CRYPTOGRAPHY LECTURE, 2023

Master Level, PWr

Mirosław Kutyłowski

Zero Knowledge

Challenges for a cryptographic protocol based on a secret key sk:

Problem 1

- each time when a key is used then some information about sk may be leaked
- how to make sure that gradually sk becomes not that "private"?

Problem 2

- a protocol execution may be used for more than intended
- how to prove that no extra protocol is executed so that one participant is not aware of it

Example: Challenge+response authentication $\Rightarrow$ blind signature

Original protocol:

- Alice holds private RSA key d , Bob holds the corresponding e and modulus n
- Alice shows that show holds d:
  i. Bob chooses r at random
  ii. Alice computes $c := r^d \bmod n$
  iii. Bob checks that $c^e = r \bmod n$

Corrupted execution (Bob is malicious)

i. Bob chooses a message $h = \mathrm{Hash}(\mathrm{padding}(m))$ and random u

ii. Bob calculates: $r := h \cdot u^e \bmod n$ and presents r as the random challenge

iii. Alice computes $c := r^d \bmod n$

iv. Bob computes $s := c/u \bmod n$

note that

$$s = r^d/u = (h \cdot u^e)^d/u = h^d \cdot (u^e)^d/u = h^d \cdot u/u = h^d \mod n$$

voila! Bob has the signature of Alice under m

Consequence: "key usage"– if for "authentication", then never for "signing"

Blind signature – non-malicious application:

the mechanism of **Privacy Pass**

- Issuer signs blinded tokens presented by the client
- the client unblinds the signatures ...
- ... and presents them instead of solving Captchas on third party webpages

token is a proof that it comes from a client

privacy protected against the Issuer

Example 2: Schnorr authentication converted to proof-of-presence

Original protocol:

Alice: secret key $\mathrm{SK} = x$, public key $\mathrm{PK} = g^x$

i. Alice chooses k at random, $r := g^k$

ii. Alice sends r to Verifier

iii. Verifier chooses c at random and sends to Alice

iv. Alice calculates $s := k - x \cdot c \bmod q$ and sends s to Verifier

v. Verifier checks that $g^s \cdot \mathrm{PK}^c = r$

Malicious: $c := \mathrm{Hash}(z)$ where $z := \mathrm{Sign}_{\mathrm{sk(verifier)}}(r)$:

– Alice cannot detect that c has not be chosen at random

– Verifier can show (r, s) and z to prove that he has interacted with Alice

Simultability

Alice holds private key sk and authenticates against Bob

Bob cannot create a proof that he has interacted with Alice:

no matter how clever are the challenges of Bob,

... Bob can create the responses of Alice by himself

Consequence:

each transcript of interaction can be simulated and therefore

useless as a proof of interaction when presented by Bob to a third person

Password authentication example

situation:

- Alice knows a password π
- a smartcard of Alice holds π as well
- smartcard establishes a session with a reader iff the reader proves to know π

problems:

- π must not be sent in cleartext
- ... passive (offline) and active attacks should not reveal π
- no proof-of-presence can be created by the reader

Password Authenticated Key Exchange (PAKE) protocol

- session key established iff the same password used on both sides
- password and session key are secure

example: PACE protocol (obligatory on European personal ID documents, on passports)

PACE executed with password π

Phase 0:

i. card and reader: compute symmetric key $K_\pi = \mathrm{Hash}(\pi \| 0)$

ii. card: choose s at random, send $z = \mathrm{Enc}_{K_\pi}(s)$ to the reader

iii. reader: $s := \mathrm{Dec}_{K_\pi}(z)$

Phase 1: (Diffie-Hellman Key Exchange)

i. reader: choose x_B at random, $X_B := g^{x_B}$, send X_B to the reader

ii. card: choose x_A at random, $X_A := g^{x_A}$, send X_A to the reader

iii. reader: $h := (X_A)^{x_B}$, $\hat{g} := h \cdot g^s$

iv. card: $h := (X_B)^{x_A}$, $\hat{g} := h \cdot g^s$

Phase 2: (Diffie-Hellman Key Exchange based on password dependant $\hat{g}$)

