Zero-Knowledge Proofs

Saravanan Vijayakumaran

Department of Electrical Engineering
Indian Institute of Technology Bombay

December 18, 2023

Interactive Proofs

- Cryptographic protocols that enable a **prover** to prove the validity of a statement to a **verifier**
- Traditional proofs
  - *No interaction:* Prover writes down a sequence of statements each of which is an axiom or follows from axioms
  - False statements are impossible to prove
- Interactive proofs
  - Prover and verifier exchange messages
  - They can toss unbiased coins and keep the outcomes *secret*
  - An invalid proof can pass verification with a small probability
- Examples of statements
  - Two graphs G_1, G_2 are not isomorphic
  - For a composite integer N, $x \in \mathcal{QNR}_N$

    There exists no integer y such that $x = y^2 \bmod N$
- **Zero-Knowledge Proofs:** Interactive proofs that allow a prover to prove the validity of a statement **without** revealing anything else

Knowledge vs Information

- In information theory, entropy is used to quantify information
- Entropy of a discrete random variable X defined over an alphabet $\mathcal{X}$ is
$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$
- Knowledge is related to computational difficulty, whereas information is not
  - Suppose Alice and Bob know Alice's public key
  - Alice sends her private key to Bob
  - Bob has not gained new information (in the information-theoretic sense)
  - But Bob now knows a quantity he could not have calculated by himself
- Knowledge is related to publicly known objects, whereas information relates to private objects
  - Suppose Alice tosses a fair coin and sends the outcome to Bob
  - Bob gains one bit of information (in the information-theoretic sense)
  - We say Bob has not gained any knowledge as he could have tossed a coin himself

Modeling the Prover and Verifier

- PPT = Probabilistic Polynomial Time
  - Algorithms with running time that is polynomial in input size
  - They can toss coins and use the outcomes to take decisions
- Prover and verifier will be modeled as algorithms
  - Verifier is assumed to be PPT
  - Prover may or may not be PPT
- Prover is attempting to prove a statement
- Malicious or dishonest provers will try convincing the verifier that incorrect statements are true
- When the prover is forced to be PPT, we get an **argument** (not a proof)
  - The A in SNARK and STARK
  - All the ZK protocols deployed in the real-world are arguments
- When the prover is attempting a ZK proof or argument, the verifier is possibly malicious
  - The verifier attempts to extract something more than the statement's validity from the prover

ZK = Existence of a Simulator

- The prover is trying to prove a statement without leaking knowledge
- The set of messages exchanged by the prover and verifier is called a **transcript**
- An interactive proof is ZK if there is a PPT simulator who can **simulate** the transcript
  - Simulation = Generation of identically distributed transcript without knowledge of prover's secret
  - Distributions can also be negligibly different
- Does the existence of a simulator mean that proofs can be forged?
  - No, because simulation usually involves "forbidden" actions or information
  - Forbidden = Unavailable in a regular execution of the IP
  - Like reversing the arrow of time
  - Or using a simulation trapdoor

Modeling Statements

- A **language** is a subset of $\{0,1\}^*$
  - $\{0,1\}^*$ is the set of all finite-length bit strings
- A prover is interested proving membership of a public value in a language
- Examples of languages
  - Set of pairs of non-isomorphic graphs G_1, G_2
    - A pre-determined encoding will represent a graph as a bitstring
    - Two specific graphs G_1, G_2 will be specified as part of the statement
  - $\mathcal{QR}_N$ for a composite N
    - Set of quadratic residues modulo N
    - Each quadratic residue is an integer in the set $\{0, 1, \ldots, N-1\}$
    - Each integer can be represented using $\lceil \log_2 N \rceil$ bits

