Assigmment

Question 1

> *a. What is the difference between symmetric cryptography and asymmetric cryptography?*

The main difference between symmetric cryptography and asymmetric cryptography is that symmetric cryptography uses the same key in both encryption and decryption, while asymmetric cryptography uses different keys.

> *b. Given that both types of cryptography can protect security, why should we still need both of them?*

The advantage of asymmetric cryptography is its safety, but the cost is the effciency of encryption and decryption since it uses different keys.

Symmetric cryptography has better performance on encryption/decryption speed, but it is not as safe as asymmetric cryptography because it uses the same key.

In practice, we'd like to use symmetric cryptography to encrypt/decrypt the true message, especially for messages of large size, and use asymmetric cryptography to encrypt/decrypt the key used in the symmetric cryptography.

> *c. What is the algorithm framework of RSA?*

RSA is an asymmetric cryptography algorithm involving 4 steps: key generation, key distribution, encryption, and decryption.

Key generation

- Choose two distinct prime numbers p and q
- Compute $n = pq$
- Compute $\lambda(n) = lcm(\varphi(p), \varphi(q)) = lcm(p-1, q-1)$
- Choose an integer e such that $1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1$, that is, e and $\lambda(n)$ are coprime
- Determine d as $d \equiv e^{-1} \mod \lambda(n)$

Key distrubution

Suppose that Bob wants to send messages to Alice and they decide to use RSA, then Bob must know Alice's public key for encrypting and Alice must use her private key for decrypting.

Encryption

After Bob obtains Alice's public key, he can send a message M to Alice.

Bob should first turn M into an integer m, such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a padding scheme, and then computes the ciphertext c using Alice's public key e, corresponding to

$$m^e \equiv c \mod n$$

Bob then transmits c to Alice.

Decryption

Alice can recover m from c by using her private key exponent d by computing

$$c^d \equiv (m^e)^d \equiv m \mod n$$

Given M, she can recover the original message M by reversing the padding scheme.

Question 2

> *a. Given an n-bit password, what is the average trying time for cracking the password using a brute force attack? Provide the detailed derivation.*

There are 2^n possibilities which the true password can be. As the a brute force attack is simply trying all possibilities one by one, the average trying time is $O(\frac{2^n}{2}),$ that is, $O(2^n)$.

More adequately, suppose each single attempt costs a time unit, the mean cost is

$$\begin{aligned}
&1 \times \frac{1}{2^n} + 2 \times \frac{2^n - 1}{2^n} \times \frac{1}{2^n - 1} + \cdots + 2^n \times \prod_{i=1}^{2^n - 1} \frac{2^n - i}{2^n - i + 1} \\
=&\frac{1}{2^n} + \frac{2}{2^n} + \cdots + \frac{2^n}{2^n} \\
=&\frac{1}{2^n} \sum_{i=1}^{2^n} i = \frac{1}{2^n} \cdot \frac{(1 + 2^n)2^n}{2} \\
=&2^{n-1} + \frac{1}{2}
\end{aligned}$$

> *b. How does a replay attack work? How to address it?*

Replay attack is a form of network attack in which valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and re-retransmits it, possibly as part of a sproofing attack by IP packet substitution. This is one of the lower-tier versions of man-in-middle attack. Usually, replay attacks are passive in nature.

Suppose Alice wants to prove her identity to Bob. Bob requests her password as a proof, which Alice dutifully provides (possibly after some transformation like hashing or salting). Meanwhile, Eve is eavesdropping on the conversation and keeps the password. After the interchange is over, Eve (acting as Alice) connects to Bob and sends Alice's password when asked for a proof of identity.

There are many methods to prevent replay attacks, like using more on-session identifiers, using one-time passwords, using timestamps, or using nonces including MAC.

> *c. How does a man-in-the-middle attack work? How to address it?*

MITM (man-in-the-middle) attack is a cyberattack where the attacker secretly relays and possibly alter the communications between two parties who believe that they are directly communicating with each other.

One example is active eavesdropping, in which the attacker makes independent connections with the victims and relays messages between them to make them believe that they are talking directly to each other over a private connection, when in fact entire conversation is controlled

by the attacker. The attacker must be able to intercept all relevant messages passing between the two victims and injected new ones.

