

Double filtering approach for cracking the structure seed

Assumptions

Let's assume we found a structure and we know its chunk position. Here's the relationship, assuming the [document](#) is true:

$$s_0 = (w + Px_r + Qz_r + salt) \oplus M \pmod{2^{48}}$$

$$x_c = x_r \cdot spacing + (s_1 \gg 17) \text{ mod } bound$$

$$z_c = z_r \cdot spacing + (s_2 \gg 17) \text{ mod } bound$$

Let's do a bit of renaming here:

- $Px_r + Qz_r + salt = D$
- $x_c - x_r \cdot spacing = \Delta x$
- $z_c - z_r \cdot spacing = \Delta z$

With that, the original equations become:

$$s_0 = (w + D) \oplus M \pmod{2^{48}}$$

$$(s_1 \gg 17) \text{ mod } bound = \Delta x$$

$$(s_2 \gg 17) \text{ mod } bound = \Delta z$$

I will also claim the following relationships are true:

1. The bitshift expansion:

$$x \gg n = \frac{x - (x \text{ mod } 2^n)}{2^n}$$

2. Mod factoring:

$$(m \cdot x) \text{ mod } (m \cdot n) = m \cdot (x \text{ mod } n)$$

- where $m, n \in N$

3. Splitting the mod bracket:

If $(a \text{ mod } n) + (b \text{ mod } n) \geq n$:

$$(a + b) \text{ mod } n = (a \text{ mod } n) + (b \text{ mod } n) - n$$

If $(a \text{ mod } n) + (b \text{ mod } n) < n$:

$$(a + b) \text{ mod } n = (a \text{ mod } n) + (b \text{ mod } n)$$

4. Double mod relationship:

If a is divisible by b :

$$(x \bmod a) \bmod b = x \bmod b$$

Linearizing world seed to RNG state conversion

Let's look at the s_0 equation:

$$s_0 = (w + D) \oplus M \pmod{2^{48}}$$

We know that $M = 25214903917$. Notice that $\lfloor \log_2(M) \rfloor = 34$, which means that it affects this amount of low bits plus one. But we still have $48 - 35 = 13$ unaffected high bits.

Rewriting the equation to account for that, we get:

$$s_0 = 2^{35} \cdot (w + D) \gg 35 + e_0 \pmod{2^{48}}$$

where:

- $e_0 = (w_l + D) \oplus M \pmod{2^{35}}$
- w_l is the lowest 35 bits of w

So, at the cost of adding another variable, albeit small, we are able to make the expression slightly more algebraically malleable. We could keep $2^{35} \cdot (w + D) \gg 35$ as it is, but the problem is that this expression varies for each structure. Since we are trying to combine constraints here, we'll have to split it up.

Bitshifts can be defined as such:

$$x \gg n = \frac{x - x \bmod 2^n}{2^n}$$

With that in mind, let's do a bit of algebra on s_0 :

$$s_0 = 2^{35} \cdot (w + D) \gg 35 + e_0 \pmod{2^{48}}$$

Expanding the bitshift definition:

$$s_0 = w + D - (w + D) \bmod 2^{35} + e_0 \pmod{2^{48}}$$

Now, we know that:

If $w \bmod 2^{35} + D \bmod 2^{35} \geq 2^{35}$:

$$(w + D) \bmod 2^{35} = w \bmod 2^{35} + D \bmod 2^{35} - 2^{35}$$

If $w \bmod 2^{35} + D \bmod 2^{35} < 2^{35}$:

$$(w + D) \bmod 2^{35} = w \bmod 2^{35} + D \bmod 2^{35}$$

To save on clutter, we can rewrite the equation as such:

$$(w + D) \bmod 2^{35} = w \bmod 2^{35} + D \bmod 2^{35} - 2^{35} \cdot c_0$$

where:

- $c_0 \in \{0, 1\}$

Notice that we also will subtract $D \bmod 2^{35}$ from D , which will also leave us with just the higher bits of D . This means that:

$$2^{35} \cdot (w + D) \gg 35 = w + D - w \bmod 2^{35} - D \bmod 2^{35} + c_0 \cdot 2^{35} = 2^{35} \cdot (w \gg 35 + D \gg 35 + c_0)$$

This property extends to arbitrary a and b , so we found a way to give bitshifts almost linear properties. Almost is good enough here.

As discussed earlier, let's rename $w \gg 35$ to w_h and $D \gg 35$ to D_h to reduce clutter.

