Topic 1. Cryptanalysis on Symmetric Ciphers

Scenario:

A cybersecurity firm wants to assess the strength of a newly developed symmetric cipher. They aim to identify potential vulnerabilities and weaknesses in the encryption algorithm.

Gaps, Motivations, and Desired Security Features:

- **Gaps**: Many symmetric ciphers, while theoretically sound, may have practical vulnerabilities when implemented.
- **Motivations**: To ensure that the symmetric cipher can withstand real-world attacks and is suitable for securing sensitive data.
- **Desired Security Features**:
  - Resistance to known plaintext attacks
  - Resistance to chosen plaintext attacks
  - Resistance to differential cryptanalysis
  - Resistance to linear cryptanalysis

Proposed Solutions:

- **Solution Architecture**:
  - **Ciphertext Collection**: Gather a significant amount of encrypted data for analysis.
  - **Cryptanalysis Tools**: Utilize specialized software to analyze the ciphertext.
  - **Attack Models**: Define specific attack scenarios to test the cipher's resistance.
- **Solution Details**:
  - **Data Collection**: Use the cipher to encrypt known data sets.
  - **Analysis**: Apply various cryptanalysis techniques to attempt to decipher the collected data without the key.

Implementation and Testing:

- **Implementation**:
  - Set up a controlled environment for encryption and data collection.
  - Implement the cipher and generate a significant amount of ciphertext.
- **Testing**:
  - **Functional Testing**: Ensure the cipher encrypts and decrypts correctly.
  - **Security Testing**: Use tools or your test case to attempt to break the encryption.
  - **Attack Scenarios**: Test against known plaintext, chosen plaintext, differential cryptanalysis, and linear attacks.

Deployment:

- If the cipher is found to be secure, it can be recommended for deployment in real-world applications.
- If vulnerabilities are found, they should be documented and shared with the cipher's developers for rectification.

References:

- Cite academic papers, textbooks, and research articles on RSA encryption and cryptanalysis techniques.
- Refer to relevant standards and guidelines (e.g., NIST publications).
- CTF tools

Assessment Rubric:

Assign percentages to each of the following criteria:

1. **Quality of Research (20%):** How well the student has explored vulnerabilities and provided evidence.
2. **Effectiveness of Proposed Solutions (30%):** How well the suggested improvements address the identified gaps.
3. **Testing and Analysis (25%):** How effectively the student conducted experiments and analyzed results.
4. **Deployment Considerations (10%):** Evaluation of the potential real-world deployment of the enhanced RSA-based algorithm.
5. **Presentation and Documentation (15%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 2. Cryptanalysis on RSA-based Algorithms

Scenario:

Secure service company uses RSA-based algorithms for securing transactions and digital signatures. They want to ensure the robustness of their RSA implementation against potential attacks.

Gaps, Motivations, and Desired Security Features:

- **Gaps**: While RSA is a widely accepted and used public-key cryptosystem, improper implementations or usage of weak parameters can lead to vulnerabilities.
- **Motivations**: To ensure the integrity and confidentiality of financial transactions and to maintain the trust of clients and stakeholders.
- **Desired Security Features**:
    - Resistance to factorization attacks (e.g., Pollard's rho algorithm).
    - Resistance to timing attacks.
    - Resistance to chosen ciphertext attacks.
    - Yours?

Proposed Solutions:

- **Solution Architecture**:
    - **Key Generation and Collection**: Generate RSA keys with varying bit lengths for analysis.
    - **Cryptanalysis Tools**: Utilize specialized software (CTF), hardware or your test cases to analyze the RSA implementation.
    - **Attack Models**: Define specific attack scenarios to test the RSA algorithm's resistance.
- **Solution Details**:
    - **Data Collection**: Use the RSA algorithm to encrypt known data sets and generate digital signatures.
    - **Analysis**: Apply various cryptanalysis techniques to attempt to decipher the collected data or forge signatures without the private key.

Implementation and Testing:

- **Implementation**:
    - Set up a controlled environment for key generation, encryption, and signature creation.
    - Implement the RSA algorithm and generate a significant amount of encrypted data and signatures.
- **Testing**:
    - **Functional Testing**: Ensure the RSA algorithm encrypts, decrypts, and signs correctly.
    - **Security Testing**: Use tools like YAFU, RSA toolkits, RsaCtfTool, or your test case to attempt to break the encryption or forge signatures.

- **Attack Scenarios**: Test against factorization attacks, timing attacks, and chosen ciphertext attacks.

Deployment:

- If the RSA implementation is found to be secure, it can continue to be used for financial transactions and digital signatures.
- If vulnerabilities are found, they should be documented and shared with the institution's IT team for rectification, and potentially, a switch to a more secure algorithm or implementation might be recommended.

References:

- Cite academic papers, textbooks, and research articles on RSA encryption and cryptanalysis techniques.
- Refer to relevant standards and guidelines (e.g., NIST publications).

Assessment Rubric:

Assign percentages to each of the following criteria:

6. **Quality of Research (20%):** How well the student has explored vulnerabilities and provided evidence.
7. **Effectiveness of Proposed Solutions (30%):** How well the suggested improvements address the identified gaps.
8. **Testing and Analysis (25%):** How effectively the student conducted experiments and analyzed results.
9. **Deployment Considerations (10%):** Evaluation of the potential real-world deployment of the enhanced RSA-based algorithm.
10. **Presentation and Documentation (15%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 3. Cryptanalysis on ECC-based Algorithms

Scenario:

A tech company has integrated ECC-based algorithms for secure communications in their IoT devices. They want to validate the strength and security of their ECC implementation to prevent potential breaches.

