Introduction to Cryptography and Data Security
Problem Set 1

Handed: Wednesday, October 31, 2007
Due: Wednesday, November 14, 2007

Problem 1:

Joe Smart has a text from an English book. The text is n-characters long, and it contains only lower case characters. There are no white spaces in the text – for example: "I was here" ➔ "iwashere". Joe decided to encrypt his text using a variant of the One-Time-Pad scheme. He assigned each of the 26 letters in the English alphabet a unique numeric value between 0 and 25 (with '**a**' being 0, '**b**' being 1, and so on, up to '**z**' being 25).

Recalling the One-Time-Pad scheme, Joe decided to generalize the XOR operation (which is good for bits) into a modular addition operation (which is also good for numeric values). In Joe's scheme, the i-th character of the ciphertext, C[i], is obtained by adding in modulo 26 the i-th character of the cleartext, M[i], and the i-th character in the key, K[i]. That is – $C[i] = M[i] + K[i] \pmod{26}$.

For efficiency, Joe selected a secret key (pad) K of length m characters (for $m << n$). The characters of the key were also taken from the 26-symbol alphabet described above. But, since the key was shorter than the text to be encrypted, Joe decided to reuse the key as follows. The first time, the pad will be used as is. The second time, the pad will be reused by adding 1 (modulo 26) to each of its m characters. The third time, the pad will be reused by adding 2 (modulo 26) to each of its m characters. And so on. This transformed the scheme into a Many-Time-Pad scheme, which enabled Joe to reuse the key many times before it expired.

In our example, the file is n>300 characters long, and the key is m=20 characters long. This implies that, in this scheme, we will use the pad at least 15 times. The encrypted version of the file is:

```
wuhklwkxdldsaxitinoh
pmawshxuzxyzpqjbfkgn
anyauefynbvecjzotxqs
zxkzpgzlybfrvulcdiab
mqwmqikrwlotvenmmjvi
wqbkvrmydccszsjejkrx
wekdoacmgcvejxbwbgbo
jsdetkdnhzomjqdtyury
xhigaihpjxavwxfyvuwx
tdtiarqnqpccchblueba
izhfovgvzfdoybzwkuyo
uvrcjlheftmqwnibzqqt
zrrhdyjnvybifcjbqtqr
uctmyvakbvcpqdwvafkt
szfyudgydlbhbjzadgpa
ddhllqjtafinij
```

You need to break Joe's Many-Time-Pad scheme – that is you need to find the content of Joe's text file, and you need to find Joe's key. Explain how your result was derived. (Do not use exhaustive computer searches on all the possible 20-character keys and/or the 300-character files!!!) You can use scripts or programs in any computer language you like, or you can make calculations by hand. Submit also your scripts / calculations.

Problem 2:

Below is a proposed block cipher that has plaintext and ciphertext blocks of size 64-bits. Let the bit positions of the plaintext and ciphertext blocks be labeled from 0, starting with the low-order bit, up to 63, which is the high-order bit. The secret key shared between the sender and the receiver consists of three parts:

(a) An initial permutation P from {0, 1, …, 63} to {0, 1, …, 63}.

(b) A 64x64 binary linear transformation K.

(c) A final permutation T from {0, 1, …, 63} to {0, 1, …, 63}.

The block cipher works as follows. Let the input plaintext message be the 64-bit vector A[0..63]. Then, create the ciphertext block C as follows:

1. Create an intermediate vector B1[0..63] by applying the permutation P to the vector A:

   for j = 0 .. 63 **do** B1[P[j]] = A[j].

2. Create an intermediate vector B2 by applying the binary linear transformation K:

   for j = 0 .. 63 **do** B2[j] = K[j,0] * B1[0] + ... + K[j,63] * B1[63] (mod 2)

3. Create the ciphertext vector C[0..63] by applying the permutation T to B2:

   for i = 0 .. 63 **do** C[T[i]] = B2[i].

Analyze the security of this block cipher scheme against the three types of attacks identified in class: (i) known ciphertext, (ii) known plaintext, and (iii) chosen plaintext.

