Final Assignment

Murcaduzzaman Asha
IT-21014

Q.1. How does shor's algorithm threaten the security of RSA and Ellipic Curve cryptography (ECC), and what are the potential consequences for current digital infrastructure?

Answer:

Shor's algorithm threatens RSA and ECC because it can use a quantam computer to quickly solve the hard math problem (factoring large numbers for RSA, solving discrete logarithms for ECC) that keep them secure. If broken, attackers could decrypt sensitive data, forge digital signatures, and compromise most

of today's digital infrastructure, including HTTPS, online banking, and secure communications.

Q.2. Discuss the role of quantum key distribution (QKD) in future crytographic system. How does it differ from classical public-key encryption?

Answer:

Quantum key Distribution (QKD) is a method of securely sharing encryption keys using quantum physics. It's special because if anyone tries to intercept the key, the quantum state change and both parties can detect the eavesdropper. Unlike classical public-key encryption (which relies on mathematical problems like factoring large numbers), QKD's

Security is based on the laws of physics, making it resistant to future quantum computer attacks.

Q.5 What are the main differences between lattice-based cryptography and traditional number-theoretic approaches like RSA. particularly in the context of quantum resistance?

Answer:

The main differences between lattice-based cryptography and traditional number-theoretic approaches like RSA

Subject	RSA (Traditional)	Lattice-based
Math based	Integer factorization/ Discrete logarithm	Lattice problems (Shortest vector problem, Learning with Errors
Quantum Security	Weak - Vulnerable to Shor's algorithm on quantum computers.	Strong - no known efficient quantum attack exists.

Subject	RSA (Traditional)	Lattice-based (Cryptography)
Key size	very large keys (thousands of bits). gets even bigger for quantum-safe versions	Relatively smaller keys. still secure against quantum attacks
performance	Encryption fast, key generation and decryption slower	key generation and/encryption/decryption generally faster. but message sizes may be larger
Usage	widely used in current internet security (HTTPS, email, etc)	A leading candidate for post-quantum cryptography standards

This is the main differences.

Q.4. Develop a python-based PRNG that uses the current System time and a custom seed value. Write complete program and corresponding output.

Answer:

Simple python PRNG using Time an seed.

Code:

```
import time
def simple_prng(seed):
    t = int(time.time())
    num = seed + t
    while True:
        num = (num*3+7)%100
        yield num
seed = int(input("Enter a number as seed:"))
gen = simple_prng(seed)
print("Here are 5 random numbers:")
for _ in range(5):
    print(next(gen))
```

Output:

```
Enter a number as seed : 10
Here are 5 random numbers:
77
38 70
21 17
```

Q.5. Explain the sieve of Eratosthenes algorithm and use it to find all prime numbers less than 50. How does its complexity compare to trial division?

Answer:

The Sieve of Eratosthenes is an efficient algorithm to find all prime numbers up to a given number n.

It works by repeatedly marking the multiples of each prime number starting from 2. The numbers that remain unmarked are primes.

Example: Find primes less than 50

Step-by-Step:

1. Start 2, 3, 4, 5, 6, 7, 8, 9, ..., 49
2. Remove multiples of 2 → 4, 6, 8, 10, ..., 48
3. Next prime 3 → remove multiple: 9, 12, 15, 18, ..., 48
4. Next prime 5 → remove multiples: 25, 30, 35, 40, 45

5. Next prime 7 → remove multiples: 49

6. Remaining numbers are primes.

prime less than 50:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

Time complexity:

* Sieve of Eratosthenes : $O(n \log \log n)$
  - faster for large n

* Trial Division: $O(n\sqrt{n})$ - much slower for large n.

Q. 6.

State and explain the necessary and sufficient conditions for a composite number to be a Carmichael number. Then verify whether the number $n = 561$ and $n = 1105$ and $n = 1729$ are Carmichael number?

Answer:

A Carmichael number is a special

composite number that behaves like a prime in Fermat's little theorem test.

Necessary & Sufficient Condition:
(Korselt's criterion)

A composite number n is Carmichael if:

1. n has no repeated prime factors (called square-free)
2. For every prime factor p of n, $p-1$ divides $n-1$.

<u>Example:</u> check 561

* Factorization: $561 = 3 \times 11 \times 17$
* Calculate $n-1$: 560
* check divisibility:
  - $3-1 = 2$, 2 divides 560 → OK
  - $11-1 = 10$, 10 divides 560 → OK
  - $17-1 = 16$, 16 divides 560 → OK

Check 1105

- $1105 = 5 \times 13 \times 17$, square-free
- $n-1 = 1104$
- check:

  $5-1=4$ divides $1104 \rightarrow$ OK

  $13-1 = 12$ divides $1104 \rightarrow$ OK

  $17-1 = 16$ divides $1104 \rightarrow$ OK

Check 1729:

- $1729 = 7 \times 13 \times 19$, square-free
- $n-1 = 1728$
- check:

  $7-1 = 6$ divides $1728 \rightarrow$ OK

  $13-1 = 12$ divides $1728 \rightarrow$ OK

  $19-1 = 18$ divides $1728 \rightarrow$ OK

7. Determine whether the following are valid algebraic structures and justify your answer:

- Is the set Z_{11} with operations $(+, \cdot)$ a ring?
- Are the sets $(Z_{37}, +)$ and $(Z_{35}, \times)$ Abelian groups?

Answer:

Yes, In fact Z_{11} is a commutative ring with unity and moreover a field.

Justification:

- $(Z_{11}, +)$ is an abelian group: closure, associatively, identity 0, inverses $(-a \equiv 11-a)$, and commutativity hold (addition mod 11)
- Multiplication mod 11 is closed and associative, and distributes over addition.
- There is a multiplicative identity $1 \in Z_{11}$.

- Because 11 is prime, every nonzero element $a \in \mathbb{Z}_{11}$ has a multiplicative inverse mod 11. Thus all ring axioms hold and every nonzero element is invertible $\rightarrow \mathbb{Z}_{11}$ is a field.

Yes, $(\mathbb{Z}_{37}, +)$ is an Abelian group.

<u>Justification:</u>

- Set $\mathbb{Z}_{37} = \{0, 1, \cdots, 36\}$ with addition modulo 37.
- closure, associativity and commutativity follow from integer addition.
- Identity is 0. For each a, additive inverse is $37 - a$.
- Thus $(\mathbb{Z}_{37}, +)$ satisfies all group axioms and is abelian.

