Midterm 2
CMSC 414: Computer and Network Security
April 23, 2015

Name

Instructions

Do not start until told to do so!
- This exam has 8 pages (including this one); make sure you have them all
- You have 75 minutes to complete the exam
- The exam is worth 100 points. Allocate your time wisely: some hard questions are worth only a few points, and some easy questions are worth a lot of points.
- If you have a question, please raise your hand and wait for the instructor.
- You may use the back of the exam sheets if you need extra space.
- In order to be eligible for partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

<table>
<thead>
<tr>
<th></th>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>To err is human</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PKIs</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Composing mechanisms</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Encryption modes</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Randomness</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Extra credit</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. To err is human (12 points total, 6 points each)

   (a) The weakest link of most cryptographic schemes is the human using them. **Concisely list three** ways that one could **misuse** cryptography in a way that would leave the system susceptible to attack (you can be as specific as you want, but you do not need to demonstrate any attacks or provide justifications).

   **Answer:** *Static keys, bad source of randomness, ECB, share secret keys, use a broken scheme (e.g., MD5), ... many possible answers here.*

   (b) When users forget their passwords, most responsible websites let users **reset** their passwords, but they do not tell what their password was. Why is this?

   **Answer:** *Because the website doesn’t know their password: they only know the (multi-)hash of their password, which they cannot invert.*
2. PKIs (16 points total)

(a) (8 points)

Assuming the traditional PKI, suppose an attacker were able to insert a certificate into a victim user’s set of trusted root certificates without the victim’s knowledge. Mark each of the below statements as (T)rue or (F)alse:

<table>
<thead>
<tr>
<th>Statement</th>
<th>True/False</th>
</tr>
</thead>
<tbody>
<tr>
<td>The attacker could revoke any valid certificate to that user.</td>
<td>T</td>
</tr>
<tr>
<td>After that user authenticates to a real website using a valid root certificate, the attacker cannot eavesdrop on that communication.</td>
<td>F</td>
</tr>
<tr>
<td>The attacker could impersonate any website to that user.</td>
<td>T</td>
</tr>
<tr>
<td>The attacker could impersonate any website to any user, on that machine or not.</td>
<td>F</td>
</tr>
</tbody>
</table>

Answer: F, T, T, F

(b) (4 points)

Suppose a root certificate’s secret key were compromised. Circle each of the following revocation strategies that would ultimately protect valid clients.

A. Do not revoke any certificates.
B. Revoke only the root certificate.
C. Revoke the root certificate and all certificates it signed, but no further.
D. Revoke the root certificate, all certificates it signed, all certificates they signed, and so on, all the way down to the leaf certificates.
E. Revoke only the leaf certificates

Answer: B, C, D

(c) (4 points)

Suppose a root certificate’s secret key were compromised. Circle each of the following reissue strategies that ultimately protect valid clients.

A. Do not reissue any certificates.
B. Reissue only the root certificate.
C. Reissue the root certificate and all certificates it signed, but no further.
D. Reissue the root certificate, all certificates it signed, all certificates they signed, and so on, all the way down to the leaf certificates.
E. Reissue only the leaf certificates

Answer: D
3. Composing cryptographic mechanisms (30 points total, 5 points each)

For this problem, assume that Alice wants to send a single message $M$ to Bob. To do so, Alice and Bob can potentially use a number of different approaches and cryptographic mechanisms. Assume the following terminology:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>Plaintext for a single message</td>
</tr>
<tr>
<td>$A</td>
<td></td>
</tr>
<tr>
<td>$\text{PK}_A$</td>
<td>Alice’s public key.</td>
</tr>
<tr>
<td>$\text{SK}_A$</td>
<td>Alice’s corresponding secret key.</td>
</tr>
<tr>
<td>$\text{PK}_B$</td>
<td>Bob’s public key.</td>
</tr>
<tr>
<td>$\text{SK}_B$</td>
<td>Bob’s corresponding secret key.</td>
</tr>
<tr>
<td>$E(\text{PK}, x)$</td>
<td>Public-key encryption of $x$ with public key $\text{PK}$.</td>
</tr>
<tr>
<td>$\text{Sgn}(\text{SK}, x)$</td>
<td>Public-key signing of $x$ with secret key $\text{SK}$.</td>
</tr>
<tr>
<td>$s$</td>
<td>Symmetric key</td>
</tr>
<tr>
<td>$\text{AES}(s, x)$</td>
<td>Symmetric-key encryption of $x$ in CBC mode using key $s$</td>
</tr>
<tr>
<td>$\text{MAC}(s, x)$</td>
<td>Symmetric-key MAC of $x$ using key $s$.</td>
</tr>
<tr>
<td>$H(x)$</td>
<td>SHA-256 hash function of $x$</td>
</tr>
</tbody>
</table>

You can assume that the public keys have been securely distributed, so Alice and Bob know their correct values. You can also assume that the message is small enough to be encrypted with public-key encryption.

