

Cryptographic Protocols Summary

Andreas Biri, D-ITET

29.06.17

1. Introduction

1.1 Mathematics

Group: $\langle G; * \rangle$ with operator $* : G \times G \rightarrow G$

associative: $x * (y * z) = (x * y) * z$

neutral element: $e : x * e = e * x = x \quad \forall x \in G$

inverse: $\hat{x} : x * \hat{x} = \hat{x} * x = e$

additive group: $* \triangleq +, e \triangleq 0, \hat{x} = -x$

multiplicative group: $* \triangleq \times, e \triangleq 1, \hat{x} = x^{-1}$

Order: element order divides group order

$|G|$: number of elements in the group

$$\begin{aligned} \text{ord}(x) : x^{\text{ord}(x)} &= x * \dots * x = e \\ x^{|G|} &= x^{k * \text{ord}(x)} = e^k = e \end{aligned}$$

$$\begin{aligned} \mathbb{Z}_m^* &= \{x \in \mathbb{Z} \mid 0 \leq x < m, \gcd(x, m) = 1\} \\ |\mathbb{Z}_p^*| &= p - 1, \quad p \text{ is a prime} \end{aligned}$$

Cyclic group: a generator g such that

$$G = \langle g \rangle = \{g^0, g^1, \dots, g^{p-1}\}$$

Isomorphism: $\langle G; * \rangle, \langle H; \bullet \rangle$ are isomorph if a bijection $\psi : G \rightarrow H$ exists for all $x, y \in G$:

$$\psi(x * y) = \psi(x) \bullet \psi(y)$$

Modulo calculation: $x, y \in \mathbb{Z}$ are congruent modulo m if

$$x \equiv y \pmod{m} \Leftrightarrow x \bmod m = y \bmod m$$

Inverse modulo m : $y : x * y \equiv 1 \pmod{m}$

Quadratic residue: $a : r^2 \equiv a \pmod{m}$

Functions: a function $f : \mathbb{N} \rightarrow \mathbb{R}^+$ is said to be

polynomial: $\exists c \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq n^c$

An algorithm is *efficient* if running time is polynomial

negligible: $\forall c \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq \frac{1}{n^c}$

noticeable: $\exists c \in \mathbb{N} : \forall n \geq n_0 : f(n) \geq \frac{1}{n^c}$

poly x negligible : negligible (cannot be amplified)

poly x noticeable : “large enough” (can be amplified)

1.2 Terminology & Languages

Proof of Statement: There exists a solution for ...

Proof of Knowledge: I know the solution for ...

(PoK is automatically a PoS, as it has an explicit solution)

If P can answer to all challenges, she can just as well compute the secret; therefore, it is a PoK as if she didn't know it before, she sure can know it now!

Languages & model of computation

Language L: contains all true statements / words

Decision problem: is some word member of a language L?

Witness: used for verification $\exists \omega : V(x, \omega) = 1, x \in L$

TM accepts L: $x \in L \Leftrightarrow TM(x) = 1, \text{ else something}$

TM decides L: $x \in L \Leftrightarrow TM(x) = 1$

$x \notin L \Leftrightarrow TM(x) = 0$

Complexity Classes

- P** = $\{L : \exists \text{ polytime TM that accepts } L\}$
- NP** = $\{L : \exists \text{ non-det. polytime TM that accepts } L\}$
- NP** = $\{L : \exists \text{ poly TM s.t. } (x \in L \Leftrightarrow \exists w : TM(x, w) = 1)\}$
→ Thm 1.8: These two definitions are equivalent!
- NP-hard** = $\{L : \forall L' \in \text{NP} : L' \text{ can be reduced to } L\}$
- NP-Complete** = $\text{NP} \cap \text{NP-hard}$
- PSPACE** = $\{L : \exists \text{ TM that accepts } L \text{ with poly memory (in any time)}\}$

Interactive Proof: **IP** = **PSPACE** (poly memory, exp. time)

2. Interactive proofs & Zero-Knowledge protocols

Proof something to someone without transferring the knowledge / revealing the secret to other parties

2.1 Proof systems

$(\text{statement}, \text{proof}) \rightarrow \{\text{accept}, \text{reject}\}$

Requirements

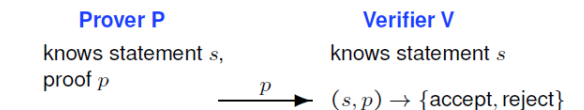
Soundness: only true statements have proofs

(there exists no proof for wrong statements)