No, $(Z_{35}, \times)$ is not a group.

<u>Justification:</u>

- Although multiplication mod 35 is associative and has identity 1, not every element has a multiplicative inverse in Z_{35}.

- Example: $5 \in Z_{35}$. $\gcd(5, 35) = 5 > 1$.

There is no x with $5x \equiv 1 \pmod{35}$ because any product $5x$ is divisible by 5 and cannot be congruent to 1. Hence 5 has no inverse.

- Therefore the inverse axiom fails and $(Z_{35}, \times)$ is not a group; so it is not an abelian group.

Question 8: What is the remainder when -52 is reduced modulo 31?

Answer:

We want r with $0 \leq r < 31$ and $-52 \equiv r \pmod{31}$.

Compute:

$$-52 + 2 \cdot 31 = -52 + 62 = 10$$

So, $-52 \equiv 10 \pmod{31}$.

Alternate: $-52 \equiv -52 + 31 = -21 \equiv -21 + 31 = 10$

So, the remainder is 10

Question 9: Determine the multiplicative inverse of 7 mod 26, if it exists. (use extended Euclidean algorithm)

Answer:

We solve $7x \equiv 1 \pmod{26}$ or find integers x, y with $7x + 26y = 1$

Use the Euclidean algorithm:

$$26 = 3 \cdot 7 + 5,$$
$$7 = 1 \cdot 5 + 2,$$
$$5 = 2 \cdot 2 + 1,$$
$$2 = 2 \cdot 1 + 0.$$

Back-substitute to express 1 as a linear combination:

$$1 = 5 - 2 \cdot 2$$
$$= 5 - 2(7 - 1 \cdot 5) = 3 \cdot 5 - 2 \cdot 7$$
$$= 3(26 - 3 \cdot 7) - 2 \cdot 7 = 3 \cdot 26 - 11 \cdot 7$$

Thus $-11 \cdot 7 + 3 \cdot 26 = 1$. Therefore $x \equiv -11 \pmod{26}$

$$x \equiv -11 \equiv 15 \pmod{26}.$$

The multiplicative inverse of 7 modulo 26 is 15.

Question : 10

Evaluate $(-8 \times 5) \bmod 17$, and explain how to simplify negative modular multiplication.

Answer :

Step 1 – Multiply the numbers :

$$-8 \times 5 = -40$$

Step 2 – Reduce modulo 17 :

We find the equivalent positive remainder:

$$-40 + 3 \times 17 = -40 + 51 = 11$$

$$-40 \equiv 11 \pmod{17}$$

$$(-8 \times 5) \bmod 17 = 11$$

Explain at simplification method:

Negative numbers in modular arithmetic can be converted to their positive equivalent before multiplying.

$$-8 \equiv 9 \pmod{17}$$

$$9 \times 5 = 45$$

Reduce 45 modulo 17:

$$45 - 2 \times 17 = 45 - 34 = 11$$

This match the privious result.

$\therefore$ Final answer — 11

Question 11 : State and prove Bezout's Theorem use it to find the multiplicative ~~modular~~ ~~multiplication~~ inverse of 97 modulo 385.

Answer :

For integers a and b not both zero, there exist integers x, y such that

$$ax + by = \gcd(a, b)$$

In particular, if $\gcd(a,b) = 1$ there are integers x, y with $ax + by = 1$; then x is the multiplicative inverse of a modulo b.

Proof :

Let, $d = \gcd(a, b)$.

- By the Euclidean algorithm, d divides both a and b.
- The set of all integers of the form $ax + by$ is closed under addition and

substraction.

- Let m be the smallest positive number in this set. Then m divides both a and b.
- Hence $m = d$, and there exist integers x, y such that:

$$ax + by = d$$

Finding the multiplicative inverse of 97 modulo 385:

We want x such that:

$$97x \equiv 1 \pmod{385}$$

This is equivalent to:

$$97x + 385y = 1$$

We use the Extended Euclidean Algorithm

1. Euclidean algorithm:

$$385 = 97 \cdot 3 + 94$$
$$97 = 94 \cdot 1 + 3$$
$$94 = 3 \cdot 31 + 1$$
$$3 = 1 \cdot 3 + 0$$

So, $\gcd(97, 385) = 1$.

2. Back substitution:

From $94 = 3 \cdot 31 + 1$:

$$1 = 94 - 3 \cdot 31$$

From $97 = 94 \cdot 1 + 3$:

$$3 = 97 - 94$$

substitude into 1:

$$1 = 94 - (97 - 94) \cdot 31$$
$$1 = 94 - 97 \cdot 31 + 94 \cdot 31$$
$$1 = 94(1 + 31) - 97 \cdot 31$$
$$1 = 94 \cdot 32 - 97 \cdot 31$$

From $385 = 97 \cdot 3 + 94$:

$$94 = 385 - 97 \cdot 3$$

Substitute:

$$1 = (385 - 97 \cdot 3) \cdot 32 - 97 \cdot 31$$

$$1 = 385 \cdot 32 - 97 \cdot 96 - 97 \cdot 31$$

$$1 = 385 \cdot 32 - 97 \cdot (96 + 31)$$

$$1 = 385 \cdot 32 - 97 \cdot 127$$

3. Conclusion:

We have:

$$1 = (-127) \cdot 97 + 32 \cdot 385$$

Therefore, $x \equiv -127 \pmod{385}$.

Since we want the positive inverse:-

$$-127 \equiv 385 - 127 = 258 \pmod{385}$$

The multiplicative inverse of 97 mod 385 is 258.

Question 12: Sing Bezout's identity, prove that the equation $ax+by=\gcd(a,b)$ has integer solutions. Find x such that $43x \equiv 1 \mod 240$.

Answer:

Bezout Identity Statement:

For any integers a and b, there exist integers x and y such that:

$$ax+by=\gcd(a,b)$$

Proof:

1. Let $d=\gcd(a,b)$ By definition, d devides, d divides both a and b, so $a=da'$ and $b=db'$, where a', b' are integers with $\gcd(a',b')=1$.

