System Security Assingment 2
Name: Jeslyn Ho Ka Yan
Sim id: 10241485

Question 1

---

a) How many cases each hash is needed for each Puzzle

Puzzle A

has one sub-puzzle with k=6

Worst expected hashes needed $= m \times 2^k = 1 \times 2^6 =$ 64 hashes

Puzzle B

has six sub-puzzles with k=4

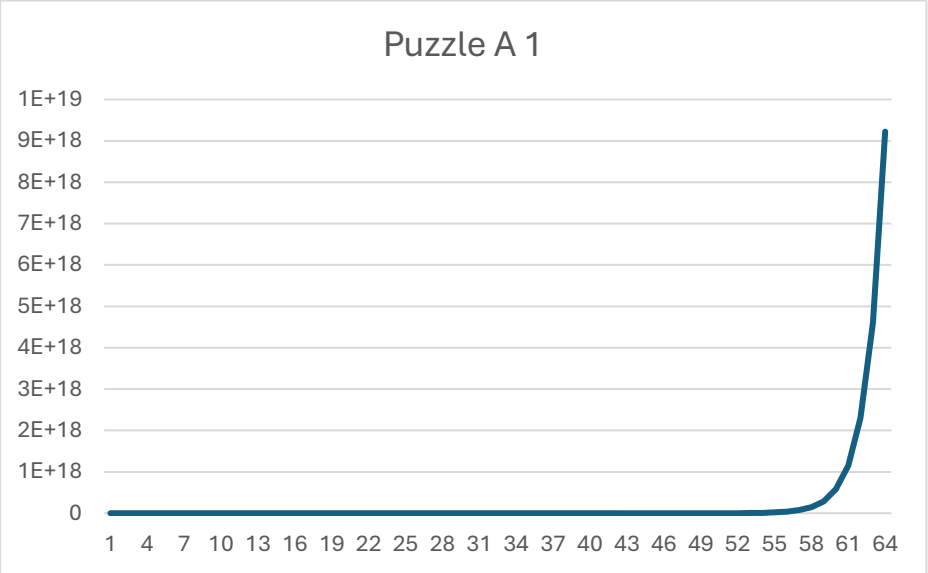
Worst expected hash per sub-puzzle $= 1 \times 2^4 = 16$

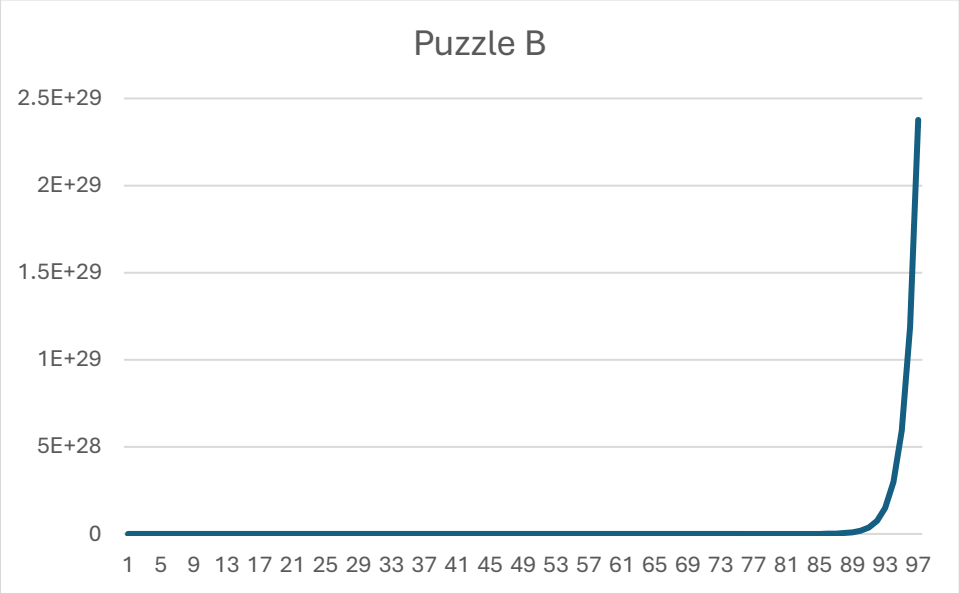
Worst expected hash for six sub-puzzle $= 16 \times 6 =$ 96 hashes

b) Describe how you acquired your distributes.

