Homework 1 Solutions

CS161 Computer Security, Fall 2008
Assigned 9/15/08
Due 9/24/08

1 Encryption

1. (1 point) Which of the following security goal(s) does encryption address : (1) Confidentiality (2) Integrity (3) Sender authentication (4) Non-repudiation.

   Solution. Encryption aims to provide confidentiality only.

2. (1 point) A one-time pad is provably secure. What makes it hard to use in practice?

   Solution. One-time pad requires the key length to be same as the message length. The difficulty of setting up a shared key each time users wish to communicate makes it hard to use in practice.

3. (1 point) Suppose you obtain two ciphertext C, C' encrypted using one-time pad, with key K and its bitwise complement $\bar{K}$ respectively. What can you infer about the corresponding plaintext messages?

   Solution. You can obtain $m \oplus m'$. Note that $\bar{K} = K \oplus 1$, therefore,

   $C \oplus C' \oplus 1 = (m \oplus K) \oplus (m' \oplus K \oplus 1) \oplus(1)$

   $= m \oplus m'$.

4. (1 points) What main advantage does public-key encryption offer over secret-key encryption.

   Solution. Public-key cryptography eleminates the need for a shared secret key between each pair of users that communicate. For 'N' users, public-key cryptography requires N (private,public) key pairs, whereas secret key cryptography requires on the order of N^2 shared keys.

2 Block Ciphers

This question contains a small programming assignment that is designed to ensure that your named UNIX accounts are properly set up for CS161 while also illustrating some properties of block ciphers.

We've published a code skeleton with a number of BMP image files[1] at :
`http://inst.eecs.berkeley.edu/~cs161/fa08/Homeworks/hw1-program.tar.bz`.

Untar the file by running `gtar jxf hw1-program.tar.bz`, and run `make` to compile the program. Then type

```
./encrypt plaintext-medium.bmp out.bmp
```

to to attempt to encrypt the image. The skeleton will pass a null pointer into `write()` and fail with a "bad address" error until you fill in the solution to the assignment.

We've tested the code on Linux and on `cory.eecs.berekeley.edu`. We will use `cory.eecs.berkeley.edu` to grade the assignment, so make sure that your code builds and runs on that machine.

The two bitmap files contain uncompressed versions of the same image. One contains a monochrome (1 bit per pixel) version of the picture, and the other contains a 200% scaled RGB (24 bits per pixel) version of the picture.

When you've completed the question, submit the files `encrypt_ecb.c`, `plain.bmp`, `crypt.bmp`, and `encrypt_cbc.c`. Instructions for electronic submissions will be given on the website later this week.

1. (3 points) Edit `encrypt_ecb.c` so that it implements DES encryption in electronic codebook (ECB) mode. Encrypt the two included image files.

   Solution. We expect to see a discernible pattern for both images.

2. (3 points) The skeleton intentionally avoids encrypting the BMP file's header so you can use standard software to inspect the encrypted data. Open the two encrypted files in your favorite image editor. (We used the GIMP.) Was the encryption effective? Explain the difference between the two encrypted files.

   Solution. No, ECB mode encryption always encrypts the all-white or all-black regions of the file as the same pattern. The encrypted version of the smaller, monochrome file format can store 64 pixels per ECB block, while only 64/24 pixels in the larger image fit in an ECB block. Therefore, the larger file is easily readable, while the smaller file's text is scrambled.

3. (3 points) Find a small (under 1MB uncompressed) image file that would be more effectively encrypted using a block cipher in ECB mode. Explain your choice of image. What sorts of attacks is this scheme vulnerable to?

   Convert the image to an uncompressed (not RLE encoded) 24-bit BMP format and encrypt it. Name the files `plain.bmp` and `crypt.bmp` and submit them.

   Solution. We were looking for a photograph, or other image that is unlikely to contain identical runs of pixels. In addition to revealing repetitions within a single image, this scheme is vulnerable to repetitions across multiple images. Also, the image's blocks could be rearranged. CBC encryption is a much better choice for this type of application.

