Authentication: Passwords

CSC 154

Slides borrowed from Peng Liu, IST, PSU

Authentication is critical

- Imagine the scenario when a bad guy can convince people that “he is you”
  - He can use your credit cards
  - He can withdraw money from your bank account
  - He can buy a new house by adding a new loan on your shoulder
- Identity theft is not that rare today

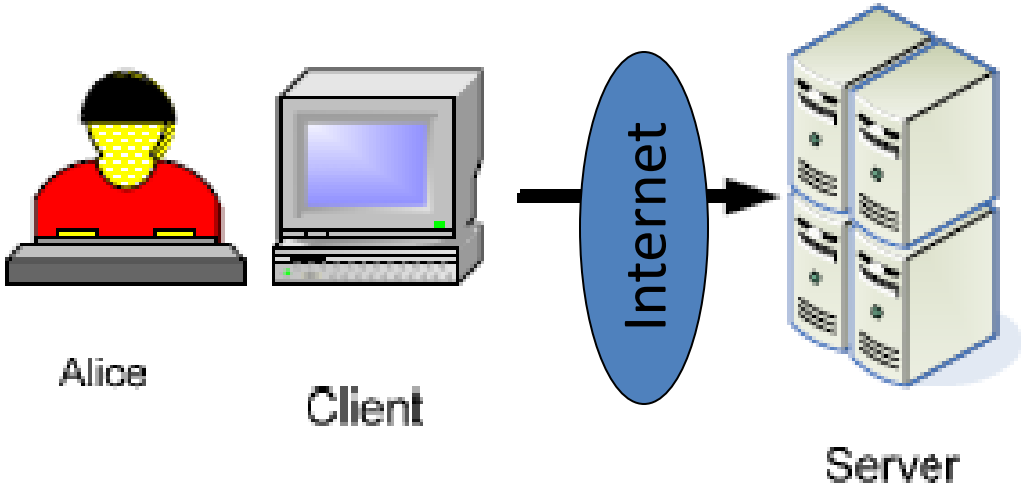
Three authentication problems in network security

- Identity authentication
  - When you log into an email server via a browser, you need to prove your identity
  - When you log into EBay
- Message origin authentication
  - Is the sender authentic or impersonated?
- Message content authentication
  - Is the message authentic or altered?

How can I prove my identity?

- You could NOT use your name or CSUS ID or SSN to prove your identity
  - They are known to many people
- You need to use something:
  - Only you know (a password)
  - Only you have (a fingerprint)
  - Only you own (a smart card)