Problem 3:

Consider the following specification of a Feistel Cipher. The input block M is 6 bits long, the output block C is 6 bits longs, the key K is 3 bits long (and its 3 bits are denoted below as K[3] K[2] K[1] from left to right), and the number of rounds is r = 1. The function F takes 3 bits as inputs and produces 3 bits as outputs. The function F consists of three steps: (1) a cyclic shift by 1 step to the left of its 3 input bits; (2) a bitwise AND operation between the resultant 3-bit vector and the 3-bit key vector; and (3) another cyclic shift by 1 step to the left of the resultant 3-bit vector. Calculate the cipher output for the following two messages: 001100 and 0101010. Show your calculations.

Problem 4:

In DES, there are four keys that are called "weak keys". For each of these four keys, the first half of the key after the Permuted-Choice-1 is either all 0's or all 1's, and the second half of the keys after the Permuted-Choice-1 is either all 0's or all 1's. Explain why, if K is a weak key, then for each plaintext block X we have – $E_K(E_K(X)) = X$**.**

Introduction to Cryptography and Data Security
Problem Set 2

Handed: Wednesday, November 14, 2007
Due: Wednesday, November 28, 2007

Problem 1:

In generating pseudo-one-time-pads using (a) blocks of 64 bits of a 19-bit LFSR, (b) blocks of 64 bits of the OFB mode of DES, and (c) blocks of 64 bits of a 64-bit hash function, we generate a pseudo random stream of fixed-sized blocks. This stream must eventually repeat. Why? Will the first block in the sequence necessarily be the first one to be repeated? Is there any difference between the LFSR, the OFB mode of DES, and the hash function?

Problem 2:

Consider the following message encryption scheme that uses a block cipher algorithm E and a symmetric secret key K. To encrypt a message M, it is first broken into fixed-size blocks $M_1\ M_2\ \ldots\ M_L$, where each block M_i can be input to the encryption function $E_K()$. Then, a random IV block is picked and the following L cipher blocks are computed:
$C_1 = E_K(M_1 \oplus IV)$, and
$C_i = E_K(\ M_i \oplus M_{i-1} \oplus C_{i-1}\ )$, for $2 \le i \le L$.

(a) Describe the corresponding decryption scheme, which, given the IV block and the L cipher blocks $C_1\ C_2\ \ldots\ C_L$, recovers the L plaintext blocks $M_1\ M_2\ \ldots\ M_L$.

Now consider the following proposal to use a variation of the above scheme as a combined encryption and MAC scheme. Given the message $M = M_1\ M_2\ \ldots\ M_L$, the sender picks a random IV block and computes the L cipher blocks $C_1\ C_2\ \ldots\ C_L$, as described above. The sender then sends IV and the L cipher blocks $C_1\ C_2\ \ldots\ C_L$. The receiver can recover the L plaintext blocks $M_1\ M_2\ \ldots\ M_L$, and it can also check the integrity of the message by checking that the value C_L is indeed as expected.

(b) Explain and demonstrate why the above proposal for a combined encryption and MAC scheme is not secure.

(c) Will the scheme be more secure if, in addition to sending IV and $C_1\ C_2\ \ldots\ C_L$, the sender will also compute and send the following value $C^* = E_K(C_L)$, and the receiver will use this value to verify the integrity of the message? Explain and demonstrate your answer.

Problem 3:

A MAC scheme is considered secure if an adversary cannot create a valid Message / MAC pair, after having seen the MACs of a number of other messages of his choice. Following is a description of a MAC scheme based on the block cipher DES:

1. Pad the message by appending a "0" bit, and then enough "1" bits to make the message's total length be a multiple of 64 bits.
2. Break the padded message up to 64-bit blocks.
3. Encrypt each block M_x separately, yielding an intermediate ciphertext $I_x = E_k(M_x)$.
4. Encrypt the number of the block and XOR it with the intermediate ciphertext to yield a final ciphertext block $C_x = I_x \oplus E_k(x)$.
5. Take the high order 10 bits of each ciphertext block, and concatenate them for the final MAC. This makes a MAC of length approximately 1/6 of the length of the message.