MITM attacks can be prevented or detected by two means: authentication and tamper detection. All cryptographic systems that are secure against MITM attacks provide some method of authentication for messages. Latency examination can potentially detect the attack in certain situations, such as with calculations that lead into tens of seconds like hash functions.

> *d. How does a relay attack work in wireless communication? How does distance bounding work against a relay attack?*

Relay attack is a type of hacking technique related to man-in-the-middle and replay attacks.

In a classic relay attack, communication with both parties is initiated by the attacker who then merely relays messages between the two parties without manipulating them or even necessarily reading them.

Suppose that Peggy works in a high security building that she accesses using a smart card in her purse. When she approaches the door of the building, the building detects the presence of a smart card and initiates an exchange of messages that constitute a zero-knowledge password proof that the card is Peggy's. The building then allows Peggy to enter.

Now Mallory wants to break into the building.

(1)Mallory approaches the building with a device that simulates a smart card, and the building responds by initiating the exchange of messages. (2)Mallory forwards the message to her accomplice Evelyn who is tailing Peggy as she runs errands in another part of town. (3)Evelyn relays the message to Peggy's smart card, listens for the answer, and forwards the answer to Mallory, who relays it to the building. (4)Continuing in this way, Mallory and Evelyn relay messages between the building and Peggy's smart card until the building is satisfied that it is communicating with Peggy's smart card. (5)The building opens and Mallory enters.

Distance bounding could prevent the risk of relay attacks on contactless cards, by measuring how long a card takes to respond to a request from a terminal for identifification. As long as the estimation is accurate enough, relay attacks can be detected.

Question 3

> *a. What are the key features of the five typical delivery schemes?*

Unicast: deliver msg to a single node

Broadcast: deliver msg to all nodes in network

Multicast: deliver msg to a group of nodes

Anycast: deliver msg to any one of a group

Geocast: deliver a message to a group of nodes based on geographic location

> *b. What is the framework of the Dijkstra algorithm?*

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph. For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the *unvisited set*.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbours and calculate their *tentative* distances through the current node. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one. For example, if the current node *A* is marked with a distance of 6, and the edge connecting it with a neighbour *B* has length 2, then the distance to *B* through *A* will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, the current value will be kept.

4. When we are done considering all of the unvisited neighbours of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

Pseudocode:

```
1  function Dijkstra(Graph, source):
2      create vertex set Q
3      for each vertex v in Graph:
4          dist[v] ← INFINITY
5          prev[v] ← UNDEFINED
6          add v to Q
7          dist[source] ← 0
8          while Q is not empty:
9              u ← vertex in Q with min dist[u]
10             remove u from Q
11             for each neighbor v of u:
12             // only v that are still in Q
13                 alt ← dist[u] + length(u, v)
14                 if alt < dist[v]:
15                     dist[v] ← alt
16                     prev[v] ← u
```

```
17          return dist[], prev[]
```

> *c. What is the framework of the Bellman-Ford algorithm?*

The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph. It is slower than Dijkstra's algorithm for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers.

Like Dijkstra's algorithm, Bellman–Ford proceeds by relaxation, in which approximations to the correct distance are replaced by better ones until they eventually reach the solution.

In both algorithms, the approximate distance to each vertex is always an overestimate of the true distance, and is replaced by the minimum of its old value and the length of a newly found path.

However, Dijkstra's algorithm uses a priority queue to greedily select the closest vertex that has not yet been processed, and performs this relaxation process on all of its outgoing edges; by contrast, the Bellman–Ford algorithm simply relaxes all the edges, and does this $|V| - 1$ times, where $|V|$ is the number of vertices in the graph.

In each of these repetitions, the number of vertices with correctly calculated distances grows, from which it follows that eventually all vertices will have their correct distances. This method allows the Bellman–Ford algorithm to be applied to a wider class of inputs than Dijkstra.

