CHALMERS TEKNISKA HÖGSKOLA
Datavetenskap
Björn von Sydow DIT250/TDA351

Solutions to exam in Cryptography

Thursday, December 20, 2012 14.00 – 18.00

1. Both take as argument an arbitrarily long message and produce a short (typically 160 or 256 bits) result. The MAC takes additionally a secret key as argument. Both are used to ensure integrity, i.e. that a message has not been tampered with. A MAC additionally provides authentication, i.e. evidence about who the sender is. Hash functions have many additional uses in cryptography.

2. (a) See course book or lecture slides.

   (b) In CTR mode, unblinding is easily achieved. After encryption, block C_i for $i > 0$ is $M_i \oplus N \oplus E_K(IV||i)$. Just xoring this once more with N eliminates the N and gives the ordinary CTR encryption of M_i. So, unblinding is the same as blinding, except that the 0-th block (the IV) should not be touched.

   (c) This new blinding scheme is not a good idea. Indeed, for any $i > 1$ Charlie can compute $M'_i \oplus M'_{i-1}$, which is M_i. So, Charlie can immediately get all plaintext blocks except the first, which is efficiently blinded.

3. The practice of changing a proposed document to sign in some insignificant way is recommended in order to prevent a form of birthday attack.

   The attack consists of the adversary preparing two entirely different contracts, and then producing many similar versions of each, in order to produce a collision in the hash values of two documents, one of each kind. The birthday "paradox" implies that the adversary will have to prepare $O(2^{n/2})$ versions of each in order to find a collision.

   By modifying the proposed contract, Alice prevents this attack, forcing the adversary to attack the one-wayness of the hash function, with brute-force complexity $O(2^n)$.

4. (a) This is the famous one time pad cipher. It has perfect secrecy, meaning that it cannot be broken in a ciphertext only attack, even with unlimited computational resources. However, it also needs enormous amounts of key material, so it is only feasible in some very limited, extreme-security situations.

   (b) Stream ciphers based on a single hardware LFSR are extremely cheap and fast, and can be built also into cheap devices. On the other hand, they offer very little protection. In a known plaintext attack, they can be broken, also if the tap sequence is unknown, with only $2L$ bits of output, where L is the length of the LFSR.

5. The essential observation is to verify that Bob will compute the same value of K as Alice used:

$$(X_A \cdot g^r)^{s \cdot x_B} = (g^{x_A + r})^{s \cdot x_B} = g^{((x_A + r) \cdot s) \cdot x_B} = g^{z \cdot x_B} = X_B^z,$$

where the third equality makes use of the fact that $(x_A + r) \cdot s = z \in \mathbb{Z}_q$, which follows from the definition of s in the signing operation.

It then follows that Bob will retrieve m when decrypting and that his hash computation will be the same as Alice's and thus satisfy the required equality.

6. (a) He concatenates his own nonce, the received nonce and B:s identity and computes the MAC of this. He then takes the xor of this with the second part of the received message and this is the session key.

   (b) A believes that the key is fresh, since its computation involves her own freshly chosen nonce. B knows that it is fresh, since he chose it himself before sending message 2.

   (c) If the adversary gets hold of the session key for a session where he has recorded the protocol run, then he can xor the key and the second part of the recorded message 2. This will give him $\mathrm{MAC}_{K_{AB}}(N_A||B)$ for the nonce that A chose. He can then, whenever he pleases, start a new run, pretending to be A and using the same nonce N_A. If B does not check the reuse of the nonce (which is commonly omitted), he will proceed with the protocol and the adversary can use his saved MAC value to compute the session key.

   (d) One reason is that it is a matter of trust to allow someone to choose a key which you will use. Thus, a trusted third party (TTP), who by definition is trusted, is a better choice than an arbitrary user you need to communicate securely with. Another reason is that with a TTP, it is enough for each party to share a long term key with the TTP, and not with every other user. In an n party community, this reduces the number of keys from $O(n^2)$ to $O(n)$.

   (e) See book or lecture slides.

7. (a) The adversary constructs $c_1 = 2^e \cdot c \in \mathbb{Z}_N^*$. Textbook decryption then gives $m'_p = c_1^d = 2^{ed} m_p = 2m_p$, since $ed = 1 \in \mathbb{Z}^*_{\Phi(N)}$.

   (b) Now the first eight bits are not zero anymore, so $2m_p \geq B$. We already knew that $m_p < B$. Summarizing, the adversary knows that $B/2 \leq m_p < B$.

   (c) Similarly as in (a), he constructs $c_2 = (k+1)^e \cdot c \in \mathbb{Z}_N^*$.

   (d) We do some computations in $\mathbb{Z}$, i.e no mod operations involved, noting that $\frac{N}{2} \leq k \cdot B < N$:

$$(k+1) \cdot m_p < (k+1) \cdot B = k \cdot B + B < N + B$$
$$(k+1) \cdot m_p \geq (k+1) \cdot \frac{B}{2} \geq \frac{N}{2} + \frac{B}{2}$$

   Thus $\frac{N}{2} + \frac{B}{2} \leq (k+1) \cdot m_p < N + B$. The lower bound is bigger than B, so if decryption does not fail in step 2, it must be because actually $(k+1) \cdot m_p \geq N$ and a mod operation is done in the RSA decryption. Thus the adversary can conclude that

$$\frac{N}{k+1} \leq m_p < \frac{N+B}{k+1}.$$

   This is a tightening of the lower bound, since $\frac{N}{k+1} > \frac{N}{N/B+1} > \frac{B}{2}$.

   (e) Padding should contain some redundancy, i.e. include a fixed bitstring (e.g. a sequence of zero bits) at a fixed position in the padded message.