(a) Explain why the padding operation is done the way it is, rather than just adding one bits as necessary to fill up the last block.
(b) Show that this MAC scheme is not secure against a Truncation Attack.
(c) Show that this MAC scheme can be attacked by the Birthday Attack.
(d) Show a third attack on this MAC scheme. In this attack, the adversary need not ask for many MAC's.
(e) Show a fourth attack on this MAC scheme. In this attack, the adversary will cut, rearrange, and paste the results of one MAC computation to yield another valid MAC. (The message for the first MAC may need to be chosen appropriately.)
(f) Which of the above four attacks still work if the XOR step is moved earlier, so that we XOR the message block with its number, encrypt the result, and take the low order ten bits. (Explain briefly as necessary.)
(g) Which of the above four attacks still work if the XOR is moved to the key input, so that the x-th message block is encrypted with key $k \oplus x$, and then the low order 10 bits of the result are used? (Assume that DES is suitably "random".)

Problem 4:

Definition 1: A hash function is called *weakly collision-free* if, given a message x, it is computationally infeasible to find a message $x' \neq x$ such that $h(x') = h(x)$.

Definition 2: A hash function is called *strongly collision-free* if it is computationally infeasible to find messages x and x' such that $x' \neq x$ and $h(x') = h(x)$.

Suppose $h_1: \{0,1\}^{2m} \rightarrow \{0,1\}^m$ is a strongly collision-free hash function.

1. Define $h_2: \{0,1\}^{4m} \rightarrow \{0,1\}^m$ as follows:
   - write $x \in \{0,1\}^{4m}$ as $x = x_1 \| x_2$, where $x_1, x_2 \in \{0,1\}^{2m}$, and
   - define $h_2(x) = h_1(h_1(x_1) \| (h_1(x_2))$.

   Prove that h_2 is strongly collision-free.
2. For any integer $i \geq 2$, define a hash function $h_i: \{0,1\}^{2im} \rightarrow \{0,1\}^m$ recursively as follows:
   - write $x \in \{0,1\}^{2im}$ as $x = x_1 \| x_2$, where $x_1, x_2 \in \{0,1\}^{2(i-1)m}$, and
   - define $h_i(x) = h_1(h_{(i-1)}(x_1) \| (h_{(i-1)}(x_2))$.

   Prove that h_i is strongly collision-free.

Introduction to Cryptography and Data Security
Problem Set 3

Handed: Wednesday, November 28, 2007
Due: Wednesday, December 12, 2007

Problem 1:

In the El-Gamal Digital Signature Scheme, how does knowing the secret number used for a signature reveal the signer's private key? How do two signatures using the same secret number reveal the signer's private key? [Hint: Assume that p is a **strong prime**, that is p is a prime and also $(p-1)/2$ is a prime. Even though not all numbers have inverses modulo p-1, division can still be performed if one is willing to accept two possible answers.]

Problem 2:

Suppose two parties Alice and Bob want to communicate privately. They both hold public keys in the traditional Diffie-Hellman model. An eavesdropper Eve stores all the encrypted messages between them and one day manages to break into Alice and Bob computers and find the private keys, which correspond to their public keys. Show how using only public key cryptography we can achieve *perfect forward secrecy*, i.e., Eve will not be able to gain any knowledge about the messages Alice and Bob exchanged before the disclosure of their secret keys.

Problem 3:

Remember that an RSA public-key is a pair (n, e), where n is the product of two primes. The encryption of a message M is computed as follows:

$$\mathrm{RSA}_{(n, e)}(M) = M^e \bmod n.$$

Assume that three users in the network, Alice, Bob, and Carl, use RSA public-keys $(n_A, 3)$, $(n_B, 3)$, and $(n_C, 3)$ respectively. Suppose David wants to send the *same* message M to the three of them. So, David computes:

$$y_A = M^3 \bmod n_A\,,\ y_B = M^3 \bmod n_B\,,\ \text{and } y_C = M^3 \bmod n_C\,,$$

and sends each ciphertext to the appropriate user.