2. Since a' and b' are comprime, threre exists integers x_0, y_0 such that:

$$a'x_0+b'y_0=1.$$

3. Multipdying through by d gives:

$$a(dx_0)+b(dy_0)=d$$

Thus $x=x_0$ and $y=y_0$ are integer solutions to:

$$ax+by=\gcd(a,b)$$

Hence proved.

<u>Find x such that $43x \equiv 1 \pmod{240}$.</u>

This asks for the multiplicative inverse of 43 modulo 240. We solve the Diophantine equation:

$$43x-240y=1$$

by the Extended Euclidean Algorithm.

Compute gcd chain (Euclidean algorithm):

$$240=5\cdot 43+25,$$
$$43=1\cdot 25+18,$$
$$25=1\cdot 18+7,$$
$$18=2\cdot 7+4,$$
$$7=1\cdot 4+3,$$
$$4=1\cdot 3+1,$$

$3 = 3 \cdot 1 + 0.$

So gcd $(43, 240) = 1$ and the inverse exists. Now Back-substitute to express 1 as a linear combination of 43 and 240.

Back-substitution:

$$
\begin{aligned}
1 &= 4 - 1 \cdot 3 \\
&= 4 - 1 \cdot (7 - 1 \cdot 4) = 2 \cdot 4 - 1 \cdot 7 \\
&= 2 \cdot (18 - 2 \cdot 7) - 1 \cdot 7 = 2 \cdot 18 - 5 \cdot 7 \\
&= 2 \cdot 18 - 5 \cdot (25 - 1 \cdot 18) = 7 \cdot 18 = 5 \cdot 25 \\
&= 7 \cdot (43 - 1 \cdot 25) - 5 \cdot 25 = 7 \cdot 43 - 12 \cdot 25 \\
&= 7 \cdot 43 - 12 (240 - 5 \cdot 43) \\
&= 7 \cdot 43 - 12 \cdot 240 + 60 \cdot 43 \\
&= 67 \cdot 43 - 12 \cdot 240
\end{aligned}
$$

Thus

$$43 \cdot 67 - 240 \cdot 12 = 1,$$

So, $x = 67$ is a solution.

$43^{-1} \equiv 67 \pmod{240}$ or $43 \cdot 67 \equiv 1 \pmod{240}$.

Question 13 : Prove Fermat's Little Theorem and explain how it is used to test for primality. Is 561 a prime number based on this test?

Evaluate 5^{123} mod 175 using Fermat's Little Theorem. Show all steps.

Answer:

Fermat's Little Theorem (FLT) - statement:

If p is a prime number and a is any integer with gcd (a, p) = 1, then

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof:

Consider the nonzero residues modulo p: 1, 2, ... , p−1. Multiply each by a (with gcd (a, p) = 1)

The set $a \cdot 1, a \cdot 2, \ldots a \cdot (p-1)$

is a permutation of 1, 2, ... p−1 modulo p.

Taking the product of all elements in both sets and reducing mod p gives –

$$a^{p-1} \cdot (1 \cdot 2 \cdots (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}$$

Cancelling the nonzero product $1 \cdot 2 \cdots (p-1)$ (which is invertible mod p) yields $a^{p-1} \equiv 1 \pmod{p}$

Primality Test Using FLT:-

Choose a with $1 < a < p$ and $\gcd(a,p) = 1$

If $a^{p-1} \not\equiv 1 \pmod{p} \Rightarrow p$ is composite.

If it is 1, p may be prime — but there are composites (Carmichael numbers) that also pass.

Is 561 prime?

Take $a = 2$, $\gcd(2, 561) = 1$;

$$561 = 3 \cdot 11 \cdot 17$$

By chinese Remainder theorem and FLT:

$$2^{560} \equiv 1 \pmod{3},\ 1 \pmod{11},\ 1 \pmod{17}$$

so, $2^{560} \equiv 1 \pmod{561}$ — 561 passes FLT but is not prime. (It's a carmichael number)

Evaluate 5^{123} mod 175

1. Factor the modulus: $175 = 25 \cdot 7$. work modulo 25 and 7 and combine by CRT.

2. Mod 25: $5^2 = 25 \Rightarrow 5^n \equiv 0 \pmod{25}$ for all $n \geq 2$ since $123 \geq 2$

$$5^{123} \equiv 0 \pmod{25}$$

3. Mod 7: $\gcd(5,7) = 1$ so by Fermat $5^6 \equiv 1 \pmod{7}$. Reduce the exponent:

$$123 \equiv 123 - 6 \cdot 20 = 3 \pmod{6},$$

$$5^{123} \equiv 5^3 = 125 \equiv 6 \pmod{7}.$$

4. combine by CRT: find x with

$$x \equiv 0 \pmod{25}, \quad x \equiv 6 \pmod{7}.$$

Check multiples of 25: $25 \cdot 5 = 125 \equiv 6 \pmod{7}$

Thus the solution modulo 175 is

$$5^{123} \equiv 125 \pmod{175}.$$

Fermat's Little Theorem cannot be applied directly modulo 175 because 5 and 175 are not coprime - that's why we used CRT.

Question 14: State and prove the chinese Remainder Theorem. Then solve the following system of congruences:

$$x \equiv 2 \pmod 3, \; x \equiv 3 \pmod 5,$$
$$x \equiv 2 \pmod 7$$

Answer:-

Statement: The Chinese Remainder Theorem states that a system of linear congruences with pairwise coprime moduli has a unique solution modulo the product of the moduli.

If $m_1, m_2 \cdots m_k$ are ~~compairwise~~ pairwise coprime positive integers and $a_1, \cdots a_k$ are any integers, then the system

$$x \equiv a_i \pmod{m_i} \; (i = 1, \cdots k)$$

has a unique solution modulo $M = m_1 m_2 \cdots m_k$

Proof (constructive):

Let $M = \prod_{i=1}^{k} m_i$ and $M_i = M/m_i$. Since $\gcd(M_i, m_i) = 1$, there exists an inverse y_i such that $M_i y_i \equiv 1 \pmod{m_i}$ Then

$$x = \sum_{i=1}^{k} a_i M_i y_i$$

satisfies $x \equiv a_j \pmod{m_j}$ for each j (because for $i \neq j$, M_i is divisible by m_j). Uniqueness modulo M follows because any two solutions differ by a multiple of all m_i, hence by a multiple of M.