Gaps, Motivations, and Desired Security Features:

- **Gaps**: ECC, while offering good security with shorter key lengths compared to traditional methods, can be vulnerable if not implemented correctly or if weak curves are chosen.
- **Motivations**: To ensure secure communication between IoT devices and to maintain user trust and data integrity.
- **Desired Security Features**:
    - Resistance to Pollard's rho attack for elliptic curves.
    - Resistance to side-channel attacks.
    - Resistance to invalid curve attacks.
    - Yours

Proposed Solutions:

- **Solution Architecture**:
    - **Key Generation and Collection**: Generate ECC keys using various elliptic curves for analysis.
    - **Cryptanalysis Tools**: Utilize specialized software to analyze, your test case the ECC implementation, CTF tools.
    - **Attack Models**: Define specific attack scenarios to test the ECC algorithm's resistance.
- **Solution Details**:
    - **Data Collection**: Use the ECC algorithm to encrypt known data sets.
    - **Analysis**: Apply various cryptanalysis techniques to attempt to decipher the collected data without the private key.

Implementation and Testing:

- **Implementation**:
    - Set up a controlled environment for key generation and encryption.
    - Implement the ECC algorithm and generate a significant amount of encrypted data.
- **Testing**:
    - **Functional Testing**: Ensure the ECC algorithm encrypts and decrypts correctly.
    - **Security Testing**: Use tools like SageMath, ECC toolkits, CTF Tools to attempt to break the encryption.
    - **Attack Scenarios**: Test against Pollard's rho attack, side-channel attacks, and invalid curve attacks,...

Deployment:

- If the ECC implementation is found to be secure, it can continue to be used for secure communications in IoT devices.
- If vulnerabilities are found, they should be documented and shared with the company's development team for rectification. Recommendations might include using different elliptic curves or enhancing the implementation to counter specific attack vectors.

Topic 4. Cryptanalysis on Lattice-based Algorithms

Scenario:

A government agency is considering the adoption of lattice-based cryptographic algorithms for post-quantum secure communications. Before full-scale deployment, they want to evaluate the resilience of lattice-based schemes against both classical and quantum attacks.

Gaps, Motivations, and Desired Security Features:

- **Gaps**: Lattice-based cryptography is believed to be resistant to quantum attacks, but its resilience against certain classical cryptanalytic techniques is still under study.
- **Motivations**: With the advent of quantum computing, there's an urgent need for cryptographic schemes that can withstand quantum threats. Lattice-based cryptography offers a promising solution.
- **Desired Security Features**:
    - Resistance to basis reduction attacks.
    - Resistance to quantum algorithms like Shor's algorithm.
    - Efficient performance even with increased security parameters.

Proposed Solutions:

- **Solution Architecture**:
    - **Key Generation and Collection**: Generate keys using various lattice-based cryptographic schemes.
    - **Cryptanalysis Tools**: Utilize both classical and quantum simulating software to analyze the lattice-based implementation.
    - **Attack Models**: Define specific attack scenarios to test the algorithm's resistance.
- **Solution Details**:
    - **Data Collection**: Use the lattice-based algorithm to encrypt known data sets.
    - **Analysis**: Apply various cryptanalysis techniques, both classical and quantum-inspired, to attempt to decipher the collected data.

Implementation and Testing:

- **Implementation**:
    - Set up a controlled environment for key generation and encryption.
    - Implement the lattice-based algorithm and generate a significant amount of encrypted data.
- **Testing**:
    - **Functional Testing**: Ensure the lattice-based algorithm encrypts and decrypts correctly.
    - **Security Testing**: Use tools like NTL (Number Theory Library) or other lattice-specific toolkits to attempt to break the encryption.
    - **Attack Scenarios**: Test against basis reduction attacks and simulated quantum attacks.

Deployment:

- If the lattice-based implementation is found to be secure against both classical and quantum threats, it can be recommended for deployment in secure government communications.
- If vulnerabilities are found, they should be documented and shared with the agency's cryptographic team for rectification. Recommendations might include adjusting security parameters or considering alternative post-quantum cryptographic schemes.

References:

- Cite academic papers, textbooks, and research articles on Lattice-based cryptanalysis techniques.
- Refer to relevant standards and guidelines (e.g., NIST publications).

Assessment Rubric:

Assign percentages to each of the following criteria:

11. **Quality of Research (20%):** How well the student has explored vulnerabilities and provided evidence.
12. **Effectiveness of Proposed Solutions (30%):** How well the suggested improvements address the identified gaps.
13. **Testing and Analysis (25%):** How effectively the student conducted experiments and analyzed results.
14. **Deployment Considerations (10%):** Evaluation of the potential real-world deployment of the enhanced RSA-based algorithm.
15. **Presentation and Documentation (15%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 5. Multimedia Product Service Platform (e.g., Netflix, Spotify)

Application Scenarios:

1. **Gaps:** Identify security vulnerabilities and gaps in the existing security mechanisms of multimedia product service platforms like Netflix or Spotify.
2. **Reliable Arguments for the Gaps:** Provide evidence-based arguments that support the existence of identified security vulnerabilities and gaps, with reference to real-world examples and case studies.
3. **Motivations:** Emphasize the critical motivation behind securing these platforms, considering the vast amounts of user data, payment information, and intellectual property they handle.
4. **Desired Functional and Security Features:**
    - Strong data protection and confidentiality.
    - Secure user authentication and authorization.
    - Data integrity and authenticity assurance.
    - Protection against various cyber threats like piracy and unauthorized access.
5. **Related Stakeholders:** Identify stakeholders such as content providers, users, regulatory bodies, and the platform's administrators.

