Cryptography and IT-Security

Question 1: Consider the following threats to Web security and describe how each is countered by a particular feature of TLS.

1. Brute-Force Cryptanalytic Attack: An exhaustive search of the key space for a conventional encryption algorithm.

Brute-Force Cryptanalytic Attack:

- TLS (Transport Layer Security) counters brute-force cryptanalytic attacks primarily through the use of strong encryption algorithms with **sufficiently large key sizes**. TLS uses strong encryption algorithms with sufficiently large key sizes, making brute-force attacks impractical.
- The larger the key size, the more possible keys there are, exponentially increasing the time it would take for a brute-force attack to try every possible key.
- For instance, TLS frequently uses AES (Advanced Encryption Standard) keys with widths of 128, 192, or 256 bits. With today's technology, trying to brute-force a 256-bit key is not viable due to the enormous processing time required.

2. Known Plaintext Dictionary Attack: Many messages will contain predictable plaintext, such as the HTTP GET command. An attacker constructs a dictionary containing every possible encryption of the known-plaintext message. When an encrypted message is intercepted, the attacker takes the portion containing the encrypted known plaintext and looks up the ciphertext in the dictionary. The ciphertext should match against an entry that was encrypted with the same secret key. If there are several matches, each of these can be tried against the full ciphertext to determine the right one. This attack is especially affective against small key sizes (e.g., 40-bit keys).

Known Plaintext Dictionary Attack:

- TLS employs **robust encryption algorithms** like AES with big key sizes, dictionary attacks cannot effectively penetrate the key space. The handshake process generates **unique session keys** for every TLS session, so even if the same plaintext is provided more than once, the encryption will be different each time.
- **Initialization Vectors** (IVs) are used by TLS implementations to guarantee that distinct ciphertexts will be generated from the same plaintext in each session.
- TLS can use **Perfect forward secrecy** (PFS) to generate a new key for each session, so even if one key is compromised, past communications remain secure. These features combined make the known plaintext attack impractical against TLS protected communications.

3. Password Sni ng: Passwords in HTTP or other application traffic are eavesdropped.

Password Sniffing:

- TLS counters password sniffing by encrypting the data transmitted between the client and server. When a user enters a password on a website secured with TLS, the **password is encrypted** before it is sent across the network. This means that even if an attacker is able to intercept the traffic (from sniffing), the encrypted data should be unreadable and unusable without the corresponding decryption key.

- Moreover, TLS offers integrity checks to guarantee that the information has not been altered, hence enhancing the security of the password during transmission

4. Man-in-the-Middle Attack: An attacker interposes during key exchange, acting as the client to the server and as the server to the client.

Man-in-the-Middle Attack:

- TLS counters Man-in-the-Middle (MitM) attacks using a combination of **asymmetric cryptography** during the key exchange process and **certificate-based authentication**.
- The server gives the client its public key in its digital certificate during the TLS handshake. In order to create a secure channel, the client utilizes this public key to encrypt data that can only be decrypted by the server using its private key.
- TLS uses reputable certificate authorities (CAs) to provide servers with digital certificates. The clients are configured to trust certificates issued by these CAs. When the server presents its certificate, the client can verify its authenticity by checking the digital signature against the CA's public key. In order to successfully prevent an attacker, they would require a legitimate certificate for the client's domain that has been signed by an authorized CA, which is hard to get illegally.

Through the requirement that all certificates be signed by a reliable CA, TLS significantly reduces the difficulty for a MitM attacker to pretend as the intended server or client.

Question 2: In SSL and TLS, why is there a separate Change Cipher Spec Protocol rather than including a change cipher spec message in the Handshake Protocol? Discuss your answer.

