Assignment 4: ABCs of Digital Certificates

PART-A: Comparison of Digital Certificates in the chain of trust of a popular website

- We visit the website #09 in this list of top-100 most visited websites globally ,i.e, imdb.com and download all the certificates in .CER format in the chain of trust from the root Certificate, intermediate certificate(s), to the end-user (website) certificate at the leaf in the hierarchy.
- Then, we compare the digital certificates in the chain in terms of various field values by filling this table.

Field Name	Subject (CN) of certificate holder (website)	Subject (CN) of certificate holder (intermediate)	Subject (CN) of certificate holder (root)	Remarks/observations
Issuer	Amazon RSA 2048 M01	Amazon Root CA 1	Amazon Root CA 1	Website Cert is issued by the intermediate.
Version No.	3	3	3	All are V3 certs
Signature Algo	sha256WithRSAEncryption	sha256WithRSAEncryption	sha256WithRSAEncryption	All cert uses SHA-256 with RSA Algo.
Size of digest that is signed to generate Cert Sign	32 bytes or 256 bits	32 bytes or 256 bits	32 bytes or 256 bits	SHA-256 provides a digest of size 32 bytes or 256 bits.
Size of Cert Signature	256 bytes or 2048 bits	256 bytes or 2048 bits	256 bytes or 2048 bits	RSA is used and the count of hexadecimal pairs is 256 hence 256 bytes or 2048 bits
Validity period	Not Before: Jan 18 00:00:00 2024 GMT Not After: Dec 18 23:59:59 2024 GMT	Not Before: Aug 23 22:21:28 2022 GMT Not After: Aug 23 22:21:28 2030 GMT	Not Before: May 26 00:00:00 2015 GMT Not After: Jan 17 00:00:00 2038 GMT	Website Cert Validity is 336 days. Intermediate Cert Validity is 2922 days. Root Cert Validity is 8272 days.
Is Subject field (CN), FQDN?	Yes (us.dd.imdb.com)	No	No	Only the Website (end-user) is having the FQDN.
Certificate type: DV, IV, OV or EV? Tell also how you are able to determine the type!	DV, we can determine it by looking at the Subject of the Certificate which contains only CN.	OV, we can determine it by looking at the Subject of the Certificate which contains O=Amazon	OV, we can determine it by looking at the Subject of the Certificate which contains O=Amazon	WEBSITE IS OF TYPE DV AND INTERMEDIATE AND ROOT IS OF TYPE OV. OV Stands for organization validation cert.

