CHALMERS TEKNISKA HÖGSKOLA
Datavetenskap
Daniel Hedin

DIT250/TDA351

Solutions to exam in Cryptography December 17, 2013

Hash functions

1. A cryptographic hash function is a deterministic function that takes a long message m and computes a fixed size hash value v. Typically $|m| >> |v|$ and $|v|$ is in the range $128 - 1024$ bits. In addition, a cryptographic hash function should be *one-way* and *collision resistant*.

   Being one-way entails that it should be easy to compute the hash value for a given message, while it should be computationally infeasible to find a message that has a given hash value.

   Being collision resistant entails that it should be computationally infeasible to find two messages m_1 and m_2 with the same hash value, $H(m_1) = H(m_2)$.

2. The birthday problem refers to the question of how many people, when selected uniformly at random, is needed before you can expect two to share birthday. In general, if one considers n possible outcomes instead of 365 possible birthdays the number of uniform samples needed can be safely (under) approximated by $\sqrt{n}$.

   For a cryptographic hash function this means that if the size of the hash value is k bits ($n = 2^k$ possible different hash values) we can only hash $\sqrt{2^k} = 2^{k/2}$ randomly selected messages before we expect one collision. This means that the birthday problem gives us an attack against collision resistance in $2^{k/2}$ steps.

Public key

1. For $N = p \cdot q$, where p and q are large primes select e such that $gcd(e, \Phi(N)) = 1$. The public key for RSA is the pair (N, e) and encryption is performed by

   $$RSA_{(N,e)}(m) = m^e \ mod \ N$$

   To show that RSA is multiplicatively homomorphic it suffices to show

   $$RSA_{(N,e)}(m_1) \cdot RSA_{(N,e)}(m_2) = m_1^e \ mod \ N \cdot m_2^e \ mod \ N =$$
   $$m_1^e \cdot m_2^e \ mod \ N = (m_1 \cdot m_2)^e \ mod \ N =$$
   $$RSA_{(N,e)}(m_1 \cdot m_2)$$

2. The secret key of RSA consists of (p, q, d) where d is the multiplicative inverse of e in $\mathbb{Z}^*_{\Phi(N)}$. Signing using RSA can be performed by decrypting the message to be signed using the secret key, i.e.,

$$\sigma = sign_{(p,q,d)}(m) = m^d$$

The corresponding verification is then performed by encrypting the signature with the public key. The result of this step is the message.

$$verify_{(N,e)}(\sigma, m) = \sigma^e \stackrel{?}{=} m$$

Anyone in possession of the public key can verify the signature, and the signature can only be created by someone in possession of the secret key.

3. Assume two message-signature pairs (m_1, σ_1) and (m_2, σ_2). A new pair can be constructed by pointwise multiplication of the pairs, i.e., $(m_1 \cdot m_2, \sigma_1 \cdot \sigma_2)$. The justification is identical to problem 1 above.

4. We can protect against this existential forgery attack by signing, e.g., the hash value of the message. Let H be a hash function

$$sign_{(p,q,d)}(m) = H(m)^d$$
$$verify_{(N,e)}(\sigma, m) = \sigma^e \stackrel{?}{=} H(m)$$

The verification function must the be changed accordingly to check that the signature 'encrypts' to the hash value of the message m. The attack no longer works since with overwhelming probability it holds that

$$H(m_1) \cdot H(m_2) \neq H(m_1 \cdot m_2)$$

and thus

$$sign_{(p,q,d)}(m_1) \cdot sign_{(p,q,d)}(m_2) = H(m_1)^d \ mod \ N \cdot H(m_2)^d \ mod \ N = H(m_1)^d \cdot H(m_2)^d \ mod \ N = (H(m_1) \cdot H(m_2))^d \ mod \ N \neq H(m_1 \cdot m_2)^d \ mod \ N = sign_{(p,q,d)}(H(m_1 \cdot m_2))$$

Block ciphers

1. **diffusion** The idea of diffusion is to spread statistical properties of the plaintext over the ciphertext to decouple the statistics of the ciphertext and the statistics of plaintext, which makes it harder to recover the plaintext from the cipher text using statistical attacks like n-gram frequencies.

   confusion The idea with confusion is to decouple the ciphertext and the key making it harder to recover information about the key from the ciphertext.

2. The design of f is at the core of the security of a Feistel network. One important property we expect from f is that it should be non-linear, since the rest of the Feistel network is entirely linear. If f is linear, this opens up for powerful attacks like linear cryptanalysis.

