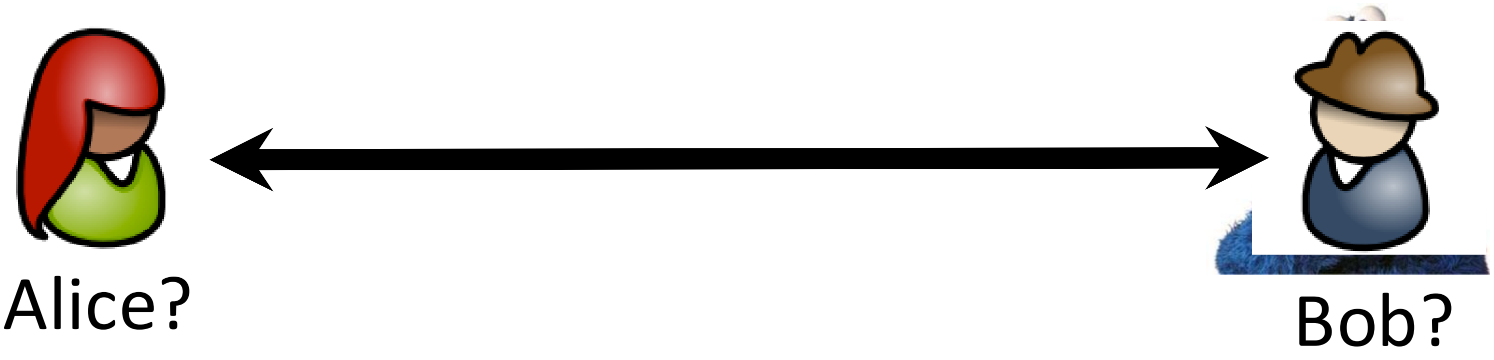
Class Notes

- **2 Reminders:**

  1. **Homework 4 due today.**
  2. **Student Assessment of Instruction**

     Respond to the course assessment survey.

User Authentication



Authentication

What is Identity?

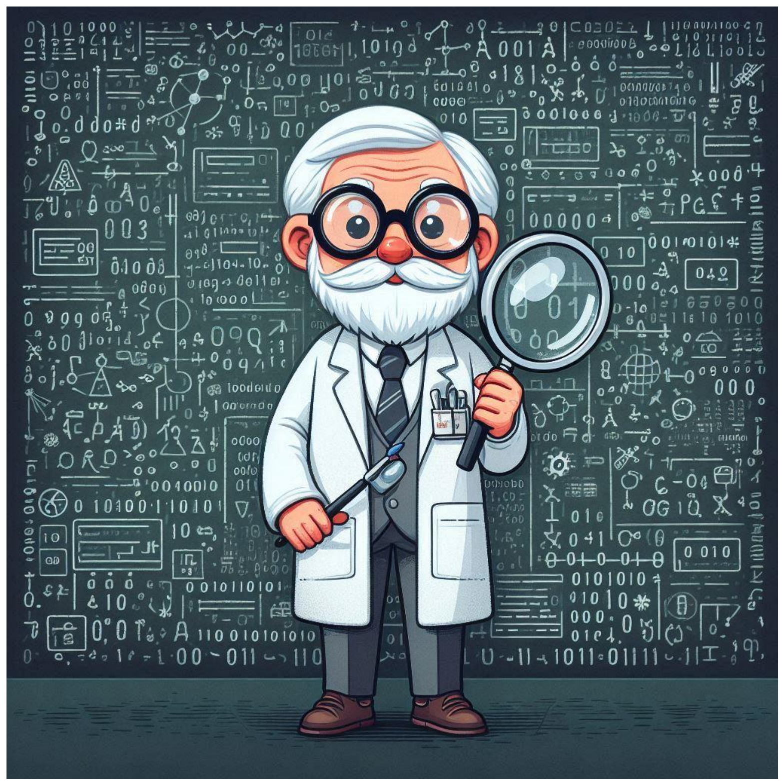
- That which gives you access (your credential) … which is largely determined by context
  - We all have lots of identities
  - Pseudo-identities
- Really, determined by who is evaluating credential
  - Driver's License, Passport, SSN prove …
  - Credit cards prove …
  - Signature proves …
  - Password proves …
  - Voice proves …

- ***Exercise***: Give an example of bad mapping between identity and the purpose for which it was used.

Three Flavors of Credentials

- … are evidence used to prove identity
- Credentials can be
  1. **Something I am**
  2. **Something I have**
  3. **Something I know**

Credential:
Something I am.



But how do you prove who you are in the digital world?

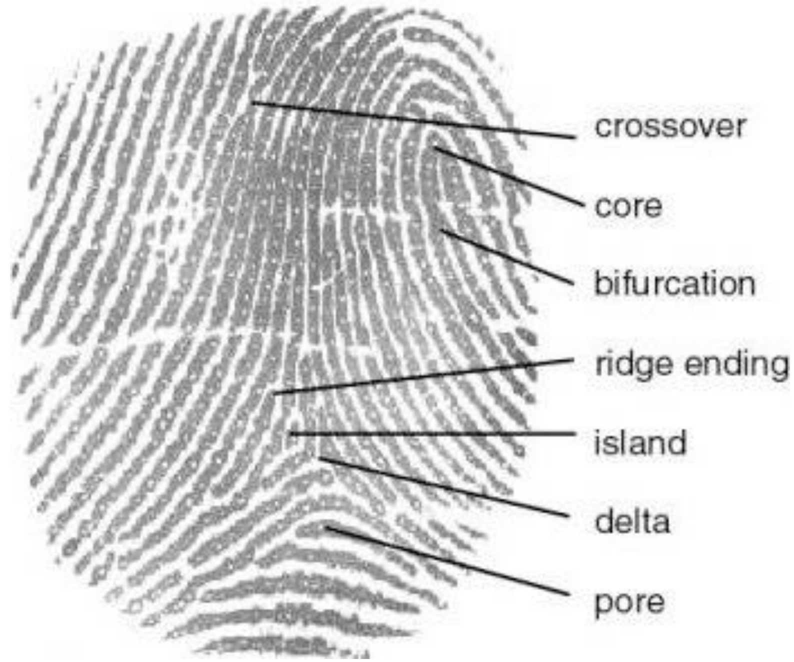
Biometrics

- Biometrics measure some physical characteristic
  - Fingerprint, face recognition, retina scanners, voice, signature, DNA
  - Pixel phones, Apple Face ID, Apple touch ID
  - Can be extremely accurate and fast

