ITIS 6200/8200 Principles of Information Security and Privacy

Final Exam

**Question 7**. **Confidentiality & Integrity (20 points)**

Alice wants to send messages to Bob in an insecure channel. The attackers may eavesdrop, intercept, and modify the messages sent in the channel. By combining the techniques that we have learned in this course, **develop two schemes** **that can ensure both integrity and confidentiality** of the communications. Describe how Alice produces the ciphertext and how Bob decrypts the ciphertext to read the message. Given a plaintext M, write down the formula of the ciphertext sent from Alice to Bob, and describe how Bob can decrypt C into M. Explain why the scheme provides both confidentiality and integrity.

You can make the following assumptions:

1. Alice and Bob already shared the secret key if they use symmetric key encryption.
2. Alice already knows Bob’s public key, and Bob already knows Alice’s public key.
3. Alice and Bob’s private keys are not compromised.
4. Every technique is secure with respect to its requirements, e.g., AES uses random IV or nonce.
5. Cryptographic tools do not interfere with each other when used in combination, e.g., the same key can be used for AES and MAC.

Q 7.1: Scheme #1

Q 7.2: Scheme #2

**Extra credit (5 points)**

We have learned other techniques in the course. Can you use these techniques to relax any assumption above? How?

**Question 2**. **Block Cipher Design (15 points)**

A student in our class suggests the following design of block cipher operating mode: the same IV is used for every block. The message M is split into j plaintext blocks M1 … Mj each of size *n*. The encryption mode outputs (IV, C1, …, Cj) as the overall ciphertext. Assume IV is randomly generated per encryption.

****

Q 2.1: Write down the encryption formula. That is, what is the formula for Ci (0 < i <= j) given (1) plaintext M1 … Mj (2) block cipher encryption Enc(K, M) which takes a key K and message M as inputs, and (3) a randomly generated IV. (You can use notation or XOR for exclusive OR)

Q 2.2: Write the decryption formula for Mi (0 < i <= j) using this mode. That is, how to get Mi (0 < i <= j) given (1) ciphertext (IV, C1, …, Cj) and (2) Decryption Dec(K, M).

Q 2.3: Is this mode IND-CPA secure? If yes, explain why; if not, describe how an attacker can break IND-CPA. That is, find two messages that if Eve sends to Alice for encryption and Alice randomly chooses one for encryption and sends back to Eve. Then Eve can find a way to tell which message Alice encrypts with a probability > 0.5

**Question 3**. **Hashing (10 points)**

Alice’s computer stores the files in the following way: for every file F, the computer will calculate the hash value of the file hash(F) and store it after the file, i.e., F || hash(F). Every time when Alice login, the machine will automatically hash all the files and compare the results to the stored hash values. In this way, if by accident the hard drive is mis-functioning and flips a few bits in a file, Alice can immediately detect it since the hash value will be different. Now an attacker hacks into Alice’s machine and he tries to change several files. The attacker also knows the hash function that the computer uses.

Q 3.1: Describe what the attacker needs to do so that the next time Alice login, the machine will not detect the changes.

Q 3.2: How do we improve the mechanism to prevent / detect such changes?

**Question 5**. **Public-Key Encryption (15 points)**

Bob has a public-private key pair (pub\_Bob, priv\_Bob). Alice needs to send some information to Bob. She wants to make sure that when Bob opens the message, he can verify that this is from Alice but not anyone else.

Q 5.1: If Alice sends out the message as: [ Alice || Epub\_Bob(message) ] to Bob. That is, she sends out her name in clear text, followed by ciphertext of the message encrypted with Bob’s public key. Can powerful attacker Mallory impersonate Alice and send out a packet in Alice’s name? How can she do it? Assume that Mallory also has the public key of Bob.

Q 5.2: If Alice sends out [ Epub\_Bob(Alice || message) ], where Alice puts her name in the encryption. Can Mallory still impersonate Alice?

Q 5.3: Design a way that Bob can ensure the message comes from Alice.

**Question 6**. **RSA Signature (15 points)**

To use RSA signatures on messages, we first create a RSA key pair: (N, e) is the RSA public key and d is the RSA private key, where N is the RSA modulus. For standard RSA signatures, we typically set e to a small prime value such as 5.

Q 6.1: For RSA signatures, we often sign the hash of a message, rather than the message directly.

Why is that?

Q 6.2: Assume that we **skip using a hash function**, and sign the messages directly. That means, if Alice wants to send a signed message to Bob, she will send (M, S) to Bob where S = Md mod N is computed using her private signing key d. With such a scheme, how does Bob verify this message come from Alice? What formula does Bob need?

Q 6.3: Mallory learns that Alice and Bob are using the simplified signature scheme that without using hash functions. Can Mallory find a (M, S) pair such that S will be a valid signature on M?

Assume that Mallory knows Alice’s public key N and e, but not Alice’s private key d.