Solve the system:

Start with $x \equiv 2 \pmod 3$. put $x = 2+3k$

Plug into mod 5:

$$2+3k \equiv 3 \pmod 5 \Rightarrow 3k \equiv 1 \pmod 5$$

Inverse of 3 mod 5 is 2 (since $3 \cdot 2 = 6 \equiv 1$)s

$$k \equiv 2 \pmod 5 \Rightarrow k = 2 + 5t.$$

Thus

$$x = 2 + 3k = 2 + 3(2 + 5t) = 8 + 15t.$$

Now impose mod 7:

$$8 + 15t \equiv 2 \pmod 7 \Rightarrow 15t \equiv 6 \equiv 1 \pmod{7}$$

Since $15 \equiv 1 \pmod 7$, this gives $t \equiv 1 \pmod{7}$

So. $t = 1 + 7s$. Then

$$x = 8 + 15(1 + 7s) = 8 + 15 + 105s = 23 + 105s.$$

Therefore the solution modulo $3 \cdot 5 \cdot 7 = 105$

$$\boxed{x \equiv 23 \pmod{105}.}$$

Question 15: Briefly explain the CIA triad in information security. How does each component contribute to building a secure system?

Answer:

The CIA triad stands for Confidentiality, Integrity and Availability. It's a fundamental for information security.

- Confidentiality:
  - Ensures data is disclosed only to authorized parties.
  - Mechanisms: encryption, access controls, authentication, secure channels.
  - Contribution: Prevents data leaks, protects privacy and secrets.
- Integrity:
  - Ensures data is accurate and has not been tampered with.

- Mechanisms : Cryptographic hashes, digital signatures, MACs, checksums, version control.

- Contribution : detects and prevents unauthorized modification, supports trust in data and transactions.

- Availability :

  - Ensures authorized users can access data and services when needed.

  - Mechanisms : redundancy, backups, DDoS protection, fault tolerance, monitoring.

  - Contribution : keeps systems usable and resilient; prevents service disruption which could harm business ~~together~~ operations.

All three are required together; strong confidentiality with no availability is useless; availability without integrity/confidentiality is risky. Good security design balances CIA based on system requirements.

Question 16: How does steganography differ from cryptography in the context of information security, and what are common techniques used for hiding data in digital media?

Answer:-

Difference: Steganography vs Cryptography

- Cryptography transforms plaintext into an unreadable form (ciphertext) so that the content is hidden (confidential), but the existence of the message is usually obvious. Goal: Unreadability without

the key.

- Steganography hides the existence of the message by embedding it into another innocuous object (image, audio, video, text)

Goal : secrecy of existence (covert communication).

They can be used together. First encrypt the secret, then hide the ciphertext with steganography.

Common steganography techniques (digital media)

1. LSB (Least Significant Bit) substitution (image/audio) : replace least significant bits of pixels/samples with message bits. Simple and high capacity but vulnerable to processing/noise.

2. Masking & filtering (images): hide data in perceptually significant areas (watermarking style), robust against some image processing

3. Transform-domain methods: embedded data in frequency coefficients. More robust to compression and common edits.

4. Spread-spectrum steganography: Spread message bits across many samples, low detectability and robust.

5. Protocol/metadata hiding: Store message in file metadata, unused header fields, or network protocols fields.

6. Text steganography: Using whitespace, synonyms, or formating changes.

Question 17 : What are the key difference between phishing, malware, and denial-of-service (DoS) attacks in terms of their method and impact on system security?

Answer :

Phishing :

- Method : Social engineering - attackers send deceptive emails/messages or create fake websites to trick users into revealing credentials, personal data, or clicking malicious links.

- Goal/Impact : Credential theft, identity theft, initial foothold in networks.

- Defences : User education, email filtering, multi-factor authentication (MFA), URL/websites scanners, anti phishing policies.

2. Malware

- **Method:** Software (viruses, worms, trojans, ransomware, spyware) delivered via email attachments, drive-by downloads, infected devices, or malicious installers.

- **Goal/Impact:** Data theft, destruction, system compromise, encryption for ransom establishing persistent backdoors.

- **Defences:** Endpoint protection/AV, application whitelisting, patch management, least privilege, network segmentation, backups.

3. Denial-of-service (DOS / DDOS):

- **Method:** Overwhelm a service or network with traffic or exploit resource limits

- **Goal/Impact:** Availability loss - legitimate

users cannot access service, can cause business disruption and revenue loss.

Defence : Rate limiting, traffic filtering, DDOS mitigation services (scrubbing), content delivery networks, redundancy.

Comparison summary :

- phishing targets people
- Malware targets systems/software to steal, modify or destroy data

Question 18: Explain how legal frameworks such as the General Data protection Regulation (GDPR) help mitigate cyber attacks and protect user privacy.

Answer:-

Legal frameworks and cybersecurity: Role of GDPR:

The General Data Protection Regulation (GDPR) is a European Union (EU) law designed to strengthen the protection of personal data and privacy of individuals. It plays an important role in mitigating cyber attacks and protecting user privacy through the following ways

1. Data Minimization:- GDPR requires organizations to collect only necessary personal data, reducing the amount of sensitive information that can be stolen in a cyber.

attack.

2. Security Measures: It mandates strong technical and organizational measures to safeguard data against unauthorized access, alteration, or loss.

3. Breach Notification – Organizations must report personal data breaches to authorities withing 72 hours. This ensures quick containment and response to cyber attacks.

4. User Rights: Individuals have rights such as access to their data, correction, detection and data probability.

GDPR significantly reduces the risk and impact of cyber attacks, while ensuring the privacy and trust of users.

Question 19: Explain the basic working of the DES algorithm using a simple 64-bit plain-text block and a 56-bit key. Show how the initial permutation, round functions, and final permutation contribute to the encryption process.

Answer:

Basic Working of the DES Algorithm:

The Data Encryption Standard (DES) is a symmetric-key block cipher that encrypts data in fixed-size blocks.

It operates on:

- Plaintext block : 64 bits (8 bytes)
- Key : 56 bits
- Output : 64-bit ciphertext.

1. Initial Permutation (IP)

- The 64-bit plaintext is rearranged according to a fixed table.
- This step does not involve the key; it simply changes bit positions to prepare for the rounds.
- Example: If the plaintext is $P = [P1, P2, \ldots, P64]$, the IP might move P58 to position 1, P50 to position 2, etc.