In SSL and TLS, the Change Cipher Spec Protocol is separate from the Handshake Protocol for a few reasons:

- **Simplicity and Clarity**: The Handshake Protocol is kept more focused and straightforward by keeping the Change Cipher Spec apart from it. The Change Cipher Spec serves as a signal to switch to the negotiated parameters, whereas the Handshake Protocol deals with the negotiation of cryptographic parameters.
- **Layering and Modularity**: The design follows a layered approach, with each protocol serving a specific purpose, by keeping them apart. The protocol is simpler to understand, use, update, and maybe modify as a result of its modularity.
- **Security**: It is easier to define the security boundaries within the communication when the protocols are separated. Overall security is improved since it is simpler to confirm and guarantee that the Change Cipher Spec happens at the appropriate moment in the protocol flow.
- **Immediate effect**: The Change Cipher Spec message acts as a trigger that causes the immediate switch to the newly negotiated security parameters. It's a simple signal, essentially saying "start using the new security settings now". By having a separate protocol for this, it ensures that this message is distinct and cannot be confused with the other handshake messages.

In general, this division makes the SSL/TLS protocol more efficient, secure, and well-organized.

Question 3: Assume a scenario, in CSC318/M18, we have four students Carol, Jack, May, Andrew, and an instructor Kelly. Consider, each own a copy of cryptography textbook, but May does not. Also suppose the instructor can ask students questions and students can answer; at the same time, students can also ask instructor, to which both instructor and other students can answer; however, a student cannot ask other students questions in class. You need to draw an access control matrix using HRU approach to model the above rules.

In the Harrison-Ruzzo-Ullman (HRU) model, an access control matrix is used to represent the rights of subjects (like users or processes) over objects (like files or resources). For the scenario in CSC318/M18, the subjects are the students (Carol, Jack, May, Andrew) and the instructor (Kelly), and the objects are the cryptography textbooks and the ability to ask questions.

Here's how the access control matrix might look:

Subjects/ Objects	**Textbook (Carol)**	**Textbook (Jack)**	**Textbook (May)**	**Textbook (Andrew)**	**Textbook (Kelly)**	**Ask Instructor**	**Ask Student**	**Answer Instructor**	**Answer Student**
Carol	Own, Read					Yes		Yes	Yes
Jack		Own, Read				Yes		Yes	Yes
May			None			Yes		Yes	Yes
Andrew				Own, Read		Yes		Yes	Yes
Kelly (Instructor)					Own, Read		Yes		Yes

- **Textbook Columns**: Students and the instructor have "Own, Read" access to their own textbooks. May has "None" since she doesn't own a textbook.
- **Ask Instructor**: All students can ask the instructor questions.
- **Ask Student**: Only the instructor has this right. Students cannot ask each other questions.
- **Answer Instructor**: All subjects (students) can answer questions posed by the instructor.
- **Answer Student**: Both the instructor and other students can answer when a student asks the instructor a question, so 'Yes' for all. However, students cannot answer each other directly, so there is no separate object for student-to-student communication.

This matrix models the access control rules as described for the class scenario.

Question 4: Assume that passwords are limited to the use of the 95 printable ASCII char acters and that all passwords are 10 characters in length. Assume a password cracker with an encryption rate of 6.4 million encryptions per second. How long will it take to test exhaustively all possible passwords on a UNIX system?

To exhaustively test all possible passwords with the given parameters, the calculations are as follows:

- As the passwords are limited to the use of the 95 printable ASCII characters and that all passwords are 10 characters in length. Total number of possible passwords: $95^{10} = 59{,}873{,}693{,}923{,}837{,}890{,}625$

- Assuming a password cracker with an encryption rate of 6.4 million encryptions per second as given in the question. Time required to crack the password in seconds is.

  Time in seconds: $\frac{95^{10}}{6.4 \times 10^{6}} \approx 9{,}355{,}264{,}675{,}599.672\ seconds$

- Time in hours: $\frac{9{,}355{,}264{,}675{,}599.672}{3600} \approx 2{,}598{,}684{,}632.111\ hours$

- Time in days: $\frac{2{,}598{,}684{,}632.111}{24} \approx 108{,}278{,}526.338\ days$

- Time in years: $\frac{2{,}598{,}684{,}632.111\ hours}{365} \approx 296{,}653.497\ years$

  This calculation demonstrates the significant amount of time required for a brute-force attack on a system with such password constraints, illustrating the strength of using long passwords with a large character set. Therefore, it would take approximately 296,653.497 years to exhaustively test all possible 10-character passwords composed of the 95 printable ASCII characters at an encryption rate of 6.4 million encryptions per second.