Determine Specific Algorithms:

- **Chaotic-based Stream Cipher:**
    - Known for its unpredictability and encryption efficiency.
    - Suitable for secure multimedia content streaming.
    - Applicable for ensuring data confidentiality in multimedia platforms.
- **AES (Advanced Encryption Standard):**
    - Widely recognized symmetric encryption algorithm.
    - Ensures data confidentiality and security.
    - Ideal for encrypting and decrypting stored content.

Solutions:

Solution Architecture:

- Implement a hybrid cryptographic system using Chaotic-based Stream Cipher and AES.
- The components include encryption/decryption modules, key management, user authentication, and access control mechanisms.

Detail of Functional Features:

- **Encryption and Decryption:**
    - Chaotic-based Stream Cipher for real-time data streaming.
    - AES for data at rest and during transmission.
- **User Authentication and Authorization:**
    - Secure user login and session management.
    - Role-based access control for content distribution.

Detail of Security Features:

- **Confidentiality:**
  - Strong encryption mechanisms to protect the confidentiality of multimedia content.
- **Authentication:**
  - Secure user authentication processes to verify the identities of users and prevent unauthorized access.
- **Integrity:**
  - Ensure data integrity to prevent tampering or modification of multimedia content.

Implementation and Testing:

Tools or Libraries Recommendation:

- Use programming languages like Python, Java, or C++ for implementation.
- Utilize cryptographic libraries like PyCryptodome for Chaotic-based Stream Cipher and existing AES libraries.
- Conduct testing using platforms such as Jupyter Notebook for code development.

Experimental Scenarios:

- Simulate real-time streaming of multimedia content.
- Test the AES encryption for data at rest.

Testing Goals and Conducts:

- Goal: Ensure that the hybrid encryption system effectively secures multimedia content and prevents unauthorized access.
- Conduct:
  - Real-time streaming tests to assess the performance of Chaotic-based Stream Cipher.
  - Data integrity tests for AES-encrypted content at rest.

Deployment:

- Deploy the hybrid encryption system in a controlled environment to assess its real-world performance and security.

References:

- Cite academic papers, textbooks, and research articles on Chaotic-based Stream Ciphers, AES, and multimedia platform security.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in the multimedia platform's security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing Chaotic-based Stream Cipher and AES.

3. **Solution Architecture (15%):** The clarity and effectiveness of the hybrid cryptographic system's design.
4. **Functional Features (20%):** The effectiveness of the encryption, authentication, and authorization features.
5. **Security Features (20%):** How well the student has addressed confidentiality, authentication, and integrity concerns.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 6. Application Scenarios: Online Shopping Service Platform (e.g., Amazon, Shopee)

Application Scenarios:

1. **Gaps:** Identify and analyze security vulnerabilities and gaps in the existing security measures of online shopping platforms.
2. **Reliable Arguments for the Gaps:** Provide strong evidence and arguments to support the identified security vulnerabilities, with references to real-world breaches and risks.
3. **Motivations:** Emphasize the motivation behind securing online shopping platforms, considering the financial and personal data involved.
4. **Desired Functional and Security Features:**
    - Strong data protection for user information.
    - Secure payment processing.
    - Protection against fraud, data breaches, and unauthorized access.
5. **Related Stakeholders:** Identify stakeholders such as customers, sellers, payment processors, and platform administrators.

Determine Specific Algorithms:

- **AES (Advanced Encryption Standard):**
    - A widely recognized symmetric encryption algorithm.
    - Ideal for securing data at rest and during transmission.
    - Suitable for encrypting sensitive user information.
- **ECDHE (Elliptic Curve Diffie-Hellman Ephemeral):**
    - A key exchange algorithm that provides secure key agreement between clients and servers.
    - Suitable for secure communications and establishing session keys.
- **Post-Quantum Digital Signature FALCON:**
    - A digital signature algorithm designed to be quantum-resistant.
    - Ideal for ensuring the authenticity of transactions and communications.

Solutions:

Solution Architecture:

- Implement a comprehensive security system for the online shopping platform, incorporating AES, ECDHE, and FALCON.
- Components include secure data storage, secure payment processing, secure user authentication, and secure communications.

Detail of Functional Features:

- **Secure Data Storage:**
    - Use AES for encrypting and securing user data in the database.
    - Implement access control mechanisms for authorized access.
- **Secure Payment Processing:**
    - Use ECDHE for secure key exchange between the platform and payment processors.
    - Implement fraud detection and prevention measures.

- **Secure User Authentication:**
    - Implement strong user authentication mechanisms.
    - Ensure the privacy and security of user credentials.

Detail of Security Features:

- **Data Protection:**
    - AES for data encryption and protection.
    - Regular security audits and monitoring.
- **Secure Communications:**
    - ECDHE for secure and confidential data exchange.
    - FALCON for quantum-resistant digital signatures (Digital certificate, invoice, receipt

Implementation and Testing:

Tools or Libraries Recommendation:

- Utilize programming languages such as Python, Java, and libraries like Bouncy Castle.
- Set up secure payment gateways for testing, use Wireshark for packet analysis, and conduct penetration testing.

Experimental Scenarios:

- Simulate user transactions, including registration, product searches, payments, and reviews.
- Perform penetration testing and vulnerability assessments.

Testing Goals and Conducts:

- Goal: Ensure the secure functionality of the online shopping platform and identify potential vulnerabilities.
- Conduct:
    - Real-time transaction testing with secure payments.
    - Evaluate the strength of encryption and secure key exchange.

Deployment:

- Deploy the enhanced online shopping platform in a controlled environment to assess its real-world performance and security.

