Assignment 1

Due Friday, October 7th, 2016 at 23:59

Assignments are to be completed individually. We generally expect you to make an honest effort searching around online before contacting course staff with technical questions (`stackexchange.com` and `crypto.stackexchange.com` are great resources btw).

Submission Instructions

Please use common document file formats such as `.txt`, `.pdf`, `.doc`, etc. Place your answers into separate files and/or subfolders for each of the 4 questions. Submit a `.zip` of the assignment files via OWL. The submission form in OWL will be available only up to 48 hours after the assignment due date. Email submissions are not accepted. Be sure to consult the course outline regarding the late policy.

1. [20 marks] Encryption Basics

For each of the following, state whether the given ciphertext is a valid encryption under the given cipher. Your answer should be of the form: *Yes*, *No*, or *Insufficient information*, followed by a one or two sentence justification.

(a) [5 marks] Is `HWCFGKOVZPFKE` a valid Enigma encryption of the plaintext `WEATHERREPORT`?

Solution. No. The 'P' of 'REPORT' is encrypted to ciphertext letter 'P,' which is not possible under Enigma's design.

(b) [5 marks] Is `HJXDDUDAYIZQRFGSJJ` a valid one-time pad encryption of `WESTERNENGINEERING`?

Solution. There are two acceptable answers. Either (A) Yes, it's is valid in that there exists SOME key which would decrypt that ciphertext to that plaintext. Or (B) Insufficient information. Due to the information theoretic security of OTP, we can recover no information about a plaintext given its ciphertext only.

(c) [5 marks] Let `a={0x1E, 0x3F, 0x81, 0xDF, 0x64, 0x24, 0x01, 0x61}` be a vector of bytes. Is `a` a valid DES ciphertext?[1]

Solution. YES. A block cipher is a one-to-one function defined by a key, and in the case of DES, each ciphertext string $c \in \{0,1\}^{64}$ is associated with a unique plaintext string $m \in \{0,1\}^{64}$ for a given key. Thus all 64-bit ciphertexts correspond to a unique 64-bit plaintext.

[1] Note: The prefix `0x...` denotes a hexidecimal value.

(d) [5 marks] Let `b={0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}` be a vector of bytes. Is `b` a valid AES ciphertext?

Solution. Yes. The AES decryption function accepts 16 byte blocks, and thus is valid for the same reasons as above.

2. [24 marks] Security Games

For each of the following, use the appropriate security game to prove the given cipher is insecure under the stated security notion.

Solution. Strategies may vary. The main goal here is to (1) provide a strategy that gives the adversary an advantage and (2) to explain why its non-negligible. In each of the following sample solutions the advantage is *constant* in the security parameter. Under our definition, a function is negligible if it grows more slowly than the inverse of any polynomial in the security parameter. A constant advantage does not meet this criterion, and is, therefore, non-negligible.

(a) [8 marks] Prove that Enigma is not IND-EAV secure.

Solution. Strategy: We exploit the flaw that a plaintext letter doesn't encrypt to itself. Here's one way: The adversary A constructs two messages: a string of k `a`'s, and a string of k `b`'s. If A sees an `a` in the challenge ciphertext, it guesses B chose the all-`b` string, and vice versa. **Non-negligibility**: The probability the challenge ciphertext will contain one or more `a`'s given the message was k `a`'s is 0 (since in Enigma `a`'s cant encrypt to `a`'s). On the other hand, the probability the challenge will contain one or more `a`'s given the message was k `b`'s is:

$$1 - P(\text{challenge contains no a's})$$

$$= 1 - P(\text{each of the } k \text{ ciphertext characters are some letter other than a})$$

$$= 1 - \left(\frac{25}{26}\right)^k$$

So with an 18-character message of all b's there is slightly greater than 50% chance the corresponding challenge will contain an a. The probability, of course, is the same for the opposite case, *i.e.,* the chance of getting one or more `b`'s in the ciphertext given the all `a` message was chosen. Note that this probability is constant in the security parameter, *i.e.,* adding more rotors or plugs doesn't change the adversary's advantage.

