CSCI 454/554 Computer and Network Security

Topic 6. Authentication

Authentication

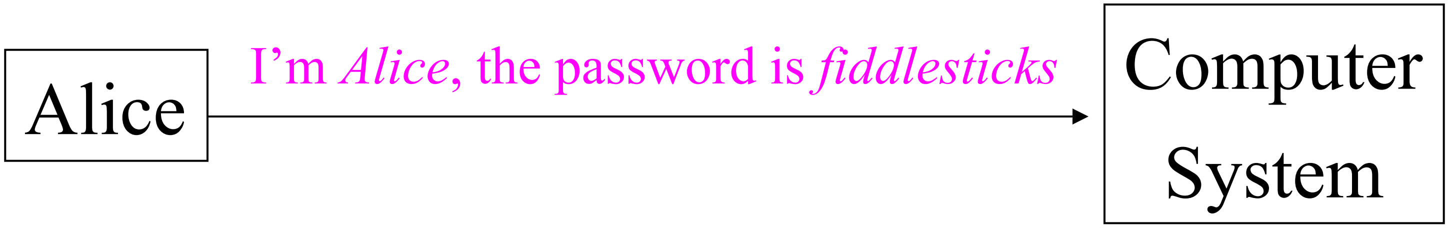
- Authentication is the process of reliably verifying certain information.
- Examples
  - User authentication
    - Allow a user to prove his/her identity to another entity (e.g., a system, a device).
  - Message authentication
    - Verify that a message has not been altered without proper authorization.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

Identification

- *Identification* is a process through which one ascertains the identity of another person or entity.
- Authentication and identification are different.
  - Identification requires that the verifier check the information presented against all the entities it knows about,
  - Authentication requires that the information be checked for a single, previously identified, entity.
  - Identification must, by definition, uniquely identify a given entity,
  - Authentication does not necessarily require uniqueness.

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



Alice

I'm *Alice*, the password is *fiddlesticks*

Computer System

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.
  - Adopted early in UNIX and VMS
- Berkeley *rtools* (*rsh*, *rlogin*, etc)
  - */etc/hosts.equiv* file
    - List of computers
  - Per user *.rhosts* file
    - List of <computer, account>
- 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

CSCI 454/554 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 authenticate
  - a *verifier* performs the authentication
- What’s relationship of identity to *role*, or job function?

User Authentication Can Be Based On…

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

Which of the above is best? Best in what way?

Crypto-Based Authentication

- Basic idea: user performs a requested cryptographic operation on a value (a challenge) that the verifier supplies
- Usually based on knowledge of a key (secret key or private key)
- Examples: RSA, zero knowledge proofs, …
- *We'll look at such protocols in more detail next time*

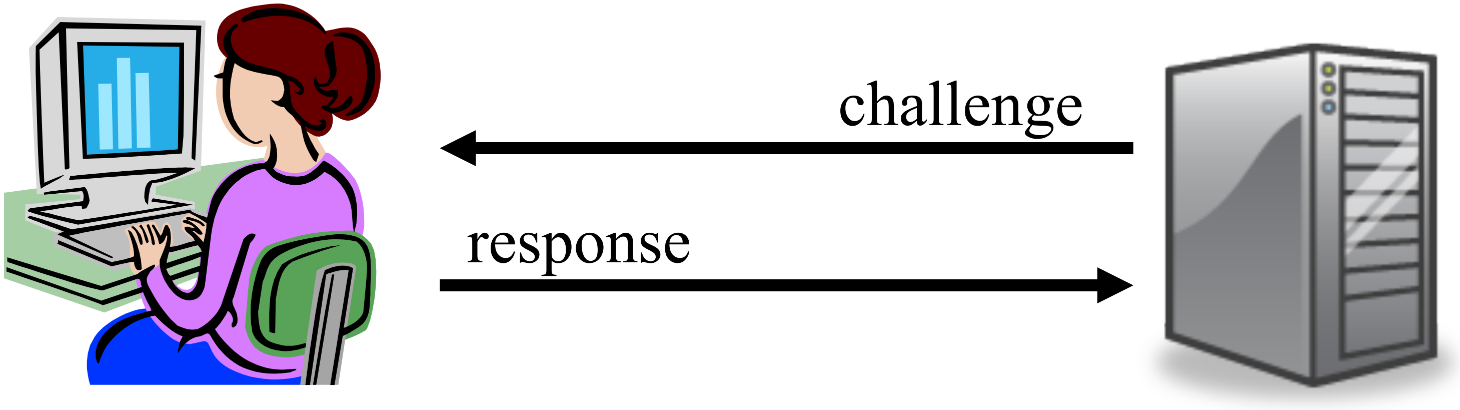
Address-Based User Authentication

- Associates identity with network address or email address
  - used by many web services
- Several early OS functions and tools worked this way
- Benefits? Problems?