References:

- Cite academic papers, textbooks, and research articles on AES, ECDHE, Post-Quantum cryptography, and online shopping platform security.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in online shopping platform security.

2. **Algorithm Selection (10%):** Appropriateness and justification for choosing AES, ECDHE, and FALCON.
3. **Solution Architecture (15%):** The clarity and effectiveness of the security system's design.
4. **Functional Features (20%):** The effectiveness of data protection, payment processing, and user authentication.
5. **Security Features (20%):** How well the student has addressed data protection, secure communications, and digital signatures.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution..
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 7. Encryption, Access Control, and Query in Cloud-Native DBMS

Application Scenarios:

1. **Gaps:** Identify security vulnerabilities and gaps in cloud-native database management systems (DBMS) related to data encryption, access control, and query security.
2. **Reliable Arguments for the Gaps:** Provide robust evidence and arguments to support the identified security vulnerabilities, with reference to real-world breaches and data exposure risks.
3. **Motivations:** Emphasize the motivation behind securing cloud-native DBMS, considering the sensitivity of data, the cloud's shared environment, and regulatory requirements.
4. **Desired Functional and Security Features:**
    - Strong data encryption to protect data at rest and in transit.
    - Fine-grained access control to restrict data access based on user roles.
    - Secure and efficient querying of encrypted data using ABE.
5. **Related Stakeholders:** Identify stakeholders such as database administrators, cloud service providers, and end-users.

Determine Specific Algorithms:

- **Authentication and Key Agreement:**
    - Implement secure authentication and key agreement mechanisms for users to access encrypted data.
    - Ensure secure data transmission.
- **Attribute-based Encryption (ABE):**
    - Utilize ABE for fine-grained access control.
    - Implement ABE policies to determine data access based on user attributes.

Solutions:

Solution Architecture:

- Implement a cloud-native DBMS with a layered security architecture.
- Include secure authentication, encryption, fine-grained access control, and efficient ABE-based query processing.

Detail of Functional Features:

- **Data Encryption:**
    - Use robust encryption algorithms for data at rest and in transit.
    - Secure and efficient data encryption and decryption processes.
- **Access Control:**
    - Implement fine-grained access control based on user attributes using ABE.
    - Enforce data access policies based on user attributes.
- **Secure Query Processing:**
    - Implement ABE-based query processing to enable secure and efficient querying of encrypted data.

Detail of Security Features:

- **Authentication and Key Agreement:**
  - Strong user authentication to prevent unauthorized access.
  - Secure key agreement for data encryption.
- **Access Control:**
  - Fine-grained access control based on user attributes.
  - Enforcement of access policies.

Implementation and Testing:

Tools or Libraries Recommendation:

- Utilize programming languages like Python, Java, and cryptographic libraries.
- Implement ABE using libraries such as Charm-crypto.
- Conduct testing using cloud-based DBMS platforms.

Experimental Scenarios:

- Simulate user access scenarios with different attributes.
- Perform ABE-based query processing tests.

Testing Goals and Conducts:

- Goal: Ensure the secure functionality of the cloud-native DBMS and identify potential vulnerabilities.
- Conduct:
  - Real-time user access scenarios.
  - Evaluate the efficiency of ABE-based query processing.

Deployment:

- Deploy the enhanced cloud-native DBMS in a controlled cloud environment to assess its real-world performance and security.

References:

- Cite academic papers, textbooks, and research articles on authentication and key agreement, Attribute-based Encryption, cloud-native DBMS, and data security in the cloud.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in cloud-native DBMS security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing authentication, key agreement, and Attribute-based Encryption.
3. **Solution Architecture (15%):** The clarity and effectiveness of the security system's design.

4. **Functional Features (20%):** The effectiveness of data encryption, access control, and ABE-based query processing.
5. **Security Features (20%):** How well the student has addressed authentication, key agreement, fine-grained access control, and data encryption.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 8. Secure Network Protocols in IoT-based Smart Cities

Application Scenarios:

1. **Gaps:** Identify security vulnerabilities and gaps in the network protocols used in IoT-based smart city applications.
2. **Reliable Arguments for the Gaps:** Provide strong evidence and arguments supporting the identified security vulnerabilities, citing real-world examples of IoT-related security breaches.
3. **Motivations:** Emphasize the motivation for securing IoT networks in smart cities, considering the potential consequences of security breaches in critical infrastructure.
4. **Desired Functional and Security Features:**
    - Strong authentication and authorization mechanisms.
    - Secure key agreement for device communication.
    - Data integrity and confidentiality.
    - Protection against unauthorized access and data tampering.
5. **Related Stakeholders:** Identify stakeholders, including city administrators, IoT device manufacturers, and citizens.

Determine Specific Algorithms:

- **Authentication and Key Agreement:**
    - Implement robust authentication and key agreement mechanisms for IoT devices.
- **Securing End-Devices:**
    - Apply encryption and access control measures to secure end-devices.
- **Securing Data Sink:**
    - Protect the data sink and its storage mechanisms from unauthorized access and data tampering.

Solutions:

Solution Architecture:

- Implement a secure network protocol for IoT devices in smart cities.
- Components include strong authentication, key agreement, end-device security, and secure data sink.

Detail of Functional Features:

- **Authentication and Key Agreement:**
    - Implement secure and efficient authentication and key agreement mechanisms for IoT devices.
- **Securing End-Devices:**
    - Use encryption and access control to protect the data stored on IoT devices.
- **Securing Data Sink:**
    - Protect data sink storage with encryption and access control mechanisms.

