Cs22mtech12005

Assignment 4: ABCs of Digital Certificates

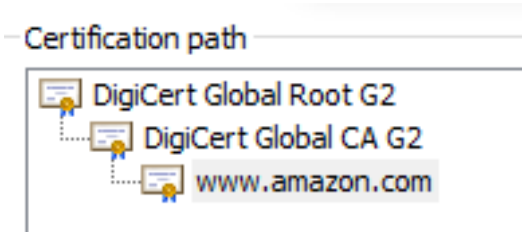
Individual Assignment

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

- Visit the website #N in this list of top-100 most visited websites globally where is #N is the last two digits in your roll number and download all the certificates in .CER or .PEM 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.
- Compare the digital certificates in the chain in terms of various field values by filling this table.

Website : www.amazon.com

Chain of Trust :



Field Name	**Subject (CN) of certificate holder (website) www.amazon.com**	**Subject (CN) of certificate holder (intermediate) DigiCert Global CA G2**	**Subject (CN) of certificate holder (root) DigiCert Global Root G2**	**Remarks/observations**
Issuer	DigiCert Global CA G2	DigiCert Global Root G2	DigiCert Global Root G2	Issuer name in current certificate matches the subject name in issuer's certificate. For root CA, the issuer is same because it is self signed certificate.

Version No.	V3	V3	V3	It is the version of the X.509 certificate format.
Signature Algo	PKCS #1 SHA-256 With RSA Encryption	PKCS #1 SHA-256 With RSA Encryption	PKCS #1 SHA-256 With RSA Encryption	Hashing is used along with encryption for integrity and authentication both.
Size of digest that is signed to generate Cert Sign	256 bits	256 bits	256 bits	Sha256 fingerprint size is 256 bits
Size of Cert Signature	2048 bits	2048 bits	2048 bits	Certificate signature value size (16 rows * 16 columns * 2 hexadecimals in each cell * 4 bits in each hexadecimal)
Validity period	1 Year [19 October 2022 05:30:00 to 19 October 2023 05:29:59]	15 Years [01 August 2013 17:30:00 to 01 August 2028 17:30:00]	24 Years & 5 months approx [01 August 2013 17:30:00 to 15 January 2038 17:30:00]	Validity increases as we go from leaf to the root. Reason being, as we go up in the hierarchy, trust increases. More the trust, more the validity period. Also, end entities are having less validity than the CAs.
Is Subject field (CN), FQDN?	Yes	No	Yes	CN is not partial. It is given www.amazon.com; so there are 3 levels specified. For Root, I can see www.digicert.com, but for intermediate, no domain name was visible.
Certificate type: DV, IV, OV or EV? Tell also	DV	OV/ EV	EV	DV is explicitly mentioned in the certificate. Also, I

how you are able to determine the type!				verified by the object id attached to the certificate in the object repository, there it said DV. One more way to check it is that it shows only the common name field in subject. For the intermediate, organization details are mentioned, thus it could be EV or OV. There was no field to check explicitly if it is EV or OV since it also doesn't have any domain url. Root CA is EV (verified from chrome, it says EV in the issued by field). Also, Root CA must be EV because its the one we are supposed to trust. It should have the highest level of validation i.e EV.
Subject Alternative Name(s) (SAN/UCC), if any	☐ yp.amazon.com ☐ yellowpages.amazon.com ☐ www.m.amazon.com ☐ www.cdn.amazon.com ☐ www.amzn.com ☐ www.amazon.com ☐ us.amazon.com ☐ uedata.amazon.com ☐ test-www.amazon.com ☐ p-yo-www-amazon-com-kalias.amazon.com	No SANs	No SANs	A normal certificate protects a primary domain name. But SAN certificates(Certificate with SAN field) allows protection of multiple domains. The SAN field in X.509 certificates allows sharing a certificate for many domain names. This same certificate can be used for protection of multiple domain names given in this SAN field. SAN field is very useful in scenarios where multiple domains are mapped to a single IP address.