Password Authentication

Password-Based User Authentication

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



- Threats to password-based 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 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
- This would produce 10^{18} possible passwords; 320,000 years to try them all at 10 million a second!
- 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

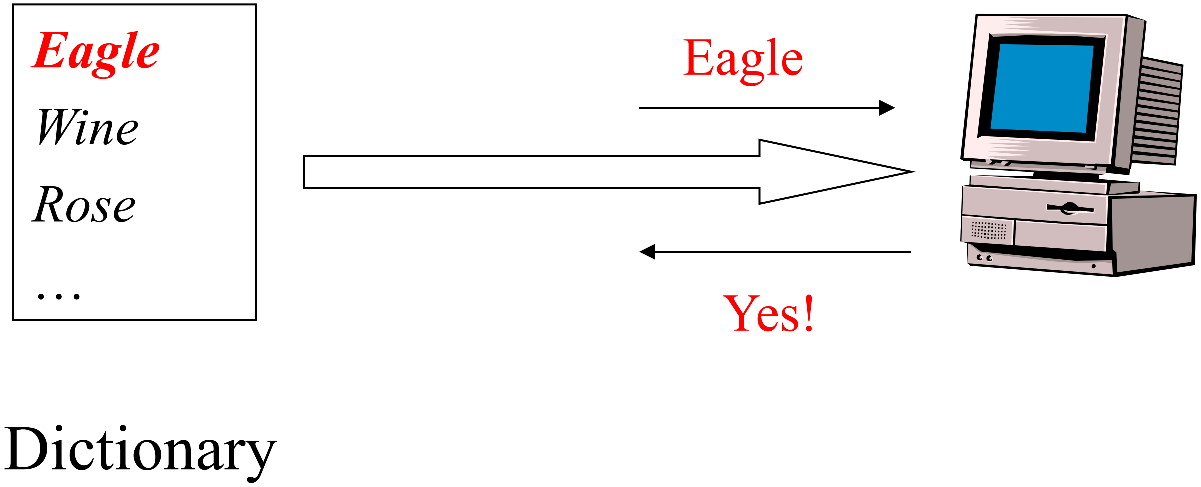
Length in characters	1	2	3	4	5	6
Number of passwords	15	72	464	477	706	605 (lower case only)

Common Password Choices

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

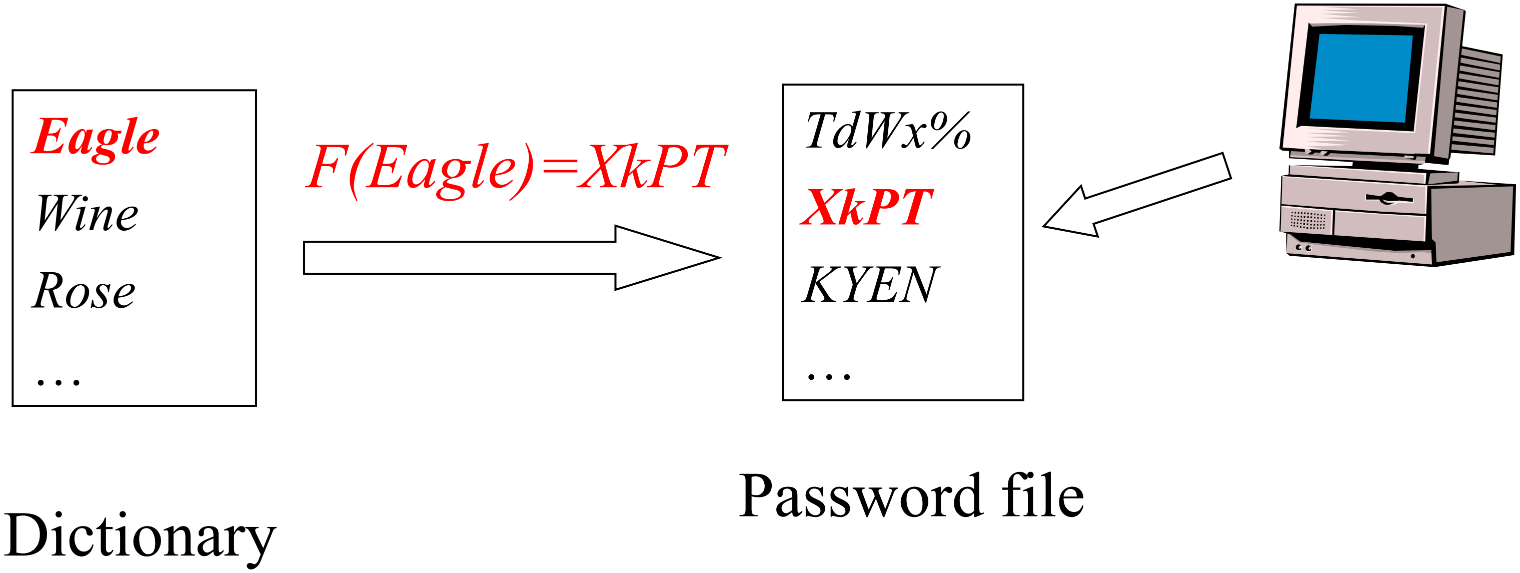
Dictionary Attacks

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



Dictionary Attacks (Cont'd)

- Attack 2 (offline):
  - Usually *F* is public and so is the password file
    - 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



