601.445 / 601.645: Practical Cryptographic Systems March 25, 2020

Midterm

Instructor: Matthew Green *Due: 6:00pm EDT, March 27*

Name: ____

This take-home midterm should be completed individually. Do not collaborate with any other student. You are permitted to use the Internet and any printed references, provided that your answers are your own. Please submit the completed assignment via Gradescope.

Problem 1: True, False or Unknown (20 points). Mark each question with T/F/U. You do not need to explain your work.

1. **T/F/U.** Textbook RSA encryption ($m^e \ mod \ N$ without padding) satisfies the IND-CPA definition for encryption.
2. **T/F/U.** The TLS 1.3 protocol removes support for RSA encryption.
3. **T/F/U.** Variants of the square-and-multiply algorithm, described in class, can be implemented in such a way that each iteration through the loop processes *more than one* bit from the exponent.
4. **T/F/U.** The average air speed velocity of a laden swallow is about 11 meters per second.
5. **T/F/U.** In implementations of RSA that use the CRT optimization, once the secret exponent d has been computed, there is no reason to retain knowledge of the prime factors p, q.
6. **T/F/U.** It is safe to use a unique but *predictable* Initialization Vector in CBC-mode encryption (*e.g.,* an ascending counter value.)
7. **T/F/U.** The ECDSA and Schnorr signature schemes become catastrophically insecure if the same random nonce (r) is used to sign two different messages.
8. **T/F/U.** The safest way to verify a PKCS #1v1.5 padded *signature* is to first compute $v = s^e \ mod \ N$ and parse each byte of the padded message v to ensure that it has the correct structure (header bytes, some padding bytes, a 0 byte, followed by a hash.)
9. **T/F/U.** The cause of the flaws identified in the "Mining your Ps and Qs" paper was a bad implementation of square-and-multiply.
10. **T/F/U.** The "Imperfect Forward Secrecy" paper notes that TLS includes the ciphersuite in the signed portion of the `ServerKeyExchange` message.

Problem 2: Short answers (30 points)

Provide a written answer to each of the following questions. Justify your explanation.

1. **Lucky13.** Explain exactly what causes the measurable timing signal used in the Lucky13 attack.
2. **RSA blinding.** Explain why RSA blinding prevents timing attacks on RSA decryption, such as the Kocher attack and the Boneh-Brumley remote timing attack. (Please describe how RSA blinding works.)
3. **Hash functions and Flame.** In 2012 a piece of malware named "Flame" was discovered in the wild. The developers of Flame used a flaw in the MD5 hash function to enable digital signing of the malware. Using online resources (there are many), read about Flame and explain the cryptographic attack at a high level. Why was this attack not blocked, given that MD5's flaws have been known for many years?

Problem 3: Making random lists (50 points)

You have been given the task of designing a system that can randomly and *fairly* select patients for a clinical trial. For simplicity we will assume that in this trial, there are n total patients of which $m < n$ will receive a drug and the remainder will receive placebo.

To facilitate this, you will design an online (*e.g.,* web-based) service that can be accessed by clinicians. To use the system, clinicians first log into the system using an account. They next enter a list of patient identifiers into a web form, one identifier per row. When the list entry is complete, the clinician presses a button to submit the list to the system. The service then *randomizes* the ordering of the patient identifiers. At the conclusion of this process, the clinician must select the top m patients from the randomly-ordered list to receive the drug. For an example of what such an interface might look like, see `https://random.org/lists`.

Your goal in devising this system is to ensure that the results of the test are fair, unbiased, and *verifiable* (if possible). This means that the output of the system should be random, and clinicians should not be able to "game" the system in order to ensure that specific patients receive the drug. Answer the following questions:

1. In designing this system you have access to a secure cryptographic random bit generator. This generator can output any number of *bits* (note: it does not output random integers.) We will assume that each bit this generator produces is random and unbiased. Devise an algorithm that uses this bit generator to produce a random sorting (permutation) of a list of n items, for arbitrary list sizes n. You can provide your algorithms as pseudocode.
2. An obvious concern in this system is that clinicians can *bias* the output of the system, simply by pressing the "randomize" button multiple times until certain patients appear at the top of the list. Furthermore, the person who developed the website could collude with physicians to sort patients in specific ways. Since there is no way

to verify that a specific list was sorted honestly, there is no protection against these attacks.

One solution to this problem is to use a *deterministic* and verifiable system in which each set of patient identifiers has only one possible (verifiable) sorting. Using *e.g.,* hash functions or any other cryptographic primitive, how would you change the system above to enable this?

3. Are there any other protections you might implement in designing this system, to ensure that the results are fair and verifiable? These do not have to be cryptographic. They can involve software, auditing and/or usage policy changes. Remember that you should not prevent clinicians from doing their job effectively.