# CSE345/545: Mid-semester Exam

Mihir Chaturvedi (2019061)

---

**1. a.**

- Name: Mihir
- Length: 5
- 5 mod 2 = 1, therefore **3-anonymized table**

| **Customer ID** | **Name** | **Place** | **City** | **Country** | **No. items purchased** | **Price** |
|---|---|---|---|---|---|---|
| C000** | * | * | * | * | 2 | 6000 |
| C000** | * | New York | New York | USA | 2 | 3000 |
| C000** | * | * | * | India | 3 | 5000 |
| C000** | * | New York | New York | USA | 2 | 5000 |
| C000** | * | * | * | * | 1 | 10000 |
| C000** | * | New York | New York | USA | 3 | 5000 |
| C000** | * | * | * | * | 2 | 7000 |
| C000** | * | * | * | * | 1 | 7000 |
| C000** | * | * | * | India | 1 | 8000 |
| C000** | * | * | * | India | 1 | 7000 |
| C000** | * | * | * | India | 1 | 7000 |
| C000** | * | * | * | India | 2 | 7000 |

**1.b.**

| **Email ID** | **Smoker or Non-smoker** |
|---|---|
| mihir19061@iiitd.ac.in | Smoker |
| john19062@iiitd.ac.in | Smoker |
| bob20012@iiitd.ac.in | Non-smoker |
| alice20021@iiitd.ac.in | Smoker |
| carol21187@iiitd.ac.in | Smoker |
| dillon21098@iiitd.ac.in | Non-smoker |

- I would opt for **k-anonymity** to anonymously record, analyze and transmit the data.
- **The K-anonymity** scheme ensures that the insensitive attributes of the recorded person, such as personally identifiable information (PII), is not disclosed.
- It severs the link between the PII and sensitive attributes.
- Through k-anonymity, there are at least *k* records in the table that have identical insensitive attributes.
- This is achieved through:
    - Suppression: hiding completely an entry in column
    - Generalization: replacing the attribute with a less specific entry

2-anonymising the above table, we get:

| **Email ID** | **Smoker or Non-smoker** |
|---|---|
| *19*@iiitd.ac.in | Smoker |
| *19*@iiitd.ac.in | Smoker |
| *@iiitd.ac.in | Non-smoker |
| *@iiitd.ac.in | Smoker |
| *@iiitd.ac.in | Smoker |
| *@iiitd.ac.in | Non-smoker |

**2.**

- In e-commerce platforms, compliance according to guidelines for handling customers' financial data, and carrying out transactions in a secured manner is important for the customer and the company's safety.
- Platforms must actively alert and notify customers about what data is being collected and stored by them.
    - This includes the exact reasons why specific data is required
    - Whether the provision of the data is required or optional
    - Steps taken by the platform to securely collect and store the data
- Platforms must allow users to opt-in or opt-out of data collection, if it is possible
- Platforms must allow users to readily and quickly access, change or remove their personal data.
- Platforms, through a proper technical and hierarchical system, limit the accessibility of customer data by other parties.
- Platforms must be able to enforce the above and provide quick remedy. They must be able to readily address aggrieved customers.
- Financial data of customers, such as card number, card name, date of expiry should be stored, if at all, in an encrypted manner. Their CVV must not be stored.
- Per laws in India, all financial data including transaction data must be stored in servers that are physically present in India and under the Indian jurisdiction.
- During the transaction, all information regarding the card, and other input fields, must be channeled through a protocol that supports encryption, such as HTTPS.
- Before any transaction, or modification of payment method, multi-factor authentication must ensure that only the authorized user is making these changes.
- Platforms must keep audit logs that track what modifications each user is making.
- Regular external security audits must take place to ensure the infrastructure is not compromised or vulnerable.

