KISS: A Bit Too Simple

Greg Rose
ggr@qualcomm.com

Outline

- KISS – random number generator
- Subgenerators
- Efficient attack
- New KISS and attack
- Conclusion

One approach to PRNG security

"A random number generator is like sex:
When it's good, its wonderful;
And when it's bad, it's still pretty good."
Add to that, in line with my recommendations
on combination generators;
"And if it's bad, try a twosome or threesome.”

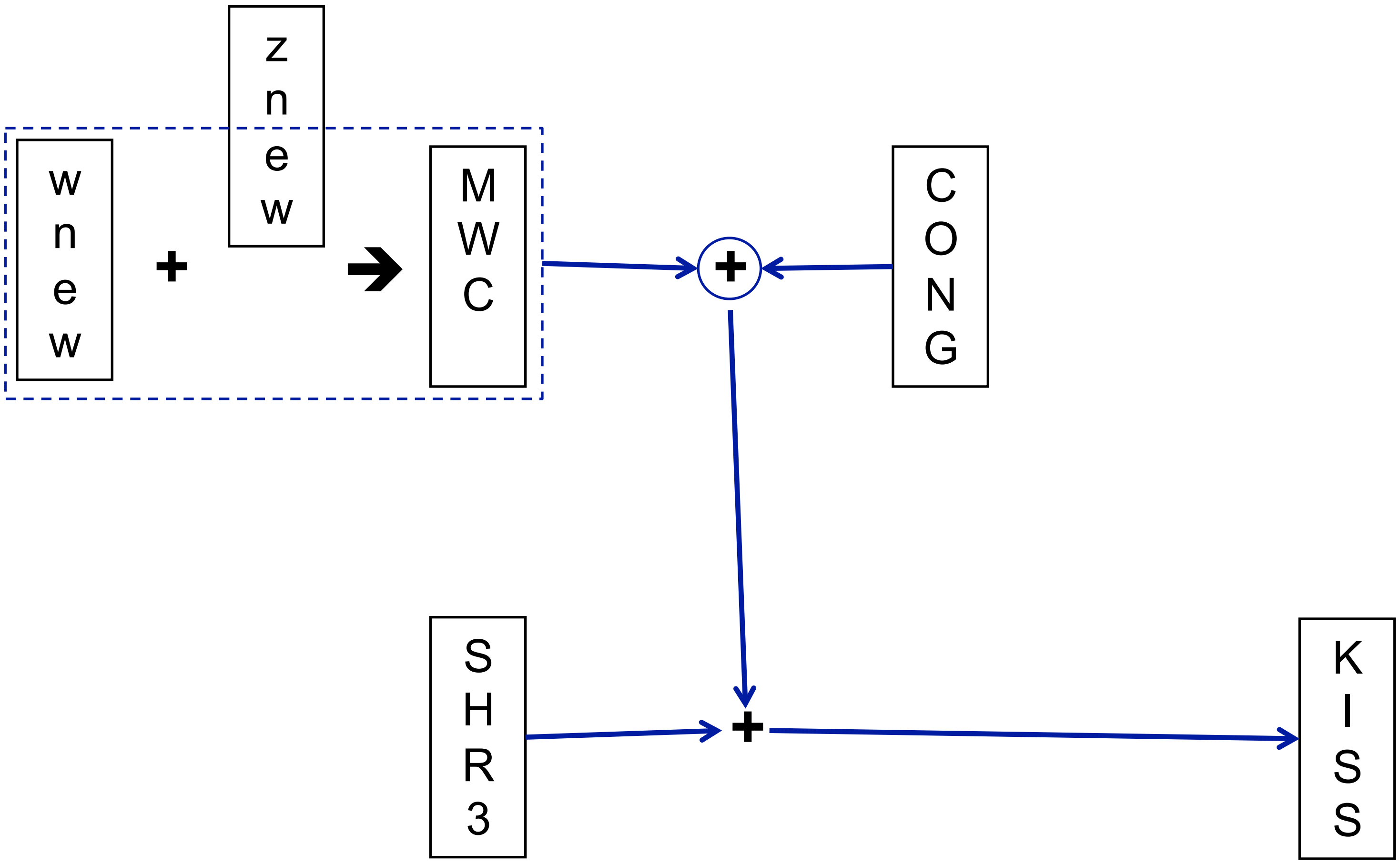
-- George Marsaglia, quoting himself (1999)

KISS – a Pseudo-Random Number Generator

- “Keep it Simple Stupid”
- Marsaglia and Zaman, Florida State U, 1993
- Marsaglia posts C version to `sci.crypt`, 1998/99, took off
- Never said it was secure!
  - Good thing, too…
  - But others seem to think it is.

```
#define znew  (z=36969*(z&65535)+(z>>16))
#define wnew  (w=18000*(w&65535)+(w>>16))
#define MWC  ((znew<<16)+wnew )
#define SHR3  (jsr^=(jsr<<17), jsr^=(jsr>>13), jsr^=
   (jsr<<5))
#define CONG  (jcong=69069*jcong+1234567)
#define KISS  ((MWC^CONG)+SHR3)
```

KISS diagram



PRODUCT SECURITY INITIATIVE
QUALCOMM

Multiply With Carry subgenerator

```
#define znew (z=36969*(z&65535)+(z>>16))
#define wnew (w=18000*(w&65535)+(w>>16))
#define MWC ((znew<<16)+wnew )
```

- *znew* and *wnew*
- 16 bits "random looking", 32 bits of state
- Multiply by constant (18000, 36969 resp), add carry from previous multiplication
- Periods about $2^{29.1}$ and $2^{30.2}$ – two long cycles each
- Two bad values (0 and something else) repeat forever
- Large states go into smaller ones after one update
- $f(x) = cx \bmod 2^{16}c - 1$
  - modulus is prime for the two constants shown

- *znew* only affects high order bits.

Linear Congruential subgenerator

```
#define CONG (jcong=69069*jcong+1234567)
```

- Well studied, period 2^{32}, single long cycle
- Low order bits form smaller linear congruential generators

- In particular, LSB goes "01010101010…"

3-Shift Register subgenerator

```
#define SHR3 (jsr^=(jsr<<17), jsr^=(jsr>>13), jsr^=
   (jsr<<5))
```

- Linear, but not like LFSR
- Authors assume long period, but wrong
- LSBs of output form one of 64 LFSRs
- Periods range from 1 to $2^{28.2}$ (*not* 2^{32}-1!)