How authentication is done: the simplest method



Browser

EBay

ID	Password
Alice102	aliceebay
Bob103	bobebay
… …	… …

The Password File

Alice types in

-- User ID: Alice102

-- Password: aliceebay

The simplest method has two big security flaws

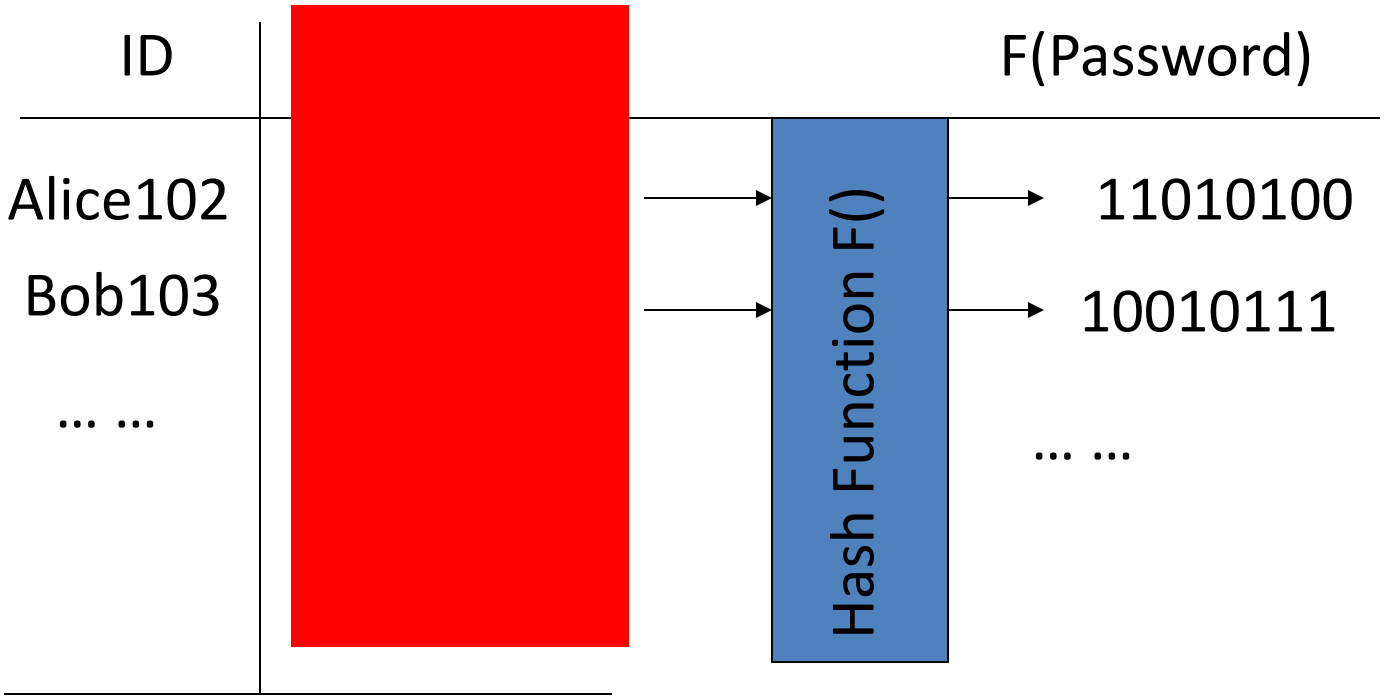
- [Flaw 1] The password is sent to EBay web server in clear-text
  - What if Mallory is sniffing
- [Flaw 2] The password file contains a lot of clear-text passwords
  - What if the password file is read by a corrupted insider

To remove flaw 1

- Idea 1: encrypt the password
  - 1A: if you use symmetric cryptography, you need to solve the key exchange problem
    - Meet in Starbuck, “I will call you”, IKE, Kerberos
  - 1B: if you use public key cryptography, you do not have the key exchange problem
    - Alice uses the EBay server’s public key to encrypt her password, only EBay can decrypt it because only EBay knows the matched *private* key
- Idea 2: one-time password
  - Because a password will never be used again, no need to hide!

To remove Flaw 2

- Idea: hash the passwords stored in the password file!

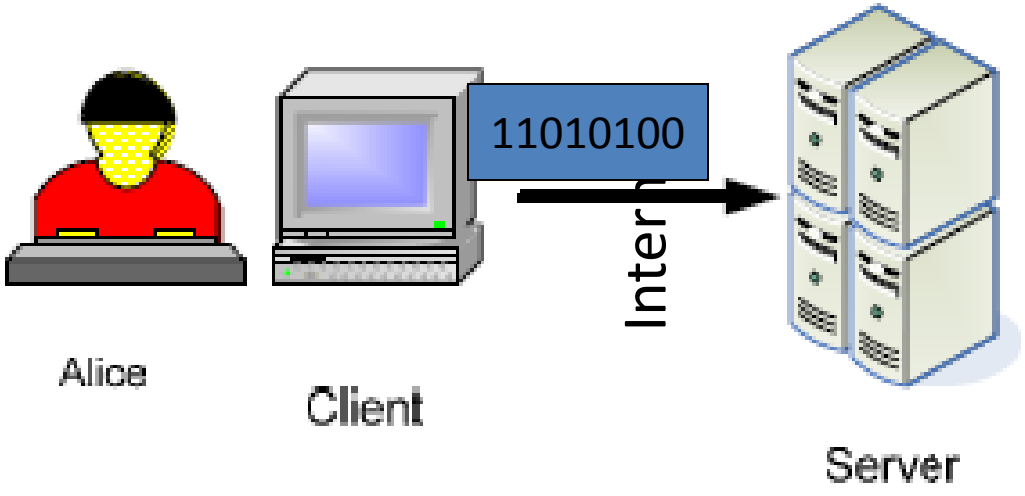


The benefit of hashing passwords

- The corrupted insider can still read the password file, but he will get a bunch of “digested” passwords
  - Each digested password has 8 random bits
- The hash function is a juice maker known to everybody, but the hash function is one-way: non-reversible
- So the bad guy *cannot* know any passwords

How to use a hash password file to do authentication?