Interactive Proof Systems

- Let $\langle A, B\rangle(x)$ denote the output of B when interacting with A on common input x
- Output 1 is interpreted as "accept" and 0 is interpreted as "reject"
- **Definition:** A pair of interactive machines (P, V) is called an **interactive proof system for a language** L if V is PPT and the following conditions hold:
  - **Completeness:** For every $x \in L$, we have $\Pr[\langle P, V\rangle(x) = 1] \geq \frac{2}{3}$
  - **Soundness:** For every $x \notin L$ and every interactive machine B, we have $\Pr[\langle B, V\rangle(x) = 1] \leq \frac{1}{3}$
- Remarks
  - Soundness condition $\rightarrow$ any possible prover
  - Completeness condition $\rightarrow$ only prescribed prover
  - By repeating interaction and taking majority, probabilities can be made close to 1 and 0
  - The $\frac{2}{3}$ and $\frac{1}{3}$ are arbitrary choices by convention
  - Any $c(n), s(n)$ such that the **acceptance gap**

    $$c(|x|) - s(|x|) \geq \frac{1}{p(|x|)}$$

    for a polynomial p will do

Alternative Definition of IP Systems

- Let $c, s : \mathbb{N} \to \mathbb{R}$ be functions satisfying $c(n) > s(n) + \frac{1}{p(n)}$ for some polynomial $p(\cdot)$.
- **Definition**: A pair of interactive machines (P, V) is called an **interactive proof system for a language** L if V is PPT and the following conditions hold:
  - **Completeness**: For every $x \in L$, we have

$$\Pr\left[\langle P, V\rangle(x) = 1\right] \geq c(|x|)$$

  - **Soundness**: For every $x \notin L$ and every interactive machine B, we have

$$\Pr\left[\langle B, V\rangle(x) = 1\right] \leq s(|x|)$$

Interactive Proof Example

- Setting
  - Suppose Peggy claims that Pepsi in large bottles tastes different than Pepsi in small bottles
  - Victor challenges Peggy to prove her claim
- Protocol
  - Victor asks Peggy to leave the room
  - He selects either a large bottle or a small bottle randomly and pours some Pepsi into a glass
  - Peggy is called into the room and asked to tell which bottle the Pepsi came from by tasting it
  - Victor accepts if Peggy answers correctly
- Analysis
  - If the claim is correct, $\Pr[\langle P, V\rangle(x) = 1] = 1$
  - If the claim is wrong, $\Pr[\langle P, V\rangle(x) = 1] = \frac{1}{2}$ for any P
  - The acceptance gap is $1 - \frac{1}{2} = \frac{1}{2}$

Graph Isomorphism

- An undirected graph consists of a set of vertices V and edges represented by a subset E of $V \times V$
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a bijection $\pi : V_1 \mapsto V_2$ such that $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$

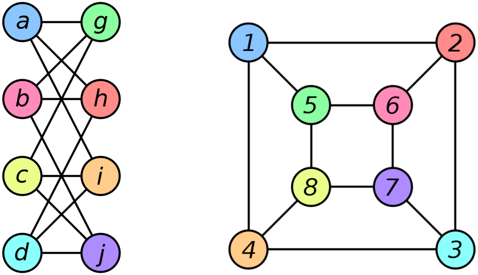

Image source: `https://en.wikipedia.org/wiki/Graph_isomorphism`

$$\pi(a) = 1, \pi(b) = 6, \pi(c) = 8, \pi(d) = 3,$$
$$\pi(g) = 5, \pi(h) = 2, \pi(i) = 4, \pi(j) = 7$$

Proving Graph Non-Isomorphism

- Proving that two graphs G_1, G_2 are isomorphic is easy if ZK is not required
    - Prover can send an isomorphism π
    - Verifier is polynomial-time
- How can we prove that two graphs G_1, G_2 are non-isomorphic?
    - Checking all bijections $\implies$ exponential-time verifier
- Assume that G_1 and G_2 have the same number of nodes and edges
    - Otherwise, non-isomorphism is trivial
- We need a subroutine that picks a graph randomly from the set of graphs isomorphic to a graph G
- Suppose $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$ and $E \subseteq V \times V$
    - Pick a random permutation π of V
    - Calculate the relabelled edge set $E' = \{(\pi(u), \pi(v) \mid (u, v) \in E)\}$
    - Output the graph $\pi(G) = (V, E')$
    - Note that the vertex set is unchanged by π

