CIS 4360
Secure Computer Systems

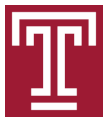
Entity Authentication

Professor Qiang Zeng

Spring 2017

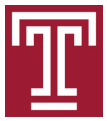
Previous Class

- PKI
  - Digital Certificate
  - Certificate Authority
  - Verifying Certificates and Chain of Trust
  - Revoking Certificates
- Kerberos



Previous Class

- Both PKI and Kerberos can be used for authentication; But PKI is mainly used to authenticate a service, while Kerberos is to authenticate both services and users
- PKI mainly builds on asymmetric cryptography, while Kerberos mainly builds on symmetric cryptography
- PKI is used over the Internet, while Kerberos is typically used within a single organization

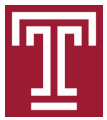


Previous class…

Is PKI subject to Single Point of Failure?

Yes. If the CA system is compromised, all the certificates issued by the CA system are in danger.

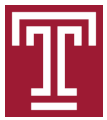
In Kerberos, the client needs to contact the KDC at each authentication, while the client (e.g., your browser) in the PKI only needs to contact the CA system infrequently and periodically for downloading the CRL (Certificate Revocation List).



Previous class…

What is the big pain of PKI?

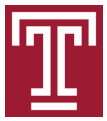
Revocation of certificates. Without the issue, the CA system could have been kept completely offline.



Previous class…

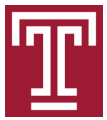
What happens when you type "citi.com"

(1) Domain name resolution: to get the IP of citi.com
(2) TCP handshake: to establish TCP connection
(3) SSL handshake: to establish cipher-suite and the master key
(4) Get homepage, which is encrypted and authenticated using the session keys derived from the master key



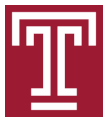
Outline

- What is Entity Authentication?
- The purpose of Entity Authentication?
- How to perform Entity Authentication?



What is Entity Authentication?

- **Entity Authentication** is the process of verifying credentials that are used to prove the identity of an entity
  - Determining whether someone or something is, in fact, who or what it is declared to be
  - Usually performed at the beginning of a session
- Different from Message Authentication, which only authenticates the origin of one message

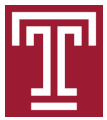


Purposes of Entity Authentication

- Entity Authentication usually precedes Authorization
  - Authentication establishes identity, while Authorization dictates what an identity is allowed to do

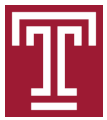
What is Identity?

- It tells who you are.
- Each entity has a lot of identities. E.g., you are
  - A student in the classroom
  - A customer in a bank
  - A patient in a hospital
  - The owner of your laptop
  - A passenger when taking a flight



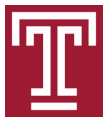
How to perform Entity Authentication?

- Credentials are evidence used to prove identities
- Three categories
  - Something you know
  - Something you have
  - Something you are



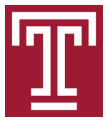
Something You Know

- In real-world
  - SSN
  - Credit Card Number
  - Mother's maiden name
  - …
- In cyber-world: Passwords
  - Generally weak
  - One password used in multiple places
  - Subject to password-guessing attack



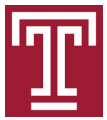
How to store user passwords

- Store hash values only (i.e., never store passwords)
  - It will be a disaster if you store user passwords as plaintext and the server is compromised
- Adding "salts" when hashing
  - If "hash(password)" is simply stored, the attacker who obtains the hash value can simply search in a pre-computed rainbow table, which contains the hash values of a large number of passwords
  - Store "salt1, hash(salt1, password1); salt2, hash(salt2, password2); …"
  - Now the attacker has to crack the password one by one; the pre-computed table is useless
- Using a slow hash algorithm
  - Why? Let's discuss Dictionary Attack first



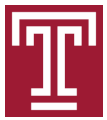
Brute Force Attack

- The attacker may try every possibility of a password
- E.g., a perfectly random 8-char password has less entropy than a 56-bit key
  - Each char can have 2^7 possibilities
  - Thus, an attack needs to try at most 2^{7x8} times
- Can the attacker do better?