We do not have to demand that f is invertible, since the result of f xored with the left part of the Feistel network state, L_i. To decrypt this xor has to be inverted which is done using the output of f, since $x \oplus y \oplus y = x$, i.e.

$$L_i = R_{i+1} \oplus f(R_i, K_i)$$

3. The meet-in-the-middle attack for 2DES is a known plaintext attack that relies on the possibility to decouple the keys in the following way

$$c = E_{k_1}(E_{k_2}(m)) \Longleftrightarrow D_{k_1}(c) = E_{k_2}(m)$$

In the right formulation there is only one key used on each side of the equality, which opens up for tabulation, given known plaintext-ciphertext pairs (m,c). The attacker proceeds by tabulating $D_k(c)$ for all possible k, storing the results $(D_k(c),k)$ in such a way that lookup based on $D_k(c)$ is $O(1)$. For DES this requires $O(2^{56})$ computation steps and storage. Thereafter the attacker starts computing $E_k(m)$ until a k' is found such that $E_{k'}(m) = D_k(c)$. The corresponding k can be found in the table built in the previous step. The key pair (k,k') is a candidate key pair (it does not have to be unique). The set of candidate key pairs can then be reduced by considering more plaintext-ciphertext pairs.

For DES, finding all candidate keys for a plaintext-ciphertext pair requires $O(2^{56} + 2^{56}) = O(2^{57})$ operations, and it suffices to repeat the process with with two plaintext-ciphertext pairs yielding an attack that runs in $O(2^{58})$ steps.

Protocols - the ISO 9798-2 protocol

1. The ISO 9798-2 protocol achieves mutual authentication based on the shared knowledge of key K_{AB}. In the first step of the protocol A sends a message to B that can be interpreted as

$$A \rightarrow B : \text{"Hi, I'm } A\text{, please authenticate using nonce } N_A$$

In the second messge B responds with

$$B \rightarrow A : \text{"Hi, } A\text{, in response to the authentication request with nonce } N_A \text{ I respond authenticating using } K_{AB}\text{. In addition, please authenticate using nonce } N_B.$$

There are three fundamental details that ensure the authentication of B to A in the first two steps.

- The inclusion of the nonce N_A in the reply prevents *replay attacks*, and
- the inclusion of the name A in the reply prevents *refection attacks*. Together they guarantee to A that the received response is a response to his authentication request and not a replay or reflection.
- This together with the fact that the response was encrypted with the key, K_{AB}, that A and B share, allows A to conclude that B created the response.

Finally, A replies

$$A \rightarrow B : \text{"Hi, in response to the authentication request with nonce } N_B \text{, tied to the authentication request with nonce } N_A \text{ I authenticate using } K_{AB}.$$

At this point B can conclude that A created the reply and mutual authentication is achieved.

2. One possibility is to use cipher block chaining, CBC, with random IV. The encryption of the second message would then be

$$2.B \rightarrow A : C_0, C_1, C_2, C_3 = IV, E_{K_{AB}}(A \oplus C_0), E_{K_{AB}}(N_A \oplus C_1), E_{K_{AB}}(N_B \oplus C_2)$$

This works since the nonces in the third message are swapped, which prevents the third message from being constructed from the second.

$$3.A \rightarrow B : C_0, C_1, C_2 = IV, E_{K_{AB}}(N_B \oplus C_0), E_{K_{AB}}(N_A \oplus C_1)$$

Another possibility is to use counter mode, CTR, with random nonce N.

$$2.B \rightarrow A : C_0, C_1, C_2, C_3 = N, A \oplus E_{K_{AB}}(N \oplus 0), N_A \oplus E_{K_{AB}}(N \oplus 1), N_B \oplus E_{K_{AB}}(N \oplus 2)$$

Again, the third message cannot be constructed from the second.

$$3.A \rightarrow B : C_0, C_1, C_2 = N, N_B \oplus E_{K_{AB}}(N \oplus 0), N_A \oplus E_{K_{AB}}(N \oplus 1)$$

It is not possible to use ECB, since the third message

$$3.A \rightarrow B : C_1, C_2 = E_{K_{AB}}(N_B), E_{K_{AB}}(N_A)$$

is easily constructed from the second message

$$2.A \rightarrow B : C_1, C_2, C_3 = E_{K_{AB}}(A), E_{K_{AB}}(N_A), E_{K_{AB}}(N_B)$$

3. Assuming we selected CBC in the previous question the answer is no. If the nonces are not swapped it is easy to construct the third message from the second message by dropping the C_0 (the IV). This will cause C_1 to be interpreted as the new IV and decryption can proceed without problems since for

$$2.B \rightarrow A : C_0, C_1, C_2, C_3 = IV, E_{K_{AB}}(A \oplus C_0), E_{K_{AB}}(N_A \oplus C_1), E_{K_{AB}}(N_B \oplus C_2)$$

C_1, C_2, C_3 is $\{N_A, N_B\}_{K_{AB}}$ using CBC with C_1 as IV, and we have that decryption of C_1, C_2, C_3 yields N_A, N_B.