- Issues with biometrics?
  - Revocation – lost fingerprint?
  - *"Fuzzy" credential*, e.g., your face changes based on mood
  - Privacy?
  - Great for physical security, not feasible for on-line systems

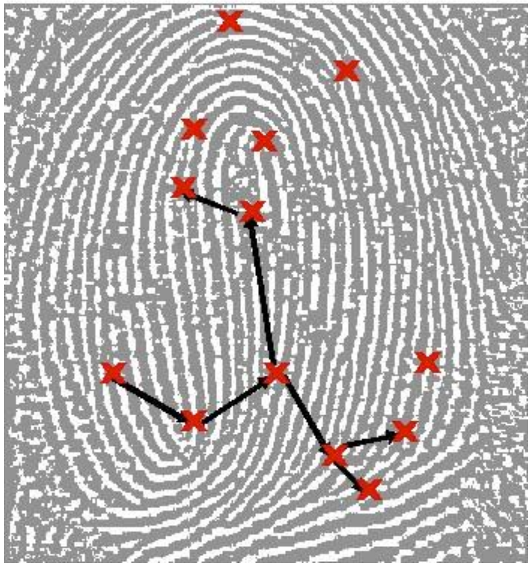
Biometrics Example

- Fingerprint readers record the conductivity of the surface of your finger to build a “map” of the ridges
- Scanned map converted into a graph by looking for landmarks, e.g., ridges, cores, ...



Fingerprint Biometrics

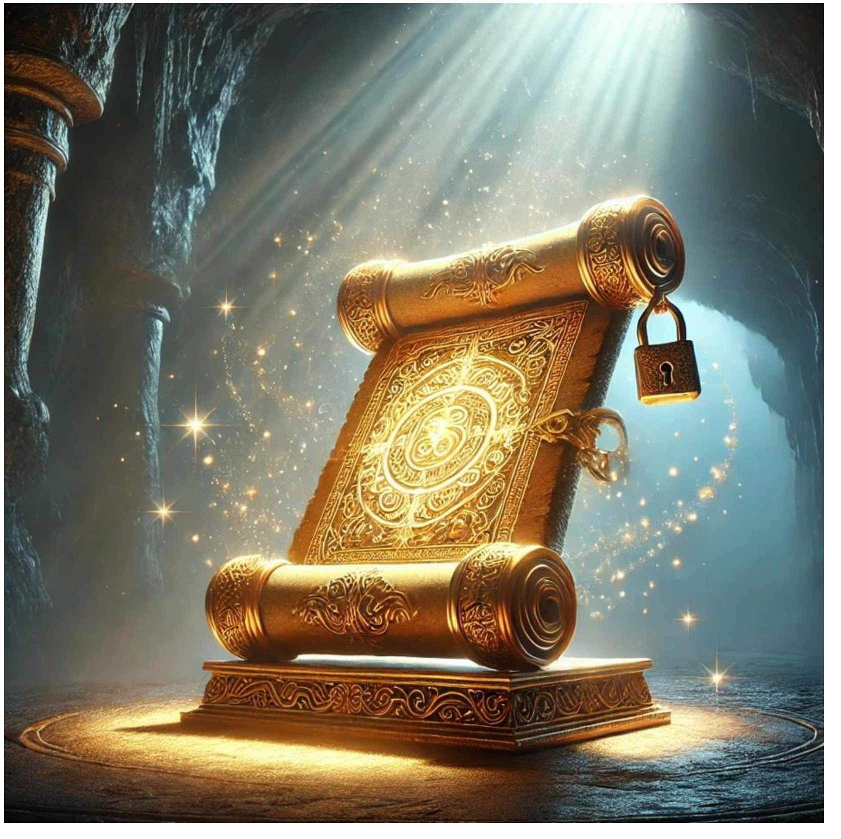
- Graph is compared to database of authentic identities
- If graph is same (enough), then person deemed "authentic"
  - Problem: what does it mean to be "same enough"?
    - rotation
    - imperfect contact
    - finger damage
  - ***Fundamental Problem***: False accept (FP) vs. false reject rates (FN)?



Dynamic Biometrics

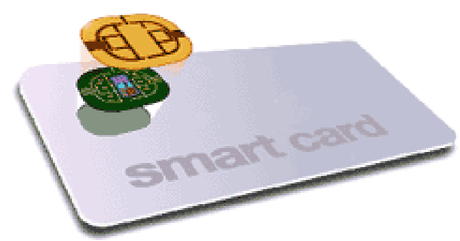
- Biometrics can be broken into two types
  - Static and dynamic
  - Prior examples are static biometrics
- Dynamic biometrics include
  - How we type on keyboard, gait analysis, voice, eye movement

Credential:
Something you have.



Credential: Something you have

- Digital Certificates
- Tokens (transponders, …)
  - EZ-pass
  - SecurID
- Smartcards
  - Unpowered processors
  - Small NV storage
  - Tamper *resistant*



A (simplified) sample token device

- A one-time password (or half of a two-factor authentication system)
- Secret key K
    - One-time password for epoch i is $\mathrm{HMAC}_K(i)$
    - Tamperproof token encodes K in firmware
    - Time synchronization allows authentication server to know what i is expected, and authenticate the user.
- *Note*: somebody can see your token display at some time but learn nothing useful for later periods.

Credential:
Something you know.

Something you know…

- Passport number, mother's maiden name, last 4 digits of your social security, credit card number
  - **Q: Are these good credentials?**
- Passwords and pass-phrases
  - Note: passwords are generally pretty weak, and may be used in more than one place
  - Computers can often guess very quickly
  - Easy to mount offline attacks
  - Easy countermeasures for online attacks

Some Issues for Password Systems

- A password should be easy to remember but hard to guess
  - that's difficult to achieve!
- Some questions
  - what makes a good password?
  - where is the password stored, and in what form?
  - how is knowledge of the password verified?