	☐ p-y3-www-amazon-com-kalias.amazon.com ☐ p-nt-www-amazon-com-kalias.amazon.com ☐ mp3recs.amazon.com ☐ konrad-test.amazon.com ☐ iphone.amazon.com ☐ home.amazon.com ☐ corporate.amazon.com ☐ buybox.amazon.com ☐ amzn.com ☐ amazon.com			
Certificate category: Single domain, wildcard, Multi-domain SAN/UCC cert?	Multi domain SAN certificate	Single domain	Single domain	SAN certificate could be multi domain or/and wildcard. But certificates with no SAN field indicates single domain.
Public Key Info like key algo, key length, public exponent (e) in case of RSA	Key Algorithm : RSA Key Length : 2048 bits Public exponent(e) : 65537	Key Algorithm : RSA Key Length : 2048 bits Public exponent(e) : 65537	Key Algorithm : RSA Key Length : 2048 bits Public exponent (e) : 65537	Gives the public key information. This will be used for encryption/ decryption in RSA algorithm.
Public key or modulus (n) in case of RSA	B9:70:F6:D6:BE:FE:19:BC:A0:A9:96:CC:53:6F:D7:20:01:C8:47:9F:0C:E8:9B:23:66:A9:1B:DE:F4:DE:5A:04:97:1F:D5	D3:48:7C:BE:F3:05:86:5D:5B:D5:2F:85:4E:4B:E0:86:AD:15:AC:61:CF:5B:AF:3E:6A:0A:47:FB:9A:76:91:60:0B:8A:6	BB:37:CD:34:DC:7B:6B:C9:B2:68:90:AD:4A:75:FF:46:BA:21:0A	

	:6A:16:4E:BD:08:87:23:EF:EA:E8:C5:9F:08:D1:14:0D:C2:20:6A:2F:E5:36:1F:86:A7:0A:90:4A:18:82:26:C6:3B:58:EB:27:77:33:10:17:9B:CD:A0:38:17:73:28:10:EA:18:CF:6B:33:F6:B0:52:B4:4B:17:4E:56:49:C7:E0:B8:97:3D:5F:CF:B7:3B:83:91:7F:DB:2A:3F:CA:37:F4:E7:96:2D:C8:28:52:4A:D4:1C:85:E3:82:D0:0C:62:3E:03:3D:CB:9E:98:55:9B:29:6B:89:D1:10:58:25:A4:64:E2:54:10:8C:FA:EF:F4:44:ED:05:DF:02:C3:56:F0:01:E0:33:74:F8:14:84:D9:49:BE:6B:3A:05:61:2F:FD:39:20:76:D8:FE:2B:AA:5E:2A:78:39:A6:CF:C3:B5:2E:AA:3B:EE:9E:A2:66:F4:0C:CE:34:69:A5:3D:2E:48:9D:5F:DD:3D:9F:69:59:75:C5:A2:58:C5:85:BA:E6:73:3B:6D:2B:BA:72:DE:4A:4E:AA:30:BF:1B:0D:75:F7:7B:9C:A3:CF:BC:CC:12:63:4F:D5:2E:68:D6:00:EB:B9	B:CD:CF:DC:57:7E:60:98:0B:E4:54:D9:56:ED:21:CC:02:B6:5A:81:5F:97:6A:EE:02:2F:23:27:B8:6D:D4:B0:E7:06:02:78:0B:1F:5C:A9:99:36:FE:BB:AC:1B:05:FA:57:CD:81:10:40:67:D6:30:8B:58:35:D4:96:61:BE:D0:8C:7A:97:9F:1A:F9:22:E6:14:2F:A9:C6:E8:01:1F:AB:F8:26:0F:AC:8E:4D:2C:32:39:1D:81:9B:8D:1C:65:B2:1C:DB:61:A8:89:2F:60:E7:EB:C2:4A:18:C4:6F:2A:E9:10:92:09:ED:17:D1:00:2B:E6:7D:EF:04:89:14:4E:33:A1:B2:0F:97:87:9F:B3:A0:CD:2F:BC:2C:EC:B8:83:68:31:3D:1F:D5:4A:90:10:19:0B:81:95:D6:29:76:51:F9:36:76:D0:B7:09:7A:38:4A:D7:6F:8C:BF:13:7C:39:ED:BA:AE:90:FC:95:F7:7B:78:09:36:5E:74:93:1E:25:F0:FF:D4:AD:AE:68:6B:C6:FF:0F:D5:35:F1:55:6E:48:49:F8:F8:B8:EF:88:F8:F1:5E:11:77:AA:DF:02:B3	:08:8D:F5:19:54:C9:FB:88:DB:F3:AE:F2:3A:89:91:3C:7A:E6:AB:06:1A:6B:CF:AC:2D:E8:5E:09:24:44:BA:62:9A:7E:D6:A3:A8:7E:E0:54:75:20:05:AC:50:B7:9C:63:1A:6C:30:DC:DA:1F:19:B1:D7:1E:DE:FD:D7:E0:CB:94:83:37:AE:EC:1F:43:4E:DD:7B:2C:D2:BD:2E:A5:2F:E4:A9:B8:AD:3A:D4:99:A4:B6:25:E9:9B:6B:00:60:92:60:FF:4F:21:49:18:F7:67:90:AB:61:06:9C:8F:F2:BA:E9:B4:E9:92:32:6B:B5:F3:57:E8:5D:1B:CD:8C:1D:AB:95:04:95:49:F3:35:2D:96:	