Dictionary Attack

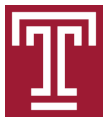
- In contrast to Brute Force attack, Dictionary Attack tries the collection of passwords that are frequently used, e.g., “123456”, “password”, “qwerty”
- Even passwords are stored with salts, the attacker can apply Dictionary Attack (or even Brute Force attack)
- If a fast hash algorithm is used, it helps the attacker speed up such attacks



Question

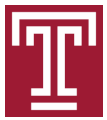
Should cryptographic hash algorithms, e.g., MD5, SHA-1, SHA-2, be used for storing password hashes?

No. These hash algorithms are designed to be quick. Fast hash algorithms favor attackers to crack passwords once the hash values are leaked



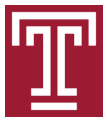
How to obtain a slow hash algorithm?

- "salt, $Hash^n$(salt, password)"
  - E.g., n = 1024;
  - n should increase if faster CPUs are produced
- "Hash" can be, e.g., MD5, SHA1, SHA2
- It significantly increases the time required to perform Dictionary Attack or Brute Force Attack



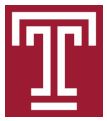
Password Hashing Algorithm: Bcrypt

- It takes around 100ms to compute the hash value of a salted password
- 100ms is fast enough that the user won't notice when they log in, but slow enough that it becomes less feasible to execute against a long list of likely passwords.
- For instance, if a hacker wants to compute bcrypt() against a list of one billion likely passwords, it will take about 30,000 cpu-hours (about $1200) --- to crack a single password



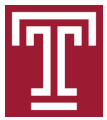
68 Million Hacked Dropbox Accounts

- Hacked and on sale now (since Sept, 2016)
- 32 million of the passwords are protected with bcrypt --- they should be fine
- Others are protected using salted SHA-1 --- and, it is said that salts are not leaked --- so, it is unlikely to crack the passwords with the hash values only
- What lessons do you get?



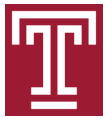
Something You Have

- Digital Certificates
  - If you trust the CA, you trust the certificate issued by her and the identity dictated on the certificate
- Security tokens
  - Electronic keys used to prove one's identity



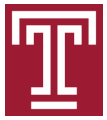
Multi-factor Authentication

- Multi-factor Authentication is an authentication method that requires the user to provide multiple pieces of credentials for authentication, from typically at least two of the three categories:
  - Something you know (knowledge factors)
  - Something you have (possession factors)
  - Something you are (inherence factors)
- Two-factor Authentication is a type of Multi-factor Authentication



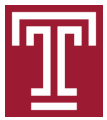
Security Tokens

- Examples
  - One-time-password token, e.g., RSA's SecurID tokens
  - Mobile token app
  - Smartcard, e.g., Credit Card
  - RFID token, e.g., E-Zpass
  - USB, Bluetooth, NFC etc.
- One type of widely used security tokens generates and displays one-time passwords (OTPs), where the generated password cannot be used twice
  - E.g., RSA's SecurID token
- Another type relies on an mobile token app to generate and display one-time passwords
  - E.g., Google's two-factor authentication