Subject Alternative Name(s) (SAN/UCC), if any	DNS:imdb.com DNS:www.imdb.com DNS:secure.imdb.com DNS:s.media-imdb.com DNS:us.dd.imdb.com DNS:www.imdb.to DNS:imdb-webservice.amazon.com DNS:origin-www.imdb.com DNS:origin.www.geo.imdb.com	N/A	N/A	The end-user (website) is observed to have many SAN/UCC. Whereas the intermediate and the root have a single domain cert which is also not FQDN.
Certificate category: Single domain, wildcard, Multi-domain SAN / UCC cert?	Multi-domain SAN/UCC cert	Single Domain	Single Domain	The end-user (website) is observed to have many SAN/UCC. Whereas the intermediate and the root have a single domain cert which is also not FQDN.
Public Key Info like key algo, key length, public exponent (e) in case of RSA	Key Algo: RSA (2048 bits) Exponent: 65537 (0x10001)	Key Algo: RSA (2048 bits) Exponent: 65537 (0x10001)	Key Algo: RSA (2048 bits) Exponent: 65537 (0x10001)	----------
Public key or modulus (n) in case of RSA	Modulus: 00:ed:85:2b:f4:7c:b5:90:8c:6f:e0:64:24:c9:71:47:fd:eb:7c:8f:b2:21:6a:96:68:56:a1:e8:1a:f9:4f:3f:c3:47:40:92:c8:24:8e:21:d6:98:4b:c1:57:86:49:5e:d3:6b:06:2b:99:59:54:f9:50:d9:90:f4:b1:aa:a4:08:fc:a3:0b:0d:1c:ed:5e:b8:b8:fe:e9:18:fd:66:2f:dc:48:92:9f:98:06:a1:31:dd:b3:79:b7:94:45:79:b8:75:f3:43:07:c9:eb:43:74:ed:f2:99:1a:bc:36:43:a7:d4:4c:d3:88:ad:ea:4b:51:97:6f:f5:b9:aa:b4:5e:de:97:4b:1c:fa:68:7f:1b:13:f8:cd:f6:99:6c:b5:33:df:8d:f1:49:e9:a6:ca:41:ed:55:d5:7d:9c:2c:4d:45:eb:c1:28:dd:68:5a:04:cb:29:46:11:4b:d9:18:29:1e:e3:09:37:d0:73:8a:f3:02:94:d3:44:ed:e5:fe:99:56:a1:f8:c1:f1:49:47:81:6c:6a:2c:05:e6:be:69:d3:11:5d:40:4b:c9:27:d8:45:a1:fd:6e:92:46:3a:28:de:4d:e5:8a:28:59:23:80:28:f6:d0:76:d3:a7:3a:65:dd:47:1	Modulus: 00:eb:71:2c:a9:cb:1f:88:28:92:32:30:af:8a:57:0f:78:b7:37:25:95:55:87:ac:67:5c:97:d3:22:c8:da:a2:14:67:6b:7c:f0:67:da:e2:03:2a:b3:56:12:5d:c6:b5:47:f9:67:08:a7:93:7a:95:92:18:0f:b4:f9:f9:10:36:9a:7f:2f:80:b6:4f:ba:13:4e:c7:5d:53:1e:e0:dd:96:33:07:20:d3:96:bc:12:e4:74:50:42:a1:05:13:73:b5:4f:9b:44:24:fe:2d:7f:ed:bc:22:85:ec:36:21:33:97:75:06:ce:27:18:82:dc:e3:d9:c5:82:07:8d:5e:26:01:26:26:67:1f:d9:3f:13:cf:32:ba:6b:ad:78:64:fc:aa:ff:0e:02:3c:07:df:9c:05:78:72:8c:fd:ea:75:b7:03:28:84:da:e8:6e:07:8c:d0:50:85:ef:81:54:b2:71:6e:ec:6d:62:ef:8f:94:c3:5e:e9:c4:a4:d0:91:c0:2e:24:91:98:ca:ee:ba:25:8e:d4:f6:71:b6:fb:5b:6b:38:06:48:37:47:8d:86:dc:f2:ea:06:fb:76:37:7d:9e:ff:42:4e:4d:58:82:93:cf:e2:71:c2:78:b1:7a:ab	Modulus: 00:b2:78:80:71:ca:78:d5:e3:71:af:47:80:50:74:7d:6e:d8:d7:88:76:f4:99:68:f7:58:21:60:f9:74:84:01:2f:ac:02:2d:86:d3:a0:43:7a:4e:b2:a4:d0:36:ba:01:be:8d:db:48:c8:07:17:36:4c:f4:ee:88:23:c7:3e:eb:37:f5:b5:19:f8:49:68:b0:de:d7:b9:76:38:1d:61:9e:a4:fe:82:36:a5:e5:4a:56:e4:45:e1:f9:fd:b4:16:fa:74:da:9c:9b:35:39:2f:fa:b0:20:50:06:6c:7a:d0:80:b2:a6:f9:af:ec:47:19:8f:50:38:07:dc:a2:87:39:58:f8:ba:d5:a9:f9:48:67:30:96:ee:94:78:5e:6f:89:a3:51:c0:30:86:66:a1:45:66:ba:54:eb:a3:c3:91:f9:48:dc:ff:d1:e8:30:2d:7d:2d:74:70:35:d7:88:24:f7:9e:c4:59:6e:bb:73:87:17:f2:32:46:28:b8:43:fa:b7:1d:aa:ca:b4:f2:9f:24:0e:2d:4b:f7:71:5c:5e:69:ff:ea:95:02:cb:38:8a:ae:50:38:6f:db:fb:2d:62:1b:c5:c7:1e:54:e1:77:e0:67:c8:0f:9c:87:23:d6:3f:40:20:7f:	----------