2. Rounds (16 Iterations)

The output of IP is divided into two halves:

- Left half (L0): first 32 bits
- Right half (R0): Last 32 bits.

a) Expansion (E-box)

- The 32-bit right half (R) is expanded to 48 bits by duplicating certain bits, using the E expansion table

b. key Mixing

- A 48 bit subkey is generated from the main 56-bit key using a key schedule
- The expanded R is XORed with the subkey : R_expanded ⊕ Subkey

c. Substitution (S-boxes)

- The 48-bit result is divided into 8 blocks of 6 bits.
- Each block is passed through a S-box to produce a 4-bit output.
- Total output = 32 bits.

d. Permutation (P-box)

- The 32-bit S-box output is permuted using a fixed P-box table.

e. Swap and combine :

- The output from P-box is XORed

with the left half (L)

- The right half becomes the new left half, and the result of the XOR becomes the new right half.

Formulae for each round:

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

3. Final Permutation (FP)

- After 16 rounds, the final swap is performed.
- The two halves are joined and passed through the inverse of the Initial Permutation to produce the Ciphertext.

20. In the DES algorithm, a 64-bit plaintext block is divided into two 32-bit halves: L_0 and R_0. Given $R_0 = 0x\ F0F0F0F0$ and round key $k_1 = 0x\ 0F0F0F0F$, compute the output of the first round's function $f(R_0, k_1)$ assuming XOR operation only. Then, find $L_1 = R_0$ and $R_1 = L_0 \oplus f(R_0, k_1)$, where $L_0 = 0xAAAAAAAA$.

<u>Answer:</u>

Given:

$R_0 = 0x\ F0F0F0F0$

Round key $k_1 = 0x\ 0F0F0F0F$

$L_0 = 0x\ AAAAAAAA$

Step 1:- Compute $f(R_0, k_1)$ assuming XOR operation only.

$f(R_0 k_1) = R_0 \oplus k_1 = 0x\ F0F0F0F0 \oplus 0x\ 0F0F0F0F$

Perform XOR byte-wise :

- $F0 \oplus 0F = FF$
- $F0 \oplus 0F = FF$
- $F0 \oplus 0F = FF$
- $F0 \oplus 0F = FF$

So, $f(R_0, K_1) = 0xFFFFFFFF$

Step 2: Compute $L_1 = R_0 = 0xF0F0F0F0$

Step 3: Compute $R_1 = L_0 \oplus f(R_0, K_1)$

$R_1 = 0xAAAAAAAA \oplus 0xFFFFFFFF$

<u>XOR byte-wise:</u>

- $AA \oplus FF = 55$
- $AA \oplus FF = 55$
- $AA \oplus FF = 55$
- $AA \oplus FF = 55$

So, $R_1 = 0x55555555$

$\therefore L_1 = 0xF0F0F0F0$, $R_1 = 0x55555555$

(Answer.)

21. Given the input word :

[0x23, 0xA7, 0x4C, 0x19]

Use the following partial AES S-box to perform the SubBytes transformation. Provide the resulting output word.

Row↓ \ Col →	3	4	7	9	A	C
1	6D	•	•	C6	•	•
2	D4	•	•	•	•	•
4	•	A1	•	•	•	2E
A	•	•	63	•	D2	•

Given Input word :

[0x23, 0xA7, 0x4C, 0x19]

Partial AES S-box table :

Row↓ \ Col→	3	4	7	9	A	C
1	6D	•	•	C6	•	•
2	D4	•	•	•	•	•
4	•	A1	•	•	2E	•
A	•	•	63	D2	•	•

For each byte, the high nibble (4 bits) is the row, and the low nibble is the column.

Let's find the corresponding S-box value for each byte:

- For 0x23 : row = 2, col = 3 → From table, row 2 col 3 = D4
- For 0xA7 : row = A, col = 7 → From table, row A col 7 = 63
- For 0x4C : row = 4, col = C → From table, row 4 col C = (not given in table → dot), so we leave it as is or unknown.
- For 0x19 : row = 1, col = 9 → From table, row 1 col 9 = C6

Resulting output word:

[0xD4, 0x63, unknown, 0xC6]

Since 0x4C is not present, we indicate unknown or no mapping given.

22. In AES encryption, apply the AddRoundKey step only. Given :

- Input word : [0x1A, 0x2B, 0x3C, 0x4D]
- Round key word : [0x55, 0x66, 0x77, 0x88]

Perform XOR between the input word and the round key, and write the resulting output word.

Answer :

The AddRoundKey step in AES encryption involve performing a bitwise XOR operation between the input word and the round key word.

Given :

input word : [0x14, 0x2B, 0x3C, 0x4D]

Round key word : [0x55, 0x66, 0x77, 0x88]

Compute the output word (XOR each byte).

Solution (byte-wise XOR) :

$0x1A \oplus 0x55 = 0x4F$ (00011010 ⊕ 01010101 = 01001111)

$0x2B \oplus 0x66 = 0x4D$ (00101011 ⊕ 01100110 = 01001101)

$0x3C \oplus 0x77 = 0x4B$ (00111100 ⊕ 01110111 = 01001011)

$0x4D \oplus 0x88 = 0xC5$ (01001101 ⊕ 10001000 = 11000101)

Resulting output word:

[0x4F, 0x4D, 0x4B, 0xC5]

<u>Brief explanation</u>: The AddRoundkey step performs a bitwise XOR between each byte of the state and the corresponding byte of the round key, producing the output word shown above.

23. Show how MixColumns uses the following fixed matrix over GF (2^8):

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

to transform an input column. Use an example column with values [0x01, 0x02, 0x03, 0x04]. [convert Hex to Binary]

<u>Answer:</u>

The MixColumns operation in AES works by multiplying each column of the state matrix by a fixed matrix over the Galois field GF (2^8).

Given that,

Input column (bytes):

$$a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0x01 \\ 0x02 \\ 0x03 \\ 0x04 \end{bmatrix}$$

MixColumns Matrix :-

$$M = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

So, the output column is $b = M \cdot a$ where each entry is bytewise GF(2^8) arithmetic (XOR for addition).

