CHALMERS | GÖTEBORGS UNIVERSITET

CRYPTOGRAPHY

TDA352 (Chalmers) - DIT250 (GU)

Jan 13th 2023, 8:30-12:30

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: Elena Pagnin (phone 0723518384)
Inspection of exam: See announcement on Canvas.

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 — Semantic Security & Perfect Secrecy

1.1 Let $(\mathbf{KeyGen}, \mathbf{E}, \mathbf{D})$ be a semantically secure cipher, where the messages, ciphertexts and keys are binary strings, e.g., you can think $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$, with $n \geq 2$. For each of the following proposals, state if they are semantically secure or not. Also provide a motivation for your decision.

(a) $\mathbf{E}'(k, m) = \mathbf{E}(k, m) \oplus \mathbf{1}$, where $\mathbf{1}$ denotes the string with all ones. **(2p)**

(b) $\mathbf{E}'(k, m) = \mathbf{E}(k, m)||RB(m)$, where $RB(m)$ gives back a random bit of the input m. **(2p)**

(c) $\mathbf{E}'(k, m) = \mathbf{E}(k, m)||RB(k)$, where $RB(k)$ gives back a random bit of the input k. **(2p)**

(a) Yes, (1p) the encryption scheme is semantically secure. The ciphertexts of $\mathbf{E}'$ are simply the ciphertexts of $\mathbf{E}$ flipped. Thus, it is easy to see that any attack against $\mathbf{E}'$ can be turn into an attack against $\mathbf{E}$ (which is semantically secure by assumption). (1p) (b) No, (1p) the encryption scheme is not semantically secure. The adversary could use the following strategy to win the semantic security game. Choose $m_0 = \mathbf{0}$ (the all-0 message) and $m_1 = \mathbf{1}$ (the all-1 message). Return $RB(m)$ as b', the guess for the b chosen by the adversary to produce the challenge ciphertext. (1p) (c) Yes, (1p) the encryption scheme is semantically secure. Although $\mathbf{E}'$ is leaking one bit of the key, this does not impact semantic security given that the key is longer than 1 bit ($n \geq 2$). Indeed a randomly chosen bit from a randomly chosen string (the key), will be completely independent from any pair of messages submitted by the adversary and thus gives no noticeable advantage (better than random guessing) in determining whether the challenge ciphertext returned is an encryption of m_0 or m_1. (1p)

1.2 Prove that any symmetric encryption scheme that is perfectly secure must have the key space at least as large as the message space. In other words, perfect security implies $|\mathcal{K}| \geq |\mathcal{M}|$. (Hint, this is Shannon's theorem and holds for the One Time Pad.) **(4p)**

In order to prove the statement, assume $\mathbf{E}$ is perfectly secure. (1p) This means that for every $c \in \mathcal{C}$ we have that $\mathsf{Prob}[\mathbf{E}(k, m_0) = c | k \leftarrow \$\mathcal{K}] = \mathsf{Prob}[\mathbf{E}(k, m_1) = c | k \leftarrow \$\mathcal{K}] > 0$. (1p) Since $\mathbf{E}(k, \cdot) : \mathcal{M} \to \mathcal{C}$ is an injective function (otherwise the inverse $\mathbf{D}(k, \cdot)$ is not well-defined), different messages must be encrypted using different keys to obtain the same ciphertext c. Thus for each and every $m \in \mathcal{M}$ there exists a key $k_m \in \mathcal{K}$ such that $\mathbf{E}(k_m, m) = c$. (1p) This implies that there must be at least as many keys as there are messages, i.e., $|\mathcal{K}| \geq |\mathcal{M}|$. (1p) Note that it is not a problem if $\mathbf{E}(k, m) = c = \mathbf{E}(k', m)$ for $k \neq k'$, as this does not contradict correctness. Correctness only implies that for a fixed key, each ciphertext has a unique decryption, i.e., $\mathbf{E}(k, m) = c = \mathbf{E}(k, m')$ for $m \neq m'$ cannot happen.