[1] From `http://xkcd.com/`.

4. (3 points) Edit `encrypt_cbc.c` so that it implements DES encryption in cipher block chaining (CBC) mode. For this exercise, a block of all 0's may be used as the initialization vector. Encrypt the two original image files and view the result.

   How do the results differ when CBC is used instead of ECB? Briefly state why that is the case.

   Solution. CBC's output should be indistinguishable from random noise because it encrypts each block of the file based upon the values of previous blocks.

3 RSA Public-Key Encryption

1. (5 points) Alice and Bob are using public keys (e_1, N_1), (e_2, N_2) respectively. Suppose you are informed that their RSA moduli N_1, N_2 are not relatively prime. How would you break the security of their subsequent communication? It is sufficient to show that you can get $\phi(N_1)$ and $\phi(N_2)$.

   Solution. Since N_1, N_2 share a common divisor (they are not relatively prime), let p be the shared prime divisor. Let $N_1 = p.q$ and $N_2 = p.r$.

   $p = GCD(N_1, N_2)$, which can be computed effeciently using Euclid's Algorithm. We do not expect the solution to outline the algorithm.

   $q = N_1/p$ and $r = N_2/p$ can be computed with the knowledge of p.

   $\phi(N_1) = (p-1)(q-1)$ and $\phi(N_2) = (p-1)(r-1)$ can be computed with the knowledge of p, q, r.

2. (5 points) Suppose that a system uses textbook RSA encryption. An attacker wants to decrypt a ciphertext c to obtain the corresponding confidential plaintext m. Assume that the victim system readily decrypts arbitrary ciphertexts that the attacker can choose, except for ciphertext c itself. Show that the attacker can obtain m from c even under this setting, i.e a chosen ciphertext attack is possible.

   Solution. We know that $c = m^e \ mod \ N$,i.e, the attacker aims to obtain m from c. To do this, the attacker uses the following scheme.

   - **Step 1**. The attacker chooses a random number 'r' such that $gcd(r, N) = 1$. The attacker encrypts it as using the public key : $C_r = r^e \ mod \ N$.
   - **Step 2**. He computes $C' = c.C_r \ mod \ N$, and asks the system to decrypt C'. Let the decryption be M'. By definition of RSA encryption, we know that :
     $M' = (C')^d \ mod \ N$.
     It follows that:
     $C' = (\ (m^e \ mod \ N) \ . \ (r^e \ mod \ N)\ ) \ mod \ N$.
     $= (m.r)^e \ mod \ N$
     Therefore,
     $M' = C'^d \ mod \ N$
     $M' = (m.r)^{ed} \ mod \ N$
     $M' = (m.r) \ mod \ N$

- **Step 3**. The attacker obtains M' and recovers the confidential plaintext m by computing $M'.r^{-1} mod\ N$.
  $M'.r^{-1} mod\ N = m.r.r^{-1}\ mod\ N$
  $= m\ mod\ N$.

Keep in mind that r has to be chosen such that its multiplicative inverse modulo N exists. This is true iff $gcd(r, N) = 1$.

3. (**Extra Credit**, 5 points). Show that an attacker who discovers the private key (d, N) for a public key (e=3,N), can efficiently factor $N = p.q$. Recall, we compute $ed = 1(mod\ \phi(N))$, such that $d < \phi(N)$.

   Solution. Suppose $N = pq$, and its given that $e = 3$. We know that :

   $ed = 1\ mod\ N$. So, there exists a positive integer 'k', such that,

   $ed = k\phi(N)$ +1 or $ed- 1 = k\phi(N)$.

   We also know that $d < \phi(N)$. So, $3d < 3\phi(N)$ or $ed - 1 < 3\phi(N)$ (for $e = 3$). This implies that either $k = 1$ or $k = 2$.

