0368.3049.01 **Winter 2008**

Introduction to Modern Cryptography

Benny Chor and Rani Hod

Assignment #2

Published Sunday, February 17, 2008 and **very slightly revised** Feb. 18. Due Tues., March 4, in Rani Hod's mailbox (second floor, Schreiber building). S

This assignment contains seven "dry" problems and one "wet" problem. Efficient solutions are always sought, but a solution that works inefficiently is better than none. The answers to the the "wet" problem should be given as the output of a `maple` or `xmaple` session.

Problem 1: Substitution Ciphers

The following text has been enciphered by a simple substitution cipher; just decode it.

```
xbsafsjeozjzwohthhswqyvxztwhfccsbwoosc
msbstcqyshvzodscboycmsbzjzwohthvscmyqh
```

It should be rather easy once you have a partial key. The PATTERN program at the following link could give you a lead:
`http://www.und.nodak.edu/org/crypto/crypto/words/Pattern.lists/pattern.english.zip`

Problem 2: Enhancing DES

The following two keys enhancements to DES were proposed in order to increase the complexity of finding the keys by exhaustive search.

$$DESV_{k,k_1}(M) = DES_k(M) \oplus k_1,$$

$$DESW_{k,k_1}(M) = DES_k(M \oplus k_1)$$

The keys lengths is $|k| = 56$ and $|k_1| = 64$ (k_1 is the same length as the block length). Show that both these proposals do not increase the complexity of breaking the cryptosystem using brute-force key search. That is, show how to break these schemes using on the order of 2^{56} DES encryptions/decryptions. You may assume that you have a moderate number of plaintext-ciphertext pairs, $C_i = DESV/W_{k,k_1}(M_i)$.

Problem 3: Meet in the Middle
In lecture 2 we described a *meet in the middle* attack against *double* DES. The attack required 2^{56} decryptions, 2^{56} encryptions and storage for 2^{56} messages (the decryptions of the ciphertext under all possible keys), 64 bits each. The attack used a small number of plaintext/ciphertext pairs: M_i and $C_i = DES_{k_2}(DES_{k_1}(M_i))$.

You were hired to perform the same task, only your employer, hurt by the recent market trends, has supplied you with a machine capable of storing only 2^{40} words of 64 bits each. How many encryption and decryption operations do you need in order to recover the secret key k_1, k_2 with high probability. Does the number of required plaintext/ciphertext pairs increase?

Problem 4: Cryptographic Hash Functions
Let $m = m_1 m_2 \dots m_n$ where for every i $(i = 1, \dots, n)$, m_i is a 128 bits binary string. Define a hash function, H to operate on messages of this form.

- h_0 is defined as the all zero string of length 128.
- for every i, $1 \leq i \leq n$, define $h_i = AES_{m_i}(h_{i-1})$.
- $H(m) = h_n$.

a. Show how to find collisions for H (namely two different messages that are mapped by H to the same string) using approximately 2^{64} AES applications.

b. Given a random string m, show how to find a different string m' such that $H(m) = H(m')$, using approximately 2^{64} AES applications.
Hint: Recall the attack on double DES.

Problem 5: Claw Free Permutations
Two permutations $f_0, f_1 : D \mapsto D$ is called *claw free* if it is infeasible to find $x, y \in D$ such that $f_0(x) = f_1(y)$.
a. Let p be a prime number, g a primitive element in Z_p, and $a \in Z_p^*$. Define the two permutations $f_0, f_1 : Z_p^* \mapsto Z_p^*$ by $f_0(x) = g^x \pmod p$ and $f_1(y) = ag^y \pmod p$. Assume it is infeasible to find a z such that $g^z = a$. Prove that f_0, f_1 are claw free permutations.

b. Let $m = b_1 b_2 \dots b_n$ be an n bit message (the b_i's are bits). Let f_0, f_1 be claw free permutations on D. Define the function H by

$$H(m) = f_{b_1}(f_{b_2} \dots (f_{b_n}(IV)\dots)) \ ,$$

where IV is the all zero string in D. For example, if $m = 011$ then $H(m) = f_0(f_1(f_1(IV)))$.
Assume that it is infeasible to find a $z \in D$ such that $f_0(z) = IV$ or $f_1(z) = IV$. Prove that H is a collision resistant hash functions. In other words, show that if $m_1 \neq m_2$ and $H(m_1) = H(m_2)$, then we can *efficiently* either find a pair $x, y \in D$ such that $f_0(x) = f_1(y)$, or a $z \in D$ such that $f_0(z) = IV$ or $f_1(z) = IV$. **Note that** m_1 and m_2 can have different lengths.

Problem 6: CBC MACs and variable length messages
In this problem we will explore the security of CBC MACs when the length of the message is allowed to vary. The constructions use a block cipher, $E : \{0,1\}^k \times \{0,1\}^n \mapsto \{0,1\}^n$ which you should assume to be secure ($E_K(t)$ is the encryption of n length t under k length key K).

In general, let $\mathbf{x} = x_1, x_2, \ldots, x_\ell$, where for each i, $x_i \in \{0,1\}^n$. For all the variants considered in this problem, the authentication of the message $\mathbf{x}$ is defined as the concatanation of $\mathbf{x}$ with $MAC_K(\mathbf{x})$, where K is the secret key (shared by Alice and Bob), and $MAC_K(\mathbf{x})$ is of length n.

