650.445: Practical Cryptographic Systems March 25, 2010

Midterm

Instructor: Matthew Green

Name: _____

As with any exam, please do not collaborate or otherwise share information with any other person. You *are* permitted to use printed notes and textbooks, but computer and Internet use is not permitted.

Problem 1: True or False (10 points)
You do not need to justify your answer.

1. True or False: The CBC mode of operation is preferred for many applications because it lends itself naturally to parallelization.

2. True or False: If the RSA-PSS signature scheme — implemented with SHA1 as the hash function — is broken (*e.g.,* an efficient algorithm is found that wins the EU-CMA game with 100% probability), then all protocols based on the RSA assumption are necessarily broken.

3. True or False: Even if a compiler disables the %n formatting option for printf()-type functions, it may be possible for an attacker to read system memory using a format string attack.

4. True or False: Most modern smartcards include protections against DPA and SPA attacks.

5. True or False: Based on the current state of the art, doubling the length of an RSA modulus N (*e.g.,* from 1024 bits to 2048 bits) should roughly double the effort required to factor it.

6. True or False: Daniel Bleichenbacher's 1998 attack (on SSL implementations using the PKCS #1v1.5 encryption padding scheme) allows the attacker to recover the RSA secret exponent used by an SSL server after several million queries.

7. True or False: The AES block cipher is based on a Feistel-network design.

8. True or False: The SSL/TLS specification supports an ephemeral key agreement protocol based on RSA encryption.

9. True or False: In a reduction proof, we show that the existence of an algorithm for solving a hard mathematical problem implies an algorithm for breaking a cryptosystem.

10. True or False: Both client- and server-side certificates are necessary to prevent man-in-the-middle attacks on TLS.

Problem 2: Short Answers (15 points)

1. We discussed two significant challenges that make it difficult to prove that the RSA-PKCS #1v1.5 signature scheme is EU-CMA under the RSA assumption. Recall that the RSA problem is to compute, given an instance (N, e, c), a value m such that $m^e \equiv c \; mod \; N$. Briefly, what were the two problems we identified?

2. **The Clipper/Capstone Chip.** In general terms, explain the escrow functionality of of the Clipper chip and explain why it was so important to keep the cipher and related algorithms secret.

3. Describe two significant physical protections offered by a Hardware Security Module such as the IBM 4758 and explain what attacks they defend against.

Problem 3: Chosen Ciphertext Security for Symmetric Encryption. (25 points)

Recall the definition of adaptive-chosen ciphertext security (IND-CCA2), which is a game played between an adversary and an honest challenger.

In the case of symmetric encryption (where there is no public key), we allow the adversary to obtain the encryptions of chosen messages. Let $\mathsf{Encrypt}, \mathsf{Decrypt}$ be the encryption and decryption algorithms for the scheme. The IND-CCA2 game is described below:

1. The challenger randomly selects a secret key k.
2. The adversary adaptively makes any number of the following queries to the challenger:
   (a) The adversary can submit M and receive $\mathsf{Encrypt}(k, M)$.
   (b) The adversary can submit C, and receive $\mathsf{Decrypt}(k, C)$.
3. The adversary outputs equal-length messages M_0, M_1.
4. The challenger picks a random $b \in \{0, 1\}$ and returns $C^* = \mathsf{Encrypt}(k, M_b)$.
5. The adversary again makes queries to the challenger as in step (2), but cannot ask for the decryption of C^*.[1]
6. The adversary outputs a guess b'.

The adversary wins if it guesses correctly ($b = b'$). The scheme is IND-CCA2 secure if no (efficient) adversary wins with probability significantly greater than $1/2$.

Please answer the following two questions.

[1] The adversary can ask for the decryption of any other ciphertext, even if it is one bit different from C^*.

Part 1. (15 points) Imagine that we are considering CBC-mode encryption with a secure block cipher (defined by E and D). The Encrypt algorithm picks a new random IV, parses M into blocks $M_1, \ldots, M_N$ and outputs:

$$\mathsf{Encrypt}(k, M_1, \ldots, M_N) = IV, C_1, \ldots, C_N$$

$$\text{where } C_1 = E(k, M_1 \oplus IV) \text{ and for every } i > 1, C_i = E(k, M_i \oplus C_{i-1})$$

And the Decrypt algorithm is defined by:

$$\mathsf{Decrypt}(k, IV, C_1, \ldots, C_N) = M_1, \ldots, M_N$$

$$\text{where } M_1 = D(k, C_1) \oplus IV \text{ and for every } i > 1, M_i = D(k, C_i) \oplus C_{i-1}$$

If (E, D) is a secure block cipher, we claim that CBC mode encryption is IND-CPA secure. However, can you describe an attack whereby an adversary would be able to win the IND-CCA2 game above with probability significantly greater than $1/2$?

Part 2. (10 points) Imagine that the encryption algorithm adds a secure Message Authentication Code (MAC) to authenticate each CBC-mode ciphertext. The decryption algorithm would check the MAC and reject any ciphertext that does not possess a valid MAC (outputting an ERROR message). More formally:

Let Encrypt, Decrypt be the CBC scheme described above. Let MAC be the tagging algorithm for a deterministic EU-CMA secure Message Authentication Code such as HMAC. Let us define a new encryption scheme where the secret key $k = k_1||k_2$ and $\mathsf{NewEncrypt}(k_1||k_2, M)$ is defined as follows:

1. Compute $C' = \mathsf{Encrypt}(k_1, M)$.
2. Output $C = C'||\mathsf{MAC}(k_2, C')$.

And $\mathsf{NewDecrypt}(k_1||k_2, C)$ is defined as:

1. Parse C as $C'||T$.
2. If $T \neq \mathsf{MAC}(k_2, C')$ abort and output ERROR.
3. Otherwise output $M = \mathsf{Decrypt}(k_1, C')$.

