
Midterm

Submit this take-home midterm in PostScript, PDF, or MS Word format to `6857-submit@csail.mit.edu` before lecture on the due date. Each problem solution must appear on separate sheets of paper. Mark the top of each sheet with your name, the course number (6.857), the question number, and the date.

You must complete this midterm *individually*. You are not allowed to discuss or collaborate with your classmates in any way. You are only allowed to discuss the midterm with the TAs or Professor Rivest. You are free to use materials from books, papers, or the web, but you must cite any published resources you use. Cheating or plagiarism will result in a zero score and will be reported to MIT's Dean of Student Affairs.

Problem 1. Traditional Public-Key Encryption versus Identity-Based Encryption [15 Points]

In a traditional public-key cryptosystem, anyone may invoke a key generation algorithm G to produce a public/secret key-pair (PK, SK) . Each individual can make their own public key PK available through some Public-Key Infrastructure (PKI), as you did with your PGP key in the first problem set.

An encryption algorithm E allows anyone to encrypt a message m with a public key PK . A decryption algorithm D allows only the holder of a secret key SK to decrypt a ciphertext encrypted under the corresponding PK . This triplet of algorithms (G, E, D) have the following abstract definitions:

$$\begin{aligned} G(\cdot) &\rightarrow (PK, SK) \\ E(PK, m) &\rightarrow c \\ D(SK, c) &\rightarrow m \end{aligned}$$

Note that each of these algorithms has an implied source of random bits. In contrast, an Identity-Based Encryption (IBE) system consists of four algorithms (MKG, G, E, D) . First, a trusted third-party will invoke a master-key generation algorithm MKG to produce a master public key PK_m and a master secret key SK_m . The value PK_m will be made globally available, while SK_m will be kept secret by the trusted third-party. Given any plaintext identifier ID and SK_m , this trusted third-party can generate individual secret keys SK_{ID} using a key generation algorithm G .

Rather than encrypting messages under individual public keys, in an IBE scheme the encryption algorithm E only needs an ID and the global public key PK_m as input. Ciphertexts encrypted for ID can be decrypted using a decryption algorithm D that takes SK_{ID} as input. An IBE system is summarized as follows:

$$\begin{aligned} MKG(\cdot) &\rightarrow (PK_m, SK_m) \\ G(SK_m, ID) &\rightarrow SK_{ID} \\ E(PK_m, ID, m) &\rightarrow c \\ D(SK_{ID}, ID, c) &\rightarrow m \end{aligned}$$

Alice and Bob work for a large company that wants to let customers send encrypted messages to its employees. Bob suggests using a traditional public-key cryptosystem. He proposes that each employee generates a public/secret key-pair and posts the public-key in a directory on a company's public web server.

- (a) [4 points] Explain how Bob's system works. List all the initial one-time setup steps for the system, each employee, and each customer. After this initial setup, list all the steps necessary for a customer to send an encrypted message to an employee. For example, a single step might be "The customer generates a key-pair (PK_c, SK_c) ".

- (b) [4 points] Alice suggests using an IBE system where SK_{ID} values are distributed by an internal server that is only available to employees on the company's closed intranet. In her system, each employee's ID would simply be his or her e-mail address. Explain how Alice's system works. List all the initial one-time setup steps for the system, each employee, and each customer. After this initial setup, list all the steps necessary for a customer to send an encrypted message to an employee.
- (c) [7 points] Discuss the comparative advantages and disadvantages of Alice's and Bob's proposals. You may want to consider how their respective systems fare in some of the scenarios listed below. Please keep your answers brief and limit your discussion to the given scenarios:
- A new employee is hired and must be added to the system.
 - An employee is fired and must have his or her keys revoked.
 - The company's web server is vulnerable to a security exploit.
 - An employee loses his or her secret key.
 - The company merges with another company with either the same PKI, a different PKI, or no PKI at all.
 - A corrupt insider or disgruntled ex-employee wants to deny service or try to read other people's encrypted e-mail.
 - The company receives a subpoena to provide decrypted plaintext of the CEO's encrypted e-mail. (You may wish to consider this from the viewpoint of both a crooked CEO and a corporate governance officer.)