Simply put, the algorithm (1)initializes the distance to the source to 0 and all other nodes to infinity. (2)Then for all edges, if the distance to the destination can be shortened by taking the edge, the distance is updated to the new lower value. (3)At each iteration i that the edges are scanned, the algorithm finds all shortest paths of at most length i edges (and possibly some paths longer than i edges). Since the longest possible path without a cycle can be $|V| - 1$ edges, the edges must be scanned $|V| - 1$ times to ensure the shortest path has been found for all nodes. (4)A final scan of all the edges is performed and if any distance is updated, then a path of length $|V|$ edges has been found which can only occur if at least one negative cycle exists in the graph.

Pseudocode:

```
1  function BellmanFord(list vertices, list edges,
   vertex source) is
2      // This implementation takes in a graph,
   represented as
3      // lists of vertices (represented as integers
   [0..n-1]) and edges,
4      // and fills two arrays (distance and
   predecessor) holding
5      // the shortest path from the source to each
   vertex
6
7      distance := list of size n
8      predecessor := list of size n
9
10     // Step 1: initialize graph
11     for each vertex v in vertices do
12         distance[v] := inf
```

```
13      // Initialize the distance to all vertices to
    infinity
14          predecessor[v] := null
15      // And having a null predecessor
16
17      distance[source] := 0       // The distance from
    the source to itself is, of course, zero
18
19      // Step 2: relax edges repeatedly
20      repeat |V|-1 times:
21          for each edge (u, v) with weight w in edges
    do
22              if distance[u] + w < distance[v] then
23                  distance[v] := distance[u] + w
24                  predecessor[v] := u
25
26      // Step 3: check for negative-weight cycles
27      for each edge (u, v) with weight w in edges do
28          if distance[u] + w < distance[v] then
29              error "Graph contains a negative-weight
    cycle"
30
31      return distance, predecessor
```

> *d. How does prefix hijacking work?*

distance-vector: announce 0 distance to all other nodes

link-state: drop links; claim direct link to other routers

BGP:announce arbitrary prefix; alter paths

> *e. How does RPKI work? Why is it insufficient for secure routing?*

RPKI provides a way to connect Internet number resource information (such as Autonomous System numbers and IP addresses) to a trust anchor. The certificate structure mirrors the way in which Internet number resources are distributed. That is, resources are initially distributed by the IANA to the regional Internet registries (RIRs), who in turn distribute them to local Internet registries (LIRs), who then distribute the resources to their customers. RPKI can be used by the legitimate holders of the resources to control the operation of Internet routing protocols to prevent route hijacking and other attacks. In particular, RPKI is used to secure the Border Gateway Protocol (BGP) through BGP Route Origin Validation (ROV), as well as Neighbor Discovery Protocol (ND) for IPv6 through the Secure Neighbor Discovery protocol (SEND).

1. RPKI verify the code number resource allocation relationship by issuing RPKI resource certificate. When a CA assigns a portion of its IP address resource / AS number to a lower authority, it issues a resource certifificate to that authority to confifirm the assignment. The content of the certifificate is the binding relationship between the IP address prefifix / AS number and the receiving institution.
2. Authorize an autonomous network to issue a routing origin notification to an IP address prefix by issuing a ROA signature. ROA binds the AS number of the autonomous network to the IP prefix.
3. The above resource certificates and ROA signatures are stored and published in the publish points maintained by each CA, and

the distributed database composed of all these publish points is the RPKI database.

4. The RPKI dependent party (RP) is responsible for periodically downloading these certifificates and signatures synchronously from the RPKI database and verifying their validityto obtain the true authorization relationship between the IP prefifix and the AS number. The router obtains this data from the RPKI dependent party to determine the authenticity of the BGP routing message, that is, whether the originating AS in the routing message has the legal authorization to notify the IP prefifix.

For why is RPKI insufficient for secure routing, it is beacause Malicious routers can pretend to connect to the valid origin.

Question 4

> *a. What is the difference between DoS attacks and DDoS attacks?*

A DoS attack is a denial of service attack where a computer is used to flood a server with TCP and UDP packets. A DDoS attack is where multiple systems target a single system with a DoS attack. The targeted network is then bombarded with packets from multiple locations.

The key difference between DoS and DDoS attacks is that the latter uses multiple internet connections to put the victim’s computer network offline whereas the former uses a single connection. DDoS attacks are more difficult to detect because they are launched from multiple

locations so that the victim can't tell the origin of the attack. Another key difference is the volume of attack leveraged, as DDoS attacks allow the attacker to send massive volumes of traffic to the target network.