   Since p and q are primes, we know that $p\ mod\ 3 \neq 0$ and $q\ mod\ 3 \neq 0$. Therefore we can construct the following table for all possible values taken by $p\ mod\ 3$ and $q\ mod\ 3$.

$p\ mod\ 3$	$q\ mod\ 3$	$\phi(N)\ mod\ 3$
1	1	0
1	2	0
2	1	0
2	2	1

   Claim $k \neq 1$. Proof by contradiction.

   For $e = 3$, ed is a multiple of three. So, $ed\ -1 = 2\ mod\ 3$. If $k = 1$, then $ed\ -1 = \phi(N)$. From the table above, we can see that $\phi(N) \neq 2\ mod\ 3$. So, $k = 1$ is not possible.

   So, $k = 2$. We know that $\phi(N) = (ed - 1)/(2)$. It is easy to compute factorization of N, given $\phi(N)$

4 Hash Functions

You are designing a multi-user OS. In your OS, users log into their respective accounts using passwords. It is dangerous to store user passwords in a file on the computer, because someone who obtains the file gets access to all passwords. As a solution, you decide to store the username and corresponding password's hash value in the file called `hpasswd`. Assume that you use an idealized, perfectly random hash function $h(x)$, i.e h is selected uniformly at random from all functions mapping $\{0,1\}^*$ to $\{0,1\}^k$. As always, h is publicly known.

When a user logs in with password p, the OS grants access to the user iff $h(p)$ matches the entry for that user in `hpasswd`. Assume $k = 20$ in your OS.

There are some weaknesses in this mechanism of your OS. A collision in the password hash of 2 users, allow them to log in as each other.

1. (5 points) Suppose an attacker (a normal user) wishes to log in as the system administrator (the superuser) by trying random passwords (without repeating a guess he has already tried). What is the minimum number of password guesses that the attacker has to try to have a success probability greater than 0.6%.

   Solution. The attacker never retries a guess which he has tried before (and since the hash function is public, in a stronger attack he never uses a guess that hashes to a value that he has previously generated). Probability of success after 't' tries = 1 − Probability of failure after 't' tries. Let $P\{F_t\}$ be the probability of the event that the attacker fails after 't' attempts, and let $P\{S_t\}$ be the probability of success in 't' attempts. Then, we know that $P\{S_t\} = 1- P\{F_t\}$

   $P\{F_1\} = (2^k - 1)/2^k$

   $P\{F_2\} = ((2^k - 1)/(2^k)) \times ((2^k - 2)/(2^k - 1)) = (2^k - 2)/2^k$

   $\vdots$

   $P\{F_t\} = (1 - t/2^k)$

   $P\{S_t\} = 1 - P\{F_t\} = t/2^k$

   For $P\{S_t\} > 0.006$,

   $t > 0.006 * 2^k$

   Substituting k = 20,

   $t = 2^{12.62} = 6295.04.$

   The answer is approximately 6295

2. (5 points) You want to limit is the probability of user password collisions, to below 20% in your design. That is, the probability of *any* two users' password hashes matching should be below 20%. What is the maximum number of users (n) you should allow in your OS?

   Solution. . Let the probability of choosing 'n' distinct passwords for each of the 'n' users are added, be denoted by the variable A_n

   $P\{A_n\} = (1 - 1/2^k) * (1 - 2/2^k) * (1 - 3/2^k)...(1 - (n - 1)/2^k).$

   In this question, you want to ensure $1 - P\{A_n\}$ to be lesser than 0.2.

   $1-P\{A_n\} < 0.2$

   $P\{A_n\} > 0.8$

   Using the approximation, $e^x > 1 + x$, when $|x| < 1$,

   $e^{-1/2^k} e^{-2/2^k} \ldots e^{-(n-1)/2^k} > 0.8$

   $e^{-\Sigma_{i=1}^{n-1}(i/2^k)} > 0.8$

$e^{-n(n-1)/2^{k+1}} > 0.8$

Taking natural logarithm on both sides,

$-n(n-1)/(2^{k+1}) > -0.223$

For $k = 20$,

$n < 685$ (approximately).