Problem 2 — Data Integrity

2.1 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 $t_0^* \leftarrow \$\{0,1\}^{128}$ 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 $t_1 = f(k \oplus t_0 \oplus m) = f(k \oplus t_0^* \oplus m^*)$.

2.2 The security notion for message authentication codes is unforgeability under chosen message attack. Consider the security game defined below, where Q_S, Q_V are positive integers polynomial in the security parameter, and $\mathcal{K}$ (resp. $\mathcal{M}$) is the set of keys (resp. messages).

Challenger $\mathcal{C}$		Attacker $\mathcal{A}$
$k \leftarrow \$\mathcal{K}$		$\mathcal{Q} = \varnothing$
(MAC queries)	$\xleftarrow{m_i}$	for i in $\{1, \ldots, Q_S\}$ choose $m_i \in \mathcal{M}$
$\mathbf{MAC}(k, m_i) = t_i$	$\xrightarrow{t_i}$	store $\mathcal{Q} = \mathcal{Q} \cup \{(m_i, t_i)\}$
(verification queries)	$\xleftarrow{(m_j, t_j)}$	for j in $\{1, \ldots, Q_V\}$ choose $(m_j, t_j) \notin \mathcal{Q}$
$\mathbf{Ver}(k, m_j, t_j) = d_j$	$\xrightarrow{d_j}$	

We say that $\mathcal{A}$ wins the above game if at least one verification query $(m_j^*, t_j^*) \notin \mathcal{Q}$ is answered with $d_j = 1$.

(a) Referring to the above game, give the definition of a secure MAC. **(2p)**

(b) Show that verification queries do not help. (Hint: Note that the game without verification queries can be obtained setting $Q_V = 1$, so the single message-tag pair (m^*, t^*) submitted is the output forgery attempt, rather than an adaptive verification query. Show that for every $\mathcal{A}$ that wins the unforgeability game with verification queries with probability $\mathsf{Prob}[\mathcal{A}\ wins]$, you can construct a new adversary $\mathcal{B}$ that wins the unforgeability game without verification queries ($Q_V = 1$) with probability $\mathsf{Prob}[\mathcal{B}\ wins] = \mathsf{Prob}[\mathcal{A}\ wins]/Q_V$.) **(4p)**

(a) A message authentication code is said to be secure if for any efficient (probabilistic, polynomial time) adversary $\mathcal{A}$ (1p) the probability that $\mathcal{A}$ wins the above game is negligible (1p) in the security parameter. (b) The reduction $\mathcal{B}$ plays the role of the challenger to $\mathcal{A}$ (which entails simulating answers to verification queries) and, at the same time, it plays the role of adversary in the game with $Q_V = 1$. The reduction relies all communication regarding MAC queries: any m_i submitted by $\mathcal{A}$ is sent to $\mathcal{B}$'s MAC oracle, and all responses are forwarded to $\mathcal{A}$. (1p) Upon receiving a verification query (m_j, t_j), $\mathcal{B}$ stores the pair in a list and returns $d_j = 0$ to $\mathcal{A}$. (1p) At the end of the game, $\mathcal{B}$ randomly selects one of $\mathcal{A}$'s verification queries and submits it as its output forgery attempt (m^*, t^*). The selected message-tag pair is a valid forgery with probability $\mathsf{Prob}[\mathcal{A}\ wins]/Q_V$ (1p), because in the game with verification queries $\mathcal{A}$ submits all its forgery attempts as verification queries. (1p) Alternatively, $\mathcal{B}$ can make its guess $\hat{i}$ (for the index of the verification query to forward as its own forgery attempt) at the beginning of the game, before any MAC query is preformed. In this case the game terminates with $\mathcal{B}$ forwarding $\mathcal{A}$'s $\hat{i}$-th verification query.