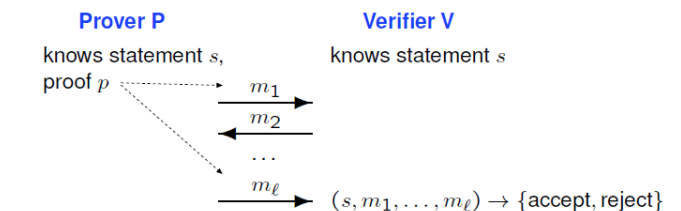
Completeness: every true statement has a proof

Verifiability: verification is efficient / not too complex

Static Proof



Interactive Proof



Prover is unbounded, but Verifier must be efficient

Verifier must be randomized, prover may be deterministic (however, for ZK prover must be randomized as well)

Completeness: V always accept correct proof by P with probability at least $\geq 3/4$

Soundness: accept wrong proof with negligible probability (at most $q \leq 1/2$ for one round of the protocol)

2.2 Zero knowledge

Zero-Knowledge: Verifier learns nothing but that the statement is true (prover knows claimed information)

► Any verifier has no more information than before

An interactive proof (P, V) is **zero-knowledge** if $\forall V'$ there exists an efficient **simulator** S producing a transcript with the same distribution as an actual interaction $V' \leftrightarrow P$ (running time of S is polynomially bounded)

“Everything V could learn, she could also compute herself”

“Only trust results if I can choose input myself”
(as otherwise, might be simulated & not PoK)

Blackbox zero-knowledge: the transcript between $S \leftrightarrow V'$ for any (unknown) V' has the same distribution as $P \leftrightarrow V'$

Honest-verifier zero-knowledge (HVZK): simulator exists for the honest verifier V

c-simulatable: $\forall c$, can efficiently generate triple (t, c, r) with the same distribution as the real protocol with c

► A 3-move c-simulatable protocol is HVZK
(assumption: challenge is efficiently samplable)

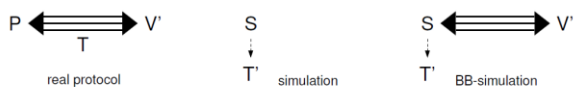
► HVZK round with c uniform from C , $|C|$ small, is ZK

Def: (P, V) is **zero-knowledge (ZK)** $\Leftrightarrow \forall V' \exists S$:

- Transcript T of $(P \leftrightarrow V')$ and output T' of S are **indistinguishable**,
- Running time of S is polynomially bounded in running time of V' .

Def: (P, V) is **black-box zero-knowledge (BB-ZK)** $\Leftrightarrow \exists S \forall V'$:

- Transcript T of $(P \leftrightarrow V')$ and output T' of S in $(S \leftrightarrow V')$ are indist.,
- Running time of S is polynomially bounded.



Def: (P, V) is **honest-verifier zero-knowledge (HVZK)** if S exists for $V' = V$.

2.3 Proof of Knowledge

Witness ω : predicate Q with $Q(x, \omega) = 1$ for x (“secret” / “proof” that x is a member of the language L)

Knowledge extractor: efficient algorithm K which tries to extract ω by interacting with prover P' on input x with non-negligible probability (can amplify by repeating) (can *rewind* the prover with the same randomness)

2-extractable: can extract ω from two accepting triples (t, c, r) and (t, c', r') for same x , $c \neq c'$

► Interactive protocol is a *proof of knowledge* if \exists a *knowledge extractor* K which outputs ω with $Q(x, \omega) = 1$ if V accepts an interaction with P' on input x

► Interacting proof consisting of s 2-extractable 3-move rounds with uniformly chosen challenge is a *proof of knowledge* if $1/|C|^s$ is negligible.

(repeat s rounds; chance that prover can guess all challenges is negligible, as $1/|C|^s$)

Commitment Schemes

Name	Setup	Value	Commit	Type	Comments
GI	G_0, G_1 $G_1 = \sigma G_0 \sigma^{-1}$	$x \in \{0, 1\}$	$B = \pi G_x \pi^{-1}$	H	Trapdoor: σ
DL	$ H = q$ $H = \langle h \rangle$	$x \in \mathbb{Z}_q$	$b = h^x$	B	OR: $\text{LSB}(x)$
Pedersen	$ H = q$ $H = \langle g \rangle = \langle h \rangle$	$x \in \mathbb{Z}_q$	$b = g^x h^r$	H	Trapdoor $\text{DL}_{g,h}$
QR B	$m = pq$, $t \in \text{QNR}$, $(\frac{t}{m}) = 1$	$x \in \{0, 1\}$	$b = r^2 t^x$	B	
QR H	$m = pq$, $t \in \text{QR}$	$x \in \{0, 1\}$	$b = r^2 t^x$	H	Trapdoor \sqrt{t}

