CHALMERS | GÖTEBORGS UNIVERSITET

CRYPTOGRAPHY

TDA352 (Chalmers) - DIT250 (GU)

EXAM DATE, TIME

No extra material is allowed during the exam except for an approved calculator. *No other electronic devices allowed.* Your answers in the exam must be written in *English.* Your language skills will not be graded (but of course we cannot grade your answer if we do not understand it), so try to give *clear answers.* Your thoughts and ways of reasoning must be clearly understood!

Examiner: Elena Pagnin
Questions during the exam:
Inspection of exam: See web page for announcement.

The exam has 3 *parts* (following the course module structure) each part gives up to 20 points, for a total of maximum 60 points.
Grades are :

CTH Grades:	30-40 → 3,	41-50 → 4,	51 or above → 5
GU Grades:	30-50 → G,	51 or above → VG	

Good luck!

Symmetric Key Cryptography

Problem 1 — Block Ciphers

1.1 Provide the definition of a block cipher. **(3p)**

A Block Cipher is a deterministic, keyed function that is invertible.(1p) Formally, $\mathbf{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, where $\mathcal{K} = \{0,1\}^K$ is a key space made of binary strings of length $K \in \mathbb{N}$, $\mathcal{M} = \{0,1\}^n$ is the block space. (1p). For every key $k \in \mathcal{K}$, the function $\mathbf{E}(k,\cdot) : \{0,1\}^n \to \{0,1\}^n$ is efficiently computable and admits an efficiently computable inverse function $\mathbf{D}(k,\cdot) : \{0,1\}^n \to \{0,1\}^n$. Correctness ensures that $\mathbf{D}(k,\mathbf{E}(k,m)) = m$ for every block $m \in \{0,1\}^n$.(1p)

1.2 Describe the design principles of a secure block cipher. **(4p)**

There are four main design principles: 1- Iteration (0.5p), which consists of repeatedly applying a not-so-strong block cipher, each time with a different key, creating a sequence of rounds (0.5p). This technique is not provably secure, but heuristics show that it works. 2- Use of Feistel Networks (0.5p), which consists of applying a cipher function F at every round following the Feistel pattern defined by: $\begin{cases} L_{i+1} = R_i \\ R_{i+1} = L_i \oplus F(K_i, R_i) \end{cases}$ (0.5p) 3- Confusion (0.5p) is implemented in AES by the S-boxes. Inuitively this design principle works by dividing the input into sub-blocks, and applying a substitution table within each sub-block. (0.5p) 4- Diffusion (0.5p) is implemented in AES by the P-boxes. Intuitively it works by applying a permutation that scrambles the input bits in such a way that the bits of each input sub-block are distributed across all output sub-blocks. (0.5p)

1.3 Let $\mathbf{E}$ denote the encryption function of a semantically secure block cipher. Show that $\mathbf{E}'(k,m) = \mathbf{lbs}(m)||\mathbf{E}(k,m)$ where $\mathbf{lsb}(m)$ denotes the least significant bit of m, is not semantically secure. **(3p)**

In order to show that $\mathbf{E}'$ is not semantically secure it suffices to give two distinct messages $m_0, m_1 \in \{0,1\}^n$ that can be used to win the semantic security game with non-negligible probability. (1p) In this case, since $\mathbf{E}'$ leaks the least significant bit of the plaintext, we can choose an m_0 with $\mathbf{lbs}(m_0) = 0$ and m_1 with $\mathbf{lbs}(m_1) = 1$. (1p) Let c denote the ciphertext returned by the semantic security challenger. By construction we observe that c contains $\mathbf{lbs}(m_b)$ as its left-most bit, i.e., $c[0] = \mathbf{lbs}(m_b)$, where b denotes the challenge bit. So by returning as a guess $b' = c[0]$ we are guarantee to win the semantic security game with probability 1. (1p)

Problem 2 — One Way Functions & Integrity

2.1 For each of the following proposals, state if they are one-way functions or not. Also provide a motivation for your decision.