Public Key Cryptography

Problem 3 — Encryption

3.1 Consider the RSA encryption scheme.
(a) Describe in detail the three algorithms that define the cryptosystem. **(4p)**

(b) Define correctness for a public key encryption scheme and show that RSA is correct. **(2p)**

(c) Show that if an adversary $\mathcal{A}$ finds out $\Phi(N)$, then $\mathcal{A}$ can easily factorize N. **(2p)**

(a) The RSA encryption scheme is defined by three PPT algorithms ($\mathbf{KeyGen}, \mathbf{Enc}, \mathbf{Dec}$). In detail, **KeyGen** takes in input the security parameter that determines the desired bit length of the prime numbers involved. It picks two distinct primes p, q. It computes the RSA modulo $N = p \cdot q$, (0.5p) and e, d such that $e \cdot d = 1 mod \Phi(N)$, (0.5p) where $\Phi(N)$ denotes Euler's totient function. The secret key is $\mathbf{sk} = (N, d)$ (0.5p) while the public key is $\mathbf{pk} = (N, e)$. (0.5p) The encryption algorithm is $\mathbf{Enc}(\mathbf{pk}, m)$, and takes in input the public key and a message $m \in \mathbb{Z}_N$ and returns the ciphertext $c = m^e \mod N$. (1p) The decryption algorithm is $\mathbf{Dec}(\mathbf{sk}, c)$, and takes in input the secret key and a ciphertext $c \in \mathbb{Z}_N$, and returns the plaintext $m = c^d \mod N$. (1p) (b) To show correctness, we need to argue that for every message and key pair it holds that $\mathbf{Dec}(\mathbf{sk}, \mathbf{Enc}(\mathbf{pk}, m)) = m$, i.e., the decryption returns the initial plaintext. (1p) More formally, for all key pairs $(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathbf{KeyGen}(n)$ and for all messages $m \in \mathbb{Z}_N$ it holds that $\mathsf{Prob}[\mathbf{Dec}(\mathbf{sk}, c) = m | c = \mathbf{Enc}(\mathbf{pk}, m)] = 1$. RSA is correct since for every N, p, q, e, d constructed as described in point (a) it holds that $\mathbf{Enc}(\mathbf{pk}, m) = c = m^e \mod N$ and $\mathbf{Dec}(\mathbf{sk}, c) = c^d \mod N = (m^e)^d \mod N = m^{e \cdot d \mod \Phi(N)} \mod N = m \mod N$. (1p) (c) Recall that: $\Phi(N) = (p-1)(q-1) = pq - p - q + 1 = N - p - q + 1$. (1p) So knowing $\Phi(N)$ it is computationally easy to learn also $p + q = N + 1 - \Phi(N)$. But if you know both $p \cdot q = N$ and $p + q = a$ it is easy to compute p and q: using $q = N/p$, you know $p + N/p = a$, which gives the ordinary second-degree equation $p^2 - ap + N = 0$, which is efficiently solvable in p. (1p)

3.2 Alice and Bob use the RSA cryptosystem with the same modulus, but different encryption exponents e_A and e_B such that $gcd(e_A, e_B) = 1$. An eavesdropping adversary gets hold of one ciphertexts for Alice c_A and one ciphertext for Bob c_B. In addition, $\mathcal{A}$ knows that both ciphertexts encrypt the same message m. Show how $\mathcal{A}$ can retrieve m from c_A, c_B, e_A, e_B, N. **(2p)**

The general idea here is that the adversary should use the extended Euclid's algorithm (1p) on e_A and e_B to get (s, t) such that $se_a + te_b = 1$ since e_A and e_B are relative prime by assumption. Next, the adversary computes $c_A^s \cdot c_B^t$ which in $\mathbb{Z}_N$ equals to $c_A^s \cdot c_B^t = m^{se_A + te_b} = m \mod N$ (1p)

Problem 4 — Digital Signatures

4.1 Provide a brief definition of blind signatures. **(3p)**

A blind signature scheme is a signature scheme ($\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Ver}$) where the signing algorithm algorithms **Sign** is replaced by an interactive protocol run between a signer/issuer (S) and a receiver (R). (1p) The protocol starts with R who has as input a message m, and S who has as input a signing secret key **sk**. (1p) At the end of the interaction R obtains a signature σ on m, (1p) and S learns nothing about m or σ.