Pedersen: $b = g^x h^r$, g and h generators

2.4 Commitment schemes

COMMIT: P uses x as input (V nothing)

OPEN: V outputs either x' (*accept*) or \perp (*reject*)

Correctness: V always outputs $x' = x$ for correct protocol

Hiding: After COMMIT, V has no information about x

Binding: After COMMIT, only one value x will be accepted by V in the OPEN phase (P cannot open commit differently)

Blob: $b = C(x, r)$ for input x and randomness r

OPEN phase uses (x, r) to verify that $C(x, r) = b$

Type H: perfectly hiding (computationally binding)
only computationally PoK (can open two ways)

Type B: perfectly binding (computationally hiding)
only computationally ZK (can find secret)

Trapdoor: can “cheat” binding by knowing this value
→ open blob in at least two ways

One-Way Group Homomorphisms (OWGH)

Setting: Groups $\langle G, \star \rangle$ and $\langle H, \otimes \rangle$

Definition: A **group homomorphism** is a function f with:
 $f : G \rightarrow H, f(a \star b) = f(a) \otimes f(b)$

Notation: We write $[a]$ for $f(a)$, hence

$$[] : G \rightarrow H, [a \star b] = [a] \otimes [b]$$

(One-way homomorphism: easy one way, hard other way)

Knowledge Extractor of OWGH PoK

Theorem 1.5: Protocol round is 2-extractable if
 $\exists \ell \in \mathbb{Z}, u \in G$ s.t. (1) $\forall c_1, c_2 \in C, c_1 \neq c_2 : \gcd(c_1 - c_2, \ell) = 1$
(2) $[u] = z^\ell$

Show zero-knowledge PoK:

- show Graph homomorphism
- Show 2-extractability (i.e. come up with u and ℓ)

3. Multi-Party Computation

Interact with each other without actually knowing & trusting the other parties; act as one trusted party

3.1 Secure MPC Computation

n mutually distrusting parties P_1, \dots, P_n compute function without revealing about individual inputs

Trusted third party (TPP): ideal, reference specification
MPC simulates TPP with multiple parties and “securely realizes” specification if adversary cannot do more

Model: secure channels, synchronous, broadcasts

Central adversary corrupts up to $t < n$ players

Passive corruption: follow protocol, but share info

Active corruption: arbitrarily deviate from protocol







Security properties

Privacy: adversary must not learn about inputs & outputs of uncorrupted parties except what is in specification

Correctness: adversary cannot falsify computation output

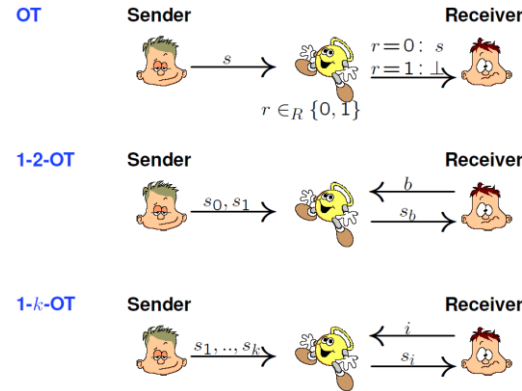
Fairness: adversary cannot abort with an advantage

Robustness: adversary cannot abort protocol at all

			...		
 x_1	x_{11}	x_{12}	...	x_{1n}	
 x_2	x_{21}	x_{22}	...	x_{2n}	
\vdots					
 x_n	x_{n1}	x_{n2}	...	x_{nn}	
	y_1	y_2	...	y_n	$y = \sum_{i=1}^n y_i$

Basic idea: create shares & calculate function based on them so no one knows the original inputs on his own

3.2 Oblivious Transfer



Evaluate function $g : \mathcal{X} \times \mathcal{Y} \rightarrow \Omega$

1. Alice sends function table $g(x, \cdot)$
2. Bob chooses $y \in \mathcal{Y}$, $|\mathcal{Y}| = k$
3. Bob evaluates $g(x, y)$ and sends result back to Alice

(Passively secure, as Alice & Bob can misbehave)

3.3 Passive protocol

Share input

- P_i has input s .
- P_i selects r_1, \dots, r_t at random.
- P_i comp. $\begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} = A \begin{pmatrix} r_1 \\ \vdots \\ r_t \end{pmatrix}$.
- P_i sends s_j to every P_j .

Reconstruct Output

- a is shared by a_1, \dots, a_n .
- every P_j sends a_j to P_i .
- P_i comp. $a = \mathcal{L}(a_1, \dots, a_n)$.