	2:97:23:96:55:db:17:55:9e:b2:d6:28:a3:f5:88:2e:02:2a:2f	:4b:5b:94:37:88:81:e4:d9:af:24:ae:f8:72:c5:65:fb:4b:b4:51:e7	20:80:c4:80:4c:3e:3b:24:26:8e:04:ae:6c:9a:c8:aa:0d	---------- ---------- ---------- ----------
Key usages; how do they vary in the chain, mention in the remarks?	Key Usage: critical Digital Signature, Key Encipherment Extended Key Usage: TLS Web Server Authentication, TLS Web Client Authentication	Key Usage: critical Digital Signature, Certificate Sign, CRL Sign Extended Key Usage: TLS Web Server Authentication, TLS Web Client Authentication	Key Usage: critical Digital Signature, Certificate Sign, CRL Sign	Differences: Website cert: key encipherment Intermediate & Root CA: cert sign, CRL sign. Extended usage consistent except Root CA which doesn't have any extended key usages.
Basic constraints, how do they vary in the chain?	CA:FALSE The website cert has CA set to FALSE, indicating it cannot issue other certs as it is the end entity.	CA:TRUE, pathlen:0 The intermediate cert has CA set to TRUE, allowing it to issue certs, but with a path length of 0, which means it can only be used for end-entity certs.	CA:TRUE The root CA has CA set to TRUE, allowing it to act as a CA without any path length restriction.	---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
Name constraints (if any), how are these useful?	None	None	None	---------- ---------- ---------- ---------- ----------
Size of the certificate	1633 bytes	1122 bytes	837 bytes	---------- ---------- ----------
URI of CRL	http://crl.r2m01.amazontrust.com/r2m01.crl	http://crl.rootca1.amazontrust.com/rootca1.crl	N/A	---------- ----------
URI of OCSP Responder	http://ocsp.r2m01.amazontrust.com	http://ocsp.rootca1.amazontrust.com	N/A	---------- ----------
Any other parameters that you found interesting?	None	None	None	---------- ---------- ---------- ---------- ----------

Answer the following queries after filling out the above table:

1. Which certificate type (DV/OV/IV/EV) is more trustable, secure, and expensive?

   -> Extended Validation (EV) certificates are the most trustworthy, secure, and expensive type of SSL certificate. They require the most stringent validation process, which includes verifying the organization's legal, physical, and operational existence. EV certificates display a green address bar and the organization's name, providing the highest level of trust and security to website visitors. EV certificates are also more secure than other types of certificates because they use a stronger encryption algorithm. This makes it more difficult for attackers to eavesdrop on or tamper with communications between the user and the website.

2. What is the role of the Subject Alternative Name (SAN) field in X.509 certificates?

   -> The SAN field in X.509 certificates is important for secure communication and entity verification in digital applications. It allows certificates to include multiple subject names, such as domain names (DNS), email addresses, IP addresses, and URIs. During certificate verification, clients rely on the SAN field to match alternative names with server hostnames, ensuring trust and secure connections. Extended Validation (EV) SSL certificates use SAN for stringent identity verification, while Multi-Domain SSL certificates simplify securing multiple domains. SAN supports virtual hosting by allowing a single certificate for multiple hosts and accommodates IP-based connections. Wildcard SANs enable a certificate to cover all subdomains of a domain. Overall, the SAN field enhances certificate flexibility, security, and ease of use, playing a crucial role in diverse digital applications and encryption scenarios.