4.2 In Chaum's untraceable eCash system, to implement secure eCoin withdrawal Alice sends a message $B = r^e H(x) \mod N$ to the Bank. This is the beginning of a blind signature procedure based on the RSA-FDH (full domain hash) signature scheme.
(a) Explain what e, r, H, x are. **(2p)**

(b) Let $\bar{S} = B^d \mod N$ be the Bank's reply to Alice's message. First, show how Alice can unblind $\bar{S}$ and obtain the signature $S \in \mathbb{Z}_N$. Then show that S verifies. **(2p)**

(a) Since we are using RSA-FDH, e is the RSA public exponent (0.5p); $r \leftarrow \$\mathbb{Z}_N^*$ is some randomness (0.5p)

used to mask (blind) what should be signed by the Bank; H is a full-domain hash function (0.5p); and x is the coin (string or message) (0.5p) that should be blindly signed by the Bank. (b) In order to obtain a valid signature from the blinded value $\bar{S}$ Alice simply needs to remove the masking factor r as follows: $S = r^{-1} \cdot \bar{S} \mod N$. (1p) In order to show that S verifies, we need to show that it satisfies the RSA-FDH verification equation: $S^e \mod N =^? H(x)$, which clearly holds since $(S)^e = (r^{-1} \cdot \bar{S})^e = (r^{-1} \cdot (B^d))^e (r^{-1} \cdot (r^e H(x))^d)^e = (r^{-1} \cdot r^{e \cdot d} H(x)^d)^e = H(x)$ (1p), where all equations hold modulo N, and $e \cdot d = 1 \mod \Phi(N)$ because we are working in an RSA modulo.

4.3 Consider the following variant of the Elgamal signature scheme. The **KeyGen** algorithm outputs the description of a multiplicative group $\mathbb{G}$ of order p generated by g (for an opportune large prime p), the description of a full domain hash function $H : \{0,1\}^* \to \mathbb{G}$, a secret signing key $\mathbf{sk} \leftarrow \$\mathbb{Z}_p \setminus \{0\}$ and the corresponding public verification key $\mathbf{pk} = h = g^{\mathbf{sk}}$. The **Sign** algorithm takes in input the secret key **sk** and a message m, and it returns the signature $(s_1, s_2) \in \mathbb{Z}_p^* \times \mathbb{Z}_p$ such that $h^{s_1} g^{H(m)} = g^{s_2}$. This latter equation is used for verification, we do not specify how the signing procedure works.
(a) Show how to extract the secret signing key from any given a message-signature pair, i.e., given m and (s_1, s_2) that satisfy the verification equation $h^{s_1} g^{H(m)} = g^{s_2}$, give a procedure to compute **sk**. **(2p)**

(b) Is this scheme unforgeable? Briefly explain why. **(1p)**

(a) Given m and a valid signature (s_1, s_2) it is possible to compute $H(m)$ and verify that $h^{s_1} g^{H(m)} = g^{s_2}$ holds. Substituting h with $g^{\mathbf{sk}}$ (where **sk** is the sought after unknown value) the equation reads: $g^{\mathbf{sk} \cdot s_1 + H(m)} = g^{s_2}$. (1p) Taking the equality in the exponents (the discrete logarithm is unique $\mod p$) reveals the linear equation: $\mathbf{sk} \cdot s_1 + H(m) = s_2 \mod p$ from which it is possible to extract **sk** as $\mathbf{sk} = (s_2 - H(m))(s_1)^{-1} \mod p$ (1p) computing the modular inverse of s_1. So this is not a secure signature scheme. (b) No, (0.5p) the signature scheme in not unforgeable since every signature reveals the secret signing key as shown in the previous point. (0.5p) A successful attack strategy is to query for the signature of any message m, extract the signing key as shown above, select new message $m^* \neq m$ and use the recovered **sk** to run the **Sign** algorithm as the honest signer would to obtain a valid signature (forgery).