Password Storage

- Storing unencrypted passwords in a file is high risk
  - compromising the file system compromises all the stored passwords
- Better idea: use the password to compute a one-way function (e.g., a hash), and store the output of the one-way function
- When user inputs the requested password…
  1. compute its one-way function
  2. compare with the stored value

Attacks on Passwords

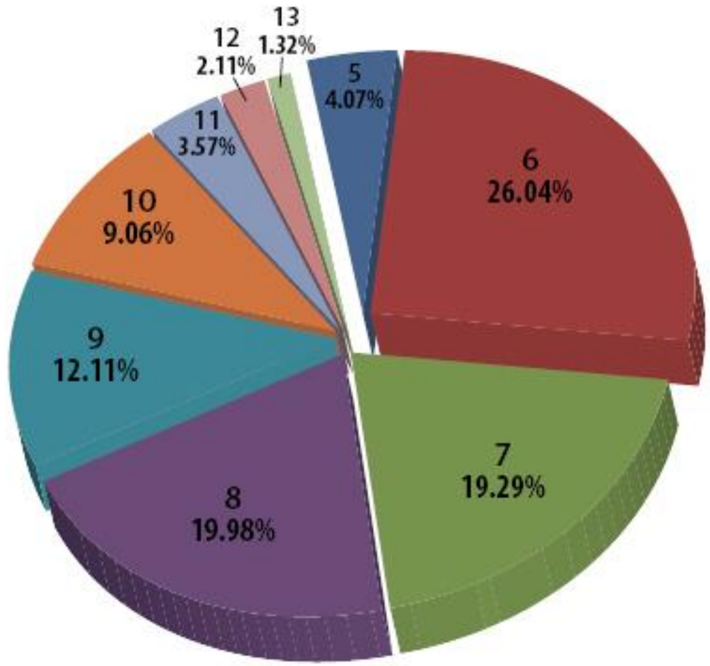
- Suppose passwords could be up to 9 characters long
  - 26 uppercase + 26 lowercase + 10 digits + 32 special characters -> nearly 10^2
- This would produce around ~10^{18} possible passwords; ~3200 years to try them all at 10 million a second!
- Unfortunately, not all passwords are equally likely to be used: password = *password!*

Password Popularity – Top 20

Rank	Password	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Rank	Password	Number of Users with Password (absolute)
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856

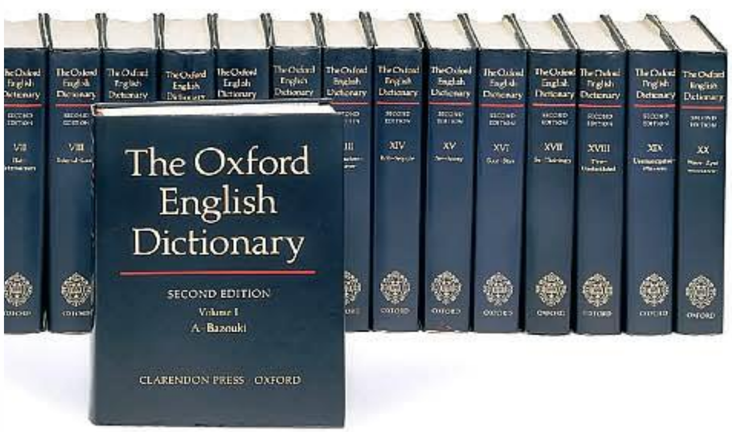
Password Length Distribution

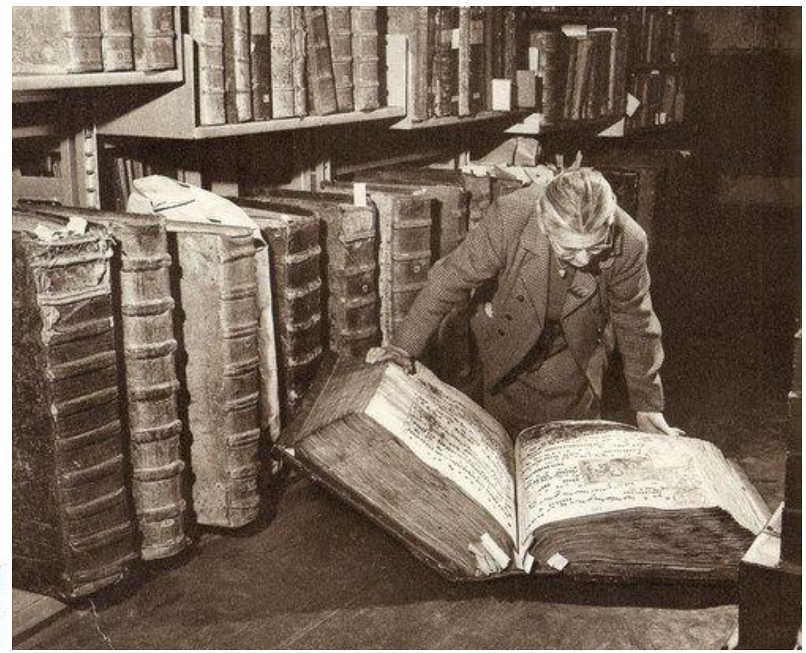


Source: iMPERVA 2010 study

Dictionary Attacks

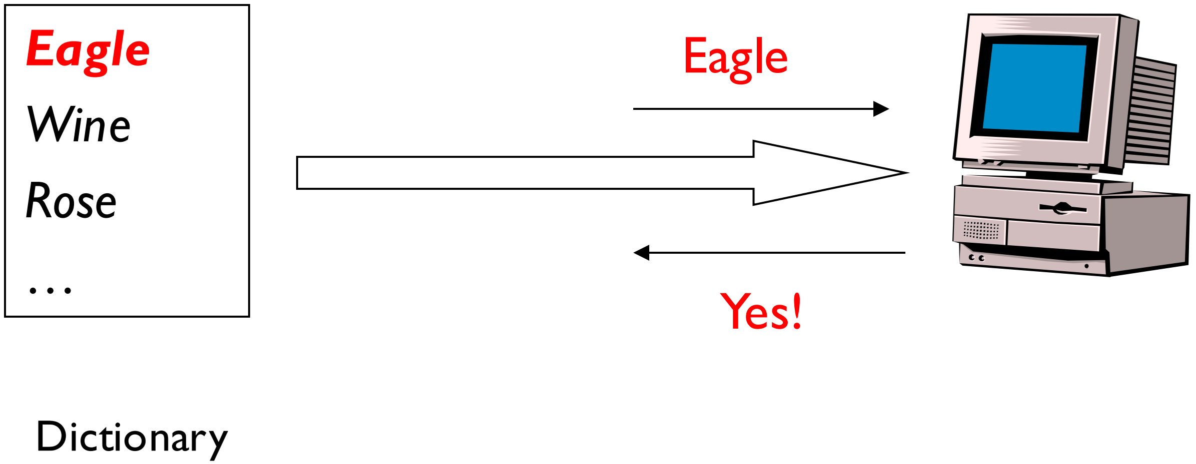
- Brute-force password by trying every word in a "dictionary"
- Plenty of automated tools: e.g., John the Ripper