- Idea 1: ask the browser to hash the password and send only the hash to the EBay server
  - However, idea 1 has a big flaw!



ID	F(Password)
Alice102	11010100
Bob103	10010111
… …	… …

Browser EBay Password File

The flaw of Idea 1

- Flaw: the digested passwords (i.e., hashes) stolen by the corrupted insider can still be used to fool the EBay server
  - The bad guy does not have Alice's password, but he can let his browser send a hash he steals from the password file to the server
- How to remove the flaw?
  - Answer: let the browser send the password to the server; let the server calculate the hash before matching the password file
  - Of course, the password should be encrypted in transmission

Other password attacks

- Brute-force attack
  - Try every possible combinations of valid symbols
  - Too time consuming
- Dictionary attack
  - “I know many people use birthdays as passwords, and they are only 366 possible birthdays!”
- Spoofing and Phishing attack
- Social engineering

One Time Password
Reusing passwords is risky

- Whenever you disclose your password
  - You lost your walet
  - You lost your contact book
  - People pass by your cubicle and read a sticker
  - In a bar: social engineering attack
  - Spoofing and Phishing attack
- Mallory can do dictionary attacks or the brute-force attack against the password file
- Although your password is encrypted when you log on EBay,
  - Mallory can sniff and get the ciphertext of your password
  - Then Mallory can do ciphertext only crypto-analysis

How to avoid reusing passwords?

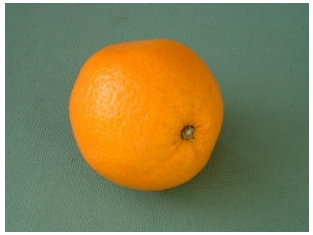
- One time passwords
  - Every password is used only once
  - No password is ever reused
  - Old passwords are useless in guessing the next one-time password

How to get one-time passwords?

- Idea 1: S/Key
- Idea 2: Time synchronized authentication
  - Using the current time to generate the current password
- Idea 3: challenge-response authentication
  - Secret key cryptography based challenge-response
  - Public key cryptography based challenge-response

S/Key (1)

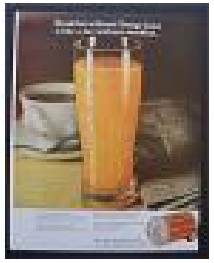
- S/Key is based on the idea of “hash of hash”



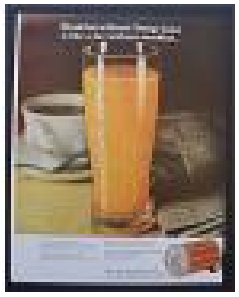
Seed x

F

Hash Function
(Juice Maker)



Hash F(x)



Hash F(x)

F

Hash Function
(Concentrate Maker)

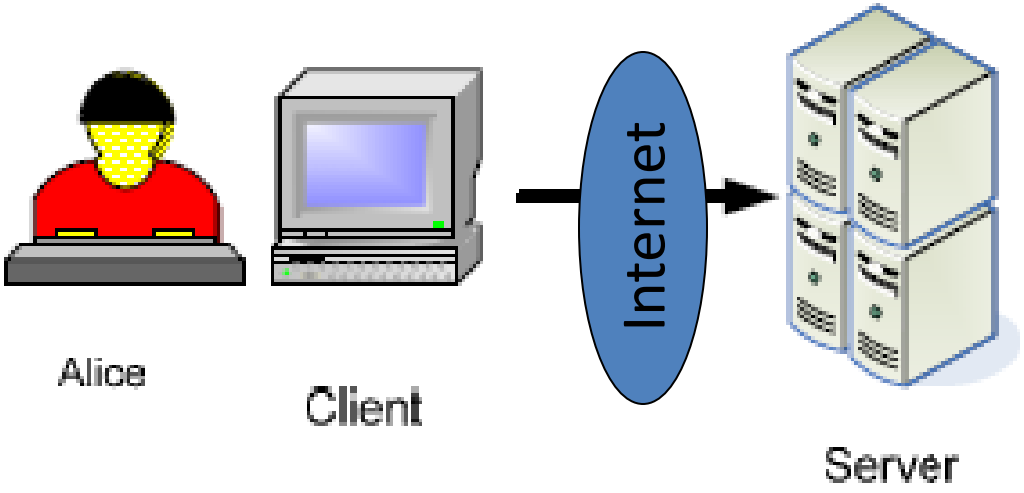
Hash of hash $F(F(x))=F^2(x)$
(orange concentrate)