.

5 MAC and Signatures

1. What is non-repudiation?

   Solution. Non-repudiation is the concept of ensuring that a party in a dispute cannot repudiate, or refute the validity of a statement or contract.

2. (7 points) Suppose Alice has to sign a contract with Mallory using an RSA signature where, the signature is computed by $s = (h(m))^d \ mod \ N$ (d is Alice's private key). Assume that the hash function is an idealized, perfectly random hash function $h(x)$, i.e h is selected uniformly at random from all functions mapping $\{0,1\}^*$ to $\{0,1\}^k$, and h is publicly known. Mallory has managed to find 40 distinct places where he can make a slight change in the contract: adding a space at the end of line, adding a comma, replacing with equivalent words (like replacing "agreed to pay" with "is obliged to pay"), etc. Surely, Alice does not object any such minor change in the contract and is willing to sign a contract with any of these minor changes.

   - (2 points) How many possible versions of the contract can Mallory generate that Alice would be willing to sign?

     Solution. For each slight change, Mallory can choose to use it or not. There are 40 such changes. So, Mallory can generate 2^{40} contracts that Alice will be willing to sign.

   - (3 points) Mallory creates a fraudulent contract reflecting a substantial increase in amount that Alice owes to Mallory. He computes the hash of the fraudulent contract as $h(f)$ and finds a version of the good contract that hashes to the same value $h(f)$. What is a minimum safe output size for the hash function (i.e k =?) to make these collisions unlikely? (An approximate answer with appropriate reasoning is acceptable. You need not show any calculations to get 'k'.)

     Solution. Given that Mallory was able to compute 2^{40} different versions of the good contract, we compute how likely is it that one of them hashes to the hash value of his fraudulent contract. Let the size of the hash output be 'k' bits.

     The probability that the first good version hashes to the fraudulent one, is $(1/2^k)$. The probability that the first good version does *not* hash to the same hash as the fraudulent one is $(1 - (1/2^k))$. Therefore, the probability that none of the 2^{40} good contracts hash to the same value as the hash of the fraudulent case, is

$(1 - 1/2^k)^{2^{40}}$. From this it follows that the probability that atleast one of the 2^{40} versions has the same hash as the fraudulent one is $1 - (1 - 1/2^k)^{2^{40}}$.

You can choose how unlikely the event of such collisions should be. For $k = 40$, this probability becomes 63%. For $k = 41$, this becomes nearly 40%. For $k > 50$, the probability is less than 0.1%.

- (2 point) What can Mallory do by finding such a collision, to force Alice to pay an increased amount in court?

  Solution. Mallory finds a version out of the 2^{40} good contracts, that hashes to the same value as the fraudulent contract. He asks Alice to sign the version of the good contract, and Alice complies. Since the hash values are same, the signatures for the fraudulent contract and the good version will be same. Mallory presents the fraudulent contract appended with the signature of the good version in court, to claim that Alice had signed the fraudulent version.

3. (5 points) Recall from class that in an RSA signature scheme, the signature is computed by $s = (h(m))^d mod\ N$, where d is the private key. Consider an naive revised scheme where the signature is computed as $s' = m^d mod\ N$ instead. Show that it is possible to forge signatures for some messages in the latter (revised) scheme.

   Hint : Suppose Alice uses the latter (revised) scheme to sign two contracts m_1, m_2, generating signatures s'_1, s'_2 respectively. Construct an attack from s'_1, s'_2.

   Solution. Suppose a bank signs messages m_1 and m_2, to generate signatures s'_1 and s'_2. An attacker can forge the bank's signature for the message $m_1.m_2$, by simply multiplying the given signatures $s'_1 s'_2\ mod\ N$.