CIS 6930/4930 Computer and Network Security

Topic 6. Authentication

Authentication

- Authentication is the process of reliably verifying certain information.
- Examples
  - User authentication
    - Allow a user to prove his/her identity to another entity (e.g., a system, a device).
  - Message authentication
    - Verify that a message has not been altered without proper authorization.

Authentication Mechanisms

- Password-based authentication
  - Use a secret quantity (the password) that the prover states to prove he/she knows it.
  - Threat: password guessing/dictionary attack

Authentication Mechanisms (Cont'd)

- Address-based authentication
  - Assume the identity of the source can be inferred based on the network address from which packets arrive.
- Threat
  - Spoof of network address
    - Not authentication of source addresses

Authentication Mechanisms (Cont'd)

- Cryptographic authentication protocols
  - Basic idea:
    - A prover proves some information by performing a cryptographic operation on a quantity that the verifier supplies.
  - Usually reduced to the knowledge of a secret value
    - A symmetric key
    - The private key of a public/private key pair

CIS 6930/4930 Computer and Network Security

Topic 6.1 User Authentication

Authentication and Identity

- What is identity?
  - which characteristics uniquely identifies a person?
  - do we care if identity is unique?
- Authentication: verify a user's identity
  - a *supplicant* wishes to be authenticated
  - a *verifier* performs the authentication

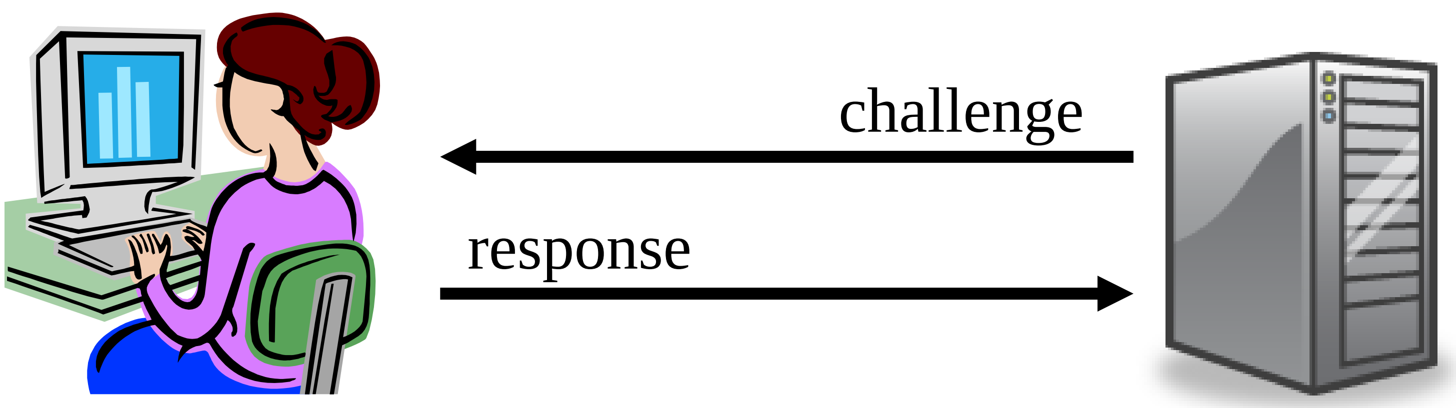
User Authentication Can Be Based On...

- What the user knows
  - passwords, personal information, a key, a credit card number, etc.
- Where the user is or can be reached
  - email address, IP address, …
- Physical characteristics of the user
  - fingerprints, voiceprint, signature dynamics, iris pattern, DNA, etc.
- What the user has in their possession
  - smart card, (physical) key, USB token, …

Password Authentication

Password-Based User Authentication

- User demonstrates knowledge of a secret value to authenticate
  - most common method of user authentication



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, an encryption), and store the output of the one-way function
- When a user inputs the requested password…
  1. compute its one-way function
  2. compare with the stored value