S/Key(2)

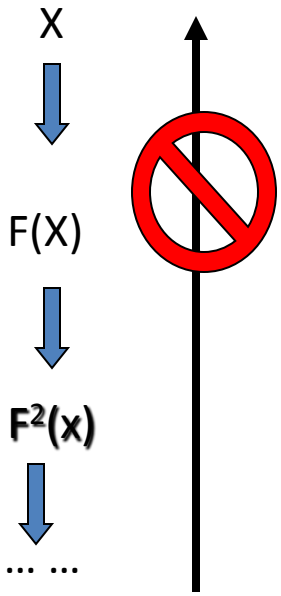
- Given seed x, let's generate 3 one-time passwords
  - F(x)
  - F(F(x)), denoted $F^2(x)$
  - F(F(F(x))), denoted $F^3(x)$

S/Key (3)

- Given the 3 one-time passwords F(x), $F^2(x)$, and $F^3(x)$, which one to use first?



S/Key (4)



- Observation 1: if F(x), the juice, is the first password to use, and $F^2(x)$, the orange concentrate, is the second password to use
  - Security Breech: Mallory can use the first password (he sniffers) to derive the second one, because orange concentrate can be perfectly made from orange juice

S/Key (5)

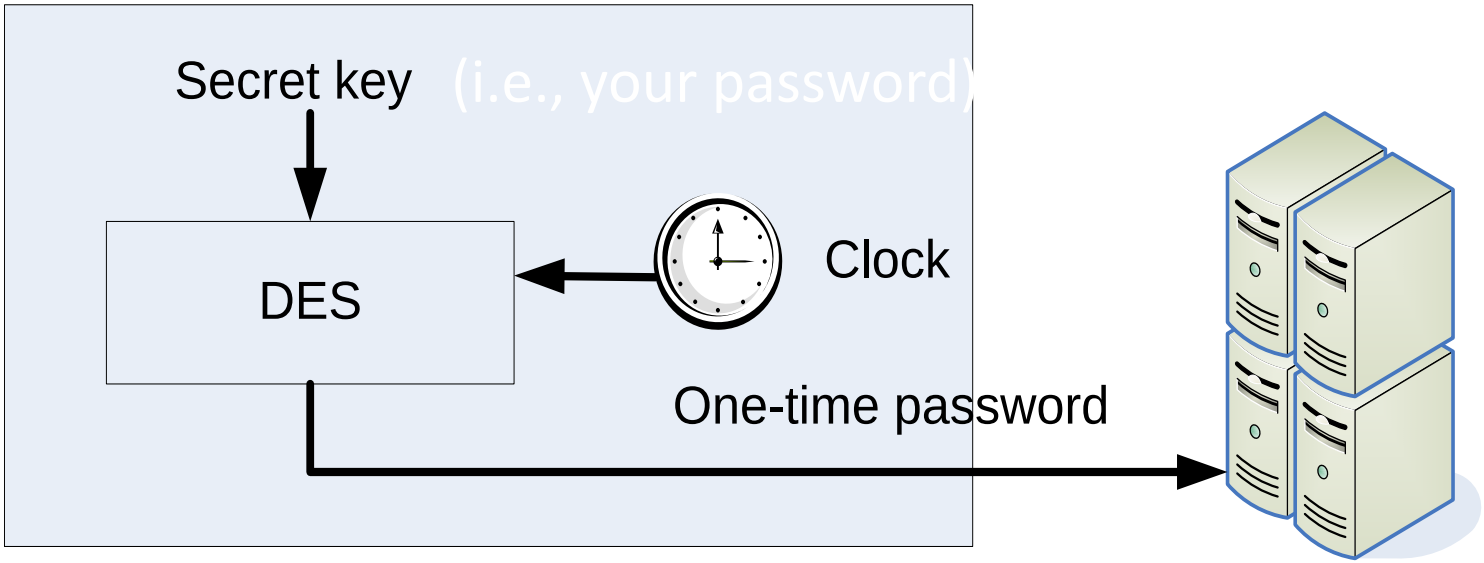
- Hence $F^3(x)$ should be first password to use, $F^2(x)$ should the second, and F(x) should be the last password to use
- Observation 2: from $F^3(x)$, Mallory cannot guess $F^2(x)$
- Observation 3: from $F^2(x)$, the concentrate, Mallory cannot get F(x), the original juice
- Observation 4: Mallory cannot guess the value of x from any password