For each of the following possible communication approaches, circle all of the properties that will securely hold with respect to the message $M$ that Alice is trying to send to Bob in the presence of Mallory, an attacker who can both eavesdrop and tamper with messages. If none of the properties hold, then circle None. If an approach fails entirely—that is, if it requires a step that either Alice or Bob cannot execute—then circle Broken (and do not worry about the other properties).

(a) Alice generates a new symmetric key $s$. She also splits her message $M$ into two halves: $M_L$ and $M_R$, such that $M = M_L||M_R$, and sends to Bob:

$$E(\text{PK}_B, s), \quad \text{Sgn}(\text{SK}_A, H(E(\text{PK}_B, s))), \quad \text{AES}(s, M_L), \quad \text{AES}(s, M_L \oplus M_R)$$

**Confidentiality**  **Integrity**  **Authenticity**  **Non-repudiation**  **None**  **Broken**

**Answer:** Confidentiality (Bob can recover the message). Because she is only signing the key, and not the message itself, no other properties.

(b) Alice and Bob privately exchange a symmetric key $s$ in advance. Alice later generates a random number $r$ and sends to Bob:

$$\text{MAC}(s, r \oplus M), \quad r, \quad M$$

**Confidentiality**  **Integrity**  **Authenticity**  **Non-repudiation**  **None**  **Broken**

**Answer:** None: an attacker can flip a bit in $M$ so long as he flips the corresponding bit in $r$.

*Continued on the next page*
(c) Alice and Bob privately exchange a symmetric key $s$ in advance. Alice later uses this key to send to Bob:

\[ \text{AES}(s, M), \quad \text{MAC}(s, H(M)) \]

<table>
<thead>
<tr>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Authenticity</th>
<th>Non-repudiation</th>
<th>None</th>
<th>Broken</th>
</tr>
</thead>
</table>

**Answer:** Confidentiality, Integrity, Authenticity, None

(d) Alice generates a new symmetric key $s$ and sends to Bob:

\[ E(PK_A, s), \quad E(PK_B, s), \quad Sgn(SK_A, s), \quad \text{AES}(s, M) \]

<table>
<thead>
<tr>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Authenticity</th>
<th>Non-repudiation</th>
<th>None</th>
<th>Broken</th>
</tr>
</thead>
</table>

**Answer:** Confidentiality, Integrity, Authenticity, None

(e) Alice generates a new symmetric key $s$ and sends to Bob:

\[ E(PK_B, s), \quad \text{AES}(s, M), \quad Sgn(SK_A, H(M)) \]

<table>
<thead>
<tr>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Authenticity</th>
<th>Non-repudiation</th>
<th>None</th>
<th>Broken</th>
</tr>
</thead>
</table>

**Answer:** Confidentiality, Integrity, Authenticity, None

(f) Alice generates a symmetric key $s$ and a new public-private key pair, $(PK_t, SK_t)$, and sends to Bob:

\[ E(PK_t, SK_t), \quad E(PK_t, s), \quad \text{AES}(s, M), \quad Sgn(SK_t, H(m)) \]

<table>
<thead>
<tr>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Authenticity</th>
<th>Non-repudiation</th>
<th>None</th>
<th>Broken</th>
</tr>
</thead>
</table>

**Answer:** Broken: Bob would need the secret key in order to recover the secret key!
4. Encryption modes (30 points total, 5 points each sub-part)

Consider the following algorithm for creating a hash function that maps a pre-image $M$ (that is a multiple of 128-bits long) to an image of 128 bits using AES under different modes of encryption:

- Compute $c = \text{AES}(K, M)$ using a publicly known key $K$, an IV of 0, and mode of encryption X (below).
- Return the last 128 bits of $c$ (i.e., the last block of the ciphertext).

For each of the following modes of encryption, circle YES or NO as to whether the above algorithm would result in a hash function that is **Pre-image resistant** (given $H(M)$, it is difficult to compute $M$) and/or **Collision-resistant** (given $M$ and $H(M)$, it is difficult to find $M' \neq M$ such that $H(M') = H(M)$), and provide a single-sentence justification for your answer.