i. reader: choose y_B at random, $Y_B := \hat{g}^{y_B}$, send Y_B to the reade

ii. card: choose y_A at random, $Y_A := \hat{g}^{y_A}$, send Y_A to the reader

iii. reader: $K := (Y_A)^{y_B}$

iv. card: $K := (Y_B)^{y_A}$

v. both: derive K_{MAC} and K_{Enc} via hashing from K

Phase 3: (proof of key possesion)

i. reader: send $T_B := \mathrm{MAC}(K_{\mathrm{MAC}}, Y_A)$ to card

ii. card: abort if T_B invalid

iii. card: send $T_A := \mathrm{MAC}(K_{\mathrm{MAC}}, Y_B)$ to reader

iv. reader: abort if T_A invalid

Start session

all messages encrypted with K_{Enc}

Simulation (reader pretending to talk with the card)

Phase 0:

i. reader: $K_\pi = \mathrm{Hash}(\pi \| 0)$

ii. "card": send z chosen at random

iii. reader: $s := \mathrm{Dec}_{K_\pi}(z)$

Phase 1: (Diffie-Hellman Key Exchange)

i. reader: choose x_B, $X_B := g^{x_B}$, send X_B to the reader

ii. "card": send X_A chosen at random

iii. reader: $h := (X_A)^{x_B}$, $\hat{g} := h \cdot g^s$

iv. "card": —

Phase 2: (Diffie-Hellman Key Exchange based on password dependant $\hat{g}$)

i. reader: choose y_B, $Y_B := \hat{g}^{y_B}$, send Y_B

ii. "card": send Y_A chosen at random

iii. reader: $K := (Y_A)^{y_B}$

iv. "card": —

v. reader: derive K_{MAC} and K_{Enc}

Phase 3: (proof of key possesion)

i. reader: send $T_B := \mathrm{MAC}(K_{\mathrm{MAC}}, Y_A)$

ii. "card": abort if T_B invalid

iii. "card": send $T_A := \mathrm{MAC}(K_{\mathrm{MAC}}, Y_B)$

iv. reader: abort if T_A invalid

Observation: **exactly the same probability distribution as in the case of real interactions**

Session confidentiality

how to find out that an observer will not learn anything about the data exchanged after establishing K_{Enc} ?

maybe some information will be leaked

Proof method??

Abdalla model

two models considered:

- real model: protocol executed as described, session key K established
- artificial model: protocol executed as described, but finally K replaced by a random key

Game:

i. run some number of instances of real model (adversary involved, keys revealed, etc)

ii. choose b at random, run

→ real protocol, if $b = 0$

→ artificial protocol, if $b = 1$

iii. again run some instances like in point (i)

iv. challenger reveals the session key K from point (ii)

v. adversary wins if guesses b

Observations

1. if the artificial protocol is run, then the adversary cannot learn anything about the workload data from key agreement (obvious - key K is unrelated)

2. for the game: one can add the session created with K and run an adversary that wants to break session confidentiality

   if adversary succeeds, then we conclude that it is rather a real session

So

it suffices to show that the adversary has a negligible advantage to win the game

How is it for PACE (passive adversary)?

can the adversary detect that master key K has been replaced with a random key?

that is:

K is random instead of being the solution for DH tuple $\hat{g}, Y_A, Y_B$?

– give adversary the discrete logarithm of $\hat{g}$ – it would only help

– nevertheless, the adversary would have to solve the Decisional Diffie-Hellman Problem

Noninteractive proof:

a language L (e.g. tautologies), an element x (e.g., a sentence),

a proof for $x \in L$ **to be verified offline**

Interactive Proofs

Actors:

→ **prover:** aims to show that $x \in L$

→ **verifier:** should check the proof for $x \in L$

Protocol: exchange of messages after which the verifier says "valid" ($x \in L$) or "invalid"

Completeness: if $x \in L \Rightarrow$ the prover can convince the verifier to answer "valid"

Soundness: if $x \notin L \Rightarrow$ the probability that the verifier answers "valid" is ≤ 0.5

if $x \notin L$ then $\forall$ prover strategy P: $\Pr((V, P)[x] = \text{valid}) \leq 0.5$

all NP languages, example: to prove that a graph has a Hamiltonian path:

i. Prover shows a Hamiltonian path P

ii. Verifier checks that P is Hamiltonian

Generally:

i. Prover sends a witness w for $x \in L$

ii. Verifier: test on (w, x)

Interactive proofs with many messages exchanged:

Theorem.