Proving Graph Non-Isomorphism

- Suppose $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
- Protocol
  - Verifier picks $\sigma \in \{1, 2\}$ randomly and a random permutation π from the set of all permutations over V
  - Verifier sends the graph $G' = \pi(G_\sigma)$ to prover
  - Prover finds $\tau \in \{1, 2\}$ such that G' is isomorphic to G_τ and sends τ to verifier
  - If $\tau = \sigma$, verifier accepts claim. Otherwise, it rejects.
- Remarks
  - Verifier is PPT but no known PPT implementation for prover
    - But even an exponential-time prover cannot cheat
  - If G_1 and G_2 are not isomorphic, then verifier always accepts
  - If G_1 and G_2 are isomorphic, then verifier rejects with probability $\frac{1}{2}$
- **Takeaway:** Interactive proofs enable a PPT verifier for graph non-isomorphism

Zero-Knowledge Interactive Proofs

- **Informal definition**: An interactive proof system is **zero-knowledge** if
  - whatever can be efficiently computed **after interaction** with P on input x
  - can also be efficiently computed from x (**without interaction**)
- Let $\text{view}^{P}_{V*}(x)$ denote the verifier's view of the protocol
  - It is the messages V^* receives and any randomness it generates
- Should be possible to generate something with the same distribution as $\text{view}^{P}_{V*}(x)$ without interacting with P

Perfect Zero-Knowledge (Ideal)

- **Formal definition (ideal)** : We say (P, V) is **perfect zero-knowledge** if
  - for every PPT interactive machine V^*
  - there exists a PPT algorithm M^* such that
  - for every $x \in L$
  - the random variables $\text{view}_{V^*}^{P}(x)$ and $M^*(x)$ are **identically distributed**
- M^* is called a **simulator** for the interaction of V^* with P
- Actually, P is zero-knowledge. The V is there to make it an interactive proof system
- Unfortunately, the above definition is too strict

Perfect Zero-Knowledge

- **Definition** : We say (P, V) is **perfect zero-knowledge** if
  - for every PPT interactive machine V^*
  - there exists a PPT algorithm M^* such that
  - for every $x \in L$ the following two conditions hold:

    1. With probability at most $\frac{1}{2}$, algorithm M^* outputs a special symbol $\perp$
    2. Let $m^*(x)$ be the random variable describing the distribution of $M^*(x)$ conditioned on $M^*(x) \neq \perp$. Then the random variables $\text{view}_{V^*}^{P}(x)$ and $m^*(x)$ are **identically distributed**

- What if the simulator fails?
  - The simulator fails with probability at most $\frac{1}{2}$
  - It can be run repeatedly until it generates the non-failure output
  - On the average it requires two runs

HVZK Proof of Graph Non-Isomorphism

- HVZK = Honest Verifier Zero-Knowledge
- Suppose $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
- Protocol
  - Verifier picks $\sigma \in \{1, 2\}$ randomly and a random permutation π from the set of all permutations over V
  - Verifier sends the graph $G' = \pi(G_\sigma)$ to prover
  - Prover finds $\tau \in \{1, 2\}$ such that G' is isomorphic to G_τ and sends τ to verifier
- Simulator
  - $\text{view}_V^P = (\sigma, \pi, G', \tau)$ where $G' = \pi(G_\sigma)$ and $\tau = \sigma$
  - A simulator M can pick σ and π randomly, set $\tau = \sigma$, and set $G' = \pi(G_\sigma)$
- Only HVZK
  - Protocol is ZK **only** when the verifier follows the protocol honestly
  - Suppose there is a third graph G_3 which the verifier wants to check for isomorphism with G_1 or G_2
  - The verifier can set $G' = G_3$ and use the prover's response to gain knowledge it could not have calculated by itself

ZK Proof of Graph Non-Isomorphism

- How to ensure a honest verifier?
  - The verifier needs to convince the prover that the graph $G' = \pi(G_\sigma)$ for $\sigma = 1$ or 2
  - The value of σ cannot be revealed to the prover
- Protocol
  - Verifier picks $\sigma \in \{1, 2\}$ randomly and a random permutation π of V
  - Verifier sends the graph $G' = \pi(G_\sigma)$ to prover
  - For $i = 1, 2, \ldots, s$, verifier generates a random bit β_i and two permutations π'_i, π''_i of V
    - If $\beta_i = 0$, verifier sends $\left(H_1^i, H_2^i\right) = \left(\pi'_i(G_1), \pi''_i(G_2)\right)$
    - If $\beta_i = 1$, verifier sends $\left(H_1^i, H_2^i\right) = \left(\pi'_i(G_2), \pi''_i(G_1)\right)$
  - Prover generates s random bits $b_1, b_2, \ldots, b_s$ and sends them to the verifier
    - If $b_i = 0$, verifier sends π'_i and π''_i
    - If $b_i = 1$, verifier sends an isomorphism from G' to one of $\left(H_1^i, H_2^i\right)$
    - In both cases, the prover checks that the appropriate isomorphisms were sent
    - If the checks fail for any i, the prover stops
  - The prover sends τ such that G_τ is isomorphic to G'. If no such τ exists, he sends a random value from $\{1, 2\}$
  - The verifier accepts if $\sigma = \tau$. Otherwise, she rejects