Show how an eavesdropper Eve can now compute the message M even without knowing any of the secret keys of Alice, Bob, and Carl.

Problem 4:

Alice and Bob are good friends. In order to save time, they agree to simply find one good pair of primes, p and q, and therefore to use the same public key modulus $n = pq$. Of course, to preserve privacy and to not have any confusion over who signed which message, they select different exponents e_a and e_b.

Show that, in this system, it is possible to decrypt a message, M, sent to both of them if $\gcd(e_a, e_b) = 1$. That is, given:

$$C_a = M^{e_a} \pmod n \quad \text{and} \quad C_b = M^{e_b} \pmod n$$

an adversary can compute M.

Introduction to Cryptography and Data Security
Problem Set 4

Handed: Wednesday, December 12, 2007
Due: Wednesday, December 26, 2007

Problem 1:

Because of the known risks of the UNIX password system, the SunOS documentation recommends that the password file be removed and replaced with a publicly readable file called "/etc/publickey". An entry in the file for a user A consists of a user identifier ID(A), the user's public key Pub(A), and the corresponding private key Prv(A). This private key is encrypted using DES with a key derived from the user's login password PW(A). When A logs in, the system decrypts the stored value of $E_{PW(A)}[Prv(A)]$ to obtain Prv(A). The system then verifies that PW(A) was correctly supplied.

a. How does the system verify the password?
b. How can an opponent attack this system?

Problem 2:

Assume that Alice and Bob know each other's public key. Design an authentication protocol that uses exactly two messages (one message from Alice to Bob and one message from Bob to Alice) and accomplishes both mutual authentication and establishment of a session key.

Problem 3:

We examine a protocol for symmetric key exchange between two stations A and B. We assume that stations A and B have a symmetric encryption algorithm E and a corresponding decryption algorithm D. In addition, stations A and B hold a long-lived symmetric key K that is used for authentication and for session key establishment. Each time A and B wish to converse, they execute the following 4-message authentication protocol:

1. A sends to B: A, R_1 (the name of A and a random challenge R_1)
2. B sends to A: B, R_2 (the name of B and a random challenge R_2)
3. A sends to B: $E_K(R_2)$ (the encryption of R_2 with algorithm E and key K)
4. B sends to A: $E_K(R_1)$ (the encryption of R_1 with algorithm E and key K)

Explain which of the following choices for a session key are good and which are not good.

(a) $K \oplus R_1 \oplus R_2$
(b) $E_K(R_1 + R_2 + 1)$
(c) $D_K(R_1 + R_2 + 1)$
(d) $E_K(R_1 \oplus R_2)$
(e) $D_K(R_1 \oplus R_2)$
(f) $E_K(K \oplus R_1 \oplus R_2)$
(g) $D_K(K \oplus R_1 \oplus R_2)$

Problem 4:

In this question, you are asked to devise three mutual-authentication protocols and argue their correctness in detail.

a) Devise a protocol for mutual authentication, under the assumption that both Alice and Bob have synchronized clocks and share a common secret. Use symmetric-key crypto. Try to use the smallest number of messages possible.
b) Devise a protocol for mutual authentication, under the assumption that both Alice and Bob have synchronized clocks and share a common secret. Use hash functions instead of encryption. Try to use the smallest number of messages possible.
c) Devise a protocol for mutual authentication, in which Alice has a public/private RSA key pair and where Alice's public-key is known only to Bob.

Introduction to Cryptography and Data Security
Problem Set 5

Handed: Wednesday, December 26, 2007
Due: Wednesday, January 9, 2008

Problem 1:

In standard Kerberos, the AS (Authentication Server) generates a TGT (Ticket Granting Ticket) for user A at workstation WS. Design a variant of Kerberos in which the workstation WS generates the TGT. How will the TGT be protected? How will the TGT be understood by the TGS (Ticket Granting Server)? How does your scheme compare with standard Kerberos in terms of efficiency and security? What happens in each scheme if the user changes her password during a login session?

