zk-SNARKS Cheatsheet v0.9

Erik Brauer
Iomete Labs
erik@iometelabs.io

Definitions

General

Properties of ZKPs[9]

1. Completeness: Given a statement and a witness, the prover can convince the verifier.
2. Soundness: A malicious prover cannot convince the verifier of a false statement.
3. Zero-Knowledge: The proof does not reveal anything but the truth of the statement, i.e. it does not reveal the provers witness.

Additional for SNARKS: Succinctness, Non-interactiveness

Set of p-adic integers

$$\mathbb{Z}_p := \{\sum_{i=0}^{\infty} a_i p^i | a_i \in \{0, 1, ..., p-1\}\},\ p \in \mathbb{P}$$

$\mathbb{Z}_p^*$ Group[1]

cyclic $\Leftrightarrow \exists g \in \mathbb{Z}_p^*$ s.t. $\mathbb{Z}_p^* = \{g^a | a \in \{0, ..., p-2\}; g^0 = 1\}$
discrete logarithm problem (DLP) believed to be hard in $\mathbb{Z}_p^*$
group operation: $g^a \cdot g^b = g^{a+b(mod\ p-1)}\ \forall a, b \in \mathbb{Z}_{p-1}$

$(\mathbb{F}_p, +, \cdot)$ Field

$\mathbb{F}_p = \{0, ..., p-1\}$
multiplication and addition over the field are also done $mod(p)$

Kleene star

Given a set S:
$S^* = \bigcup_{i \geq 0} S^i = S^0 \cup S^1 \cup S^2 \cup ...$
e.g.: $\{a, b, c\}^* = \{\epsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, ...\}$

Pairings

G_1,G_2,G_T groups and $|G_T| = p$
$P \in G_1$, $Q \in G_2$ generators of G_1 and G_2
$e : G_1 \times G_2 \to G_T$ with the following properties:

1. Bilinearity: $\forall a, b \in \mathbb{Z} : e(aP, bQ) = e(P, Q)^{ab}$
2. Non-Degeneracy: $e(P, Q) \neq 1$
3. Efficient Computability

Symmetric iff $G_1 = G_2 = G$
Commutative iff G cyclic: $e(P, Q) = e(Q, P)$

Elliptic-Curve Cryptography[1, 4, 6]

Define elliptic curve $\mathcal{C} = \{(x, y)|y^2 = x^3 + ax + b$ for some $a, b \in \mathbb{F}_p\}$
Group $\mathcal{C}(\mathbb{F}_p) = \{(x, y)|(x, y) \in \mathbb{F}_p^2$ are on $\mathcal{C}\}$ with $e = \mathcal{O}$
$+$ rule: $P + Q + R = \mathcal{O} \Rightarrow P + Q = -R$
(mirror intersection point of line passed by $(P, Q) \in \mathcal{C}(\mathbb{F}_p)$)
$|\mathcal{C}(\mathbb{F}_p)| = r,\ r \neq p \in \mathbb{P}$
embedding degree of $\mathcal{C}$: smallest int k s.t. $(p^k - 1 \bmod r) = 0$
DLP for sufficiently large k is "very hard"

Tate Reduced Pairings

Define subgroup G_T of multiplicative group $\mathcal{C}(\mathbb{F}_{p^k})$ with order r
$\mathcal{C}(\mathbb{F}_{p^k})$ contains $r - 1$ additional subgroups of order r
Generators $g \in G_1, h \in G_2$

1. Tate$(g, h) = \mathbf{g}$ for a generator $\mathbf{g}$ of G_T
2. Given a pair $(a.b) \in \mathbb{F}_r$, Tate$(a \cdot g, b \cdot h) = g^{a \cdot b}$

Homomorphic Hidings[1]

E(x) over $\mathbb{Z}_p^*$ with following properties:

1. trapdoor function, for given E(x) "hard" to find x
2. Collision-resistance: $x \neq y \Rightarrow E(x) \neq E(y)\ \forall x, y$
3. homomorphic, algebraic structure-preserving mapping

Common Reference String Model[1, 15]

Setup phase: CRS generated according to randomized process ("toxic waste" λ, has to be destroyed after)
CRS broadcast to all parties, used to construct and verify proofs divided into "proving key" and "verification key"

Polynomial Interpolation

Lagrange: Given a set of $n+1$ points $(x_0, f_0), ..., (x_i, f_i), ..., (x_n, f_n)$

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} = \frac{x - x_j}{x_i - x_j},\ L(x) = \sum_{i=0}^{n} f_i \ell_i(x)$$

$L(x)$ is the resulting interpolation polynomial in the Lagrange form

Merkle Trees

expansion and extension of hash lists
every leaf node labelled with hash of a data block
nodes further up are hashes of their respective children

Pedersen Hash Function[16, 17]

