Proofs, Arguments and ZK
Chap 7
A first succinct argument for circuit SAT, from IP

1 Basis

- **Circuit**: C(x,w)=y
- **KNown**: x, y
- **Unknown**: w
- **MLE**:$\tilde{w}$

2 Knowledge soundness vs STandard soundness

Standard soundness ensures the existence of a witness w in C(x,w) = y, given x and y, whereas Knowledge soundness ensures not just the existance but that the prover knows such a w. Knowedge soundness is particularly useful. Take the case of a hash function h(w) = y, given y and the algorithm h(), there could be multiple w, as the hash function is a surjective function, no doubt it is called a compression function. But this is only telling us about the existence of w, not that the prover computing h(w) and sending it to the verifier knows an w.

Knowledge sound args can be particularly useful when they are NI , meaning the proof is just a static string that is accepetd or rehected by the verifier and succinct. Such args are calleld SNARK.

3 Naive approach

prover sends the witness w to verifier. Problems:

- w can be very large

4 A first succinct arg for CircuitSAT

4.1 Approach

- Trivial GKR but without sending w. How, you ask?

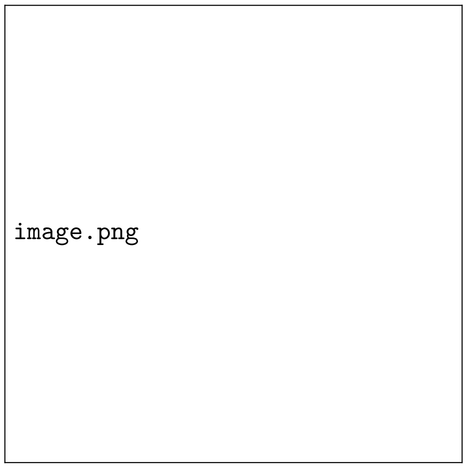

Figure 1: merkle tree

- Using PCS but not an actual one but a relaxed one with Merkle tree and Low degree test to emulate a PCS. It is impractical, mind you because of large porver runtime
- Commitment scheme -¿ satisfy hiding and binding.
- Poly Commit scheme -¿ Com scheme where the OBJECT BEING COMMITTED IS ALL EVALUATIONS OF f, a low degree polynomial (Multilinear Extension)

4.2 Details

PCS The scheme made using merkle tree and a low degree test (LDT) is not a genuine PCS becoz it only binds the prover to a function that is close to a polynomial.But as we will see, it is enough to transform a GKR protocol into a succinct arg for CircuitSAT.

4.2.1 Merkle tree

4.2.2 How is the polynomial committed using merkle tree?

Instead of a string commitment, the alphabets are replaced by all the evaluations of the polynomial. The prover on the verifier's inquisition, would give the point evaluation as well as the sibling of all the nodes at each level until the root.

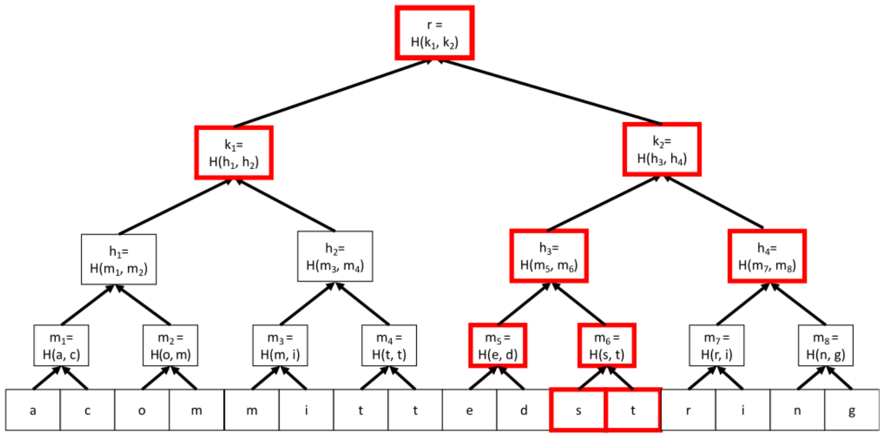

Figure 7.1: A Merkle-tree committing to the string "acommittedstring" using hash function H. Boxes with a red, bold outline represent the authentication path to reveal the twelfth entry of the committed string, namely the letter t. This consists of every node along the path from the root to the twelfth leaf, as well as each such node's sibling.

4.2.3 Does merkle tree give out a PCS?

Unfortunately, this approach does not directly yield a polynomial commitment scheme. The reason is that while the Merkle tree does bind the prover to a fixed string, there is no guarantee that the string is equal to all evaluations of some multilinear polynomial.

That is, when the verifier V asks P to reveal p(r) for some input r, the binding nature of Merkle trees does force P to respond with the r'th entry of the committed string and the associated authentication information. But V has no idea whether the committed string consists of all evaluations of a multilinear polynomial—the committed string could in general consists of all evaluations of some totally arbitrary function.

4.2.4 Need for a Low Degree test(LDT)

To address this issue, we combine Merkle trees with a low-degree test. The low-degree test ensures that not only is the prover bound to some (possibly completely unstructured) string, but actually that the string contains all evaluations of a low-degree polynomial. More precisely, it ensures that the string is "close" to the evaluation-table of a low-degree polynomial, so its use here yields a somewhat weaker object than an actual polynomial commitment scheme. The low-degree test guarantees this despite only inspecting a small number of entries of the string—often logarithmic in the length of the string—

4.2.5 MLE amplifying the distance

Low degree multilinear extension is used here becoz MLE makes two functions which differ at only a few points differ almost everywhere, provided the field is big enough.

4.2.6 Problems with LDT

It only looks at a few points, fraction of all the points. Then unless the test gets lucky and chooses the input on which s and p disagree, the test has no hope of distinguishing between s and p itself.98 What the low-degree test can guarantee, however, is that s is close in Hamming distance to (the string of all evaluations of) a low-degree polynomial. That is, if the test passes with probability , then there is a low-degree polynomial that agrees with s on close to a fraction of points.

4.2.7 Line vs Point test, an example of a LDT

In this test, one evaluates s along a randomly chosen line in Fm , and confirms that s restricted to this line is consistent with a univariate polynomial of degree at most m (see Section 4.5.2). Clearly, if the string s agrees perfectly with a multilinear polynomial then this test will always pass. The works [RS96,AS03] roughly show that if the test passes with probability , then there is a low-degree polynomial that agrees with s at close to a fraction of points.99 In this survey, we will not discuss how these results are proved.

5 Knowledge soundness: Extractable polynomial commitments

xtractability of a polynomial commitment scheme is a stronger property than mere binding. Roughly, extractability is to binding as knowledge-soundness is to standard soundness. binding of a polynomial commitment scheme guarantees that there is some polynomial p of the appropriate degree that “explains” all of the evaluations that the prover can open the commitment to, but a priori it is possible that the prover itself doesn’t actually know what that the polynomial is. Extractability guarantees that in fact the prover does know p.