3. Why are key usages and basic constraints different for root, intermediate and end certificates? What could go wrong if all of them have the same values?

   -> Having distinct key usages and basic constraints for root, intermediate, and end-entity certificates is crucial for maintaining the integrity and security of a public key infrastructure (PKI).

   - ➔ Root certificates, being the trust anchors, typically possess key usages enabling them to sign other certificates and Certificate Revocation Lists (CRLs), along with basic constraints identifying them as Certificate Authorities (CAs) with a path length constraint of none or 0. If pathlen is none, they can issue additional CA certificates. If pathlen is 0, it prevents root certificates from issuing additional CA certificates and establishes the foundation of trust.
   - ➔ Intermediate certificates, positioned between root and end-entity certificates, inherit key usages and basic constraints from root certificates, allowing them to sign certificates and CRLs as delegated authorities. However, their path length constraint may vary, permitting a limited depth of certificate chaining beyond themselves or 0. This hierarchical arrangement ensures controlled delegation of trust.
   - ➔ End-entity certificates, utilized for securing communication, possess key usages such as digital signature and key encipherment but lack CA designation in basic constraints, preventing them from acting as CAs and issuing further certificates.

   If all certificates shared identical key usages and basic constraints, several issues could arise:

   - ➔ Looping Certificate Chains: Allowing end-entity certificates to act as CAs could lead to circular dependencies in certificate chains, rendering them invalid.
   - ➔ Overlapping Authorities: Multiple certificates being marked as CAs might create confusion about which authority to trust, potentially compromising the PKI's integrity.

➔ Invalid End Certificates: Mistakenly designating end-entity certificates as CAs could enable them to sign certificates, violating security principles and jeopardizing the trust model.

To mitigate these risks, it's imperative to maintain hierarchical distinctions in key usages and basic constraints. Root certificates establish trust, intermediates delegate authority, and end-entity certificates serve specific functions related to securing communication between clients and servers without the ability to act as CAs. This ensures the proper functioning and security of the PKI.

4. What is the difference between Signature value and Thumbprint aka Fingerprint of a digital certificate?

-> The difference between Signature value and Thumbprint aka Fingerprint of a digital certificate are:

Signature Value

- ➔ The signature value in a digital certificate is a unique value that is generated by the Certificate Authority (CA) using a cryptographic algorithm.
- ➔ It is created by hashing the certificate's contents (including the public key, subject, issuer, validity period, and other information) and then encrypting the hash with the CA's private key.
- ➔ The purpose of the signature value is to ensure the integrity of the certificate and to verify that it was issued by a trusted CA.
- ➔ If the signature value is invalid, it means that the certificate has been tampered with or is otherwise untrustworthy.
- ➔ The signature value is typically represented as a hexadecimal string.

Thumbprint

- ➔ The thumbprint aka fingerprint of a digital certificate is a unique identifier that is derived from the certificate's signature value.
- ➔ It is a condensed representation of the signature value that is typically generated using a cryptographic hash function such as SHA-1 or SHA-256.
- ➔ The thumbprint is used to quickly and easily identify a digital certificate and to verify its authenticity.
- ➔ It is often used in certificate management systems and other security applications.
- ➔ The thumbprint is typically represented as a hexadecimal string.

Key Differences:

- ➔ Purpose: The signature value is used to ensure the integrity and authenticity of the certificate, while the thumbprint is used to uniquely identify the certificate.
- ➔ Generation: The signature value is generated by the CA using a cryptographic algorithm, while the thumbprint is generated from the signature value using a cryptographic hash function.
- ➔ Representation: The signature value is typically represented as a hexadecimal string, while the thumbprint is also typically represented as a hexadecimal string.
- ➔ Usage: The signature value is used to verify the authenticity of the certificate, while the thumbprint is used to quickly and easily identify the certificate.