Detail of Security Features:

- **Authentication and Authorization:**
  - Strong user authentication and authorization to ensure secure device access.
- **End-Device Security:**
  - Encryption to protect data at rest on end-devices.
  - Access control mechanisms to restrict access to authorized users.
- **Data Sink Security:**
  - Encryption to protect data stored at the data sink.
  - Access control mechanisms to restrict unauthorized access.

Implementation and Testing:

Tools or Libraries Recommendation:

- Utilize programming languages suitable for IoT, cryptographic libraries, and IoT development platforms.

Experimental Scenarios:

- Simulate IoT device communication and data storage scenarios in a smart city environment.
- Test the efficiency of authentication and key agreement mechanisms.

Testing Goals and Conducts:

- Goal: Ensure the secure functionality of the IoT network protocol and identify potential vulnerabilities.
- Conduct:
  - Real-time testing of device communication.
  - Evaluate the efficiency of authentication and key agreement mechanisms.

Deployment:

- Deploy the enhanced IoT network protocol in a controlled smart city environment to assess its real-world performance and security.

References:

- Cite academic papers, textbooks, and research articles on secure network protocols, IoT security, and smart city applications.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in IoT network protocol security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing authentication, end-device security, and data sink protection algorithms.
3. **Solution Architecture (15%):** The clarity and effectiveness of the IoT network protocol's design.

4. **Functional Features (20%):** The effectiveness of authentication, end-device security, and data sink protection mechanisms.
5. **Security Features (20%):** How well the student has addressed authentication and authorization, end-device security, and data sink security.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 9. Secure Commercial Transactions in Online Shopping and Payment

Application Scenarios:

1. **Gaps:** Identify security vulnerabilities and gaps in the commercial transaction process, especially in online shopping and payment systems.
2. **Reliable Arguments for the Gaps:** Provide strong evidence and arguments to support the identified security vulnerabilities, citing real-world examples of payment fraud and data breaches.
3. **Motivations:** Emphasize the motivation for securing commercial transactions in the online shopping and payment domain, considering the financial and personal data involved.
4. **Desired Functional and Security Features:**
    - Identity-based public key authentication for secure user access.
    - Strong data encryption for payment data and user information.
    - Secure digital signatures for transaction integrity and authenticity.
5. **Related Stakeholders:** Identify stakeholders, including online shoppers, e-commerce platforms, payment processors, and regulatory bodies.

Determine Specific Algorithms:

- **Identity-Based Public Key for Authentication and Encryption:**
    - Implement identity-based public key cryptography for secure user authentication and data encryption.
- **Post-Quantum Digital Signature (CRYSTALS-Dilithium):**
    - Utilize post-quantum cryptography for secure transaction digital signatures.

Solutions:

Solution Architecture:

- Implement a secure commercial transaction system for online shopping and payment.
- Components include identity-based public key authentication, data encryption, and secure digital signatures.

Detail of Functional Features:

- **Identity-Based Public Key Authentication:**
    - Implement identity-based public key cryptography for secure user authentication.
- **Data Encryption:**
    - Use strong encryption algorithms to protect payment data and user information.
- **Secure Digital Signatures:**
    - Implement CRYSTALS-Dilithium for secure transaction digital signatures.

Detail of Security Features:

- **Authentication:**
    - Identity-based public key authentication to ensure secure user access.
- **Data Protection:**

    - Encryption to protect payment data and user information.
- **Transaction Integrity:**
    - CRYSTALS-Dilithium digital signatures for ensuring transaction integrity and authenticity.

Implementation and Testing:

Tools or Libraries Recommendation:

- Utilize programming languages suitable for secure transactions, cryptographic libraries, and secure payment gateways.

Experimental Scenarios:

- Simulate online shopping and payment transactions with a focus on secure authentication, data encryption, and transaction signing.

Testing Goals and Conducts:

- Goal: Ensure the secure functionality of the commercial transaction system and identify potential vulnerabilities.
- Conduct:
    - Real-time transaction testing with secure payment processing.
    - Evaluate the efficiency of identity-based authentication and encryption.

Deployment:

- Deploy the enhanced commercial transaction system in a controlled online shopping and payment environment to assess its real-world performance and security.

References:

- Cite academic papers, textbooks, and research articles on identity-based public key cryptography, CRYSTALS-Dilithium, and online payment security.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in online shopping and payment security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing identity-based public key authentication, encryption, and CRYSTALS-Dilithium digital signatures.
3. **Solution Architecture (15%):** The clarity and effectiveness of the commercial transaction system's design.
4. **Functional Features (20%):** The effectiveness of identity-based authentication, data encryption, and secure digital signatures.
5. **Security Features (20%):** How well the student has addressed authentication, data protection, and transaction integrity.

6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 10. Cloud-Native API-Based Network Application Security for Small Company Services

Application Scenarios:

1. **Gaps:** Identify security vulnerabilities and gaps in cloud-native API-based network applications for small company services like KiotViet.
2. **Reliable Arguments for the Gaps:** Provide strong evidence and arguments to support the identified security vulnerabilities, citing real-world examples of API breaches and data exposure risks.
3. **Motivations:** Emphasize the motivation for securing cloud-native network applications, considering the importance of data protection and privacy.
4. **Desired Functional and Security Features:**
    - Authentication and key agreement for secure user access.
    - Authorization mechanisms to control access to different resources and services.
    - Efficient session management for user interactions.
    - Secure backend database for data storage.
5. **Related Stakeholders:** Identify stakeholders, including the small company, users, and application administrators.

Determine Specific Algorithms:

- **Authentication and Key Agreement:**
    - Implement secure authentication and key agreement mechanisms for user access.
- **Authorization:**
    - Apply robust authorization mechanisms to control access to resources.
- **Session Management:**
    - Efficiently manage user sessions and interactions.
- **Secure Backend Database:**
    - Implement encryption and access control to secure data stored in the backend database.