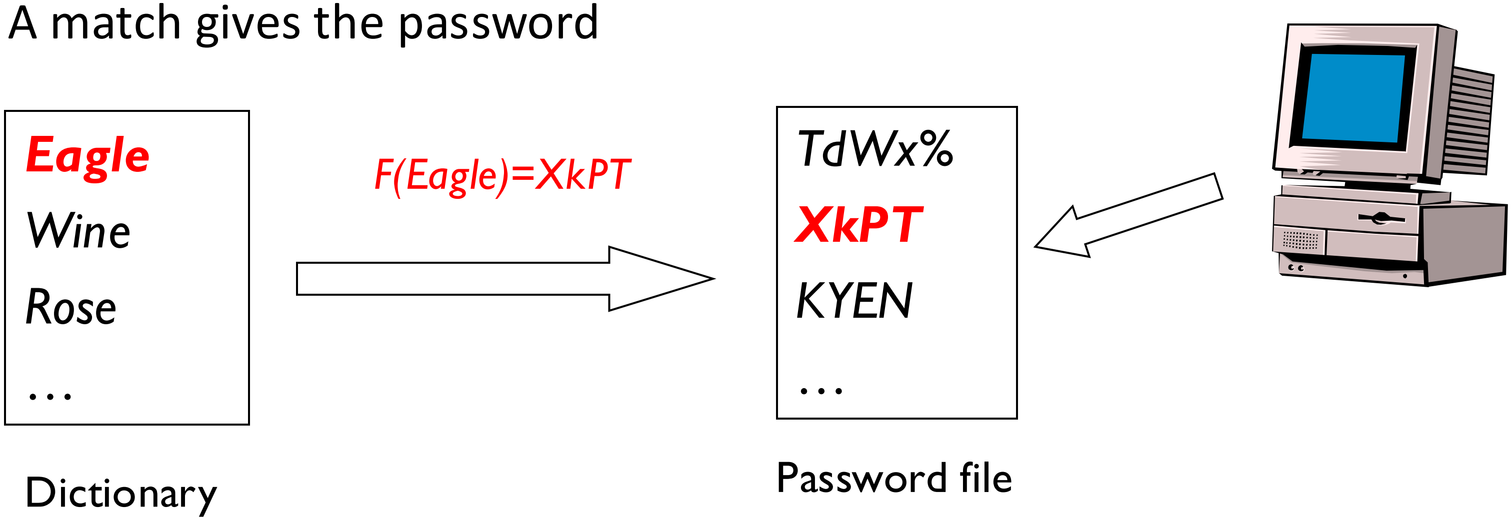
Dictionary Attacks (Cont'd)

- Attack 1 (online):
  - Create a dictionary of common words and names and their simple transformations
  - Use these to guess the password
  - *What's one easy mitigation? What does your phone do?*



Dictionary Attacks (Cont' d)

- Attack 2 (offline):
  - Usually *F* is public and so is the password file in some systems
    - In Unix, *F* is **`crypt`**, and the password file is **`/etc/passwd`**.
  - Compute *F*(*word*) for each word in the dictionary
  - A match gives the password



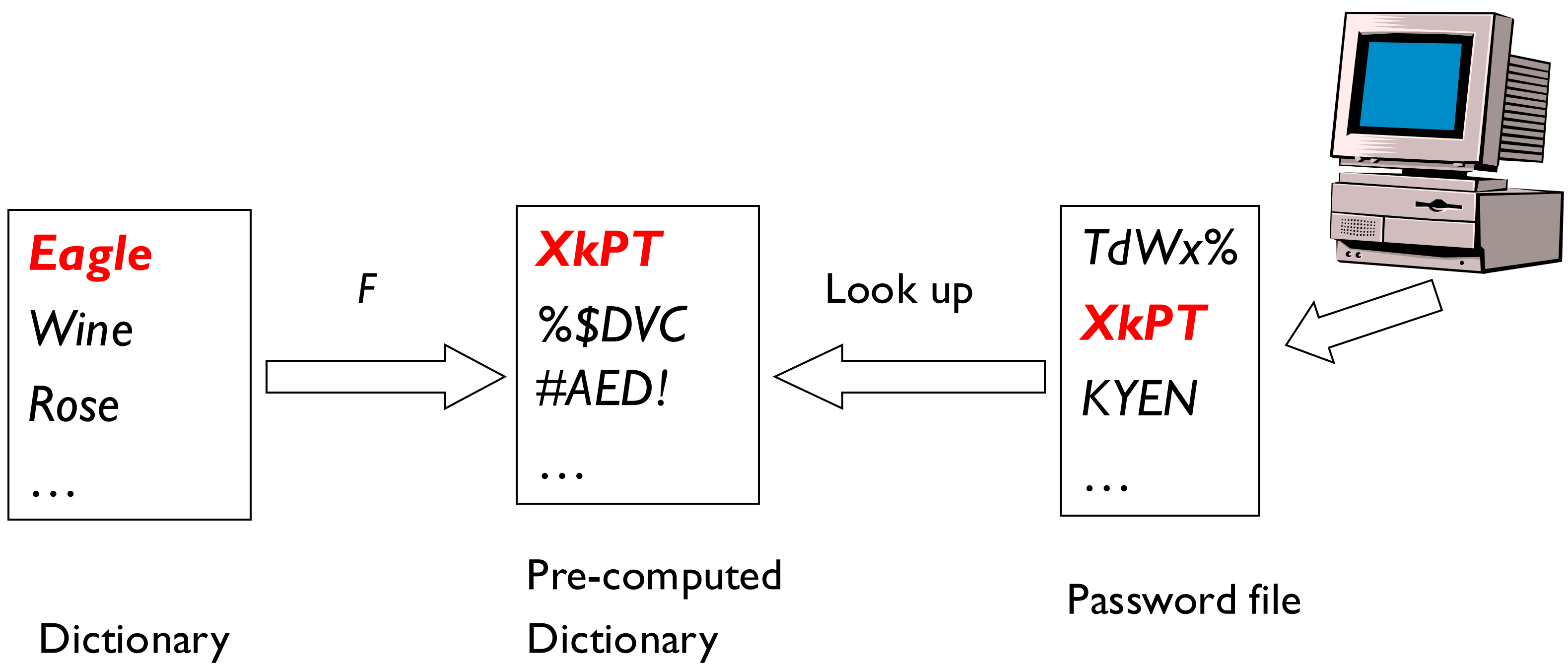
Lastpass customer vault leak

Summary of data accessed in Incident 2:

- DevOps Secrets – restricted secrets that were used to gain access to our cloud-based backup storage.
- Cloud-based backup storage – contained configuration data, API secrets, third-party integration secrets, customer metadata, and backups of all customer vault data. All sensitive customer vault

Dictionary Attacks (Cont'd)

- Attack 3 (offline):
  - To speed up search, pre-compute *F*(dictionary)
  - A simple look up gives the password



“Salt”ing passwords

- Suppose you want to make an *offline dictionary attack* more difficult
- A *salt* is a random number added to the password
- This is the approach taken by any reasonable system

$$salt_1, h(salt_1, pw_1)$$

$$salt_i, h(salt_2, pw_2)$$

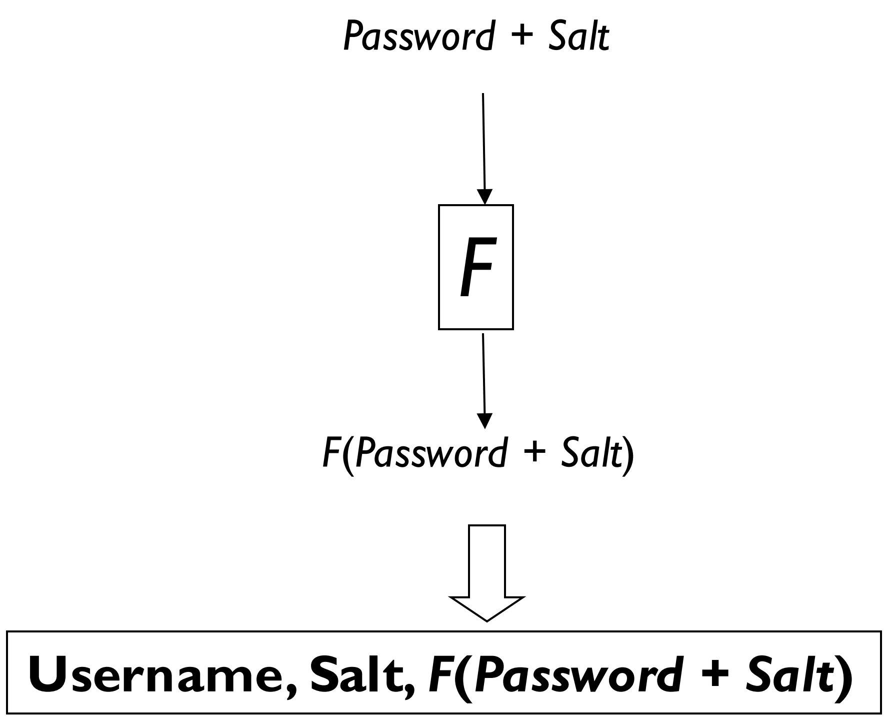
$$salt_i, h(salt_3, pw_3)$$

$$\ldots$$

$$salt_n, h(salt_n, pw_n)$$

Password Salt (Cont' d)