S/Key (6)

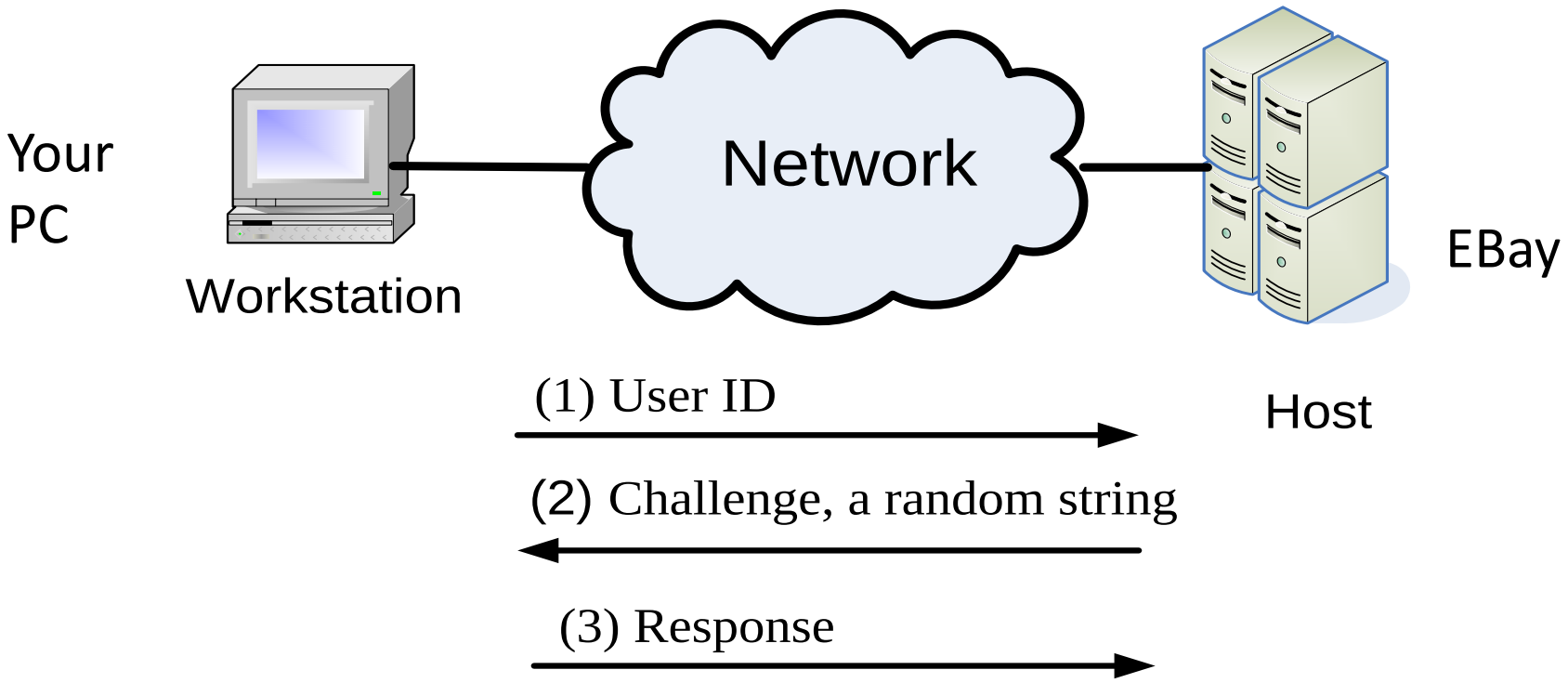
- The server still needs a password file to authenticate; but the password file keeps on changing
- When Alice uses $F^3(x)$ to log in, the password file stores: $F^4(x)$
- When Alice uses $F^2(x)$ to log in, the password file stores: $F^3(x)$
- Stealing the password file is useless!