Problem 2. Hash Functions [10 points]:

Bob designed the following hash function h that works in the finite field $GF(p)$, where p is a large prime (e.g. 256-bits long). That is, all arithmetic is done modulo p . Your goal in this problem will be to show that h is neither one-way nor collision-resistant.

There is also available a highly complex and nonlinear function f that maps \mathbb{Z}_p one-to-one to itself, and such that both f and its inverse f^{-1} are efficiently computable by anyone. (There are no "secret parameters" for f .) It is assumed that a message M to be hashed is divided into a sequence of elements (M_1, \dots, M_n) , where each element M_i is in \mathbb{Z}_p . That is, $0 \leq M_i < p$ for all i .

The hash function has an internal state vector (x, y) , where each of x and y are in \mathbb{Z}_p . The state vector is initially set to $(x_0, y_0) = (0, 0)$. Then each message element M_i is processed in turn; causing the state to be updated to some value (x_i, y_i) . The final state, after M_n is processed, is (x_n, y_n) . The desired hash value $h(M)$ is just x_n .

To process M_i and update the state from (x_{i-1}, y_{i-1}) to (x_i, y_i) takes three steps, resulting in intermediate states (r_i, s_i) , (t_i, u_i) , and (finally) (x_i, y_i) . These steps are:

•Add in message to both halves of state.

Set $(r_i, s_i) = (x_{i-1} + M_i, y_{i-1} + M_i)$.

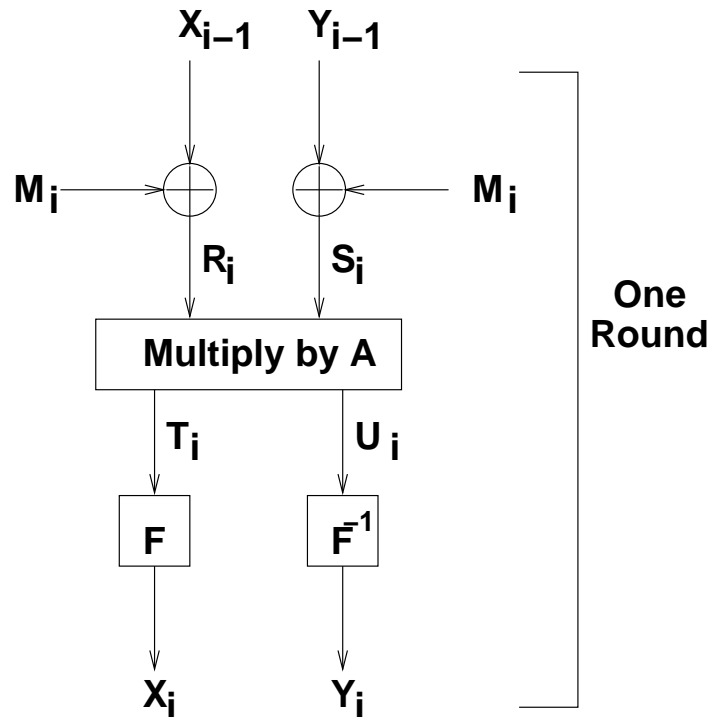
•Multiply (r_i, s_i) by matrix $A = ((a, b), (c, d))$.

Set $(t_i, u_i) = A(r_i, s_i) = (ar_i + bs_i, cr_i + ds_i)$, where A is non-singular (i.e. invertible) in $GF(p)$. (For consistency in your answers, let the inverse matrix A^{-1} be represented as $((a', b'), (c', d'))$ if you need it.) These matrices A and A^{-1} are of course public.

•Apply non-linear function f and f^{-1} to each half.

Set $(x_i, y_i) = (f(t_i), f^{-1}(u_i))$.

