Problem Set 2

CNT 5412 — Dr. Sudhir Aggarwal — Fall 2018

David Miller

Problem 1. Suppose that Alice and Bob use the following protocol to confirm that they are both in possession of the same secret symmetric key. Alice creates a random bit string the length of the key, XORs it with the key, and sends the result to Bob. Bob XORs the result he receives from Alice with his key (which should be the same as Alices key) and sends it back. Alice checks to see if she has received the original random bit string. If so, then she has verified that Bob has the same secret key. Explain how Alice is convinced that Bob has the same key she does. Note that neither party has ever transmitted the key. Is there a flaw in this scheme? If so explain the flaw.

Solution: Given an arbitrary message m and key k, it is easy to see that $(m \oplus k) \oplus m = k$. Therefore Alice will know with certainty if Bob has the same key if the message she receives is the same as the random one she generated. Now suppose adversary Eve impersonates Alice and sends random message r to Bob. Bob then sends back $r \oplus k$ to Eve which she can easily XOR with her original message to learn 1) the secret key k and 2) the protocol being used is just $m \oplus k$ for any message m. Eve can now employ this reflection attack to trick Alice into thinking she is Bob.

Problem 2. This problems explores some issues in encryption schemes and counter mode.

(a) Explain what it means for an encryption algorithm to be "malleable.

(b) Suppose you know that a bit stream cipher $K[i]$ encrypts the plaintext $M[i]$:

$$C[i] = K[i] \oplus M[i] \text{ where } i \text{ is the } i\text{th bit of the message.}$$

You intercept the ciphertext below (spaces for readability only) as part of a sting on bank operations.

01101010010000110101010111010101

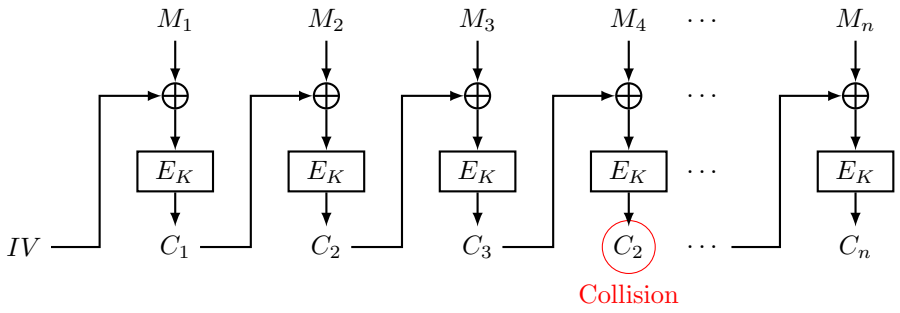
(c) Is counter mode malleable? Explain briefly.

Solution: Malleability of an encryption algorithm refers to the fact of altering a ciphertext, or "shaping" it, so that it produces a different but similar plaintext when decrypted. Let $f(c)$ denote the mapping from character to ASCII byte value. Then "\$501" in ASCII is $f(\$)||f(5)||f(0)||f(1) = 00100100\ 00110101\ 00110000\ 00110001$. If we XOR this with the intercepted message we get key $K = 01001110\ 01110110\ 01100101\ 11100100$. We can now shape our desired ciphertext by $(f(\$)||f(0)||f(0)||f(1)) \oplus K$ and getting 01101010 01000110 01010101 11010101. Since CTR mode uses a stream s and message m to produce ciphertext $c = m \oplus s$, an adversary can easily flip a bit in m. For example an adversary can do $c' = c \oplus 01000000$ for an 8 bit ciphertext, effectively flipping the second bit of m. The decryption of CTR would use the same stream s and therefore c' would decrypt to $m' = m \oplus 01000000$. This shows that CTR is malleable.

Problem 3. Suppose a very long message is encrypted using CBC with DES. A "collision" is when two encrypted blocks are identical. How many blocks would be needed in the message so that the likelihood of a collision is more than 50%? Explain. Assume that a collision does occur. Explain the problem that arises using a CBC diagram and the equation that arises because of the collision and could be used to determine part of the plaintext.

Solution: This is a direct application of the Birthday Attack. Given block size N, we can bound the probability of a collision within $0.3\frac{q(q-1)}{2^N} \leq \Pr[\text{collision}] \leq 0.5\frac{q(q-1)}{2^N}$, given q queries. Adversary A must

satisfy $q \leq \sqrt{2^{N+1}}$ for the lower bound to hold true. Setting $q = 2^{N/2}$, we can see that A achieves at best 50% accuracy. Therefore if A wants a collision to happen with at least 50% chance, then they will have to query more than $2^{N/2}$ times (but not significantly more). The Sweet32 Attack is the known birthday attack on CBC with DES.



Since block ciphers just create permutations over input domain and are deterministic, if $C_i = C_j$ for $i \neq j$ we get $C_i \oplus M_{i+1} = C_j \oplus M_{j+1}$. Another way to rewrite this is $M_{i+1} \oplus M_{j+1} = C_i \oplus C_j$. Now without loss of generality, assume there is a collision between C_2 and C_4 in the figure above. This means if we knew something about M_3 or M_5 we can derive information about the other. However this doesn't tell us anything about the plaintext as a whole, just these small segments.