Addition and linear functions \mathcal{L}

- a, b, \dots shared by $a_1, \dots, a_n, b_1, \dots, b_n$, etc.
- every P_i computes $c_i = \mathcal{L}(a_i, b_i, \dots)$.

Multiplication

- a, b are shared by $a_1, \dots, a_n, b_1, \dots, b_n$.
- every P_i computes $d_i = a_i b_i$.
- every P_i shares $d_i \rightarrow d_{i1}, \dots, d_{in}$.
- every P_j computes $c_j = \mathcal{L}(d_{1j}, \dots, d_{nj})$.

CSP: Commitment Sharing Protocol

CTP: Commitment Transport Protocol

CMP: Commitment Multiplication Protocol

Setting	Adv. Type	Condition
cryptographic	passive	$t < n$
cryptographic	active	$t < n/2$
information-theoretic	passive	$t < n/2$
information-theoretic	active	$t < n/3$
i.-t. with broadcast	active	$t < n/2$

Sharing Schemes: passive information-theoretic

t players have no information about s

$t + 1$ players can collaboratively reconstruct the secret

Lagrange Interpolation: n points $(\alpha_1, s_1), \dots, (\alpha_n, s_n)$

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^n \frac{x - \alpha_j}{\alpha_i - \alpha_j}, \quad l_i(\alpha_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

$$g(x) = \sum_{i=1}^n l_i(x) * s_i, \quad \text{goes through all } n \text{ points}$$

Sharing: $p(x) = s + r_1 x + \dots + r_t x^t$, $p(0) = s$

$$s_i = p(\alpha_i) \rightarrow i^{th} \text{ share for player } P_i$$

Can compute secret s uniquely with any $t + 1$ shares

Addition & linear functions: compute share of $c = a + b$

$$c_i = a_i + b_i, \quad \text{as Shamir sharings are linear}$$

Multiplication: compute share of $c = a * b$

1. $d_i = a_i * b_i$
2. Share d_i as d_{ij} to P_j
3. Compute share of c

$$c_i = \sum_{j=1}^n \omega_j d_{ji}, \quad \omega_i = \prod_{\substack{k=1 \\ k \neq j}}^n \frac{\alpha_k}{\alpha_k - \alpha_j}$$

3.4 Active protocol

Divulging secret information: as adversary knows values of corrupted parties anyway, no further harm done

Not sending values: corrupted party cannot send values

Reconstruction: still possible, as $n - t \geq t + 1$ shares

Not receiving share: use **public accusation**

Player not receiving a share publicly accuses dealer

Dealer then broadcasts corresponding share; if he refuses, is disqualified and default value is assumed as input

Not sending product share: either re-run everything, or reconstruct missing share or re-share everything

Sending wrong values: if detected from honest player, react as if nothing had been sent

Commit to *every value* a player knows at *every given time*

Proof in zero-knowledge (e.g. with **BCC Circuit SAT**) that the computation of the new commitment was correct

Homomorphic: can compute a commitment to the sum of two values with only their individual commitments known

Used to calculate commitments to show that it is a valid commitment for result of a linear function *for free* (locally)

CTP: send value & commitment to new party

→ now “committed to it” in exactly the same way

CMP: prove knowledge (& existence) of pre-image of (A, C)

allows $t < n/2$, whereas IT needs $t < n/3$

Cryptographic security [$t < n/2$]

Petersen: type H → unconditional *secrecy*

$$[x, \alpha] = g^x h^\alpha, \quad G = \langle g \rangle = \langle h \rangle, \quad x, \alpha \in \mathbb{Z}_{|G|}$$

El Gamal: type B → unconditional *correctness*

$$[x, \alpha] = (g^\alpha, \gamma^x h^\alpha), \quad G = \langle g \rangle = \langle h \rangle = \langle \gamma \rangle$$

Information-theoretical security [$t < n/3$]

Commitment scheme which is **perfectly hiding & binding**

by constructing a *distributed* scheme based on Shamir

Sharing: use 2dim function against active adversary

Commit Protocol

1. Distribution

D selects random polynomial

$$f(x, y) = \sum_{i=0}^t \sum_{j=0}^t f_{ij} x^i y^j, \text{ with } f_{0,0} = s,$$

and sends $h_i(x) = f(x, \alpha_i)$, $k_i(y) = f(\alpha_i, y)$ to P_i .

2. Consistency checks

$\forall P_i, P_j$: P_i sends $k_i(\alpha_j)$ to P_j , P_j complains if $k_i(\alpha_j) \neq h_j(\alpha_i)$.

D broadcasts $f(\alpha_i, \alpha_j)$.

