Tel Aviv University 1/12/2011 Benny Applebaum

Introduction to Modern Cryptography (0368.3049.01) – Ex. 2

Submission in pairs on 14/12/11, 4pm

5 bonus points will be given to printed (or exceptionally clear and organized) submissions.
When analyzing the complexity of your attacks you may treat encryption and decryption in AES/DES as unit-cost operations. Similarly, "basic operations" such as string-comparison, table look-up, etc, can also be treated as unit-cost.

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

$$\begin{aligned} \mathrm{DES^V}_{\mathrm{k,k_1}}(\mathrm{M}) &= \mathrm{DES_k}(\mathrm{M}) \oplus \mathrm{k_1}, \\ \mathrm{DES^W}_{\mathrm{k,k_1}}(\mathrm{M}) &= \mathrm{DES_k}(\mathrm{M} \oplus \mathrm{k_1}), \end{aligned}$$

   where the keys' lengths are $|k| = 56$ and $|k_1| = 64$ (k_1 has 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, $C_i = \mathrm{DES^V}/\mathrm{DES^W}_{\mathrm{k,k_1}}(\mathrm{M_i})$

2. (**Tweaking AES**) This question deals with a variant of AES. Recall that in AES (Rijndael), a round of encryption consists of the following four operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. For each of the following changes to AES, describe an efficient attack that breaks the new cipher. Here "breaking" means distinguishing modified-AES from a random function with complexity significantly smaller than 2^{128} applications of the cipher.

   (a) All the SubBytes operations are omitted from the cipher.

   (b) All the MixColumns operations are omitted from the cipher.

   (c) All ShiftRows operations are omitted from the cipher.

   (d) All operations of the same type are put together, i.e., the encryption is changed to: 10 SubBytes, followed by 10 ShiftRows, followed by 9 MixColumns, followed by 11 AddRoundKey.

3. (**MACs**) In the following we describe several constructions of MACs which are all based on a block-cipher E. Assume that the block-cipher is AES with key k and that all messages are "full", i.e., each message can be written as $M = (M_1, \ldots, M_n)$ where M_i is 128-bit long and n is some integer. (In the real world this can be guaranteed via padding.) Break the following schemes with less than 5 queries to the MAC (and with probability 1). Here "break" refers to existential forgery as defined in class.

   (a) OFB-MAC. Namely, for a message M consists of n blocks of 128 bits $M = (M_1, \ldots, M_n)$ and an AES-key k of 128-bits we define the MAC $\mathrm{OFB-MAC_k(M)}$ as follows:

- Initialize $S_0 = 0^{128}$.
- Let $S_i = E_k(S_{i-1})$ and $C_i = S_i \oplus M_i$.
- Output C_n.

Show that this MAC is insecure even if all messages are restricted to have the same number of blocks n.

(b) $\mathrm{OFB} - \mathrm{MAC_k(M, \langle n \rangle)}$ where $\langle n \rangle$ is the 128-bit binary representation of n the length of M. In your attack, you may use messages with varying number of blocks.

(c) a modification of $\mathrm{OFB} - \mathrm{MAC_k(M)}$ in which the initialization vector S_0 is chosen uniformly at random and is appended to the output of the MAC as part of the tag.

(d) **Bonus:** Attack $\mathrm{ECBC} - \mathrm{MAC_k(M)}$ with complexity 2^{65} and success probability $1/2$.

4. (**Claw Free Permutations**) In this question we will show how to construct collision-resistance hash function based on the discrete-log assumption.

Definition: Two permutations $f_0, f_1 : D \to D$ are called *claw free* if it is infeasible to calculate $x, y \in D$ such that $f_0(x) = f_1(y)$.

(a) Show how to construct claw free permutations based on the discrete-log assumption. **Guidance**: Let p be a prime number, g be a generator of Z_p^*, and $a \in Z_p^*$. Define two permutations $f_0, f_1 : Z_p^* \to Z_p^*$ by $f_0(x) = g^x \bmod p$ and $f_1(y) = ag^y \bmod p$. Prove that f_0, f_1 are claw free permutations assuming that it is infeasible to calculate a z such that $g^z = a$.

(b) Show how to construct collision-resistance hash function H from claw free permutations. **Guidance**: Define the function H by

$$H(M) = f_{M_1}(f_{M_2} \ldots (f_{M_n}(\mathrm{IV}) \ldots)),$$

where IV is the all zero string in D and M_i is the i-th bit of M. For example, if $M = 011$ then $H(M) = f_0(f_1(f_1(\mathrm{IV})))$. Prove that H is a collision resistant hash function for fixed-length messages. In other words, show that given two equal-length messages $M \neq M'$ for which $H(M) = H(M')$, we can *efficiently* find a pair $x, y \in D$ such that $f_0(x) = f_1(y)$. Finally, transform H to variable-length hash function via the general transformation showed in class.