Attacks on Passwords

- Suppose passwords can be from 1 to 9 characters in length
- Possible choices for passwords = $26^1 + 26^2 + \ldots + 26^9 = 5 * 10^{12}$
- At the rate of 1 password per millisecond, it will take on the order of 150 years to test all passwords
- Unfortunately, not all passwords are equally likely to be used

Example of a Study

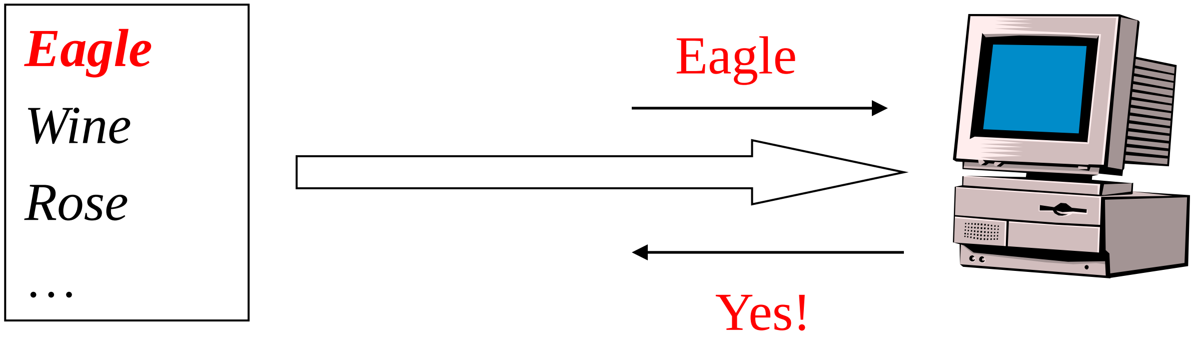
- In a sample of over 3000 passwords:
  - 500 were easily guessed versions of dictionary words or first name / last name
  - 86% of passwords were easily guessed

Common Password Choices

- Pet names
- Common names
- Common words
- Dates
- Variations of above (backwards, append a few digits, etc.)

Dictionary Attacks

- Attack 1 (online):
  - Create a dictionary of common words and names and their simple transformations
  - Use these to guess the password



Dictionary

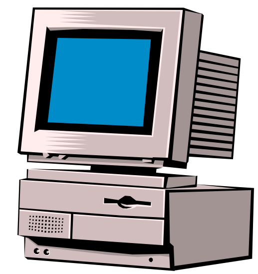
Dictionary Attacks (Cont'd)

- Attack 2 (offline):
  - Usually *F* is public and so is the password file
  - Compute *F*(*word*) for each word in the dictionary
  - A match gives the password