> *b. How does the TCP SYN Flood attack work?*

A SYN flood is a form of denial-of-service attack in which an attacker rapidly initiates a connection to a server without finalizing the connection. The server has to spend resources waiting for half-opened connections, which can consume enough resources to make the system unresponsive to legitimate traffic. The packet that the attacker sends is the SYN packet, a part of TCP's three-way handshake used to establish a connection.

A SYN flood attack works by not responding to the server with the expected `ACK` code. The malicious client can either simply not send the expected `ACK`, or by spoofing the source IP address in the `SYN`, cause the server to send the `SYN-ACK` to a falsified IP address – which will not send an `ACK` because it "knows" that it never sent a `SYN`.

The server will wait for the acknowledgement for some time, as simple network congestion could also be the cause of the missing `ACK`. However, in an attack, the half-open connections created by the malicious client bind resources on the server and may eventually exceed the resources available on the server. At that point, the server cannot connect to any clients, whether legitimate or otherwise. This effectively denies service to legitimate clients. Some systems may also malfunction or crash when other operating system functions are starved of resources in this way.

> *c. How does the solution of SYN Cookies against TCP SYN Flood attacks work?*

The use of SYN cookies allows a server to avoid dropping connections when the SYN queue fills up. Instead of storing additional connections, the SYN queue entry is encoded into the sequence number sent in the SYN+ACK response. If the server then receives a subsequent ACK response from the client with the incremented sequence number, the server is able to reconstruct the SYN queue entry using information encoded in the TCP sequence number and proceed as usual with the connection.

In order to initiate a TCP connection, the client sends a SYN packet to the server. In response, the server sends a SYN+ACK packet back to the client. One of the values in this packet is a sequence number, which is used by the TCP to reassemble the data stream. According to the TCP specification, that first sequence number sent by an endpoint can be any value as decided by that endpoint. As the sequence number is chosen by the sender, returned by the recipient, and has no otherwise-defined internal structure, it can be overloaded to carry additional data. The following describes one possible implementation, however as there is no public standard to follow, the order, length, and semantics of the fields may differ between SYN cookie implementations.

SYN cookies are initial sequence numbers that are carefully constructed according to the following rules:

- let t be a slowly incrementing timestamp (typically `time()` logically right-shifted 6 positions, which gives a resolution of 64 seconds)
- let m be the maximum segment size (MSS) value that the server would have stored in the SYN queue entry
- let s be the result of a cryptographic hash function computed over the server IP address and port number, the client IP address

and port number, and the value t. The returned value s must be a 24-bit value.

The initial TCP sequence number, i.e. the *SYN cookie*, is computed as follows:

- Top 5 bits: $t \mod 32$
- Middle 3 bits: an encoded value representing m
- Bottom 24 bits: s

When a client sends back a ACK packet to the server in response to the server's SYN+ACK packet, the client must (according to the TCP spec) use $n + 1$ in the packet's Acknowledgement number, where n is the initial sequence number sent by the server. The server then subtracts 1 from the acknowledgement number to reveal the SYN cookie sent to the client.

The server then performs the following operations.

- Checks the value t against the current time to see if the connection has expired.
- Recomputes s to determine whether this is, indeed, a valid SYN cookie.
- Decodes the value m from the 3-bit encoding in the SYN cookie, which it then can use to reconstruct the SYN queue entry.

From this point forward, the connection proceeds as normal.

By specifically calculating the TCP sequence number with a specific, secret math function in the SYN-ACK response, the server does not need to maintain this state table. On receipt of the ACK from the Client, the TCP sequence number is checked against the function to determine if

this is a legitimate reply. If the check is successful, then the server will create the TCP session and the user connection will proceed as normal.

> *c. How does the DNS Amplification Attack work? How to defend against it?*

A DNS amplification attack is a popular form of distributed denial of service (DDoS) that relies on the use of publically accessible open DNS servers to overwhelm a victim system with DNS response traffic.