Solutions:

Solution Architecture:

- Implement a cloud-native network application with a layered security architecture.
- Components include authentication, authorization, session management, and backend database security.

Detail of Functional Features:

- **Authentication and Key Agreement:**
    - Implement secure authentication mechanisms.
    - Establish secure key agreements for data encryption.
- **Authorization:**
    - Apply role-based or attribute-based authorization to control access to resources.
- **Session Management:**

    - Efficiently manage user sessions, including login, logout, and session expiration.
- **Secure Backend Database:**
    - Implement encryption and access control mechanisms to protect data at rest.

Detail of Security Features:

- **Authentication:**
    - Strong user authentication to prevent unauthorized access.
- **Authorization:**
    - Robust access control to limit access to authorized users.
- **Session Management:**
    - Ensure secure session handling to prevent session hijacking.
- **Database Security:**
    - Implement encryption and access control for data stored in the backend database.

Implementation and Testing:

Tools or Libraries Recommendation:

- Utilize programming languages suitable for cloud-native applications, cryptographic libraries, and API development tools.

Experimental Scenarios:

- Simulate user interactions and API calls in the cloud-native network application.
- Test the efficiency of authentication, authorization, session management, and backend database security.

Testing Goals and Conducts:

- Goal: Ensure the secure functionality of the cloud-native network application and identify potential vulnerabilities.
- Conduct:
    - Real-time testing of user interactions and API calls.
    - Evaluate the efficiency of security mechanisms, including authentication, authorization, and session management.

Deployment:

- Deploy the enhanced cloud-native network application in a controlled environment to assess its real-world performance and security.

References:

- Cite academic papers, textbooks, and research articles on authentication and key agreement, authorization, session management, and database security in network applications.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in cloud-native network application security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing authentication, authorization, session management, and database security algorithms.
3. **Solution Architecture (15%):** The clarity and effectiveness of the network application's design.
4. **Functional Features (20%):** The effectiveness of authentication, authorization, session management, and backend database security mechanisms.
5. **Security Features (20%):** How well the student has addressed authentication, authorization, session management, and data storage security.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 11. Public Administrative Services via Citizen Services Portal

Application Scenarios:

1. **Gaps:** Identify security vulnerabilities and gaps in the provision of public administrative services through a citizen services portal.
2. **Reliable Arguments for the Gaps:** Provide strong evidence and arguments to support the identified security vulnerabilities, with references to real-world breaches in public administrative services.
3. **Motivations:** Emphasize the importance of securing public administrative services due to the sensitive nature of government-related data and the need for privacy and data protection.
4. **Desired Functional and Security Features:**
    - Secure digital signatures for document validation.
    - Efficient authentication and key agreement mechanisms.
    - Enhanced user access control and gather data using QR codes.
5. **Related Stakeholders:** Identify stakeholders, including government agencies, citizens, and administrators of the citizen services portal.

Determine Specific Algorithms:

- **Post-Quantum Digital Signature (FALCON):**
    - Utilize post-quantum cryptography for secure digital signatures on documents and transactions.
- **QR Codes:**
    - Use QR codes for user authentication and access control and gather data.
- **Authentication and Key Agreement:**
    - Implement robust authentication and key agreement mechanisms for secure user access.

Solutions:

Solution Architecture:

- Develop a citizen services portal with strong security features.
- Components include document digital signing, QR code-based authentication, and secure user access controls.

Detail of Functional Features:

- **Digital Signatures:**
    - Implement FALCON digital signatures for document validation and integrity assurance.
- **QR Code Authentication:**
    - Use QR codes for secure and efficient user authentication.
- **Authentication and Key Agreement:**
    - Implement secure authentication and key agreement mechanisms for user access.

Detail of Security Features:

- **Digital Signatures:**
  - Ensure the integrity and authenticity of documents and transactions using FALCON digital signatures.
- **QR Code Authentication:**
  - Secure user authentication and access control, gather data using QR codes.
- **Authentication and Key Agreement:**
  - Robust user authentication and secure key agreement for user access.

Implementation and Testing:

Tools or Libraries Recommendation:

- Utilize programming languages suitable for web development, cryptographic libraries, and QR code generation tools.

Experimental Scenarios:

- Simulate citizen interactions with the portal, including document signing and authentication.
- Test the efficiency of FALCON digital signatures and QR code-based authentication, QR storage.

Testing Goals and Conducts:

- Goal: Ensure the secure functionality of the citizen services portal and identify potential vulnerabilities.
- Conduct:
  - Real-time testing of document signing and authentication processes.
  - Evaluate the efficiency of digital signatures and QR code-based authentication.

Deployment:

- Deploy the enhanced citizen services portal in a controlled government environment to assess its real-world performance and security.

References:

- Cite academic papers, textbooks, and research articles on post-quantum cryptography, QR codes, authentication, and digital signatures in government services.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in public administrative services' security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing FALCON digital signatures, QR codes, and authentication mechanisms.