Dictionary
Eagle
Wine
Rose
…

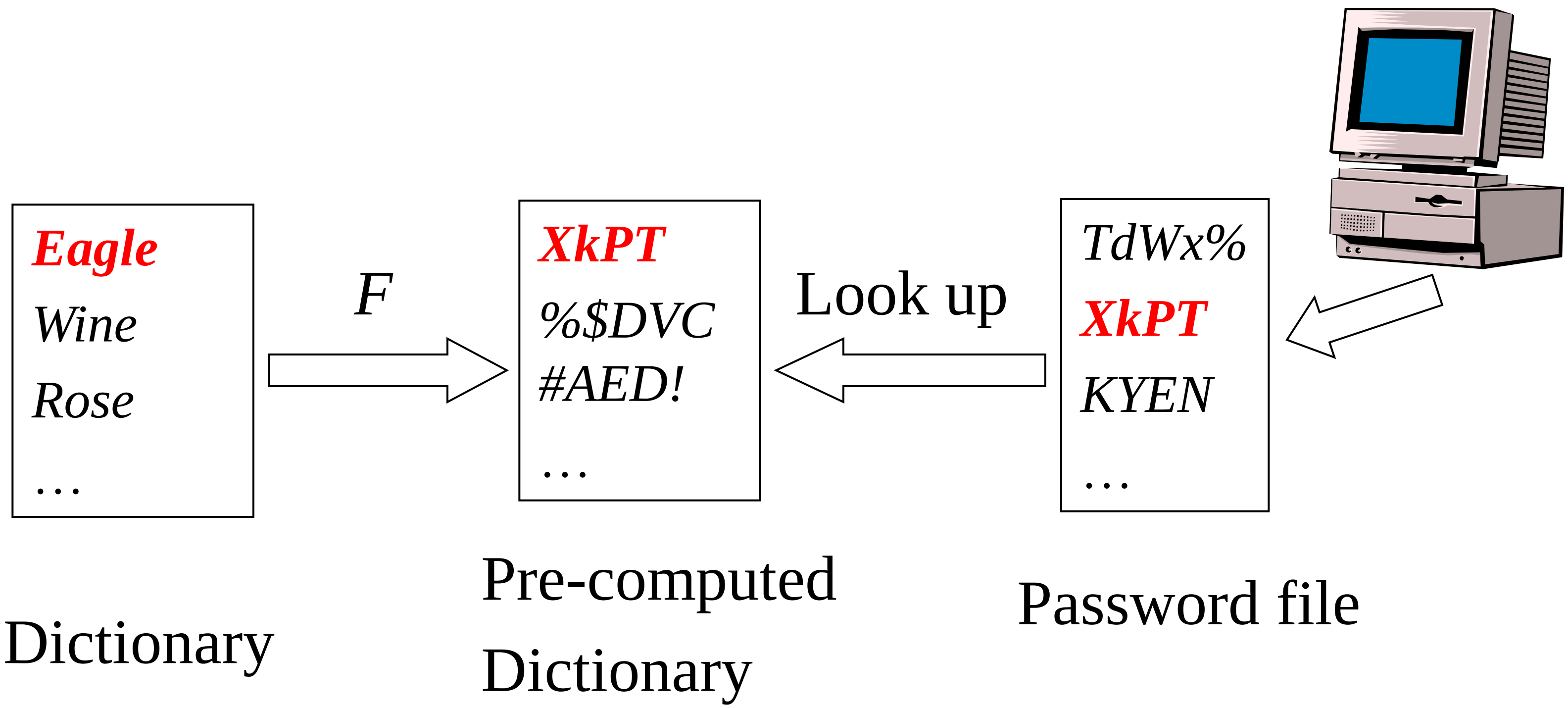
F(Eagle)=XkPT ⇒

Password file
TdWx%
XkPT
KYEN
…



Dictionary Attacks (Cont'd)

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



Password Salt

- To make the dictionary attack a bit more difficult
- Salt is a n-bit number between 0 and 2^n
- Derived from, for example, the system clock and the process identifier

Password Salt (Cont'd)