Eagle
Wine
Rose
…

F(Eagle)=XkPT

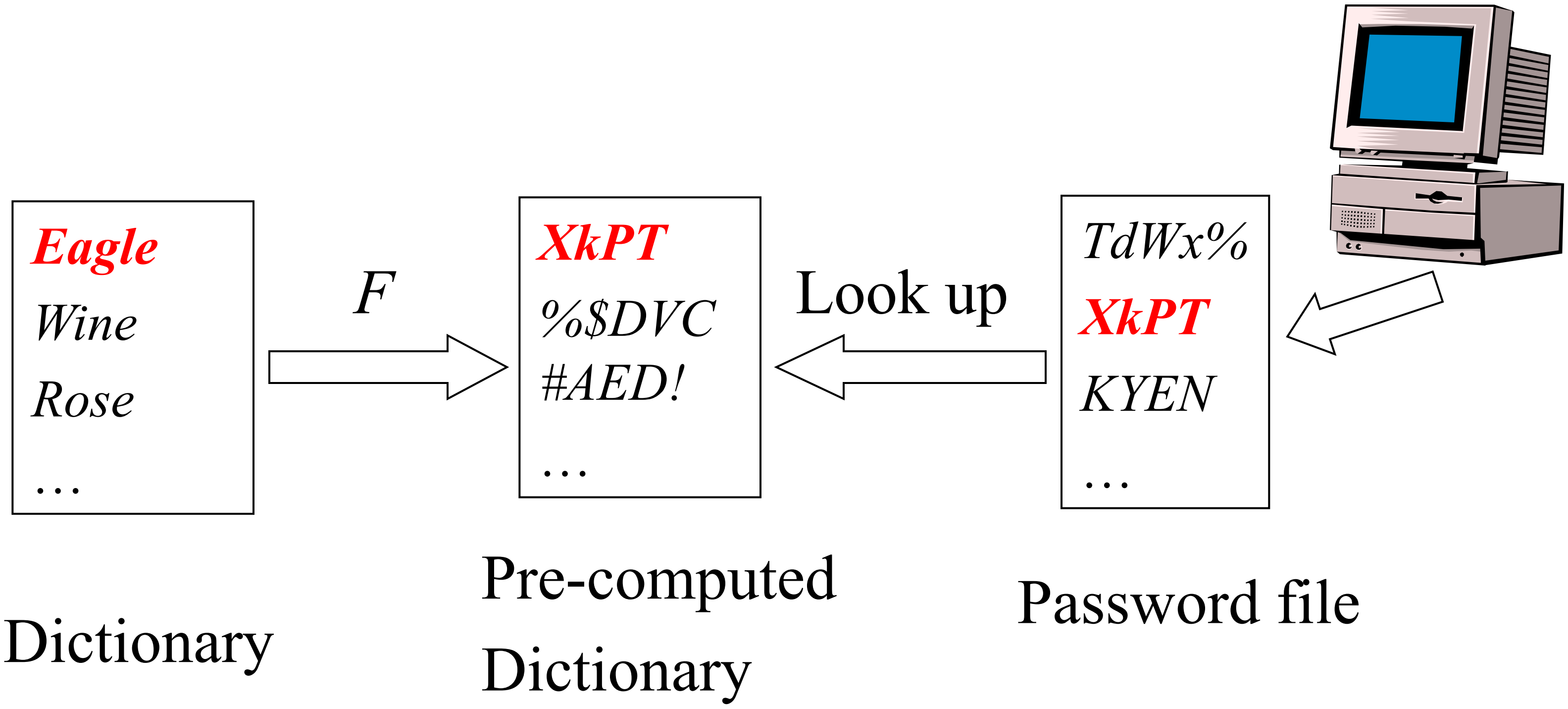
TdWx%
XkPT
KYEN
…

Dictionary

Password file

Dictionary Attacks (Cont'd)