ZK Proof of Graph Isomorphism

- Setting
  - Two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are isomorphic
  - Prover wants to prove that they are isomorphic without revealing the isomorphism $\phi : G_1 \mapsto G_2$
- Protocol
  - Prover picks a random permutation π of V
  - Prover sends the graph $G' = \pi(G_2)$ to verifier
  - Verifier picks $\sigma \in \{1, 2\}$ randomly and sends it to prover
    - If $\sigma = 2$, then prover sends π to the verifier
    - If $\sigma = 1$, then prover sends $\pi \circ \phi$ to the verifier
  - If the received mapping is an isomorphism between G_σ and G', the verifier accepts. Otherwise, it rejects
- Analysis
  - Verifier is PPT
  - If ϕ is known, prover is PPT
  - If G_1 and G_2 are isomorphic, then verifier always accepts
  - If G_1 and G_2 are not isomorphic, then verifier accepts with probability $\frac{1}{2}$

Simulator for Graph Isomorphism Protocol

- For an arbitrary PPT verifier V^*, $\text{view}_{V^*}^{P}(x) = \langle G', \sigma, \psi \rangle$ where ψ is an isomorphism between G_σ and G'
- The simulator M^* uses V^* as a subroutine
- On input (G_1, G_2), simulator randomly picks $\tau \in \{1, 2\}$ and generates a random isomorphic copy G'' of G_τ
  - Note that G'' is identically distributed to G'
- Simulator gives G'' to V^* and receives $\sigma \in \{1, 2\}$ from it
  - V^* is asking for an isomorphism from G_σ to G''
- If $\sigma = \tau$, then the simulator can provide the isomorphism $\pi : G_\tau \mapsto G''$
- If $\sigma \neq \tau$, then the simulator outputs $\perp$
- If the simulator does not output $\perp$, then $\langle G'', \tau, \pi \rangle$ is identically distributed to $\langle G', \sigma, \psi \rangle$

GMR85

The First ZK Interactive Proof Protocols

- Published by Shafi Goldwasser, Silvio Micali, Charles Rackoff in ACM STOC 1985
- Involves quadratic residues and non-residues modulo a composite integer
- Preliminaries
    - For integers x, N, r, we write
    $$x \bmod N = r$$
    if $x = qN + r$ where q, r are integers with $0 \le r \le N - 1$
    - For an integer $N > 1$, we define
    $$\mathbb{Z}_N^* = \{x \mid 1 \le x \le N - 1 \text{ and } \gcd(x, N) = 1\}$$
    - $\mathbb{Z}_N^*$ forms a group under multiplication modulo N
        - It is closed under multiplication modulo N
        - Every element has a multiplicative inverse modulo N
    - An $x \in \mathbb{Z}_N^*$ is called a **quadratic residue** if there exists a $y \in \mathbb{Z}_N^*$ such that
    $$x = y^2 \bmod N$$
    - If no such y exists, x is called a **quadratic non-residue**

Properties of Quadratic Residues Modulo a Prime

- If $N = p$ where p is a prime, checking if an integer is a quadratic residue is easy
- Jacobi symbol modulo a prime
  - Let $\mathcal{QR}_p$ denote the set of quadratic residues modulo p
  - Let $\mathcal{QNR}_p$ denote the set of quadratic non-residues modulo p
  - For prime $p > 2$ and $x \in \mathbb{Z}_p^*$, the **Jacobi symbol of** x **modulo** p is given by

$$\mathcal{J}_p(x) = x^{\frac{p-1}{2}} \bmod p = \begin{cases} +1 & \text{if } x \in \mathcal{QR}_p, \\ -1 & \text{if } x \in \mathcal{QNR}_p. \end{cases}$$

