9 zk-snark

tkkwon@snu.ac.kr

source: Ariel Gabizon, https://z.cash/blog/snark-explain

outline

- zk-snark?
- Homomorphic Hidings (HH)
- polynomials and linear combinations
- knowledge of Coefficient (KC) test
- extended KCA
- verifiable blind evaluation protocol
- Quadratic Arithmetic Program (QAP)
- Pinocchio Protocol

zk-snark

- “Zero-Knowledge Succinct Non-Interactive Argument of Knowledge”
- “Zero-knowledge” proofs allow one party (the prover) to prove to another (the verifier) that a statement is true, without revealing any information beyond the validity of the statement itself
- “Succinct” zero-knowledge proofs can be verified within a few milliseconds, with a proof length of only a few hundred bytes even for statements about programs that are very large
- “non-interactive” constructions, the proof consists of a single message sent from prover to verifier
    - we need initial setup phase that generates a common reference string shared between prover and verifier.
- “argument of Knowledge” the prover can convince the verifier not only that the number exists, but that they know such a number – again, without revealing any information about the number

homomorphic hiding (HH)

- An HH $E(x)$ of a number x is a function satisfying the following:
  - for most x's, given $E(x)$, it is hard to find x
  - different inputs lead to different outputs
    - if $x \neq y$, $E(x) \neq E(y)$
  - if someone knows $E(x)$ and $E(y)$, he can generate the HH of arithmetic expressions in x and y
    - e.g. $E(x+y)$ can be calculated from $E(x)$ and $E(y)$

HH example

- Alice wants to prove to Bob that she knows number x, y such that x+y=7
  1. Alice sends E(x), E(y) to Bob
  2. Bob computes E(x+y) from these values
  3. Bob also computes E(7), and now checks whether E(x+y) = E(7)
- in this case, we say HH supports addition

before HH: revisit modular multiplication

- group: a set of elements with a binary operation
  - the outcome of the operation should satisfy four properties below
- The group of positive integers modulo a prime *p*

  $Z_p^* \equiv \{1, 2, 3, \ldots, p\text{-}1\}$

  $*_p \equiv$ multiplication modulo p

  Denoted as: $(Z_p^*, *_p)$
- Required properties
  1. Closure. Yes.
  2. Associativity. Yes.
  3. Identity. 1.
  4. Inverse. Yes.
- Example: $Z_7^* = \{1,2,3,4,5,6\}$

  $1^{-1} = 1$, $2^{-1} = 4$, $3^{-1} = 5$, $6^{-1} = 6$

HH construction

(mod p) is omitted below

- if we want to prove we know x, y with x+y=7
  - g: generator of group of order p where DLP is hard.
  - Prover: sends $E(x) = g^x$, $E(y) = g^y$
  - Verifier: Checks $E(x+y) = g^{x+y \text{ mod } (p-1)} = g^x g^y = E(x)E(y)$

polynomial

- F_p is the field of size p; the elements of F_p are {0,…,p−1} and addition and multiplication are done mod p
- a polynomial P of degree d over is an expression as follows:

  $$P(X) = a_0+a_1X^1+a_2X^2+..+a_dX^d \text{ for some } a_0,\dots, a_d \in F_p$$
- we can evaluate P at a point $s \in F_p$

  $$P(s) = a_0+a_1s^1+a_2s^2+..+a_ds^d$$
- note that P(s) is a linear combination of $1,s^1,s^2,\dots,s^d$
- HH supports linear combinations, which means
  - given a,b,E(x),E(y), we can compute E(ax+by)

  $$E(ax+by) = g^{ax+by} = g^{ax}\, g^{by} = (g^x)^a\,(g^y)^b \quad = E(x)^a\, E(y)^b$$

blind evaluation of a polynomial: a naïve approach

- Alice has a polynomial P of degree d, Bob has a point $s \in F_p$
- Bob wishes to learn E(P(s)); how?
- two naïve ways
  - Alice sends P to Bob; he computes E(P(s))
  - Bob sends s to Alice; she computes E(P(s)) and sends it back to Bob
- however, in blind evaluation problem,
  - we want Bob to learn E(P(s)) without learning P
    - d is order of millions in Zcash; sending P is too much overhead; recall succinct!
  - we don't want Alice to learn s (so-called blind evaluation)