$P_0, ..., P_k$ generators of $\mathbb{G} = \{P \in E(\mathbb{F}_p)|rP = O\}$
Hash $M = M_0 M_1 ... M_k$
$H(M) = \langle M_0 \rangle \cdot P_0 + ... + \langle M_k \rangle \cdot P_k$

MiMC-p/p[18]

Encryption function: $E_k(x) = (F_{r-1} \circ F_{r-2} \circ ... \circ F_0)(x) \oplus k$
$x \in \mathbb{F}_p$ plaintext, $r = \frac{\log(p)}{\log_2(3)}$ number of rounds, $k \in \mathbb{F}_p$ key
round functions over $\mathbb{F}_p$: $F_i(x) = (x \oplus k \oplus c_i)^3$
c_i random round constants in $\mathbb{F}_p$, $c_0 = 0$

Perpetual Powers of Tau Ceremony[14]

Multi-Party Computation for Trusted Setup Phase
(Parties jointly compute CRS without leaking inputs)
no limit to number of rounds/contributions of participants

Construction from QAP

General[8]

1. Generation algorithm (trusted setup): $\text{Gen}(1^\lambda, \text{C}) \to (\text{crs,vrs})$
(secret parameter λ, circuit C, proving and verification key (crs,vrs))
2. Prover: $\text{Prove}(\text{crs}, u, w) \to \pi$
(some statement u, witness w, proof π)
3. Verifier: $\text{Ver}(\text{vrs}, u, \pi) \to 0/1$

QAP Conversion[1, 3, 12]

Code Flattening

Convert original code to arithmetic circuit:
$x = y$, (y can be variable or number)
$x = y(\text{op})z$, $\text{op} \in (+, -, \cdot, /)$
(y and z can be variables, numbers or sub-expressions)

Rank-1 Constraint System

Convert flattened code to constraint system (R1CS):
$\langle \underline{l_i}, \underline{s} \rangle \cdot \langle \underline{r_i}, \underline{s} \rangle - \langle \underline{o_i}, \underline{s} \rangle = 0 \, \forall i$
($\underline{x}$ denotes a Vector in Tensor notation)
ensures that prover provides valid values for the circuit

Quadratic Arithmetic Program

Express R1CS as QAP with Polynomial Interpolation:
QAP Q of degree d and size m consists of polynomials
$L_1, ..., L_m, R_1, ..., R_m, O_1, ..., O_m$ and a target polynomial T.
An assignment $(c_1, ..., c_m)$ satisfies Q if, defining
$P := L \cdot R - O$ for $L := \sum c_i \cdot L_i, R := \sum c_i \cdot R_i, O := \sum c_i \cdot O_i$
we have that T divides P.
Alice has a satisfying assignment iff $\exists H : P(s) = H(s) \cdot T(s) \, \forall s \in \mathbb{F}_p$

Blind Evaluation of Polynomials[1]

Alice has polynomial P, Bob has random point $s \in \mathbb{F}_p$:
1. Bob sends to Alice the hidings $E(1), E(s), ..., E(s^d)$
2. Alice computes $E(P(s))$ and sends the result to Bob

Conclusion: Neither Alice learned s, nor Bob learned P (Blindness)
Verifier (B) able to check if prover (A) knows the correct polynomial

Verifiable BEP

1. B chooses random $\alpha \in \mathbb{F}_p^*$ and sends to A the hidings
$E(1), E(s), ..., E(s^d)$ and $E(\alpha), E(\alpha s), ..., E(\alpha s^d)$
2. A computes $a = E(P(s))$ and $b = E(\alpha P(s))$, sends both to B
3. B accepts iff $b = \alpha \cdot a$

Knowledge of Coefficient Test[1, 11]

Alternatively Knowledge of Exponent (KEA)
Let $\alpha \in \mathbb{F}_p^*$ and α-pair $\Leftrightarrow a, b \neq 0 \, \wedge \, b = \alpha \cdot a \, \forall (a, b) \in G$
1. Bob chooses random $\alpha \in \mathbb{F}_p^*$ and $a \in G$. He computes $b = \alpha \cdot a$.
2. He sends to Alice the challenge pair (a, b).
3. Alice must now respond with a different α-pair (a', b').
4. Bob accepts Alice's response iff (a', b') is an α-pair.

