University of Washington – T. Kohno and J. Manferdelli
Homework 6

Due. 4:30pm February 22, 2007.

Turn-in instructions. See the course website (`http://www.cs.washington.edu/education/courses/csep590b/07wi/`) for instructions on how to submit your homework via the UW Catalyst Tools. For this assignment, you should submit a PDF file named 'YourLastName-YourFirstInitial-HW6.pdf. Please write your name on the first page. The entire assignment is worth 25 points and there is an opportunity for 8 extra credit points.

Homework overview. We've talked a lot in class about cryptography, and about how easy it is to accidentally misuse a cryptosystem. This homework will give you an opportunity to further explore just how subtle the design of cryptosystems can be.

Notation. We first need some notation. The notation may look complex at first, but when dealing with cryptography it's much better to be precise. (As an aside, lots of security problems can arise because of different interpretations of non-precise definitions.)

If x and y are strings, then $x\|y$ denotes their concatenation. Let $\{0,1\}^a$ denote the set of all a-bit strings. Let $x \xleftarrow{\$} \{0,1\}^a$ denote the process of assigning x a *random* element from the set $\{0,1\}^a$. Let $x \leftarrow y$ denote the assignment of y to x. Let $x \oplus y$ denote the exclusive or (xor) of x and y.

Let $\text{AES}: \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ denote the AES block cipher, and let $\text{AES}_K(M)$ be short-hand for $\text{AES}(K,M)$. For each key K, $\text{AES}_K(\cdot)$ is actually a *permutation* on $\{0,1\}^{128}$; let $\text{AES}_K^{-1}(\cdot)$ denote the inverse of this permutation. Basically, this means that for all $M \in \{0,1\}^{128}$, if $C = \text{AES}_K(M)$, then $M = \text{AES}_K^{-1}(C)$.

CTR and CBC mode encryption schemes. We can now define the AES-CTR (Counter) mode encryption and decryption operations as follows. Note below that our simplified AES-CTR mode scheme is only designed for 128-bit messages.

Algorithm CTR-ENC(K,M) // defined for $M \in \{0,1\}^{128}$.
  $c \xleftarrow{\$} \{0,1\}^{128}$ // pick a 128-bit random IV (initialization vector)
  $S \leftarrow \text{AES}_K(c)$ // S gets encipherment of c with key K; S is the *keystream*
  $C' \leftarrow S \oplus M$ // xor S and M to get C'
  $C \leftarrow c\|C'$ // the ciphertext
  Return C

Algorithm CTR-DEC(K,C) // defined for $C \in \{0,1\}^{256}$.
  Break C into two 128-bit chunks c and C' // $c\|C' = C$
  $S \leftarrow \text{AES}_K(c)$ // S gets encipherment of c with key K; *reconstruct the keystream*
  $M \leftarrow S \oplus C'$ // xor S and C' to get back M
  Return M

Let's also define AES-CBC, an AES-based CBC (cipher block chaining) encryption scheme designed to take 128-bit messages as input.

Algorithm CBC-ENC(K,M) // defined for $M \in \{0,1\}^{128}$.
  $c \xleftarrow{\$} \{0,1\}^{128}$ // pick a 128-bit random IV
  $C' \leftarrow \text{AES}_K(c \oplus M)$ // C' gets encipherment of c xor M under key K
  $C \leftarrow c\|C'$ // the ciphertext
  Return C

Algorithm CBC-DEC(K, C) // defined for $C \in \{0,1\}^{256}$.
  Break C into two 128-bit chunks c and C' // $c \| C' = C$
  $M \leftarrow \text{AES}_K^{-1}(C') \oplus c$ // Recover M
  Return M

Technically, AES-CTR and AES-CBC above *provably preserve privacy under chosen plaintext attacks assuming that AES is a secure block cipher*. For this assignment, you don't need to know exactly why AES-CTR and AES-CBC are secure. All that's important to know is that a cryptographer would say that AES-CTR and AES-CBC are secure. So people might — and do — end up using them in a larger system. We'll see, however, that larger systems using these secure components may actually still be insecure.