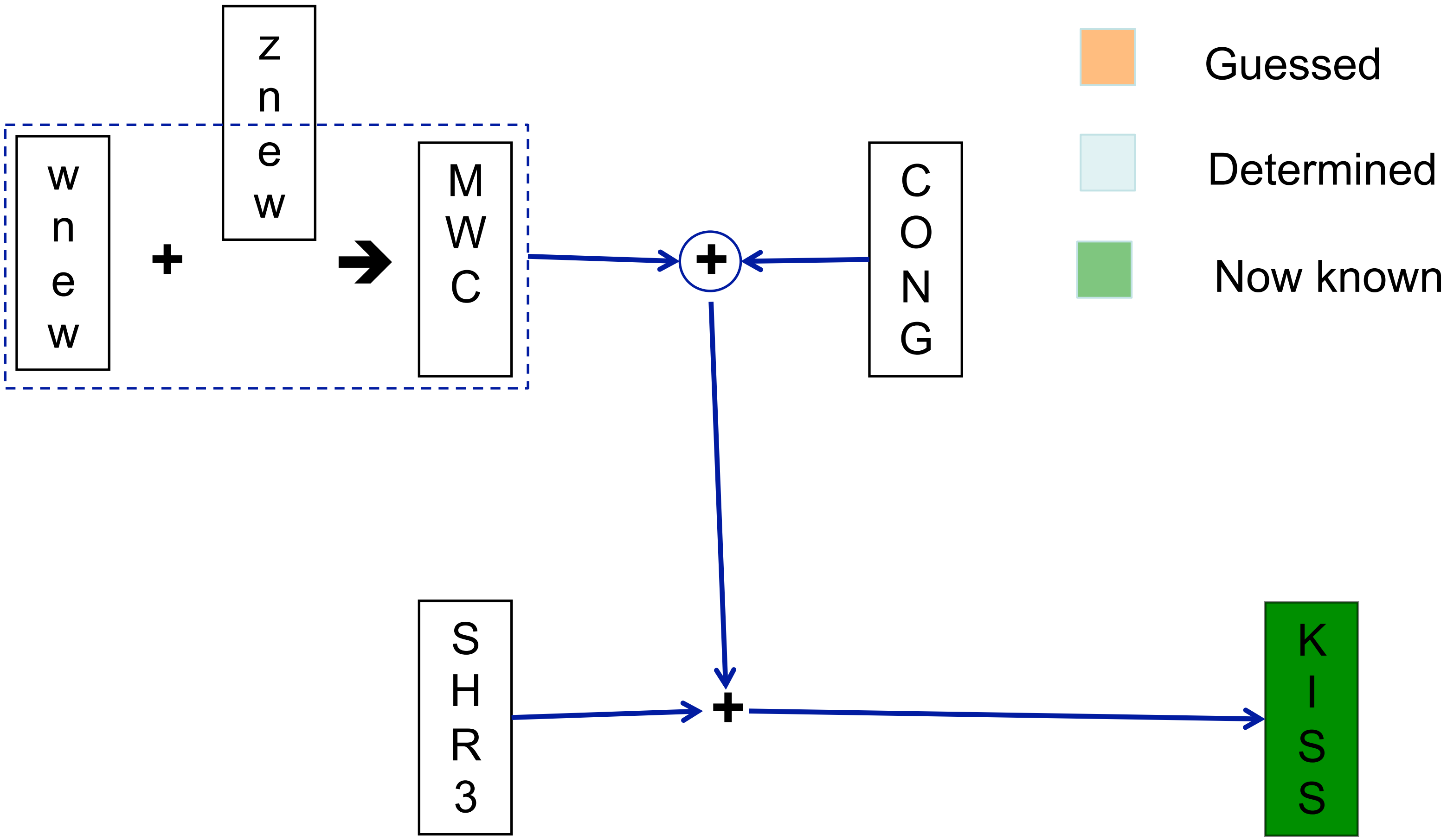
- Can recover initial state from 32 consecutive LSBs easily
  - Binary matrix multiplication

- (It turns out that Marsaglia got the constants 13 and 17 back-to-front; subsequent versions of KISS get them right and the generator then has a full period.)

