A[1…93] – 93 bit – 12 bytes

B[1…84] – 84 bit => 11 bytes

C[1…111] – 111 bit => 14 bytes

IV[1…80] – 80 bit => 10 bytes

Key[1…80] – 80 bit => 10 bytes

Out\_A = A[93] xor (A[91] and A[92]) xor A[66] = (A[93] + A[91] \* A[92] + A[66]) % 2

Out\_B = B[84] xor (B[83] and B[82]) xor B[69] = (B[84] + B[83] \* A[82] + B[69]) % 2

Out\_C = C[111] xor (C[109] and C[110]) xor C[66] = (C[111] + C[110] \* C[109] + C[66]) % 2

S = Out\_A xor Out\_B xor Out\_C = (Out\_A + Out\_B + Out\_C) % 2

In\_A = A[1]’ = A[69] xor Out\_C = (A[69] + Out\_C) % 2

In\_B = B[1]’ = B[78] xor Out\_A = (B[78] + Out\_A) % 2

In\_C = C[1]’ = C[87] xor Out\_B = (C[87] + Out\_B) % 2

Initialization:

A[1…80] = IV[1…80]

B[1…80] = key[1…80]

C[109…111] = 1

else = 0

Phase 1: output first 1152 bits

Phase 2: keystream[i] = S[i+1152]

A screenshot of a computer code

Description automatically generated with medium confidence

Sizeof(int) = 4 bytes = 32 bits

A register = 93 first bits of array A

B register = 84 first bit of array B

C register = 111 first bit of array C

IV(Key) = array IV(key)

Feedfoward, feedback and inputs are keep tracked by each var, syntax: RegisterName\_Index

Output bits = Out\_RegisterName

IV and Key generated by key\_iv\_gen.py (TRNG: os.urandom) to iv.txt and key.txt