Time-synchronized authentication

- Idea: the time will never go backward



Challenge-response authentication

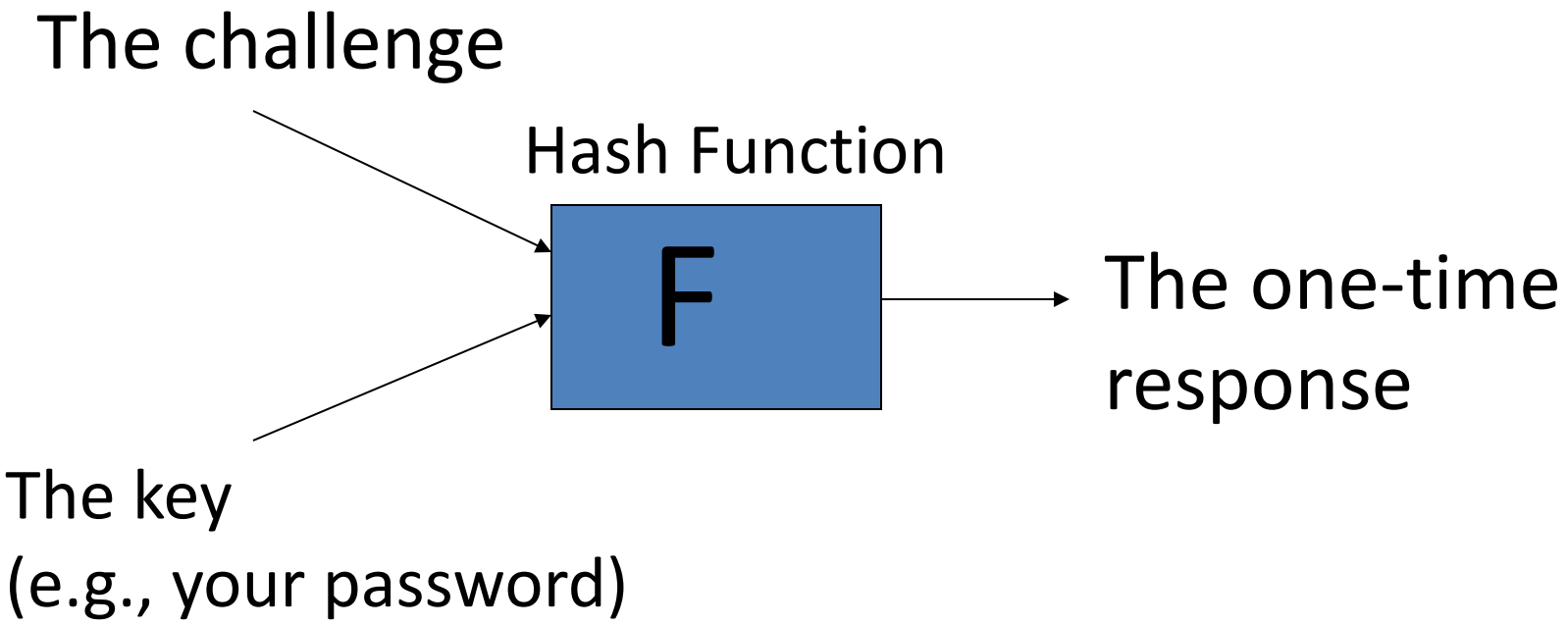


- Note: your response is actually a one-time password, because no challenge will be reused