Let's use this new property, then:

$$s_0 = 2^{35} \cdot (w_h + D_h + c_0) + e_0 \pmod{2^{48}}$$

where:

- $e_0 = (w_l + D) \oplus M \pmod{2^{35}}$

Rewriting chunk coordinate equations in terms of s0

Let's grab the equations:

$$(s_1 \gg 17) \bmod \text{bound} = \Delta x$$

$$(s_2 \gg 17) \bmod \text{bound} = \Delta z$$

In this case, it'd be far more convenient to work entirely under $\bmod \text{bound}$. So, let's rewrite these equations under $\bmod \text{bound}$:

$$s_1 \gg 17 = \Delta x \pmod{\text{bound}}$$

$$s_2 \gg 17 = \Delta z \pmod{\text{bound}}$$

Since this is an LCG, we know that internal RNG states relate to each other as such:

$$s_{n+1} = s_n \cdot M + A \pmod{2^{48}}$$

If we wish to step two times:

$$s_{n+2} = s_{n+1} \cdot M + A = (s_n \cdot M + A) \cdot M + A = s_n \cdot M^2 + A \cdot (M + 1) \pmod{2^{48}}$$

Since we are interested $n = 0$, we get the following:

$$s_1 = s_0 \cdot M + A \pmod{2^{48}}$$

$$s_2 = s_0 \cdot M^2 + A \cdot (M + 1) \pmod{2^{48}}$$

Now, we can carefully substitute these equations in the original:

$$((s_0 \cdot M + A) \bmod 2^{48}) \gg 17 = \Delta x \pmod{\text{bound}}$$

$$((s_0 \cdot M^2 + A \cdot (M + 1)) \bmod 2^{48}) \gg 17 = \Delta z \pmod{\text{bound}}$$

Note that we can also swap bitshifts and mods like this:

$$((s_0 \cdot M + A) \gg 17) \bmod 2^{31} = \Delta x \pmod{\text{bound}}$$

$$((s_0 \cdot M^2 + A \cdot (M + 1)) \gg 17) \bmod 2^{31} = \Delta z \pmod{\text{bound}}$$

Both of these representations will be useful later.

Defining the weak condition

Look at the second representation of the equations:

$$((s_0 \cdot M + A) \gg 17) \bmod 2^{31} = \Delta x \pmod{\text{bound}}$$

$$((s_0 \cdot M^2 + A \cdot (M + 1)) \gg 17) \bmod 2^{31} = \Delta z \pmod{\text{bound}}$$

It's really painful to work under two mods at once. We know that we can bring all of these terms under one mod if we find the common factor $g = \gcd(2^{31}, \text{bound})$. You can convince yourself that g can only be a power of 2. With bring these expressions under mod g :

$$(s_0 \cdot M + A) \gg 17 = \Delta x \pmod{g}$$

$$(s_0 \cdot M^2 + A \cdot (M + 1)) \gg 17 = \Delta z \pmod{g}$$

Let's expand our bitshift definition here. Note that we'll be multiplying both sides by 2^{17} and modulo as well:

$$s_0 \cdot M + A - r_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot g}$$

$$s_0 \cdot M^2 + A \cdot (M + 1) - r_2 = 2^{17} \cdot \Delta z \pmod{2^{17} \cdot g}$$

Note that $r_1, r_2 \in [0, 2^{17} - 1]$.

Remember how we defined s_0 :

$$s_0 = 2^{35} \cdot (w_h + D_h + c_0) + e_0 \pmod{2^{48}}$$

where:

- $e_0 = (w + D) \oplus M \pmod{2^{35}}$
- $c_0 \in \{0, 1\}$

bound is typically very small and doesn't contain too many factors of 2. With that assumption, we can claim that 2^{35} will vanish under mod $2^{17} \cdot g$, leaving us only with e_0 term. So, with all of that, we get:

$$e_0 \cdot M + A - r_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot g}$$

$$e_0 \cdot M^2 + A \cdot (M + 1) - r_2 = 2^{17} \cdot \Delta z \pmod{2^{17} \cdot g}$$

where:

- $e_0 = (w + D) \oplus M \pmod{2^{17} \cdot g}$

Remember that from the expansion of bitshifts:

- $r_1 = (s_0 \cdot M + A) \bmod 2^{17}$
- $r_2 = (s_0 \cdot M^2 + A \cdot (M + 1)) \bmod 2^{17}$

From our definition of s_0 :

$$s_0 = 2^{35} \cdot (w_h + D_h + c_0) + e_0 \pmod{2^{48}}$$

The 2^{35} term will vanish, leaving us with:

- $r_1 = (e_0 \cdot M + A) \bmod 2^{17}$
- $r_2 = (e_0 \cdot M^2 + A \cdot (M + 1)) \bmod 2^{17}$

Notice that we have $e_0 \cdot M + A$ and $e_0 \cdot M^2 + A \cdot (M + 1)$ in the equations already. This means, we can apply our bitshift definition here to get rid of r_1 and r_2 terms:

$$\begin{aligned} 2^{17} \cdot ((e_0 \cdot M + A) \gg 17) &= 2^{17} \cdot \Delta x \pmod{2^{17} \cdot g} \\ 2^{17} \cdot ((e_0 \cdot M^2 + A \cdot (M + 1)) \gg 17) &= 2^{17} \cdot \Delta z \pmod{2^{17} \cdot g} \end{aligned}$$

