Author: 0xParc
Type: #source #class
Link: https://zkiap.com/#34e5b6cf6e1d4dd3901940d4be2edb0b
Topics: cryptography Computer Science

Session 1: Introduction to ZK

class

Zero Knowledge Cryptography abides by three rules:

- Zero Knowledge: Prover's responses don't reveal underlying information; The verifier learns nothing from the interaction and could have actually simulated the whole thing by himself.
- Completeness: If the prover knows the underlying information, he'll always be able to answer satisfactorily;
- Soundness: If the prover doesn't know the underlying information, he'll eventually get caught.

Zero knowledge cryptography has been around for a while, see private-public key pair scheme whereby anyone can prove ownership of a public address with a private key without ever showing it. Nonetheless, the innovation relies on succinctly proving generic functions : zkSNARKs

- Zk : Zero Knowledge, hidden inputs
- s: succinct, for the size of the proof, allowing for quick verification
- n: non-iterative, as in submit a proof to the verifier who can attest to its correctness without two way messaging
- ARK: Argument of Knowledge, prove that you know the inputs

Exercise

Exercise 1: It wouldn't be because the proof wouldn't allow us to verify that adjacent edges have different colors
Exercise 2: This follows a bernoulli process whereby each iteration is independent, which is correct as the colors get "re-shuffled" at each step. Hence, there are indeed no priors in the formula. A good way to visualize it is using as inputs 3 edges with 1 trial, the confidence being two-thirds, as in the two-edges we've seen but lacking the remaining

edge which could share colors with any of the two other edges (assuming they share a node)

zkmessage.xyz

- This secret value, a hash, can serve as a unique, private identifier for my account. It allows for proof generation, by functioning like a private key.
- The statement being proven is the one that I possess the account writing this message, without showing which account it is.
- Because one could make probabilistic assumptions and fall back to my account for certain messages.

Session 2: Circom 1

Using https://zkrepl.dev/ to write our first circuits.

Initially, the goal is to prove that we know the inputs of a very simple equation, and then it is to show that we know the bitwise representation of an integer.

The most important element in that course is learning to frame circuit constructions. Some synthetic sugars are explained, such as the ⇐ operator that should be used at all time when assigning witnesses, as it also allows to assert for correctness. Important is also the quadrant split between public / private inputs / outputs and how to visualize these differences with dedicated syntax.

Exercises main takeaway:

- Main point of circom is to build protocols around constraints. When implementing LessThan(n) for example, the goal is not only to return yes or no, but to validate during the process that there is no way the machine can be mistaken / cheated on. That is by far the most important point, as some exercise might seem easy, but once one remember that the theater of operation is a large prime field, and that the goal is to make sure that the whole set of constraints is respected, it becomes harder. The exercises are actually easier for people well versed in math and less in code than the opposite.

Session 3: Mathematical Building Blocks

1. ZK proofs and zksnarks primer:
   As discussed, zk proofs observe three properties: completeness, soundness and zero-knowledge. zk proofs are not new, and some papers from before our millennium

discuss implementations of zk proofs. In this class, specifically, is discussed the issue of proving that I know an Hamiltonian circuit without showing it.

What is new comes from what constitutes a zksnark:

- non-iterative: Fiat-Shamir heuristic which made it possible;
- succinct: a proof has to be lightweight

sigma Protocol : commitment → Challenge → Response

2. Elliptic curves:
   Of the form y^2 = x^3 + Ax + B for example
   In computer science, we don't really speak about numbers in the Real universe (since we're aware of 1s and 0s). Hence, the points are over Mod p ⟹ Fp which is equivalent to integer mod P

Group Law: Allows you to add points on the elliptic curve.

g for generator means that if we start with g we will get all the elements as below:
(g, 2g, 3g) ...

