

Introduction to Cryptography 89-656

Brief Solutions and Remarks

Final Exam, Moed Aleph 2008

Exam instructions:

1. Open book: all material is allowed
2. Answer all questions
3. Time: 2.5 hours
4. **Good luck!**

Question 1: Are the following symmetric encryption schemes perfectly secret, computationally secret or not secret at all, when used to encrypt a single bit? Answer for each scheme separately.

1. Scheme 1:

- *Key generation:* choose a random 128-bit key $k \leftarrow \{0, 1\}^{128}$
- *Encryption:* upon input a bit $b \in \{0, 1\}$ and a key k , choose a random 63-bit string $r \leftarrow \{0, 1\}^{63}$ and encrypt as $c = AES_k(b||r)$ where $||$ denotes concatenation.

2. Scheme 2:

- *Key generation:* Let k_1, k_2 be two distinct, arbitrary 128-bit strings. The key k is chosen at random between k_1 and k_2 .
- *Encryption:* upon input a bit $b \in \{0, 1\}$ and a key k , choose a random 63-bit string $r \leftarrow \{0, 1\}^{63}$ and encrypt as $c = AES_k(b||r||0^{64})$ where $||$ denotes concatenation.

3. Scheme 3:

- *Key generation:* Let k_1, k_2 be two distinct, arbitrary 128-bit strings. The key k is chosen at random between k_1 and k_2 .
- *Encryption:* upon input a bit $b \in \{0, 1\}$ and a key k , find a 128-bit string $r \in \{0, 1\}^{128}$ such that the least significant bit of $AES_{k_1}(r)$ equals 0 and the least significant bit of $AES_{k_2}(r)$ equals 1. Then, if $k = k_1$ set $c = (r, b)$ and if $k = k_2$ set $c = (r, 1 - b)$. The decryption of $c = (r, b)$ with key k is as follows: if the least significant bit of $AES_k(r)$ equals 0 then output b ; otherwise output $1 - b$.

Fully prove your answers; assume that AES is a secure pseudorandom permutation where necessary.

Solutions 1:

1. This scheme is computationally secure. It is not perfectly secure because this would imply that for every ciphertext c , AES has the property that exactly half of the keys would decrypt c to a string beginning with 0 and exactly half would decrypt c to a string beginning with 1. The proof of computational security is easy and is directly from the assumption that AES is a pseudorandom function.
2. This scheme is not secure at all. In the question as stated above (with 64 zeroes), it is easy to distinguish by just decrypting under both k_1 and k_2 and seeing which plaintext has 64 zeroes (with overwhelming probability only one will). However, even in the original question, with probability about $1/2$, decrypting c under both keys will give the same value and with probability about $1/2$ it will give different values. When it gives the same value, the attacker knows with full certainty what the encrypted value was. When it gives different values, the attacker can just guess (and will be right half the time, adding overall another $1/4$ to its advantage). Overall we get that the attacker has a $3/4$ advantage! The scheme is therefore not secure.
3. The scheme is perfectly secure. However, you cannot use Shannon because the ciphertext space is large (allowing many possible r 's). Rather you have to work directly from the definition.

Question 2:

1. Define a fixed-length hash function $h(m) = AES_m(0)$ for inputs of length 128 bits. Is this function preimage resistant (one-way) when applied to a random input $m \leftarrow \{0, 1\}^{128}$?
2. Define a fixed-length hash function $h(m) = AES_0(m)$ for inputs of length 128 bits. Is this function collision-resistant?

Justify your answers; assume that AES is a secure pseudorandom permutation where necessary.

Solutions 2:

1. This is actually not a very good question, because no-one says that given c it's hard to find *some* k such that $AES_k(0) = c$. Nevertheless, anyone who said that this is the case, and relied on the pseudorandomness of AES to state one-wayness received full points.
2. This function has no collisions whatsoever because AES is a permutation and thus is 1-1. This was disappointing: many students missed this and got it wrong.

Question 3: Are the following schemes secure message authentication codes or not?

1. Let (Gen, H) be a collision-resistant hash function with output-length n . The MAC-key is a key s for the hash function and a random $k \leftarrow \{0, 1\}^n$ (you can assume that s is known to the adversary, but of course, not k). A MAC-tag is computed as $t = \text{MAC}_{k,s}(m) = H^s(m) \oplus k$.
2. Let p be a prime and let g be a generator of \mathbb{Z}_p^* . The MAC-key is a random value $x \leftarrow \mathbb{Z}_p$. A MAC-tag is computed as $t = \text{MAC}_k(m) = (g^x \cdot m \bmod p, p)$. This is a fixed-length MAC scheme and it is assumed that $m \in \mathbb{Z}_p^*$.

- Let p be a prime and let g be a generator of \mathbb{Z}_p^* . The MAC-key is a random value $x \leftarrow \mathbb{Z}_p$. A MAC-tag is computed as $t = \text{MAC}_k(m) = (m^x \bmod p, p)$. This is a fixed-length MAC scheme and it is assumed that $m \in \mathbb{Z}_p^*$.

Solutions 3:

- This is easy to break: given (m, t) compute $k = t \oplus H^s(m)$. Then compute any MAC you want.
- This is easy to break: given (m, t) compute $g' = g^x = t/m$ and then compute any MAC you wish as $(m', g' \cdot m')$.
- This is easy to break: given (m_1, t_1) and (m_2, t_2) the pair $(m_1 \cdot m_2, t_1 \cdot t_2)$ is a valid MAC. This is a multiplicative attack.

Most students did reasonably well on this question.

Question 4: Consider the following variant of El Gamal encryption. The public and private keys are as in the original scheme. Given public-key (\mathcal{G}, q, g, h) and a message $m \in \mathcal{G}$, choose a random $y \leftarrow \mathbb{Z}_q$ and compute $c = \langle g^y \cdot m, h^y \rangle$.

- Show that given x where $h = g^x$, it is possible to efficiently decrypt. Describe the decryption process in detail and prove that it works. (Beware: this is not as straightforward as in the original El Gamal.)
- Is this scheme secure? Formally prove your answer.

Solution 4:

- Given $c = (u, v)$, compute $u^x/v = g^{yx}m^x/h^y = m^x$. Let $z = x^{-1} \bmod q$ in the group. Then, the result is given by $(u^x/v)^z$. I expected explanation that such a z can be found and why this works. Many students said that they take the x th root with no further explanation.
- Assume by contradiction that \mathcal{A} can distinguish. Given input $\mathcal{G}, q, g, g_1, g_2, g_3$, algorithm D wishes to know if there exist a, b such that $g_1 = g^a, g_2 = g^b, g_3 = g^{ab}$. D sets the public key pk to be (\mathcal{G}, q, g, g_1) and works like in the proof of El Gamal. However, unlike in El Gamal, D encrypts m_b by $c_1 = g_2 \cdot m$ and $c_2 = g_3$. If the input is a DH-tuple, then this is correct encryption. However, if the input is not a DH-tuple, then b appears nowhere and so get perfect hiding. I.e., there exists a b' s.t. $g^b \cdot m = g^{b'}$ and this has the exact same distribution.

Remark: *In general, from the exams my feeling is that the exam was not too hard. Indeed a number of students succeeded and many students succeeded on different parts of the exam. There was no overly hard question that only few got correct. I believe that with more studying, higher grades would be obtained. Some students had no idea how to prove. However, quite a few showed good proficiency in this.*