Multiplication rules used (AES)

- Multiply by 01 : value itself.
- Multiply by 02 (called xtime): left shift by 1 bit; if original MSB = 1 then XOR with 0x1B after the shift.
- Multiply by 03: $03 \cdot x = 02 \cdot x \oplus 01 \cdot x$

Input bytes in binary :

- $0x01 = 00000001_2$
- $0x02 = 00000010_2$
- $0x03 = 00000011_2$
- $0x04 = 00000100_2$

Row 1 Calculation

$$b_0 = (02 \cdot a_0) \oplus (03 \cdot a_1) \oplus (01 \cdot a_2) \oplus (01 \cdot a_3)$$

Compute terms :

$\rightarrow 02 \cdot a_0 = 02 \cdot 0x01 = xtime(0x01) = 0x02$

$\rightarrow 03 \cdot a_1 = (02 \cdot 0x02) \oplus 0x02 = xtime(0x02)$ XOR $0x02 = 0x04$ XOR $0x02 = 0x06$

$\rightarrow 01 \cdot a_2 = 0x03$

$\rightarrow 01 \cdot a_3 = 0x04$

Now XOR them :

$b_0 = 0x02 \oplus 0x06 \oplus 0x04 = (0x02 \oplus 0x06) = 0x04)$
$\oplus 0x04 \oplus 0x03 \oplus 0x04 = 0x04 \oplus 0x03$
$\oplus 0x04 = 0x03.$

So, $b_0 = 0x03.$

Row 2 calculation :

$$b_1 = (01 \cdot a_0) \oplus (02 \cdot a_1) \oplus (03 \cdot a_2) \oplus (01 \cdot a_3)$$

Terms :

- $01 \cdot a_0 = 0x01$
- $02 \cdot a_1 = xtime(0x02) = 0x04$

- $03 \cdot a_2$ = xtime (0x03) XOR 0x03 = (xtime (0x03) = 0x06) XOR 0x03 = 0x05

- $01 \cdot a_3$ = 0x04

XOR :

b_1 = 0x01 ⊕ 0x04 ⊕ 0x05 ⊕ 0x04 = (0x01 ⊕ 0x04 = 0x05) ⊕ 0x05 ⊕ 0x04 = 0x00 ⊕ 0x04 = 0x04

So, b_1 = 0x04

<u>Row 3 calculation :</u>

$$b_2 = (01 \cdot a_0) \oplus (01 \cdot a_1) \oplus (02 \cdot a_2) \oplus (03 \cdot a_3)$$

<u>Terms :</u>

→ $01 \cdot a_0$ = 0x01

→ $01 \cdot a_1$ = 0x02

→ $02 \cdot a_2$ = xtime (0x03) = 0x06

→ $03 \cdot a_3$ = xtime(0x04) XOR 0x04 = 0x08 XOR 0x04 = 0x0C

XOR :

b_2 = 0x01 ⊕ 0x02 ⊕ 0x06 ⊕ 0x0C

Compute stepwise :

$0x01 \oplus 0x02 = 0x03$; $0x03 \oplus 0x06 = 0x05$;

$0x05 \oplus 0x0C = 0x09$.

so, $b_2 = 0x09$.

<u>Row 4 calculation :</u>

$$b_3 = (03 \cdot a_0) \oplus (01 \cdot a_1) \oplus (01 \cdot a_2) \oplus (02 \cdot a_3)$$:

Terms :

- $03 \cdot a_0$ = xtime (0x01) XOR 0x01 = 0x02 XOR 0x01 = 0x03
- $01 \cdot a_1 = 0x02$
- $01 \cdot a_2 = 0x03$
- $02 \cdot a_3$ = xtime (0x04) = 0x08

XOR :

$$b_3 = 0x03 \oplus 0x02 \oplus 0x03 \oplus 0x08$$

stepwise :

$0x03 \oplus 0x02 = 0x01$; $0x01 \oplus 0x03 = 0x02$;

$0x02 \oplus 0x08 = 0x0A$

so, $b_3 = 0x0A$.

Final result (output column)

$$b = \begin{bmatrix} 0x03 \\ 0x04 \\ 0x09 \\ 0x0A \end{bmatrix}$$

Binary forms —

- $0x03 = 00000011_2$
- $0x04 = 00000100_2$
- $0x09 = 00001001_2$
- $0x0A = 00001010_2$

24. Describe how AES-OFB mode works. How does it ensure synchronization between encryption and decryption streams?

Answer:-

AES-OFB (Output Feedback) is a mode of operation for the Advanced Encryption Standard (AES) that ~~trans~~ turns a block cipher into a stream cipher. It works by generating a keystream, which is then XORed with the plaintext to produce the ciphertext.

How it works:

1. Initialization: The process starts with a unique Initialization Vector (IV). Which is typically a random value. This IV is fed into the AES block cipher.

2. Keystream Generation: The output of the AES encryption of the IV becomes the first part of the keystream. This output

is also fed back into the AES block cipher for the next round, generating the next part of the keystream. This continues, creating a sequence of keystream blocks.

3. Encryption : Each block of the keystream is XORed with a corresponding block of plaintext to produce the ciphertext.

4. Decryption : The decryption process is identical to the encryption process. The same IV is used to generate the exact same keystream. The keystream is then XORed with the cipertext to recover the original plaintext.

The key to OFB mode is that the keystream is generated independently of the plaintext and ciphertext.

Synchronization in AES-OFB :

- Both sender and receiver start with the same Initialization vector (IV) and same secret key.
- keystream blocks are generated only from the previous keystream block, not from the ciphertext or plaintext.
- This means if IV and key are identical, both sides produce exactly the same keystream sequence, keeping encryption and decryption aligned.

If IV or key changes, synchronization will break.

25. Which AES modes cause error propagation during decryption, and how does this affect the integrity of the decrypted message? Illustrate with CBC and CFB modes.

Answer :

AES Modes Causing Error Propagation :

Some AES modes (CBC and CFB) cause an error in a ciphertext block to affect multiple plaintext blocks during decryption. This reduces the integrity of the decrypted message because even a single-bit error can corrupt more than one block.

1. CBC (Cipher Block Chaining) Mode :

- Decryption formula : $P_i = AES^{-1}(C_i) \oplus C_{i-1}$
- Effect effect :
  - if C_i has a 1-bit error $\rightarrow P_i$ becomes completely random.
  - The same bit position in P_{i+1} will

also be flipped.