Henceforth, Alice is prover and Bob is verifier

blind evaluation of a polynomial

- Using HH, we perform blind evaluation as follows
  1. Bob sends to Alice the hidings E(1), E(s), ..., E(s^d)
  2. Alice computes E(P(s)) from the linear combination of the elements in the 1st step, and sends E(P(s)) to Bob

- why do we need this?
  - verifier (Bob) has a correct polynomial in mind and wishes to check the prover knows it
  - making the prover (Alice) blindly evaluate the polynomial at a random point not known to prover
  - if the prover has the wrong polynomial, she will give the wrong answer

Schwartz-Zippel Lemma: different polynomials are different at most points

operation change in finite group

- from now on, we write the finite group additively rather than multiplicatively
- For $\alpha \in F_p$, we used to write g^{α} mod p
- Now we write $\alpha \cdot g$ mod p,
  - the result of summing α copies of g
  - if someone receives $\alpha \cdot g$, she cannot know α

- recall ECC

Knowledge of Coefficient (KC)

- Prover (Alice) can compute E(P(s)) but may not send E(P(s))
- how can we enforce the prover to send E(P(s))?

- KC test

for $\alpha \in F_p^*$, a pair of elements (a,b) in G is an α-pair if $b=\alpha \cdot a$

1. Bob chooses random $\alpha \in F_p^*$, $a \in G$; he computes $b=\alpha \cdot a$
2. He sends to Alice the "challenge" pair (a,b), which is an α-pair
3. Alice must respond with a different pair (a',b'), another α-pair
4. Bob accepts Alice's response only if (a',b') is an α-pair

Again, Alice is prover and Bob is verifier; only Bob knows α

How can Alice generate another α-pair?

- Alice knows only $\alpha \cdot a$, not α
  - since G is a group for DLP
- Alice chooses some $\gamma \in F_p^*$, and responds with $(a', b') = (\gamma a, \gamma b)$
  - $b' = \gamma b = \gamma\, \alpha\, a = \alpha\, a'$,
- Knowledge of Coefficient Assumption (KCA)
  - if she sends (a', b') in response to Bob's challenge (a,b), then she knows the ratio γ such that $a' = \gamma\, a$

Make Blind Evaluation Verifiable

- Want to construct a protocol that allows Bob to learn E(P(s)) with two additional properties
    1. blindness: Alice will not learn s (and Bob will not learn P)
    2. Verifiability: the probability that Alice sends a value not E(P(s)), but Bob still accepts is negligible

An extended KCA

- Bob sends Alice several α-pairs $(a_1, b_1),\ldots,(a_d, b_d)$ (for the same α)
- After receiving these pairs, Alice is challenged to generate another α-pair (a', b')
- Alice now takes a linear combination of the given d pairs

  $(a',b') = (\sum_{i=1}^{d} c_i a_i\,,\, \sum_{i=1}^{d} c_i b_i)$, where Alice chooses any $c_i \in F_p$
- The extended KCA states that this is the only way Alice can generates an α-pair; she knows a linear relation between a' and $a_1,\ldots,a_d$ – called d-power knowledge of coeff. assumption (d-KCA)
- d-KCA: Bob sends Alice (g, αg),(sg, αsg),…, $(s^d g, \alpha s^d g)$; then Alice outputs another α-pair (a',b')

→ Alice knows $c_0, c_1,\ldots,c_d \in F_p$ s.t. $\sum_{i=0}^{d} c_i s^i g = a'$ $\left(and\ \sum_{i=0}^{d} \alpha c_i s^i g = b'\right)$