			E3:49:6D:DD:77:E3:FB:49:4B:B4:AC:55:07:A9:8F:95:B3:B4:23:BB:4C:6D:45:F0:F6:A9:B2:95:30:B4:FD:4C:55:8C:27:4A:57:14:7C:82:9D:CD:73:92:D3:16:4A:06:0C:8C:50:D1:8F:1E:09:BE:17:A1:E6:21:CA:FD:83:E5:10:BC:83:A5:0A:C4:67:28:F6:73:14:14:3D:46:76:C3:87:14:89:21:34:4D:AF:0F:45:0C:A6:49:A1:BA:BB:9C:C5:B1:33:83:29:85	
Key usages; how do they vary in the chain, mention in the remarks?	● Digital Signature ● Key Encipherment (a0)	● Digital Signature ● Certificate Signing ● Off-line CRL Signing ● CRL Signing (86)	● Digital Signature ● Certificate Signing ● Off-line CRL Signi	Key usages vary according the the purpose that that entity will use that public key for. End entities will use it for different purposes than intermediate and root CAs.

			• ng • CRL Signing (86)	
Basic constraints, how do they vary in the chain?	Subject Type=End Entity Path Length Constraint=None	Subject Type=CA Path Length Constraint=0	Subject Type=CA Path Length Constraint=None	Gives information on if the entity is leaf or a CA, i.e whether that entity has rights to issue certificates or not.
Name constraints (if any), how are these useful?	None	None	None	These are domain name constraints.
Size of the certificate	2,816 Bytes	1,662 Bytes	1,316 Bytes	Calculated from properties.
URI of CRL	http://crl3.digicert.com/DigiCertGlobalCAG2.crl http://crl4.digicert.com/DigiCertGlobalCAG2.crl	http://crl4.digicert.com/DigiCertGlobalRootG2.crl http://crl3.digicert.com/DigiCertGlobalRootG2.crl	None	Gives the url where one can find the revocation list entries of the certificates which are revoked along with the reason for the same.
URI of OCSP Responder	http://ocsp.digicert.com	http://ocsp.digicert.com	None	
Any other parameters that you found interesting? **[Extended Key Useage, Thumbprint, SCT List, Certificate Policies]**	Extended Key Useage : TLS WWW Server Authentication TLS WWW Client Authentication	none	None	A CA must keep track of the certificate issuance with one or more CT logs while issuing certificates (publicly run servers that provably record certificate issuance). The SCT provides information on the issuance's timing, the log it was logged in, and where to look for it (using the signature data). The client is required to confirm that

				the certificate in the log matches the certificate you are validating in order to ensure the accuracy of the information.

Answer the following queries after filling out the above table:

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

   The order goes like this :
   1. DV
   2. OV/IV
   3. EV

   With EV most trustable and expensive. DV only validates anyone who is having the control of that domain, while OV validates the actual organization or individual along with its location check. So, one can trust the owner of the website. EV requires most tasks from CA's side to validate, happens in the presence of people and an actual person goes to the location for validation. Since it is more rigorous, it is more expensive and because of so much deeper validation, it is the most trustful certificate type amongst all.