3. Accusation

$\forall P_i$: if P_i has received contradicting values from D : **accuse** D .

D broadcasts $h_i(x)$ and $k_i(y)$.

Repeat until no further accusation.

4. Compute share

If $> t$ accusations: disqualify dealer.

If $\leq t$ accusations: $s_i = k_i(0)$.

Open Protocol

1. Open

D broadcasts $g(x)$.

2. Check consistency

P_i accuses dealer if $g(\alpha_i) \neq s_i$.

3. Compute secret

If $\leq t$ accusations: $s = g(0)$.

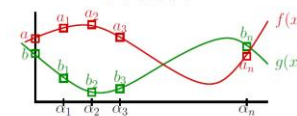
If $> t$ accusations: disqualify dealer.

Generic Commitment Multiplication Protocol

0. **Starting point:** D is committed to a, b, c by \boxed{a} , \boxed{b} , and \boxed{c} .

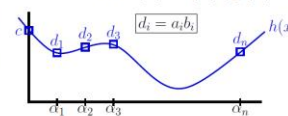
1. CSP of a, b with degree t

$\Rightarrow f(x), g(x)$



2. CSP of c with degree $2t$

use $h(x) = f(x)g(x)$



3. Checks

$\forall P_i$: $d_i \stackrel{?}{=} a_i b_i$, broadcast accusation bit.

On accusation: Open $\boxed{a_i}$, $\boxed{b_i}$, $\boxed{d_i}$, check $a_i b_i \stackrel{?}{=} d_i$.

IT security requires $t < n/3$ (more restrictive than crypto.)

4. Broadcast

Talk bilaterally, agree on what we heard; everyone hears the same & knows this is the case for all parties

4.1 Broadcast

Allows sender to distribute a value to all players with the guarantee that all honest player receive same value & agree on the value sent by the receiver

Consistency: All honest players output same, agreement

Termination: All honest players decide at some point

Validity: If the sender is honest, honest players decide on the value sent by him as input

4.2 Consensus

Every player holds an input; in the end, honest players agree on a value & preserve so-called *pre-agreement*

Pre-agreement: honest parties all have same input

Consistency: All honest players output same, agreement

Termination: All honest players decide at some point

Persistency: If all honest players receive same input, keep

For $t < n/2$, the two can be transformed into each other:

Broadcast vs Consensus

Broadcast: $(x, \perp, \dots, \perp) \rightarrow (y_1, \dots, y_n)$

Consensus: $(x_1, \dots, x_n) \rightarrow (y_1, \dots, y_n)$

Broadcast from Consensus

1. P_1 : send x to every P_j , P_j receives x_j

2. $(y_1, \dots, y_n) = \text{Consensus}(x_1, \dots, x_n)$

3. $\forall P_j$: return y_j

Consensus from Broadcast

1. $\forall P_i$: Broadcast(x_i)

2. $\forall P_j$: return $y_j = \text{majority of received } x_i\text{'s}$

Consensus types [$t < n/3$]

Weak consensus: If some honest player decides on an output $y_i \in \{0,1\}$, all other players decide on $y_j \in \{y_i, \perp\}$

Graded consensus: Player decide how sure they are of their decision by giving a grade $g_i \in \{0,1\}$

$g_i = 0$: "not sure", $g_i = 1$: "consistency achieved"

King consensus: If king is honest, achieve consensus
Otherwise, we keep our pre-agreement and go on

For **consensus**, just keep doing *King consensus* with all parties once as king; due to persistency, we will keep a correct result as soon as once as an honest party was king

As broadcast is necessary for MPC, we can show that it's not working for $t \geq n/3$ in information-theoretic setting

4.3 Adversary structure

Adversary Structure

Notation

- Party set \mathcal{P} , $|\mathcal{P}| = n$
- Monotone adversary structure $\mathcal{Z} = \{Z_1, Z_2, \dots, Z_\ell\} \subseteq 2^{\mathcal{P}}$
(Monotone: $Z \in \mathcal{Z}$, $Z' \subseteq Z \Rightarrow Z' \in \mathcal{Z}$)

Adv. chooses one of them

Definitions

- $Q^2(\mathcal{P}, \mathcal{Z}) : \Leftrightarrow \forall Z_1, Z_2 \in \mathcal{Z} : Z_1 \cup Z_2 \neq \mathcal{P}$ (no two sets add up to \mathcal{P})
- $Q^3(\mathcal{P}, \mathcal{Z}) : \Leftrightarrow \forall Z_1, Z_2, Z_3 \in \mathcal{Z} : Z_1 \cup Z_2 \cup Z_3 \neq \mathcal{P}$
(no three sets add up to \mathcal{P})