5. Why do RSA key lengths increase over the years? Why is ECDSA being preferred over RSA now-a-days?

-> RSA key lengths increase over the years due to the increasing computational power of computers, which allows attackers to break RSA keys more quickly. As computers become more powerful, attackers can use more sophisticated algorithms to attack RSA keys. To stay ahead of these attacks, key lengths need to be increased periodically.

RSA and ECDSA are both widely used public-key cryptography algorithms. RSA has been the dominant algorithm for many years, but ECDSA is gaining popularity due to its advantages in terms of key size, performance, and security.

ECDSA is being preferred over RSA now-a-days because it offers several advantages, including:

- ➔ Smaller key sizes: ECDSA keys are typically much smaller than RSA keys, which makes them more efficient to use.
- ➔ Faster performance: ECDSA is generally faster than RSA, which can be important for applications where speed is critical.
- ➔ Better security/encryption: ECDSA is considered to be more secure than RSA against certain types of attacks.

6. What are pros and cons of pre-loading root and intermediate certificates in the root stores of browsers and OSes?

-> The pros and cons of pre-loading root and intermediate certificates in the root stores of browsers and OSes are listed below:

Pros:

- ➔ Improved security:
  By pre-loading root and intermediate certificates in the root stores of browsers and OSes, users can be confident that they are connecting to legitimate websites and that their data is being encrypted and protected. This helps to prevent man-in-the-middle attacks and other forms of eavesdropping.
- ➔ Simplified certificate management:
  Pre-loading certificates in the root stores reduces the need for users to manually install and manage individual certificates. This makes it easier for users to maintain a secure connection to websites without having to worry about the technical details of certificate management.
- ➔ Increased compatibility:
  By pre-loading certificates in the root stores, browsers and OSes can automatically trust websites that are signed with those certificates. This improves compatibility between different browsers and OSes, making it easier for users to access websites from any device.

Cons:

- ➔ Reduced flexibility: Pre-loading certificates in the root stores limits the ability of users to choose which certificates they trust. This can be a concern for users who want to be able to customize their security settings or who have specific security requirements.
- ➔ Potential for abuse: If a certificate authority (CA) is compromised, the pre-loaded

certificates in the root stores could be used to sign malicious websites. This could allow attackers to impersonate legitimate websites and steal user data or infect users' computers with malware.

➔ Increased attack surface: Pre-loading certificates in the root stores increases the attack surface for attackers. This is because attackers can target the root stores themselves or the CAs that issue the pre-loaded certificates.

7. Why are root CAs kept offline? How do they issue certificates to intermediate CAs?

-> Root CAs are kept offline due to the following reasons:

➔ Security: Keeping root CAs offline protects them from unauthorized access and compromise. If a root CA is compromised, it could allow an attacker to issue fraudulent certificates, which could be used to impersonate legitimate websites or services. By keeping root CAs offline, the risk of compromise is significantly reduced.

➔ Reliability: Offline root CAs are less likely to experience outages or other disruptions. If a root CA is hosted online, it could be vulnerable to denial-of-service attacks or other network issues. By keeping root CAs offline, the risk of outages is minimized.

Root CAs issue certificates to intermediate CAs in the following ways:

➔ Generation of Signing Request:
Root CAs are needed to issue certificates to intermediate CAs and other entities. Intermediate CAs then issue certificates to end-entities, such as websites and email servers. The process of issuing a certificate begins with the generation of a signing request by the intermediate CA.

➔ Approval of Signing Request:
Once the signing request is generated, it is sent to the root CA for approval. The root CA will verify the identity of the intermediate CA and ensure that it is authorized to issue certificates. If the request is approved, the root CA will issue a certificate to the intermediate CA.

➔ Distribution of Certificate:
The certificate issued by the root CA is then distributed to the intermediate CA. This can be done through a variety of methods, such as secure email or physical delivery. Once the intermediate CA receives the certificate, it can begin issuing certificates to end-entities.

8. Why are root and intermediate certificates of new CAs cross-signed by the legacy CAs? Answer this by taking Let's Encrypt and its parent organization as root and intermediate CAs.