The following diagram illustrates this process:



Show that h is not one-way and not collision-resistant.

Problem 3. Web Authentication [5 Points]

Consider a web authentication system where after a user logs in with a user ID and password, the web server sets a cookie-based authenticator using a Message Authentication Code (MAC) as follows:

$$\text{Cookie} = (\text{user ID}, \text{expiration time}, \text{MAC}_k(\text{user ID} \oplus \text{expiration time}))$$

The user ID and expiration time are fixed-length binary integers. The key value k is a key known only to the web server. The \oplus symbol denotes the XOR operation.

- (a) [3 points] Suppose you know a valid cookie of the form (u, e, m) , where e is drawn from a uniform distribution. (This may be because the value e is determined by some arbitrary epoch, rather than by human time.) Suppose we know some other valid user identity v such that $u \neq v$. Show how to selectively forge a valid cookie for v and some expiration date $e' \geq e$ with probability at least $1/2$.
- (b) [2 points] Propose a similar web authentication scheme that addresses this selective forgery problem. Would your scheme still be secure if the MAC output were malleable?

Problem 4. One-time MAC's [10 Points]

Alice and Bob have agreed upon a sequence K_1, K_2, \dots , of shared secret keys that they wish to use to authenticate the messages they send back and forth to each other.

They are planning to use a “one-time MAC” scheme, as explained in 6.857. Each key K_i consists of a pair (a_i, b_i) of values in $\mathbb{Z}_p^* \times \mathbb{Z}_p$, so that the MAC of message M computed with the key K_i is just $MAC(K_i, M) = a_i M + b_i \pmod{p}$; here p is a publicly known large prime.

Alice and Bob agree on the following protocol for managing their use of keys. (Of course, they understand that each key K_i should not be used to MAC more than one message.)

Both parties work to keep track of the index t most recently used by either themselves or their counterparty. Then, when Alice sends a message M , she increments her t by one and then MAC's M with key K_t . The message she sends has the format:

$$(\text{“Alice”}, \text{“Bob”}, M, t, MAC(K_t, M))$$

When Bob receives this message, he checks that the MAC is correctly computed for the given value of t . If it is not correct for that value of t , he ignores the message altogether. If the MAC is correct for that value of t , Bob accepts the message and sets his local value of t to be equal to the t of the received message.

The same procedures are followed symmetrically (with roles switched) when Bob sends to Alice. (Each party only maintains a single value of t , however.)

- (a) [4 points] Identify as many vulnerabilities as you can in the above protocol design. You may assume that the messages are sent on a communications channel controlled by an active adversary that may insert, delete, modify, or replay messages on the channel.
- (b) [6 points] Propose a revised protocol that doesn't suffer from these problems.

Problem 5. Phun with ϕ [10 points]

- (a) [2 Points] Recall the definition of the ϕ function from class. Suppose that you are given a number n and know that it is the product of two distinct odd primes p and q . Which is “harder”, factoring n or computing $\phi(n)$ (or are they the same difficulty)? (If given the answer to one problem you can easily find the answer to another, the first problem is at least as hard as the second.)
- (b) [2 points] A *generator* g modulo prime p is an element of \mathbb{Z}_p^* of order $p-1$; any of the $p-1$ elements relatively prime to p can be expressed as a power of g . Recall that Fermat's theorem states that for any a relatively prime to prime p , $a^{p-1} = 1 \pmod{p}$. A more general theorem (due to Euler) states that for any n , if a is relatively prime to n , then $a^{\phi(n)} = 1 \pmod{n}$. Thus, it is natural to think that there might be “generators” modulo composite numbers as well—that is, elements of order $\phi(n)$ modulo n .

Assume that $n = pq$ where p and q are distinct odd primes. Find the smallest example you can where g is a generator both modulo p and modulo q , but is *not* a generator modulo n .