The primary technique consists of an attacker sending a DNS name lookup request to an open DNS server with the source address spoofed to be the target's address. When the DNS server sends the DNS record response, it is sent instead to the target. Attackers will typically submit a request for as much zone information as possible to maximize the amplification effect. In most attacks of this type observed by US-CERT, the spoofed queries sent by the attacker are of the type, "ANY," which returns all known information about a DNS zone in a single request. Because the size of the response is considerably larger than the request, the attacker is able to increase the amount of traffic directed at the victim. By leveraging a botnet to produce a large number of spoofed DNS queries, an attacker can create an immense amount of traffic with little effort. Additionally, because the responses are legitimate data coming from valid servers, it is extremely difficult to prevent these types of attacks.

While the attacks are difficult to stop, network operators can apply several possible mitigation strategies.

How to defend against:

- reduce the number of NTP servers that support monlist
- source IP verification – stop spoofed packets leaving network

Question 5

> *a. What are the key cryptographic techniques used in blockchain? What are they used for therein?*

Digital signature

- Verify identifies when a transaction occurs

Hash

- Proof of work
- Calculate block headers

> *b. How is double spending addressed in blockchain?*

To prove that no attempts to double-spend have occurred, the blockchain provides a way for all nodes to be aware of every transaction. With bitcoin, all transactions are publically announced to all nodes. They can then agree on a single history of the order in which they were received.

If the majority of the nodes agree on which transaction was first to be received, later attempts to double-spend are irrelevant.

Satoshi Nakamoto's white paper proposed the use of a timestamp server as a solution to the double-spending problem. This server takes a hash of a block of transactions and then broadcasts this hash to all the nodes in the bitcoin network. This timestamp proves that all the data in the hash couldn't have been created *after* the hash was published (obviously).

As each timestamp includes the previous timestamp in its hash, this forms an immutable (unchangeable) record of the order in which transactions took place. Each timestamp reinforces the ones before it.

> *c. How does Proof of Stake work and save blockchain from intensive computation?*

Proof of Stake (PoS) is a type of algorithm which aims to achieve distributed consensus in a Blockchain.

Nodes on a network stake an amount of cryptocurrency to become candidates to validate the new block and earn the fee from it. Then, an algorithm chooses from the pool of candidates the node which will validate the new block. This selection algorithm combines the quantity of stake (amount of cryptocurrency) with other factors (like coin-age based selection, randomization process) to make the selection fair to everyone on the network.

A typical PoS based mechanism workflow:

1. Nodes make transactions. The PoS algorithm puts all these transactions in a pool.
2. All the nodes contending to become validator for the next block raise a stake. This stake is combined with other factors like "coin-age" or "randomized block selection" to select the validator.

3. The validator verifies all the transactions and publishes the block. His stake still remains locked and the forging reward is also not granted yet. This is so that the nodes on the network can "OK" the new block.
4. If the block is "OK"-ed, the validator gets the stake back and the reward too. If the algorithm is using a coin-age based mechanism to select validators, the validator for the current block's has its coin-age reset to 0. This puts him in a low-priority for the next validator election.
5. If the block is not verified by other nodes on the network, the validator loses its stake and is marked as "bad" by the algorithm. The process again starts from step 1 to forge the new block.

For PoS,

- There is no existence of bringing new coins into existence(as in by mining in case of bitcoin and other PoW based systems).
- Every transaction is charged some amount of fee.
- To conduct a 51% attack, the attacker will have to own 51% of the total cryptocurrency in the network which is quite expensive. This deems doing the attack too tedious, expensive and not so profitable.

Question 6

> *a. How does a DNS hijacking attack affect network security?*

Domain Name Server (DNS) hijacking, also named DNS redirection, is a type of DNS attack in which DNS queries are incorrectly resolved in order to unexpectedly redirect users to malicious sites. To perform the attack, perpetrators either install malware on user computers, take over routers, or intercept or hack DNS communication.

There are four basic types of DNS hijacking:

- Local DNS hijack — attackers install Trojan malware on a user's computer, and change the local DNS settings to redirect the user to malicious sites.
- Router DNS hijack — many routers have default passwords or firmware vulnerabilities. Attackers can take over a router and overwrite DNS settings, affecting all users connected to that router.
- Man in the middle DNS attacks — attackers intercept communication between a user and a DNS server, and provide different destination IP addresses pointing to malicious sites.
- Rogue DNS Server — attackers can hack a DNS server, and change DNS records to redirect DNS requests to malicious sites.