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

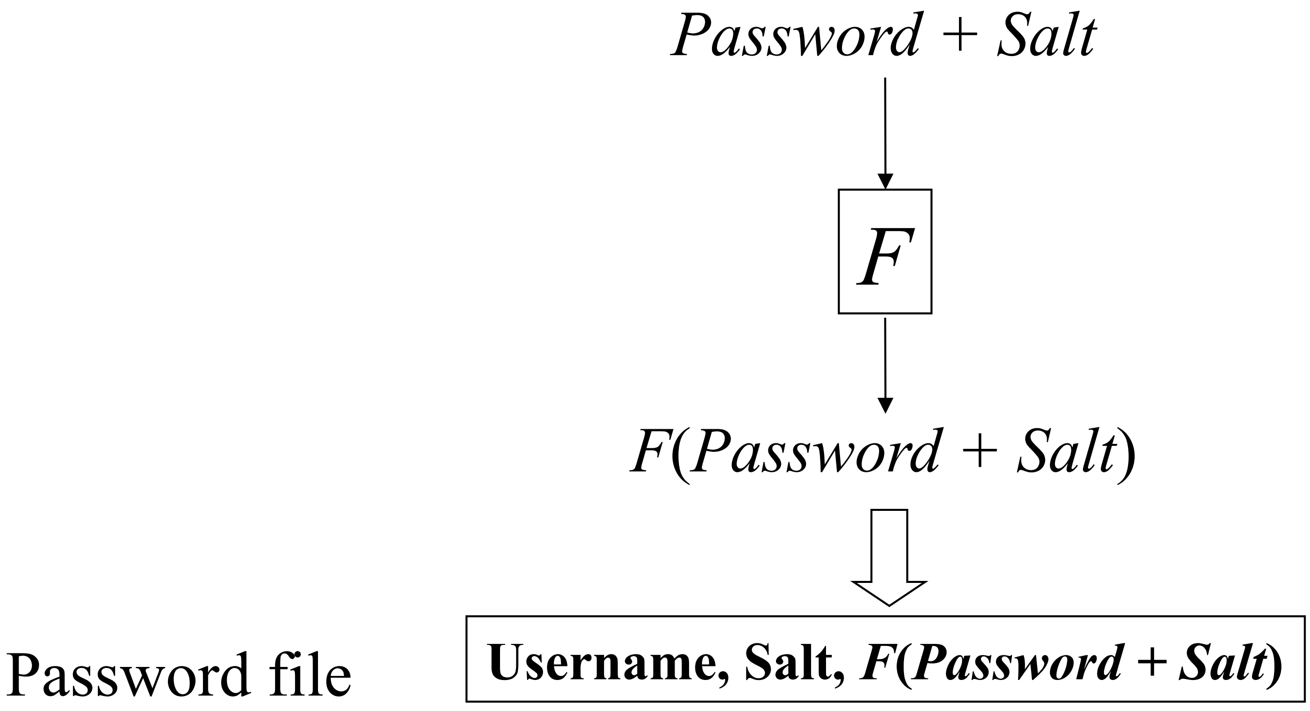


Password Salt

- 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*

F

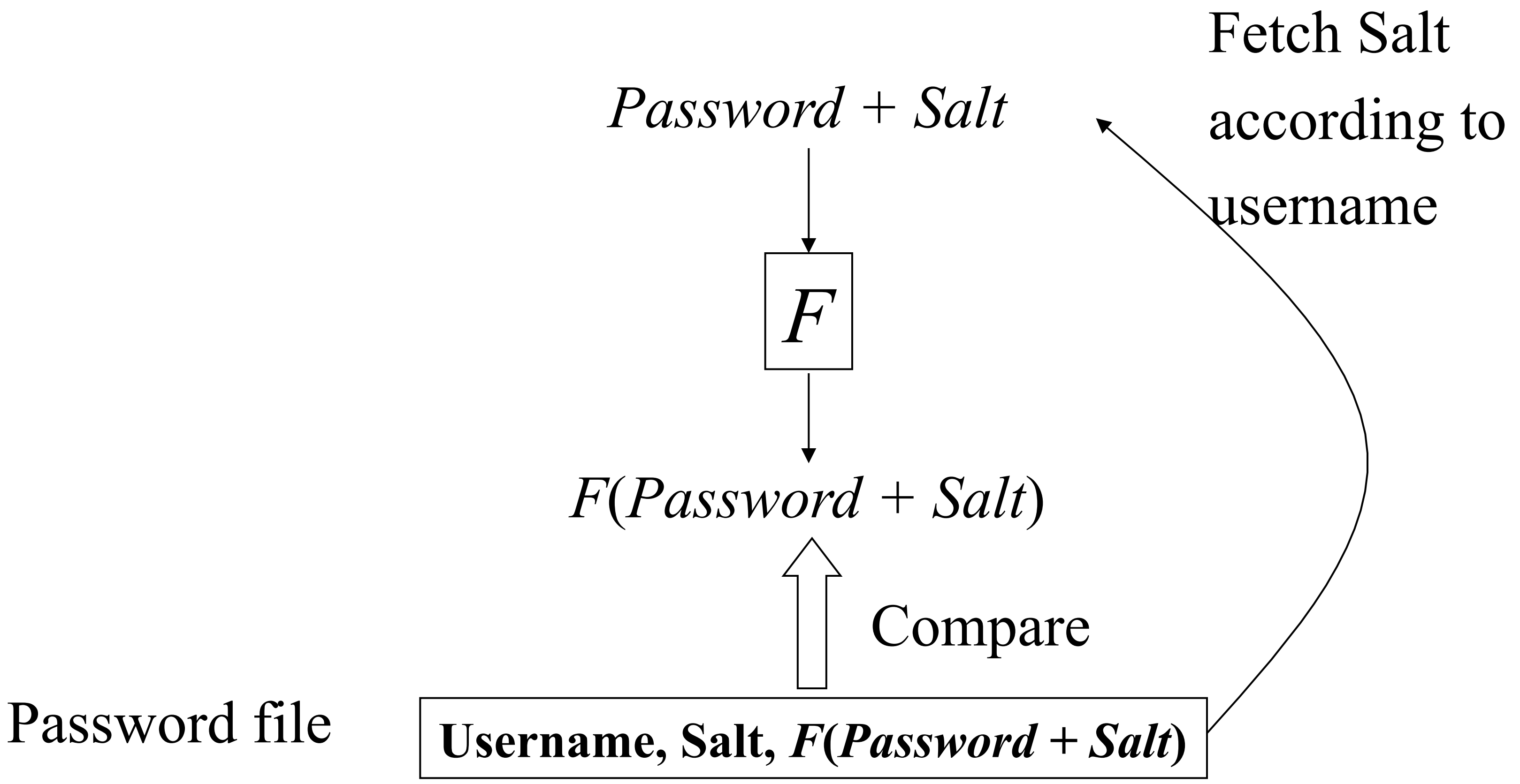
F(*Password* + *Salt*)

Password file

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

Password Salt (Cont'd)

- Verifying the passwords



Fetch Salt according to username

Password + *Salt*

F

F(*Password* + *Salt*)