Discret Log Assumption: If I give you a point that belongs to the elliptic curve generated by G, with my secret x that belongs to Zq ⟹ you can't recover x

Schnorr protocol:

- P wants to convince V that:
  - P knows the secret s element of Z mod q such that x = sg (element of the group, public information); basically this public key is mine (i.e. I know the private key)

Necessary reminder of basics for the exercises:

- Modular Arithmetic is simple, but divisions are ill defined compared to the real number group, the solution is as follows.
  - x is divisible by y, iff y has an inverse. y has an inverse if GCD(y,n) = 1 with y belonging to Zn. As one can see, that leads us to Euclids's algorithm.
  - Then, if GCD(y,n) = 1, once can use extended Euclid's algorithm to find the inverse of y, which is just a specific case of division that extends to all divisions assuming the GCD equality is observed.
  - Chinese remainder theorem implies bijection between two sets of fields, allows us to break a field into sub-components to perform computations more efficiently (or better said, cheaply)
    - step 1: get N which is product of all modulo

- get Mi, which is N/moduloi
- Get yi = inverse of each Mi mod modulo i
- sum the product of each bi yi mi

Example 5. Use the Chinese Remainder Theorem to find an x such that

$$x \equiv 2 \pmod 5$$
$$x \equiv 3 \pmod 7$$
$$x \equiv 10 \pmod{11}$$

Solution. Set $N = 5 \times 7 \times 11 = 385$. Following the notation of the theorem, we have $m_1 = N/5 = 77$, $m_2 = N/7 = 55$, and $m_3 = N/11 = 35$.
We now seek a multiplicative inverse for each m_i modulo n_i. First: $m_1 \equiv 77 \equiv 2 \pmod 5$, and hence an inverse to m_1 mod n_1 is $y_1 = 3$.
Second: $m_2 \equiv 55 \equiv 6 \pmod 7$, and hence an inverse to m_2 mod n_2 is $y_2 = 6$.
Third: $m_3 \equiv 35 \equiv 2 \pmod{11}$, and hence an inverse to m_3 mod n_3 is $y_3 = 6$.
Therefore, the theorem states that a solution takes the form:

$$x = y_1 b_1 m_1 + y_2 b_2 m_2 + y_3 b_3 m_3 = 3 \times 2 \times 77 + 6 \times 3 \times 55 + 6 \times 10 \times 35 = 3552.$$

Since we may take the solution modulo $N = 385$, we can reduce this to 87, since $2852 \equiv 87 \pmod{385}$.

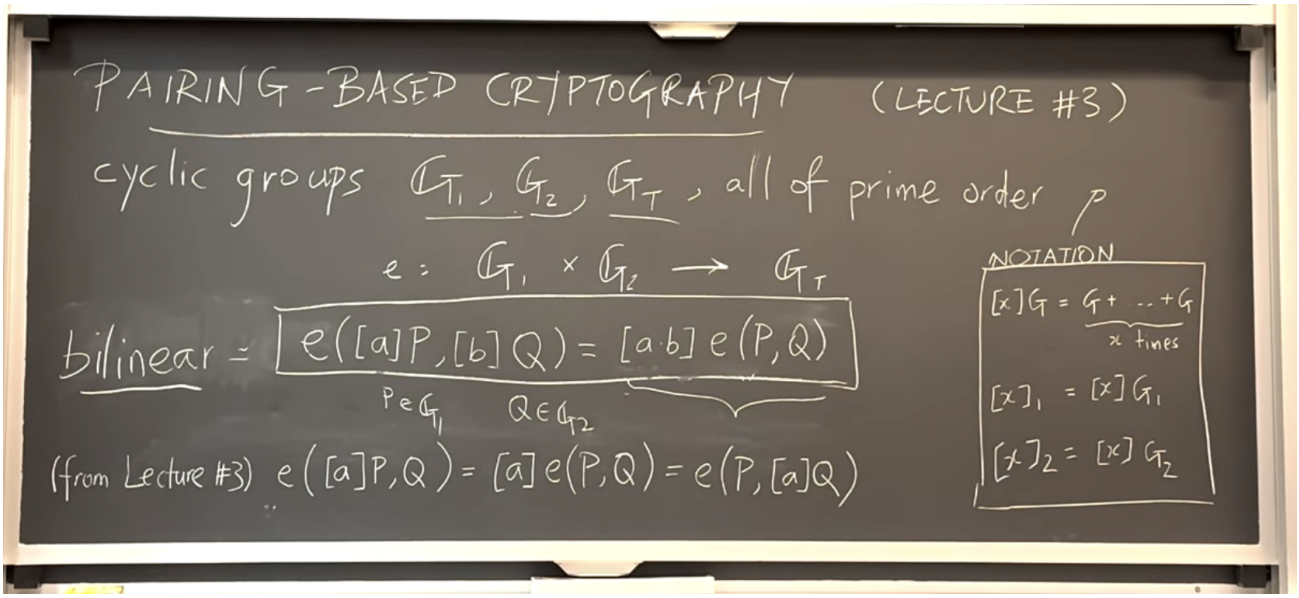
- the order of a unit: can be seen as 0:
  - can use Fermat's little theorem (for prime n) or Euler's
    - Fermat with p-1 to find the order of a unit, p-2 to find the inverse of the base
- about generators:

  A unit $g \in \mathbb{Z}_n^*$ is called a *generator* or *primitive root* of $\mathbb{Z}_n^*$ if for every $a \in \mathbb{Z}_n^*$ we have $g^k = a$ for some integer k. In other words, if we start with g, and keep multiplying by g eventually we see every element.
- if a multiplicative group (every element is coprime to n in Zn) has a generator, then it is a cyclic group. Can form subgroup by using a element of Zn and raising it to arbitrary power up to d, which will give its inverse, then the subset element restarts. All subgroup of a cylic group are cyclic. subset of a subgroup has size <= main group

Exercise on Self-Pairing (with the help of Alec | Magic Eden)

- We first take the hints and compute pairings against g:
  - g_t = e(g, g)
  -for the next few operation, we use the facts that the pairing is bilinear
  - e(alpha g, g) = alpha e(g,g) = alpha g_t
  - e(beta g, g) = alpha e(g,g) = beta g_t
  - e(alpha g, beta g) = alpha beta e(g,g) = alpha beta g_t
  - Let's suppose that y = gamma g, with gamma guaranted to exists because g is a generator of G

-e(gamma g, g) = gamma e(g,g) = gamma g_t
Hence, it appears that we can use the expression e(alpha g, beta g) = e(y, g) to check wether alpha beta g = y



Session 5: Commitment Scheme

Random Class Notes:
- Cryptographic group : a group where the discrete log problem is assumed to hold / is hard in this group;
- $ means we uniformally select a field element r;
- Pedersen reveals m, r: breaks the zero knowledge assumption; but still bidding and hidding
merkle proof: the message was committed amongst the vector m of all committed message (so C = Commit(vector m) AND mi belongs to vector m);
- Polinomial Commitment Scheme:
Verifier evalues the polynomial at a commitment point z, hence should be convinced that f(x) =y AND that C = the committed polynomial f(X);

Formal definition of cryptographic commitment scheme, which are a compiled (i.e. usable, non-theoretical) version of Interactive Oracle Proof. The main difference relies on assumptions of unboundness in computation power, or the lack thereof.

1. Formal Definition of a commitment Scheme
   Tuple (Setup, Commit, Open) of PPt algorithms.
   a Commitment Scheme can be binding if no two different message will lead to same commit if they take the same public parameter
   A commitment Scheme can be hidding if there is no way to reverse engineer the committed message. Note again that this is in practical terms.

2. Constructions
   Vector Commitment Scheme

- Position Binding ⇒ reshuffling of vector will lead to different result to Open function (nice parallel with merkle tree proofs where leafs also have to be kept in order to generate same root hash)

Polynomial Construction Scheme:

- Leverage on math properties:
  - Setup: Create public paramaters commit key and private param verify key
  - Commit: commit polynomial f(x)
  - by checking that the roots exist at q(X), the opening function leverage math properties of polynomials (basically same as solving for roots as when we were in high school, but this time at a different point than 0, so instead of f(x) = 0, f(x) = y). polynomial is the one we committed, without declaring its form.
  Exercise 1:
  The trick here is that if we know the secret value, we can evaluate the polynomial at a point that will equal y by simply passing the secret value into the function. That will cancel out both side which will be equal to one with the exponent being y-y

Lecture 6: Algorithms for Efficient Cryptographic Operations

[n] p means n times p or p^n, depending on the group

binary expansion: writing the number in base b

- Fast Fourier Transform (FFT): An algorithm to efficiently multiply polynomials, used in cryptographic schemes like lattice-based cryptography
- Montgomery Reduction: A method for efficient modular arithmetic, often used in public-key cryptography algorithms
- Pippenger's Algorithm: An algorithm for efficient scalar multiplication in elliptic curve cryptography

