Quiz 1

1. This quiz is intended to provide a fair measure of your understanding of the course material to date (Homeworks 1–3 and Lectures 1–10).
2. Do not open this quiz booklet until the quiz begins. Read all the instructions first.
3. **Do not discuss any aspect of this quiz with anyone (except 6.857 staff) until noon on Friday, October 10, 2003.**
4. When the quiz begins, write your name on every page of this quiz booklet.
5. This quiz booklet contains 12 pages, including this one. An extra sheet of scratch paper is attached.
6. This quiz is open-book, open-notes. No calculators or programmable devices (including laptop computers) are permitted.
7. Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem; pages may be separated for grading.
8. Partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
9. Good luck!

Problem	Points	Grade	Initials
1	24		
2	9		
3	36		
4	12		
5	10		
6	10		
7	10		
Total	111		

Your Name: ______________________________

Problem Q1-1. Short Answer [24 points]

(a) At a recent Red Sox game, you observed the catcher making a variety of signals to the pitcher with his hands before every pitch. Following the signals, you observe the type of pitch: e.g., fast ball, curve ball, slider, knuckleball, bean-ball, etc. By the end of the game, you are able to predict every pitch after seeing the catcher's signals. What sort of cryptographic attack have you successfully executed?

Solution: This is a known-plaintext attack. Half credit was given for the answer "passive attack."

(b) What is $11^{-1} \pmod{29}$? Show your work.

Solution: $11^{-1} \pmod{29} = 8$, because $8 \cdot 11 = 88 = 3 \cdot 29 + 1$. We can get this result by using Euclid's extended algorithm.
Half credit was given for the answer $11^{-1} = 11^{28} \pmod{29}$, based on Fermat's Little Theorem.

(c) You are watching an encrypted conversation between Alice and Bob. You notice that the prefixes of many of the ciphertexts agree for several hundred bytes. In addition, these identical prefixes are always a multiple of 16 bytes long. However, you never observe two identical chunks of ciphertext of any significant length following the identical prefixes. Conjecture what cipher is being used, what mode of operation is being used, and what Alice and Bob are doing wrong.

Solution: The answer we had in mind was AES (or DES) under CBC mode, (incorrectly) using the same IV for every message. We also gave full credit for an answer such as AES or DES in ECB mode, with some explanation (e.g., all messages have long, common headers.)

(d) Next, you start spying on a different encrypted conversation between Alyssa and Ben. You see many ciphertexts of exactly the same length (which is not a multiple of 16 bytes), in which only a few bits differ in every message. Once again, conjecture what cipher is being used, what mode of operation is being used, and what Alyssa and Ben are doing wrong.

Solution: The answer we had in mind was OFB (with any block cipher), (incorrectly) using a duplicated IV. Other correct answers include any stream cipher in which Alyssa and Ben reset the state of the stream with every message.

(e) Describe how to efficiently test if g is a generator modulo a prime p, where $p-1 = 4q^2$ for some prime $q > 2$.

Solution: Check that $g^x \neq 1 \pmod{p}$, where x is every divisor of $4q^2$ (other than $4q^2$ itself): $x = 1, 2, 4, q, q^2, 2q, 4q, 2q^2$. If all tests pass, g is a generator. In fact, it is sufficient to do the test only for $x = 4q, 2q^2$, because if any of the above tests fail, they will also fail for one of those two values of x.

(f) The year is 2050 and Moore's law has continued to hold, doubling computer performance every 18 months. Is it time to retire AES-128? How about AES-192 or AES-256? Explain your reasoning.

Solution: Today, 80 bit symmetric keys offer only decent security. In our scenario of the future, computers will be about 2^{31} times faster, implying that 111 bit keys are questionable. At this point, the security of AES-128 is in question (only one student thought about the lifetime of the data being protected; it may be time to retire the cipher if the data needs to stay secure long after 2050, which is often the case). AES-192 and AES-256 are still fine.
Of course, not many people believe that Moore's Law will hold for so long ...

Problem Q1-2. Security Policy [9 points]

For each of the 9 items listed on the front page of this quiz, choose whether the item contains elements of a security *policy*, security *mechanism*, *both*, or *neither*. Check one box per row.

