Q1 Finding Common Patients

7 Points

This homework has instant feedback. When you click "Save Answer," if the answer is correct, you will see an explanation. You can resubmit as many times as you want.

Relevant sections:
Symmetric-key Encryption (textbook),
Hashing (textbook)

Caltopia has two hospitals: Bear Hospital and Tree Hospital, each with a database of confidential medical records. Each record is a tuple (p_i, m_i), where p_i is a patient's full name and m_i is the patient's medical record. Each Caltopian citizen has a unique name.

Each hospital has a list of records, $(x_1, m_1), ..., (x_n, m_n)$ for Bear Hospital and $(y_1, m_1), ..., (y_n, m_n)$ for Tree Hospital.

Note: the values of m_i may differ between the two hospitals, even for the same patient.

The two hospitals wish to identify patients that attend both hospitals, but are afraid of eavesdroppers like Eve (who has a list of all the plaintext names of the citizens of Caltopia) listening in.

Bear Hospital and Tree Hospital share a key k that is not known to anyone else. Assume r_i is some random bitstring, and $||$ is the bitwise concatenation operation.

Q1.1
1 Point

Tree Hospital suggests applying some cryptographic function to its list of users, transforming it into a list $(y_1^*, y_2^*, ..., y_n^*)$, which it will send to Bear Hospital.

Bear Hospital will then either decrypt y_i^*, or compute x_i^* in the same way and compare (whichever is appropriate).

Which of the following give y_i^* such that Eve cannot win the IND-CPA game? Select all that apply.

- [ ] $y_i^* = \mathrm{SHA}(y_i)$
- [ ] $y_i^* = r_i || \mathrm{SHA}(y_i || r_i)$
- [x] $y_i^* = \mathrm{AES}_k(r_i) || \mathrm{SHA}(y_i || r_i)$
- [x] $y_i^* = \text{AES-CBC}_k(y_i)$
- [x] $y_i^* = \text{AES-CBC}_k(\mathrm{SHA}(y_i))$
- [ ] None of the above

Q1.2
1 Point

For the rest of the question, assume that the lengths of each name are different, and each name is between 1 and 127 bytes long.

Which of the following give y_i^* such that Eve cannot learn anything about the patient names in this new threat model? Select all that apply.

Hint: This question requires stronger security than IND-CPA, because it cannot leak the lengths of names either. This means if a scheme is not IND-CPA secure, it is not secure in this threat model either.

- [ ] $y_i^* = \mathrm{SHA}(y_i)$
- [ ] $y_i^* = r_i || \mathrm{SHA}(y_i || r_i)$
- [x] $y_i^* = \mathrm{AES}_k(r_i) || \mathrm{SHA}(y_i || r_i)$
- [ ] $y_i^* = \text{AES-CBC}_k(y_i)$
- [x] $y_i^* = \text{AES-CBC}_k(\mathrm{SHA}(y_i))$

Q1.3
1 Point

The hospitals soon realize that, due to privacy laws, they cannot share any plaintext information about patients *even with each other* (including their names) unless **both** hospitals know in advance that a patient has in fact used both hospitals.

Bear Hospital will transform its names using $x_i^* = F_k(x_i)$, and Tree Hospital using $y_i^* = F_k(y_i)$, for some function F. A trusted third party S agrees to take the transformed names $x_1^*, \ldots, x_n^*, y_1^*, \ldots, y_n^*$ from both hospitals, and compute a set of pairs:

$$P = \{(i, j) : x_i^* = y_j^*\}$$

We want to ensure three requirements with the above scheme:

1. if $x_i = y_j$, then $(i, j) \in P$
2. if $x_i \neq y_j$, then it is very unlikely that $(i, j) \in P$
3. even if Eve compromises S, she cannot learn the name of any patient at either hospital or the medical information for any patient.
4. Your solution must still provide guarantee #3 even if a new user with an extremely long (and unique) name were to join Caltopia.

Below are potential candidates for a function F. Select all candidates that meet all four requirements:

Hint: To narrow down the options, consider condition 4 to eliminate two of the options.