- Storing the passwords

Password + Salt

↓

$\boxed{F}$

↓

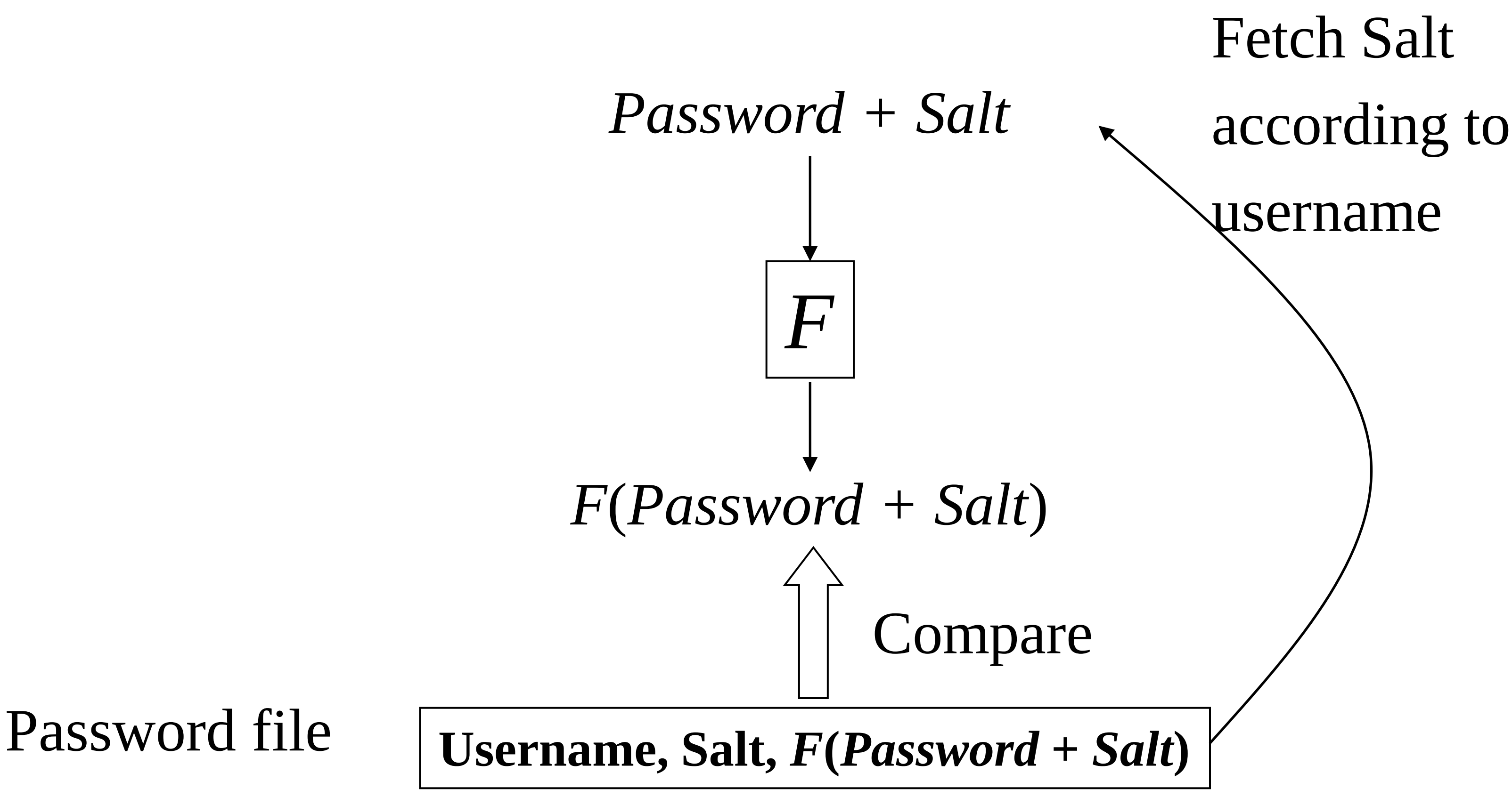
F(*Password + Salt*)

⇩

Password file | **Username, Salt, *F*(*Password + Salt*)**

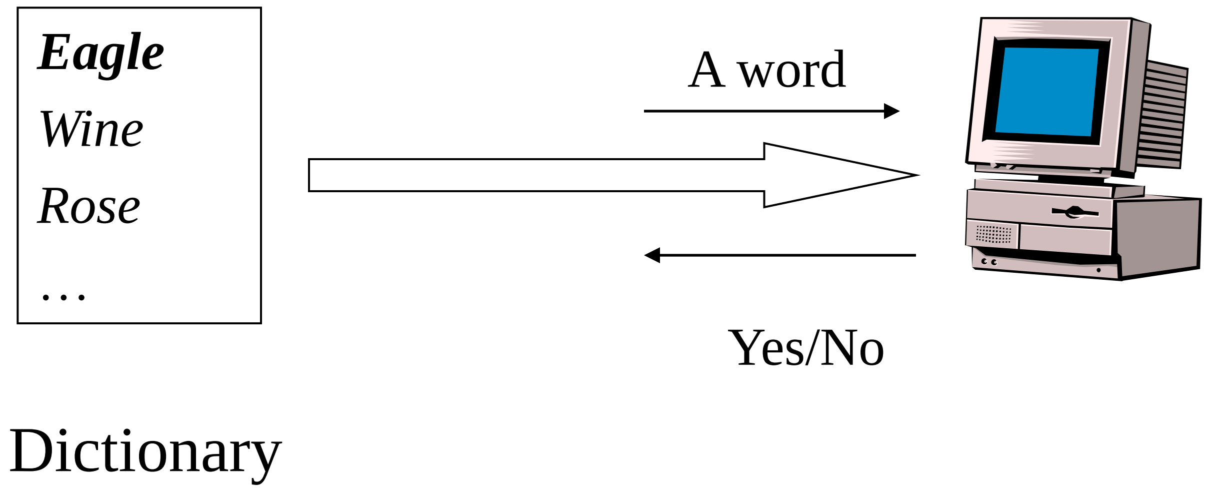
Password Salt (Cont'd)