Message authentication codes. Let's now define the AES-CBC-MAC message authentication code (MAC). We sometimes called these "signature" schemes in the lecture. Recall that the goal of a MAC is to be able to cryptographically protect the integrity of a message. Our version of AES-CBC-MAC is *very very very* simple — it's only defined for message in $\{0,1\}^{128}$.

Algorithm CBC-MAC(K, M) // defined for $M \in \{0,1\}^{128}$.
  $T \leftarrow \text{AES}_K(M)$ // T, which we call the *tag*, is just the encipherment of M
  Return T

Algorithm CBC-VF(K, M, T) // defined for $M \in \{0,1\}^{128}$.
  $T' \leftarrow \text{AES}_K(M)$
  If $T = T'$ then return true // The tag T matches the recomputed one T'
  Else return false // the tag T does not verify

While the books talk about MACs, let's just summarize the key properties here. We assume that two parties, Alice and Bob, share the same key K. If Alice wants to send a 128-bit message M to Bob and doesn't want anyone to be able to undetectably modify the message in transit, she would compute $T \leftarrow \text{CBC-MAC}(K, M)$ and then send the pair (M, T) to Bob.

Bob, on input some message tag pair (M^*, T^*), would compute CBC-VF(K, M^*, T^*) and accept M^* as from Alice only if the response from CBC-VF is true. Using AES the way we do above makes it virtually impossible for an attacker to make Bob accept a message M^* that Alice didn't already send to Bob.

The construction above is a provably secure message authentication scheme, but for the purposes of this assignment don't need to know exactly *why*.

The model. So far we've defined AES-CTR, AES-CBC, and AES-CBC-MAC. The first two are encryption schemes — *designed to protect the privacy of messages sent from Alice to Bob from an adversary that sniffs on the connection but does not modify the connection* — i.e., AES-CTR and AES-CBC provide privacy against a *passive* listener. AES-CBC-MAC, on the other hand, *provides integrity against an active attacker*.

So the natural question arises: *can one combine AES-CTR or AES-CBC with AES-CBC-MAC to provide privacy* and *integrity against an active attacker*? Here's the model. The attacker, Mallory, has full control over the channel between Alice and Bob. If Alice sends a ciphertext C across the wire, Mallory could decide to give C to Bob. Or Mallory could decide to give Bob some other value C'. In all cases, Mallory gets to see Bob's reaction — does Bob accept the message as valid, or does Bob reject the message? Is it possible to preserve privacy even when giving the adversary more power? And can do we still provide the integrity property? Let's make this more formal with a few problems.

Problem 6.1 (6 points). Let's look at one possible way to combine AES-CBC and AES-CBC-MAC. Here's the combined encryption and decryption algorithms. Note that unlike a basic encryption algorithm, this one can reject messages that it thinks might be forgeries.

Algorithm NEW-ENC(K, M) // defined for $K \in \{0,1\}^{256}$ and $M \in \{0,1\}^{128}$.
  Break K into two 128-bit keys K_1 and K_2 // $K_1 \| K_2 = K$
  $C^* \leftarrow$ CBC-ENC(K_1, M) // encrypt M with AES-CBC to get back C
  $T \leftarrow$ CBC-MAC(K_2, M) // MAC M with AES-CBC-MAC to get a tag T
  $C \leftarrow C^* \| T$ // the ciphertext
  Return C

Algorithm NEW-DEC(K, C) // defined for $K \in \{0,1\}^{256}$ and $C \in \{0,1\}^{384}$.
  Break K into two 128-bit keys K_1 and K_2 // $K_1 \| K_2 = K$
  Break C into a 256-bit portion C^* and a 128-bit portion T // $C^* \| T = C$
  $M \leftarrow$ CBC-DEC(K_1, C^*) // decrypt the AES-CBC portion of the ciphertext
  $f \leftarrow$ CBC-VF(K_2, M, T) // check whether T is a valid MAC of M
  If $f = \mathsf{true}$ then return M
  Else return an error code.

Let's start off with an attack that works even when the adversary is passive. In particular, assume an adversary has passively captured two ciphertexts C_1 and C_2 output from NEW-ENC.