Verifiable Blind Evaluation Protocol

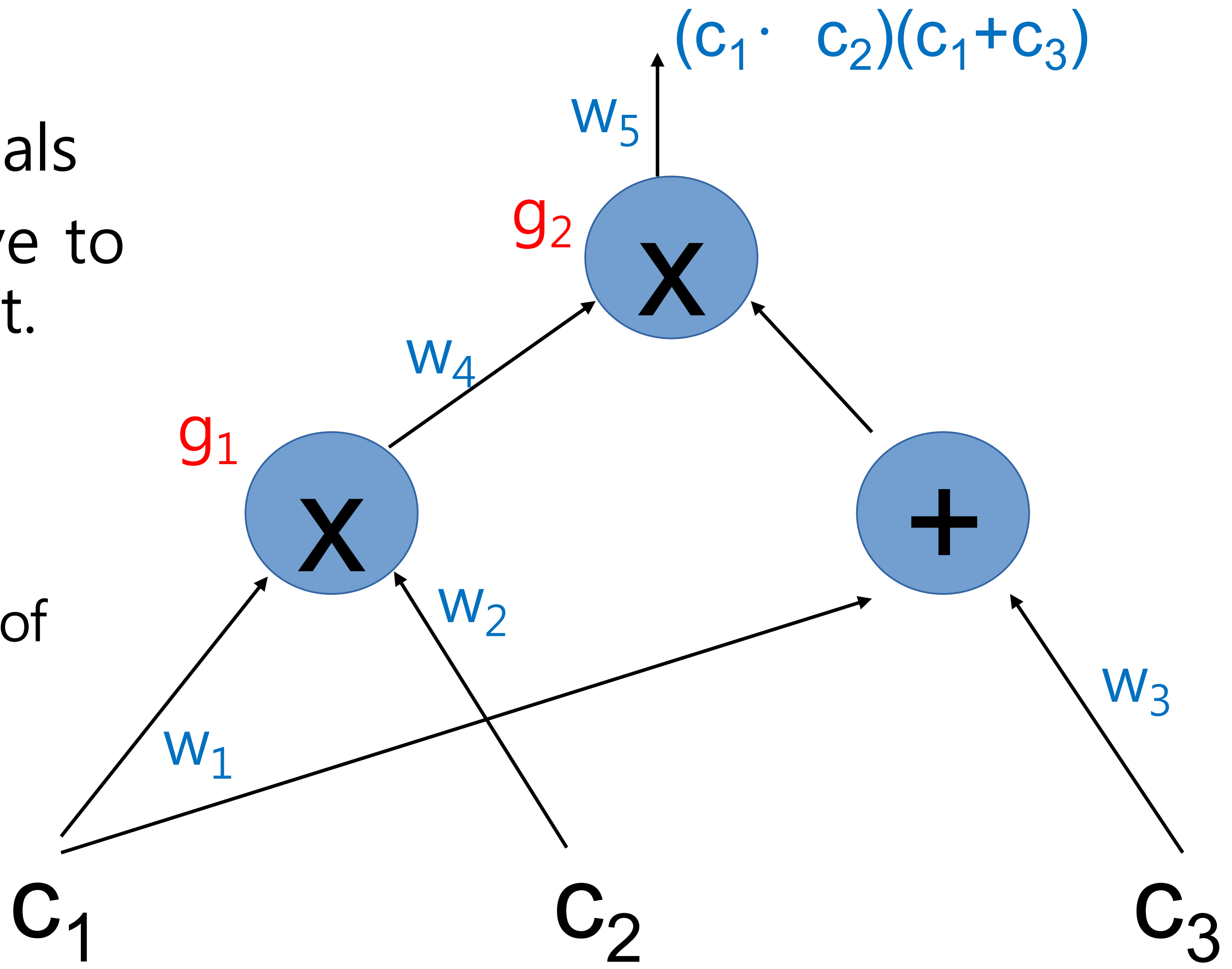
- HH is the mapping E(x) = x·g for generator g of G
- We present the protocol for this E(x)

1. Bob chooses a random $\alpha \in F_p^*$, and sends Alice the following
    - the hidings of $1, s^1, s^2, \ldots, s^d$, which are $g, s \cdot g, \ldots, s^d \cdot g$
    - the hidings of $\alpha, \alpha \cdot s, \alpha \cdot s^2, \ldots, \alpha \cdot s^d$, which are $\alpha \cdot g$, $\alpha \cdot s \cdot g, \ldots, \alpha \cdot s^d \cdot g$
2. Alice computes $a = P(s) \cdot g$ and $b = \alpha \cdot P(s) \cdot g$, which are sent to Bob
3. Bob checks that $b = \alpha \cdot a$, and accepts iff this equality holds