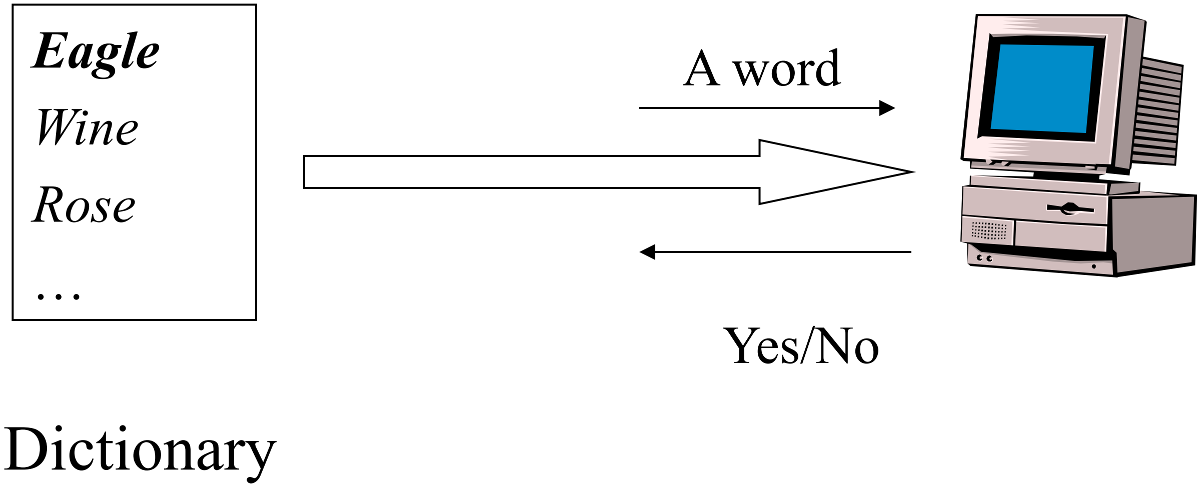
Compare

Password file

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

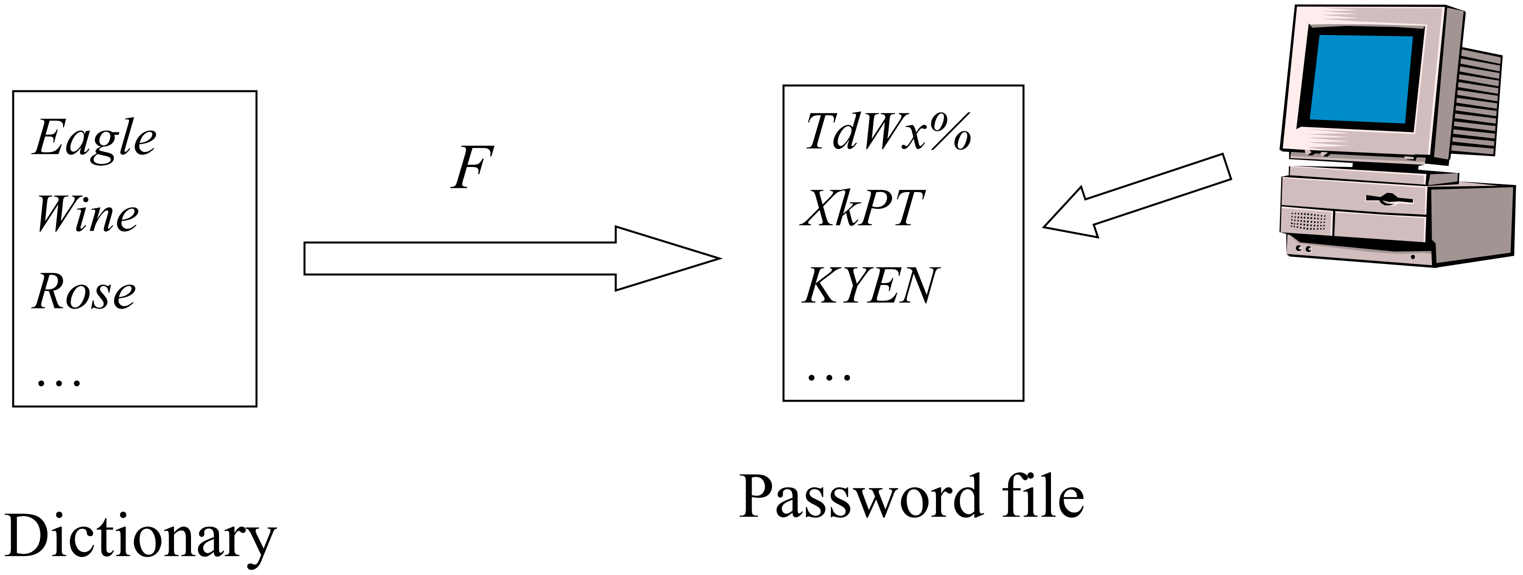
Does Password Salt Help?