- Verifying the passwords



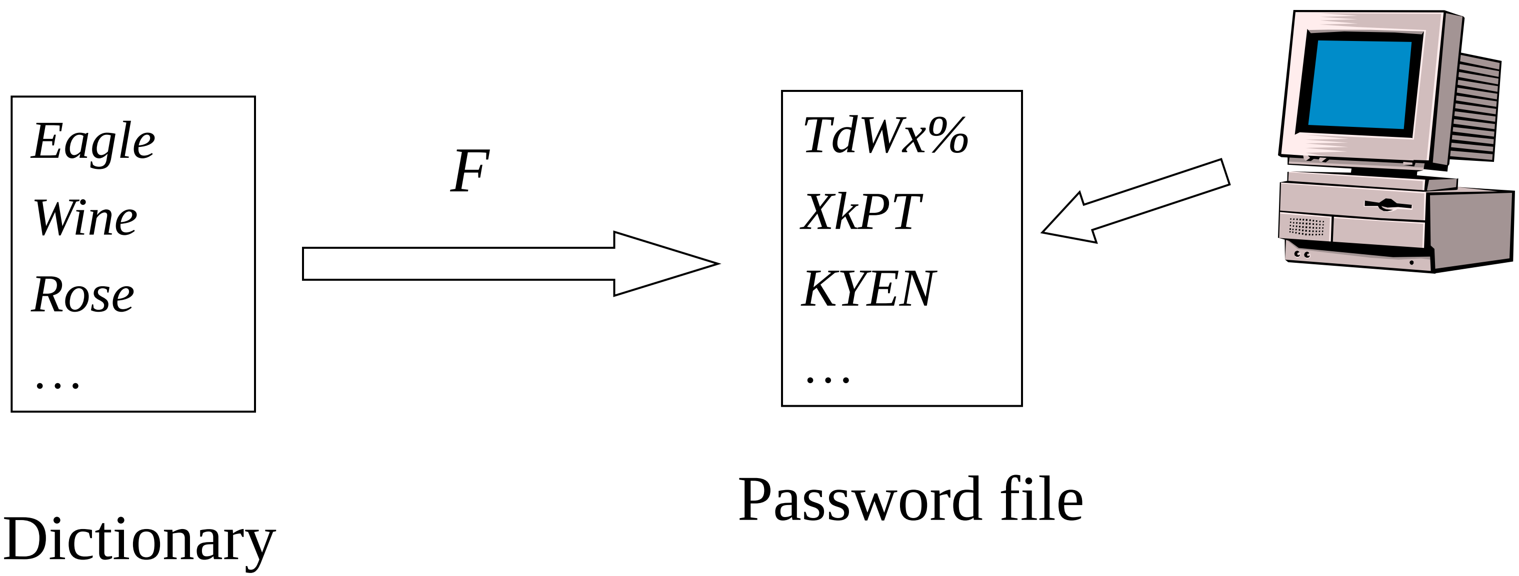
Does Password Salt Help?