IP=PSPACE

That is:

there is a polynomial interactive proof for L

$\Leftrightarrow$ there is an algorithm A using polynomial space that $A(x) = 1$ iff $x \in L$

How to prove without disclosing no information but the fact that $x \in L$?

example: graph isomorphism: prove that graphs G and H are isomorphic without disclosing the isomorphism ϕ

protocol consists of k independent rounds.

A round:

i. Prover generates permutation π, computes $Z = \pi(G)$, and sends Z to Verifier

ii. Verifier chooses bit b at random and sends to Prover

iii. Prover returns: π if $b = 0$ (isomorphism $G \to Z$) else $\pi \circ \phi^{-1}$ (isomorphism $H \to Z$)

iv. Verifier aborts if the function is not an isomorphism

Completeness: obvious

Soundness: if H and G are not isomorphic, then Z can be isomorphic to at most one of them,

proof rejected in at least 50% cases

Interactive proofs are stronger than non-interactive

example: how to show that graphs G_0 and G_1 are **not isomorphic**

Round

i. Verifier: choose b at random, permutation π, calculate $C := \pi(G_b)$, send C

ii. Prover: finds bit b' such that $G_{b'}$ isomorphic to C, returns b'

iii. Verifier: aborts if $b' \neq b$

Prover: unlimited computational power, verifier: randomized polynomial time

Prover: if G_0 and G_1 isomorphic $\Rightarrow$ pbb to guess b is 0.5

no clever strategy of Prover can increase this probability

Knowledge gained by Verifier

zero-knowledge $\equiv$ *Verifier learns nothing but that* $x \in L$

(e.g. $x \in L$ iff x is a product of two large primes , then the proof must not reveal the factors of x)

we are talking about additional knowledge about x:

i. any polynomial time computation of Verifier is not additional knowledge

ii. output of RNG is not additional knowledge

View of of interactive proof for $x \in L$

- all messages exchanged
- all values of internal variables of Verifier

Concept of a simulator

an interaction brings no additional knowledge if

the Verifier can create views without the help of Prover so that they are indistinguishable from real ones:

$$\exists\,\text{simulator}\,S \quad \forall x \in L \quad \text{VIEW}_{P,V}(x) \backsimeq S[x]$$

where:

$\backsimeq$ means:

- the same distribution, or
- statistically indistinguishable distributions, or
- computationally indistinguishable

Dishonest Verifier

different strategies,

for example instead of choosing r uniformly at random use a different distribution

a smart strategy $\Rightarrow$ information leakage?

Definition

$\forall\,\text{verifier}\,W \quad \exists\,\text{simulator}\,S \quad \forall x \in L \quad \text{VIEW}_{P,W}(x) \backsimeq S[x]$

for each strategy of the Verifier:

knowledge gained from Prover can be obtained without interaction

Black Box Simulator

a simulator that works for any strategy of the verifier – treated as a black box

Definition

$\exists\,\text{simulator}\,S \quad \forall\,\text{verifier}\,W \quad \forall x \in L \quad \text{VIEW}_{P,W}(x) \backsimeq S_W[x]$

S_W uses W as an oracle

Black Box simulator for graph isomorphism

round i

i. Simulator: selects bit b_i and permutation π_i, $C := \pi_i(G_{b_i})$

ii. Simulator: feeds C_i to the black box Verifier

iii. Verifier: returns b_i'

iv. Simulator: if $b_i \neq b_i'$ then goto (i), else record C_i, π_i, b_i to the transcript of round i

this is like "rejection sampling"

Protocol Example:

Zero Knowledge with an honest Verifier

not Zero Knowledge with dishonest Verifier

Round

i. Verifier: choose b at random, permutation π, calculate $C := \pi(G_b)$, send C

ii. Prover: finds bit b' such that $G_{b'}$ isomorphic to C, returns b'

iii. Verifier: aborts if $b' \neq b$

Malicious Verifier: aiming to learn whether C is isomorphic to G_0 or G_1

* Verifier: send C
* Prover: finds bit b' such that $G_{b'}$ isomorphic to C, returns b' , or aborts
* Verifier: learns that $G_{b'}$ isomorphic to C

this cannot be simulated!