(a) $f(x,y) = x + y$, where $x, y \in \{0,1\}^{128}$ and x, y are interpreted as natural numbers. **(2p)**

(b) $g(x) := x||f(x)$ where $f(\cdot)$ is a one-way function. **(2p)**

(c) $g(x_0||x_1) := x_0||f(x_1)$ where $f(\cdot)$ is a one-way function. **(2p)**

(a) is not one way (1p). The reason is that given a number z in the range of $f(\cdot)$ it is easy to find x, y such that $x = f(x,y)$. An efficient algorithm to do so is to sample a random $x \in \{0,1\}^{128}$ and compute $y = z - x$ (1p), if $y \notin \{0,1\}^{128}$ then sample another x. (b) is not one way (1p). The reason is that it leaks its input. (1p) (c) is one way (1p). The reason is that even if it leaks half of its input, to find a pre-image of the second half of the output would imply the existence of an efficient algorithm that given $y = f(x_1)$ returns x' (possibly equal to x_1) such that $f(x') = y$, which contradicts the assumption that $f(\cdot)$ is one way. (1p)

2.2 Let $f : \{0,1\}^{128} \to \{0,1\}^{128}$ be a one way function. Consider the following proposal of a message authentication code (MAC) over the keyspace $\{0,1\}^{128}$ and message space $\{0,1\}^{128}$ defined as $MAC(k,m)$:

$t_0 \leftarrow \$\{0,1\}^n$
$t_1 = f(k \oplus t_0 \oplus m)$
`return` $t = (t_0, t_1)$

(a) Describe the verification algorithm for this MAC construction. **(2p)**

(b) Show that the proposed MAC is insecure by devising an attack that breaks unforgeability (it suffices to do one authentication query to the MAC oracle). **(2p)**

(a) The verification algorithm works as follows $Ver(k,m,t)$:

`parse` $t = (t_0, t_1)$
$t'_1 = f(k \oplus t_0 \oplus m)$(1p)
`if` $t'_1 == t_1$ `return` 1 `else return` 0.(1p)

(b) Note that the tag reveals the randomness t_1 which is used as a one time pad on the message. The attack works as follows. $\mathcal{A}$ queries the MAC oracle on a message $m \in \{0,1\}^{128}$ and receives the tag $t = (t_0, t_1)$. Now t_1 is a valid tag for any input $x = key \oplus (t_0 \oplus m)$. We will use the malleability of the one time pad to ensure the same t_1 for a different message (and randomness). This is done by sampling a random $t_0^* \xleftarrow{R} \{0,1\}^{128}$ and setting $m^* = t_0^* \oplus t_0 \oplus m$. (1p) The pair $(m^*, (t_0^*, t_1))$ is a valid existential forgery against MAC. (1p) Indeed, $Ver(k, m^*, t^*) = 1$ since $t_1 = f(k \oplus t_0 \oplus m) = f(k \oplus t_0^* \oplus m^*)$.

Public Key Cryptography

Problem 3 — Key Exchange

3.1 In the Diffie-Hellman key exchange it is important that the two parties involved in the protocol agree on a common base value g. Consider you are working modulo the prime number $p = 251$. You are asked to choose whether to use $g = 3$ or $g = 6$ to perform a secure key exchange. Is one choice better than the other? Why? **(2p)**

In the DH key exchange it is important that the base g used by both parties is a generator of the group $\mathbb{Z}_p^*$. It is easy to check that 3 is not a generator of $\mathbb{Z}_p^*$, indeed $3^{125} \mod 251 = 1$ so 3 generates only a subgroup, but not the whole group. (0.5p) In contrast $6^{\frac{p-1}{2}} \mod p = -1$, which makes 6 a generator of $\mathbb{Z}_p^*$ (0.5p) (since 2 is not invertible modulo $250 = \Phi(p)$). So, yes, $g = 6$ is a better choice because it provides better security as it generates the whole group instead of a smaller subgroup. (1p)

3.2 State the two security assumptions needed for the DH key exchange to be secure. **(4p)**

The two security assumptions are related to the Discrete Logarithm Problem (1p) and the Computational Diffie-Hellman Problem (1p) and essentially state that these problems are believed to be hard. In detail, the Discrete Logarithm (DL) assumption states that given an element $h \in \mathbb{G} = \wedge g\rangle$ to find a value x such that $X = g^x$ is computationally infeasible. (1p) The Computational Diffie-Hellman (CDH) assumption states that given the description of a group $\mathbb{G} = \wedge g\rangle$ and two elements $A = g^a$, $B = g^b$ finding $K = g^{ab}$ is computationally infeasible. (1p)

3.3 Prove that one of the computational problem behind the assumptions for the previous question reduces to the other **(4p)**

The CDH problem reduces to the DL problem (1p), that is $CDH \leq DL$. For the proof, it suffices to show that given access to an efficient DL solver, call this $\mathcal{A}_{DL}$ we can efficiently solve a CDH challenge. (1p) In detail, given the CDH challenge (g, A, B) the reduction can send A to $\mathcal{A}_{DL}$. (1p) Since $\mathcal{A}_{DL}$ is able to efficiently compute the discrete logarithm of a given element, $\mathcal{A}_{DL}$ will return to the reduction $a = dLog(A)$. Note that now the reduction has the same knowledge as Alice has, in the textbook DH key exchange, and thus can compute K as $K = B^a$ (1p), and return this K as its solution to the CDH challenger.