- Exactly half the elements of $\mathbb{Z}_p^*$ are quadratic residues

Properties of Quadratic Residues Modulo a Composite

- If $N = pq$ for distinct odd primes p, q, then

$$x \in \mathcal{QR}_N \iff [x \bmod p] \in \mathcal{QR}_p \text{ and } [x \bmod q] \in \mathcal{QR}_q$$

- Corollaries
  - Exactly $\frac{1}{4}$ of the elements of $\mathbb{Z}_N^*$ are quadratic residues
  - If the factorization of N is known, then checking if $x \in \mathcal{QR}_N$ is easy
- If the factorization of N is unknown, then checking if $x \in \mathcal{QNR}_N$ is sometimes easy
  - For $x \in \mathbb{Z}_N^*$, we define

$$\mathcal{J}_N(x) = \mathcal{J}_p([x \bmod p]) \cdot \mathcal{J}_q([x \bmod q])$$

  - There is a polynomial time algorithm to calculate $\mathcal{J}_N(x)$ without using the factorization of N
  - If $\mathcal{J}_N(x) = -1$, we know $x \in \mathcal{QNR}_N$
  - If $\mathcal{J}_N(x) = +1$, then x could still be in $\mathcal{QNR}_N$ with $\mathcal{J}_p(x \bmod p) = \mathcal{J}_q(x \bmod q) = -1$
  - If $\mathcal{J}_N(x) = +1$, then there is **no known polynomial-time algorithm** for deciding the quadratic residuosity of x

ZK Proof for Quadratic Residuosity

- Setting
  - For $N = pq$, prover wants to prove $x \in \mathcal{QR}_N$
  - Prover knows $w \in \mathbb{Z}_N^*$ such that $x = w^2 \bmod N$
  - Verifier does not know factorization of N
  - Prover does not want to reveal w to the verifier
- Protocol
  - P picks $r \xleftarrow{\$} \mathbb{Z}_N^*$ and sends $y = r^2$ to V
  - V picks a bit $b \xleftarrow{\$} \{0, 1\}$ and sends b to P
  - If $b = 0$, P sends $z = r$. If $b = 1$, P sends $z = wr$
  - If $b = 0$, V checks $z^2 = y$. If $b = 1$, V checks $z^2 = xy$
- Simulator
  - For an arbitrary PPT verifier V^*, $\text{view}_{V^*}^P(x) = \langle y, b, z \rangle$ where $z^2 = x^b y$
  - Consider a simulator M^* which does the following
    - M^* picks $z \xleftarrow{\$} \mathbb{Z}_N^*$ and $b \xleftarrow{\$} \{0, 1\}$
    - M^* sets $y = \frac{z^2}{x^b}$
    - If $V^*(y) = b$, then M^* outputs $\langle y, b, z \rangle$. Otherwise, M^* outputs $\perp$

Interactive Proof for Quadratic Non-Residuosity

- Setting
  - For $N = pq$, prover wants to prove $x \in \mathcal{QNR}_N$
  - Assume $\mathcal{J}_N(x) = +1$
  - Verifier does not know factorization of N
  - How can P prove x is a quadratic non-residue without revealing the factorization of N?
- Protocol
  - V picks $y \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$ and a bit $b \stackrel{\$}{\leftarrow} \{0, 1\}$
  - If $b = 0$, V sends $z = y^2$. If $b = 1$, V sends $z = xy^2$
  - If $z \in \mathcal{QR}_N$, P sends $b' = 0$.
  - If $z \in \mathcal{QNR}_N$, P sends $b' = 1$
  - V accepts if $b' = b$
- If $x \in \mathcal{QNR}_N$, then the z sent by V is in $\mathcal{QNR}_N$ for $b = 1$
  - The prover knows the factorization of N and can decide the quadratic residuosity of z
  - So the prover can estimate b correctly
- If $x \in \mathcal{QR}_N$, then the z sent by V is in $\mathcal{QR}_N$ for both $b = 0$ and $b = 1$
  - The prover can estimate b correctly only with probability $\frac{1}{2}$