ZK protocol for Graph non-isomorphism with black box ZK

i. Verifier: chooses bit b, permutation π at random, $X := \pi(G_b)$

ii. Verifier: for $i \leq 2k$, choose b_i and π_i at random, $X_i := \pi_i(G_{b_i})$

iii. Verifier: present $X, X_1, ..., X_{2k}$ to Verifier

iv. Prover: choose $2k$ random bits d_i, send them to Prover

v. Verifier: if $d_i = 0$, then respond with π_i

if $d_i = 1$, then respond either with $\perp$ or an isomorphism $\phi_i : X \to X_i$

vi. Prover: rejects the proof if the number of $\perp$ is not higher than $2k/3$, checks the isomorphisms

vii. Prover: sends b'

viii. Verifier: accepts if $b = b'$

Completeness

answer $\perp$ for the positions where Verifier does not know the isomorphism

Prover: calculations (exponential) but knows that Verifier is not cheating

Soundness

assume that G_0 and G_1 are isomorphic. Can the prover gain some knowledge about b?

- case $d_i = 0$: π_i does not help as it is independent of b
- case $d_i = 1$: $\phi_i : X \to X_i$ but such isomorphism can be presented if Verifier has chosen ($X = G_0$ and $X_i = G_0$) or ($X = G_1$ and $X_i = G_1$), no information on X!
- case $d_i = 1$: $\perp$ only information that ($X = G_1$ and $X_i = G_0$) or ($X = G_0$ and $X_i = G_1$)

ZK - construction of Black Box Simulator

1. simulator starts a conversation with Black Box Verifier V,
2. it is run until simulator has to present b (impossible), simulation frozen for a moment
3. rerun with Black Box with the same randomness for Black Box, but different response of Prover at step (iv)
4. if in a response: one Black Box gave π_i and another gave ϕ_i then we have an isomorphism between G_b and X . Then return to the first simulation and resond with b
5. if not the case: cosntant amount of work of exponential algorithm that decides that G_0 and G_1 are not isomorphic and goto (3)

execution time of the simulator: probabilistic polynomial

ZK Proof of 3-colorings of a graph

3-coloring is a NP-complete problem:

given a NP problem L instance , there is transformation T such that

$$x \in L \qquad \Leftrightarrow \qquad T(x) \text{ has 3-coloring}$$

Corollary: a ZK Interactive proof for 3-coloring can be used to get ZKIP for L

(efficiency – not necessarily optimal)

ZKIP for 3-coloring

given a graph G, with vertices V and edges E, Prover knows a 3-coloring C

used: probabilistic encryption Enc, $\mathrm{Enc}(b, v)$ means encoding of b using randomness v

round of interaction:

i. Prover: permutes the colors $C \to C'$

   Prover: for each $v \in V$, presents $\mathrm{Enc}(C'(v))$

ii. Verifier: chooses an edge (v_i, v_j) at random

iii. Prover: reveals $C'(v_i), C'(v_j)$ and randomness used for encrypting them

iv. Verifier: checks: $C'(v_i) \neq C'(v_j)$, recalculates $\mathrm{Enc}(C'(v_i)), \mathrm{Enc}(C'(v_j))$ and compares with ciphertexts from step (i)

Completeness: if Prover knows 3-coloring then succeeds in each round

Soundness: if does not know 3-coloring then in each round at least 1 pair of vertices with invalid colors

$\rightarrow$ probability to succeed in one round at most $1 - \frac{1}{|E|}$

$\rightarrow$ in $n \cdot |E|$ rounds: $\left(1 - \frac{1}{|E|}\right)^{n \cdot |E|} \approx e^{-n}$