Problem 4 — Encryption

4.1 Describe the IND-CCA game, including the winning condition. **(5p)**

The indistinguishability under chosen ciphertext attack game (0.5p) for a public key encryption scheme ($\mathbf{KeyGen}$, $\mathbf{Enc}, \mathbf{Dec}$) over the message space $\mathcal{M}$ is played between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ that interact as follows. The challenger runs the key generation algorithm to obtain a key pair $(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathbf{KeyGen}(sec.par)$ and returns $\mathbf{pk}$ to $\mathcal{A}$. (1p) Next, the adversary is allowed a first query phase, where $\mathcal{A}$ can interact with the decryption oracle. Next, is the challenge phase, where $\mathcal{A}$ submits two messages $m_0, m_1 \in \mathcal{M}$ (of the same length) chosen by the adversary. (1p) The challenger samples a random bit $b \leftarrow \$\{0, 1\}$, and returns to $\mathcal{A}$ the ciphertext $c \leftarrow \mathbf{Enc}(\mathbf{pk}, m_b)$. (1p) Next, $\mathcal{A}$ has another query phase with the decryption oracle where $\mathcal{A}$ is allowed to receive decryption of any ciphertext except for the challenge ciphertext c (created in the previous phase by $\mathcal{C}$). (0.5p) At the end of the game, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ for the bit b embedded in the challenge. The adversary wins the game if $b = b'$. (1p)

4.2 Give the definition of an IND-CCA secure encryption scheme. **(1p)**

A public key encryption scheme ($\mathbf{KeyGen}$, $\mathbf{Enc}$, $\mathbf{Dec}$) is said to be IND-CCA secure if the probability that a PPT adversary $\mathcal{A}$ playing the IND-CCA security game (described in the previous point) wins is negligibly close to 1/2:

$$Prob[b = b'] = \frac{1}{2} + negl(sec.par) \quad (1p)$$

4.3 Recall how ElGamal encryption works, then show that it is not IND-CCA secure. **(4p)**

The ElGamal encryption scheme works on groups of large prime order q. Let g denote a generator of the group. The secret key is $\mathbf{sk} \in \mathbb{Z}_q$ and the corresponding public key is $\mathbf{pk} = g^{\mathbf{sk}}$. To encryption a message m, one first samples a random value $r \leftarrow \$\mathbb{Z}_q$, the ciphertext is the defined as $c = (c_0, c_1)$ where $c_0 = g^r$ and $c_1 = m(\mathbf{pk})^r$. (1p) Decrytpion uses the knoweldge of the secret key to recover a message from a ciphertext $c = (c_0, c_1)$ using the following formula: $m = c_1 \cdot c_0^{-\mathbf{sk}}$. (1p) ElGamal encryption is not IND-CCA secure because it is malleable (homomorphic). A succesfull attack strategy would be to skip the first query phase, then submit $m_0 \neq m_1$ for any pair of messages in the group. Let$c = (c_0, c_1)$ denote the challenge ciphertext provided by $\mathcal{C}$. Construct the ciphertext $c' = (c_0, 2 \cdot c_1)$ (1p) and submit c' to the decryption oracle in the secon query phase. Since $c' \neq c$ the oracle will decrypt c' and return $m' = 2 \cdot m_b$ to $\mathcal{A}$. Now dividing m' by the modular inverse of 2, the adversary can extract m_b compare it against m_0, m_1 and return the correct guess $b' = b$ with probability 1. (1p)

Cryptographic Protocols

Problem 5 — Commitments

5.1 Prove that no commitment scheme can be simultaneously unconditionally secure hiding and unconditionally secure binding. **(2p)**

The reasoning proceeds by reduction to absurd. Suppose we have a scheme which is both unconditionally concealing and binding, and suppose the committing party makes a commitment $c \leftarrow \mathsf{Commit}(m, r)$ for some message m and randomness r. The information-theoretically hiding property implies that there must exist values $m^* (\neq m)$ and r^* such that $c = \mathsf{Commit}(m^*, r^*)$, otherwise an infinitely powerful receiver could break the concealing property by finding the unique pair (m, r) that generates the commitment c. (1p) So there must exist an m^* s.t. $\mathsf{Commit}(m^*, r^*) = c = \mathsf{Commit}(m, r)$. The last equality implies that the commitment is not binding. (1p) So we conclude that if the commitment scheme is information-theoretically hiding, it cannot be binding for a computationally unbounded sender, which contradicts the statement.

