1. Study the Rijndael (AES) encryption function and the design decisions taken to optimize it in implementation from the handout on the class webpage.


(b) Explain the functionality of each of the four basic operations, ByteSub, ShiftRow, MixColumn, AddRoundKey in the cipher.

(c) See how the round function can be implemented by just four table lookups and four XORs on a 32-bit platform. Discuss which properties of the diffusion part make this feature possible. (For instance, would a similar feature be possible if a bit-wise permutation were used for diffusion? Or, if a non-linear operation with no matrix representation were used for MixColumn? Or, what if a “MixRow” similar to MixColumn were used instead of ShiftRow?) Is the same trick applicable on an 8- or 16-bit platform? Why/why not?

(d) i. Is the performance of the inverse cipher as good as the cipher itself? On a 32-bit platform? On a 8-bit platform?

ii. Why is the performance of the inverse cipher not as important as the performance of the cipher itself according to the designers of Rijndael?

2. Consider the following mode of encryption with three keys $k$, $k_1$, $k_2$, where $k$ is of the length of the block size and $k_1$ and $k_2$ are of the length of the key size (denoted by $\ell$) of the block cipher $E$. (E.g., for DES, $k$ would be 64 bits, and $k_1$ and $k_2$ would be 56 bits each.)
(a) Describe the decryption operation for this mode of encryption.

(b) Describe a chosen-ciphertext attack where the attacker can discover the full key \((k, k_1, k_2)\) with \(O(2^\ell)\) runs of the encryption/decryption algorithm. You can assume as much memory as you need for the attack. (Hint: Consider two ciphertext messages \((C_1, C_2)\) and \((C'_1, C_2)\), for some randomly chosen \(C_1, C_2, C'_1\).)

(c) Would a similar chosen-plaintext attack work? Argue briefly.

3. Recall from the DESX construction that for a block cipher \(F\) with an \(n\)-bit key and \(\ell\)-bit block size, \(FX\) is defined by

\[
FX_{k,k_1,k_2}(x) = F_k(x \oplus k_1) \oplus k_2
\]

where \(k \in \{0,1\}^n\), \(k_1, k_2 \in \{0,1\}^\ell\).

Show that the simplified constructions

\[
FY_{k,k_1}(x) = F_k(x \oplus k_1)
\]

\[
FZ_{k,k_1}(x) = F_k(x) \oplus k_1
\]

do not increase the strength of the cipher against exhaustive search. That is, show that \(FY\) and \(FZ\) each can be broken using in the order of \(2^n\) operations. (You can assume that a moderate number of known plaintext-ciphertext blocks are available for your attacks.)

4. In this exercise, you will prove that the CBC-MAC in its plain form is not secure to authenticate variable-length messages.

(a) Consider the CBC-MAC scheme with an \(n\)-bit block cipher where the CBC checksum of a message is calculated with a zero IV. Describe an attack where an attacker Eve can construct the MAC of a message different from those she obtained from the legitimate sender. (Hint: Let the attacker obtain the MAC of two \(n\)-bit messages and from them compute the MAC of a \(2n\)-bit message.)

(b) An attempt to solve this problem could be to append the number of blocks in the message as a final block to the message; i.e., to apply CBC on \((M\|b)\) instead of \(M\) alone, where \(b\) denotes the number of blocks in \(M\). Show that this construction is not secure either. (Hint: Let the attacker obtain the MAC of some more messages of his choice.)

(c) How about prepending the number of blocks; i.e., to apply CBC on \((b\|M)\)? Does a similar attack work on this construction as well? How would its performance compare to the appending scheme of part (b)?

5. Hash functions: Answer the following questions regarding MD5 and SHA-1:

(a) Note that a message is still padded even if its length is already a multiple of the block length. Why is this important? i.e., what would be the problem if such messages are digested as they are without any padding?
(b) Discuss the relation between these hash functions and the Davies-Meyer construction based on a block cipher.

(c) Why do you think byte operations such as and, or, xor are used instead of S-boxes in the nonlinear F function? What would happen if a structure like the DES F function were used instead of F?