**3.**

a. Assuming encryption scheme to be asymmetric.
   - Bob will not be able to decrypt this message, since it is signed using *Alice*'s public key.
   - Only Alice can decrypt this message using her private key.
   - It is confidential in that the message has not been compromised, even though Bob cannot decrypt it.
   - It is not non-repudiable, since there is no information about the encryptor/signee in the exchanged message.

b. Assuming encryption scheme to be asymmetric.
   - Bob will be able to decrypt this message.
   - This message will be confidential, and only decryptable using Bob's private key.
   - This message will not ensure non-repudiation, because it is not accompanied by a digital signature.
   - Steps to decrypt:
     - Decrypt the received ciphertext using Bob's private key
     - Verify validity of data by comparing message with its hash

c. Assuming encryption scheme to be asymmetric.
   - Bob will be able to decrypt this message.
   - This message will be confidential, and only decryptable using Bob's private key.
   - This message will be non-repudiable, since along with the ciphertext is included a digitally signed hash of the message
   - Steps to decrypt:
     - Decrypt the received ciphertext using Bob's private key
     - Verify signature using Alice's public key
     - Verify validity of data by comparing message with its hash

d. Assuming encryption scheme to be asymmetric.
   - Bob will be able to decrypt this message.
   - This message will be confidential, and only decryptable using Bob's private key.
     - However, we notice that Alice has Bob's private key. This leads us to believe that Bob's private key has been compromised, and so has the confidentiality of the message.
   - Since the hash of the message has been digitally signed by Bob's private key, authorship of the message won't accurately reflect the author. Moreover, since Bob's private key has been leaked, it can be used to sign any message and falsely affirm that Bob is the signee.
     - Even though private key leakage does not mean the signing algorithm is not non-repudiable, it is contestable in legal terms.
   - Steps to decrypt:
     - Decrypt the received ciphertext using Bob's private key

e. Assuming first two encryption schemes to be asymmetric and third to be symmetric.
    - Bob will be able to decrypt this message.
    - This message will be confidential, and decryptable by anyone who knows `sym` or Bob's private key
    - This does not ensure non-repudiation since it is not accompanied with a digital signature.
    - Steps to decrypt:
        - Decrypt `sym` using Bob's private key (asymmetric scheme)
        - Decrypt `m` using `sym` (symmetric scheme)
f. Assuming all encryption schemes to be asymmetric except for the 5th line.
    - Bob will be able to decrypt this message.
    - This message will be confidential, and decryptable by anyone who knows `sym_1` and `sym_2` or Bob's private key
    - This does not ensure non-repudiation since accompanied with a digital signature of the symmetric keys but not the message.
        - There is no *proof of sender* present in the payload of the encrypted message, or sent separately. This Alice will be able to repudiate the authorship of the sent message. Signing symmetric keys does not bring any useful information for Bob to verify authorship, but the fact that *Alice* has access to `sym_1` and `sym_2`.
    - Steps to decrypt:
        - Decrypt `sym_1` using Bob's private key (asymmetric scheme)
        - Decrypt `sym_2` using Bob's private key (asymmetric scheme)
        - Decrypt first ciphertext using `sym_1` (symmetric scheme)
        - Decrypt message using `sym_2` (symmetric scheme)

**4.** We must first start by identifying our targets for the attack. Our main target will be Google Classroom, where the collection of student submissions are stored. Our secondary target will be the TAs and instructor's machines, which might be in possession of the files.

The following attacks can be employed to retrieve information from the first target:

- Phishing
    - This is the most obvious and lucrative method to infiltrate.
    - Since Google Classroom uses the Google login, we can use any Google service (Classroom, Drive, Gmail, Calendar, etc.) and create an authentic-looking clone of the page.
    - After which, we can send a link (a domain resembling that of Google services) to the TAs and instructor
    - If they fall trap, the TA or instructor will enter their valid email address-password combination into our fake Google login and intercept their credentials
    - These credentials will be used by us to login to their Google Classroom and access the motherlode of submissions.
- XSS Attack
    - Usually common with backends that don't properly sanitize their input fields or query parameters
    - It can allow us to inject malicious scripts into the website that will be run by the website (assuming its ownership) and possibly compromise user security.
    - In our case, we can use XSS to inject a script that will send contents or URLs of the submission files from the TAs browser to our external endpoint.