5.2 Consider the following proposal of a commitment function: $Commit(m, r) = (g^r, h^{m+r})$ where g, h are generators in a suitable prime order group $\mathbb{G}$.
(a) Write the *Open* algorithm for this commitment. **(1p)**

(b) Is this scheme unconditionally secure for binding or hiding? Motivate your answer. **(1p)**

(a) The *Open* algorithm simply reconstructs the commitment from the given m and r and checks the result against the previously received value c. In detail, $Open(m, r, c)$ computes $c' = (g^m, h^{m+r})$, returns 1 if $c' = c$, and 0 otherwise. (1p) (b) The scheme is information theoretically binding (0.5p) since r acts as a one time pad on m and g^r is uniquely determined by r. (0.5p) Note that it is not possible to find another r' and $m' \neq m$ that generate the same commitment $c = (c_0, c_0) = (g^r, h^{m+r})$ unless $r' = r \mod p$ which then implies $m' = m \mod p$ (where $p = |\mathbb{G}|$) by the uniqueness of the discrete logarithm. The scheme is computationally hiding since m and r are given in the exponents (so a computationally unbounded adversary could efficiently solve the discrete logarithm problem, recover r from c_0 and then m from c_1).

5.3 A Pedersen commitment scheme is defined by a set up algorithm that outputs the description of a group suitable for cryptographic $\mathbb{G}$ of prime order q with generators g, h. To commit to a message $m \in \mathbb{Z}_q$ one computes $c = g^m h^r \mod q$ where $r \leftarrow \$\mathbb{Z}_q$. There are a few important facts for Pedersen commitments:
(a) The elements g and h are both generators of the whole group. Why does this matter? **(1p)**

(b) The discrete logarithm of h in base g should be unknown. What goes wrong otherwise? (describe an attack) **(2p)**

(c) The commitment is malleable. Show how from a commitment c of a message m you can derive a commitment c^* to the message $m + 1$ without knowing m and r. **(1p)**

(d) The commitment is malleable. Show how from a commitment c of a message m you can derive a commitment c' to the message $2m$ without knowing m and r. **(1p)**

(e) Given the opening of the original commitment c, would you be able to provide an opening for the c' from point (d)? How? **(1p)**

(a) If g (resp. h) is not a generator of the whole group, it means that it generates a subgroup, thus instead of having $q = |\mathcal{G}|$ possible messages (resp. randomnesses) there are only $q' = |\langle g \rangle| < q$ (resp. $q' = |\langle h \rangle| < q$), which affects security as it reduces the difficulty of solving the discrete logarithm problem. (1p) (b) If $dLog_g(h) = x$ is known then the binding property no longer holds (1p). For instance given $c = g^m h^r = g^{m+xr}$ one could provide alternative openings of c for any chosen $a \in \mathbb{Z}_q$ computed as $m' = m + ax$ and $r' = r - a$ (1p), indeed $g^{m'} h^{r'} = g^{m+ax+x(r-a)} = c$. (c) Let $c = g^m h^r$ for some r, then we can obtain $c' = g \cdot c = g^{m+1} h^r$. (1p) (d) Let $c = g^m h^r$, we can obtain $c^* = (c)^2 = g^{2m} h^{2r}$. (1p) (e) Let m, r be the opening of a commitment c, then the opening for $c^* = c^2$ is $2m, 2r$. (1p)

Problem 6 Zero Knowledge & Threshold Cryptography

6.1 Consider the following variation of Schnorr's protocol (where $x \in \mathbb{Z}_p$ denotes the witness and $X = g^x \in \mathbb{G}$ is part of the public statement):

Prover (x)		Verifier (X)
$r \leftarrow \$\mathbb{Z}_p^*$		
$a = g^r$	$\xrightarrow{a}$	
	$\xleftarrow{e}$	$e \leftarrow \$\{0,1\}^t$
$z = r \cdot e + x$	$\xrightarrow{z}$	

(a) Write the verifier's equation for this protocol. **(1p)**

(b) Recall the definition of special honest-verifier zero-knowledge and show that this protocol satisfies it. **(3p)**