Lecture 7: Arithmetizations

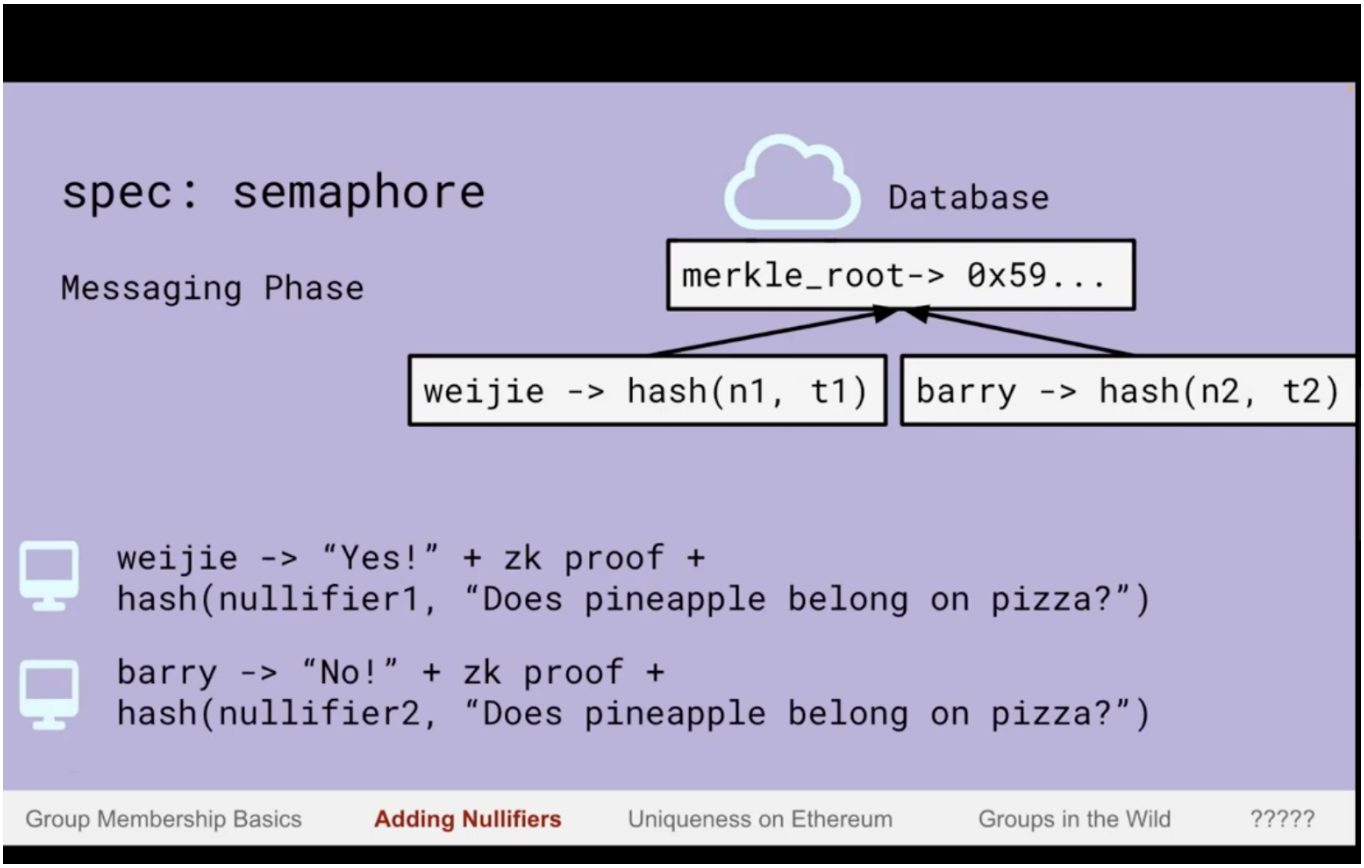
→ Rics to QAP, or how to reduce proving computations to a single polynomial identity

Lecture 8: PLONK and Polynomial Identities

Lecture 9: Proving Systems Stack; Recursion and Composition

Lecture 10: Applied ZK Constructions 1

Nullifiers: nullify ability of an user to do an action twice while keeping the user anon



Lecture 11: Applied ZK Constructions 2

Information Asymmetric vs Fully Transparent games