- Attack 1?
  - Without Salt
  - With Salt



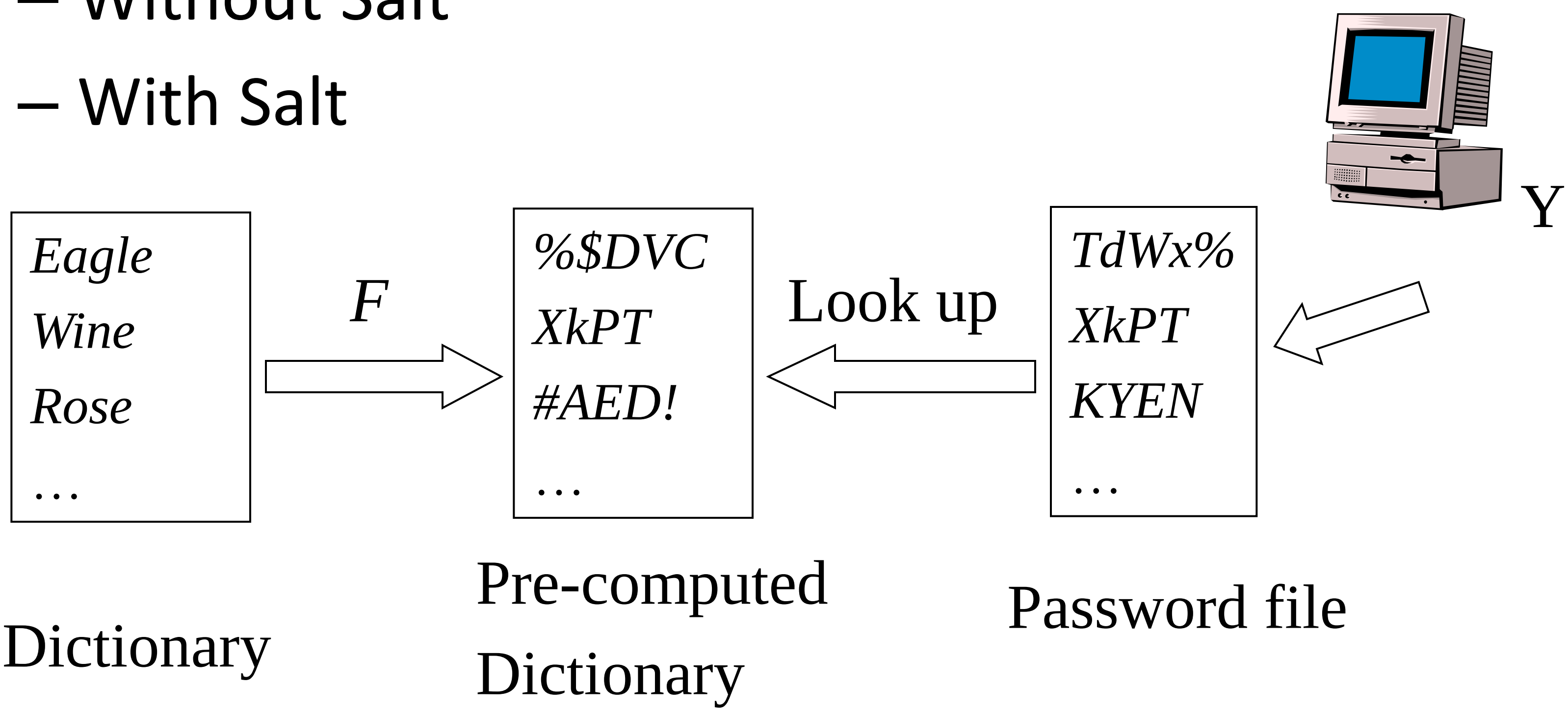
Does Password Salt Help?

- Attack 2?
  - Without Salt
  - With Salt



Does Password Salt Help?

