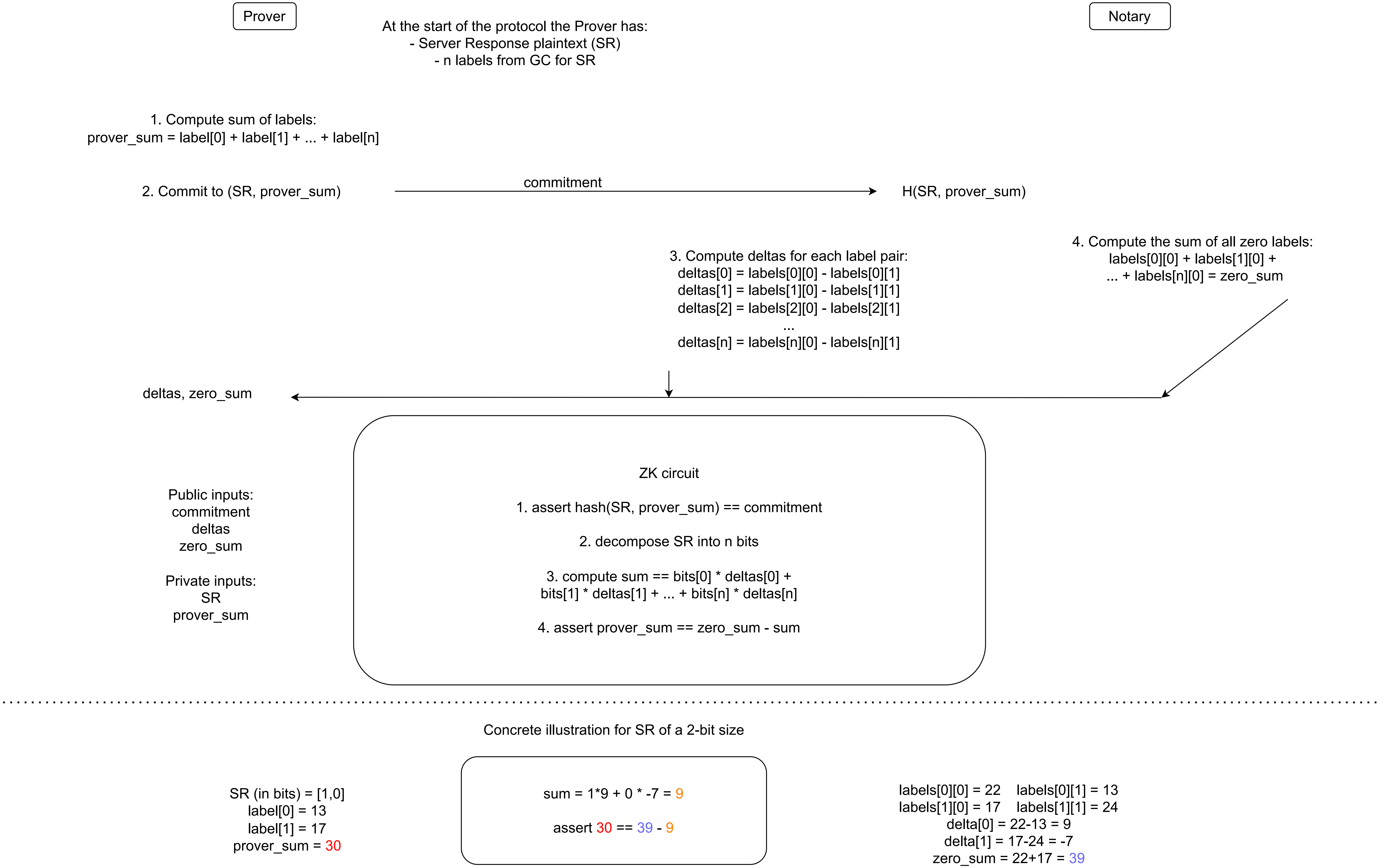

Prover

Notary

At the start of the protocol the Prover has:
- Server Response plaintext (SR)
- n labels from GC for SR

1. Compute sum of labels:
prover_sum = label[0] + label[1] + ... + label[n]

2. Commit to (SR, prover_sum)

commitment

H(SR, prover_sum)

3. Compute deltas for each label pair:
deltas[0] = labels[0][0] - labels[0][1]
deltas[1] = labels[1][0] - labels[1][1]
deltas[2] = labels[2][0] - labels[2][1]
...
deltas[n] = labels[n][0] - labels[n][1]

4. Compute the sum of all zero labels:
labels[0][0] + labels[1][0] +
... + labels[n][0] = zero_sum

deltas, zero_sum

Public inputs:
commitment
deltas
zero_sum

Private inputs:
SR
prover_sum

ZK circuit

1. assert hash(SR, prover_sum) == commitment

2. decompose SR into n bits

3. compute sum == bits[0] * deltas[0] +
bits[1] * deltas[1] + ... + bits[n] * deltas[n]

4. assert prover_sum == zero_sum - sum

Concrete illustration for SR of a 2-bit size

SR (in bits) = [1,0]
label[0] = 13
label[1] = 17
prover_sum = 30

sum = 1*9 + 0 * -7 = 9

assert 30 == 39 - 9

labels[0][0] = 22   labels[0][1] = 13
labels[1][0] = 17   labels[1][1] = 24
delta[0] = 22-13 = 9
delta[1] = 17-24 = -7
zero_sum = 22+17 = 39