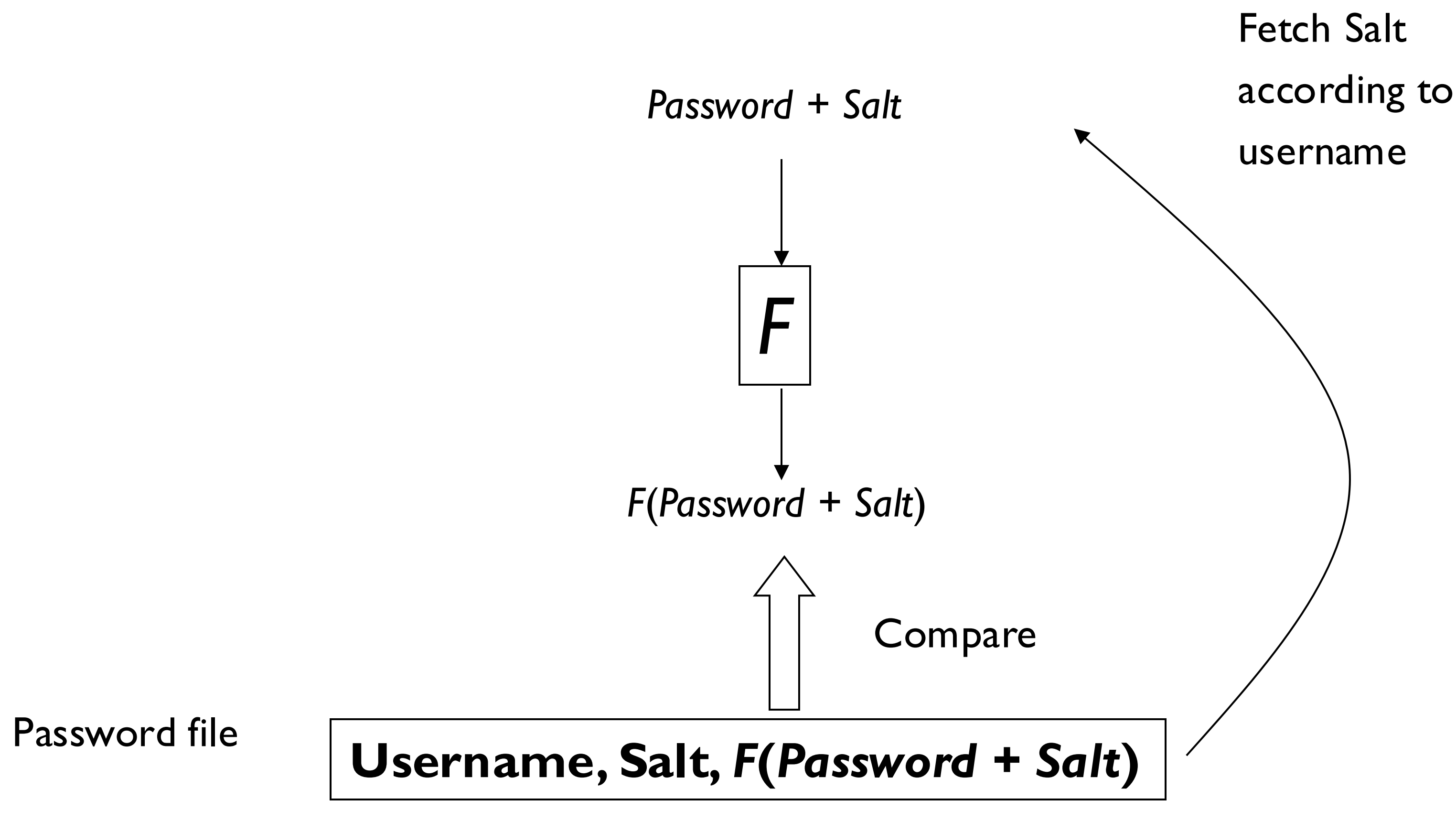
- Storing the passwords



Password file

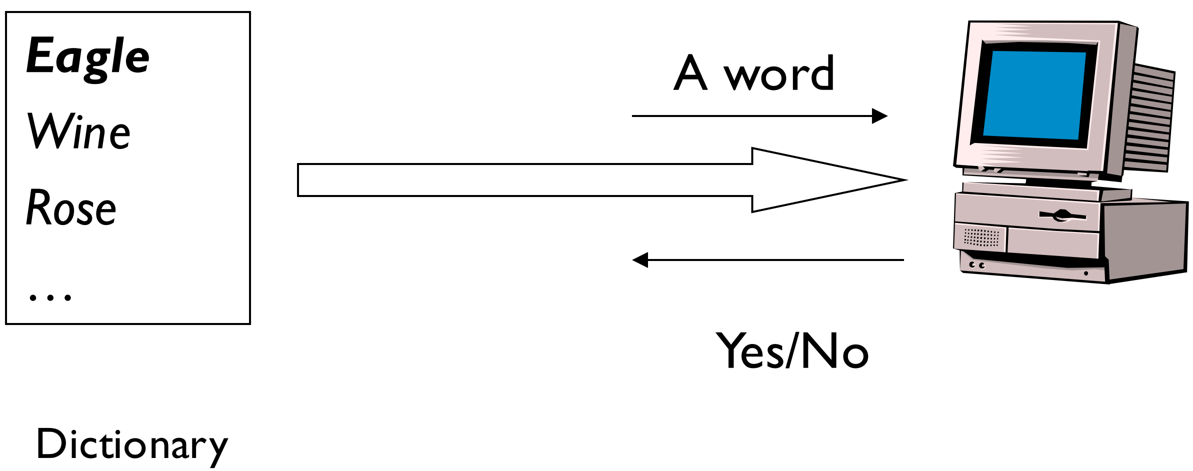
Password Salt (Cont' d)

- Verifying the passwords



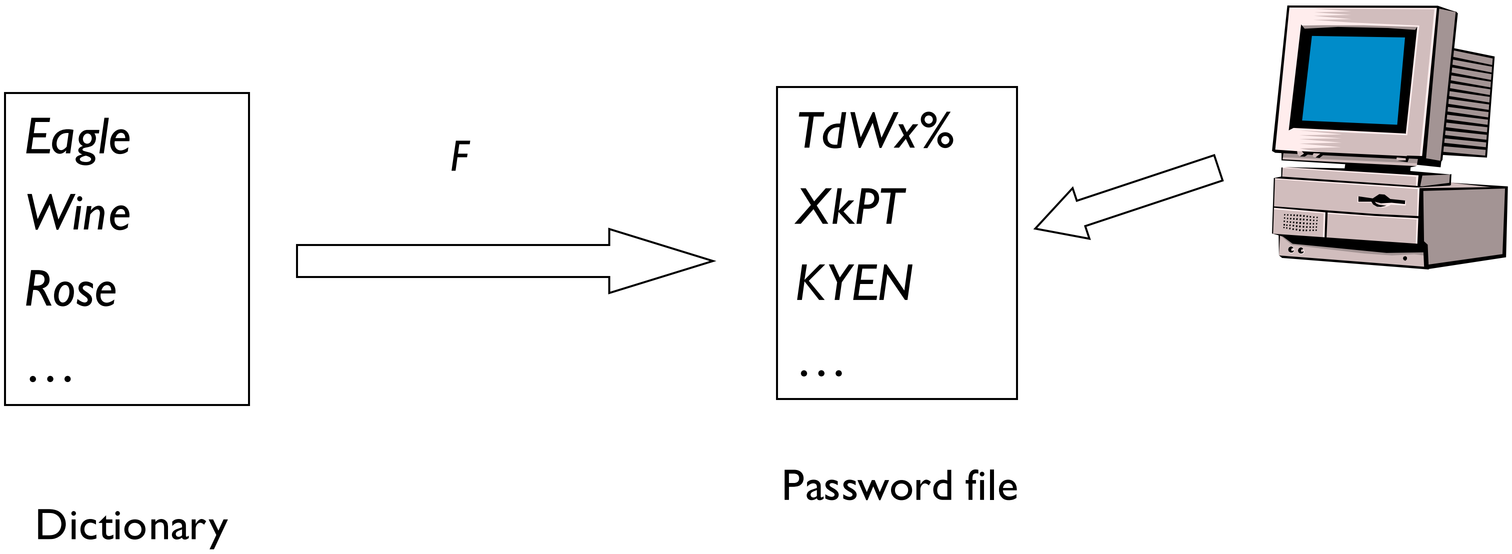
Does Password Salt Help?