We say that Fred, the forging adversary, succeeds if after seeing a small number of messages $\mathbf{z_1}$,$\mathbf{z_2}, \ldots, \mathbf{z_s}$ *of his choice* and their MACs under the *unknown* secret key K, he can produce a new message $\mathbf{w}$ ($\mathbf{w} \notin \{\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_s}\}$) together with $MAC_K(\mathbf{w})$. We emphasize that w can, and typically will, be constructed out of pieces depending on the z_i's. By small number we mean s is either a constant or at most a (fixed) polynomial in n, the block length of $\mathbf{x}$. In addition to the number s of message/MAC pairs, Fred is also limited to polynomial time computations (polynomial in n).
Remark: This type of forgery is called *adaptive existential forgery* (adaptive since the choice of each z_{i+1} can depend on all previous i message/MAC pairs, and existential because it demonstrates the existence of a message whose MAC can be forged). This is the strongest form of "reasonable adversary" considered in the crypto world.

a. Consider the application of "regular" CBC MAC to messages of arbitrary length. Formally, given $\mathbf{x} = x_1, x_2, \ldots, x_\ell$, we define $y_0 = 0^n$, and for $0 \leq i \leq \ell - 1$, $y_{i+1} = E_K(y_i \bigoplus x_{i+1})$. Then $CBC - MAC_K(\mathbf{x}) = y_\ell$. Show that this MAC is completely insecure: Break it with a constant number of queries.

b. In order to overcome the problem of applying "regular" CBC MAC to messages of arbitrary length, consider the following patch:

$$MAC_K(x_1, x_2, \ldots, x_\ell) = CBC - MAC_K(x_1, x_2, \ldots, x_\ell, \ell) \ ,$$

where ℓ, the number of blocks in $\mathbf{x}$ is written in binary using n bits. Show

that this patch does not hold water either: Break it with a constant number of queries.

c. Consider the following attempt to allow one to MAC messages of arbitrary length. The domain for the MAC is $(\{0,1\}^n)^+$. To MAC the message $\mathbf{x} = x_1, x_2, \ldots, x_\ell$, under the secret key (K, K'), compute $CBC - MAC_K(\mathbf{x}) \bigoplus K'$, where K has k bits and K' has n bits. Show that this MAC is completely insecure: Break it with a constant number of queries.

Problem 7: Orders

a. Let a, m be two positive integers, with $1 \leq a \leq m-1$. The order of a modulo m, $ord_m a$, is defined as the minimum positive integer ℓ such that $a^\ell = 1 \pmod{m}$, and ∞ if no such ℓ exists. Prove that $ord_m a < \infty$ if and only if $\gcd(a, m) = 1$.

b. Let x be an integer, and let p be an odd prime divisor of $x^{16} + 1$. Prove that $p = 1 \pmod{32}$.

Problem 8: Primitive Elements in Z_p

a. Let $p > 2$ be a prime number and let g be a primitive element in Z_p^*. Find a characterization of all integers e, $1 \leq e \leq p-2$ such that $g^e \bmod p$ is also a primitive element in Z_p^*. Prove the characterization.

b. Let $p > 2$ be a prime number and let $g \neq 0$ be an element in Z_p^*. Suppose $p_1, \ldots, p_k$ are all the prime factors of $p-1$. Show that g is a primitive element iff for all i, $1 \leq i \leq k$, $g^{(p-1)/p_i} \neq 1 \pmod{p}$. Show that this enables to test efficiently if g is a primitive element, provided the factorization of $p-1$ is known.

c. Unfortunately, factoring integers is a hard problem, even if they are of the form $p-1$. Take $p = 2^{521} - 1$. Using Maple's `isprime(·)` for testing primality of an integer, verify that p is indeed a prime. Now apply Maple's `ifactor(·)` to $p-1$. This procedure tries to factor its integer argument. For large integers it obviously not always succeeds. Neither does it always succeeds for primes minus 1. However for our p, `ifactor` produces the complete factorization after a few minutes (even on Benny's MacBook). Produce this factorization.

d. For $p = 2^{521} - 1$, find a **random** integer g, $g > 10^7$, such that g is a primitive element of Z_p but $g+1$ is *not* a primitive element of Z_p. Print a Maple session that explicitly proves both statements. (The probability that among the fewer than 100 g's submitted by the course participants there will be a collision is so small that we will interpret such collision as a collusion and will act accordingly.)

Hint: You can make the search easier by using Maple's commands

$$FF{:=}\,\texttt{GF}(p,1);$$

$$x := FF[\texttt{ConvertIn}](g);$$

$$FF[\texttt{isPrimitiveElement}](x);$$

for verifying g's "primitivity status" in the field $Z_p = GF(p,1)$. However in your "explicit proof" you should *not* use such `FF` implementation, but rather use only the `mod` p function and &^ (exponentiation) to appropriate powers.

e. Instead of looking for a prime p and trying to factor $p-1$, a different procedure is to look at random for a prime q and then test if $p = 2q+1$ is also a prime. In that case, the only factors of $p-1$ are 2 and q. You may think that having both q and $2q+1$ being primes is such a rare event that we will never run into one. Write a short Maple code that prints out *two* such pairs, with $q > 2^{400}$ in both cases. You can definitely use `isprime(·)` here.