- Error affects two consecutive plaintext blocks.

2. CFB (cipher Feedback) Mode

- Decryption formula : $P_i = C_i \oplus AES(C_{i-1})$
- Error effect :
  - if C_i has a 1-bit error → the same bit in P_i is wrong.
  - P_{i+1} becomes completely random.
  - After that, decryption returns to normal.

26. Which AES mode would you recommend for encrypting large files with parallel processing, and why? Justify your choice between ECB, CBC and CTR.

Answer:

Recommended Modes: AES- CTR (counter Mode)

Justification:

- Parralled processing:
  - In CTR, each block is encrypted by xoring the plaintext with a keystream generated from AES (key, Counter)
  - The counter values for all blocks are known in advance, so encryption and decryption of different blocks can happen independently and in parralled.
- Performance:
  - No chaining between blocks - faster than CBC for large files.

- Security:
  - Unlike ECB, CTR does not produce identical ciphertext for identical plaintext blocks.
  - As secure as CBC when IV/nonce is unique.

- Flexibility:
  - Works efficiently for both stream-like and random access data.

Why not ECB?

- Reveals patterns in the data - insecur for large files.

Why not CBC?

- Each block depends on the previous ciphertext block → cannot be parralized in encryption.

For large file encryption with parralled

processing, AES-CTR is best because it is secure and highly parrallelizable.

27. Given a message "A" represented as $M=1$, encrypt it using public key $e=5$, $n=14$. What is ciphertext? Then decrypt using private key $d=11$.

<u>Solution :</u>

Given, $M=1$, $e=5$, $n=14$, $d=11$.

Encryption : $C \equiv M^e \bmod n = 1^5 \bmod 14 = 1$.

Decryption :

$$M \equiv C^d \bmod n = 1^{11} \bmod 14 = 1.$$

Why this works : RSA relies on $M^{ed} \equiv M \pmod{n}$ when $ed \equiv 1 \pmod{\varphi(n)}$. Here, $\varphi(14) = 6$ and $e \cdot d = 5 \cdot 11 = 55 \equiv 1 \pmod{6}$, so decryption recovers M.

$\therefore$ Ciphertext $C = 1$ and Decrypted Message, $M=1$

28. Given : Message hash : $H(M) = 5$, RSA Private key : $d = 3, n = 33$. Generate the digital signature.

Answer :

Given : hash $H(M) = 5, d = 3, n = 33$.

Signature generation (sign with private key d)

We need to generat a digital signature for a message hash.

- The message hash is $H(M) = 5$
- The RSA private key is $(d, n) = (3, 33)$
- The digital signature S is computed using the formula :

$$S = H(M)^d \pmod{n}$$

$$S = 5^3 \pmod{33}$$

$$= 125 \pmod{33}$$

- $125 = 3 \times 33 + 26$, so $125 \pmod{33} = 26$
- The digital signature is 26

29. Aleya and Badol are using the Diffie Hellman key exchange protocol. They agree on the following public values: Prime modulus: $p = 17$, Base (generator): $g = 3$, Aleya chooses a private key, $a = 4$, Badol chooses a private key $b = 5$. Compute public key of Aleya and Badol.

Solution:

We'll compute the public keys for Aleya and Badol based on the Diffie-Hellman protocol.

- Public values: Prime modulus $p = 17$, base $g = 3$.
- Aleya's private key: $a = 4$
- Badol's private key: $b = 5$
- Aleya's public key (A):
  - $A = g^a \pmod p = 3^4 \pmod{17} = 81 \pmod{17}$
  - $81 = 4 \times 17 + 13$. so $81 \pmod{17} = 13$
  - Aleya's public key is 13

- Badol's public key (B):
  - $B = g^b \pmod p = 3^5 \pmod{17} = 243 \pmod{17}$.
  - $243 = 14 \times 17 + 5$, so $243 \pmod{17} = 5$
  - Badol's public key is 5

30. A simple (non-cryptographic) hash function $H(x)$ is defined as the sum of ASCII values of the characters in a message, modulo 100:

$$H(x) = (\Sigma \text{ASCII of characters in } x) \bmod 100$$

Compute the hash value of the message; "AB" and "BA" using this function. Do the two messages produce the same hash? What does this imply about collision resistance in weak hash functions?

Answer:

Definition:

$$H(x) = \left(\sum \text{ASCII (chars in } x)\right) \bmod 100$$

ASCII values: A = 65, B = 66

Compute hashes:

$$H(\text{"AB"}) = (65+66) \bmod 100 = 131 \bmod 100 = 31,$$

$$H(\text{"BA"}) = (66+65) \bmod 100 = 131 \bmod 100 = 31$$

Result:

Both "AB" and "BA" produce the same hash → collision.

Collision Resistance:

- The two different messages, "AB" and "BA", produce the same hash value. This is called collision.
- This hash function is not collision-resistant because it's based on a simple modular sum.

Question 31: In a secure message system, a simple Message Authentication Code (MAC) is computed using modular addition:

$$MAC = (Message + Secret\ key) \bmod 17$$

Given: Message: 15, Secret key: 7. Compute the MAC for the message. Suppose an attacker changes the message to 10 but doesn't know the key. Can they forge the correct MAC easily? Explain Briefly.

<u>Answer:</u>

Given, Message $M = 15$, $k = 7$

Compute MAC:

$$MAC = (15 + 7) \bmod 17 = 22 \bmod 17 = 5$$

Attacker scenario: Suppose attacker changes message to $M' = 10$ but does not know K.

Read MAC for M' is:

$$MAC' = (10 + 7) \bmod 17 = 17 \bmod 17 = 0$$

Why forging is easy (explain):

Because MAC is linear: $MAC = M + k \pmod{17}$, if attacker knows one valid pair (M, MAC), and wants to create $M' = M + \Delta$, they can compute $MAC' = MAC + \Delta \pmod{17}$ without knowing k.

Example here: $\Delta = -5 \equiv 12 \pmod{17}$;

$MAC' = 5 + 12 = 17 \equiv 0$, which matches the true MAC

Conclusion: This MAC is insecure - it is trivially forgeable because it leaks linear relation with the message.