-> When Let's Encrypt was first established, it faced a significant challenge in gaining the necessary trust and credibility to issue SSL/TLS certificates that would be widely accepted by browsers and operating systems. To tackle this issue, Let's Encrypt started cross-signing its certificates.

Let's Encrypt cross-signs its root and intermediate certificates with legacy CAs to increase client compatibility and ensure that its certificates are widely accepted by older devices and operating systems.

Let's Encrypt root and intermediate certificates are cross-signed by legacy CAs, such as

IdenTrust's "DST Root CA X3" (now "TrustID X3 Root"), to increase client compatibility. This means that each of Let's Encrypt's RSA intermediates has two certificates representing the same signing key: one signed by the legacy CA (IdenTrust) and the other signed by Let's Encrypt's own root certificate, ISRG Root X1.

By having cross-signatures, Let's Encrypt ensures that its certificates are widely accepted by older devices and operating systems that may not yet trust ISRG Root X1. For example, Windows XP and Android 7 have better compatibility with the IdenTrust root, which has been around longer.

Similarly, Let's Encrypt's ECDSA root certificate, ISRG Root X2, is represented by two certificates: one that is self-signed and one that is signed by ISRG Root X1. This cross-signing ensures compatibility until ISRG Root X2 becomes widely trusted.

9. What challenge is posed to the certificate seekers (Alice) by Let's Encrypt CA before issuing wildcard certificates? How does Alice respond to it so that she passes it?

-> Let's Encrypt CA imposes a validation challenge on certificate seekers like Alice before issuing wildcard certificates, requiring them to demonstrate or prove their control over the entire domain associated with the certificate request. This verification process aims to ensure the security and integrity of the certificate issuance process, preventing unauthorized entities from obtaining certificates for domains they do not control.

To successfully meet this challenge, Alice must provide evidence of her authority or ownership over the domain for which she seeks a wildcard certificate. Let's Encrypt typically presents Alice with specific instructions on how to fulfill this requirement, commonly involving the creation of a DNS TXT record.

In response, Alice accesses the DNS management interface provided by her domain registrar or hosting provider and creates a TXT record containing a unique value provided by Let's Encrypt. This action proves Alice's ability to control the DNS settings for the domain in question.

After creating the DNS TXT record, Alice must wait for it to propagate throughout the DNS system, which can take some time depending on the DNS provider's settings. Once the TXT record is visible across the DNS infrastructure, Let's Encrypt performs a validation check by querying the DNS records associated with Alice's domain.

If Let's Encrypt successfully retrieves the TXT record and verifies that its content matches the expected value, it confirms that Alice has passed the validation challenge. With this confirmation, Alice can proceed with the issuance of the wildcard certificate for her domain.

10. List out names of OS/Browser/Company whose root stores were pre-populated with Root and Intermediate CA certificates of the website #09 (imdb.com)?

-> Names of OS/Browsers whose root stores were pre-populated with Root/Intermediate CA certs.

OS: Windows 11, MacOS, Linux (Kali) - Root CA is pre-populated in all. Only in Windows 11 - Intermediate CA is pre-populated.

Browser: Chrome (Chromium), Microsoft Edge, Mozilla Firefox - Pre-populated only with Root CA.

PART-B

1. A browser X has received the digital certificate of the website #09 (imdb.com) over a TLS connection. How does it verify whether the certificate is valid? Write a psuedo-code of browser X's verifier function named myCertVerifier( ) and explain how it works by picking the entire chain of trust of an end-user cert (of the website #09 (imdb.com)) in PART-A of this assignment.

