650.445/600.454: Practical Cryptographic Systems April 5, 2011

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: Where possible, the recommended approach to validating RSA signature padding is to re-construct the expected padded message, recover the padded message from the signature (by exponentiating the signature to the public exponent e mod N), then compare the two buffers.

2. True or False: Skype uses the RC4 algorithm to protect the confidentiality of voice calls.

3. True or False: Padding Oracle attacks could be prevented by MACing the message *and* any encryption padding, then encrypting the message, padding and MAC tag.

4. True or False: The most common approach to preventing timing attacks on RSA decryption is to use *quantization*, i.e., make every RSA decryption take the same amount of time.

5. True or False: Replay attacks can frequently be prevented by adding a counter or timestamp to messages, and protecting the entire message with a secure MAC.

6. True or False: The standard TLS handshake protocol is forward-secure, meaning that if the server's long-term secret key is stolen, this does not compromise the security of any previous communications that may have been captured via eavesdropping.

7. True or False: It is safe to use CBC mode encryption with a non-random Initialization Vector, *e.g.,* an ascending counter value.

8. True or False: In Capsicum, sandbox security policies are decoupled form source code into their own policy files using capabilities.

9. True or False: Compromising the tamper-resistance of an individual Clipper/Capstone chip would have been problematic for the owner of that specific chip, but would not have compromised the security of the system as a whole.

10. True or False: A good way to pick random 4-digit PIN codes is to pick a random 14-bit value B using a strong random bit generator, and output $B \ mod \ 10,000$.

Problem 2: Short Answers (25 points)

1. **AACS.** Describe two major cryptographic protections offered by the AACS system (used in Blu-Ray and HD-DVD disks) and how they differ from the CSS system used in DVDs. Explain briefly what the protections are, and what threats they protect against.

2. **SSL and ciphersuites.** Describe an attack on the SSL (v1 and v2) protocols that caused the participants to use export-weakened ciphers. (1) Explain what export-weakened ciphers are, and (2) give a brief overview of how and why an attacker was able to force participants to use these ciphersuites even when both parties supported stronger ciphers.

3. **Side Channel Attacks.** Modern smart cards include protections against Simple Power Analysis (SPA) and Differential Power Analysis (DPA). Explain very briefly (1) what power analysis attacks are, (2) why this is a particularly important issue for smart cards, as opposed to other cryptographic devices, and (3) what techniques modern smart cards use to protect against these attacks.

Problem 3: Full Disk Encryption. (35 points)

Many full disk encryption schemes encrypt data at the *sector* level. A sector is a fixed-size block of data stored on a hard drive. When a drive is first formatted, it is divided into sectors. Whenever a file is read or written to the disk, the computer's operating system breaks the file up into sectors and reads/writes each sector to the disk.

A typical full-disk encryption system sits between the operating system and the disk drive. It intercepts each sector-level write, and encrypts the sector using a secret key prior to sending it to the disk. Similarly, it decrypts each sector that the operating system requests from the disk.

For this problem you can assume that the disk has N sectors, and each one is 4,096 bytes long. Each sector has a unique *sector number* in the range 1 to N. The sector number is known by the hard drive, and will be specified by the OS in all read/write requests.

1. Part 1 (5 points). What are some technical advantages of encrypting at the sector level, rather than encrypting at the level of individual files?

2. Part 2 (10 points). Sectors can only store a fixed number of bytes (*e.g.,* 4,096 bytes) and cannot be expanded. Unfortunately we must assume that operating system is writing sectors of the same size — hence there is no room on the disk for any "extra" data that our encryption scheme generates. What challenges does this present if we would like to use AES in CBC mode to encrypt each sector? Remember that sectors must be read and written independently of each other.

3. Part 3 (15 points). Recall that each sector of the disk has a unique sector number associated with it, and the disk encryption system will always know the sector number when it reads/writes a sector. (The sector number is stored in a separate location on the drive, and does not take up any of the 4,096 data bytes in the sector itself.)

   Can you describe a mode of operation for AES that would encrypt a sector without adding any additional data or increasing its storage requirements on the disk. Please be very specific about how this mode will work!

4. Part 4 (5 points). Considering your solution in Part 3, imagine that the Operating System determines that the data in a sector is no longer needed, and overwrites that sector with new data. What possible security concerns might this raise?

Problem 4: Key wrapping (30 points)

Sometimes it is necessary to encrypt a cryptographic key (K) using a different key — typically referred to as a "key encrypting key" (KEK). One way to do this is to use a mode of operation known as *key wrapping*. For example, imagine that K is 64 bits long, and consider the following proposed AES-based key wrapping algorithm:

$$C = \mathsf{AES\_ECB\_ENC}(KEK, 0_{64}||K)$$

Where $0_{64}||K$ denotes 64 "0" bits concatenated with K.[1] To *unwrap* C, first compute:

$$P = \mathsf{AES\_ECB\_DEC}(KEK, C)$$

And then check that the first 64 bits of P are equal to 0_{64}. If this is not the case, output error. Otherwise output the remaining bits of P as the decrypted key.

Part 1 (10 points). What purpose is served by pre-pending 0_{64} to K before encryption, and by checking this value during decryption? Explain why this mechanism achieves this purpose.

[1] Note that key wrapping does not require an IV to add randomness, since the key itself provides a high degree of entropy.

Part 2 (10 points). Would this algorithm still achieve the stated goal if K was longer than 64 bits — say, 128 or 256 bits long? Why or why not? (Recall that AES has a 128-bit block size.)

Part 3 (10 points). What if we had a secure block cipher with a very, very large block size — for example, 4,096 bytes in length. How could we apply this to our disk encryption scheme from the previous problem to get *authenticity* as well as security? What disadvantages might this have.

Note: since we may not have room to add a 0_{64} *check area, are there other ways we could detect tampering by examining decrypted sector data?*