Item	Policy	Mechanism	Both	Neither
1	x			
2		x		
3		x		
4		x		x
5				x
6		x		
7				x
8				x
9				x

Solution Note: We accepted either “mechanism” or “neither” for item 4.

Problem Q1-3. True or False [36 points]

Circle **T** or **F** for each of the following statements. No justification is required, but if you think the question is ambiguous, state your clarifying assumptions.

True **False** In the Fermat primality test, there is a small (but non-zero) probability that the candidate number will be declared composite when it is actually prime.

Solution: False. The one-sidedness of the error goes the other way.

True **False** A deterministic public-key encryption algorithm can never be secure against an adaptive chosen-ciphertext attack.

Solution: True. See the lecture defining the attack.

True **False** A PGP public key is more trustworthy if it has been signed by a lot of other public keys.

Solution: False. First, you shouldn't trust signatures from keys you haven't authenticated. Second, even if you believe the key is *valid* (authenticated), you shouldn't necessary *trust* it to properly sign other keys.

True **False** A good way to generate an AES encryption key is to apply SHA-1 to the number of seconds that have elapsed since January 1st, 1970.

Solution: False. The domain is too small (only about 2^{30} seconds have elapsed since the epoch). Additionally, if the adversary knows the approximate time at which the key was created (which is almost certainly true), the search space is reduced significantly.

True **False** It is possible to compute the SHA-1 value of every number in the range $1, \ldots, 10^{10}$, on today's standard PC, in a single day.

Solution: True. One can invoke SHA-1 over $200,000 = 2 \cdot 10^5$ times per second, so it would take $50,000$ seconds, or about 14 hours, to complete the given task.

True **False** It is not known if computing $\phi(N)$ is harder than computing the prime factors of N, where N is an RSA modulus.

Solution: False. The problems are known to be equivalent: if you can find primes p, q such that $N = pq$, then $\phi(N) = (p-1)(q-1)$. Conversely, if $\phi(N)$ is known, then $p+q = N+1-\phi(N)$, and $pq = N$. Finding p and q is now just a matter of solving this quadratic equation.

True **False** In RSA, for a given modulus N, any prime larger than 2 may be used as the public exponent e.

Solution: False: e must be relatively prime to $\phi(N)$, so any e that divides $\phi(N)$ cannot be used.

True **False** It is possible to have a digital signature scheme where forgery is impossible for a computationally unbounded adversary.

Solution: False. Signatures are publicly verifiable, therefore an unbounded adversary can simply try all signatures until she finds one that verifies correctly.

True **False** If $p = 2q+1$ where p and q are primes, then -1 is not a square mod p.

Solution: True (sort of). Whenever "safe primes" are used, q is meant to be an *odd* prime, but we did not make this explicit. A number a is a square mod p if and only if $a^q = 1 \pmod{p}$. Because q is odd, $(-1)^q = -1 \pmod{p}$, so -1 is not a square.

We gave full credit to anyone who pointed out the (only!) counterexample of $q = 2$.

True **False** Given an AES key K, it is possible to efficiently construct a one-kilobyte plaintext message M such that an encryption of M under AES (using key K) in CBC mode consists of all 0s.

Solution: True. Simply build the ciphertext of 1024 bytes (plus 128 bits for the IV), and decrypt it under CBC mode with key K. The result is a message which can encrypt to all 0s.

True **False** Using a one-time pad for encryption, followed by a CBC-MAC on the resulting ciphertext, provides unconditional secrecy and computationally secure authentication.

Solution: True. No secrecy is compromised, because the message is encrypted *before* the MAC is performed.

True **False** When k is large, testing a randomly-generated k-bit number t for primality can be done for use in a cryptosystem by testing whether $2^t \equiv 2 \pmod{t}$.

Solution: True. The chance that a randomly-generated number (of the size used in cryptosystems) is composite, but also a base-2 pseudoprime, is at most $10^{-41} \approx 2^{-140}$. The risk of this event is negligible, and in any case, the first ciphertext probably won't decrypt properly, alerting the key owner to the problem.

True **False** If $h(\cdot)$ is collision-resistant, then $h(h(\cdot))$ is collision-resistant.