(b) [8 marks] Prove that AES in CBC mode is not IND-CPA secure if the IV can be predicted by the adversary.

Solution. Strategy: First note that the fact we're using AES is largely irrelevant to this attack; it would apply to any (otherwise secure) block cipher used in CBC mode.

A submits two distinct single-block messages m_0 and m_1 during the query phase and receives their respective encryptions $IV_0||c_0$ and $IV_1||c_1$. Observe $c_0 = Enc(IV_0 \oplus m_0)$, and $c_1 = Enc(IV_1 \oplus m_1)$. Suppose A is able to predict ahead of time what initialization vector B will use to create the challenge ciphertext. We'll call this IV_b. A now prepares two "new" messages: $m'_0 = m_0 \oplus IV_0 \oplus IV_b$, and $m'_1 = m_1 \oplus IV_1 \oplus IV_b$ and submits them to B as the challenge. B picks one of them, m_b and encrypts it as : $c = Enc(IV_b \oplus m_b)$. Now suppose $m_b = m'_0$. Then

$$c = Enc(IV_b \oplus m'_0)$$

$$= Enc(IV_b \oplus m_0 \oplus IV_0 \oplus IV_b)$$

$$= Enc(m_0 \oplus IV_0)$$

$$= c_0$$

Thus if $c = c_0$, then $m_b = m'_0$, and A would output guess $b = 0$, otherwise A guesses $b = 1$. **Non-negligibility**: A will always win, *i.e.,* win with probability 1, which is independent of the security parameter, and thus non negligible.

(c) [8 marks] Prove that AES in CBC mode in not IND-CCA2 secure, even if the adversary cannot predict the IV.

Solution. Strategy: Once again, this attack proceeds independently of the specific internals of AES, and would apply to any other (otherwise secure) block cipher. Here's one possible strategy: A submits any two distinct messages m_0 and m_1, each of one block in length, and receives the challenge ciphertext $IV_b||c$ where $c = Enc(IV_b \oplus m_b)$. As per the CCA2 game, A can continue to make decryption queries. It cannot submit c itself, but it can build a trivially different ciphertext c' based on the c. For example, the adversary can submit c appended with block of all 0's, *i.e.,* $c' = c||z$ where z is a zero-block (*i.e.,* 16 bytes of all 0's). A submits decryption query $IV_b||c'$. Because $c' \neq c$, B will respond with $Dec(c')$. As per the structure of CBC, the first block of c' will decrypt to the same thing as c:

$$Dec(c') = Dec(c||z) = (IV_b \oplus m_b)||(c \oplus Dec(z))$$

A can just discard the second block $(c \oplus Dec(z))$. A then computes $IV_b \oplus (IV_b \oplus m_b)$ to recover m_b. **Non-negligibility**: A now knows m_b with certainty and therefore will always win. Note A's strategy didn't require the ability to predict the IV, since the strategy succeeds regardless of the IV's value; A is just resubmitting the challenge's IV (*i.e.,* IV_b).

Note: You don't have to be formal. The goal here is to convince us that you understand the mechanics of these security games, can come up with a strategy for the adversary to follow, and then justify why that strategy gives them a non-negligible advantage.

3. [20 marks] Encryption in Practice

For this question you will explore encryption in practice. This is a somewhat open-ended problem and you will be graded according to your ability to make security-conscious choices.

The plaintext shall consist of your full name and student number. Encrypt it using 128-bit AES in CBC mode. Use any method (*i.e.,* programming language or software program) you wish. Encode characters using UTF-8 (e.g., the character '0' is encoded as the byte `0x30`, 'A' as `0x41`, etc).

Submit the following:

(a) A text file containing the source code or commands you used. In the comments, specify your name, student number, what programming language, library, or software program you used,

(b) A text file containing the exact plaintext message (i.e., name and student number),

(c) The encryption key used, written as a list of hexidecimal bytes (e.g., `00, 01, 02, ...EE, FF`),

(d) Any initialization vector (IV) used, written as a list of hexidecimal bytes,

(e) The resulting ciphertext, written as a list of hexidecimal bytes.