Problem 2:

Suppose we use Kerberos to secure electronic mail. The obvious way of accomplishing this is for Alice, when sending a message to Bob, to obtain a ticket for Bob and include that in the email message, and encrypt and/or integrity-protect the email message using the key in the ticket. The problem is that, in this solution, the KDC gives Alice a quantity encrypted with Bob's password-derived master key, and then Alice could do off-line password guessing. How might Kerberos be extended to email without allowing off-line password guessing?

Problem 3:

Analyze and describe all the steps that are required to update a CRL by a certificate authority. Focus on the following aspects, and describe the steps taken and the data structures created.
(a) Receiving a certificate cancellation request.
(b) Canceling a certificate by including it in a CRL.
(c) Distributing the CRL.
(d) Managing the CRL's..

Problem 4:

In Public-Key systems, one gains trust in other entities by receiving and verifying public-key certificates for those entities. Such certificates are issued and signed by a root Certification Authority. To verify the CA's public key, one usually holds a root certificate that contains the CA's information and public key. One of the properties of a root certificate is that it has an expiration date (like other certificates).
(a) Explain why the root certificate needs to expire.
(b) Explain what security flaw this design poses.
(c) Propose a method for solving this problem.

Introduction to Cryptography and Data Security
Problem Set 6

Handed: Wednesday, January 16, 2008
Due: Wednesday, January 30, 2008

Problem 1:

Suppose that communication from A to B is to be protected by IPSec. Suppose also that the Security Associations for such communication has been established and is kept both in A and in B. Suppose that these SA's dictate encryption and then authentication of IP packets from A to B. Describe all the steps that the IPSec stack on A takes from the time it starts handling an IP packet, which is destined at B, until the time that a corresponding protected IPSec packet leaves A to B. Describe all the steps that the IPSec stack on B takes from the time it receives the IPSec packet, which came from A, until the original IP packet is delivered to a layer above IPSec. Your answer should discuss the SPD, SAD, anti-replay mechanisms, encryption steps, authentication steps, headers of the packets at different stages, etc.

Problem 2:

Consider the following threats, and describe how each is countered by SSL.

(a) Brute Force Cryptanalytic Attack – An exhaustive search of the key space for a conventional encryption algorithm.

(b) Known Plaintext Dictionary Attack – Many messages will contain predictable plaintext, such as HTTP GET command. An attacker constructs a dictionary containing every possible encryption of the known-plaintext message. When an encrypted message is intercepted, the attacker takes the portion containing the encrypted known plaintext and looks up the ciphertext in the dictionary. The ciphertext should match against an entry that was encrypted with the same secret key. If there are several matches, each of these can be tried against the full ciphertext to determine the right one. This attack is especially effective against small key size.

(c) Replay Attack – Earlier SSL handshake messages are replayed.

(d) Man-in-the-Middle Attack – An attacker interposes during key exchange, acting as the client to the server and as the server to the client.

(e) Password Sniffing – Passwords in HTTP or other application traffic are eavesdropped.

(f) IP Spoofing – User forges IP addresses to fool a host into accepting bogus data.

(g) IP Hijacking – An active and authenticated connection between two hosts is disrupted and the attacker takes the place of one of the hosts.

(h) SYN Flooding: An attacker sends TCP SYN messages to request a connection but does not respond to the final message to establish the connection fully. The attacked TCP module typically leaves the "half-open connection" around for a few minutes. Repeated SYN messages can clog the attacked TCP module.

Problem 3:

Design a protocol that achieves "Plausible Deniability" in electronic mail. In such a protocol, A would like to send a message M to B. The protocol should guarantee the secrecy of the message M between A and B, and it should authenticate to B both the source A and the integrity of the message M. However, whenever B receives an encrypted and authenticated message M from A with this protocol, B will not be able to prove to a third party that A had indeed sent him that message (i.e., A will be able to deny having sent the message M to B).

Problem 4:

A *timestamping service* provides users with the ability to place a "signature" on a document that includes the time it was seen by the Time Server. (This service can be useful in proving that a patent is invalid due to prior art, or that a paper was written before a deadline, etc.) There have been several proposed systems that achieve some form of timestamping. List several requirements from such a service, and describe a potential implementation of this service that meets the desired requirements.