**ZK Researcher + Engineer Role:**

**Question 2: ZK Implementation Challenge**

**● Section 1: Problem Definition**

**Given a specific equation, x^2 + x + 7, the prover wants to prove to the verifier that he knows a value of x, such that the equation resolves to 9, but without revealing the value of x. Write down all the sets of public and private inputs of the circuit.**

Design a circuit that represents the equation x2+x+7=9.

The circuit will take xxx as an input, compute x2+x+7,and output whether this equals the public target value (9).

**Public Input**: The target output (9) that the equation should resolve to.

**Private Input**: The secret value x, which the prover knows.

Using a ZKP protocol, like **STARK** or **Plonk**, the prover creates a **proof** that they know a valid input x without revealing x itself.

This proof includes cryptographic commitments to values (like x and intermediate results) without disclosing them directly.

The verifier checks the proof against the public input (9) and the mathematical structure of the ZK proof.

The proof assures the verifier that there exists an xxx satisfying x2+x+7=9 without revealing x.

**Section 2: ZK Protocol Selection**

**Choose a suitable ZK protocol (e.g., Plonk, Groth16, or Stark) for the given problem.**

I choose STARK ZK proof protocol.

**Justify your choice based on factors like efficiency, security, and ease of implementation.**

Reasons for Choosing STARK:

1. Transparency:

* STARKs are **transparent** and don’t require a trusted setup, unlike protocols like Groth16, which rely on a trusted setup phase (a process that can be complex and risky if not handled securely).
* In situations where privacy and security are critical, STARK’s transparency provides a strong advantage, ensuring there’s no dependency on external trusted parties.

1. Scalability:

* STARKs are designed to handle large computations efficiently, making them ideal for applications that may grow in complexity or require high transaction volumes.
* Although this specific equation is simple, using STARKs would make it easier to scale or generalize the proof to more complex equations or larger applications in the future.

1. Post-Quantum Security:

* STARKs use hash-based cryptographic primitives, which are more resistant to attacks by quantum computers, unlike elliptic curve-based systems like Groth16.
* This provides a future-proof solution for zero-knowledge proofs that would remain secure even in a post-quantum computing era.

**Section 3: Circuit Design**

**Design a ZK circuit to represent the computation or verification process.**

**Break down the circuit into smaller arithmetic circuits or lookup arguments.**

To design a zero-knowledge (ZK) circuit that proves knowledge of a value x satisfying the equation:

x2+x+7=9

without revealing x, we can break down the circuit into specific components. Here’s how to structure it:

**Define the Inputs**

* **Private Input**: The secret value x that satisfies the equation.
* **Public Input**: The target result, 9, which is known to both the prover and verifier.

**Circuit Components**

The circuit needs to compute and verify that x^2 + x + 7 = 9 without revealing x. We can break down the circuit into the following steps:

**Square Calculation (Compute x^2)**:

Create a **multiplication gate** to compute x^2.

Let y1= x × x, where y1​ holds the value of x^2.

**Linear Addition (Compute x^2 + x)**:

* + Create an **addition gate** to add x to x^2.
  + Let y2​=y1​+x, where y2​ holds the value of x2+x.

1. **Constant Addition (Compute x2+x+7)**:
   * Add the constant 7 to y2​.
   * Let y3​=y2​+7, where y3y\_3y3​ holds the value of x2+x+7.
2. **Equality Check**:
   * Use an **equality gate** to check that y3​=9 (the public target).
   * If y3​ equals 9, the circuit verifies that the prover knows a valid x that satisfies the equation.

**3. Summary of Circuit Structure**

* **Input Gates**: x (private) and target value 9 (public).
* **Intermediate Calculations**:
  + y1​=x × x
  + y2​=y1​+x
  + y3​=y2​+7
* **Output Gate**: Check that y3=9.

**Optimize the circuit for efficiency and minimize proof size.**

Using Lookup Tables in ZK Circuits

1. Constant Lookup:
   * Store fixed values like 7 and 9 in a lookup table for easy reference.
   * This avoids recalculating these constants repeatedly, making the circuit more efficient.
2. Square Lookup (Optional):
   * For cases where multiple values of xxx are checked, a lookup table with precomputed squares (like x2) can be used.
   * If we know the range of x, this helps us skip repeated multiplications of x×x, which is especially useful in high-performance ZK setups.

Benefits of Decomposing the Circuit and Using Lookup Tables

* Efficiency: Each computation step is isolated, which speeds up proof generation and allows for easier optimization.
* Modularity: Smaller circuits can be independently tested and verified, increasing reliability.
* Scalability: The setup can easily be expanded to handle more complex equations by adding more circuits.