Protocol is Only HVZK

- The above protocol is **honest verifier zero-knowledge** but not ZK
  - Consider a PPT verifier V^* which wants to find out if some $u \in \mathbb{Z}_N^*$ is in $\mathcal{QR}_N$
  - By replacing x in the above protocol with u, verifier V^* can get information about u
  - If the protocol was ZK, then there exists a PPT M^* which can get the same information without interacting with P
  - This contradicts the non-existence of PPT algorithms for checking membership in $\mathcal{QR}_N$
- Getting to ZK
  - **Solution**: V has to prove that it either knows the square root of z or zx^{-1} to P
  - The number of interaction rounds increases from 2 to 4

ZK Proof for Quadratic Non-Residuosity

1. $V \to P$
   - P wants to prove that $x \in \mathcal{QNR}_N$ for $N = pq$
   - V picks $y \xleftarrow{\$} \mathbb{Z}_N^*$ and a bit $b \xleftarrow{\$} \{0,1\}$
   - If $b = 0$, V sends $z = y^2$. If $b = 1$, V sends $z = xy^2$
   - For $1 \leq j \leq m$,
     - V picks $r_{j,1}, r_{j,2} \xleftarrow{\$} \mathbb{Z}_N^*$ and $\text{bit}_j \xleftarrow{\$} \{0,1\}$
     - V computes $\alpha_j = r_{j,1}^2$ and $\beta_j = xr_{j,2}^2$.
     - If $\text{bit}_j = 0$, V sends $\text{pair}_j = (\alpha_j, \beta_j)$.
     - If $\text{bit}_j = 1$, V sends $\text{pair}_j = (\beta_j, \alpha_j)$.
2. $P \to V$
   - P sends V a bit string $[i_1, i_2, \ldots, i_m] \in \{0,1\}^m$
3. $V \to P$
   - V sends P the sequence $v_1, v_2, \ldots, v_m$
     - If $i_j = 0$, then $v_j = (r_{j,1}, r_{j,2})$.
     - If $i_j = 1$, then $v_j = yr_{j,1}$ if $b = 0$. So V sends a square root of $z\alpha_j$
     - If $i_j = 1$, then $v_j = xyr_{j,2}$ if $b = 1$. So V sends a square root of $z\beta_j$

ZK Proof for Quadratic Non-Residuosity

4. $P \rightarrow V$
   - P checks the following:
     - If $i_j = 0$, P checks if $(r_{j,1}^2, r_{j,2}^2 x)$ equals pair$_j$, possibly with elements in the pair interchanged.
     - If $i_j = 1$, P checks if $v_j^2 z^{-1}$ is a member of pair$_j$.
   - If $z \in \mathcal{QR}_N$, P sends $b' = 0$.
   - If $z \in \mathcal{QNR}_N$, P sends $b' = 1$
   - V accepts if $b' = b$

- How the protocol ensures a honest verifier?
  - Assume the verifier computes $\alpha_j = r_{j,1}^2$ and $\beta_j = x r_{j,2}^2$ correctly
    - Suppose a cheating verifier sends some z other than y^2 or xy^2
    - Then the verifier cannot calculate the square roots of $z\alpha_j$ or $z\beta_j$
  - Suppose the verifier cheats by setting $\alpha_j = z^{-1}u^2$
    - Then the verifier can calculate the square root of $z\alpha_j$ for arbitrary z
    - But with probability $\frac{1}{2}$ the verifier will need to calculate either $\sqrt{\alpha_j}$ or $\sqrt{\alpha_j x^{-1}}$
  - A cheating verifier can succeed only if it can predict the sequence of bits $i_1, i_2, \ldots .i_m$ sent by P perfectly
    - This occurs with probability $\frac{1}{2^m}$

References

- Sections 4.1, 4.2, 4.3, 4.4.2 of *Foundations of Cryptography, Volume I* by Oded Goldreich
- Alon Rosen's lecture in the 9th BIU Winter School on Cryptography `https://www.youtube.com/watch?v=6uGimDYZPMw`
- *The Knowledge Complexity of Interactive Proof Systems*, S. Goldwasser, S. Micali, C. Rackoff, 1989. `https://doi.org/10.1137/0218012`
- Ivan Damgard's lecture notes `https://users-cs.au.dk/~ivan/CPT1.pdf`