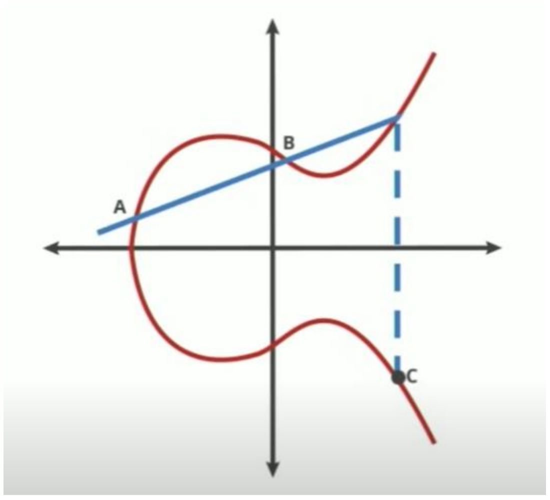
Question 5: For the final question of this coursework, you are expected to explore and produce a (fairly short) technical report (max of 3 pages) for advanced cryptographic topics. In particular, you can choose to research into one of the following cryptographic techniques: 1. El Gamal. 2. Elliptic Curve Cryptography. 3. Homomorphic Encryption.

Elliptic Curve Cryptography (ECC)

An introduction to the cryptographic approach:

- Background: Cryptography is the practice and study of techniques for secure communication in the presence of third parties. Traditional cryptographic systems, like RSA, rely on the mathematical properties of large prime numbers. As computational power has increased, these systems require larger keys to maintain security, leading to inefficiencies.
- Principle of ECC: ECC was introduced in the mid-1980s by mathematicians Neal Koblitz and Victor S. Miller. It represents a radical departure from traditional cryptographic methods, utilizing the algebraic structure of elliptic curves over finite fields. ECC offers the same level of security as traditional cryptographic systems but with shorter key lengths, leading to faster computations, lower power consumption, and reduced storage and transmission requirements. The strength of ECC lies in the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP).
- Advantages of ECC: Efficiency, Speed, Lower power and bandwidth requirement, and security.
- Applications of ECC: Secure communications, Cryptocurrencies, mobile security, and IoT security.

In conclusion, Elliptical Curve Cryptography represents a significant advancement in the field of cryptography, offering a secure and efficient alternative to traditional methods. Its ability to provide robust security with smaller key sizes makes it particularly well-suited to the needs of modern digital security, from internet communication to mobile and IoT applications [2].



Technical details into how and why the approach works

<u>Technical Foundations of ECC</u>

- Mathematical Background: An elliptic curve is the set of points that satisfy a specific mathematical equation [3]. The equation for an elliptic curve looks something like this: $y^2 = x^3 + ax + b$, where *a* and *b* are constants. The curve is plotted over a finite field (either prime or binary), providing a finite number of points on the curve.

- Key Concepts in ECC:

Point Addition and Doubling: The unique aspect of elliptic curves is the definition of an addition operation for any two points on the curve. This operation follows specific geometric rules. The result of adding a point to itself (doubling) is also defined.