(c) The honest-verifier feature is crucial. Describe a strategy a dishonest verifier could use to break zero-knowledge if the challenge e is chosen in a malicious way. **(1p)**

(a) The verifier's equation is supposed to check that if z is constructed as expected, then it matches a value that is computed from the prover's public key (statement). In this case the equation is a small variation of the one in the Schnorr protocol: $g^z = a^e \cdot X$. (2p) (b) To show that the protocol is special honest-verifier zero-knowledge we need to show that for X and witness x (such that $X = g^x$), and for every challenge $e \in \{0,1\}^t$ it holds that $\{Sim(X,e)\} =_c \{\langle P(X,w), V(X,e)\rangle\}$. (1p) This means that we need to show a simulator that is able to produce transcripts that have the same distribution as honest transcripts. We follow the standard procedure to build a Simulator in this setting, that is: we begin by sampling a random value $\tilde{z}$ for the final answer, use the input challenge e to then derive the first message $\tilde{a}$ of the simulated transcript without ever using x (which indeed is not given to Sim), in this way we obtain:

$$\{(\tilde{a}, e, \tilde{z}) \mid \tilde{z} \leftarrow \$\mathbb{Z}_p, \tilde{a} = g^r X^e\} \quad (1p)$$

while a standard transcript is $\{(a, e, z) \mid r \leftarrow \$\mathbb{Z}_p, a = g^r, z = r \cdot e + x\}$. The distributions of the two ensembles are identical: each conversation occurs exactly with probability $\frac{1}{p}$, determined by $\tilde{z}$ and r respectively. (1p) (c) A malicious verifier could pick $e = 0$ and learn $x = z$. (1p) If the verifier was honest, this case would happen with (negligible) probability $2{-}t$.

6.2 In this problem you will construct a threshold version of the RSA signature scheme. Let $N = p \cdot q$ denote an RSA modulo, e the public key and d the secret signing key. Let m denote the message to be signed.

(a) Consider the additive secret sharing scheme where $d \in \mathbb{Z}_N$ is split into 2 shares $x_1 \leftarrow \$\mathbb{Z}_N$ and $x_2 = d - x_1 \mod N$. Describe an algorithm that the two parties, knowing x_1 and x_2 respectively, can use to compute shares of an RSA signature. Given m and x_i, your algorithm should compute a partial signature σ_i for $i \in \{1, 2\}$. **(1p)**

(b) Describe a way to combine σ_1 and σ_2 from (a) into a signature σ that can be used in the standard RSA signature verification procedure. **(1p)**

(c) Does this approach generalize to a generic t out of n secret sharing scheme? Motivate your answer **(3p)**

(a) The standard RSA signature is $\sigma = H(m)^d \mod N$. Each party knows a share of the secret key, and can thus compute a share of the signature as: $\sigma_i = H(m)^{x_i} \mod N$. (1p) Note that each signature share alone cannot be verified by the standard RSA signature because nothing guarantees that $x_1 \cdot e = 1 \mod \Phi(N)$.

(b) A valid RSA signature (that can be verified against the public key e) can be obtained simply by multiplying together the two partial signatures (exploiting the homomorphic property of RSA): $\sigma = \sigma_1 \cdot \sigma_2 = H(m)^{x_i} \cdot H(m)^{x_2} = H(m)^{x_1+x_2} = H(m)^d \mod N$. (1p) Clearly σ satisfies the verification equation since $\sigma^e = H(m)^{ed} = H(m) \mod N$. (c) Yes(1p), the approach generalizes to any secret sharing scheme defined over $\mathbb{Z}_N$ where the reconstruction function f is linear in the shares. (1p) The generic approach is to first run the key generation of RSA, obtain the secret singing key d. Then use d as input to the $Distribute(N, n, t, d)$ algorithm to obtain the n shares $\{x_i\}_{i=1}^n$ of the key. Distribute each share to each signing party. The partial signature generation (run by each party independently) works exactly as in point (a), that is: $\sigma_i = H(m)^{x_i} \mod N$. To reconstruct a valid RSA signature for t signature shares σ_i with $i \in I \subseteq \{1, 2, \ldots, n\}, |I| = t$, one would need to run $Reconstruct'(I, \{\sigma_i\}_{i \in I})$ and get σ, where $Reconstruct'$ works exactly as the $Reconstruct$ algorithm for the secret sharing scheme, except that addition of shares are replaced with multiplication of partial signatures, and mutliplications (by known coefficients) are replaced with exponentiations. This is to ensure that $Reconstruct$ happens correctly 'in the exponent'. (1p)