-> To verify whether a digital certificate is valid, a browser typically performs several checks including verifying the certificate chain, checking for revocation, and ensuring the certificate is issued by a trusted certificate authority (CA). Here's a pseudo-code of a certificate verifier function named myCertVerifier() in a browser X:

```
function myCertVerifier(certificate) {
  // Step1: Extract fields from the certificate
  issuer_name = certificate.issuer_name
  subject_name = certificate.subject_name
  certificate_expiration_date = certificate.certificate_expiration_date
  public_key = certificate.public_key
  signature = certificate.signature

  // Step2: Check if the certificate is expired
  if (certificate_expiration_date < current_date) { return false }

  // Step3: Get the issuer's certificate from the trust store
  issuer_certificate = getCertificateFromTrustStore(issuer_name)

  // Step4: Verify the issuer's certificate
  if (!verifyCertificate(issuer_certificate)) { return false }

  // Step5: Verify the subject's name matches the expected hostname
  if (subject_name != expected_hostname) { return false}

  // Step6: Verify the issuer's signature on the certificate
    if (!verifySignature(issuer_certificate.public_key, signature, certificate)) {
return false }

  // Step7: Verify the certificate is within the validity period of the issuer's certificate
   if (issuer_certificate.certificate_expiration_date < certificate_expiration_date) {
return false }

  // Step8: Check if the certificate is revoked
  if (isCertificateRevoked(certificate)) { return false }

  // Step9: All verification checks passed, certificate is valid
  return true
 }
```

Explanation:

1. Browser X extracts the necessary fields from the digital certificate it received from the website #09 (imdb.com). Information like the issuer's name, subject's name, expiration date, public key, and signature is retrieved.

2. Browser X checks if the certificate is expired. If the expiration date is before the current date, the certificate is considered invalid.
3. Browser X retrieves the issuer's certificate from its trust store.
4. Browser X verifies the issuer's certificate. It uses the issuer's public key (extracted from the issuer's certificate) to verify the issuer's signature on the issuer's certificate.
5. Browser X checks if the certificate's subject name (website address) matches the expected hostname. This is an essential step to ensure that the certificate is intended for the website being visited.
6. Browser X verifies the issuer's signature on the certificate. It uses the issuer's public key to verify the signature on the certificate. If the signature is valid, it means that the issuer has issued the certificate to the website #09 (imdb.com).
7. Browser X checks if the certificate's expiration date is within the validity period of the issuer's certificate. If the certificate expires after the expiration date of the issuer's certificate, it is considered invalid.
8. Browser X checks if the certificate has been revoked. It checks with a Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) to see if the certificate has been revoked due to security concerns. If the certificate is revoked, it is considered invalid.
9. If all the checks pass, Browser X considers the certificate valid and allows the TLS connection to proceed.

2. Consider the scenario in which evil Trudy has used the domain validated (DV) digital certificate of the website (Bob) named Bob.com to launch her own web server with the domain name, xyz.com. Does your function myCertVerify( ) returns valid or invalid for this when someone like Alice (browser) tries to access Trudy's website xyz.com?

-> myCertVerify( ) returns invalid.

In this scenario, evil Trudy has used Bob's DV certificate, which is issued for the domain bob.com, to launch her own website xyz.com. When Alice tries to access Trudy's website xyz.com, her browser will attempt to verify the digital certificate presented by Trudy's server.

The myCertVerifier() function will perform the following checks:

- ➔ It will extract the subject name from the certificate, which will be xyz.com.
- ➔ It will compare the subject name with the expected hostname, which is also xyz.com.
- ➔ Since the subject name matches the expected hostname, the function will proceed to verify the issuer's signature on the certificate.
- ➔ However, the issuer of the certificate is still Bob's website (bob.com), not Trudy's website (xyz.com).
- ➔ Therefore, the signature verification will fail, and the myCertVerifier() function will return invalid.

As a result, Alice's browser will display a warning or error message indicating that the certificate is not valid for the website she is trying to access. This will prevent Alice from establishing a secure connection with Trudy's website, protecting her from potential phishing or man-in-the-middle attacks.