KC Assumption

Alice chooses some $\gamma \in \mathbb{F}_p^*$ and responds with $(a', b') = (\gamma \cdot a, \gamma \cdot b)$
Whenever Alice successfully responds with an α-pair (a', b'),
Alice's Extractor outputs γ s.t. $a' = \gamma \cdot a$.

d-KCA

Bob chooses random $\alpha \in \mathbb{F}_p^*$, $s \in \mathbb{F}_p$, sends to Alice the α-pairs
$(g, \alpha \cdot g), (s \cdot g, \alpha s \cdot g), ..., (s^d \cdot g, \alpha s^d \cdot g)$. A outputs an α-pair (a', b')
$\Leftrightarrow$ A knows $c_1, ..., c_d$ s.t. $\sum_{i=1}^{d} c_i \cdot a_i$

Pinocchio Protocol (PHGR13)[1, 13]

1. Alice chooses L, R, O, H of degree at most d.
2. Bob chooses a random point $s \in \mathbb{F}_p$, computes $E(T(s))$.
3. Alice sends Bob $E(L(s)), E(R(s)), E(O(s)), E(H(s))$.
4. Bob checks whether $E(L(s) \cdot R(s) - O(s)) = E(T(s) \cdot H(s))$.

Note: An extended version of KCA is used to make sure Alice chooses the polynomials produced from an assignment.
Alice conceals her assignment by adding a random T-shift to each polynomial. ("free" zero-knowledge)

Non-Interactive Evaluation Protocol[1]

1. Setup: Random $\alpha \in \mathbb{F}_r^*, s \in \mathbb{F}_r$ are chosen and CRS is published:
$(E_1(1), E_1(s), ..., E_1(s^d), E_2(\alpha), E_2(\alpha s), ..., E_2(\alpha s^d))$
2. Proof: Alice computes $a = E_1(P(s))$ and $b = E_2(\alpha P(s))$
3. Verification: Fix $x, y \in \mathbb{F}_r$ s.t. $a = E_1(x)$ and $b = E_2(y)$

Bob computes $E(\alpha x) = \text{Tate}(E_1(x), E_2(\alpha))$ and
$E(y) = \text{Tate}(E_1(1), E_2(y))$ and checks if $E(\alpha x) = E(y)$.

Groth16[9]

smaller proofs, faster proving and verification time compared to Pinocchio (PHGR13)
comparison between Groth16 and PHGR13:

Protocols	CRS size	Proof size	Verification time
PHGR13	linear circuit	7 G_1, 1 G_2	12 pairings
Groth16	size	2 G_1, 1 G_2	3 pairings

References and further reading

[1] Gabizon, A.: *Explaining Snarks* (Parts I-VII) [https://electriccoin.co/blog/snark-explain/]

[2] Matter Labs: *Awesome zero knowledge proofs* [https://github.com/matter-labs/awesome-zero-knowledge-proofs]

[3] Buterin, V.: *Quadratic Arithmetic Programs: from Zero to Hero* [https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649]

[4] Buterin, V.: *Exploring Elliptic Curve Pairings* [https://medium.com/@VitalikButerin/exploring-elliptic-curve-pairings-c73c1864e627]

[5] Buterin, V.: *Zk-SNARKs: Under the Hood* [https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6]

[6] Sullivan, N.: *A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography* [https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/]

[7] Petkus, M.: *Why and How zk-SNARK Works* [https://arxiv.org/abs/1906.07221]

[8] Nitulescu, A.: *zk-SNARKs: A Gentle Introduction* [https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf]

[9] Groth, J.: *On the Size of Pairing-based Non-interactive Arguments* [https://eprint.iacr.org/2016/260.pdf]

[10] Ben-Sasson, E. et al.: *Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture* [https://eprint.iacr.org/2013/879.pdf]

[11] Groth, J.: *Short Pairing-based Non-interactive Zero-Knowledge Arguments* [http://www0.cs.ucl.ac.uk/staff/J.Groth/ShortNIZK.pdf]

[12] Gennaro, R. et al.: *Quadratic Span Programs and Succinct NIZKs without PCPs* [https://eprint.iacr.org/2012/215.pdf]

[13] Parno, B. et al.: *Pinocchio: Nearly Practical Verifiable Computation* [https://eprint.iacr.org/2013/279.pdf]

[14] Wei Jie, K.: *Announcing the Perpetual Powers of Tau Ceremony to benefit all zk-SNARK projects* [https://medium.com/coinmonks/announcing-the-perpetual-powers-of-tau-ceremony-to-benefit-all-zk-snark-projects-c3da86af8377]

[15] Bottinelli, P.: *Security Considerations of zk-SNARK Parameter Multi-Party Computation* [https://research.nccgroup.com/2020/06/24/security-considerations-of-zk-snark-parameter-multi-party-computation/]

[16] Hopwood, D. et al.: *Zcash Protocol Specification* [https://github.com/zcash/zips/blob/master/protocol/sapling.pdf]

[17] Baylina, J. et al.: *4-bit Window Pedersen Hash On The Baby Jubjub Elliptic Curve* [https://iden3-docs.readthedocs.io/en/latest/iden3_repos/research/publications/zkproof-standards-workshop-2/pedersen-hash/pedersen.html]

[18] Albrecht, M. et al.: *MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity* [https://eprint.iacr.org/2016/492.pdf]