For the secondary target, we must be careful in picking subjects. This method is less fruitful due to the ambiguity present (whether they will have the files locally stored or not), and whether they will fall prey to the social engineering

- Social engineering
    - This is a major way we can access the contents of the subject's laptop.
    - Usually, we must first establish a rapport with them and build their trust in us.
    - After which, we entrap them in entering their credentials in a seemingly safe platform, as vouched by us, the attackers.
    - Through this method, we can either achieve access to their machines, or to their Google account.
- Stealing the laptop while it is unlocked
    - Speaking of hypotheticals, Professor's team is capable enough to sneakily steal the laptop while it is unlocked (say, when the subject has gone for a bathroom break), and extract information.

**5.**

**a.** Five attacks the a password-based authentication scheme must account for are:

1. Dictionary attack: In a dictionary attack, the attacker tries to determine the password by trying all the words in the dictionary.
2. Brute-force attack: In a brute-force attack, the attacker tries to determine the password by trying all the possible passwords.
3. Rainbow attack: In a rainbow attack, the attacker precomputes the hashes of all the possible passwords and then he tries to match the hash of the password provided by the user to all the precomputed hashes of passwords.
4. Phishing attack: Phishing attack is an attack in which the user is tricked into visiting a malicious site that looks exactly like the site the user thought he was visiting.
5. Man-in-the-middle attack: In a man-in-the-middle attack, the attacker tries to read the traffic between two parties (for example, Eve listens to the traffic between Alice and Bob).

**b.** The following should be implemented in an ideal password-based authentication system:

1. The password should not be stored in plaintext in the database. Further, the password should be properly hashed (using a one-way hash function) and salted, to prevent rainbow attacks.
2. Passwords should be sent through the network using a protocol that supports encryption, like HTTPS. This is to prevent man-in-the-middle attacks that would be able to eavesdrop and read the plaintext password.
3. There should not be any character limitations on the password - users should be able to enter any characters of any length, while ensuring that the password remains strong.
4. Even if the database or authentication service is compromised, the implemented system and user data remains secured.
5. Incorrect passwords are quickly identified but delayed responses are sent to prevent timing attacks.
6. Every login session can be quickly invalidated and deleted to remove unauthorized attackers.

**c.**

"probability of 0.10 of having been guessed"
=> number of passwords guessed / total number of possible passwords = 0.1

i. Total number of passwords = $\sum_{i=1}^{8} 127^i$ = T = 68212339274677760

Total number of attempts in G seconds = 10000*G

Therefore, 10000*G/T = 0.1

=> G = 0.1*68212339274677760/10000 seconds = 682123392746.7776 seconds

= (approx) 21,629 years

ii. Total number of passwords = $\sum_{i=1}^{8} 62^i$ = T = 221919451578090

Total number of attempts in G seconds = 10000*G

Therefore, 10000*G/T = 0.1

=> G = 0.1*221919451578090/10000 seconds = 2219194515.7809 seconds

= (approx) 70.37 years

iii. Total number of passwords = $\sum_{i=1}^{8} 10^i$ = T = 111111110

Total number of attempts in G seconds = 10000*G

Therefore, 10000*G/T = 0.1

=> G = 0.1*111111110/10000 seconds = 1111.1111 seconds

= (approx) 18.51 minutes

**d.**

i. This method of biometric authentication is the most secure. In most cases, authentication servers for systems are located in remote data centers, and physically accessing them is practically infeasible if not impossible, and thus it brings little possibility of vulnerability.

However, this method is not used very commonly as services that require authentication are physically far away from the authentication servers. Thus they communicate over a secured network.

ii. This setup proves to be very vulnerable because of the ease with which one can modify the standalone computer, because it is within our reach and very configurable. In this setup, we can intercept the communication between the stand-alone computer and the authentication system, and always make the standalone computer send a “YES”, indicating a successful biometric authentication.