Solution. There are many ways to solve this question, but common reasons for mark deductions include (a) the key and IV weren't independently and randomly generated (e.g., handpicked, hardcoded, etc), (b) you accidentally used ECB mode, etc.

4. [36 marks] Fun with Hashing

In this question we will explore the security properties of hash functions in the context of file hashing.

Step 1. Download the following assignment files. This zip file contains to (Linux) programs: `assignment1-good` and `assignment1-evil`.

Note: You do not need to *execute* these programs (and you may not even be able to depending on your OS). We only use them in the context of computing their respective hash values.

(a) When executed, `assignment1-good` prints:

SE 4472/ECE 9064 is a GOOD course!

(b) When executed, `assignment1-evil` prints:

SE 4472/ECE 9064 is an EVIL course. MOO HA HA!

Step 2. Download and install OpenSSL If you're running Windows, you'll need to install the binaries. If you're running Linux, you likely have it installed already. In a terminal type `openssl version` to check. Mac users can install OpenSSL via brew. Finally, hang on to your installation, because we'll be using it again in future assignments.

Step 3. Now use OpenSSL to:

- Compute the MD5 hashes of assignment1-good and assignment1-evil
- Compute the SHA-1 hashes of assignment1-good and assignment1-evil

Step 4. Submit the following:

1. [8 marks] The hashes computed in **Step 3**.
2. [28 marks] An analysis of the results from the previous step. For each of the following, answer the question in one or two sentences:
   (a) [4 marks] What do the MD5 hashes suggest about these two files? **Solution.** The files are the same.
   (b) [4 marks] What do the SHA1 hashes suggest about these two files? **Solution.** The files are different.
   (c) [4 marks] Thinking about the answers to the previous two questions, which hash function is right? Which one is wrong? How can you tell? **Solution.** SHA-1 is right. Suppose the files actually were the same. Then SHA-1, being a deterministic function, would report the same hash value for both files. But it reported different hashes. Therefore the files must be different.
   (d) [4 marks] What security property is being violated here? **Solution.** Collision resistance. What we have manufactured here with these two files is an actual MD5 collision.
   (e) [4 marks] Suggest a possible (evil) application of the ability to violate this security property? **Solution.** One evil application of collisions is being able to construct two different identities in the context of public-key certificates. But here's a scenario an evil software developer might try: Find a collision between a good and malicious program (as we have in this question). Post source code and hash of a good program, but distribute evil program instead. If anyone checks the hash, it matches the good program. Meanwhile the evil program is taking over your machine.

(f) [4 marks] On average, how many operations (*i.e.,* invocations of a hash function) would an attacker have to have to make (on average) to pull of this attack on a well-designed k-bit hash function? **Solution.** Collisions in a well-designed hash function of b-bits can be found in $2^{\frac{b}{2}}$ calls to the hash function on average as per the birthday paradox. MD5 has a 128-bit digest size, so a collision can be expected in 2^{64} calls/operations.

(g) [4 marks] Do you think Prof. Essex actually performed that much computation in order to create these files for this assignment? Feel free to be creative/speculative, but justify your answer. **Solution.** 2^{64} operations is certainly within the realm of feasibility for a well-resourced, real-world adversary. According to blockchain.info, the Bitcoin network is doing 2^{60} SHA-1 hashes per second! But that's millions (possibility billions) of dollars of hardware. No, in this case MD5 is broken, *i.e.,* has known methods for creating collisions more efficiently than by random guessing.
Although SHA-1 does not have any known collisions, NIST no longer considers 2^{80} to be a big enough safety margin. With a 160-bit digest SHA-1, therefore, can no longer be considered collision resistant and has been deprecated for use in applications where collision-resistance is critical. Pre-image resistance, however, is still assumed.