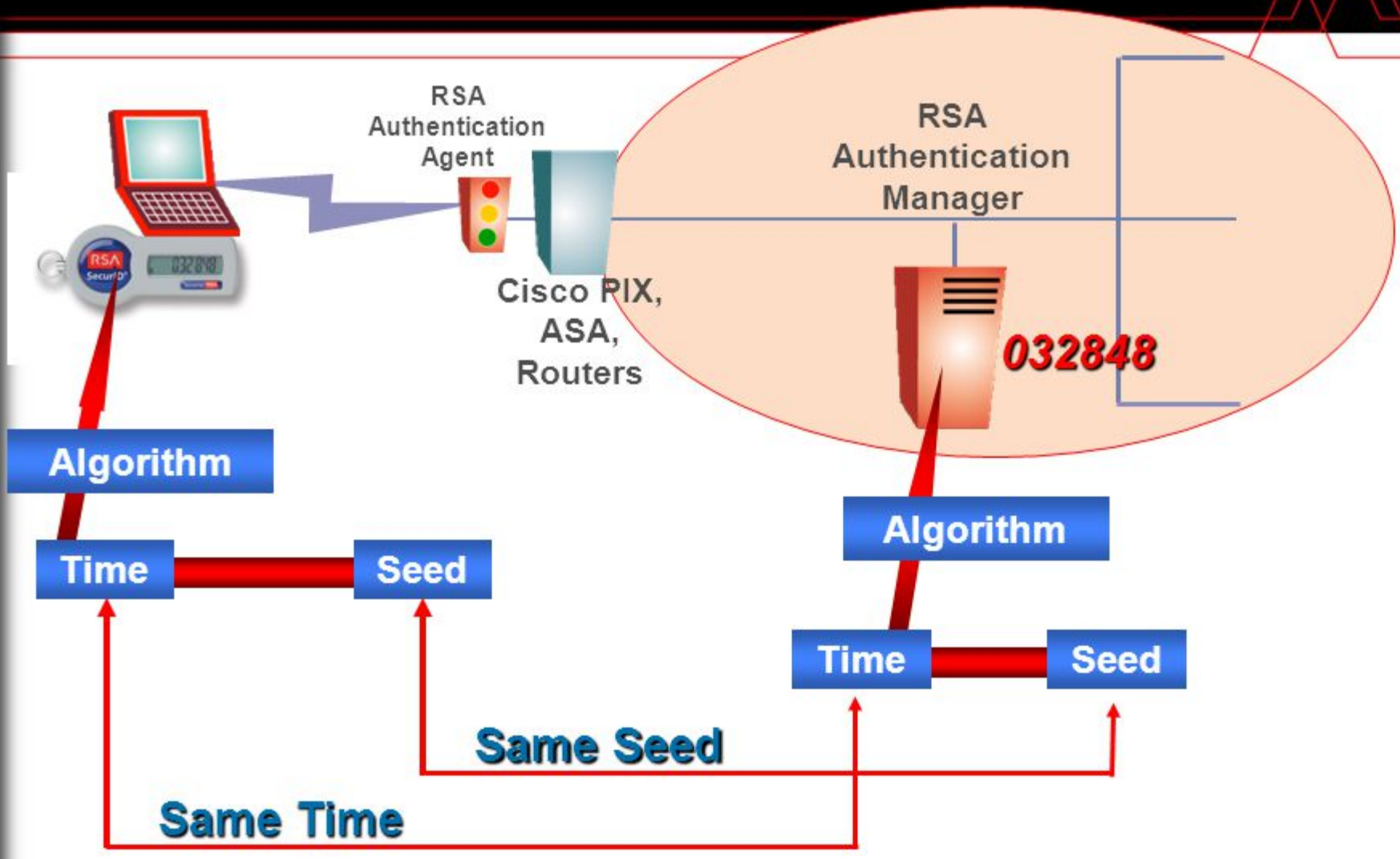
Case Study: RSA's SecurID Tokens

- OTP = Hash(time, seed) // time-based OTP
- Generate once per 30 or 60 seconds
- Seed is a 64-bit/128-bit secret
- Tamper resistant, so the seed cannot be stolen



RSA SecurID

Time Synchronous *Two-Factor Authentication*



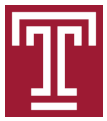
RSA SECURITY

Question

RSA's SecurID uses a proprietary hash. What other algorithms that we have learned can be used instead?

Any keyed hash (e.g., HMAC) or block cipher (e.g., AES) algorithm also works.

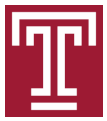
HMAC(seed, time)
Encrypt(seed, time)



Question

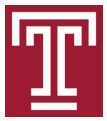
What if the clock in the token and that in the server get out-of-sync over time?

The solution to the clock drift problems is to store a "drift" parameter per token on the server. The parameter can then be adjusted to tackle clock drift



Case Study: Google's mobile app token

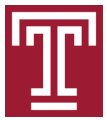
- Generated once per 30 seconds
- TC = Floor[(current_time - epoch) / 30]
- OTP = HMAC(key, TC)



Question

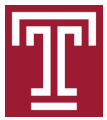
If there is no Wi-Fi or cellular signal available for my cell phone, can I still use Google's two-step verification to login in at a computer?

Yes, Google Authenticator generates passcodes based on time.



Summary

- Credentials
  - Something you know (Knowledge factors)
  - Something you have (Possession factors)
  - Something you are (Inherence factors)
- Multi-factor authentication
- Time-based One Time Password (OTP)
- Case studies
  - RSA's SecurID
  - Google Authenticator



Writing Assignments

- When you go to an ATM machine to withdraw money, is it two-factor authentication?
- If you are the security architect of your company, how would you store user passwords? Why?