3. **Solution Architecture (15%):** The clarity and effectiveness of the citizen services portal's design.
4. **Functional Features (20%):** The effectiveness of digital signatures, QR code authentication, and user access control mechanisms.
5. **Security Features (20%):** How well the student has addressed document integrity, user authentication, and access control.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation 10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 12. Attribute-based Encryption for Healthcare Systems

Application Scenarios:

- Access control systems where permissions are based on user attributes (e.g., role, department, seniority).
- Secure data storage in cloud environments where data access is contingent on user attributes.
- Encrypted content distribution, where content is only accessible to users with specific attributes.
- Your specific domains (eg. healthcare systems)

Example:

Application Scenarios Healthcare Systems:

- **Electronic Health Records (EHR)**: Secure storage and sharing of patient medical records, where access is granted based on roles (e.g., doctor, nurse, pharmacist) or specific attributes (e.g., cardiology department, emergency staff).
- **Medical Research**: Sharing of anonymized patient data for research purposes, where access is contingent on researcher attributes (e.g., oncology researcher, clinical trial member).
- **Telemedicine**: Secure communication between patients and healthcare providers, ensuring that only authorized personnel can access transmitted data based on their attributes.
-

Gaps, Motivations, and Desired Security Features:

- **Gaps**: Traditional encryption schemes focus on encrypting data for a specific recipient. They don't inherently support complex access control based on attributes.
- **Motivations**: Need for flexible encryption schemes that allow data access based on multiple attributes rather than a single recipient identity.
- **Desired Security Features**:
  - Fine-grained access control based on user attributes.
  - Resistance to collusion attacks (where multiple users combine their attributes to gain unauthorized access).
  - Efficient encryption and decryption processes.

Proposed Solutions:

- **Solution Architecture**:
  - **Attribute Authority**: Entity responsible for distributing secret keys based on user attributes.
  - **Encryption/Decryption Module**: Encrypts data based on an access policy and decrypts using secret keys corresponding to user attributes.
- **Solution Details**:
  - Implement Ciphertext-Policy ABE (CP-ABE) where data is encrypted under an access policy, and users are given attributes that allow them to decrypt if they satisfy the policy.

    - Alternatively, use Key-Policy ABE (KP-ABE) where users have an access policy associated with their secret key and can decrypt data encrypted with matching attributes.

Implementation, Tools, and Testing:

- **Implementation**:
    - Use cryptographic libraries that support ABE, such as the PBC (Pairing-Based Cryptography) library.
- **Tools**: Libraries like Charm-Crypto that offer tools for designing and testing ABE schemes.
- **Testing**:
    - **Functional Testing**: Ensure data encrypted under an access policy can be correctly decrypted by users with matching attributes.
    - **Security Testing**: Test resistance against collusion attacks and other known vulnerabilities in ABE.

Deployment:

- Deploy in cloud storage systems to provide attribute-based access control for stored data.
- Integrate into corporate networks for fine-grained access control based on employee attributes.

References:

- Cite academic papers, textbooks, and research articles on authentication and key agreement, Ciphertext-Policy ABE, ABAC, Healthcare Systems, electronic health record (EHR).

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in cloud-native network application security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing authentication, authorization, session management, and database security algorithms.
3. **Solution Architecture (15%):** The clarity and effectiveness of the network application's design.
4. **Functional Features (20%):** The effectiveness of authentication, authorization, session management, and backend database security mechanisms.
5. **Security Features (20%):** How well the student has addressed authentication, authorization, session management, and data storage security.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 13. Application of Homomorphic Encryption for Financial Services

Application Scenarios:

- **Secure Financial Analytics**: Banks and financial institutions can perform computations on encrypted financial data without decrypting it, ensuring data privacy while gaining insights.
- **Credit Scoring**: Credit agencies can compute a person's credit score using encrypted financial data without ever accessing the raw, sensitive data.
- **Encrypted Financial Transactions**: Securely processing transactions on encrypted data, ensuring that transaction details remain confidential.

Gaps, Motivations, and Desired Security Features:

- **Gaps**: Traditional encryption methods require data to be decrypted before any computation, exposing sensitive financial data to potential threats.
- **Motivations**: Ensuring data privacy and regulatory compliance (e.g., GDPR, CCPA) while still being able to perform necessary financial computations.
- **Desired Security Features**:
  - Ability to perform computations on encrypted data.
  - Ensuring data integrity and authenticity.
  - Compliance with financial data protection regulations.

Proposed Solutions:

- **Solution Architecture**:
  - **Encryption Module**: Encrypts financial data using homomorphic encryption techniques.
  - **Computation Module**: Performs necessary computations directly on the encrypted data.
- **Solution Details**:
  - Use Fully Homomorphic Encryption (FHE) or Somewhat Homomorphic Encryption (SHE) based on the computational requirements and efficiency considerations.
  - Implement specific financial algorithms that are tailored to work with homomorphic encryption.

Implementation, Tools, and Testing:

- **Implementation**:
  - Integrate homomorphic encryption capabilities into existing financial systems or platforms.
- **Tools**: Cryptographic libraries like HElib or Microsoft's SEAL that offer homomorphic encryption capabilities.
- **Testing**:
  - **Functional Testing**: Simulate real-world financial scenarios to ensure computations on encrypted data are accurate and match those on raw data.

- **Security Testing**: Test for potential vulnerabilities, ensuring financial data remains confidential during computations.

Deployment:

- Deploy as part of financial analytics platforms in banks and financial institutions.
- Integrate into credit scoring systems to compute scores without accessing raw financial data.

References:

- Cite academic papers, textbooks, and research articles on Homomorphic Encryption, Financial Services

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in cloud-native network application security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing authentication, authorization, session management, and database security algorithms.
3. **Solution Architecture (15%):** The clarity and effectiveness of the network application's design.
4. **Functional Features (20%):** The effectiveness of authentication, authorization, session management, and backend database security mechanisms.
5. **Security Features (20%):** How well the student has addressed authentication, authorization, session management, and data storage security.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 14. Functional Encryption for Smart Cities

Application Scenarios:

- **Smart Traffic Management**: Analyzing encrypted traffic data to optimize traffic light timings or detect traffic anomalies without accessing raw data.
- **Energy Consumption Analysis**: Utility companies can compute the energy consumption patterns of encrypted data from smart meters without accessing individual consumption details.
- **Public Safety and Surveillance**: Analyzing encrypted surveillance footage to detect suspicious activities or patterns without decrypting the entire footage.