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

   A normal certificate protects a primary domain name. But SAN certificates(Certificate with SAN field) allows protection of multiple domains. The SAN field in X.509 certificates allows sharing a certificate for many domain names. This same certificate can be used for protection of multiple domain names given in this SAN field. The domains could be fully qualified (FQDN eg. www.cse..iith.ac.in) and also supports single level wild card domain names (eg. It supports entries like www.*.iith.ac.in but not like *.*.iith.ac.in). A few entries(say first 3) are free, charges will apply after that per entry. Domain names can be reissued or changed because of this field easily without any extra costs. SAN field is very useful in scenarios where multiple domains are mapped to a single IP address.

3. Why are key usages and basic constraints different for root, intermediate and end certificates?

   Key usage field defines the usage of the public key given in the certificate. So, we have 3 public keys: one for the end entity, intermediate and the root. The public key usage of the end entity is different from the other two. It is mostly only used for authentication of

any other entity that this end node will try to communicate with in future. Also, if SSL/TLS protocol is used, key encipherment is also done by the same key, meaning this public key will be used for exchanging the session/ symmetric key. This is also true for public keys of intermediate and root nodes. But, public keys of theirs are also used for certificate signing which is not a requirement for the end entity. The signatures on the certificates can be verified by using this key. But this usage is only done for the CAs(int and root node). Also, the public key of these nodes can also be used for CRL signing (verification of the signature on revocation data like CRL). This is not required by the end entity. So because of the different purposes of the public key, key usages are different.

Basic constraints indicate whether the entity is a CA or an end entity. In the chain, the last leaf entity is the end entity and is not CA, whereas the intermediate and the root entities are Certificate Authorities. Also, it is used to put an upper bound on the path length on the chain of the certificate. Path length = None means unlimited CA certificates can follow the current CA certificate in chain. For root, it says PL = 1; means 1 more intermediate CA can sign below this CA certificate in chain. But for intermediate CA, it says PL =0 indicating that no other CA can provide a certificate to anyone below this. Thai certificate is the last and can be used by the one whom it's issued to. At the end entity, it says PL=none i.e no constraint. Anyhow the end entity is not a CA, so it need not bother about that detail. Because of all these differences, the basic constraints differ for these 3 different types of entities.

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

Signature value is 2048 bits(sha-256 with RSA encryption). There are two fingerprints on the digital certificate : SHA-256 Fingerprint(256 bits) and SHA-1 Fingerprint(160 bits). Although both are hashes, the use cases differ. Signature is used for verifying that in fact a CA signed this certificate and no one else. Fingerprint on other hand is a hash on the entire certificate(also includes the signature) attached to an object of the certificate and is used for locating the certificate in the certificate store. Thumbprint, as the name suggests is very similar to the actual thumbprint in real life, no one else mostly would have the same one, similarly, no one else would have the same thumbprint for the certificate, so mostly works like a unique identifier for the certificate. Whereas, we use signatures for a totally different purpose, that is for security measures, i.e to check if the certificate is not forged by anyone and is a legitimate one, i.e for “Authentication” purposes. Thumbprints are not used for security purposes, but are only a way to reference a particular certificate. There is no connection between these two fields. Signature is a part of the certificate, while thumbprint is not. For our reference of the certificate, a calculation and display is done. Usage of the SHA-1 hashing algorithm can compromise security, so it should not be used for signatures, but can be used for thumbprints(Windows uses it).

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

   The RSA algorithm's security is based on the difficulty of factoring the number into two big primes. The Bigger the key length, the harder it is to factorize. It is harder to break a key using methods like brute-forcing and factoring the longer the key is, or the more bits it contains. Integer factorization can be used to crack RSA, requiring far longer RSA keys to obtain the same level of protection as in AES. A group of scholars that worked on a project for cracking 768 bits RSA for two years used hundreds of devices. The time and materials required for attacking make it unattainable for the majority of hackers. Thus, RSA key lengths have increased over the years.

   ECDSA(Elliptic Curve Cryptography) is used over RSA because it gives better performance & better or equivalent security than RSA with smaller key size and overhead than RSA in terms of the bruteforce amount the attacker needs to do. But both can be defeated by Shor's algorithm and a modified version of it, so as such there is no clear winner in between them, it is only in terms of the key size. ECDSA signatures are smaller than RSAs'. Thus, ECDSA is preferred only because of performance over RSA.