- Attack 1?
  - Without Salt
  - With Salt



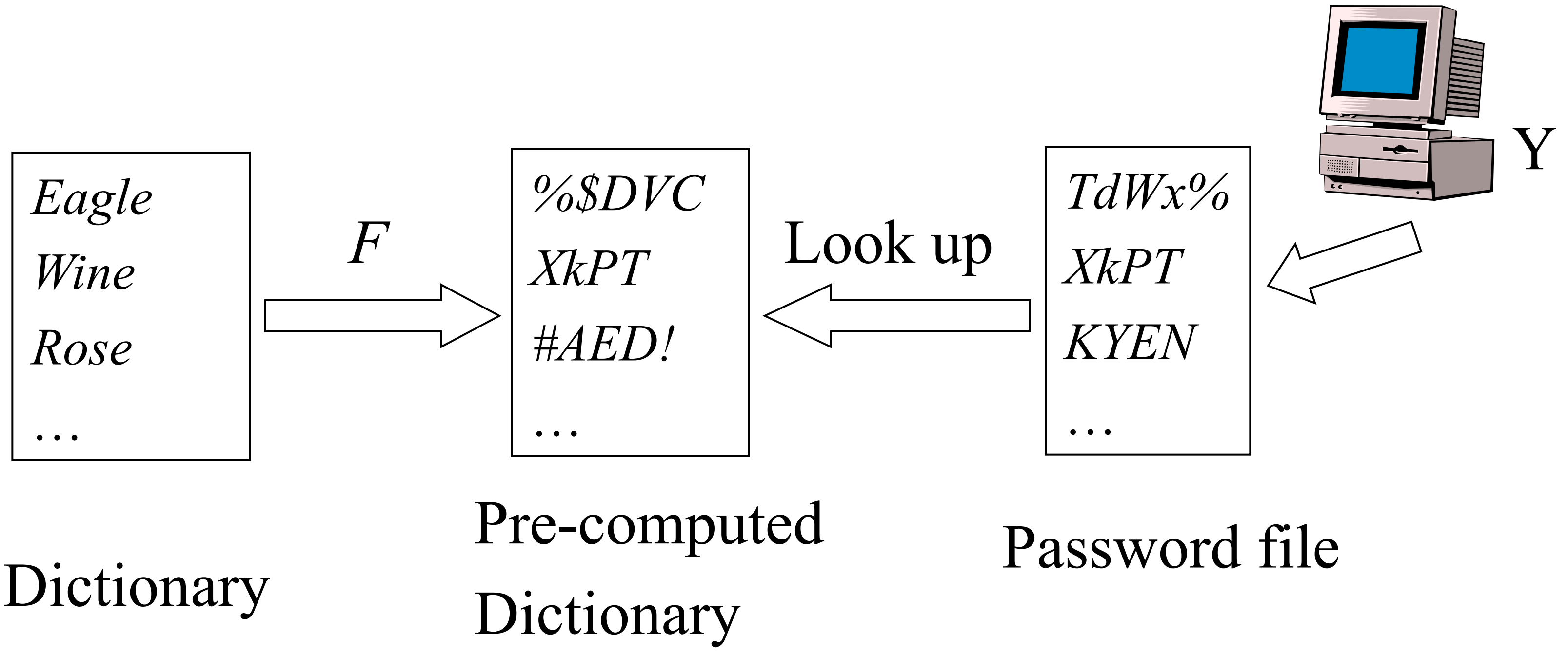
Does Password Salt Help?

- Attack 2?
  - Without Salt
  - With Salt



Does Password Salt Help?

- Attack 3?
  - Without Salt
  - With Salt (or change periodically?)



Eagle
Wine
Rose
…

F

%$DVC
XkPT
#AED!
…

Look up

TdWx%
XkPT
KYEN
…

Y

Dictionary

Pre-computed Dictionary

Password file

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

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
5. Be careful to choose the security problems and answers to recover your password
6. Etc.

Other Password Attacks

- Technical
  - eavesdropping on traffic that may contain unencrypted passwords (especially keystroke logging)
  - “Trojan horse” password entry programs
  - man-in-the-middle network attack
- “Social”
  - careless password handling or sharing
  - phishing

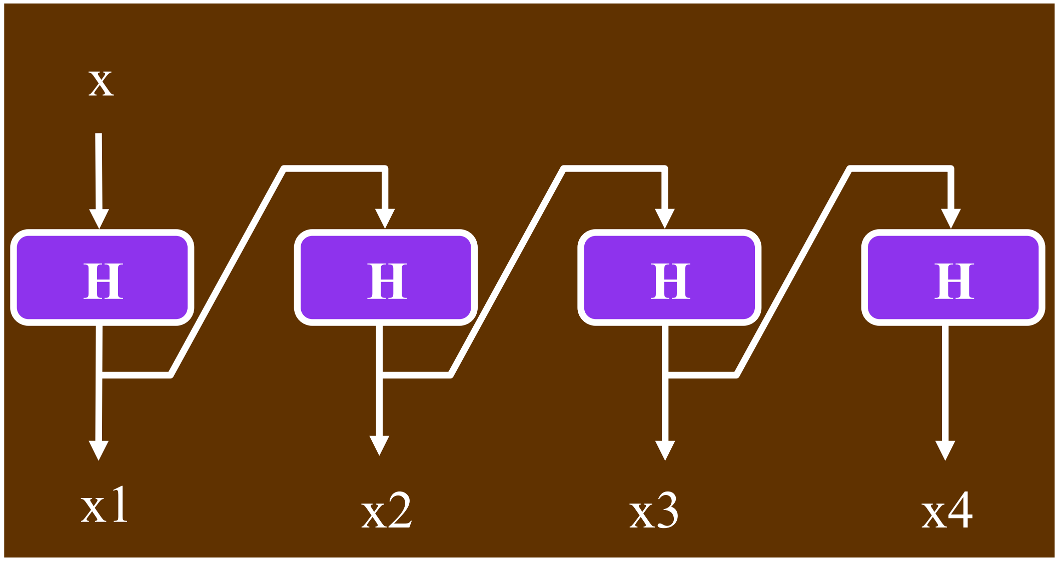
The S/Key Protocol

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
   - $x_1 = H(\mathbf{x})$
   - $x_2 = H(x_1)$
   - …
   - $x_n = H(x_{n-1})$