Results

	Threshold	Gen.Adv.
• i.t. passive:	$t < n/2$	$Q^2(\mathcal{P}, \mathcal{Z})$ [HM97, Mau02]
• i.t. active:	$t < n/3$	$Q^3(\mathcal{P}, \mathcal{Z})$ [HM97, Mau02]
• crypto. active:	$t < n/2$	$Q^2(\mathcal{P}, \mathcal{Z})$ [HM97, Mau02]

Passive Protocol

Share Input

- P_d has input a .
- P_d selects random summands a_1, \dots, a_ℓ s.t. $\sum a_q = a$.
- P_d sends a_q to every P_i in $\overline{Z_q}$

Reconstruct Output

- a is shared by a_1, \dots, a_ℓ .
- For all q : a fixed $P_i \in \overline{Z_q}$ sends a_q to P_r .
- P_r computes $a = \sum a_q$.

Addition

- a, b shared by $a_1, \dots, a_\ell, b_1, \dots, b_\ell$.
- For all q : Every $P_i \in \overline{Z_q}$ computes $c_q = a_q + b_q$.

Multiplication

- a, b shared by $a_1, \dots, a_\ell, b_1, \dots, b_\ell$.
- For all p, q : A fixed $P_i \in \overline{Z_p} \cap \overline{Z_q}$ computes and shares $a_p b_q$.
- Compute $[c] = \sum \sum [a_p b_q]$.

Multiplication requires $Q^2(\mathcal{P}, \mathcal{Z})$ so that at least one party exists which knows $a_p b_q \quad \forall 1 \leq q, p \leq \ell$

Active Protocol

Again, commit to everything for cryptographic security

Do it information-theoretically:

1. **Consistency check:** send value to other $P_i \in \overline{Z_q}$
2. **Accusation** if not everyone is happy & broadcast

Active Sharing Protocol

Goal: Share input value s of party P_d .

Share(P_d, s)

1. P_D selects random summands $s_1 + \dots + s_\ell$ s.t. $s = \sum s_q$
2. For all $1 \leq q \leq \ell$ do:
 - a) P_D sends s_q to all parties in $\overline{Z_q}$.
 - b) The parties in $\overline{Z_q}$ exchange the received values.
If $P_i \in \overline{Z_q}$ sees different values: complain using broadcast.
 - c) If any $P_i \in \overline{Z_q}$ complained: P_D broadcasts s_q .
Otherwise each $P_i \in \overline{Z_q}$ takes the value received in step 2a) for s_q .

Active Reconstruction

Goal: Reconstruct $[s]$ towards P_r

Share-Reconstruction($P_r, [s], q$)

1. Every party $P_i \in \overline{Z_q}$ sends s_q to P_r .
2. Let v_i be the value P_r received from $P_i \in \overline{Z_q}$.
 P_r outputs v such that $\{P_i \mid v_i \neq v\} \in \mathcal{Z}$.

Reconstruction($P_r, [s]$)

1. For all q invoke $s_q \leftarrow$ Share-Reconstruction($P_r, [s], q$).
2. P_r outputs $s = s_1 + \dots + s_\ell$.

Shared-Reconstruction requires $Q^3(\mathcal{P}, \mathcal{Z})$

Active Multiplication Protocol

Goal: Compute $[c] = [ab]$

Multiplication($[a], [b]$)

1. For all $1 \leq q, p \leq \ell$ do:
 - a) Every party $P_i \in \overline{Z_p} \cap \overline{Z_q}$ shares $a_p b_q$ as $[v_{pq}^k]$.
 - b) Let P_i be a fixed party in $\overline{Z_p} \cap \overline{Z_q}$.
Compute and open $[v_{pq}^i] - [v_{pq}^j]$ for all $P_j \in \overline{Z_p} \cap \overline{Z_q}$.
 - c) If all differences are zero: Set $[v_{pq}] = [v_{pq}^1]$ as a sharing for $a_p b_q$.
Otherwise, reconstruct a_p and b_q (using Share-Reconstruction).
Define $[v_{pq}]$ as the sharing with summands $(a_p b_q, 0, \dots, 0)$.
2. Compute $[c] = \sum [v_{pq}]$.

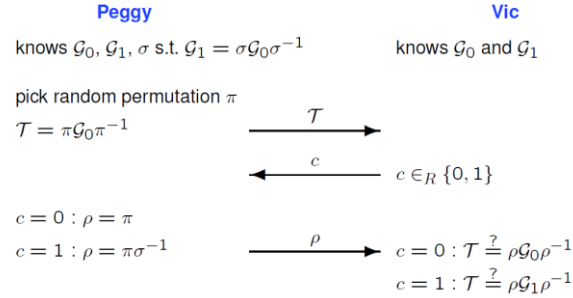
5. Algorithms & Protocols

5.1 Interactive Proofs & ZK PoK

Graph Isomorphism – One Round of the Protocol

Setting: Given two graphs \mathcal{G}_0 and \mathcal{G}_1 .