Gaps, Motivations, and Desired Security Features:

- **Gaps**: Traditional encryption methods either allow full access to data or no access at all, making it challenging to provide selective access based on specific functions or computations.
- **Motivations**: Ensuring citizen privacy in smart cities while still being able to perform necessary computations for city optimization and safety.
- **Desired Security Features**:
    - Ability to perform specific computations on encrypted data without full decryption.
    - Ensuring data integrity and authenticity.
    - Compliance with data protection regulations in smart city implementations.

Proposed Solutions:

- **Solution Architecture**:
    - **Encryption Module**: Encrypts city data using functional encryption techniques.
    - **Computation Module**: Performs specific computations directly on the encrypted data based on predefined functions.
- **Solution Details**:
    - Use functional encryption schemes that allow computations like sum, average, or pattern detection on encrypted data.
    - Implement city-specific algorithms tailored to work with functional encryption.

Implementation, Tools, and Testing:

- **Implementation**:
    - Integrate functional encryption capabilities into existing smart city platforms and infrastructure.
- **Tools**: Cryptographic libraries or platforms that offer functional encryption capabilities tailored for large-scale city data.
- **Testing**:
    - **Functional Testing**: Simulate real-world smart city scenarios to ensure computations on encrypted data are accurate and match those on raw data.
    - **Security Testing**: Test for potential vulnerabilities, ensuring city data remains confidential during computations.

Deployment:

- Deploy as part of traffic management systems to optimize traffic flow based on encrypted data.
- Integrate into utility management systems for energy consumption analysis without compromising user privacy.
- Use in public safety systems to analyze encrypted surveillance data for potential threats.

References:

- Cite academic papers, textbooks, and research articles on Functional Encryption, Smart Cities.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in cloud-native network application security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing authentication, authorization, session management, and database security algorithms.
3. **Solution Architecture (15%):** The clarity and effectiveness of the network application's design.
4. **Functional Features (20%):** The effectiveness of authentication, authorization, session management, and backend database security mechanisms.
5. **Security Features (20%):** How well the student has addressed authentication, authorization, session management, and data storage security.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.

Topic 15 Post-Quantum CRYSTALS-KYBER Encryption for Cloud-native storage

Application Scenarios:

- **Secure Data Storage**: Encrypting data before uploading to the cloud, ensuring that even with the advent of quantum computers, the data remains secure.
- **Quantum-Resistant Virtual Private Networks (VPNs)**: Establishing secure communication channels between cloud servers and clients or between different cloud servers.
- **Secure Multi-party Computations**: Performing computations on encrypted data in the cloud, where multiple parties are involved, without exposing the raw data.

Gaps, Motivations, and Desired Security Features:

- **Gaps**: Traditional encryption methods, like RSA and ECC, are potentially vulnerable to quantum attacks, making data encrypted with these methods at risk in the future.
- **Motivations**: Ensuring long-term data security in cloud environments, especially for data that will remain sensitive for many years (e.g., personal records, intellectual property).
- **Desired Security Features**:
    - Resistance to quantum attacks.
    - Efficient performance suitable for cloud operations.
    - Secure key exchange and data encryption.

Proposed Solutions:

- **Solution Architecture**:
    - **Key Generation and Management System**: Generate, distribute, and store CRYSTALS-KYBER key pairs.
    - **Encryption/Decryption Module**: Encrypts and decrypts data using CRYSTALS-KYBER before storage or transmission.
- **Solution Details**:
    - Use CRYSTALS-KYBER, a lattice-based cryptographic scheme, for key exchange and data encryption.
    - Implement protocols that are optimized for cloud operations, ensuring efficiency and security.

Implementation, Tools, and Testing:

- **Implementation**:
    - Integrate CRYSTALS-KYBER capabilities into cloud platforms and storage systems.
- **Tools**: Cryptographic libraries or platforms that offer post-quantum encryption capabilities, specifically CRYSTALS-KYBER implementations.
- **Testing**:
    - **Functional Testing**: Ensure data encrypted with CRYSTALS-KYBER can be correctly decrypted.
    - **Security Testing**: Simulate potential quantum attacks to test the resilience of the encryption scheme.

Deployment:

- Deploy as part of cloud storage solutions to ensure long-term data security.
- Integrate into VPN solutions for cloud platforms to establish quantum-resistant secure communication channels.
- Use in cloud-based multi-party computation platforms to ensure data privacy against future quantum threats.

References:

- Cite academic papers, textbooks, and research articles on authentication and key agreement, authorization, session management, and database security in network applications.

Assessment Rubric:

1. **Quality of Research (15%):** How well the student has identified and justified gaps in cloud-native network application security.
2. **Algorithm Selection (10%):** Appropriateness and justification for choosing authentication, authorization, session management, and database security algorithms.
3. **Solution Architecture (15%):** The clarity and effectiveness of the network application's design.
4. **Functional Features (20%):** The effectiveness of authentication, authorization, session management, and backend database security mechanisms.
5. **Security Features (20%):** How well the student has addressed authentication, authorization, session management, and data storage security.
6. **Implementation and Testing (10%):** The effectiveness of the testing process and its alignment with the proposed solution.
7. **Presentation and Documentation (10%):** Clarity, organization, and quality of presentation materials and project documentation.