Assuming we selected CTR mode in the previous question the answer is yes. From

$$2.B \rightarrow A : C_0, C_1, C_2, C_3 = N, A \oplus E_{K_{AB}}(N \oplus 0), N_A \oplus E_{K_{AB}}(N \oplus 1), N_B \oplus E_{K_{AB}}(N \oplus 2)$$

it is still not possible to construct

$$3.A \rightarrow B : C_0, C_1, C_2 = N, N_A \oplus E_{K_{AB}}(N \oplus 0), N_B \oplus E_{K_{AB}}(N \oplus 1)$$

since in the second message N_A, and N_B were encrypted using a counter value of 1 and 2, and in the third message they are encrypted with 0 and 1 respectively.

Protocols - the Schnorr protocol

1. If both parties are honest we have that

$$g^z = g^{c \cdot x + r} = g^{c \cdot x} \cdot g^r = g^{xc} \cdot g^r = X^c \cdot R = R \cdot X^c$$

2. We have that $z_1 - z_2 = c_1 \cdot x + r - (c_2 \cdot x + r) = c_1 \cdot x - c_2 \cdot x = (c_1 - c_2) \cdot x$. We also have c_1 and c_2, and thus, we can compute x as follows

$$x = (z_1 - z_2)/(c_1 - c_2)$$

All computations are done modulo q, for q prime, which guarantees the existense of multiplicative inverses.

Random numbers

1. We have argued that block ciphers, MACs and cryptographic hash functions should be computationally indistinguishable from their ideal variants. The ideal block cipher is a family of random permutations indexed by key, the ideal MAC is a family of random functions indexed by keys, and the ideal cryptographic hash function is a random function. Thus, the output of any of those should be essentially random.

2. Counter mode is performed as follows

keys		$E_K(N \oplus 0)$	$E_K(N \oplus 1)$	$E_K(N \oplus 2)$	$E_K(N \oplus 3)$	...
		$\oplus$	$\oplus$	$\oplus$	$\oplus$	...
plaintexts		M_0	M_1	M_2	M_3	...
ciphertexts	N	C_0	C_1	C_2	C_3	...

Thus, this is essentially a stream cipher using $E_K(N \oplus 0), E_K(N \oplus 1), E_K(N \oplus 2), E_K(N \oplus 3), ...$ as keystream. This sequence is a good way to generate random runmbers. Since a block cipher is a permutation (for a given key) the stream will repat when the input to the block cipher repeats, which occurs when the counter wraps. Using the schema above where the counter is xored with a random nonce, this occurs after 2^k encryptions, where k is the block size. In general it occurs after n steps where n is the number of values the counter can assume before repeating.

Security notions

1. If the adversary had any way of extracting a non-negligible amount of information about the plaintext from the ciphertext, the adversary would have a way to win this game with non-negligible probability. Thus, if the adversary cannot with this game with non-negligible probability then he cannot learn more than non-negligible information about the plaintext from the ciphertext.

2. The first adversary generates two messages uniformly at random

   $$A_1(pk) = m_0 \leftarrow \{0,1\}^n;\ m_1 \leftarrow \{0,1\}^n;\ \text{return } (m_0, m_1)$$

   The important thing is that they are not equal. Given a large message space $\{0,1\}^n$ this is unlikely. In principle he could test for this and regenerate if needed.

   In the second phase the adversary makes use of the fact that the encryption function is deterministic and thus preserves equality, i.e., $m_1 = m_2 \iff E_{pk}(c_1) = E_{pk}(c_2)$

   $$A_2(c) = c_0 \leftarrow E_{pk}(m_0);\ \text{return } (c_0 == c)$$

3. Since the encryption function preserves equality and there are only two possible choices — either $c == E_{pk}(m_0)$ or $c == E_{pk}(m_1)$ — the above adversary wins with probability 1.