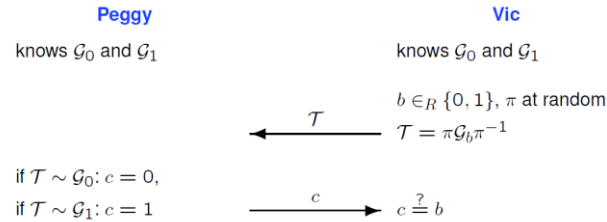
Goal: Prove that \mathcal{G}_0 and \mathcal{G}_1 are isomorphic.



Graph-NON-Isomorphism – One Round of the Protocol

Setting: Given two graphs \mathcal{G}_0 and \mathcal{G}_1 .

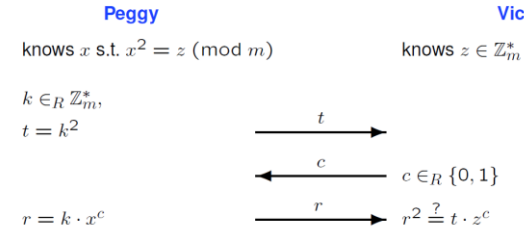
Goal: Prove that \mathcal{G}_0 and \mathcal{G}_1 are *not* isomorphic.



Fiat-Shamir – One Round of the Protocol

Setting: m is an RSA-Modulus.

Goal: Prove knowledge of a square root of a given $z \in \mathbb{Z}_m^*$.



Repeat *sequentially* \rightarrow ZK ($|C| = \{0, 1\}$ small)

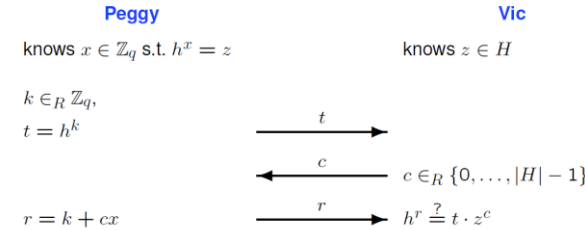
Repeat *in parallel* \rightarrow **not** ZK ($|C| = \{0, 1\}^s$ large)

Use **trapdoors** to get around this problem (don't need to repeat, as poly-time simulator can open blob how it wants)

Schnorr – One Round of the Protocol

Setting: Cyclic group $H = \langle h \rangle, |H| = q$ prime.

Goal: Prove knowledge of the discrete logarithm of a given $z \in H$.



Challenge space $C : c \in_R \{0, \dots, |H| - 1\}, |H| = 2^q$

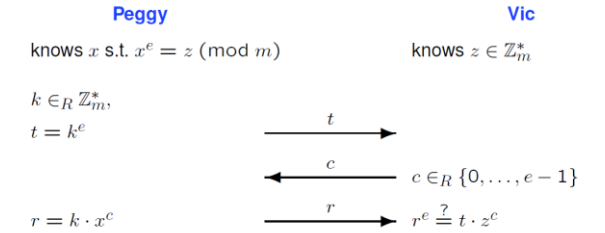
$|C|$ not polynomially bounded, chance to guess small

ZK by restricting challenge space, e.g. $c \in_R \{0, \dots, 100\}$

Guillou-Quisquater – One Round of the Protocol

Setting: m is an RSA-Modulus.

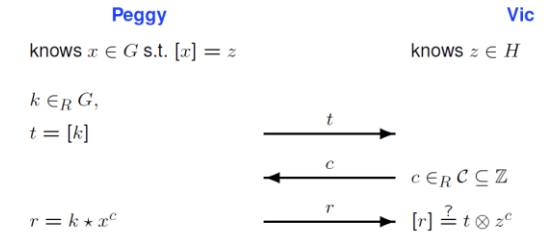
Goal: Prove knowledge of an e -th root of a given $z \in \mathbb{Z}_m^*$.