Cryptographic Protocols

Problem 5 Secret Sharing & Threshold Cryptography

5.1 Give the definition of a secret sharing scheme. **(3p)**

An (n, t)-secret sharing scheme is a method by which a dealer distributes shares of a secret value s to a set of $n \geq 2$ parties in such a way that simultaneously achieves the following three properties: 1) t-correctness: any set of t parties can jointly reconstruct s. (1p) 2) privacy: no single party alone learns anything about s. (1p) 3) $(t-1)$-security: any subset of less than t parties cannot compute s. (1p)

5.2 Explain why $t-1$ parties cannot learn anything about the secret in the (n, t) Shamir secret sharing scheme. Does this property rely on any computational assumption? **(2p)**

Shamir secret sharing scheme relies on the fact that the polynomial chosen by the dealer has uniformly random coefficients (a part from the constant term). In detail, given any set of $(t-1)$ shares $\{(x_i, y_i)\}_{i=1}^{t}$, there exists a degree $t-1$ polynomial f such that $f(x_i) = y_i$ for all i and $f(0) = s$ for all possible values of s (1p) in $\mathbb{Z}_q$. This is not a computational hardness assumption: Shamir secret sharing scheme is unconditionally secure. (1p)

5.3 You are given a group $\mathbb{G}$ of order a large prime number p, a generator g of $\mathbb{G}$ and key pairs are of the form $(\mathbf{sk}, \mathbf{pk}) = (x, g^x)$ for $x \in \mathbb{Z}_p$. Alice has decided to use Shamir secret sharing scheme to distribute her secret key $\mathbf{sk}_A \in \mathbb{Z}_p$ across $n \geq 3$ devices in such a way that any subset of $t = 2$ devices can reconstruct $\mathbf{sk}_A$, but no subset of less than t devices is able to do so. Let a_i denote the share held by device i. Bob has initiated a Diffie-Hellman Key Exchange with Alice by sending his public key $B \in \mathbb{G}$. Alice needs to respond, but she is in a situation where her available devices can only communicate with Bob but not with one another. So she responds to Bob with two messages, $A_2 = g^{a_2}$ and $A_3 = g^{a_3}$, and an explanation of her situation.

(a) Show how Bob can efficiently compute the Diffie-Hellman shared key $K = g^{\mathbf{sk}_A \cdot \mathbf{sk}_B}$. (Hint: no concrete computations are needed, but you should describe the procedure) **(3p)**

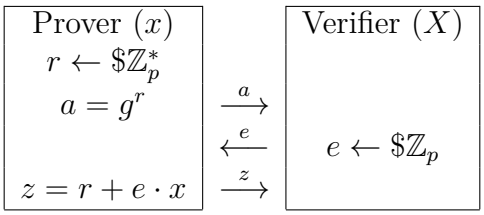
(b) Assume $n = 5$, and Alice knows that exactly one of her devices is corrupted in the sense that it is using an incorrect share value. Can she identify the malicious device? Why? **(2p)**

(c) What could Alice use, in addition to Shamir's scheme, to efficiently identify when wrong shares are used? **(1p)**