Problem 4. Message digests are reasonably fast, but consider the following faster function: take your message, divide into 128-bit chunks, and $\oplus$ all the chunks to get to get a 128-bit result. Then compute your message digest on the result. Would this be a good message digest function? Explain.

Solution: Let $m = m_1||\ldots||m_n$ denote message m broken up into its 128 bit chunks. Given message m there are $n!$ unique messages (including itself) that can be constructed by rearranging $m_1, \ldots, m_n$, for example $m' = m_n||\ldots||m_1$ is one such message. Since $(\{0,1\}^n, \oplus)$ forms an Abelian group for any bit string length n, all $n!$ distinct message would digest to the same result. This creates high amounts of collisions for distinct messages, and therefore is not a good message digest.

Problem 5. Explain the vulnerability of hash functions, based on the Merkle-Damgard construction, to an extension attack. What is the security problem if Alice and Bob use a keyed hash based on such a hash function.

Solution: Suppose Bob computes hash value h based on message m and Merkle-Damgard function f. Now assume adversary Eve knows the length of m. From this Eve can easily compute the padding value p and can now imagine a message $m' = m||p||x$ for some arbitrary bit string x that Eve chooses. Note that Eve does not know the content of m so she must imagine such a message. When f is used on m' the padding value p' is computed and appended to m' to create $m' = m||p||x||p'$. At some point in f, the end of the p string is reached because $m||p$ must have been a multiple of whatever block size f uses. Although Eve does not know the content of m she knows at this point the running state of f must be the hash value h Bob computed earlier. Therefore she no longer needs to imagine anything about m and she can start $f(m||p||x||p')$ at the beginning of x using h as the running state. Using this extension attack, one can forge valid ciphertexts. For example take SHA1 that takes as input $k||m$ for some key k and message m. An attacker can see the corresponding ciphertext for m and can compute a valid ciphertext c' that extends m. This becomes especially malicious if these types of hashes are used for MACs.

Problem 6. Give a brief description of the TrueCrypt cipher. Explain its purpose and complexity. What is its current status?

Solution: TrueCrypt is a cipher that is used for on the fly encryption used primarily for disk sector encryption.TrueCrypt was released in February 2004 but later became deprecated in May 2014. TrueCrypt supported three main ciphers: AES, Serpent, and Twofish. There were also five cascade algorithms available: AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES, and Twofish-Serpent. The mode of operation of TrueCrypt was XTS which is a mode based on XOR-encrypt-XOR (XEX). XEX uses the following scheme

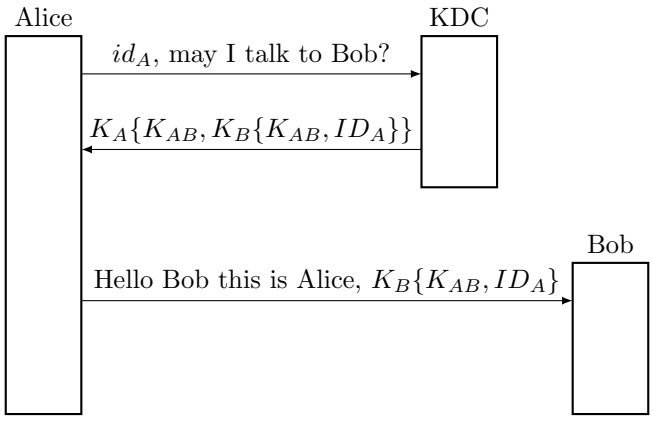
$$X = E_K(I) \otimes \alpha^j, \quad C = E_K(P \oplus X) \oplus X,$$

where P is the plaintext, I is the number of the sector, α is the primitive element of the Galois field GF(2^{128}), and j is the number of the block within the sector. Multiplication $\otimes$ and addition $\oplus$ are performed in the finite field GF(2^{128}). Note that α may be very hard to determine, so one can easily find an irreducible polynomial in GF(2^{128}). An example of this would be $x^{128} + x + 1$. Using such a polynomial gives an easy way to compute powers of α. XTS is just an extension of XEX where the sector being encrypted is not a factor of the block size.

The nature of XTS allowed for parallelizable encryption. Despite being somewhat fast in encryption, programs that required direct memory access would notice a slowdown because all data must first pass through the CPU to be decrypted rather than be directly copied from disk to RAM. Despite its seemingly secure construction, TrueCrypt was proven to be insecure. This was due to two main reasons: keys are stored in RAM and a key logger can easily capture unencrypted data.

Problem 7. Explain how the correct mutual authentication protocol for symmetric keys prevents reflection attacks whether the attacker Trudy pretends to be Bob or pretends to be Alice. Assume that Alice initiates the authentication. When Alice wants to talk to Bob through a KDC, she sends a message and gets a reply from the KDC. Is this hence insecure? Explain.