- Attack 3?
  - Without Salt
  - With Salt



Password Guidelines For Users

1. Initial passwords are system-generated, have to be changed by user on first login
2. User must change passwords periodically
3. Passwords vulnerable to a dictionary attack are rejected
4. User should not use same password on multiple sites

Other Password Attacks

- Technical
  - eavesdropping on traffic that may contain unencrypted passwords
  - “Trojan horse” password entry programs
- “Social”
  - careless password handling or sharing
  - phishing

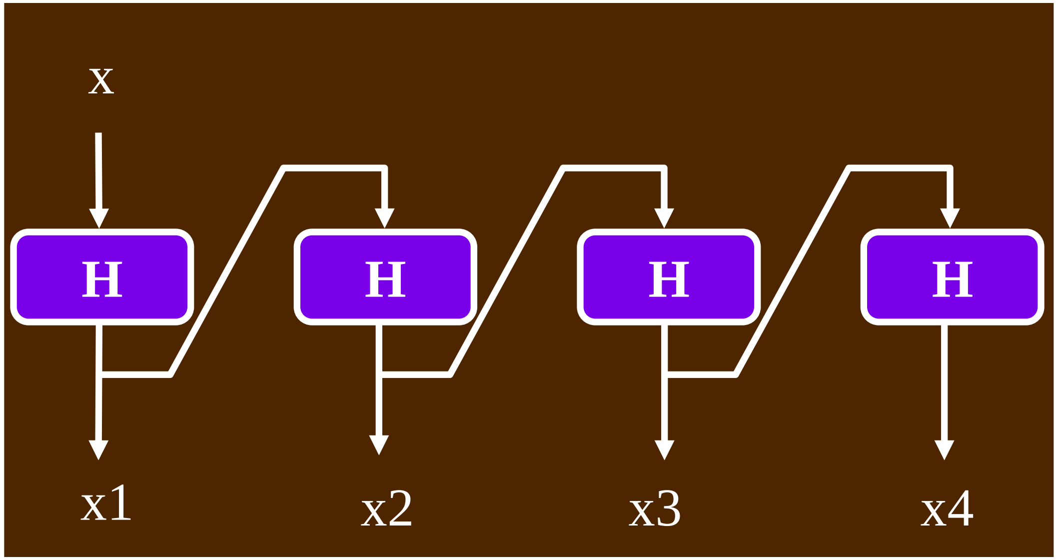
The S/Key Protocol

Using "Disposable" Passwords

- Simple idea: generate a long list of passwords, use each only one time
  - attacker gains little/no advantage by eavesdropping on password protocol, or cracking one password
- Disadvantages
  - storage overhead
  - users would have to memorize lots of passwords!
- Alternative: the S/Key protocol
  - based on use of one-way (e.g. hash) function

S/Key Password Generation

1. Alice selects a password **x**
2. Alice specifies *n*, the number of passwords to generate
3. Alice's computer then generates a sequence of passwords
   – $x_1 = H(\mathbf{x})$
   – $x_2 = H(x_1)$
   – …
   – $x_n = H(x_{n-1})$