(a) ECB mode

**Pre-image resistant:** YES / NO. Justification:

**Answer:** No, because you can decrypt AES, and it is only a function of the last block.

**Collision-resistant:** YES / NO. Justification:

**Answer:** No, any two messages with the same last 128 bits will collide.

(b) CTR mode

**Pre-image resistant:** YES / NO. Justification:

**Answer:** Yes, because you don’t know what the IV is by the time you get to the last block.

**Collision-resistant:** YES / NO. Justification:

**Answer:** No, for the same reason as ECB.

(c) CBC mode

**Pre-image resistant:** YES / NO. Justification:

**Answer:** Yes, because the final ciphertext block depends on all previous blocks, which are truncated.

**Collision-resistant:** YES / NO. Justification:

**Answer:** Yes, because the final ciphertext block depends on all previous blocks, and thus AES’s confusion/diffusion properties make each bit unpredictable. **Update:** Actually, some students have pointed out that you can launch a length-extension attack to violate the collision resistance. If you described this in your exam, come back for a regrade!
5. Randomness (12 points total)

(a) (8 points)
Recall that a good source of randomness (i.e., one that is very difficult to predict) is fundamental to the correct operation of many cryptographic schemes. Suppose you have access to a function $R()$ that generates 128-bit, seemingly random values. Unfortunately, an attacker has gained some insight into $R()$ that makes it not quite random:

- Every time $R()$ is called, either the first half (64-bits) or the second half is truly random, but the attacker can predict the other half.
- Every subsequent time that $R()$ is called, the predictable half swaps (if the first half was predictable, then this time the first half is random, and vice versa).
- However, you cannot tell just from inspecting the output of $R()$ which half is the random half (only the attacker has this particular insight).

Your task is to use $R()$ to construct a new function, $S()$, that generates 128-bit numbers that are fully unpredictable, even to this attacker.

Also, give a brief (one-sentence) justification why any given bit of your $S()$ function will be unpredictable to the attacker.

**Answer:** $S() = R() \oplus R()$ is the simplest solution. Each bit is the xor of a predictable bit with an unpredictable bit: the outcome of this is unpredictable. One could also do AES($R()$, $R()$); the rationale there is that there is enough randomness in the key for confusion/diffusion to make each output bit unpredictable.

(b) (4 points – Note: this part of the problem is independent of the above part.)

One-time pads cannot be used repeatedly, or else they leak information. However, for a one-time pad consisting of sufficiently many 8-bit long values, some of these 8-bit values will repeat. Explain why these two statements are not contradictory.

**Answer:** Each 8-bit long part of the pad is chosen uniformly at random. Even if the plaintext message repeated, an attacker cannot infer this because any 8-bit string would be equally possible.
6. Extra credit (10 points total, 5 points each)

(a) Suppose that Alice and Bob have already established a shared symmetric key $K$. They now wish to develop a communication protocol so that, when they later meet up, one can prove to the other that they know the key. They propose the following protocol to use when one of them (the “challenger”) wants to check if the other (the “responder”) knows the key:

- Challenger: Choose a random nonce $r$ that is the same length as the key, and transmit $m = (K \oplus r)$.
- Responder: Compute $K \oplus m = K \oplus (K \oplus r) = r$, and transmit $r$.
- Challenger: Verify that this equals the random $r$ they originally chose.

Only someone who knows the secret key could have computed $r$, and so only someone who knows the key can follow the above protocol. Yet there is a flaw in this scheme. Show step by step what an attacker could do to exploit this flaw, and describe what they learn.

**Answer:** Contact either one of the parties: send a random value $r$. They will reply with $m = (r \oplus K)$. Compute $r \oplus m$ to extract the secret key $K$.

(b) A spy has infiltrated another country, where she will be deep undercover for a week. She wishes to inform her home, on a daily basis, that she is alright, but the country she infiltrated would like to forge messages from her to make it appear that she is alright even if she is not.

Before going into the country, she chose a random 128-bit value $x$ that only she knows, and published $H^7(x)$ that everyone knows. (Here, $H$ is a good, cryptographic hash function, such as SHA-512.) Once in the country, however, she can only send messages out (she cannot receive).

Describe, for each day $i = 1, \ldots, 7$:

- What she can send home so that they can verify that she is alright, and
- How those at home can verify that it was from her and not forged by an attacker.

**Answer:** Day $i$: $H^{7-i}(x)$. Verify day $i$’s message $m$ by checking that the previous day’s message was $H(m)$. Cannot forge because $H$ is one-way and $x$ is too large to efficiently do a brute force attack.