- $P(s) \cdot g$ is a linear combination of $g, s \cdot g, \ldots, s^d \cdot g$, which is E(P(s))
- $\alpha \cdot P(s) \cdot g$ is a linear combination of $\alpha \cdot g$, $\alpha \cdot s \cdot g, \ldots, \alpha \cdot s^d \cdot g$
- by d-KCA, if Alice sends a,b s.t. $b = \alpha \cdot a$, then she knows $c_0, c_1, \ldots, c_d \in F_p$ s.t. $a = \sum_{i=0}^{d} c_i s^i g$

Quadratic Arithmetic Program (QAP)

- QAP: translation of computations into polynomials
- suppose Alice wants to prove to Bob she knows $c_1, c_2, c_3 \in F_p$ s.t. $(c_1 \cdot c_2)(c_1 + c_3) = 7$
- 1^{st} step: expression to arithmetic circuit
  - An arithmetic circuit consists of gates computing arithmetic operations like addition and multiplication, with wires connecting the gates



constructing an arithmetic circuit

- bottom wires are the input, and the top wire is the output
- When the same outgoing wire goes into more than one gate, we still think of it as one wire – like w_1 in the example.
- We assume multiplication gates have exactly two input wires, which we call the left wire and right wire
- We don't label the wires going from an addition to a multiplication gate, nor the addition gate; we think of the inputs of the addition gate as going directly into the multiplication gate. So in the example we think of w_1 and w_3 as both being right inputs of g_2.
- A legal assignment for the circuit, is an assignment of values to the labeled wires where the output value of each multiplication gate is indeed the product of the corresponding inputs.
  - $c_4 = c_1 \cdot c_2$ and $c_5 = c_4 \cdot (c_1+c_3)$

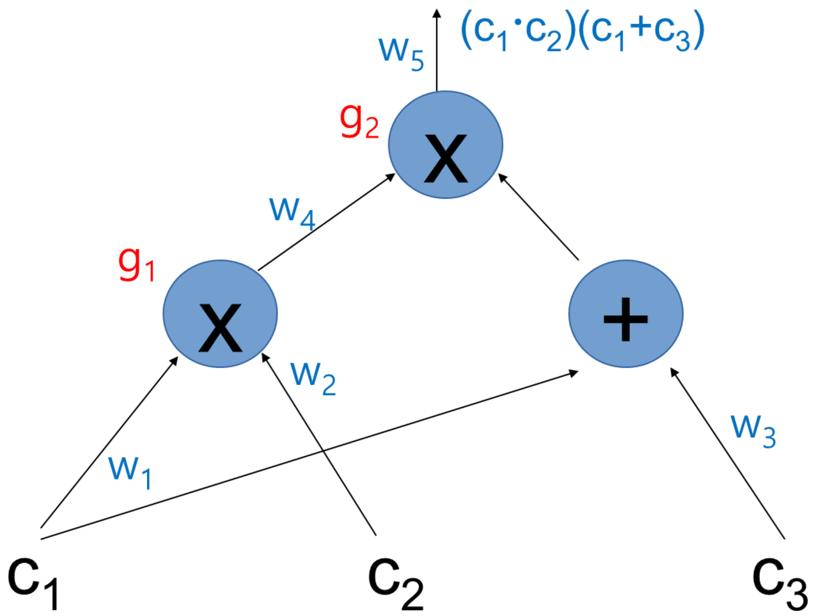
c_4 is for w_4; c_5 is for w_5

- what Alice wants to prove is that she knows a legal assignment $(c_1,\ldots,c_5)$ such that $c_5=7$

reduction to a QAP

- We associate each (label of) multiplication gate with a field element
  - g_1 will be associated with $1 \in F_p$ and g_2 with $2 \in F_p$
- We call the points {1,2} our target points. Now we need to define a set of "left wire polynomials" $L_1,\ldots,L_5$, "right wire polynomials" $R_1,\ldots,R_5$ and "output wire polynomials" $O_1,\ldots,O_5$.
- the polynomials will usually be zero on the target points
  - they will be ones at the target point's corresponding multiplication gate.