32. Explain the steps involved in the TLS handshake process. How are symmetric keys established securely using asymmetric cryptography during the handshake?

Answer:

TLS Handshake Steps & Symmetric key Establishment:

1. ClientHello → Client sends supported TLS version, cipher suites, random number.

2. ServerHello – Server selects TLS version & cipher suite, sends its random number.

3. Server Certificate – Contains servers public key, signed by CA for Authentication,

4. Key exchange →
   - RSA Mode
   - ECDHE/DHE mode.

5. Pre-Master Secret → Master Secret.

6. Session key → From master secret.

7. Finished Messages → Both parties send encrypted "Finished" message to confirm the handshake

How symmetric keys are established Secure?

Asymmetric encryption (RSA or Diffie-Hellman) ensures that only the intended parties can compute the shared secret from which symmetric keys are derived.

33. Explain the layered architecture (protocol stack) of SSH. Briefly describe the roles of each layer.

Answer:

SSH refers to "secure shell" is a protocol that provides a secure channel over an unsecured network. It has a layered architecture consisting of three main layers:

① Transport layer protocol:

This is the lowest layer, responsible for managing the secure connection.

- Handles encryption
- integrity protection
- compression
- server authentication.

2. User Authentication Protocol :

This layer runs on top of the transport layer and handles client authentication

- Varifies the client's identity using password
- public keys or other methods.

3. Connection Protocol :

Multiplexes the encrypted channel into multiple logical channels. This is the highest layer.

34. Explain the steps involved in the TLS handshake process.

Answer :- TLS handshake process step :

1. ClientHello → Purposes connection parameters.
2. ServerHello → Proposes connection parameters.
3. Certificate & key exchange.
4. Generate Master secret
5. Generate session keys
6. Finished Messages
7. Secure communication.

Question 35: What is the general form of all elliptic curve equation over a finite field, and why is it used in cryptography?

Answer:

The general form of an elliptic curve equation over a finite field is:

$$y^2 = x^3 + ax + b \pmod{p}$$

Here, p is a large prime number that defines the finite field, and a and b are constants such that

$$4a^3 + 27b^2 \not\equiv 0 \pmod{p}.$$

(ensures no singularities)

Use in cryptography:

Provides a group structure for Elliptic Curve cryptography (ECC), enabling secure key exchange, signature, and encryption with smaller keys.

Question 36: How does ECC achieve the same level of security as RSA with a smaller key size? Briefly explain.

Answer:

Elliptic Curve Cryptography (ECC) is a public key cryptography technique that provides the same cryptographic strength as RSA but with much smaller key sizes. The main reason lies in the underlying mathematical problems.

1. Mathematical Foundation:

- RSA is based on the Integer Factorization Problem (IFP).
- ECC is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP).

2. key size comparison:

- ECC 256-bit ≈ RSA 3072-bit security level
- ECC 384-bit ≈ RSA 7680-bit security level

3. Conclusion:

ECC provides strong security with reduced key sizes due to the computational hardness of the ECDLP compared to integer factorization.

37. Given the elliptic curve $y^2 = x^3 + 2x + 3 \bmod 97$, determine whether the point $P = (3, 6)$ lies on the curve.

Answer:

Given that,

Curve : $y^2 \equiv x^3 + 2x + 3 \pmod{97}$

point : $P = (3, 6)$

$LHS = y^2 = 6^2 = 36 \pmod{97}$

$RHS = x^3 + 2x + 3 = 27 + 6 + 3 = 36 \pmod{97}$

Since LHS = RHS, point lies on the curve.

38. Given public key ($p = 23$, $g = 5$, $h = 8$) and message $m = 10$, compute the ElGamal ciphertext using random $k = 6$.

Answer:

To compute the ElGamal ciphertext, we use the following public values:

$p = 23$, $g = 5$, $h = 8$, and the message $m = 10$, the random number is $k = 6$

The ciphertext consists of two parts, (C_1, C_2)

- C_1 computation:

$$C_1 \equiv g^k \pmod{p}$$

$$C_1 = 5^6 \pmod{23}$$

$$5^2 = 25 \equiv 2 \pmod{23}$$

$$5^4 = (5^2)^2 \equiv 2^2 = 4 \pmod{23}$$

$$5^6 = 5^4 \times 5^2 \equiv 4 \times 2 = 8 \pmod{23}$$

So, $C_1 = 8$.

C_2 computation :

$$C_2 = m \times h^k \pmod p$$

First, we calculate h^k :

$$h^k = 8^6 \pmod{23}$$

$$8^2 = 64 = 2 \times 23 + 18 \equiv 18 \pmod{23}$$

$$8^4 = (8^2)^2 \equiv 18^2 = 328 = 14 \times 23 \times 2 \equiv 2 \pmod{23}$$

$$8^6 = 8^4 \times 8^2 \equiv 2 \times 18 = 36 = 1 \times 23 + 13 \equiv 13 \pmod{23}$$

Now, we compute C_2 :

$$C_2 = 10 \times 13 \pmod{23} = 130 \pmod{23}$$

$$130 = 5 \times 23 + 15 \equiv 15 \pmod{23}$$

so, $C_2 = 15$

The ElGamal ciphentext is $(8, 15)$

$$\boxed{C_1, C_2 = (8, 15)}$$

Q.89. Explain how lightweight cryptography is important for securing IOT devices. Give one example of a lightweight encryption algorithm used in IOT.

Answer:

Light weight cry-tography is important for IoT because IoT devices are small, low-power, and have little memory.

It gives security (data protection, authentication, privacy) without using much power or memory.

Example:

PRESENT cipher - small, fast and good for tiny devices like RFID tags.

Q.40.

List and briefly explain and three common IoT-Specific attacks (e.g. firmware hijacking, physical tampering, botnets like Mirai). What mitigation strategies can be applied?

Answer:

Three common IoT attacks and Solutions

1. Firmware Hijacking:
   - What: Hacker changes device software to control it.
   - Fix: Use signed firmware updates (only allow trusted updates).

2. Physical Tampering:
   - What: Some physically opens or changes the device
   - Fix: Use tamper-proof casing and sensors to detect tampering.

3. Botnet Attack (e.g. Mirai):

- What: Many hacked IoT devices are used together to attack websites.
- fix: change default passwords, update software, block unknown traffic