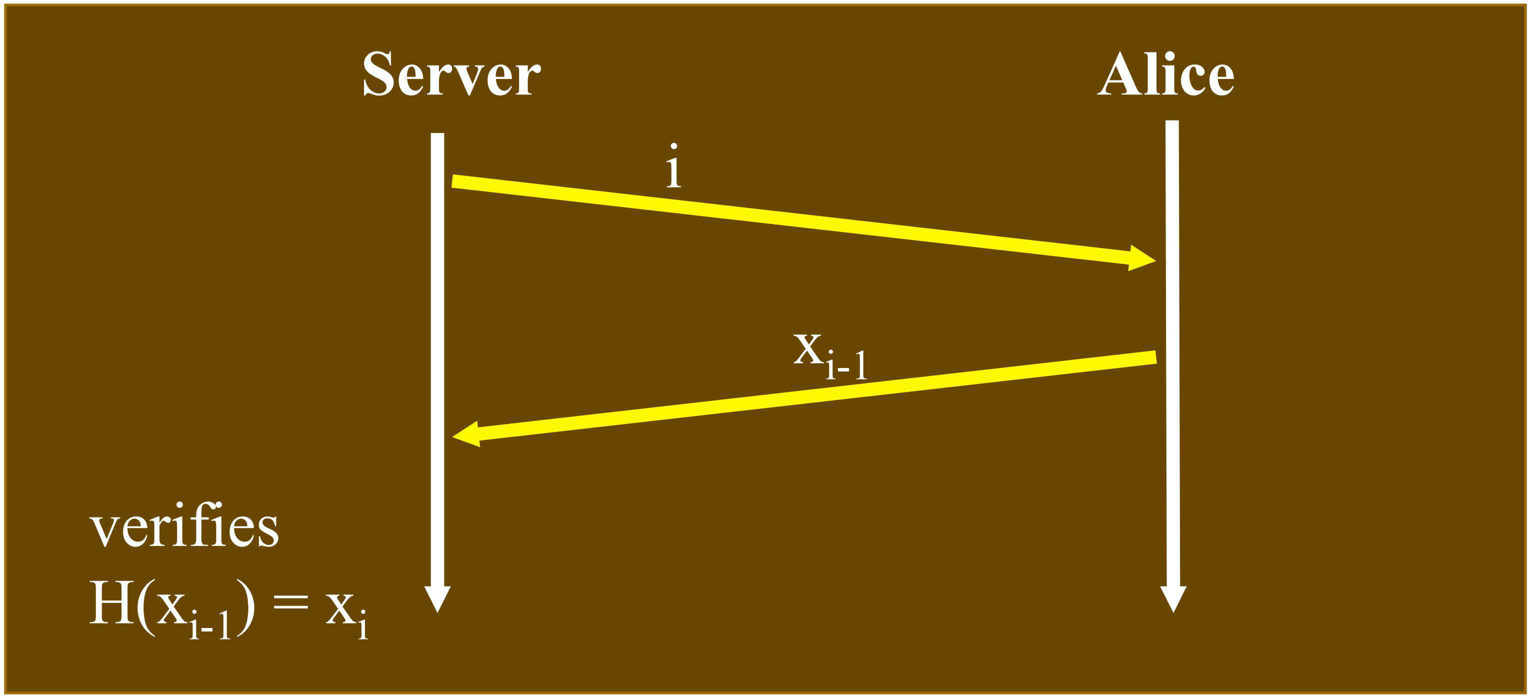
Generation… (cont'd)

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-1} such that $H(x_{i-1}) = x_i$
  - 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 masquerading as 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

Biometrics

- Relies upon physical characteristics of people to authenticate them
- Desired qualities
  1. uniquely identifying
  2. very difficult to forge / mimic
  3. highly accurate, does not vary
  4. easy to scan or collect
  5. fast to measure / compare
  6. inexpensive to implement
- Which of these are concerns for passwords?