- I wrote a program to generate random hash values based on the specified parameters for each puzzle.
- For Puzzle A, I created a single distribution with six hash counts, and for Puzzle B, I generated frequencies across six sub-puzzles with four hash counts each.
- After gathering the data, I used Python's Matplotlib library to visualize the distributions, producing bar graphs that effectively represent the frequency of hashes.

c) A graph of the distribution
refer to the excel sheet





d) The average number of hashes needed

Average Number of Hashes Formular = $\frac{\Sigma_1^n n}{n}$, Where $(1+2+3+4+\cdots.+n = \frac{n(n+1)}{2})$

Puzzle A (n = 64)

Average Number of Hashes = $\frac{\Sigma_{n=1}^{64} n}{64} = \frac{\frac{64(64+1)}{2}}{64} = \frac{2080}{64} = 32.5\ hashes$

Puzzle B (n = 16)

Average Number of Hashes per sub-puzzle = $\frac{\Sigma_{n=1}^{16} n}{16} = \frac{\frac{16(16+1)}{2}}{16} = \frac{136}{16} = 8.5\ hashes$

Given that Puzzle B consists of two six sub-puzzles, the average number of hashes = $8.5 \times 6 = 51\ hashes$

e) The standard deviation for the distribution

Puzzle A (n = 64)

The variance $= \frac{(32.5-1)^2+(32.5-2)^2+(32.5-3)^2+\cdots+(32.5-64)^2}{64}$

$= 341.25$

The Standard Deviation $= \sqrt{341.25} = 18.47285 \approx 18.47$

Puzzle B (n = 64)

The variance $= \frac{(8.5-1)^2+(8.5-2)^2+(8.5-3)^2+\cdots+(8.5-16)^2}{16}$

$= 18.917$

Since there are six sub-puzzle, the total varaince is

$= 18.917 \times 6 = 113.502$

The Standard Deviation $= \sqrt{113.502} = 10.6537 \approx 10.65$

Question 2

---

1. **Attack Rate Requirement**

   Attack Rate = $\frac{512\ requests}{3\ min} \approx$ 170.67 requests per minute

2. **Bandwidth Consumption:**

   2.1 the size in bits of each SYN packet

   1 bytes = 8

   64 bytes $\times$ 8 = 512 bits

   2.2 the rate request per second:

   $\frac{170.67}{60} \approx 2.8445\ requests\ per\ second$

   2.3 Lastly, the bandwidth in bits per second

   2.8445 $\times$ 512 bits = 1456.38 bits per second

Question 3

---

a)
Query 1: Get the total salary for all female Computing Lecturers.

```
SELECT Salary
FROM table
WHERE Gender = 'Female' AND School = 'Computing' AND Position =
'Lecturer';
```

Query 2: Get the total salary for all female Computing Lecturers except Iris (which isolates Debbie's salary).

```
SELECT Salary
FROM table
WHERE Name = 'Debbie';
```

b)
The First Select statement: Finds the total number of salary of all female lectures.
Salaries of Carol, Debbie, Fiona, and Karen.

The Second Select statement: Find the total salary of female lecturers excluding Debbie.
Salaries of Carol, Fiona, and Karen.

```
SELECT SUM(Salary)
FROM table
WHERE Gender = 'Female'
AND Position = 'Lecturer'
MINUS
SELECT SUM(Salary)
FROM table
WHERE Gender = 'Female'
AND Position = 'Lecturer'
AND Name != 'Debbie';
```

Result: Debbie's salary = (Query 1 result) - (Query 2 result).

Question 4

(a) Comparison of Protection Methods: SYN Cookies and Client Puzzles

SYN Cookies: This technique is especially utilised for TCP SYN flood assaults, in which the attacker sends a lot of SYN requests without finishing the TCP handshake. By encapsulating the initial sequence number with extra connection-related information, SYN cookies prevent these SYN queries from using up server resources as they normally would. The client must correctly reply with an ACK before the server can allocate resources. Only authentic clients can connect if the server reconstructs the session data after receiving a valid ACK.

Client Puzzles: With this method, a client must first solve a computational challenge in order to create a connection. After the client finishes a time-consuming challenge that the server presents, the server confirms the answer before continuing. Clients' computational workload is increased as a result, particularly for attackers who try to overload the server with requests. Client puzzles can be used with protocols other than TCP, making them more versatile than SYN cookies.