- Task: Explain how the passive adversary could determine whether C_1 and C_2 are the encryption of the same message or different messages. This could be a problem if the messages come from some small message space, like "buy" or "sell" or "fire" and "don't fire."

Note that your answer to the above may be a bit surprising since, when used by itself, AES-CBC does provide privacy against passive adversaries.

Problem 6.2 (4 points). OK, let's make it a bit harder. Now assume that the first 16 bits of each message is reserved for a counter. I.e., the first message Alice encrypts would have 16 zero bits followed by $128 - 16 = 112$ bits for the actual data Alice sends. The second message Alice encrypts would have 15 zero bits, a one bit, and then Alice's data. The third message would have 14 zero bits, a one bit, a zero bit, and Alice's data. And so on.

- Task: Explain how a passive adversary could still compromise the privacy of Alice's payload data (i.e., learn more than Alice might like). How many ciphertexts does the adversary need to passively sniff in order to compromise privacy?

Problem 6.3 (8 *extra credit* points). Lets assume that we're now using the above tweak with a 16 bit counter, and assume that the adversary has only intercepted the first two ciphertexts sent by Alice. The adversary knows the 16 bit counter values used when creating these ciphertexts.

- Task: Given the first two ciphertexts C_1 and C_2 produced by Alice, show how the adversary could learn whether C_1 and C_2 are the encryption of the same 112 bits of payload data or not. Recall: in our security model the adversary gets to learn whether NEW-DEC returns an error or not.

Hint. You may have other solutions, but in mine the adversary takes C_1 and C_2 as input and produces a new ciphertext C_3 (the adversary does this without knowing any secret key). The

adversary then asks Bob to decrypt C_3 and looks at Bob's response (did Bob accept C_3 as a valid ciphertext from Alice, or did Bob reject C_3 as an invalid ciphertext). From looking at Bob's reaction, the adversary can determine whether the payload messages corresponding to C_1 and C_2 are the same or not.

Problem 6.4 (5 points). In Kerberos AP_REQ step 3 (in Paul Leach's slides), A sends the message $\{\text{“A”}, T\}_{K_{ab}}$, and B replies with the message $\{\text{“B”}, T+1\}_{K_{ab}}$.

In class we said that only someone knowing K_{ab} should be able to generate the reply. Actually, it's more subtle than that. While what we said might be true for *some* encryption schemes, it is not necessarily true for *all* secure encryption schemes.

- Task: Assume that A's message and the reply are generated using the secure AES-CTR mode encryption scheme, and assume that the adversary sniffs the first message $C = \{\text{“A”}, T\}_{K_{ab}}$. Moreover, assume that “A” and T are encoded in a 128-bit string and that the adversary knows this encoding (hence the adversary also knows T). Now show that the adversary can create a ciphertext C' that decrypts to $(\text{“B”}, T+1)$ under key K_{ab} even if the adversary doesn't actually know K_{ab}.

All these problems are designed to show how subtle issues can creep into the design of secure protocols — in this particular case, one can't just modularly use any secure encryption scheme to encrypt $(\text{“A”}, T)$ and $(\text{“B”}, T+1)$.

Problem 6.5 (5 points.) Read this paper `http://www.schneier.com/paper-ssl.pdf`.

- Describe in your own words what a “protocol rollback attack” is. Explicitly refer to SSL in your explanation. Please limit your explanation to at most two reasonably sized paragraphs.

Problem 6.5 (5 points.) Consider a world in which we encrypt all of our network transmissions using some secure encryption scheme. A passive adversary can still sniff your network transmissions, but can only see: the source of the transmission, the destination of the transmission, the length of the transmission (packet size), and the time that the transmission was sent.

Assume that an attacker cannot break the encryption described above. Your job is to explain how an adversary might still learn private information from a user's transmissions.

- Task 1 (2 of 5 points): Describe in your own words how private information may leak out through the unencrypted source and destination addresses of the packets

- Task 2 (3 of 5 points): Now assume that the source and destination addresses are encrypted too. Explain how your encrypted transmissions might still reveal private information about your communications. Here the adversary can only see the time and size of the packets to and from your machine.

Additional reading. Gollmann, Chapters 11 and 13.