Solution: True. If it is feasible to find a collision x, x' under $h(h(\cdot))$, then either x, x' is also a collision under h, of $h(x) \neq h(x')$ is a collision under h. In any case, h is not collision-resistant, which is a contradiction.

True **False** SHA-1 is more resistant to attacks based on the birthday paradox than MD5.

Solution: True. The output of SHA-1 is 160 bits, whereas the output of MD5 is only 128 bits.

True **False** Moore's law implies that you should change your password once every 18 months.

Solution: False. If your password is very long to begin with, then there may be no reason to change it for a long time.

True **False** The perfect security of a one-time MAC means that an attacker can never guess the MAC for any given message.

Solution: False. It simply means that the adversary chance of *guessing* the MAC is $1/p$, where p is the range of the MAC. There is still a non-zero chance of a correct guess.

True **False** The design of the Feistel network requires that the round function F be invertible. (F is the function such that $R' = L \oplus F(R)$.)

Solution: False. The design of the Feistel network allows one to decrypt, even when F is not invertible.

True **False** Define $h(x) = g^{(x^2)} \pmod{p}$, where p is a large prime and g is a generator mod p. The function h is collision-resistant.

Solution: False. Both 1 and -1 hash to the same value.

Problem Q1-4. Multiple Choice [12 points]

(a) If an efficient algorithm for calculating discrete logs over an integer modulus is discovered, which of the following schemes will be known to be not secure (circle all that apply):

I AES

II Diffie-Hellman

III RSA

IV El Gamal

Solution Note: RSA will be insecure, because given a plaintext/ciphertext pair $(m, c = m^e \pmod N) = (c^d \pmod N, c)$ (which the adversary can compute himself), one can discover the secret decryption key d by computing the mod N discrete log of m, base c.

(b) If an efficient algorithm for factoring large numbers is discovered, which of the following schemes will be known to be not secure (circle all that apply):

I AES

II Diffie-Hellman

III RSA

IV El Gamal

(c) A hash ring for a hash function h is a set of values $v_1, \ldots, v_t$ such that $v_{i+1} = h(v_i)$ for $1 \le i \le t-1$, and $v_1 = h(v_t)$. (Two hash rings are the same if they have the same values listed in a different order.)

How many hash rings does SHA-1 have? Pick the *most specific, correct* answer.

I Unknown

II 0

III 1

IV 2

V 1 or more (nobody knows which)

VI 2 or more (nobody knows which)

Solution Note: There must be at least 1 hash ring, because if we start with any 160-bit input and iteratively hash, we must get a repeated value at some point. However, there may only 1 hash ring (e.g., if the ring contains all hash values), or there may be more.

Problem Q1-5. Hot Poker [10 points]

For his 6.857 final project, Ben Bitdiddle decides to implement a poker server, so that he can play friendly games with his chum(p)s across campus. The process works as follows: the server "deals" five cards (selected at random from a deck of 52 cards) to each player as their "hand," removing each card from the deck as it is dealt. The cards are known to their owner but must remain secret to the other players.

Each client then sends the values of their bet to the server. Finally, the server broadcasts each client's cards, and the best hand wins (for this problem, it is not necessary to know anything about the rankings of poker hands). This is the conclusion of the hand.

The process is then repeated with a new, complete deck until Ben and his friends get tired of playing.

For security, Ben decides to use some encryption. Client i keeps a long-lived secret AES key K_i, which the server also knows. During the dealing phase of the game, the server sends $AES_{K_i}(C)$ (using ECB mode) to client i for each card C in that client's hand (a card is represented in some canonical way by an integer in the range $0, \ldots, 51$). When the cards are broadcast at the end of the hand, the server sends them in the clear to all clients.

(a) Explain one attack that can be executed by a *passive* malicious player who is able to *eavesdrop* on the network.

Solution: Since each card is sent as a single block with ECB encryption, there is a one-to-one mapping between the ciphertexts and the plaintexts. The passive malicious player can watch the encrypted cards go by and build up a codebook mapping these encrypted blocks to unencrypted cards.

