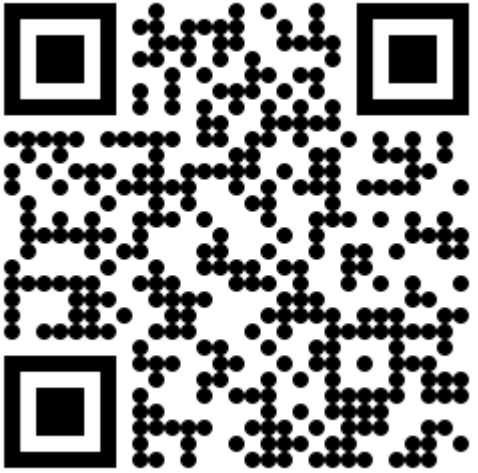
You can already start working!



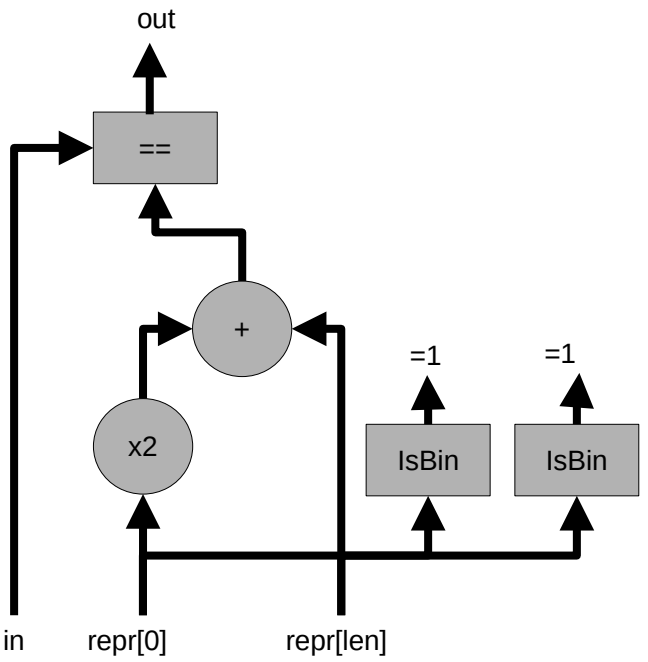
https://n.ethz.ch/~kklier/download/zkp/

Zero-Knowledge Proofs

Exercises Week 5: Circom/SnarkJS Part II

Last Time: Designing Circuits

```
template IsBinaryRepr(len) {
    signal input in;
    signal input repr[len];
    signal output out;

    component isBinary[len];

    for (var i = 0; i < len; i += 1){
        isBinary[i] = IsBinary();
        isBinary[i].in <== repr[i];
        isBinary[i].out === 1;
    }

    signal intermediate[len];
    intermediate[0] <== repr[0];

    for (var i = 1; i < len; i += 1){
        intermediate[i] <== 2*intermediate[i-1] + repr[i];
    }

    component isEq = IsEqual();
    isEq.in[0] <== intermediate[len-1];
    isEq.in[1] <== in;

    out <== isEq.out;
}
```

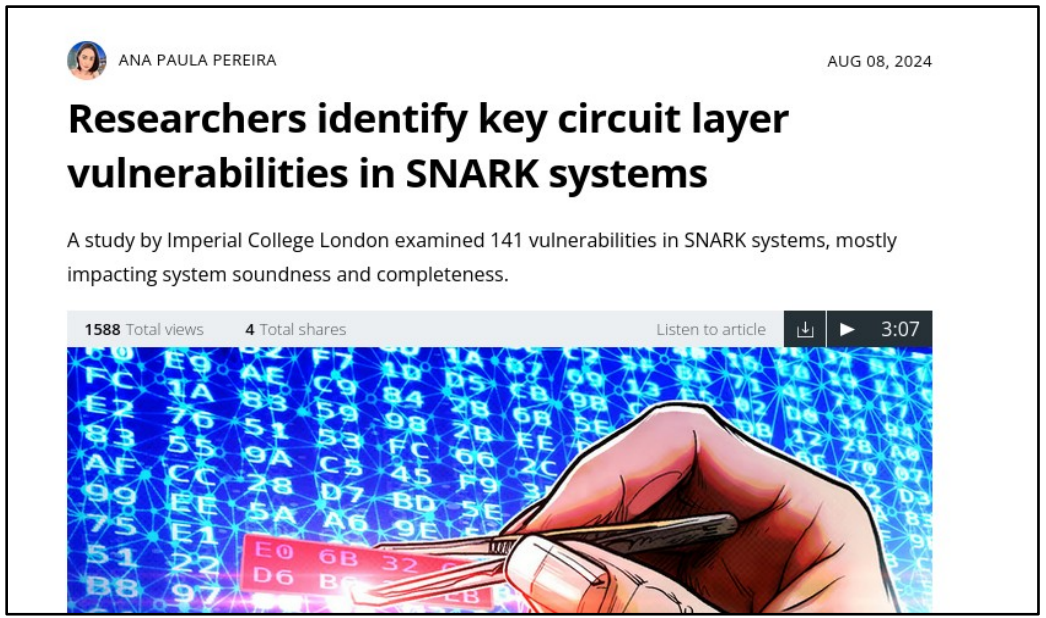


Circuit Validation

- SNARK → circuit does the thing right
- But does the circuit do the right thing?