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

   Pros :
   We already verified Bob's certificate. Now, for the chain of trust, we need to verify certificates of intermediate and root CAs. If those certificates are already stored in the root stores of browsers and OSes, there is no need to contact these entities or any certificate repository. Those can be directly verified reducing the delay in verification and the communication can be initiated quickly.

   Cons:
   If in between of the communication, for some reason, the certificate of the root or the intermediate CA is revoked, Alice would not be able to verify the same because she will be referring to the certificates stored locally. This could lead to unsafe communication and could break the chain of trust from the second level.

7. Why are root CAs kept offline?

   Root CA comes at the top of hierarchy. If it gets compromised, the entire PKI will get compromised. If lower level CAs get compromised, it does not affect the upper layers, but only the ones who trust on this lower CA will be compromised. But, every end entity and intermediate CAs trusts root CAs and thus it needs highest security. If a node is connected to a network, it is prone to many types of attacks. If an intermediate CA gets

compromised, its certificate can be reissued from root CAs, but this is not the case when root CAs get compromised. Thus, to address this, CAs are kept offline. Only when some intermediate CAs or other root CAs need certificate issuing/ cross issuing, then they are made online and kept offline again, this also happens in the presence of a lot of people. So, these only need to be physically protected. Keeping it offline will separate it from the PKI and in turn protect the PKI from collapsing.

8. Why are root and intermediate certificates of new CAs cross-signed by the legacy CAs?
   For additional client compatibility

   Cross signing is done because we want to establish a chain of trust between the CAs itself, from one trusted CA to multiple other CAs. It is about expanding trust not just in the downward direction but as the name suggests, in cross direction too, in the same or upward direction as well. If there is a CA whom we trust the most, i.e legacy CA, if this CA signs another CA or even another root CA(which might be a new one), this expands the trust to the new CA. Also, if one is having a certificate signed by CA1 and CA2, it could be more trusted because two separate entities have verified your certificate. It is also used in situations where you have clients that trust CA1 or CA2 but not both. In such a case, it can be cross-signed by both to be trusted by both. Instances where the private key of a CA is exposed. Your certificate is signed by CA1 and CA2, but let's assume CA1's key leaks. After the leak, CA1 revokes its public key, and whatever it issues thereafter cannot be trusted. However, any client who trusts CA2 can still maintain a level of faith in your certificate because it is cross-signed to CA2 as well.

   There is one more usage of cross-signing and that is some CAs are having greater client compatibility, better compatibility with older devices and OSes(Android 7, Windows XP, etc).

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 say Alice’s domain is abc.com. Alice is given the DNS-01 Challenge. LE CA challenges Alice to prove her control over the DNS for her domain name. It asks Alice to put a specific value in the TXT- record under that domain name. Basically, it gives Alice a “token”. Alice can create a new server where she can use it for responding. Alice will create a TXT-record from this token and the account key and keep it at DNS-01Challenge.abc.com. LE then fires a query to the DNS system for this record. If it finds a match, Alice will be issued the certificate. On this server, Alice has all the APIs for the DNS access. So, even if that server is compromised in all this communication, her actual web server which is hosted on some other server is not compromised. This is one way Alice can respond, or another is simply there is no need to create an entirely new server, but for security reasons, it's better to respond in the previous way. So, if LE CA gets a match after querying, Alice will pass the test and this is how Alice will be issued a

certificate.

10. List out names of OS/Browser/Company whose root stores pre-populated with Root and Intermediate CA certificates of the website #N?

    1. Mozilla
    2. Microsoft
    3. Google

    (From digicert website which led to https://www.ccadb.org/)