Similarities:

- By limiting server resources to valid connections, both strategies seek to stop DoS attacks.
- They wait for the client to authenticate the request (either by answering a puzzle or providing an ACK) before allocating resources.

Differences:

- SYN cookies are designed to primarily target TCP SYN flood attacks, whereas client puzzles are more general and applicable to several protocols.
- SYN cookies focus on deferring connection state allocation, while client puzzles add a computational cost to the client, requiring more processing power from each requester.

(b) Properties of Client Puzzles:

1. **Scalability**: Client puzzles have the ability to dynamically change their level of complexity according to perceived danger or server load. To make it more difficult for attackers to overrun the server during a suspected DoS attempt, for example, the server can make the problem more complex. This scalable complexity requires more computing resources from clients in high-risk scenarios, which helps balance server performance.

   Example: Overloaded servers may make puzzles more difficult to solve, which would slow down potential attackers' completion of each one and lower the volume of fraudulent requests they receive.

2. **Asymmetric Computational Cost**: Client puzzle designs make sure that while they are computationally expensive for clients to solve, they are simple for the server to create and validate. Because attackers typically have limited resources, this asymmetry makes it harder for them to answer a lot of riddles, which limits their capacity to flood the server.

   Example: A server can create and validate puzzles very rapidly, but an attacker must use a lot more computing power to solve each one, which makes it impossible for them to send a lot of requests in a short period of time.

3. **Transparency**: Puzzles can usually be solved quickly by legitimate users, so regular services can continue with little effect on real users. On the other hand, malevolent actors encounter a greater delay because it can take a lot of resources, particularly for bots, to solve several riddles rapidly.

   Example: The problem might cause a small delay that real users connecting to a website might not notice, but a bot processing hundreds of queries would be greatly slowed down.

4. **Protocol and Application Flexibility**: Client puzzles are suitable for a variety of applications and scenarios since they may be deployed at several layers and are flexible across multiple protocols, including TCP and HTTP.

   Example: Client puzzles in TCP, which require clients to complete a challenge before completing a handshake, can help prevent SYN flood assaults. Bots trying to flood the application layer can be slowed down by using HTTP to issue puzzles for every request.

5. **Flexibility**: Client puzzle can be modified to accommodate various attack methods, application layers, and protocols. They can be used in a wide range of settings and attack situations because of their adaptability, regardless of the particular application or protocol being used. Administrators can modify the deployment of puzzles to customise protection to the requirements of the system.

   Example: Network Layer (TCP Protocol), Client puzzles can be used at the network layer to mitigate SYN flood attacks. For example, before completing a TCP handshake, the server can issue a puzzle that the client must solve. This prevents attackers from easily establishing multiple half-open connections, a common tactic in SYN floods.

Question 5

(a)

Hour t	Infected Computers N(t)
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1,024
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152
22	4,194,304
23	8,388,608
24	16,777,216

Explanation of Process

The spread pattern follows exponential growth where the number of infected computers doubles every hour. By using the formula $N(t)=2^t$, From time t=0 to time t=24, calculate the number of compromised machines every hour.

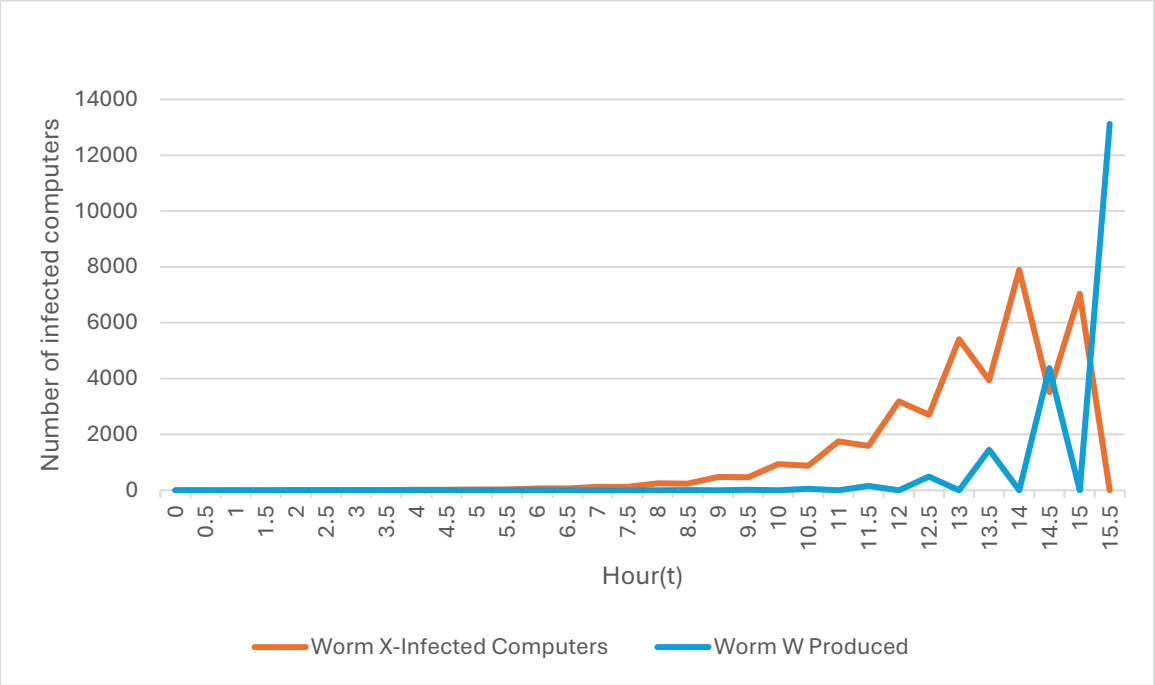
(b)