(a) Given A_1, A_2, the fact that Alice is using Shamir secret sharing scheme with threshold $t = 2$ and the identities of the devices are 2 and 3, Bob can do the following. First, he can compute the Lagrange coefficients $\delta_2^{2,3}$ and $\delta_3^{2,3}$. (1p) Next, he can reconstruct Alice's public key by doing interpolation in the exponent: $A = A_2^{\delta_2^{2,3}} \cdot A_3^{\delta_3^{2,3}} = g^{\delta_2^{2,3} \cdot a_2 + \delta_3^{2,3} \cdot a_3} = g^{\mathbf{sk}_A}$. (1p) Finally, Bob can calculate the standard DH key as $K = A^{\mathbf{sk}_B}$. (1p) (b) Yes, (1p) Alice can identify the faulty device. One strategy is to ask all possible subsets of $t = 2$ devices to reconstruct the secret. There are $\binom{n}{t} = \binom{5}{2} = 10$. Since there is exactly one malicious party, there are $\binom{n-1}{t} = \binom{4}{2} = 6$ coalitions that reconstruct the same (and the correct) value $\mathbf{sk}_A$. On the other hand, the are $\binom{n-1}{t-1} = \binom{4}{1} = 4$ coalitions that reconstruct a different value $\mathbf{sk}_A^*$. Alice can identify the faulty device as the only one party that takes part in all of the 4 incorrect reconstructions. (1p) (c) Alice could use a verifiable secret sharing scheme (1p), for instance combining Shamir's secret sharing with Pedersen commitments to each share and the secret key.

Problem 6 — Sigma Protocols

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



(a) Give the definition of special honest-verifier zero-knowledge. **(2p)**

(b) Imagine you get a hold of two transcripts (a, e, z) and (a', e', z') with $a' = a^2$. Describe a method you can use to extract the witness x. Do you need to make any assumptions? **(3p)**

(a) Special honest-verifier zero-knowledge states that for any given statement X and witness x satisfying the relation R (in this case: $X = g^x$), and for every challenge e there exists an efficient simulator Sim (1p) that outputs transcripts that are indistinguishable from honest ones. (1p) Formally, $\{Sim(X, e)\} =_c \{\langle P(X, w), V(X, e)\rangle\}$. (b) This is a small adaptation of the standard extraction technique. By construction we know that there exists and $r \in \mathbb{Z}_p^*$ such that $(a = g^r, e, z = r + e \cdot x)$ and $(a' = (a)^2 = g^{2r}, e', z' = 2r + e' \cdot x)$ where r and x are unknowns. Taking the system of equations identified by $z, z' \in \mathbb{Z}_p$ we get:

$$\begin{cases} z = r + e \cdot x \mod p \\ z' = 2r + e' \cdot x \mod p \end{cases} \quad (1p) \quad \rightarrow \quad \begin{cases} r = z - e \cdot x \mod p \\ z - z' = r - 2r + (e - e') \cdot x \mod p \end{cases}$$

Substituting r from the first equation into the second one we get: $z - z' = -(z - e \cdot x) + (e - e') \cdot x \mod p$ collecting x to the left hand side yields $x = (2z - z') \cdot (2e - e')^{-1} \mod p$ (1p). To perform this last step we need to assume that $(2e - e')$ is invertible modulo p. (1p)

6.2 In Sigma protocols, why removing interaction between prover and verifier makes the proof verifiable by multiple verifiers? What technique is used to achieve this in a secure way (give the name, how and why it works)? **(4p)**

For the security of Sigma protocol it is fundamental that the prover cannot predict the challenge e that comes from the verifier. Otherwise, if e is known (before the initial message a is set), the special honest verifier zero knowledge property guarantees that a simulator can produce an accepting transcript, i.e., it is able to produce a proof that looks like an honest one but was created without the knowledge of the secret witness. (1p) This prevents interactive proofs from being convincing (valid proofs) for any party other than the verifier who took part in the interaction. A valid proof obtained without interaction thus can be used to convince any verifier, as there is no specific verifier that interacts with the prover. The way this is achieved is via the so-called Fiat-Shamir Transform (or Heuristic) (1p), which works by computing the challenge e as the hash of the initial message a, hence $e = H(a)$. (1p) Note that while e is well-determined by a, the pre-image resistant property of the hash function H (1p) prevents a malicious prover from being able to efficiently predict an a that generates an opportune e that allows cheating.