- (c) [3 points] Letting p and q be two distinct odd primes and $n = pq$, prove that there are no generators for \mathbb{Z}_n^* . A fact you should consider is that if g has order a modulo p and g has order b modulo q , then g has order $lcm(a, b)$ modulo n , where lcm is the “least common multiple” function.
- (d) [3 points] Again, let n be the product of two distinct primes. A “super-generator” is an element g such that

$$\mathbb{Z}_n^* = \{g^{\frac{a}{b}} : a \in \mathbb{Z}_{\lambda(n)}, b \in \mathbb{Z}_{\lambda(n)}^*\}$$

where $\lambda(n) = lcm(p-1, q-1)$. Note that $\mathbb{Z}_{\lambda(n)}$ is simply the integers \mathbb{Z} modulo $\lambda(n)$. If g generates \mathbb{Z}_n^* via rational powers (not just integral powers, as for an ordinary generator) then it is a super-generator. Note that $g^{\frac{1}{b}}$ can be defined as g^c where $b * c = 1 \pmod{\lambda(n)}$.

Show that \mathbb{Z}_n^* has no “super-generators”.

Problem 6. Password Storage [10 points]

We mentioned in class that a common method for implementing a password login system was for the server to keep a hash of each password locally. This way, an attacker who compromises the password file ideally learns nothing about the users' passwords. In practice, however, it is common to do things a bit differently: one typically stores a random *salt* s for each user along with $h(s||password)$.

- (a) [3 points] Assume that the hash function that is used is ideal. By “ideal” we mean a hash function has all the desirable properties of the oracle or “elf-in-a-box” model, including one-wayness and collision resistance. From a *theoretical* perspective, is there a security benefit to including the salt for a file that contains a single password? Why or why not?
- (b) [3 points] What is the *practical* benefit of using salts?
- (c) [4 points] You wish to use a password-based login system on a single server. Assume there are controls in place that prevent users from choosing weak passwords. Suppose an adversary is able to acquire the privileges necessary to read the password file. Although there are solutions involving multiple servers or trusted third-parties, is it possible for a single server to provide a higher level of security than what the password-salting system offers? Why or why not?

Problem 7. Secret-sharing: Sharing Shares [10 points]

The Peppy-Cola company maintains the secret formula for its soda drink in an electronic safe. The safe only opens when authorized to do so by an authorized set of employees. During the opening protocol, a set T of employees wishing to open the safe insert their ID smart-cards into the safe, which determines if T is authorized. (The notion of “authorized” is of course monotonic, so that if a set S is authorized, then so is any superset of S .)

Peppy-Cola's authorization policy may be more complex than can be handled by ordinary “secret-sharing” schemes, which typically only allow any t out of n specified individuals to be authorized. For example, Peppy-Cola may wish to allow the safe to be opened if any 3 of their 7 vice-presidents agree, but any vice-president may designate a set of “personal alternates”, such that if any two of his alternates agree, it is as if the VP himself agreed to request the safe to be opened. Indeed, Peppy-Cola wants total flexibility as to what authorization policies are allowed.

The electronic safe supports recursive tree-based secret sharing: a policy is represented by a tree, where the root is the main safe-opening key. Each node in the tree represents a secret. The leaf nodes represent secrets that may be stored on individual employee's ID cards. (A card may store more than one leaf-secret.) Each non-leaf node (including the root) has a single associated secret which is not stored anywhere, but which can be recomputed given a sufficient number of secrets from its children. (The threshold as to what is a “sufficient number” may vary from node to node.) Given enough leaf-secrets, some of the intermediate nodes can be computed, bottom up. If the root node secret is computed, the safe opens up.

Argue that the policies supported by the electronic safe are sufficiently expressive that any (monotonic) authorization policy from Peppy-Cola can be handled.

Hint: Consider the set of “minimally authorized sets” – sets of employees that are authorized but which are not authorized if any one employee is removed from the set.

Problem 8. Academic Honesty [2 points]

Write a couple of sentences to testify that you have not collaborated with anyone on this midterm and that you have cited all your sources.