**6.**

**a.**

- An access control list (ACL) is a representation of the access control matrix.
- Each column of the access control matrix represents a file, and each row represents the user.
- The intersection cells between the column and row represent the user's access rights to the file.

- In an ACL, a list of files (the columns) is maintained.
- Each file in the list is associated with a list of tuples: the tuples associate the user with the various access rights they have for that file, if any.
- Tuples are there only for users that have at least one access right type to the file.

- ACL's answer the question "who has what rights to this file?".
- ACLs are widely used in networks which have a lot of incoming traffic, and who's packets need to be quickly checked for correct access by users.
- However, one must search through the ACL entirely until they find appropriate permissions for a user if the users are grouped into Groups, since one user can be part of many groups. This is slow.

**b.**

i. TOP SECRET >= SECRET but {B, C} ⊄ {A, C}, therefore **no read/write access.**
ii. CONFIDENTIAL >= CONFIDENTIAL but {B} ⊄ {C}, therefore **no read/write access.**
iii. SECRET >= CONFIDENTIAL and {C} ⊆ {C}, therefore **only read access.** (no write-down)
iv. TOP SECRET >= CONFIDENTIAL and {A} ⊆ {A, C}, therefore **only read access.** (no write-down)
v. UNCLASSIFIED <= CONFIDENTIAL, therefore **only write access on {B}.** (no read-up)

**c.**

i. With the "first relevant entry" policy, the "Group 1, R" rule will trigger since Alice is in Group 1. This rule will not give her write access to File 1.

ii. With the "any permission in list" policy, the "Group 2, RW" rule can be used to give Alice write access to File 1 since Alice is in Group 2.

**7.**

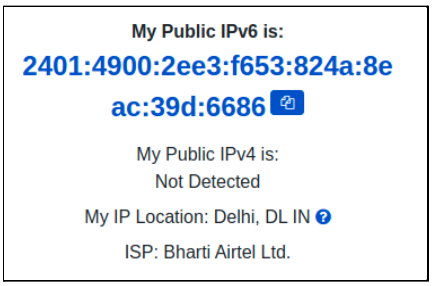
**a.** For each pair of hosts in the network, a different key is computed. Since there are ${}^{n}C_{2}$ pairs possible between n hosts, there are ${}^{n}C_{2} = n(n-1)/2$ different keys to be computed.

**b.** The "nextUpdate" field present in the CRL indicates the date by which the next CRL will be issued. A CRL could be issued before the "nextUpdate" date, but it necessarily cannot be issued *after* that date. It can also be interpreted as the date when the current CRL is rendered obsolete.

**c.**

- Certificate Authorities are human-run organizations that, even though rely heavily on automation and technology, are ultimately fallible to the errors.
- Simple misconfigurations causing downtime are possible, and are not uncommon.
- The most possible error, other than the above, is if a Certificate Authority has their private key leaked and compromised.
- If such an error would occur, bad actors would be able to issue real, but illegitimate, certificates for their malicious websites and give a sense of security to vulnerable users.
- Such errors have been known to happen before with large CAs such as Comodo.
- Thus, we can't always blindly trust Certificate Authorities.

**d.**




- My WiFi hotspot has the following **IPv6** address: 2401:4900:2ee3:f653:824a:8eac:39d:6686
- Advantages of IPv6 include:
    - Since an IPv6 address is 64bit, it allows for potentially $2^{64}$ different addresses, which is far greater than IPv4 and also far greater than what humans will possibly require.
    - Configurationally simple, since at a local level IPv6 does not need to support NATs which do complex address translation and traffic routing, because of the number of addresses it can allocate.
    - IPv6-loaded sites are measurably faster due to more efficient routing across the internet.
- Disadvantages of IPv6 include:
    - Lack of current support by network operators and applications.
    - Absence of NAT is unideal for local businesses who want to keep a common IP address for simpler configurations.
    - The address itself is very long and can bring in confusion.

**8.** I have attended all but two (or maybe three) classes. At present I can’t accurately recall.