For Worm X, infected computers double each hour: $N_X(t)=2^t$
For Worm W, each infected computer clears two Worm X infections per hour.

Time (t)	**Worm X-Infected Computers**	**Counter-Worm W**	**Worm W Produced**
0	8	0	0
0.5	8	0	0
1	16	0	0
1.5	16	0	0
2	32	0	0
2.5	32	0	0
3	64	0	0
3.5	63	0	0
4	126	0	0
4.5	124	0	0
5	248	0	0
5.5	242	0	0
6	484	0	0
6.5	466	1	1
7	932	1	0
7.5	878	3	2
8	1756	3	0
8.5	1594	9	6
9	3188	9	0
9.5	2702	18	18
10	5404	18	0
10.5	3946	72	54
11	7892	72	0
11.5	3518	234	162
12	7036	234	0
12.5	0	486	486
13	8	486	0
13.5	8	1944	1458
14	16	1944	0
14.5	16	6318	4374
15	32	6318	0
15.5	32	13122	13122

(c)

1. **Worm X Spread**: Worm X doubles each hour, leading to exponential growth.
2. **Counter-Worm W Impact**: At t=6.5, Worm W starts removing two Worm X infections per infected W computer each hour. This slows Worm X's growth, creating an alternating pattern of increases (on the hour) and decreases (on the half-hour).
3. **Overall Pattern**: Worm W gradually slows down Worm X's spread, resulting in oscillations in Worm X's infection count. Over time, Worm W could potentially neutralize Worm X.



(d)

1. **New Spread Rate for Worm X**: At t=9, Worm X evolves to spread to three computers each hour (tripling instead of doubling).
2. **Outcome**: With this tripling rate, Worm X outpaces Worm W's removal capacity, leading to rapid, uncontrolled growth of Worm X infections.

Question 6

---

1. Data Collection

- **Passenger Data**: Anonymous fare card identifiers, entry/exit times, and transfer patterns for travel insights.
- **Operational Data**: Real-time vehicle locations, driver/operator IDs, and occupancy levels to monitor service quality.
- **Security Data**: CCTV footage and system access logs to enhance safety.

2. Collection Points

- **Ticketing**: At entry/exit gates and bus tap-ins/tap-outs.
- **Vehicle Tracking**: Throughout journeys with GPS.
- **Occupancy and Security**: Stationary and onboard sensors for crowd monitoring and safety.

3. Data Format and Storage

- **Format**: JSON/XML for structured logs; compressed video for surveillance.
- **Storage**: Secure cloud storage with encryption, partitioning by time, and scalable databases (SQL/NoSQL).

4. Analysis Goals

- **Operational Efficiency**: Identify delays, route bottlenecks, and optimize schedules.
- **Passenger Experience**: Detect overcrowding, long wait times, and improve transfers.
- **Security**: Spot unauthorized access and monitor high-incident areas.

5. Problem Detection

- **Anomaly Detection**: Machine learning to flag delays, crowd spikes, and suspicious access.
- **Time Series Analysis**: Track patterns to adjust resources during peak times.
- **Correlational Analysis**: Link delays or overcrowding with external factors (e.g., holidays).