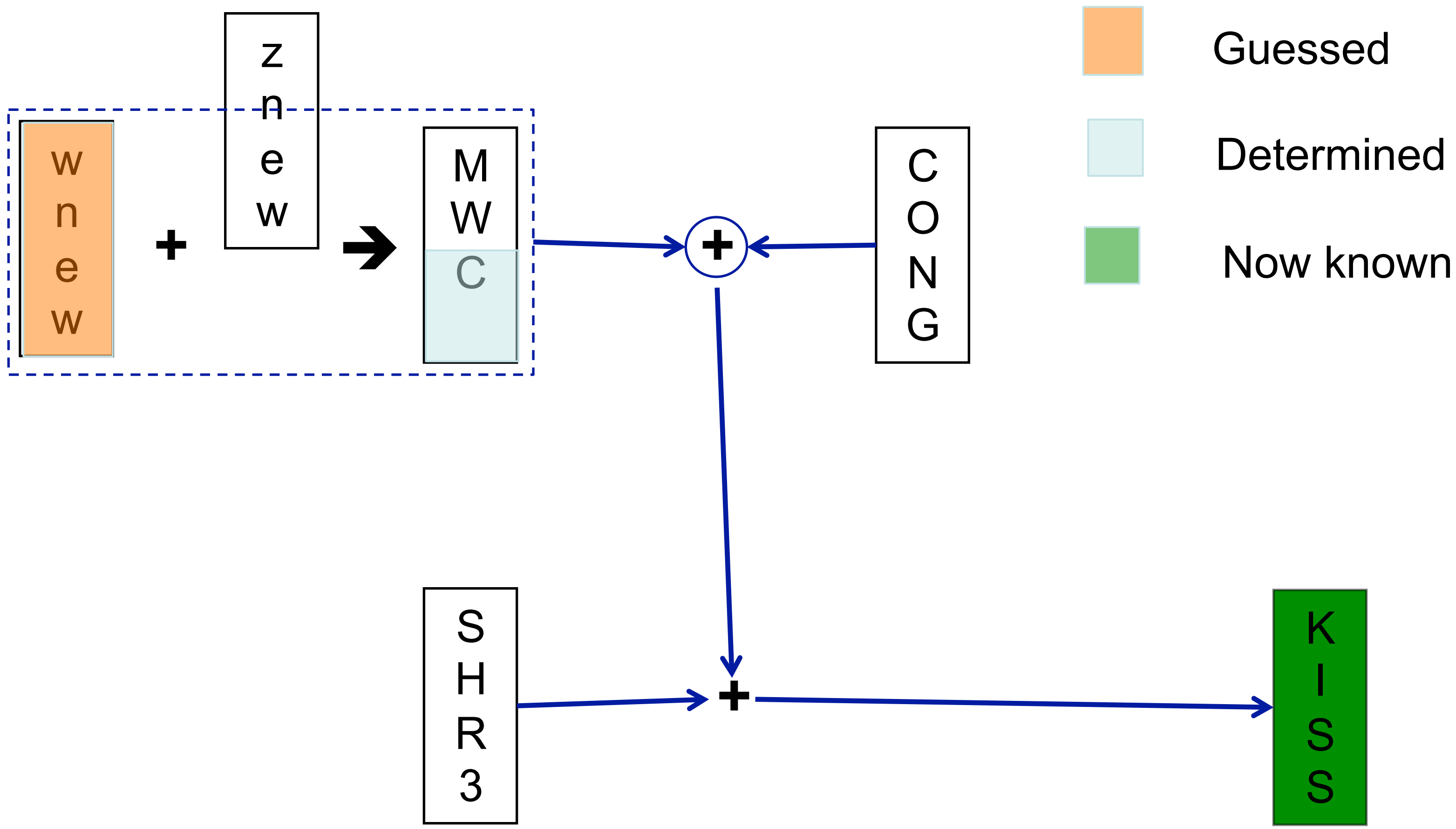
Attack idea

- Divide and Conquer
  - Registers are updated independently of each other, then combined
  - So try to get rid of effects of one or more registers
  - One of them is already partly gone!
- Exploit weaknesses (eg. Linearity of SHR3, low order bits of CONG)
- Guess and Determine
  - Guess (that is, try all possibilities) for some values, then
  - Derive other values
  - Verify whether still consistent

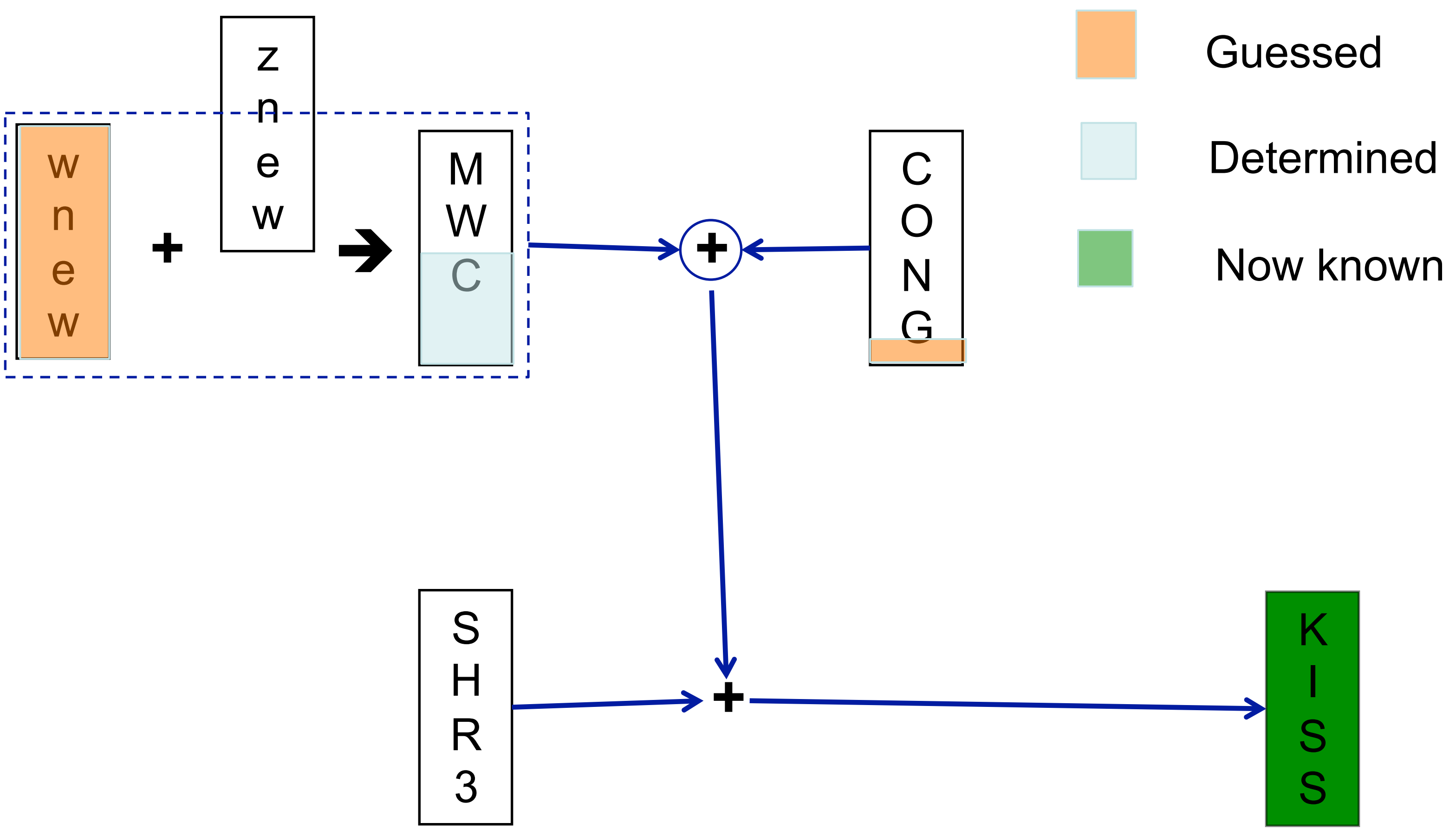
What do we know at the start?