reduction to a QAP: an example



- w_1, w_2, w_4 are the left, right and output wire of g_1
- we define $L_1=R_2=O_4=2-X$ as the polynomial $2-X$ is one on the point 1 corresponding to g_1 and zero on the point 2 corresponding to g_2
- w_1 and w_3 are *both* right inputs of g_2. Therefore, we define similarly $L_4=R_1=R_3=O_5=X-1$ as $X-1$ is one on the target point 2 corresponding to g_2 and zero on the other target point
- We set the rest of the polynomials to be the zero polynomial
- Thus, $\mathrm{L} = \sum_{i=1}^{5} c_i L_i$, $\mathrm{R} = \sum_{i=1}^{5} c_i R_i$, $\mathrm{O} = \sum_{i=1}^{5} c_i O_i$
- then we define the polynomial $P=L\cdot R-O$
- $(c_1,\ldots,c_5)$ is a legal assignment to the circuit iff P vanishes on all the target points.

illustration of the QAP reduction

- $L(1) = c_1 \cdot \quad L_1(1) = c_1$; $R(1) = c_2$; $O(1) = c_4$
- $P(1) = c_1 \cdot c_2 - c_4$
- $P(2) = c_4 (c_1+c_3) - c_5$
- P vanishes on the target points if $(c_1,\ldots,c_5)$ is a legal assignment
- For a polynomial P and a point $a \in F_p$, we have $P(a) = 0$ iff the polynomial $(X-a)$ divides P
  - $P = (X-a) \cdot H$ for some polynomial H
- Define a target polynomial $T(X) = (X-1)(X-2)$
  - T divides P iff $(c_1,\ldots,c_5)$ is a legal assignment

$P = L \cdot R - O$

QAP summary

- A Quadratic Arithmetic Program Q of degree d and size m consists of polynomials, $L_1,\ldots,L_m$, $R_1,\ldots,R_m$, $O_1,\ldots,O_m$. and a target polynomial T of degree d
- As assignment $(c_1,\ldots,c_m)$ satisfies Q if, defining $\mathrm{L} = \sum_{i=1}^{m} c_i L_i$, $\mathrm{R} = \sum_{i=1}^{m} c_i R_i$, $\mathrm{O} = \sum_{i=1}^{m} c_i O_i$, and $P = L \cdot R - O$, we have that T divides P
- Alice want's to prove that "I know c_1,c_2,c_3 s.t. $(c_1 \cdot c_2) \cdot (c_1+c_3)=7$" can be translated into an equivalent statement about polynomials using QAPs

Background before Pinocchio Protocol

- Alice can send a very short proof to Bob showing she has a satisfying assignment to a QAP
- If Alice know the legal assignment, there exists a polynomial H such that $P=H\cdot T$
  - in particular $s \in F_p$, $P(s) = H(s)\cdot T(s)$
- if Alice *doesn't* have a satisfying assignment, but she still constructs L,R,O,P as above from some unsatisfying assignment $(c_1,\ldots,c_m)$.
  - Then we are guaranteed that T does not divide P
  - if p is much larger than 2d, the prob. that $P(s)=H(s)\cdot T(s)$ for a randomly chosen $s \in F_p$ is very small

Schwartz-Zippel Lemma: different polynomials are different at most points

two different polynomials of degree at most 2d can agree on at most 2d points, $s \in F_p$

Pinocchio Protocol

- sketch of proving Alice has a satisfying assignment
  1. Alice chooses polynomials L,R,O,H of degree at most d
  2. Bob chooses a random point $s \in F_p$, and computes E(T(s)).
  3. Alice sends Bob the hidings of all these polynomials evaluated at s, i.e. E(L(s)), E(R(s)), E(O(s)), E(H(s))
  4. Bob checks if the desired equation holds at s. That is, he checks whether $E(L(s) \cdot R(s) - O(s)) = E(T(s) \cdot H(s))$

bilinear pairing

a non-interactive evaluation protocol

- setup: random F_r^*, s are chosen and the common reference string (CRS) is published
  - CRS: $E(1), E(s^1), E(s^2,), \ldots, E(s^d)$ and $E(\alpha), E(\alpha \cdot s), E(\alpha \cdot s^2), \ldots, E(\alpha \cdot s^d)$
- Proof: Alice computes $a = E(P(s))$ and $b = E(\alpha P(s))$ using the CRS