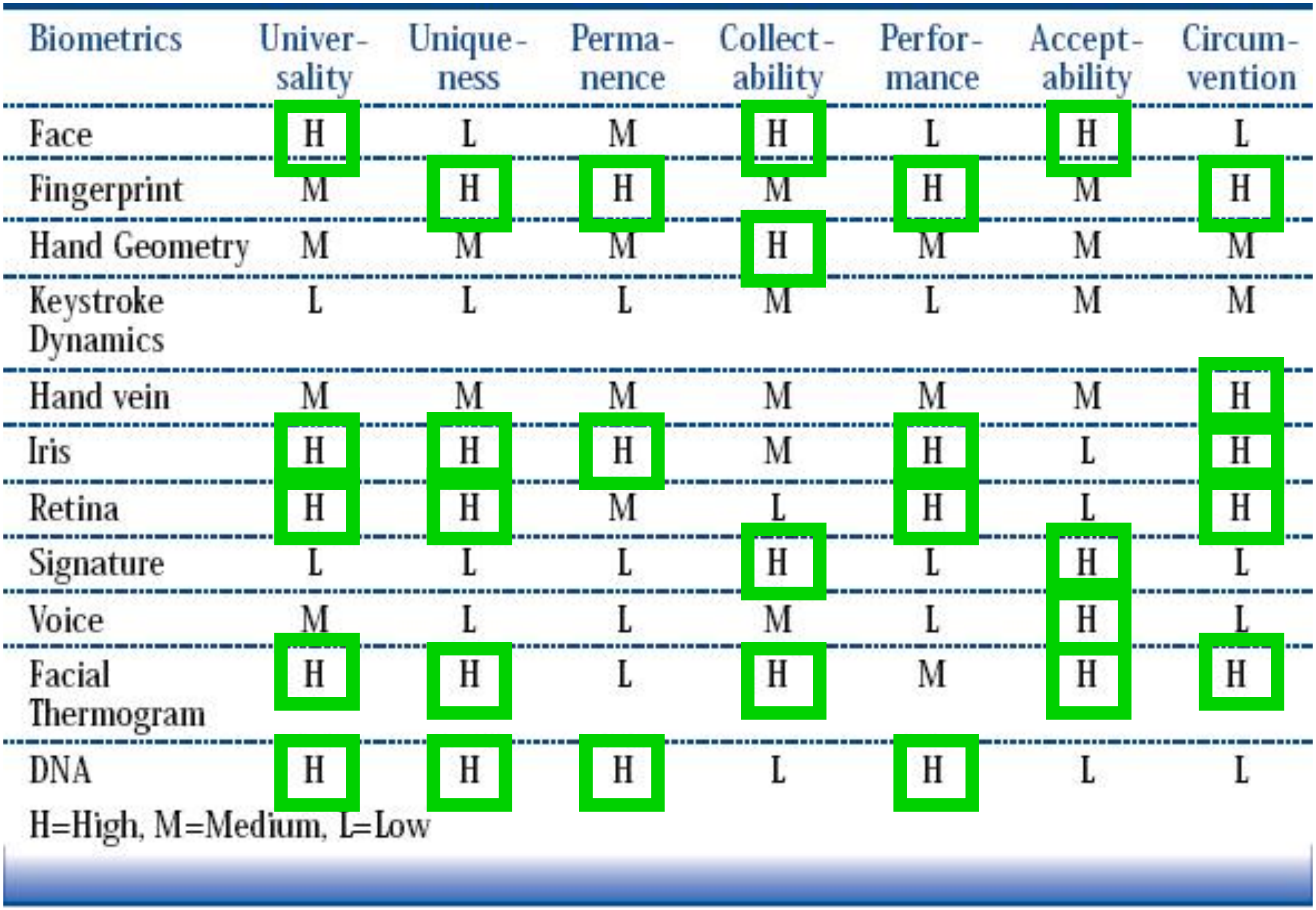
Assessment

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

Example Biometric Technologies

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

The Scorecard From One Study



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

Multifactor Authentication

- If one characteristic is pretty good, two or more characteristics should be better?
- Suppose true positive rate was AND of the two, and false positive rate was OR of the two…
  - $TP = TP1 * TP2$
  - $FP = 1 - (1-FP1)*(1-FP2)$
- Alternative: combine a biometric technique with passwords

Authentication Hardware (Tokens)

Tokens

- A token is a physical device that can be interfaced to the computer, and carries identifying information
- Types
  - passive tokens just store information
  - active tokens have processors and can perform cryptographic operations
- Examples
  - cards with magnetic strips
  - smart cards
  - USB storage devices
  - RFID tags

Design Issues for Tokens

- Cost
- Size
- Capabilities
- Robustness
- Resistance to tampering
- Usefulness if stolen / lost

An Example: *Time Synchronized* Tokens

- The token contains:
  - internal clock
  - display
  - a secret key
- Token computes a one-way function of current time+key, and displays that
  - this value changes about once per minute
- User reads this value and types it in to authenticate to the server
  - requires that server and token time stays synchronized

One-time Password on Smartphone

- Integrate physical tokens into smartphone
- Requirements:
  - Security
    - Malicious mobile OS cannot compromise the keying material in the one-time password (OTP) generator
    - It cannot read the OTP
  - Reliability
    - OTP works even if mobile OS crashes
    - Trusted inputs (e.g., clock time) for the OTP generator
    - Trusted display

TrustZone-based Solution

- ARM TrustZone Technology
  - Two isolated execution environments
  - Mobile OS cannot access the disk, memory, CPU states of the OTP generator.
  - A secure clock for OTP generator
  - A self-contained display and touchscreen.

Another Example: Alladin eToken

API / standards	PKCS#11 v2.01, CAPI (Microsoft Crypto API), Siemens/Infineon APDU commands, PC/SC, X.509 v3 certificate storage, SSL v3, IPSec/IKE
Security Algorithms	RSA 1024-bit / 2048-bit*, DES, 3DES, SHA1
Power source	Battery, 5 year lifetime
LCD	6 characters
Data retention	10 years

YOUR CUSTOMERS' ANTI-PHISHING SOLUTION
YOUR PKI SMART CARD
YOUR DATA PROTECTION KEY
YOUR EMPLOYEE'S SECURE REMOTE ACCESS
YOUR CUSTOMERS' DIGITALLY SIGNED TRANSACTIONS
YOUR SECURE PASSWORD MANAGEMENT KEY
YOUR EMPLOYEES' SECURE NETWORK ACCESS
YOUR SECURE EXTRANET ACCESS
YOUR LAPTOP SECURITY
523745

Summary

1. Passwords are by far the most widely used form of authentication, despite numerous problems
2. Biometrics hold promise but are expensive, inconvenient, and compromise privacy
3. Two factor authentication is commonly used for higher security
4. One-time passwords (S/Key) are attractive, especially if combined with hardware