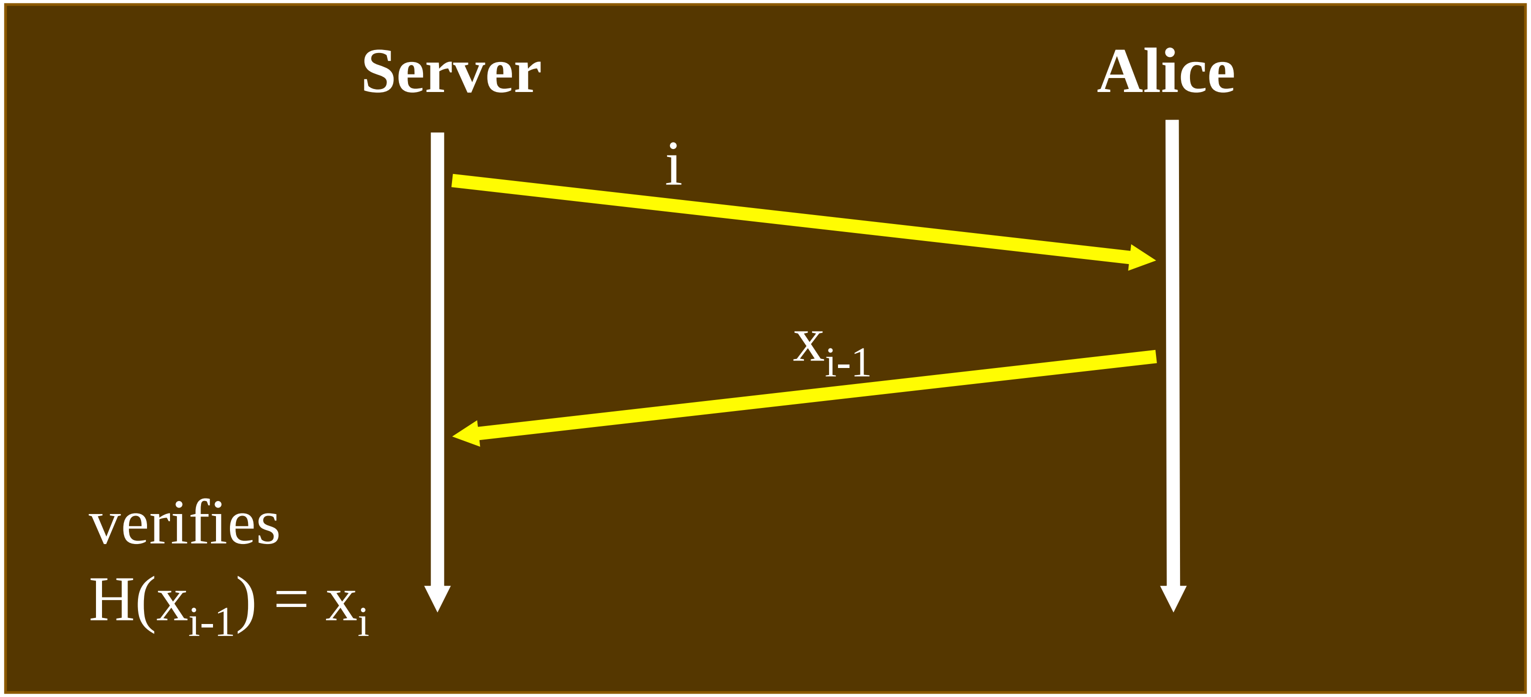
Generation… (cont'd)

4. Alice communicates (securely) to a server the last value in the sequence: x_n

- Key feature: no one knowing x_i can easily find an $x_{i\text{-}1}$ such that $H(x_{i\text{-}1}) = x_i$
  – only Alice possesses that information

Authentication Using S/Key

- Assuming server is in possession of x_i …



Is dictionary attack still possible?

Limitations

- Value of *n* limits number of passwords
  – need to periodically regenerate a new chain of passwords
- Does not authenticate server! Example attack:
  1. real server sends *i* to fake server, which is pretending to be Alice
  2. fake server sends *i* to Alice, who responds with $x_{i\text{-}1}$
  3. fake server then presents $x_{i\text{-}1}$ to real server

Biometrics

- Relies upon physical characteristics of people to authenticate them
- Desired properties
  1. uniquely identifying
  2. very difficult to forge / mimic
  3. highly accurate
  4. easy to scan or collect
  5. fast to measure / compare
  6. inexpensive to implement

Assessment

- Convenient for users (e.g., you always have your fingerprints, never have to remember them), but…
  - potentially troubling sacrifice of private information
  - no technique yet has all the desired properties

Assessment (cont'd)

Biometrics	Universality	Uniqueness	Permanence	Collectability	Performance	Acceptability	Circumvention
Face	H	L	M	H	L	H	L
Fingerprint	M	H	H	M	H	M	H
Hand Geometry	M	M	M	H	M	M	M
Keystroke Dynamics	L	L	L	M	L	M	M
Hand vein	M	M	M	M	M	M	H
Iris	H	H	H	M	H	L	H
Retina	H	H	M	L	H	L	H
Signature	L	L	L	H	L	H	L
Voice	M	L	L	M	L	H	L
Facial Thermogram	H	H	L	H	M	H	H
DNA	H	H	H	L	H	L	L

H=High, M=Medium, L=Low

Example Biometric Technologies

- Signature / penmanship
- Fingerprints
- DNA
- Palm geometry
- Retina scan
- Iris scan
- Face recognition
- Voice recognition