> *b. What is the protocol framework of HTTPS?*

HTTPS stands for hypertext transfer protocol secure and is the encrypted version of HTTP. It is used for secure communication across the internet or a network.

Unlike HTTP, HTTPS uses a secure certificate from a third-party vendor to secure a connection and verify that the site is legitimate. This secure certificate is known as an SSL Certificate.

SSL is an abbreviation for "secure sockets layer". This is what creates a secure, encrypted connection between a browser and a server, which protects the layer of communication between the two.

This certificate encrypts a connection with a level of protection that is designated at your time of the purchase of an SSL certificate. An SSL certificate provides an extra layer of security for sensitive data that you do not want third-party attackers to access. This additional security can be extremely important when it comes to running e-commerce websites.

When a HTTPS connection is going to establish, (1)a client requests to the server; (2)the server responses; (3)the client verfies the certificate; (4)key exchange; (5)the client and the server start secure communicating; (6)end and the connection closes.

> *c. How does a user verify a certificate for determining the authenticity of the website it connects to?*

To issue a certificate, the server should

- Generate public and private keys and generate certificate involving request information and signature
- Send the certificate to a Certificate Authority (CA) for checking

After CA issues the certificate, a user can

- Check the CA of the certiifcate
- Check the request information
- Check every certificate in the trust chain recursively

> *d. When is a certificate chain required? How to authenticate a certificate chain?*

Certificate chain is made up of a list of certificates that start from a server's certificate and terminate with the root certificate. If your server's certificate is to be trusted, its signature has to be traceable back to its root CA. In the certificate chain, every certificate is signed by the entity that is identified by the next certified along the chain.

Certificate chains are used in order to check that the public key and other data contained in an end-entity certificate (the first certificate in the chain) effectively belong to its subject.

A certificate chain can be authenticated recursively. The final software can be trusted to have certain properties, because if it had been illegally modified its signature would be invalid, and the previous software would not have executed it. The previous software can be trusted, because it, in turn, would not have been loaded if its signature had been invalid. The trustworthiness of each layer is guaranteed by the one before, back to the trust anchor.

Question 7

> *a. What key properties of wireless communication make it more vulnerable to attacks than wired communication?*

Broadcast Communication

- Wireless networking typically involves broadcast communication, which is far more susceptible to eavesdropping and jamming than wired networks;
- Wireless networks are also more vulnerable to active attacks that exploit vulnerabilities in communications protocols;

Higher Mobility

- far more portable and mobile, thus resulting in a number of risks;

Constrained Resource

- sophisticated OS but limited memory and processing resources to counter threats, including DoS and malware

Greater Accessibility

- may be left unattended in remote and/or hostile locations, thus greatly increasing their vulnerability to physical attacks

> *b. Why is WEP insecure?*

AP not authenticate to STA

24-bit IV in plaintext

CRC is unkeyed function

RC4 cipher: week seeds (IV) make more easily calculated keystreams

> *c. How does IEEE 802.11i provide a higher security guarantee than WEP?*

802.11i supersedes the previous security specification, Wired Equivalent Privacy (WEP), which was shown to have security vulnerabilities. Wi-Fi Protected Access (WPA) had previously been introduced by the Wi-Fi Alliance as an intermediate solution to WEP insecurities. WPA implemented a subset of a draft of 802.11i. The Wi-Fi Alliance refers to their approved, interoperable implementation of the full 802.11i as WPA2, also called RSN (Robust Security). 802.11i makes use of the Advanced Encryption Standard (AES) block cipher whereas WEP and WPA use the RC4 stream cipher.

EEE 802.11i provides a Robust Security Network (RSN) with two new protocols: the four-way handshake and the group key handshake. These utilize the authentication services and port access control described in IEEE 802.1X to establish and change the appropriate cryptographic keys. The RSN is a security network that only allows the creation of robust security network associations (RSNAs), which are a type of association used by a pair of stations (STAs) if the procedure to establish authentication or association between them includes the 4-Way Handshake.