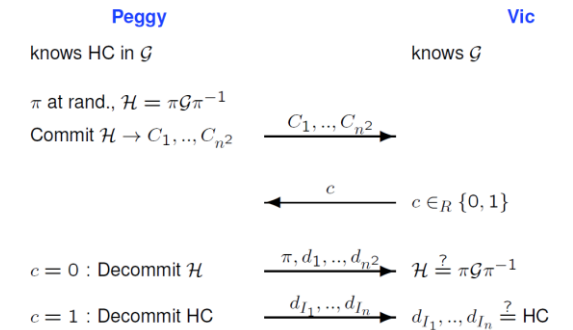
PoK of Pre-Image of OWGH – One Round of the Protocol

Setting: Groups G and H , group homomorphism $[] : \langle G, \star \rangle \mapsto \langle H, \otimes \rangle$.

Goal: Prove knowledge of pre-image of $z \in H$.



Hamiltonian Cycles — One Round of the Protocol

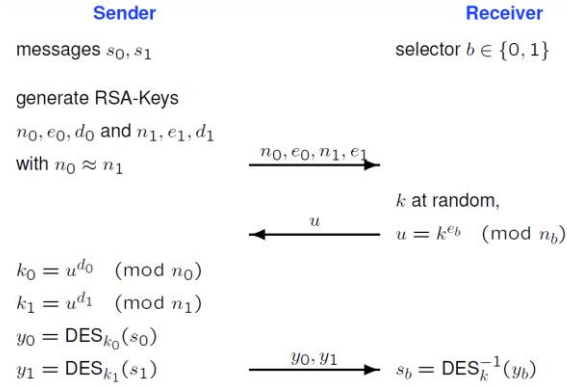


NP-complete: can reduce any NP problem to this

► ZK, PoK \rightarrow ZK proof for all NP problems

5.2 Multi-Party Computation

1-2-OST based on RSA and DES



Commitment Sharing Protocol

Starting point: Dealer is committed to some value s .

Goal: Every player has a share of s and is committed to it.

1. The dealer chooses the random coefficients used in the secret sharing scheme and commits to them.
2. Each player (locally) computes the commitments to all shares (using the homomorphic property).
3. For every player, the dealer transfers the commitment to the share corresponding to that player using the CTP.

5.3 Broadcast

Protocol Weak Consensus

$\text{WeakConsensus}(x_1, \dots, x_n) \rightarrow (y_1, \dots, y_n)$

1. $\forall P_i$: send x_i to every P_j
2. $\forall P_j$: $y_j = \begin{cases} 0 & \text{if } \#Zeros \geq n - t \\ 1 & \text{if } \#Ones \geq n - t \\ \perp & \text{else} \end{cases}$
3. $\forall P_j$: return y_j

Protocol Graded Consensus

$\text{GradedConsensus}(x_1, \dots, x_n) \rightarrow ((y_1, g_1), \dots, (y_n, g_n))$

1. $(z_1, \dots, z_n) = \text{WeakConsensus}(x_1, \dots, x_n)$
2. $\forall P_i$: send z_i to every P_j .
3. $\forall P_j$: $y_j = \begin{cases} 0 & \text{if } \#Zeros \geq \#Ones \\ 1 & \text{if } \#Zeros < \#Ones \end{cases}$

 $g_j = \begin{cases} 1 & \text{if } \#y_j\text{'s} \geq n - t \\ 0 & \text{else} \end{cases}$
4. $\forall P_j$: return (y_j, g_j)

Protocols King Consensus (King P_k) and Consensus

$\text{KingConsensus}_k(x_1, \dots, x_n) \rightarrow (y_1, \dots, y_n)$

1. $((z_1, g_1), \dots, (z_n, g_n)) = \text{GradedConsensus}(x_1, \dots, x_n)$
2. P_k : send z_k to every P_j .
3. $\forall P_j$: $y_j = \begin{cases} z_j & \text{if } g_j = 1 \\ z_k & \text{else} \end{cases}$
4. $\forall P_j$: return y_j

$\text{Consensus}(x_1, \dots, x_n) \rightarrow (y_1, \dots, y_n)$

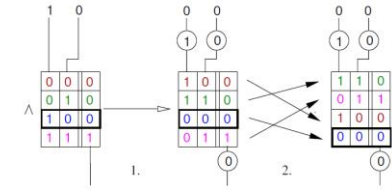
1. for $k = 1$ to $t + 1$ do
 $(x_1, \dots, x_n) = \text{KingConsensus}_k(x_1, \dots, x_n)$
 od
2. $\forall P_j$: return x_j

6. Papers

6.1 Maurer: Unifying ZK PoK

6.2 BCC: Minimum-Disclosure PoK

How to Scramble the Truth Tables



1. XOR every wire with a random bit
2. Permute the rows randomly

Challenges:

$c = 0$: show scrambled circuit

$c = 1$: unblind rows of the truth table which are used
check that output = 1 ("valid proof")

6.3 Maurer: Secure MPC made simple