- Attack 1 (online)?
  - Without Salt
  - With Salt



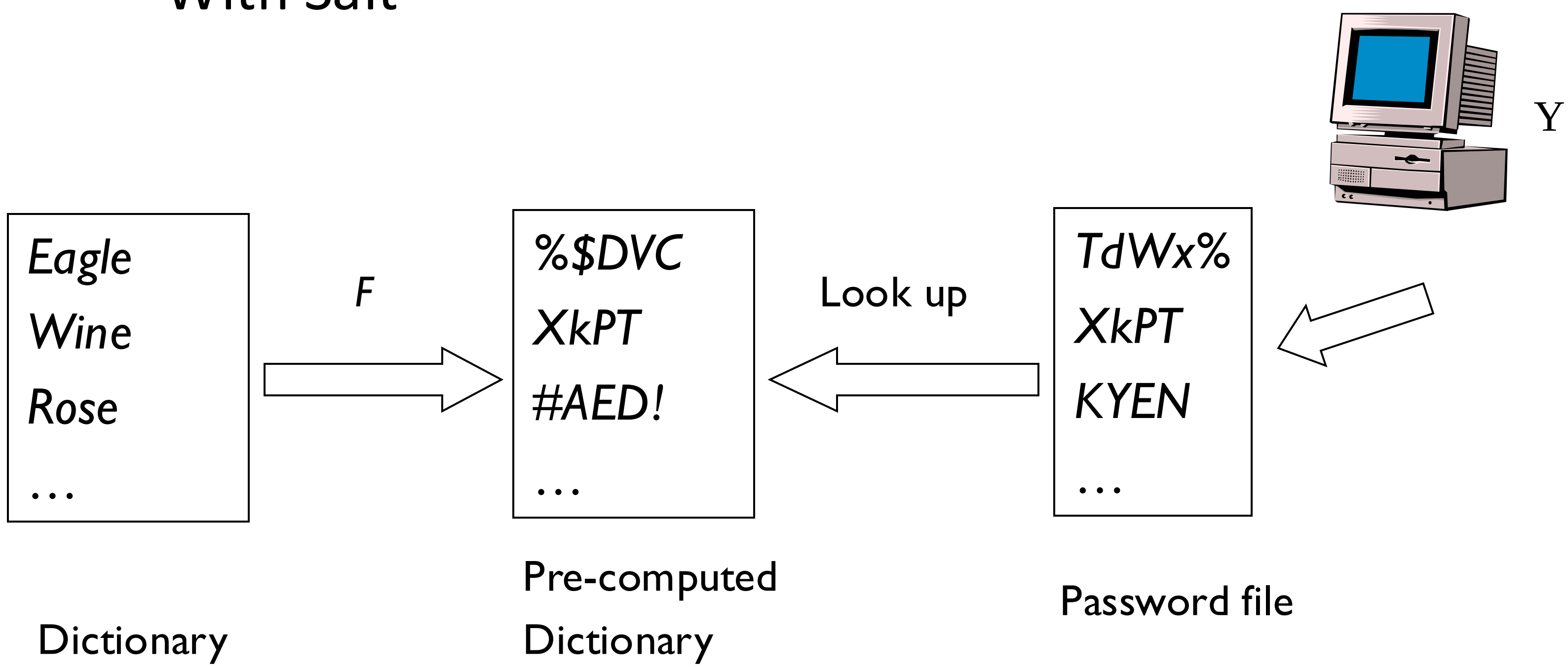
Does Password Salt Help?

- Attack 2 (offline)?
  - Without Salt
  - With Salt



Does Password Salt Help?

- Attack 3?
  - Without Salt
  - With Salt



Example: Unix Passwords

- Keyed password hashes are stored, with two-character (16 bit) salt prepended
  - password file is publicly readable
- Users with identical passwords but different salt values will have different hash values

Is this secure?

- Suppose you have a salted password cracker.
  - It takes 10 microseconds to check a guess.
  - The password is chosen from the following pattern:
  - where "d+" is 1-4 digits and "w" is a word taken out of a 100,000 word dictionary.
- How long (avg) does it take to crack the password?

  - ***{d+}***
  - ***{d+}w***
  - ***w{d+}***
  - ***{d+}w{d+}***

Brute forcing …

$$\begin{aligned}
\{d+\} &= 10^4 + 10^3 + 10^2 + 10^1 &&= 11{,}110 \\
\{d+\}w &= 11{,}110 * 100{,}000 &&= 1{,}111{,}000{,}000 \\
w\{d+\} &= 100{,}000 * 11{,}110 &&= 1{,}111{,}000{,}000 \\
\{d+\}w\{d+\} &= 11{,}110 * 100{,}000 * 11{,}110 &&= 12{,}343{,}210{,}000{,}000 \\
& &&= 12{,}345{,}432{,}011{,}110
\end{aligned}$$

$$\begin{aligned}
&12,345,432,011,110 \ guesses/100,000 = \\
&123,454,320.11 \ seconds/2 = \\
&61,727,160.05 \ seconds \ (on \ average) = \\
&17,146.43 \ hours \approx \\
&714.43 \ days \approx \\
&1.9 \ years
\end{aligned}$$

- Does not seem so bad, right?
  - Now try (in your time) d+ is 1-2 characters
  - What about dictionary of 1,000 words?

Human vs Machine

- The rule of seven plus or minus two (Link to some background info on this).
  - George Miller observed in 1956 that most humans can remember about 5-9 things more or less at once.
  - Thus is a kind of maximal entropy that one can hold in your head.
  - This limits the complexity of the passwords you can securely use, i.e., not write on a sheet of paper.
  - A perfectly random 8-char password has less entropy than a 56-bit key.

- Implication?

https://elearningindustry.com/magical-number-seven-plus-minus-two-memory-affects-the-perception-of-information

$$salt_i, h^{100}(salt_i, pw_i)$$

Slowing down the process

Compromised Passwords

- Guessing a password is only one way to lose it
- Other ways
  - Eavesdropping
  - Phishing
  - Password reuse on multiple websites
- *Solution*: each site has a different password

Password Managers

- … but the number of combinations makes the memory recall problem even harder
- A common approach is to have tiers of passwords
  - E.g., system login, banking, shopping, email, social media, throw-away
- Another solution is to have a password manager
  - Many options (in-browser, LastPass, KeePass, etc.)
  - Considerations:
    - Where is the database stored?
    - How is the database protected?
    - Integration with mobile OSes?
    - Copy to clipboard?

Multifactor Authentication

- While passwords are the standard, the other factors (are, have) can be combined to enhance security
- Examples:
  - Google's 2-step verification
  - SMS messages
- *Caution*: what if you are authenticating from a mobile device?