**Question 1. Cryptography Mechanisms (20 points)**

**Draw lines between mechanism to building blocks for implementing these techniques.**

**Or**

**Question 1. Security Mechanism versus Security Properties**

**Draw lines between mechanism to functions implementing these techniques.**

**Question 1.a**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AES-CBC** | **MAC** | **RSA encryption** | **RSA signature** |
| Hash |  |  |  |  |
| PRNG |  |  |  |  |
| Modulo operation |  |  |  |  |
| XOR |  |  |  |  |

**Question 1.b**

|  |  |
| --- | --- |
|  | **Man-In-The-Middle Attack** |
| **AES-CBC** |  |
| **MAC** |  |
| **RSA encryption** |  |
| **RSA signature** |  |
| **Diffie-Hellman Key Exchange** |  |
|  |  |

**Question 1.b**

* **You can use:**
  + **An IND-CPA encryption scheme (e.g. AES-CBC): Enc(*K*, *M*) and Dec(*K*, *M*)**
  + **An unforgeable MAC scheme (e.g. HMAC): MAC(*K*, *M*)**
* **First attempt: Alice sends Enc(*K*1, *M*) and MAC(*K*2, *M*)**
  + **Integrity? Yes, attacker can’t tamper with the MAC**
  + **Confidentiality? No, the MAC is not IND-CPA secure**
* **Idea: Let’s compute the MAC on the *ciphertext* instead of the plaintext:  
  Enc(*K*1, *M*) and MAC(k2, Enc(*K*1, *M*))**
  + **Integrity? Yes, attacker can’t tamper with the MAC**
  + **Confidentiality? Yes, the MAC might leak info about the ciphertext, but that’s okay**
* **Idea: Let’s encrypt the MAC too: Enc(*K*1, *M* || MAC(*K*2, *M*))**
  + **Integrity? Yes, attacker can’t tamper with the MAC**
  + **Confidentiality? Yes, everything is encrypted**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Confidentiality** | **Integrity** | **Authenticity** | **Availability** |
| **AES** |  |  |  |  |
| **MAC** |  |  | | |
| **RSA encryption** |  |  | | |
| **RSA signature** |  |  | | |
| **Hash function** |  |  | | |
| Diffie-Hellman Key Exchange |  |  | | |
| PRNG |  |  | | |
| Certificate |  |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Confidentiality** | **Integrity** | **Availability** |
| **MITM** |  |  |  |
| **XSS** |  |  | |
| **CSRF** |  |  | |
| Distributed Denial of Service (DDoS) |  |  | |
| Denial of Service |  |  | |
| SQL Injection Attack |  |  | |
| Buffer Overflow Attack |  |  | |
| TCP spoofing attack |  |  | |
| SYN flooding |  |  | |
| ARP spoofing attack |  |  | |

* CSRF
* CSRF token and Referer
* HTML sanitization
* Content Security Policy (CSP)
* reflected XSS
* SQL injection
* ARP spoofing
* TCP data injection
* TCP spoofing
* DoS / DDoS
* SYN flooding
* SYN cookies
* Firewall
* Packet Filters
* Path Traversal Attack
* Intrusion Detection
* Buffer Overflow

**Question 2. Block Cipher and IND-CPA (20 points) Symmetric and Asymmetric Encryption?**

**Question 3. Integrity Hash MAC HMAC**

**Question 4. Confidentiality & Integrity**

**Question 5. Access Control (Chinese Wall)**

**Question 6. Web Security (20 points)**

**Question 7. Network Security (20 points)**

**Question 8. System Security (20 points)**

****

* AES: Enc(K, M) and Dec(K, C) where K, M, and C denote the key, plaintext, and ciphertext
* HMAC: HMAC(K, V) where K, V are the inputs to the HMAC function
* Hash: Hash(M) where M is the input to the hash function
* RSA public-key encryption: Enc(pub-Bob, M), and Dec(priv-Bob, C).
* RSA signature: Enc(priv-Alice, M) and Dec(priv-Alice, C)
* Diffie-Hellman Key Exchange
* Certificate and Certificate Authority
* Access Control (different models)
* CSRF
* CSRF token and Referer
* HTML sanitization
* Content Security Policy (CSP)
* reflected XSS
* SQL injection
* ARP spoofing
* TCP data injection
* TCP spoofing
* DoS / DDoS
* SYN flooding
* SYN cookies
* Firewall
* Packet Filters
* Path Traversal Attack
* Intrusion Detection
* Buffer Overflow

**Question 8. Memory Vulnerability (20 points)**

Consider the following vulnerable C code. Assume you are on a little-endian 32-bit x86 system and no memory safety defenses are enabled.

A screen shot of a computer program

Description automatically generated

Q 5.1: Assume that execution has reached line 6. Fill in the following stack diagram. Assume that each row represents 4 bytes. Note that arrays are filled from lower addresses to higher addresses and are zero-indexed.

***Stack***

|  |
| --- |
|  |
|  |
| RIP of **foo** |
| SFP of **foo** |
|  |
|  |
|  |
|  |

Q 5.2: Assume that the address of the RIP of **foo** is 0x12345678.

Construct an input to gets that would cause the program to execute malicious shellcode. You may reference SHELLCODE as a 12-byte shellcode.