The standard also provides two RSNA data confidentiality and integrity protocols, TKIP and CCMP, with implementation of CCMP being mandatory since the confidentiality and integrity mechanisms of TKIP are not as robust as those of CCMP. The main purpose to implement TKIP was that the algorithm should be implementable within the capabilities of most of the old devices supporting only WEP.

Question 8

> *a. Why is current Internet communication vulnerable to anonymity or privacy leakage?*

The IPs of users' deivces are exposed out when users communicate via internet and are usually fix through a complete session.

> *b. In which scenarios do users require the communication anonymity or privacy as concerned in sub-question a?*

Unmonitored access to health and medical information

Preservation of democracy: anonymous election/jury

Censorship circumvention: anonymous access to otherwise restricted information

> *c. How to use proxies to secure communication anonymity? What are the possible limitations?*

How to use:

- intermediary between sender & receiver
- Sender relays all traffic through proxy
- Encrypt destination and payload
- Asymmetric technique: receiver not involved (or informed of) anonymity

Limitations:

- Require trusted third party

  Proxy may release logs, or sell them, or blackmail sender
- Anonymity largely depends on the (likely unknown) location of attacker

> *d. How does Onion Routing provide a better guarantee for anonymity?*

Onion routing uses source-routing based anonymous overlay communication.

- Connect to Tor entry
- Randomly select a series of Tors
- Relay messages across them
- Tor exit relays messages to destination
- Reply traffic from destination traverses the reverse path
- Maintains a bidirectional persistent multi-hop path between source and destination

> *e. How to infer anonymity or privacy of Onion Routing traffic?*

Path Selection Attack

- Tor path selection algorithm:

  weight nodes by selfreported bandwidth

  select each node using weighted

  probability distribution

Counting Attack

- Correlate incoming and outgoing flowsby counting the number of packets

Low Latency Attack

- Tor router assigns each anonymous circuit its own queue
- Dequeue one packet from each queue in round-robin fashion

Cross Site Attack

- Search the accounts on public websites

Question 9

> *Consider a time-consuming authentication scenario where a database records all secret keys of a large number of users. When the system authenticates a user, it first issues a challenge message to the user. The user then uses his/her key to encrypt the challenge and then returns the encrypted challenge to the system. The system then encrypts the challenge using one key in the database after another and compares the result with the received encrypted message. Once a match is found, the system accepts the user. Otherwise, the user is denied. This authentication protocol surely takes a lot of time and computation.*
>
> *Design a possible solution to speed up the authentication process.*

The user can claim his identity when requesting an authentication or before sending the encrypted message.

Then the system knows the identity of the user and can easily locate the corresponding key.

This saves the time wasted on searching possible keys.

Question 10

Share your thoughts on the course project.

> *a. Do you aim for a research output from the course project? To what extent do you devote your time and energy to it? How do you overcome the associated challenges?*

A research output is not taken into consideration for the lack of time.

The project consumes about 5~7 hours a week, mainly on discussion, reading and analyzing.

Searching related information is the main method and turning to teammates for help is another way.

> *b. Do you think that you have gradually cultivated a research/security mindset? What is the most useful idea that you learned during this process?*

A little bit but not too much.

The most useful (better for interesting) idea I have learned is to analyze potential danger directly from parsing the source code. In fact, through building the abstract syntax tree (AST), can we know almost everything about the program or script.

> *c. Provide an example to showcase how you leverage that useful idea to facilitate problem solving in study or life.*

We analyze the AST established in the paper of our project further and attempt to add some new features to the work that the paper does.

Design a question that you think is feasible as an exam question.

> *a. Which topic among the lectures you would like to consider?*

Asymmetric cryptography

> *b. Describe a (sufficiently complex) question;*

In RSA, there are some value inappropriate for the choosing of modulus, like 35. Explain the reason and raise another such modulus.

> *c. Provide also a correct sample solution, thanks.*

If $n = 35$, it is easy to obtain $p = 5$ and $q = 7$.

Since $\gcd(e, (5-1) \times (7-1)) = \gcd(e, 24) = 1$, we can find all candidate e:

$$5, 7, 11, 13, 17, 19, 23$$

Since d must follow the rule $d \equiv e^{-1} \mod 24$, we can finally obtain that $d = e$ whichever possible c is chosen.

More such values are 33, 77 and so on.