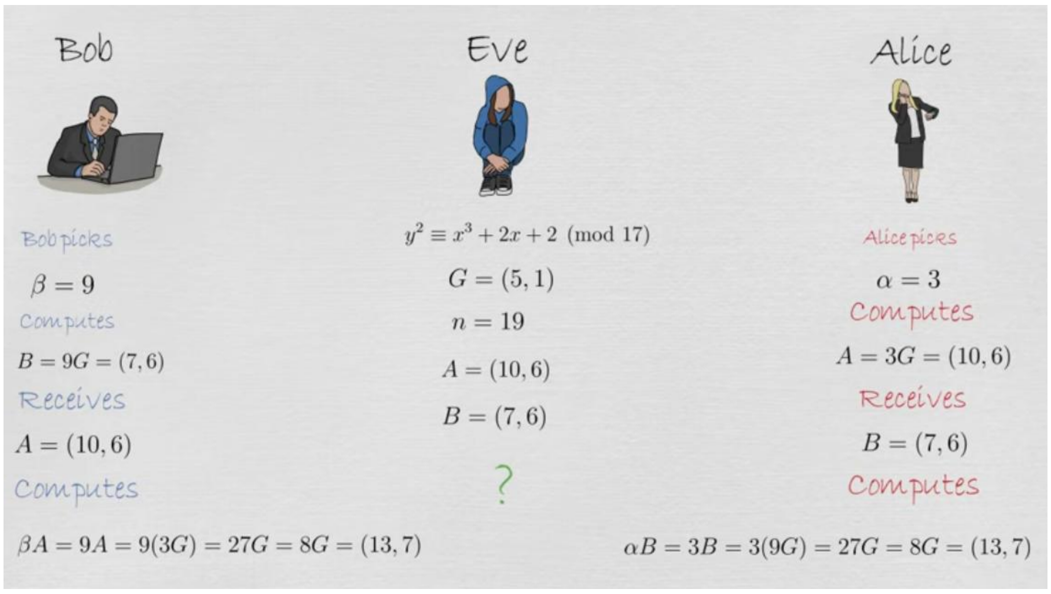
Scalar Multiplication: The fundamental operation in ECC is scalar multiplication, which is repeatedly adding a point to itself. Given a point *P* on the curve, and a scalar *k*, the scalar multiplication is finding another point *Q* which is $k \times P$.

- Discrete Logarithm Problem (DLP): The security of ECC hinges on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). Given two points *P* and *Q*, where $Q=kP$ (and *k* is unknown), ECDLP asks to find *k*. This problem is computationally hard, making it unfeasible to solve in practical timeframes, thereby ensuring the security of ECC.

<u>Why ECC Works</u>: It works due to its Computational Complexity, Computational Infeasibility of Elliptic Curve Discrete Logarithm Problem (ECDLP), Small Key Size and High Security, and Resistance to Known Cryptanalytic Attacks.

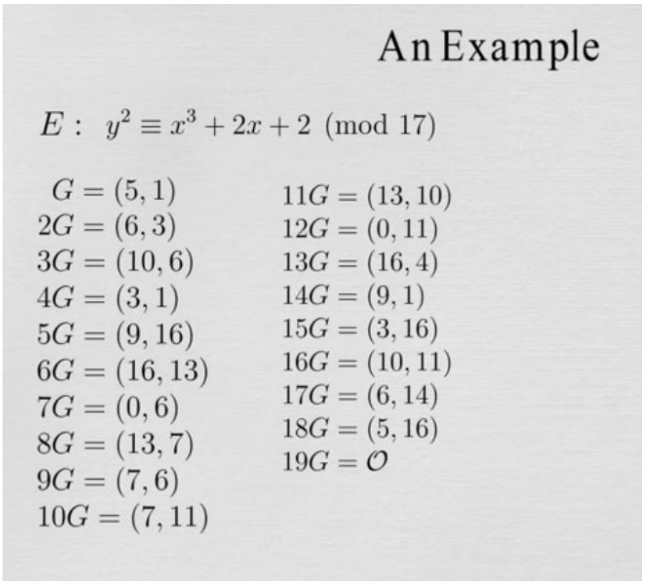
<u>Practical Implementation of ECC:</u> It includes Key generation, Encryption and Decryption, and Digital signatures.

An example of encryption and decryption of a sample message



Here's a scenario illustrated [1].

- Elliptic Curve Equation: $y^2 = x^3 + 2x + 2(mod 17)$
- Generator Point (G): (5, 1)
- Prime Number (n): 19 (This might represent the order of the curve, which is the number of points on the curve, but it's not used in the operations shown in the image.)
- Participants: Alice, Bob, and an eavesdropper, Eve.