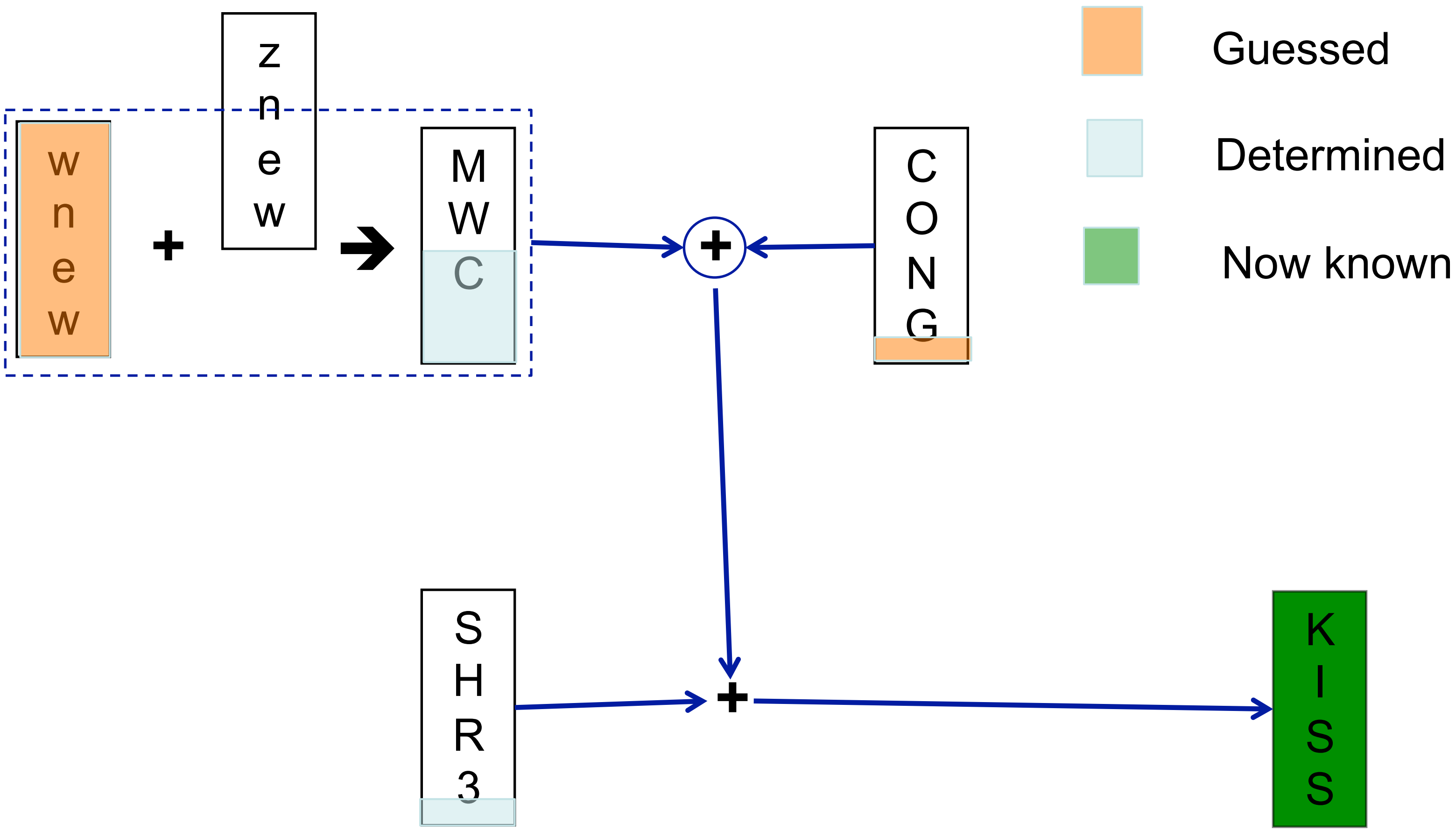
Guess *wnew*



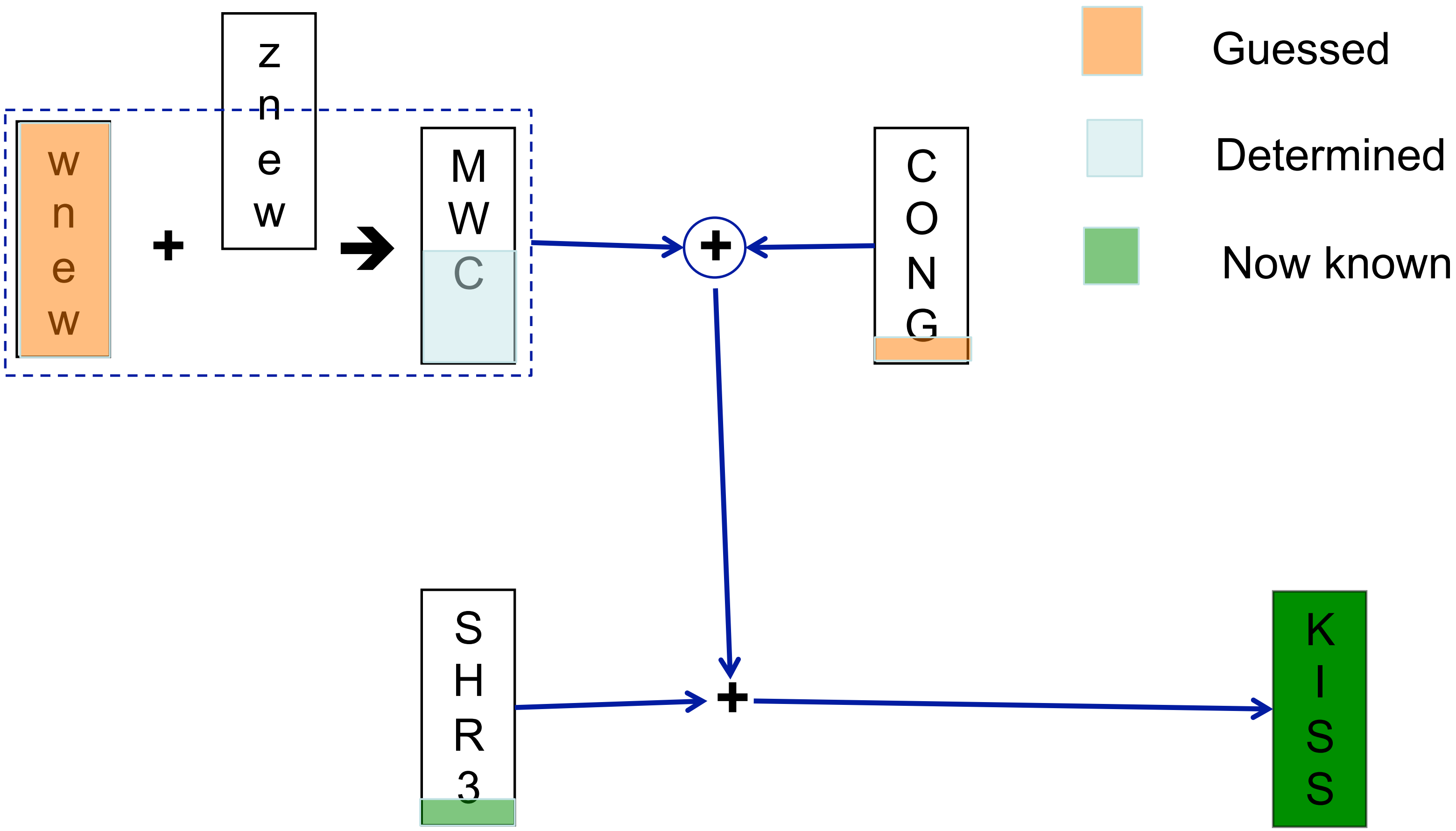
Guess LSB of *CONG* (01010… or 10101…)