Forgotten Passwords

- With all of these passwords, users often forget what password they used
- Systems must have an automated password recovery method
- Common Methods
  - Email reset
  - Security questions
  - Phone call / SMS
- What is good and bad about these?
- *FileVault on Mac:* Use Apple ID to recover data, *no MFA!*

Web Authentication

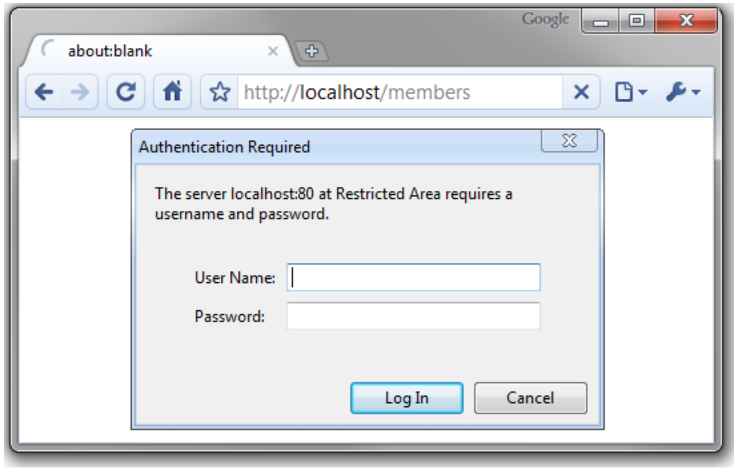
(still based on“something you know”)

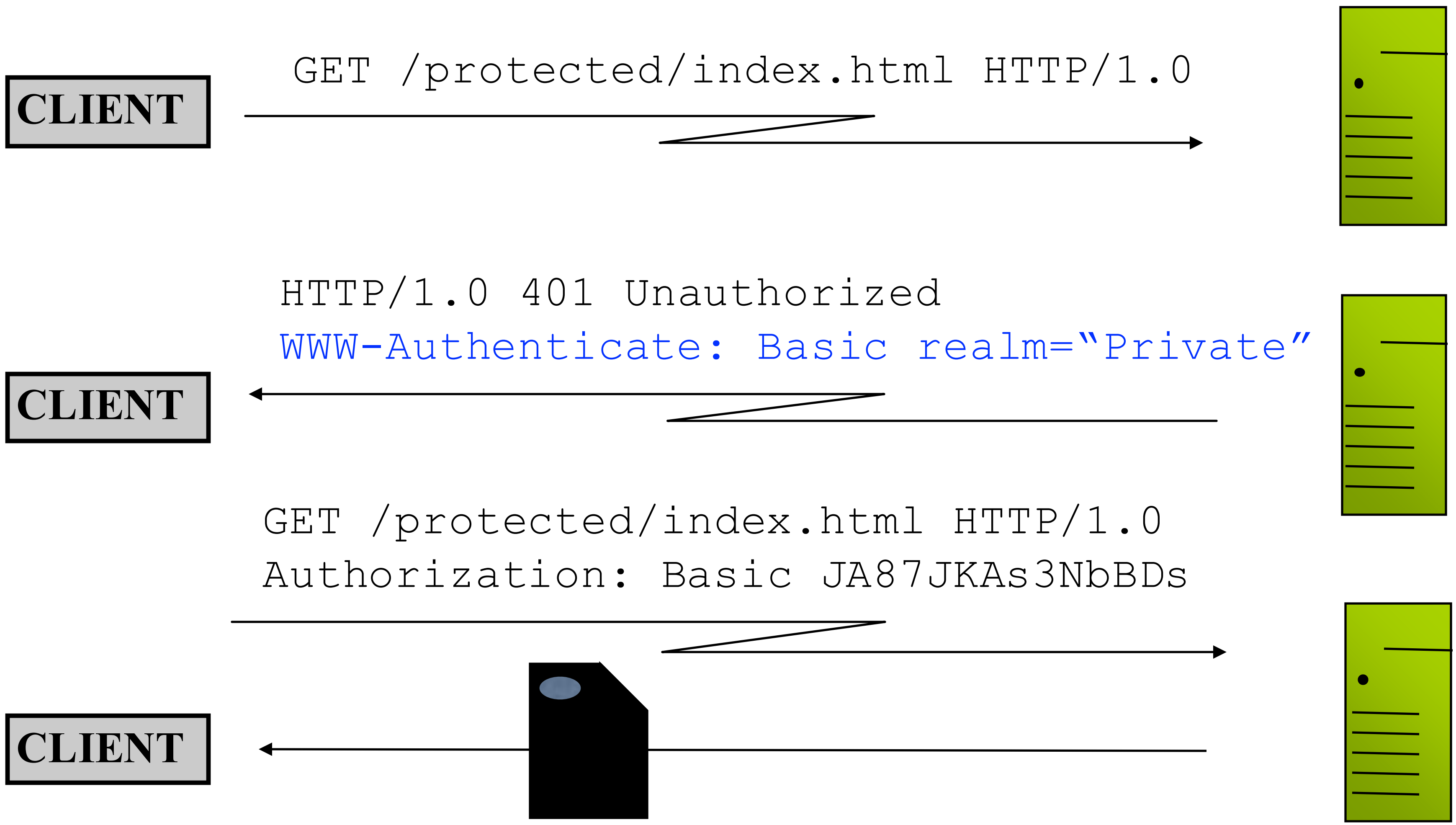
Credentials can be

1. Something I am
2. Something I have
3. **Something I know**

Web Authentication

- Authentication is a bi-directional process
  - Client
  - Server
  - Mutual authentication
- Several standard authentication tools
  - Basic (client)
  - Digest (client)
  - Secure Socket Layer (server, mutual)





Basic Authentication

Basic Authentication -- is this secure?

- Encoded ! = Encrypted
  - Passwords easy to intercept (base-64 encoded; <u>not</u> encrypted)
- Passwords:
  - easy to guess
  - easy to share
- No server authentication - easy to fool client into sending password to malicious server

Digest Authentication

Challenge/Response

- Challenge nonce is a one time random string/value

$$\text{nonce} = H(\text{IPaddress} : \text{timestamp} : \text{server secret})$$

  - more generally, a **nonce** is number or string (often randomly or pseudorandomly chosen) that is **only used once**

- Response: challenge hashed with username and password

$$response = H(H(name : realm : password) : nonce : H(request))$$

Advantages of Digest over Basic

- Cleartext password never transmitted across network
- Cleartext password never stored on server
- **Replay attacks** difficult
- Intercepted response only valid for a single URL
- Shared disadvantages
  - Vulnerable to man-in-the-middle attacks (no server-side auth)
  - Document itself can be sniffed