Circuit Validation: Bugs in the Wild

https://cointelegraph.com/news/researchers-identify-key-circuit-layer-vulnerabilities-snark-systems



ANA PAULA PEREIRA AUG 08, 2024

Researchers identify key circuit layer vulnerabilities in SNARK systems

A study by Imperial College London examined 141 vulnerabilities in SNARK systems, mostly impacting system soundness and completeness.

1588 Total views 4 Total shares Listen to article 3:07

https://github.com/0xPARC/zk-bug-tracker:

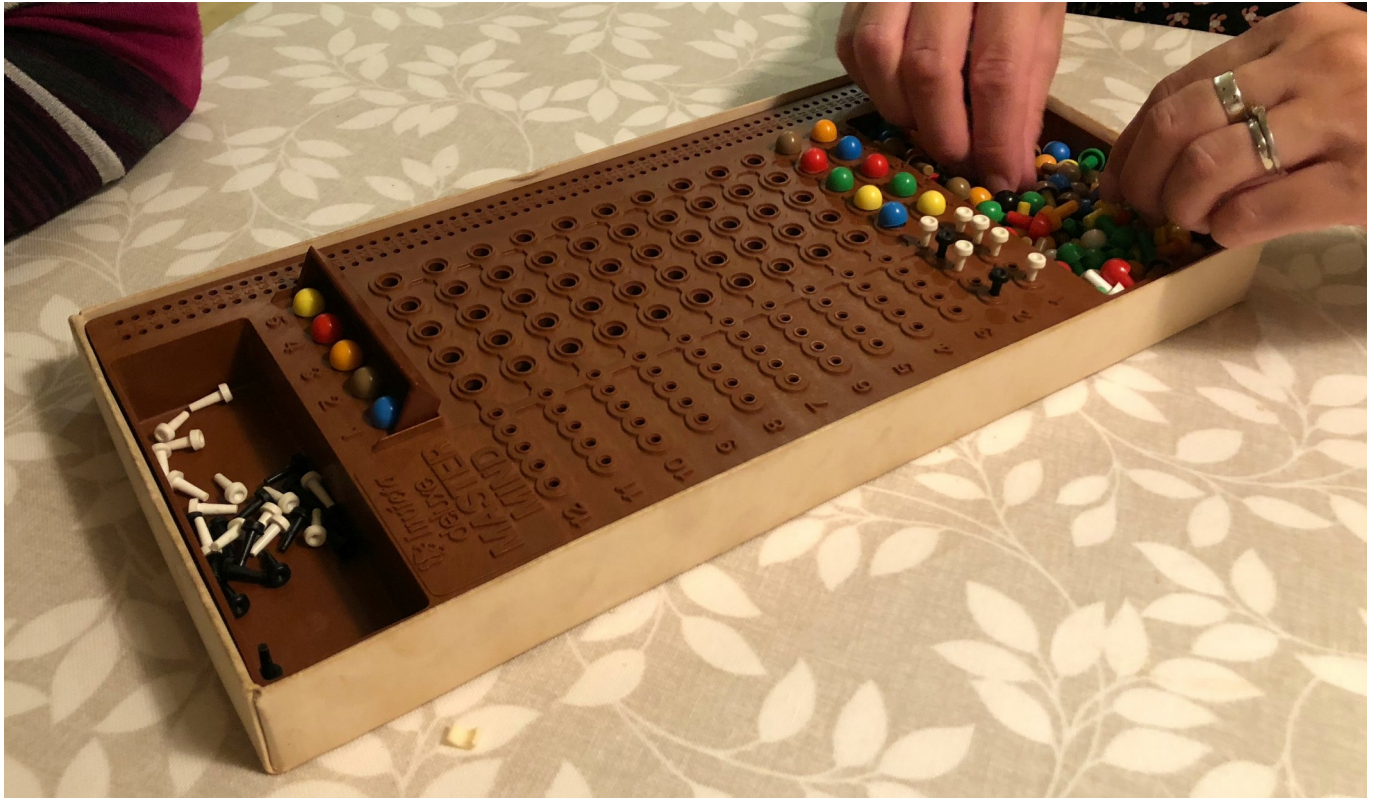
Common Vulnerabilities

1. Under-constrained Circuits
2. Nondeterministic Circuits
3. Arithmetic Over/Under Flows
4. Mismatching Bit Lengths
5. Unused Public Inputs Optimized Out
6. Frozen Heart: Forging of Zero Knowledge Proofs
7. Trusted Setup Leak
8. Assigned but not Constrained