- [ ] $F_k(p) = \text{AES-ECB}_k(p)$
- [ ] $F_k(p) = \text{AES-CBC}_k(p)$
- [ ] $F_k(p) = \text{SHA}(p)$
- [x] $F_k(p) = \text{SHA}(p||k)$
- [ ] $F_k(p) = \text{SHA}(p)||\text{SHA}(k)$
- [ ] None of the above

Q1.4
1 Point

(Same question as the previous part.) Select all candidates that meet all four requirements.

Hint: To narrow down the options, consider condition 1 to eliminate three of the options.

- [ ] $F_k(p) = \text{AES}_k(r_i)||\text{SHA}(p||r_i)$, r_i is random.
- [x] $F_k(p) = \text{AES-ECB}_k(\text{SHA}(p))$
- [ ] $F_k(p) = \text{AES-CBC}_k(\text{SHA}(p))$
- [x] $F_k(p) = \text{SHA}(\text{AES-ECB}_k(p))$
- [ ] $F_k(p) = \text{SHA}(\text{AES-CBC}_k(p))$
- [ ] None of the above

Q1.5
1 Point

Why does $F_k(p) = \mathrm{SHA}(k||p)$ meet requirement 1?

Recall requirement 1: if $x_i = y_j$, then $(i, j) \in P$

- ○ SHA-256 is one-way
- ○ SHA-256 is collision-resistant
- ◉ SHA-256 is deterministic
- ○ SHA-256 has constant-length output
- ○ An attacker cannot compute SHA-256 without knowing k
- ○ None of the above

Q1.6
1 Point

Why does $F_k(p) = \mathrm{SHA}(k||p)$ meet requirement 2?

Recall requirement 2: if $x_i \neq y_j$, then it is very unlikely that $(i, j) \in P$

- ○ SHA-256 is one-way
- ◉ SHA-256 is collision-resistant
- ○ SHA-256 is deterministic
- ○ SHA-256 has constant-length output
- ○ An attacker cannot compute SHA-256 without knowing k
- ○ None of the above

Q1.7
1 Point

Why does $F_k(p) = \mathrm{SHA}(k||p)$ meet requirement 4?

Recall requirement 4: Eve cannot learn the name of any patient at either hospital or the medical information for any patient, even if a new user with an extremely long (and unique) name were to join Caltopia.

- ○ SHA-256 is one-way
- ○ SHA-256 is collision-resistant
- ○ SHA-256 is deterministic
- ◉ SHA-256 has constant-length output
- ○ An attacker cannot compute SHA-256 without knowing k
- ○ None of the above

Q2 Go tutorial

2 Points

In Project 2, you will be writing a substantial amount of code (300-1000 lines) in Go. This question walks you through some common programming patterns and mistakes in Go.

You might find A Tour of Go helpful for a quick rundown of the basics before trying out this question. You can also use the Go Playground to try out some code snippets in this question.

Q2.1 Data Marshalling
1 Point

Consider the following code snippet (Go Playground link):

```
type BotColor struct {
        ID    int
        name  string
        color string
}

// Create a struct
group := BotColor{
        ID:    1,
        name:  "EvanBot",
        color: "Purple",
}

// Use json.Marshal to compress the struct
// into a byte array
marshalBot, err := json.Marshal(group)
if err != nil {
        fmt.Println("error:", err)
}

// use json.Unmarshal to decompress the struct
var unmarshalBot BotColor
err = json.Unmarshal(marshalBot, &unmarshalBot)
if err != nil {
        fmt.Println("error:", err)
}
fmt.Printf("%+v", unmarshalBot)
```

Assuming `err` is always `nil`, which of the following is the output of this snippet?

- ○ `{ID:1 name:EvanBot color:Purple}`
- ○ `{ID:1 name:EvanBot color:}`
- ◉ `{ID:1 name: color:}`
- ○ `{ID: name: color:}`

Q2.2 UUID
1 Point

Consider the following code snippet (Go Playground link):

```
// Create a UUID from bytes
a := []byte("A stack of pancakes")
aUUID, err := uuid.FromBytes(a[:16])
if err != nil {
        fmt.Println("error:", err)
}
fmt.Println(aUUID)

// Want to see me do it again?
b := []byte("A stack of pancakes")
bUUID, err := uuid.FromBytes(b[:16])
if err != nil {
        fmt.Println("error:", err)
}
fmt.Println(bUUID)
```

Assuming `err` is always `nil`, what should you see printed?