3. Consider another scenario in which evil Trudy has used the digital certificate of Bob's website Bob.com to launch her own web server with the domain name, xyz.com. When a web client (Alice) tries to connect with Bob's website by sending a DNS query, Trudy responds with her IP address by DNS cache poisoning (What is DNS cache poisoning? | DNS spoofing | Cloudflare) Does your function myCertVerifier( ) returns valid or invalid for this and what are the consequences? What kind of attacks can Trudy launch in this scenario?

-> myCertVerifier( ) returns valid.

In this scenario, evil Trudy has used Bob's digital certificate to launch her own website xyz.com and has poisoned the DNS cache so that when Alice tries to connect to Bob's website, she is directed to Trudy's server instead.

When Alice's browser connects to Trudy's server, it will receive Trudy's digital certificate, which is issued for the domain bob.com. The myCertVerifier() function will perform the following checks:

- ➔ It will extract the subject name from the certificate, which will be bob.com.
- ➔ It will compare the subject name with the expected hostname, which is also bob.com.
- ➔ Since the subject name matches the expected hostname, the function will proceed to verify the issuer's signature on the certificate.
- ➔ The issuer of the certificate is still Bob's website (bob.com), which is a trusted issuer.
- ➔ Therefore, the signature verification will pass, and the myCertVerifier() function will return valid.

As a result, Alice's browser will establish a secure TLS connection with Trudy's server, believing that it is connecting to Bob's website. This is because the certificate presented by Trudy's server is valid for the domain bob.com, which is the domain name that Alice intended to access.

Consequences:

- ➔ Phishing Attack:
  Trudy can launch a phishing attack by creating a fake login page for Bob's website on her server. When Alice visits xyz.com (which she believes is Bob's website), she will see the fake login page and may enter her credentials, thinking she is logging into Bob's website. Trudy can then steal Alice's credentials and use them to access her accounts on Bob's website.

- ➔ Man-in-the-Middle Attack:
  Trudy can launch a man-in-the-middle attack by intercepting the communication between Alice and Bob's website. Since Trudy has a valid certificate for bob.com, Alice's browser will trust the connection and send her data (including sensitive information like passwords and credit card numbers) to Trudy's server. Trudy can then modify or steal this data before forwarding it to Bob's website.

Briefly explain how 7-zip uses the password to encrypt compressed files using secure hash and symmetric algorithms. What role does the password length play in brute force attacks to decrypt the encrypted files?

-> The method used by 7-zip to secure compressed files involves password hashing, key derivation, and symmetric encryption. The password undergoes hashing to generate a key hash,

which is then used to derive encryption keys. A key derivation function incorporates the password hash and a unique salt to generate encryption and decryption keys. Symmetric encryption, such as AES-256, is employed to encrypt the file data using the derived encryption key. During decryption, the password is hashed again, and the key hash is derived using the same process. The encryption key is then generated, and the file is decrypted.

Regarding the impact of password length on brute force attacks:

- ➔ Longer passwords significantly increase the number of possible key combinations, making brute force attacks more computationally challenging and time-consuming.
- ➔ Shorter passwords have fewer possible combinations, making them more susceptible to brute force attacks, where attackers could potentially guess the password in less time.

In our case, the password "CS23MTECH11009" has moderate length (14) involving alphanumeric characters, providing some protection against brute force attacks compared to shorter passwords. However, it could still be vulnerable to such attacks, especially if attackers have access to advanced computing resources.

REFERENCES

[1] Crt.sh
[2] Top 100 Most Visited Websites (US and Worldwide)
[3] ASN.1 JavaScript decoder
[4] phpseclib: X.509 Decoder
[5] DV, OV, IV, and EV Certificates - SSL.com
[6] 7-Zip (7-zip.org)
[7] imdb.com
[8] net/data/ssl/chrome_root_store/root_store.md
[9] Microsoft Included CA Certificate List
[10] Mozilla Included CA Certificate List
[11]

PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honor violations by other students if I become aware of it.

Name: Kritik Agarwal (CS23MTECH11009)
Date: Feb 6, 2024
Signature: K.A.