Circuit Validation: Approaches

- Manual circuit design only by experts
- Circuit compiled from C code
  - Good compiler → circuit no worse than code
- Testing, e.g. with `circom-test`
- “Pair-Programming”
- For formally defined properties, use SAT-solver

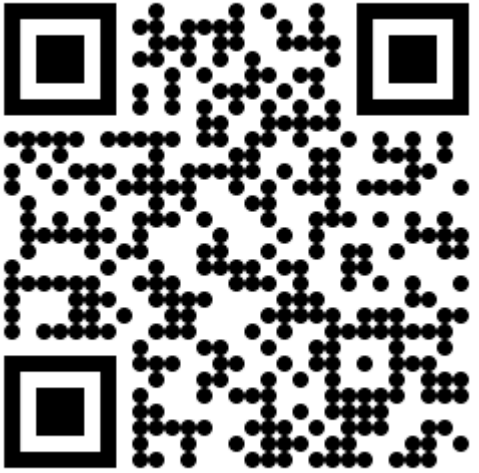
Today: Circuit for Mastermind



http://pennyplays.co.uk/wp-content/uploads/2018/09/Deluxe-Master-Mind-4-1.jpg

For Your Convenience

- Files from last time (e.g. Docker Image)
- Template files for Mastermind circuit
- CircomLib
- README.html



https://n.ethz.ch/~kklier/download/zkp/

Roadmap for Today

https://n.ethz.ch/~kklier/download/zkp/

1) Optional: Form groups of 2–3 people

2)Copy files to `mycircuits/` from last time

3) Complete templates in `mastermind.circom`

4) Play the game :D

Solutions

1. Sum

```
template Sum(N) {
    signal input in[N];
    signal output out;

    if (N == 0){
        out <== 0;
    }
    else{
        component S = Sum(N-1);
        for (var i = 0; i < N-1; i += 1){
            S.in[i] <== in[i];
        }

        out <== S.out + in[N-1];
    }
}
```

2. IsInSet

```
template IsInSet (N, set){
    signal input in;
    signal output out;

    if (N == 0){
        out <== 0;
    } else{
        var set_minus_1[N-1];
        for (var i = 0; i < N-1; i += 1){
            set_minus_1[i] = set[i];
        }

        component iis = IsInSet(N-1, set_minus_1);
        iis.in <== in;

        component ie = IsEqual();
        ie.in[0] <== in;
        ie.in[1] <== set[N-1];

        out <== iis.out + ie.out;
    }
}
```

3. CheckColors

```
template Mastermind_CheckColors(N_colors, colors, len_sequence){
    signal input in[len_sequence];
    signal output out;

    component iis[len_sequence];
    component sum = Sum(len_sequence);

    for (var i = 0; i < len_sequence; i += 1){
        iis[i] = IsInSet(N_colors, colors);
        iis[i].in <== in[i];
        sum.in[i] <== iis[i].out;
    }

    component eq = IsEqual();
    eq.in[0] <== sum.out;
    eq.in[1] <== len_sequence;

    out <== eq.out;
}
```

4. Commit

```
template Commit(len_sequence){
    signal input sequence[len_sequence];
    signal input r;
    signal output out;

    component P = Poseidon(len_sequence+1);

    for (var i = 0; i < len_sequence; i += 1){
        P.inputs[i] <== sequence[i];
    }

    P.inputs[len_sequence] <== r;

    out <== P.out;
}
```

5. Mastermind

```
template Mastermind (N_colors, colors, len_sequence){
    signal input in_guess[len_sequence];
    signal input in_solution[len_sequence];
    signal input in_r;
    signal output out_C;
    signal output out_correct;

    // Check that sequences are valid colors
    // -------------------------------------

    component checkColors_guess = Mastermind_CheckColors(N_colors,
colors, len_sequence);
    checkColors_guess.in <== in_guess;
    checkColors_guess.out === 1;

    component checkColors_solution = Mastermind_CheckColors(N_colors,
colors, len_sequence);
    checkColors_solution.in <== in_solution;
    checkColors_solution.out === 1;

    // Verifying/Computing Commitment
    // ------------------------------

    component Com = Commit(len_sequence);
    Com.sequence <== in_solution;
    Com.r <== in_r;
    out_C <== Com.out;

    // Compute Number of Correct Positions
    // -----------------------------------

    component eq[len_sequence];
    component sum = Sum(len_sequence);
    for (var i = 0; i < len_sequence; i += 1){
        eq[i] = IsEqual();
        eq[i].in[0] <== in_guess[i];
        eq[i].in[1] <== in_solution[i];
        sum.in[i] <== eq[i].out;
    }

    out_correct <== sum.out;
}
```