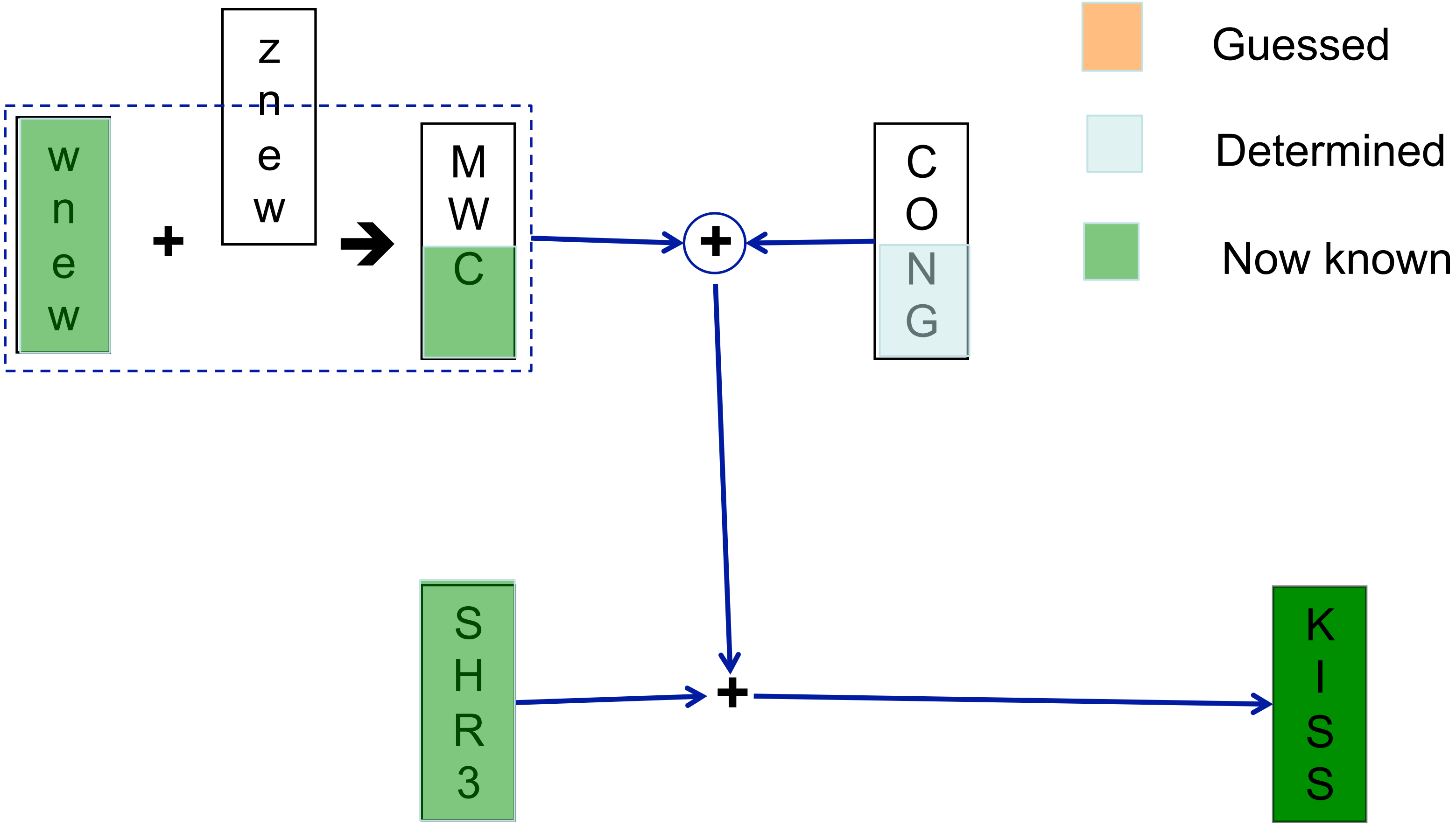
Determine LSB sequence from *SHR3*



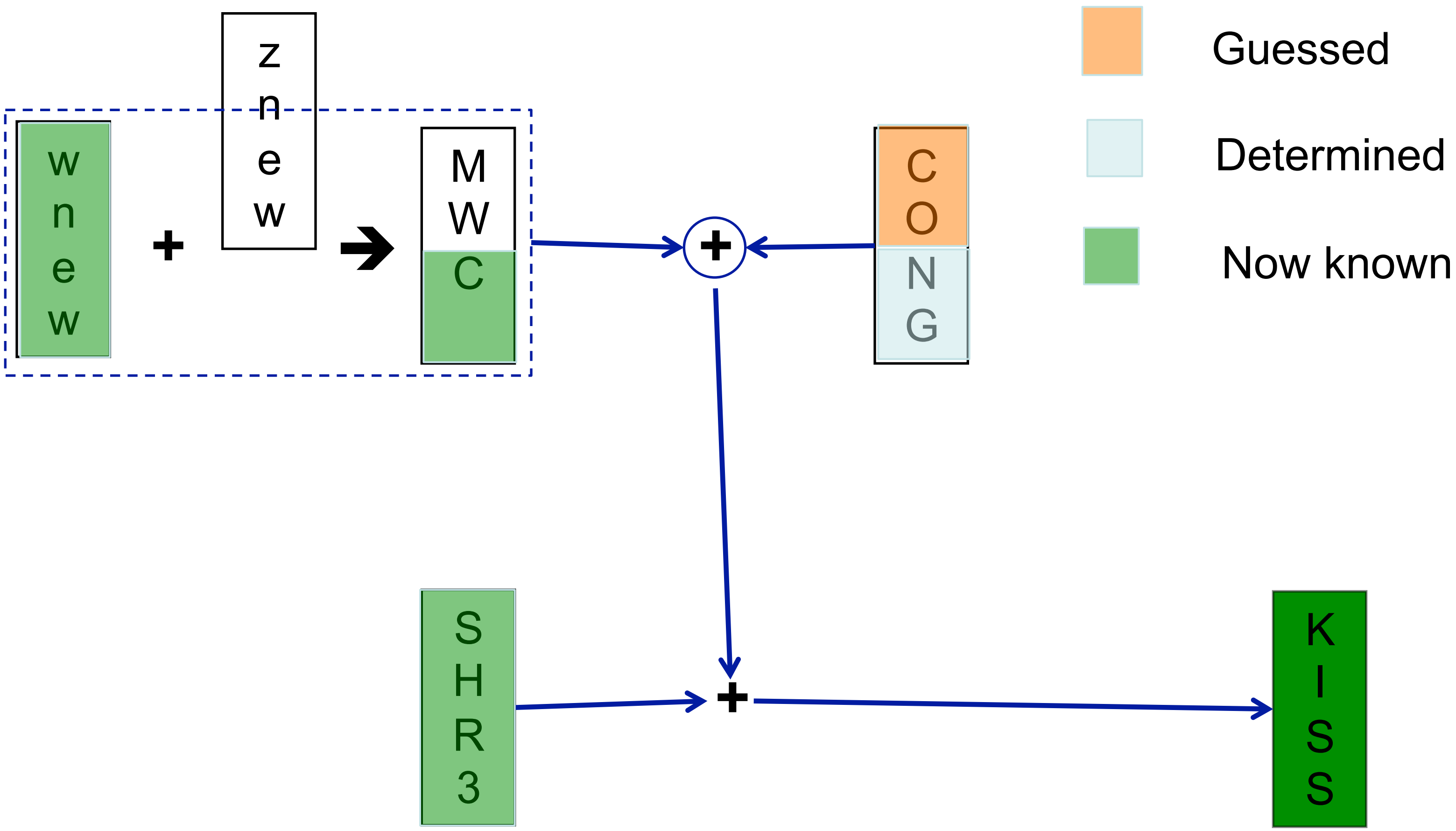
Verify LSB sequence from *SHR3* is LFSR



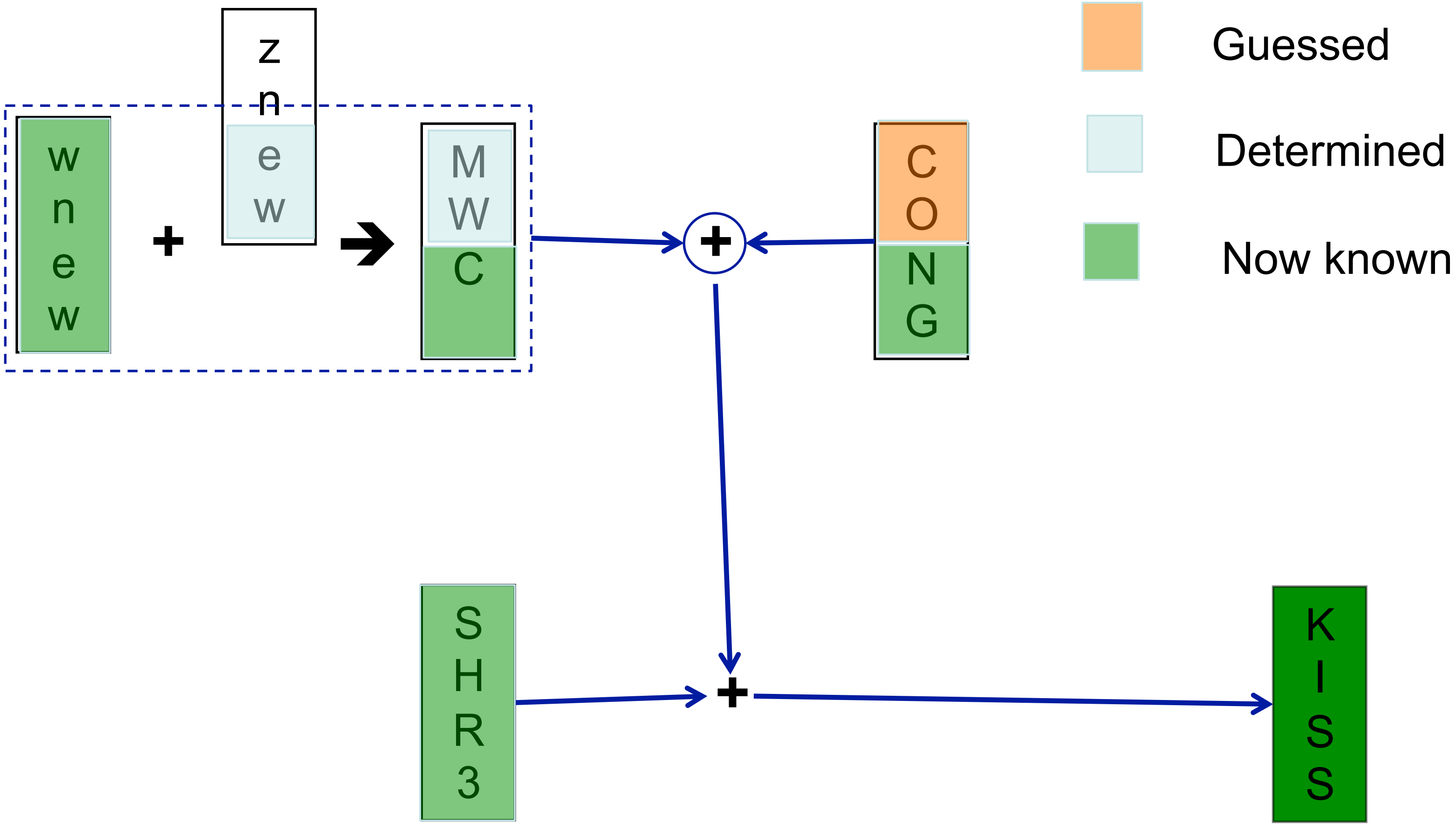
Determine half of *CONG*



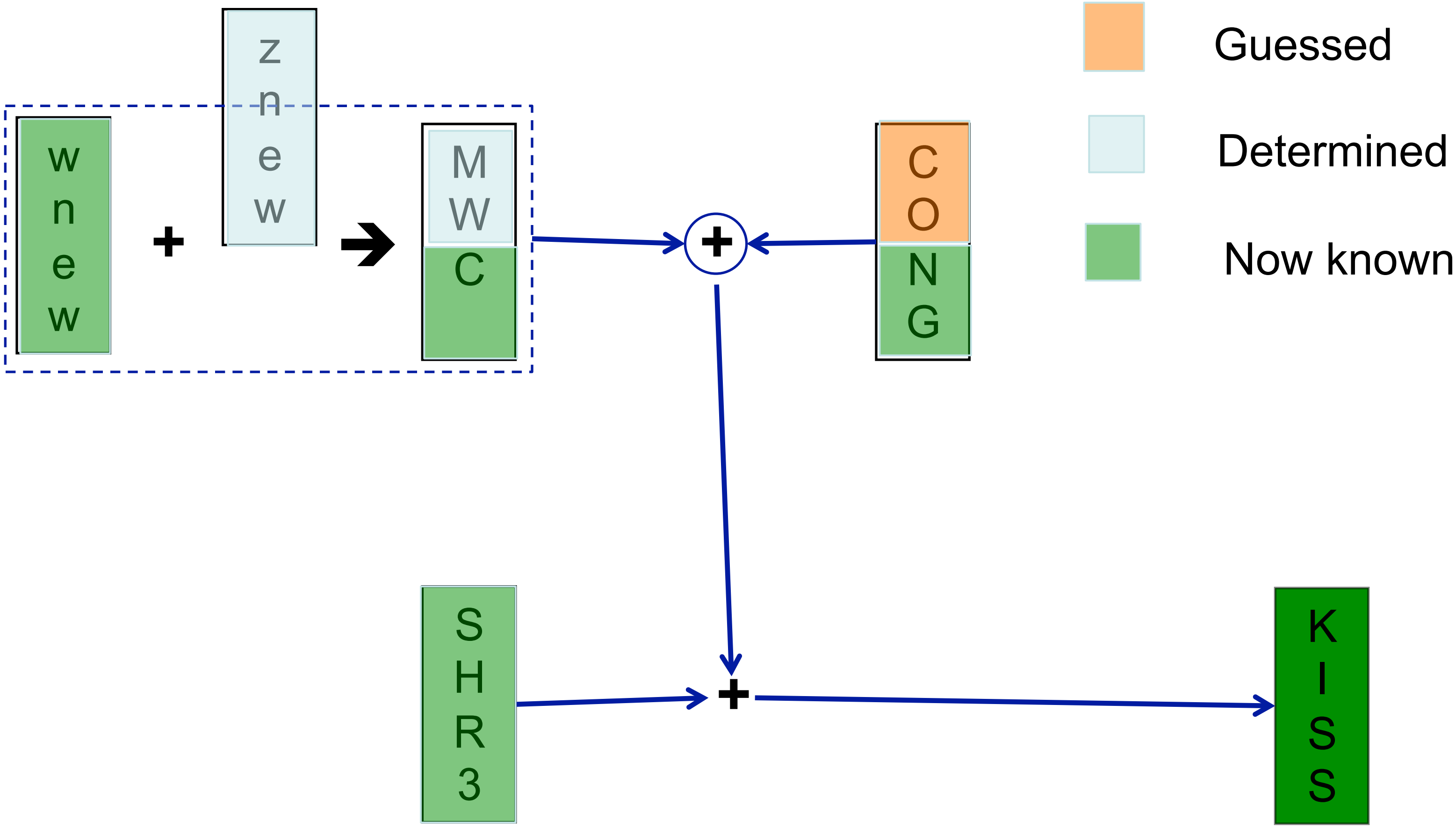
Guess top half of *CONG*



Determine low half of *znew*

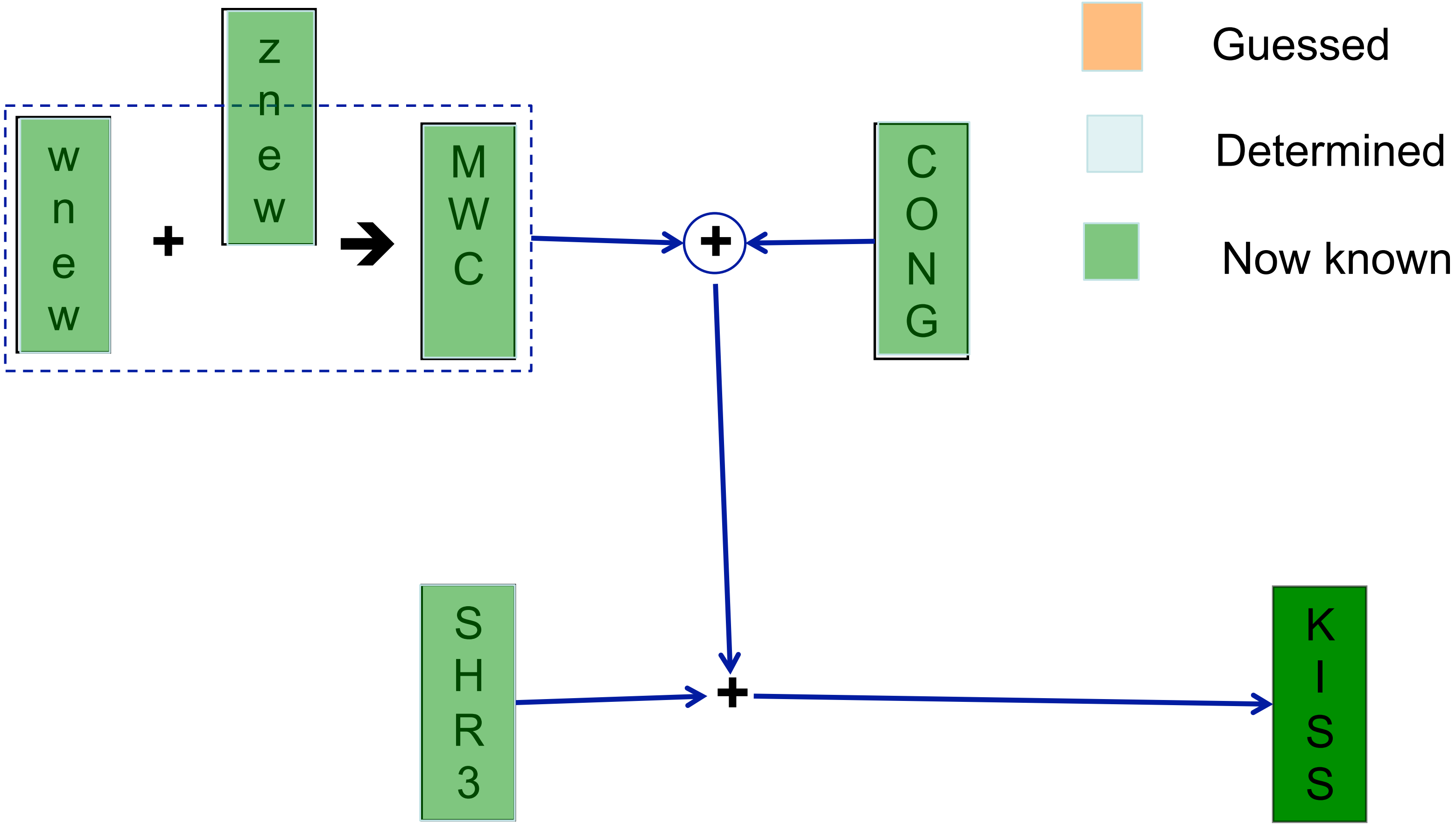


Determine high half of *znew* from low half



PRODUCT SECURITY INITIATIVE
QUALCOMM

And verify…

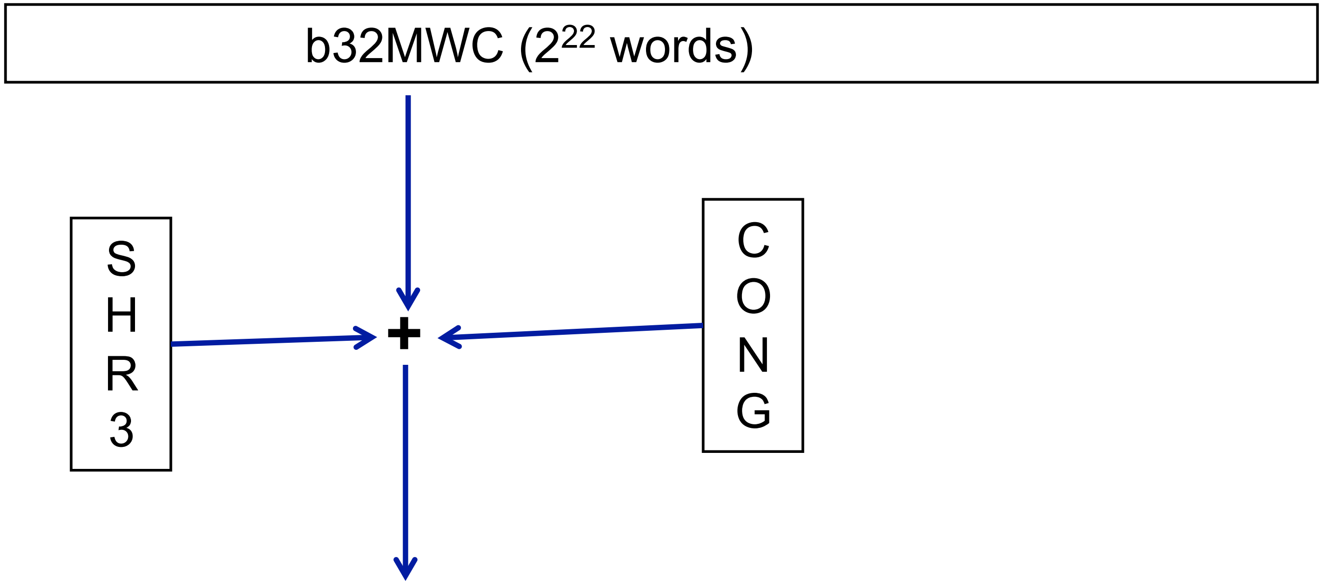


How much work?

- Dominated by trying, on average, 589,823,999 values for *wnew*
- And for each one, using Berlekamp-Massey algorithm to check whether the candidate for *SHR3* is LFSR