ZK property

trivial simulator:

i. Sim: encode random coloring with 3 colors

ii. Black Box: returns (v_i, v_j)

iii. Sim: if colors different at v_i and v_j then record this round, else goto (i)

Proofs of knowledge

proving existence $\leftrightarrow$ knowledge of solution are different issues

Example:

Cyclic group, generator g, element h:

Proof of existence: $\exists x\, h = g^x$ (some mathematical proof)

Proof of knowledge: requires x (DL Problem)

Sigma protocols -proof of knowledge of w

i. Prover: commitment for k

ii. Verifier: challenge e

iii. Prover: responds with $f(e, w, k)$

iv. Verifier: checks the answer

Example: Schnorr identification

More general: Arthur-Merlin games: verifier sends only random values

Thm

AM games with polynomial number of rounds exists for L iff $L \in \mathrm{PSPACE}$

Noninteractive Proofs of knowledge

Fiat-Shamir heuristics:

replace the random choice in Arthur-Merlin Game by hash values

works in ROM

SNARKs -Siccinct Noninteractive Argument of Knowledge

Common reference string (CRS)

$\rightarrow$ fixed before the proofs are created

$\rightarrow$ available to Prover and Verifier

Procedures: proof creation, proof verification

Completeness: if Prover knows witness w $\Rightarrow$ verification: `accept`

Knowledge soundness:

adversary creates a valid proof π $\Rightarrow$ Extractor takes internal values of adversary and yields a witness

Zero-knowledge: distributions of proofs indistinguishable from

distribution of fake proofs created with trapdoor to CRS

Target: small size of the proof, low computational complexity of verification

$\Rightarrow$ Succinct NARK (SNARK)

Idea 1: PCP

probabilistically checkable proofs $\mathrm{PCP}(r(n), q(n))$:

- for input of size n verifier reads $O(q(n))$ bits and takes $O(r(n))$ bits from RNG
- verifier accepts if proof correct
- verifier accepts with pbb ≤ 0.5 if proof incorrect

Theorem PCP(log(n),1)=NP

Interctive proof

i. create a proof π

ii. build a Merkle tree with the leaves containing bits of π

iii. send the root to the verifier

iv. verifier: choose leaves to be shown and checked

v. prover: disclose these leaves and the path to the root

vi. verifier: check consistency with the Merkle tree, check π

Circuits

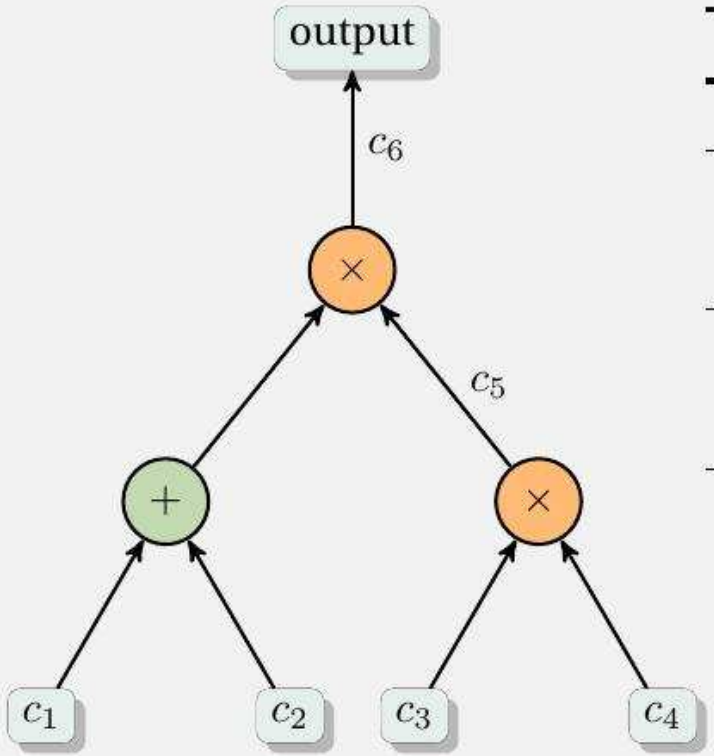
knowledge = knowledge of input x to circuit C such that $C(x) = 1$