PART-B

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

```
myCertVerifySignature(Y.certificate){

  if(!rootCA) {
  //rootCA = bool which tells if a CA is root CA or not
  //intermediate CAs

  //first verify the certificates from the issuer

    if(!checkTrustedStore(Y.Issuer)){
      //the certificate is not present in your trusted store
      pkCA,flag=myCertVerifySignature(getCertificate(Y.Issuer));
      //pkCA = public key of upper CA

    }
    else  {
      //the certificate is present in your trusted store
      pkCA,flag=myCertVerifySignature(fetchCertificate(Y.Issuer));
    }

    //you have successfully validated the upper hierarchy and got its public key
    //now validate this particular end entity/ int CA

  }
  else {
```

```
    //root CA
        if(!isPresentinTrustedStore(Y)) exit(0);
        pkCA=pk;
    }

    //verify the signature for authenticity and integrity
    decryptedSignature = decryptionAlgo(Y.signature, pkCA);
    hashedMessage=Hash(pk+identifying information in Y.certificate);

     if(decryptedSignature!=hashedMessage || Y.IssuerID ! = Y.Issuer.ID || (Y.EndEntity ==
CA && Y.PathLength>Y.Issuer.PathLength-1) || !nameConstraintCheck() ||
!policyConstraintCheck() || !keyUseageCheck() || !otherCriticalExtensionsCheck()
||(if(Y.EndEntity==EndEntity && !OCSPResponseStatus) ){
        flag = 0 ;
        exit(0);
    }

if(flag) printf("Valid certificate");
 else printf("Invalid certificate");


    return Y.pk;

}
```

Explanation :

We have Amazon.com's certificate. Now, to verify its signature, we need validated public keys from the intermediate and root CAs. So, we fetch those certificates first and validate their signatures first starting from root CA till we reach the leaf. So, once root CAs signature is validated, we then use its public key to validate the signature of the intermediate CA and then use the intermediate CA's public key to validate the signature on Amazon.com's certificate. While doing so in recursion with root CA validated firs, we also cross check the issuer's id in a certificate and the same id in the issuer certificate's subject field. We also then check for the end entity, the OCSP response, and constraint checking like name, policy, key usage and other Critical extensions checking. Once all these checks are done, in the chain of trust, if the flag is still valid, then the certificate is valid else it is not.

2. Consider the scenario in which evil Trudy has used the domain validated (DV) digital certificate of the website (Bob) named abc.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?

It will validate. Reason is that it is not checking the domain name explicitly. Even if Trudy is running xyz.com, she is giving Bob's Certificate along with Bob's public key. So, the verification above will match exactly as the verification is done over Bob's identity, over Bob's public key only, there is no information of Trudy explicitly involved here anywhere. So myCertVerify() will return valid.

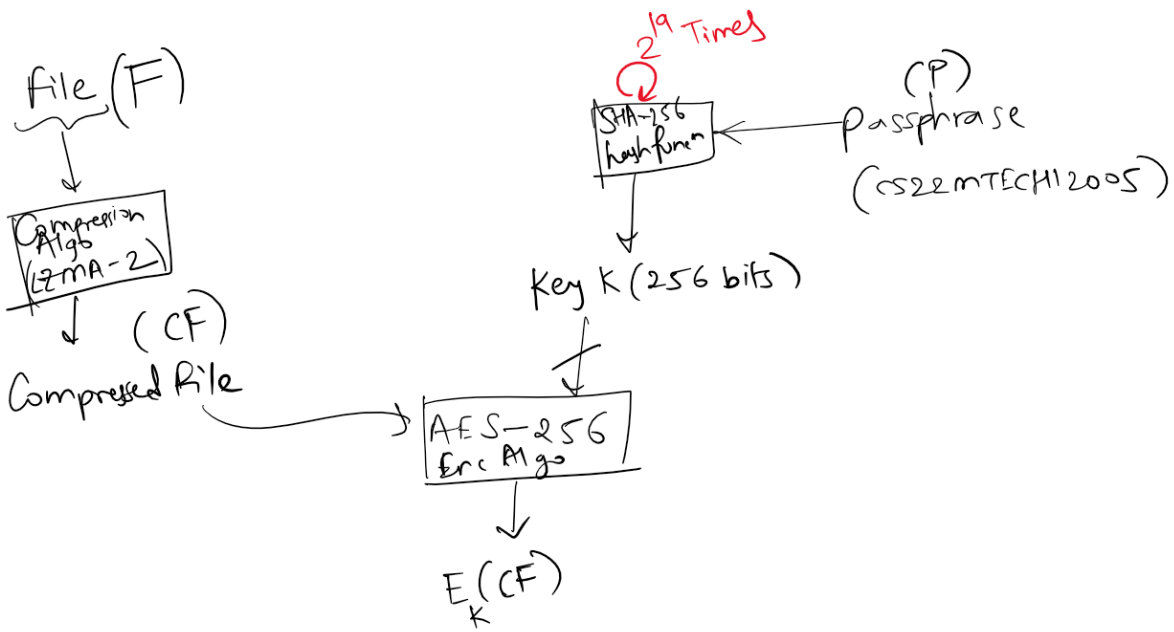
But, even if it gets validated and data exchange starts, it is still useless. Say, Alice tries to do a SSL handshake with Trudy, but it will be encrypted by Bob's public key and only Bob has its private key so Trudy who does not have access to Bob's private key can anyhow not decrypt these messages. So, there is no use or advantage of Trudy using Bob's certificate to spoof him. Moreover, domain name validation is also done, i.e the domain connected to and the domain mentioned in the certificate will not match, thus the connection might not even be set in the first place. Things mentioned above have an implicit assumption of no domain validation explicitly.