(b) Suggest a fix to this problem. You get an extra point if your fix does not increase the length of the messages from the server to the clients.

Solution: Since each card can be encoded as a 6-bit value, and since the AES block size is 128 bits, the simplest solution is for the dealer to encrypt $C \circ R$ where R is a random 122-bit block. The player software would automatically remove the nonce.
Other acceptable answers were to use CBC mode. We liked CBC answers that specified how the IV was set — for example, by using the hand number as the IV.
Yet another acceptable answer was to specify a key schedule for the encryption key. For example, specifying that each key $K_i = \text{SHA-1}(K_{i-1})$.
We took off a point for students who suggested using Diffie-Helman for exchanging new AES keys each hand. This works, but it is extremely inefficient relative to the original scheme.

Problem Q1-6. Tripwire [10 points]

Tripwire is a program that is used to assure the integrity of the files on a Windows or Unix workstation. In one incarnation, the Tripwire executable uses the computer's cryptographic library to compute the MD5 hash code for every file in directories that are specified by the Tripwire configuration file. Tripwire then creates a database that consists of each file's name and MD5 code. This database is stored on the computer's hard disk.

Ben configures Tripwire to compute the MD5 values of all the programs in his `/bin` and `/usr/bin` directories. A month later, he runs the Tripwire program again to see if any of the MD5 codes have changed. The database that is generated perfectly matches the database of MD5 values generated when Ben first ran Tripwire. But unbeknownst to Ben, his files have been compromised by an attacker.

Describe three attacks that are consistent with this scenario, and how you would defend against them.

Solution: There were many correct answers, including:

- Change the database, so that it contains the MD5 values of the modified files, rather than the original files.
- Change the function that calculates MD5, so that it returns the old values, rather than the new ones.
- Hack the kernel so that the Tripwire program thinks it is reading the files in the `/bin` and `/usr/bin` directories, when it is in fact reading other files (presumably the original copies).
- Hack the Tripwire program, so that it says that nothing is wrong, when in fact something is.
- Hack Ben's operating system, so that Ben sees the Tripwire program report that everything is okay, when in fact it is reporting something else.

One answer was not accepted. Students did not receive credit for arguing that MD5 is weak and that the attacker could simply modify the new files so that they have the same MD5 as the original files. Any student who lost credit for this answer is invited to produce two different files that have the same MD5 hash code.

Problem Q1-7. Pad Distribution [10 points]

Arnold Schwarzenegger is setting up branch offices all over the United States to assist other actors in becoming governors. This is a very delicate operation, and Alice, his head of communications security, has decided to use a one-time pad to encrypt all communications from the branch offices back to central office.

Because one-time pads are expensive to distribute by courier, Alice plans to distribute to each office j a unique key k_j. This key will be used with AES to encrypt a separate one-time pad that will be sent over the Internet to each office. The offices will then use the one-time pad to encrypt the messages that they send back.

(a) Eve is an unbounded adversary. Describe how she could decrypt messages sent with this scheme from the branch offices to headquarters, and what additional information (if any) she would need to do so. The less extra information she needs, the more points you will receive.

Solution: Here was the most popular solution: Eve can try all possible AES keys, produce all 2^{128} possible pads, and then see which of these pads produces reasonable messages when it is XORed with the bitstream sent from the branch office to HQ.
Here is a slightly harder solution: if Eve can obtain a matched ciphertext and plaintext pair (under the one-time pad), she can learn part of the pad. She can then decrypt the encrypted pad under all AES keys, and learn the actual pad (it is the one that matches the partial pad). At this point, she can read all the messages.
No credit was given to students who said that the one-time pad would probably be used twice. The problem said that the pads were only used once.

(b) In order to save on bandwidth costs, Alice decides to redesign her system. Now she distributes the one-time pad to each of the offices by courier. The one-time pad is used to send a 128-bit AES key to each office. This AES key is then used to encrypt the email messages that are sent back to headquarters.

Now describe how an unbounded adversary could decrypt the messages to headquarters.

Solution: The email messages are encrypted with AES. Bob simply tries all possible keys until he find the one that works. He can ignore the one-time pad.

SCRATCH PAPER — Please detach this page before handing in your quiz.