Lesson 6

From QAP to final proof

From our QAP we have

$$L := \sum_{i=1}^{m} ci \cdot Li, R := \sum_{i=1}^{m} ci \cdot Ri, O := \sum_{i=1}^{m} ci \cdot Oi$$

and we define the polynomial P
$P := L \cdot R - O$
Defining the target polynomial $T(X) := (X-1) \cdot (X-2)\ldots,$
This will be zero at the points that correspond to our gates, but the P polynomial, having all the contraints information would be a some multiple of this if

- it is also zero at those points
- to be zero at those points, $L \cdot R - O$ must equate to zero, which will only happen if our contraints are met.

So we want T to divide P with no remainder, which would show that P is indeed zero at the points.

If Peggy has a satisfying assignment it means that, defining L,R,O,P as above, there exists a polynomial H such that P=H⋅T. In particular, for any s∈Fp we have P(s)=H(s)⋅T(s).

Suppose now that Peggy doesn't have a satisfying witness, but she still constructs L, R, O, P as above from some unsatisfying assignment (c1,...,cm)(c1,...,cm).
Then we are guaranteed that T does not divide P.
This means that for any polynomial T of degree at most $d-2$, P and L, R, O, T will be different polynomials.
Note that P here is of degree at most $2(d-1)$, L, R, O here are of degree at most $d-1$ and T here is degree at most $d-2$.

Remember the Schwartz-Zippel Lemma tells us that two different polynomials of degree at most d can agree on at most d points.

HOMOMORPHIC HIDING REVIEW

If $E(x)$ is a function with the following properties

- Given $E(x)$ it is hard to find x
- Different inputs lead to different outputs so if $x \neq y$ $E(x) \neq E(y)$
- We can compute $E(x+y)$ given $E(x)$ and $E(y)$

The group $\mathbb{Z}_p^*$ with operations addition and multiplication allows this.

Here's a toy example of why Homomorphic Hiding is useful for Zero-Knowledge proofs: Suppose Alice wants to prove to Bob she knows numbers x,y such that $x + y = 7$

1. Alice sends $E(x)$ and $E(y)$ to Bob.
2. Bob computes $E(x+y)$ from these values (which he is able to do since E is an HH).
3. Bob also computes $E(7)$, and now checks whether $E(x+y) = E(7)$. He accepts Alice's proof only if equality holds.

As different inputs are mapped by E to different hidings, Bob indeed accepts the proof only if Alice sent hidings of x, y such that $x + y = 7$. On the other hand, Bob does not learn x and y as he just has access to their hidings 2(https://electriccoin.co/blog/snark-explain/#id5).

Creating a non interactive proof and adding zero knowledge

BLIND EVALUATION OF A POLYNOMIAL USING HOMOMORPHIC HIDING

Suppose Peggy has a polynomial P of degree d , and Victor has a point $s \epsilon \mathbb{F}_p$that he chose randomly.
Victor wishes to learn E(P(s)), i.e., the Homomorphic Hiding of the evaluation of P at s
Two simple ways to do this are:

1. Peggy sends P to Victor, and he computes E(P(s))by himself.
2. Victor sends s to Peggy; she computes E(P(s)) and sends it to Victor.

However, in the blind evaluation problem we want Victor to learn E(P(s)) without learning P which precludes the first option; and, most importantly,
we don't want Peggy to learn s, which rules out the second

Using HH, we can perform blind evaluation as follows.

1. Victor sends to Peggy the hidings E(1),E(s),...,E(sd)
2. Peggy computes E(P(s)) from the elements sent in the first step, and sends E(P(s)) to Victor. (Peggy can do this since E supports linear combinations, and P(s)is a linear combination of 1,s,...,sd.)

Note that, as only hidings were sent, neither Peggy learned s nor Victor learned P

The rough intuition is that the verifier has a "correct" polynomial in mind, and wishes to check the prover knows it. Making the prover blindly evaluate their polynomial at a random point not known to them, ensures the prover will give the wrong answer with high probability if their polynomial is not the correct one (Schwartz-Zippel Lemma).

but the fact that Peggy is able to compute E(P(s)) does not guarantee she will indeed send E(P(s)) to Victor, rather than some completely unrelated value.

Our process then becomes

1. Peggy chooses polynomials L,R,O,P, H
2. Victor chooses a random point s ∈ 𝔽p, and computes E(P(s))
3. Peggy sends Victor the hidings of all these polynomials evaluated at s, i.e. E(L(s)),E(R(s)),E(O(s)),E(P(s), E(H(s)))

Furthemore we use

- Random values added to our s to conceal the s value

- The Knowledge of Coefficient Assumption to prove Peggy can produce a linear combination of the polynomials.

If Peggy does not have a satisfying assignment, she will end up using polynomials where the equation does not hold identically, and thus does not hold at most choices of s Therefore, Victor will reject with high probability over his choice of s.

We now need to make our proof non interactive, for this we produce the Common Reference String from Victor's first message

Non-interactive proofs in the common reference string model

In the CRS model, before any proofs are constructed, there is a setup phase where a string is constructed according to a certain randomized process and broadcast to all parties. This string is called the CRS and is then used to help construct and verify proofs. The assumption is that the randomness used in the creation of the CRS is not known to any party – as knowledge of this randomness might enable constructing proofs of false claims.

Why do we need this randomness

Victor is sending challenges to Peggy, if Peggy could know what exactly the challenge is going to be, she could choose its randomness in such a way that it could satisfy the challenge, even if she did not know the correct solution for the instance (that is, faking the proof).

So, Victor must only issue the challenge after Peggy has already fixed her randomness. This is why Peggy first *commits* to her randomness, and implicitly reveals it only after the challenge, when she uses that value to compute the proof. That ensures two things:

1. Victor cannot *guess* what value Peggy committed to;
2. Peggy cannot *change* the value she committed to.

Comparison with Sigma Protocol

A Σ-protocol is a zero-knowledge proof as well, but with very different properties:

- unlike a ZK-SNARK, it is interactive (three moves)
- unlike a ZK-SNARK, it is not fully zero-knowledge (but only honest-verifier zero-knowledge)
- unlike a ZK-SNARK, it is not succinct

On the positive side, Σ-protocols exist unconditionally (ZK-SNARK require ultra strong assumptions) and do not need any trusted setup (ZK-SNARKs require a common reference string).

What are the drawbacks to zkSNARKS ?

1. The need for a trusted setup
2. Strong cryptographic assumptions are needed
3. The use of pairing to give homomorphic hiding. relies on a **Knowledge of Exponent** assumption, this is a relaitively new and little tested assumption.
4. Not post quantum secure