- Boolean circuits (Boolean gates for bits)
- arithmetic circuits (arithmetic gates for elements of field $\mathbb{F}$)

circuits are equivalent to Turing Machines

steps 1: encode C to a problem of polynomials

Conversion to QAP (Quadratic Arithmetic Program)



Roots	**Polynomials in QAP** $(\mathcal{V}, \mathcal{W}, \mathcal{Y}, t(x))$		
Gates	Left inputs	Right inputs	Outputs
r_5	$v_3(r_5) = 1$ $v_i(r_5) = 0,$ $i \neq 3$	$w_4(r_5) = 1$ $w_i(r_5) = 0,$ $i \neq 4$	$y_5(r_5) = 1$ $y_i(r_5) = 0,$ $i \neq 5$
r_6	$v_1(r_6) = v_2(r_6) = 1$ $v_i(r_6) = 0,$ $i \neq 1, 2$	$w_5(r_6) = 1$ $w_i(r_6) = 0,$ $i \neq 5$	$y_6(r_6) = 1$ $y_i(r_6) = 0,$ $i \neq 6$
$t(x) = (x - r_5)(x - r_6)$			

fig: Anca Nitulescu , for each multiplication gate a random $r_i \in \mathbb{F}$ was selected

QAP conversion

i. choose random r_i for each mnultiplication gate

ii. addition gates treated implicitly

iii. build a polynomial (fig. A.Nitulescu)

$$p(x) := \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right) \cdot \left(w_0(x) + \sum_{i=1}^{m} c_i w_i(x)\right) - \left(y_0(x) + \sum_{i=1}^{m} c_i y_i(x)\right).$$

LEMMA

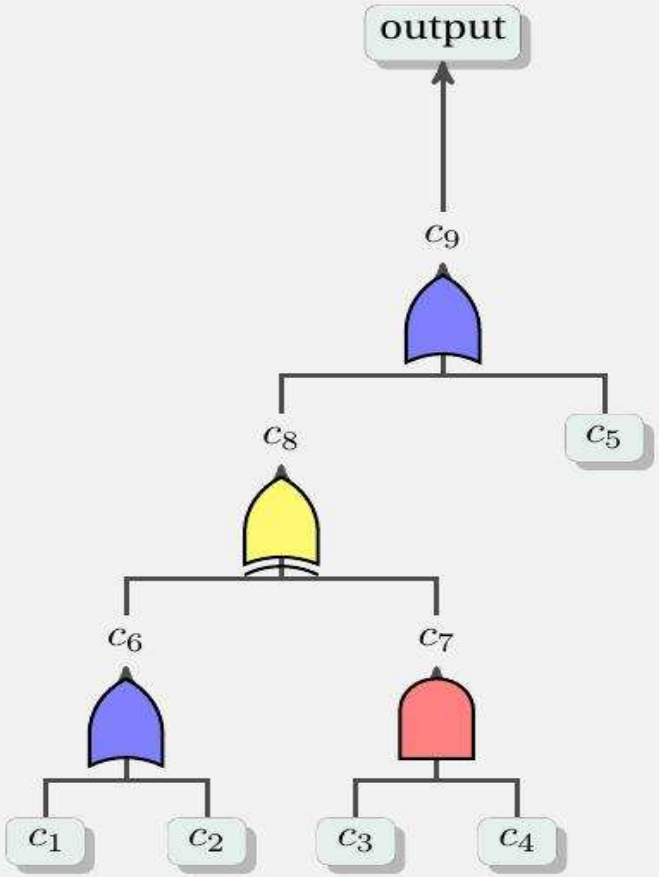
for $p(r_i) = 0$ if the values $c_{...}$ describe correctly the inputs and outputs of the $i$the gate

(irrelevant values for this gate are multiplied with 0's of polynomials $v_{...}, w_{...}, y_{...}$)

