I’ve been exploring Pedersen hash/commitment and encountered some challenges while attempting to generate the same hash in both JavaScript and Noir.

Despite my efforts, I haven’t achieved the desired outcome.

Here the noir pedersen use:

let hashPedersen = std::hash::pedersen_hash([value1, value2, value3, value4]); let commitPedersen = std::hash::pedersen_commitment([value1, value2, value3, value4]);

std::println(hashPedersen); std::println(commitPedersen);

And here multiple test in JS.

Firstly with Barretenberg:

async function pedersenHash(inputs: any[]) { const bb: Barretenberg = await Barretenberg.new(2); const inputArray: Fr[] = inputs.map((str) => Fr.fromString(str)); return (await bb.pedersenCompress(inputArray)).toString(); }

secondly with circom JS

function pedersenHash(inputs: any[]) { const hash = pedersenB.hash(inputs); return hash; }

function pedersenHashUnpack(inputs: any[]) { const unpackP = babyJub.unpackPoint(pedersenHash(inputs)); return unpackP; }

I’m aiming for a clear understanding of how to implement Pedersen hash/commitment in both JavaScript and Noir, ensuring consistency in the generated hashes.

Any assistance, code snippets, or recommendations would be immensely helpful.

Thanks in advance for your expertise!