  - Alternatively, can check parity equations.
- Few hours on laptop.

Newer KISS

- Sci.crypt 2011 posting by Marsaglia
- Looking for longer and longer cycles
- Period > $10^{40,000,000}$
- State is ridiculously large (2^{22}+3 32-bit words)
- Again combines multiple components “for security”



New KISS

```
static unsigned long Q[4194304],carry=0;
unsigned long b32MWC(void)
{unsigned long t,x; static int j=4194303;
j=(j+1)&4194303;
x=Q[j]; t=(x<<28)+carry;
carry=(x>>4)-(t<x);
return (Q[j]=t-x);
}
#define CNG ( cng=69069*cng+13579 )
#define XS ( xs^=(xs<<13), xs^=(xs>>17), xs^=(xs<<5) )
#define KISS ( b32MWC()+CNG+XS )
```

(Note 13 and 17 reversed from before)

Complemented Multiply With Carry

- Large circular buffer with carry variable
- Extremely long period
- State values are used directly for output
- Can be run backward

- After one rotation through buffer, can check consistency easily (used in attack)
- By itself has no cryptographic strength at all
  - output is state

Attack on New KISS

- Simple divide and conquer
- Guess state of CONG and SHR3
- Run generator forward slightly more than a full rotation of b32MWC's buffer
- If 3 outputs are mutually consistent, must have guessed correctly
- Run backward to recover full initial state
- Equivalent to 2^{63} key setup operations
  - But the key is huge, so is the key setup operation

Optimization of attack

- Only care about v_0, v_1, $v_{2,}$ and $v_{R,}$ v_{R+1}, v_{R+2}
- Can fast-forward the simple generators *cong* and *SHR3*
- Can maintain $cong_0$, $cong_R$ and step them forward to enumerate cycle, similarly SHR3 cycles.
- Attack is now 2^{63} basic operations, about 2^{41} key setup operations

Conclusion

- M & Z overestimated the period by about a factor of 10
- KISS is not secure
- Need about 70 words of generated output (original KISS)
- Can apply attack to unknown (but biased) plaintext
  - Replace B-M step with fast correlation attack
  - Still surprisingly efficient

- Don't use KISS if you need security!