- (●) Always the same UUID twice
- ( ) Always two different UUIDs
- ( ) Either the same UUID twice or two different UUIDs (different every run)

Q3 Project 2 Warm-Up
7 Points

As of 07/11 at 1:00 AM, the project two spec is not out. We will update this once it is!

In Project 2, you'll be implementing a secure file storage system on an insecure data storage service using the cryptographic schemes we've seen in the cryptography unit.

To get started, take a look at the Project 2 spec. We understand that this spec is longer and denser than the project specs you might be used to. As we've mentioned in class, cryptography is often very fragile with a lot of edge cases, so we've cover as many of these edge cases as possible to help make the design process easier for you.

To guide you through the spec, here are some short questions to check your understanding.

Q3.1 Threat Model
1 Point

Relevant section: 2. Threat Model

How many types of adversaries exist in our threat model?

- ○ 0
- ○ 1
- ◉ 2
- ○ 3

Q3.2 Design Requirements
1 Point

Relevant section: 3. Design Requirements

In the next few subparts, consider the given design and choose whether it's valid or invalid according to the design requirements.

When a user creates a file `myFile` with contents `Hello world`, create a variable called `myFile` with the value `Hello world`. When the user wants to retrieve `myFile`, return the value of the `myFile` variable.

- ○ Valid design
- ◉ Invalid design

Q3.3
1 Point

When a user creates a file `myFile` with contents `Hello world`, create an entry in Keystore with key `myFile` and value `Hello world`.

- ○ Valid design
- ◉ Invalid design

Q3.4
1 Point

Files are stored on Datastore. When a user wants to append to a file, download the entire file from Datastore using `DatastoreGet`, add the appended data, and then upload the changed file to Datastore using `DatastoreSet`.

- ○ Valid design
- ◉ Invalid design

Q3.5 Sharing, Receiving, and Revoking
1 Point

Relevant section: 3.6 Sharing and Revoking

In the next few subparts, consider the following sharing sequence and choose whether it's defined or undefined according to the design requirements.

Remember, undefined test cases are not tested by our autograder: your code can implement any behavior in undefined scenarios.

Alice creates a file. Alice shares the file with Bob. Alice shares the file with Charlie. Charlie shares the file with Bob.

- ○ Defined
- ◉ Undefined

Q3.6
1 Point

Alice creates a file. Alice shares the file with Bob. Bob shares the file with Charlie. Alice revokes the file from Charlie.

- ○ Defined
- ◉ Undefined

Q3.7
1 Point

Alice calls `StoreFile("Alice's filename", "my file contents")`.

Alice calls `CreateInvitation("Alice's filename", "Bob")`, which returns a UUID `invitationPtr`. Alice gives `invitationPtr` to Bob.

Bob calls
`AcceptInvitation("Alice", invitationPtr, "Bob's filename")`.

What is returned if Bob calls `LoadFile("Bob's filename")`?

- ○ Error
- ◉ `"my file contents"`

Q4 Feedback
0 Points

What's something we could do to make the class better? How was the midterm? What would you like to see in the future?

Homework 4

● Graded

Select each question to review feedback and grading details.

Student
Yiyun Chen

Total Points

16 / 16 pts

Question 1

Finding Common Patients		**7** / 7 pts
1.1	(no title)	**1** / 1 pt
1.2	(no title)	**1** / 1 pt
1.3	(no title)	**1** / 1 pt
1.4	(no title)	**1** / 1 pt
1.5	(no title)	**1** / 1 pt
1.6	(no title)	**1** / 1 pt
1.7	(no title)	**1** / 1 pt

Question 2

Go tutorial		**2** / 2 pts
2.1	Data Marshalling	**1** / 1 pt
2.2	UUID	**1** / 1 pt

Question 3

Project 2 Warm-Up		**7** / 7 pts
3.1	Threat Model	**1** / 1 pt
3.2	Design Requirements	**1** / 1 pt
3.3	(no title)	**1** / 1 pt
3.4	(no title)	**1** / 1 pt
3.5	Sharing, Receiving, and Revoking	**1** / 1 pt
3.6	(no title)	**1** / 1 pt
3.7	(no title)	**1** / 1 pt

Question 4

Feedback		**0** / 0 pts