An Example

$E: \ y^2 \equiv x^3 + 2x + 2 \pmod{17}$

$G = (5,1)$	$11G = (13,10)$
$2G = (6,3)$	$12G = (0,11)$
$3G = (10,6)$	$13G = (16,4)$
$4G = (3,1)$	$14G = (9,1)$
$5G = (9,16)$	$15G = (3,16)$
$6G = (16,13)$	$16G = (10,11)$
$7G = (0,6)$	$17G = (6,14)$
$8G = (13,7)$	$18G = (5,16)$
$9G = (7,6)$	$19G = \mathcal{O}$
$10G = (7,11)$	

The process follows these steps:

1. Private Keys: Alice picks a private key *a*=3. Bob picks a private key *β*=9.
2. Public Keys: Alice computes her public key *A*=3*G*= (10,6) by performing scalar multiplication of the generator point *G* by her private key *a*. Bob computes his public key *B*=9*G*= (7,6) by performing scalar multiplication of the generator point *G* by his private key *β*.
3. Exchange of Public Keys: Alice and Bob exchange their public keys over a public channel. Eve can intercept this exchange, but without the private keys, she cannot compute the shared secret.
4. Shared Secret Computation: Alice receives Bob's public key *B* and computes the shared secret by multiplying it by her private key: *aB*=3*B*=3(9*G*) =27*G*. Since the curve is defined over a finite field, the actual computation will take into account the modulo of the prime number defining the field. Bob receives Alice's public key *A* and computes the shared secret by multiplying it by his private key: *βA*=9*A*=9(3*G*) =27*G*. Which complies to 8G = (13,7).

5. The result of these operations is that both Alice and Bob arrive at the same point, which is their shared secret. The shared secret is used to derive a key for encryption. The security of this exchange lies in the difficulty of the elliptic curve discrete logarithm problem, as an eavesdropper would need to determine the private keys associated with the public keys, which is computationally infeasible with sufficiently large parameters. Alice and Bob have the shared secret S which is a point on the curve, say (xS,yS). They use xS to derive the symmetric key k. For simplicity, let's assume $k=xS$. Alice encrypts "HELLO" with k using AES (just as an example) to get ciphertext c. Alice sends c to Bob. Bob decrypts c using k to get back "HELLO".

A comparison of the approach to other similar techniques, such as RSA and DH [5] [6].

Feature	ECC	RSA	DH
Key Size	Small keys for high security	Large keys for high security	Key size varies, typically large for security
Computation Speed	Fast operations due to smaller key size	Slower operations with large keys	Moderate; depends on key size
Bandwidth Usage	Efficient; requires less bandwidth	Inefficient; requires more bandwidth	Inefficient; requires more bandwidth
Power Consumption	Low; ideal for mobile devices	High; not ideal for power-sensitive devices	Moderate to high; less ideal for power-sensitive devices
Security Basis	Hardness of ECDLP	Hardness of integer factorization	Based on the discrete logarithm problem
Quantum Resistance	More resistant to quantum attacks	Susceptible to quantum attacks	Susceptible to quantum attacks
Adoption	Increasingly adopted in modern systems	Well-established, widely used	Well-established in key exchange protocols
Use Cases	Mobile security, IoT, smart cards	Digital certificates, secure email	Secure key exchange, VPNs, HTTPS
Key Exchange	Can be used for key exchange	Not directly used for key exchange	Specifically used for secure key exchange
Public Key Use	Yes	Yes	No; used for shared secret derivation

References

1. https://www.youtube.com/watch?v=F3zzNa42-tQ
2. https://www.youtube.com/watch?v=gAtBM06xwaw
3. https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/
4. https://en.wikipedia.org/wiki/Elliptic-curve_cryptography

5. https://www.sectigo.com/resource-library/rsa-vs-dsa-vs-ecc-encryption#:~:text=The%20biggest%20difference%20between%20ECC,key%20of%20the%20same%20size.

6. https://ieeexplore.ieee.org/document/8551161