Does this modified scheme solve the problems you identified in Part 1? How?

Problem 4: Specifications and Implementation (25 points)

A major airline manufacturer has developed a system for remotely controlling airplanes in flight, in the event that a pilot becomes incapacitated.[2] The system uses a digital wireless transmitter to send short control messages from the ground to the airplane. For obvious reasons Airbus has decided to cryptographically authenticate all of these commands using a secure MAC.

The design specification calls for the following:

> All messages transmitted from ground station to the aircraft are to be cryptographically authenticated using a secure Message Authentication Code (HMAC-SHA256) under a shared key k. A message consists of a 2-byte command opcode (*e.g.,* ‘‘turn rudder’’), followed by up to 10 optional 4-byte parameter fields, and is preceded by a 32-byte HMAC-SHA256 tag T:
>
> T (32 bytes) || Opcode (2 bytes) || Param 1 (4 bytes) || ... || Param N
>
> To verify the correctness of a message:
>
> 1. Parse the incoming message as $T||M$, and check that $T = \mathsf{HMAC\text{-}SHA256}(k, M)$.
> 2. If the MAC check fails, the receiver must ignore the message, otherwise it should send the message to the autopilot which will treat it as a valid command.

Our threat model assumes that the attacker can inject, intercept, block and even modify messages transmitted between the legitimate transmitter and the airplane. The attacker can presumably jam all communications, but this kind of DoS is outside of our model.

[2] This is totally made up.

Part 1 (5 points). Aside from full-scale jamming, do you see any immediate problems with this specification that might allow an attacker to take control of the airplane?

Part 2 (20 points). The MAC verification is coded within the following routine.

```
void validate_message(size_t message_size, char *message_data)
{
   // MIN_LENGTH and MAX_LENGTH are correctly defined in header files
   if (message_size < MIN_LENGTH || message_size > MAX_LENGTH) {
       return -1; // invalid message
   }

   // Copy the message into a global buffer.  This pre-allocated buffer
   // is a shared location that can be accessed by the autopilot system.
   memcpy(message_data, global_buffer, message_size);

   // Validate the MAC.
   if (Validate_MAC(global_key, global_buffer, message_size) == FALSE) {
     return; // Invalid MAC
   }

   // Tell the autopilot to process an authenticated command message
   // is waiting in global_buffer
   autopilot_process_new_command();
}
```

Can you identify a flaw in the above implementation that could create a vulnerability *beyond* any you identified in Part 1? You should can assume that the `Validate_MAC()` routine works correctly, that keys are set up, and that all global variables have been correctly allocated and initialized.

Problem 5: Protocols (25 points)

Skynet Inc. has proposed the following communication protocol for controlling their T-101 mobile device. The parties initially agree on a secret key k using a secure key agreement protocol.[3] Each party then generates a random 128-bit nonce value $(N1, N2)$, and the pair conduct a "handshake".

If both parties receive the expected response, then Skynet proceeds to transmit critical control messages $(M1, M2, \dots)$ by authenticating each using the MAC (under key k). If either party does *not* receive the correct response to the handshake (*e.g.,* due to interference), they can generate new nonces and repeat the handshake process up to 3 times without changing k. After three failed attempts, the they will have to re-run the key agreement protocol and generate a new value of k.

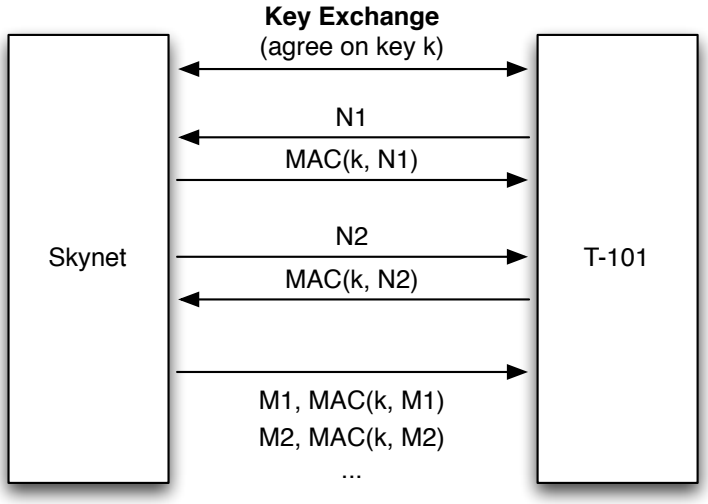

Figure 1: *Handshake and communication between two devices. The first stage is a key agreement protocol secure against active attacks.* $N1$ *and* $N2$ *are 128-bit random nonces.* MAC *is a secure Message Authentication Code.*

[3] Let's assume this protocol is secure against all possible attacks.

Assume that the key agreement protocol is secure (even against man-in-the-middle attacks), and MAC is a secure Message Authentication Code. Please answer the following:

1. (5 points) From a design perspective, what is the *purpose* of the nonces $N1, N2$ and the protocol elements that use them? (Why not just skip to communication?)

2. (20 points) Describe a security problem introduced by this handshake, and how it might be used by an active attacker to send bogus control messages to the T-101. Propose a fix for the attack.