Corollary

solution correct iff $\prod (x - r_i)$ divides polynomial $p(x)$

Linearization for Boolean circuits



Linearization of logic gates

OR ($c_1 \vee c_2 = c_6$)	AND ($c_3 \vee c_4 = c_7$)
c_1 c_2 c_6	c_3 c_4 c_7
0 0 0	0 0 0
0 1 1	0 1 0
1 0 1	1 0 0
1 1 1	1 1 1
$-c_1 - c_2 + 2c_6 \in \{0, 1\}$	$c_3 + c_4 - 2c_7 \in \{0, 1\}$

XOR gate, input and output bits

XOR ($c_6 \oplus c_7 = c_8$)	IN ($\{c_i\}_{i=1}^{8}$)	OUT (c_9)
c_6 c_7 c_8	c_i	c_9
0 0 0 0 1 1 1 0 1 1 1 0	$\in \{0, 1\}$	1
$c_6 + c_7 + c_8 \in \{0, 2\}$	$2c_i \in \{0, 2\}$	$3 - 3c_9 \in \{0, 2\}$

Trick

image of $L \in \{0, 2\}$ iff $(L-1)^2 - 1 = 0$

Construction of polynomial

i. express linearization by $\boldsymbol{V}, \boldsymbol{b}$ co that for values $\boldsymbol{c}$ on the wires the values computed are expressed $\boldsymbol{c} \cdot \boldsymbol{V} + \boldsymbol{b}$

ii. take $(\boldsymbol{c} \cdot \boldsymbol{V} + \boldsymbol{b} - \mathbf{1}) \circ (\boldsymbol{c} \cdot \boldsymbol{V} + \boldsymbol{b} - \mathbf{1}) - \mathbf{1}$

iii. take elements r_i at random

iv. polynomials $v_0(r_j) = b_j - 1$, $v_i(r_j) = V_{i,j}$

$$\prod_{i=1}^{d-1}(x - r_i) \text{ divides } \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right)^2 - 1.$$

Checking divisibility $f(x) | \, p(x)$

compute quotent polynomial h

check for random $z = z_1, z_2, \ldots$: $f(z) \cdot h(z) = p(z)$

number of tests=degree of polynomial p

all computations will be in the exponent!

Encoding of information (some briliant tricks envolved)

$$\mathrm{Enc}(a) = g^a$$

key property 1: it is homomorphic for addition:

$$\mathrm{Enc}(a) \cdot \mathrm{Enc}(b) = \mathrm{Enc}(a+b)$$

key property 2:

given $\mathrm{Enc}(s), \mathrm{Enc}(s^2), \mathrm{Enc}(s^3), \ldots, \mathrm{Enc}(s^d)$ one can compute $\mathrm{Enc}(p(s))$ for any polynomial of degree $\leq d$

Additional assumption: the group used has a bilinear mapping e

key property 3:

easy test one can check if $p(s) = 0$, when only $\mathrm{Enc}(p(s))$ given:

test: $e(\mathrm{Enc}(g^{p(s)}), g) = e(g, g)^0$?

key property 4: given $\mathrm{Enc}(h(s))$, $\mathrm{Enc}(t(s))$ and $\mathrm{Enc}(p(s))$ one can check whether

$$h(s) \cdot t(s) = p(s)$$

Test:

$$e(\mathrm{Enc}(h(s), \mathrm{Enc}(t(s)))) = e(g, \mathrm{Enc}(p(s)))\,?$$

remark: no need to reveal $h(s)$ for computation $\mathrm{Enc}(t(s))^{h(s)}$ to get $\mathrm{Enc}(t(s) \cdot h(s)\,)$

key property 5/assumption:

given: $\mathrm{Enc}(s), \mathrm{Enc}(s^2), \mathrm{Enc}(s^3), \ldots, \mathrm{Enc}(s^d)$ and $\mathrm{Enc}(\alpha \cdot s), \mathrm{Enc}(\alpha \cdot s^2), \mathrm{Enc}(\alpha \cdot s^3), \ldots, \mathrm{Enc}(\alpha \cdot s^d)$

task: compute $\mathrm{Enc}(h(s))$ and $\mathrm{Enc}(\alpha \cdot h(s))$

assumption: possible iff all coefficients of h are known

trick 6: Zero-knowledge — randomization of polynomials:

instead of showing that you know $h(x)$ such that $p(x) = t(x) \cdot h(x)$ it suffices to show it for polynomials $p'(x) = p(x) + \gamma \cdot t(x)$ where γ is random

so: pbb distribution of $p'(s)$ is uniform

key property 7:

instead of showing that $p(x) = t(x) \cdot h(x)$ (with revealing the polynomials)

it suffices to show

$$p(s) = t(s) \cdot h(s)$$

for unknown random s

(if $p(x) \neq t(x) \cdot h(x)$ then for only a few elements we have $p(s) = t(s) \cdot h(s)$

remark: there are also solutions in general groups (famous Groth-Sahai approach)

so: universal tools for ZK-SNARKs, frontline: **optimizing compexity**

SNARKs from QAP

pict: A.Nitulescu

$\mathsf{Gen}(1^\lambda, \mathsf{C})$

$\mathsf{gk} := (p, \mathbb{G}, \mathbb{G}_T, \mathsf{e})$

$s, \alpha, \beta_v, \beta_w, \beta_y \leftarrow\!\!\$\ \mathbb{Z}_q$

$Q := \left(\{v_i, w_i, y_i\}_{i\in[m]}, t\right)$

$\mathcal{I}_{\mathsf{mid}} = \{N+1, \ldots m\}$

$\mathsf{crs} := \Big(Q, \mathsf{gk},$

$\{g^{s^i}, g^{\alpha s^i}\}_{i=0}^{d}$

$g^{\beta_v}, \{g^{\beta_v v_i(s)}\}_{i\in\mathcal{I}_{\mathsf{mid}}}$

$g^{\beta_w}, \{g^{\beta_w w_i(s)}\}_{i\in[m]}$

$g^{\beta_y}, \{g^{\beta_y y_i(s)}\}_{i\in[m]}\Big)$

return crs

$\mathsf{Prove}(\mathsf{crs}, u, w)$

$u := (c_1, \ldots, c_N)$

$w := (\{c_i\}_{i\in\mathcal{I}_{\mathsf{mid}}})$

$v_{\mathsf{mid}} := \sum_{i\in\mathcal{I}_{\mathsf{mid}}} c_i v_i(x)$

$H := g^{h(s)}, \quad \widehat{H} := g^{\alpha h(s)}$

$V_{\mathsf{mid}} := g^{v_{\mathsf{mid}}(s)}, \ \widehat{V}_{\mathsf{mid}} := g^{\alpha v_{\mathsf{mid}}(s)}$

$W := g^{w_{\mathbf{c}}(s)}, \quad \widehat{W} := g^{\alpha w_{\mathbf{c}}(s)}$

$Y := g^{y_{\mathbf{c}}(s)}, \quad \widehat{Y} := g^{\alpha y_{\mathbf{c}}(s)}$

$B := g^{\beta_v v_{\mathbf{c}}(s) + \beta_w w_{\mathbf{c}}(s) + \beta_y y_{\mathbf{c}}(s)}$

$\pi := (H, \widehat{H}, V_{\mathsf{mid}}, \widehat{V}_{\mathsf{mid}}, W, \widehat{W}, Y, \widehat{Y}, B)$

$\mathsf{Ver}(\mathsf{crs}, u, \pi)$

Extractability check:

$\mathsf{e}(H, g^\alpha) = \mathsf{e}(g, g^{\widehat{H}})$

$\mathsf{e}(V_{\mathsf{mid}}, g^\alpha) = \mathsf{e}(g, g^{\widehat{V}_{\mathsf{mid}}})$

$\mathsf{e}(W, g^\alpha) = \mathsf{e}(g, g^{\widehat{W}})$

$\mathsf{e}(Y, g^\alpha) = \mathsf{e}(g, g^{\widehat{Y}})$

Divisibility check:

$\mathsf{e}(H, g^{t(s)}) = \mathsf{e}(V, W)/\mathsf{e}(Y, g)$

Linear span check:

$\mathsf{e}(B, g) = \mathsf{e}(V, g^{\beta_v}) \cdot \mathsf{e}(W, g^{\beta_w}) \cdot \mathsf{e}(Y, g^{\beta_y})$