How to generate one-time response

- Idea 1: secret key cryptography
  - E.g., keyed hash functions
- Idea 2: public key cryptography

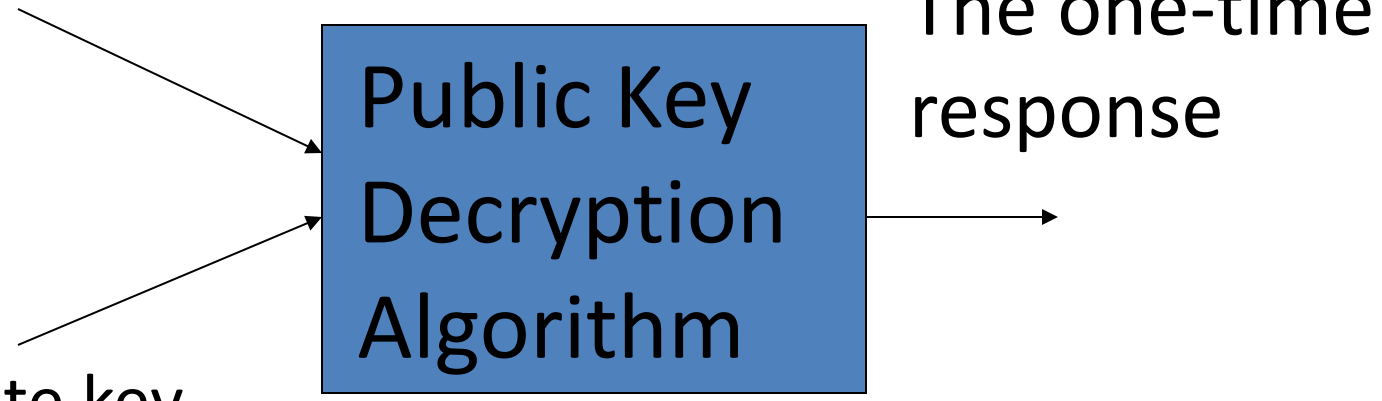
Idea 1: use a keyed hash function to generate one-time response



Remark: Because only you know the key, Mallory cannot generate the same one-time response!

Idea 2: use public key cryptography to generate one-time response

The challenge: a random string encrypted by your public key



Remark: Because only you know your private key, Mallory cannot decrypt the challenge!

Hash Chain

- Using hashes as authenticators
- Scenario: Alice (Instructor) wants to inform Bob (Student) whether a class is canceled through a 3-rd person Mallory (TA)
  - Mallory can be evil
    - 1. Alice invents a token t
    - 2. Alice directly gives Bob h(t), where h() is a hash function
    - 3. If Alice decides to cancel a class, she gives t to Mallory to give to Bob
      - If Alice does not cancel the class, she does nothing;
      - If Bob receives t from Mallory, he know the class is canceled.
        - » Mallory could not get t unless Alice decides to cancel a class and let him know. In this way, Mallory could not **fake** Alice's decision.
          - Mallory can simply choose not to forward the token t to keep Bob uninformed of a class cancellation. But, that's OK. Bob will know it when he sees the empty class.

Example borrowed from Trent Jaeger, CSE, PSU

Hash Chain

- What if Alice wants to prepare to cancel all the 34 classes in this semester?
  - 1. Alice invents a secret token t
  - 2. Alice directly gives Bob $h^{34}(t)$, where $h^{34}(t)$ is 34 repeated uses of h().
  - 3. If she cancels class on day d, she gives $h^{34-d}(t)$ to Mallory, e.g.,
    - If cancels on day 1, she gives Mallory $h^{33}(t)$
    - If cancels on day 2, she gives Mallory $h^{32}(t)$
    - …….
    - If cancels on day 33, she gives Mallory $h^{1}(t)$
    - If cancels on day 34, she gives Mallory t

    - If does not cancel class, she does nothing

- Summary: You can infer $h^{i}(t)$ from $h^{j}(t)$ if i>j, but $h^{i}(t)$ will be from earlier class cancellations, and will provide no information on later class cancellations