3. Consider another scenario in which evil Trudy has used the digital certificate of Bob's website abc.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 abc.com 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?

   DNS cache poisoning is creating a forged entry in the DNS to redirect all traffic to some network which is unsafe. myCertVerifier() will return valid, because it is not changing the scenario of certificates. Trudy is still giving Bob's public key and Bob's certificates. As the certificate aligns well with Bob's public key, it is validated. Only difference from the above scenario is that Trudy here is pretending to be Bob entirely and getting the traffic of Bob's server. One can deny availability by not letting Alice access the services by redirecting the traffic. Trudy can pretend to be a CA if the private key of Bob gets compromised if Bob is a CA and can issue certificates. PKI may collapse in this case, entirely if bob is a root CA.

PART-C

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?



It is possible to handle a variety of data compression, encryption, and pre-processing procedures with the compressed archive file format 7z.

7 zip first compresses the file(s). Here, we are choosing the LZMA2 algorithm. Then we get a compressed file, say CF. The pass phrase is a means for generating the key to the AES-256 encryption algorithm. It is fed to a SHA-256 Hash algorithm which returns a 256 bit key for AES algorithm. AES finally encrypts CF using one of the modes like CBC, and the generated key K to give the encrypted file Ek(CF). To get the file back, one needs the passphrase which will again recalculate the 256 bit key. With this key the Compressed file needs to be decrypted first to get the CF back. After that, a decompression algorithm similar to LZMA2 can be applied to get the File F back.

Passphrase is an important entity here. Brute force attack can be done and generate keys which can be used individually to break the system. For this, **SHA-256 is applied 2^19 times** iteratively in a loop. This does two things : randomizes and more importantly adds a delay even before the compression and encryption begins. This delay is very important and is called **key stretching**. This makes it difficult with the current hardware power to brute force each and every passphrase, it is almost impossible to crack the key this way. The usefulness of this specific type of key stretching is still limited by GPU-based and bespoke hardware assaults, therefore it's still crucial to pick a strong password. There is one more reason to use strong passwords, and it is that 7z does not use salt for AES, although in code it is mentioned in the decryption side, but commented out for now. So, if we have big passwords, that is the same as **adding salt**. Secondly, there is more possibility to hash in the same value if small passwords are used. Passwords are simply tried repeatedly by a password cracker, either by trying every word in a dictionary (a big file with a lot of words) or by trying every combination. Every password can be cracked with enough time. Thus, big passwords along with some randomness should be used to

ensure a high security, this reduces the attack surface and brute force is almost impossible.

Deliverables in GC:

- Certificates used for completing this assignment and a readable PDF Report with name “Asg4-<RollNo>.PDF” compressed and encrypted with AES-256 using open source 7-zip file archiver tool with your RollNo (UPPERCASE) as the password.
    - In your report, 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?

References:

1. https://crt.sh/
2. https://ahrefs.com/blog/most-visited-websites/
3. http://lapo.it/asn1js/#
4. http://phpseclib.sourceforge.net/x509/decoder.php
5. https://www.ssl.com/article/dv-ov-and-ev-certificates/
6. https://www.ccadb.org/
7. DV, OV, IV, and EV Certificates - SSL.com
8. 7-Zip (7-zip.org)

PLAGIARISM STATEMENT <Include it in your report>

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 plagiarised 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 honour violations by other students if I become aware of it.

Name: Tejas Deshmukh (cs22mtech12005)
Date: 29/01/2023
Signature: Tejas