Solution: The mutual authentication protocol using a KBC is shown below.



Since Trudy does not have K_A or K_B the response from the KDC is of no use to her. Therefore she can not attempt a reflection attack because she can not correctly use the KDC's response. The insecurity of this scheme is that the KDC is a huge fail point. Any compromise of the KDC would result in the insecurity of this scheme. This is because the adversary would have access to the keys.

Problem 8. Lamports hash claims to protect against eavesdropping and server database disclosure without using public key cryptography. Explain briefly how Lamports hash works. How can it be improved? What are its limitations?

Solution: Lamport's hash scheme is the following

1. Alice types her name and password
2. "Alice" is sent to Bob
3. Bob sends n to back to Alice
4. Alice computes $h^{n-1}(password)$ and sends it to Bob
5. Bob computes $h(h^{n-1}(password)) = h^n(password)$ and checks it against the database. If it matches, the authentication is valid, Bob stores $h^{n-1}(password)$ in the database, and replaces n with $n-1$.

Although a passive attack can capture $h^{n-1}(password)$, the connection would require $h^{n-2}(password)$ for the next login. One problem with Lamport's hash, or rather typical users, is that passwords are common. A dictionary attack would prove to be somewhat effective with common passwords. To alleviate this problem, instead of hashing just the password we can hash $password||salt$ for some randomized $salt$. Given adversary Eve, we another problem given by the following

1. Alice types her name and password
2. "Alice" is sent to Bob
3. Bob sends n to Alice. Eve intercepts this transmission and replaces n with $n' = n - \delta$ for $1 \leq delta < n$
4. Alice computes $x = h^{n'-1}(password) = h^{n-\delta-1}(password)$ and sends it to Bob. Eve intercepts this transmission and replaces it with $h^{\delta}(x) = h^{n-1}(password)$
5. Bob computes $h(h^{n-1}(password)) = h^n(password)$ and confirms authentication.

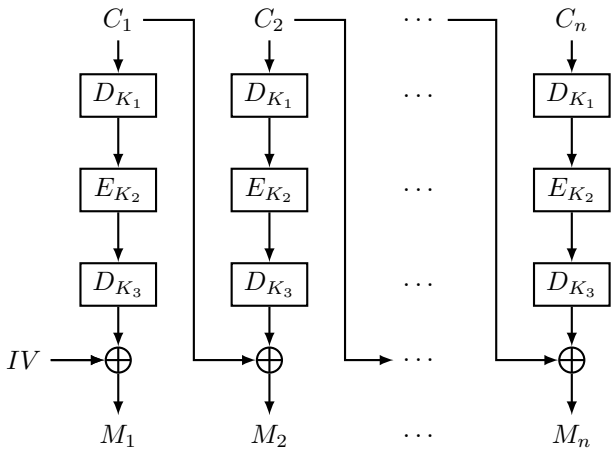
Eve now knows the next δ hashes Bob needs to receive for authentication. For example the next hash should be $h^{n-2}(password) = h^{\delta-1}(x)$.

Problem 9. The generally accepted method of making DES secure through multiple encryptions is known as 3DES (triple DES using EDE). Explain the differences between CBC on the inside versus CBC on the outside using 3DES and the pros and cons of using one or the other. Show the decryption diagram of CBC on the outside.

Solution: The encryption algorithms for CBC outside and CBC inside are the given below

Round	CBC Outside	CBC Inside
First Round	$C_0 = IV$ $C_1 = E_{K_3}(D_{K_2}(E_{K_1}(M_1 \oplus C_0)))$	$C_0 = IV$ $P_0 = E_{K_1}(M_1 \oplus C_0)$ $C_1 = E_{K_3}(D_{K_2}(E_{K_1}(M_1 \oplus C_0)))$
Round i $i = 1, \ldots, N$	$C_i = E_{K_3}(D_{K_2}(E_{K_1}(M_i \oplus C_{i-1})))$	$P_i = E_{K_1}(P_{i-1} \oplus M_i)$ $C_i = E_{K_3}(D_{K_2}(E_{K_1}(M_i \oplus P_{i-1})) \oplus C_{i-1})$

A bit flip in CBC outside cause the current block and next block of plaintext to be garbled while CBC inside cause more garbled plaintext blocks. Since CBC inside has intermediate results that affect the next ciphertext computation, we can start that computation while still finishing the current one. Therefore there is more pipelining for CBC inside. CBC inside is not parallelizable for decryption while CBC outside is. The decryption diagram is given below



Problem 10. Extend the scenario in section 9.7.4.1 Multiple KDC Domains to a chain of three KDC. In other words, assume that Alice wants to talk to Boris through a chain of three KDCs (Alices KDC, a KDC that has shared keys with both Alices KDC and Boriss KDC, and Boriss KDC). Give the sequence of events necessary to establish communication.

Solution: The diagram below depicts the sequence of events that need to happen given 3 KBCs between Alice and Boris.