We can now drop 2^{17} everywhere:

$$\begin{aligned} (e_0 \cdot M + A) \gg 17 &= \Delta x \pmod{g} \\ (e_0 \cdot M^2 + A \cdot (M + 1)) \gg 17 &= \Delta z \pmod{g} \end{aligned}$$

These conditions give a quick and easy way to verify e_0 . But we can go a bit further. We can convert e_0 to w_l as such:

$$e_0 = (w_l + D) \oplus M \pmod{g}$$

Let's apply XOR M to both sides:

$$e_0 \oplus M = w_l + D \pmod{g}$$

Now, we can get an expression for w_l :

$$w_l = (e_0 \oplus M) - D \pmod{g}$$

w_l is the same for each equation. So, when bruteforcing e_0 , to prematurely check our partial seeds, we will convert the first constraint's e_0 candidate to w_l and with that generate a corresponding e_0 for the rest of the constraints.

Note that we still can't solve this directly as we have 3 variables and only 2 linear equations. But we can reduce the search space by a factor of g for free. Let's use the first equation to generate e_0 candidates using r_1 . To do that, we need to isolate e_0 . Since we can do this for one equation anyway, let's choose the simpler one:

$$e_0 \cdot M + A - r_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot g}$$

$$e_0 \cdot M = r_1 + 2^{17} \cdot \Delta x - A \pmod{2^{17} \cdot g}$$

$$e_0 = M^{-1} \cdot (r_1 + 2^{17} \cdot \Delta x - A) \pmod{2^{17} \cdot g}$$

That's just the lower $17 + \log_2(g)$ bits however. Let's account for that lost information:

$$e_0 = M^{-1} \cdot (r_1 + 2^{17} \cdot \Delta x - A) + q \cdot 2^{17} \cdot g \pmod{2^{35}}$$

where:

- $r_1 \in [0, 2^{17} - 1]$
- $q \in [0, 2^{18} - 1]$

Notice, by bruteforcing r_1 and q , we are already accounting for the information gained from working under mod g .

Defining the strong condition

Look at the first representation of the equations:

$$((s_0 \cdot M + A) \bmod 2^{48}) \gg 17 = \Delta x \pmod{\text{bound}}$$

$$((s_0 \cdot M^2 + A \cdot (M + 1)) \bmod 2^{48}) \gg 17 = \Delta z \pmod{\text{bound}}$$

Let's apply the bitshift expansion formula here:

$$((s_0 \cdot M + A) \bmod 2^{48}) - r_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot \text{bound}}$$

$$((s_0 \cdot M^2 + A \cdot (M + 1)) \bmod 2^{48}) - r_2 = 2^{17} \cdot \Delta z \pmod{2^{17} \cdot \text{bound}}$$

where:

- $r_1, r_2 \in [0, 2^{17} - 1]$

Let's apply the new definition of s_0 here:

$$(((2^{35} \cdot (w_h + D_h + c_0) + e_0) \cdot M + A) \bmod 2^{48}) - r_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot \text{bound}}$$

$$(((2^{35} \cdot (w_h + D_h + c_0) + e_0) \cdot M^2 + A \cdot (M + 1)) \bmod 2^{48}) - r_2 = 2^{17} \cdot \Delta z \pmod{2^{17} \cdot \text{bound}}$$

where:

- $e_0 = (w + D) \oplus M \pmod{2^{35}}$

This is getting pretty messy. Let's split s_0 such that $s_0 = X + Y$. Let's label $X = 2^{35} \cdot (D_h + c_0) + e_0$ and $Y = 2^{35} \cdot w_h$. With that, we get:

$$((Y \cdot M + X \cdot M + A) \bmod 2^{48}) - r_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot \text{bound}}$$

$$((Y \cdot M^2 + X \cdot M^2 + A \cdot (M + 1)) \bmod 2^{48}) - r_2 = 2^{17} \cdot \Delta z \pmod{2^{17} \cdot \text{bound}}$$

Now, we can split off the term containing w_h using property 3:

$$((Y \cdot M) \bmod 2^{48}) + ((X \cdot M + A) \bmod 2^{48}) - 2^{48} \cdot c_1 - r_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot \text{bound}}$$

$$((Y \cdot M^2) \bmod 2^{48}) + ((X \cdot M^2 + A \cdot (M + 1)) \bmod 2^{48}) - 2^{48} \cdot c_2 - r_2 = 2^{17} \cdot \Delta z \pmod{2^{17} \cdot \text{bound}}$$

Now, we'd like to get rid of r_1 and r_2 . Thankfully, we can do something very similar to what we did, when defining the weak condition.

Notice that:

- $r_1 = ((s_0 \cdot M + A) \bmod 2^{48}) \bmod 2^{17} = (s_0 \cdot M + A) \bmod 2^{17}$
- $r_2 = (s_0 \cdot M^2 + A \cdot (M + 1)) \bmod 2^{17}$

Remember that we can represent $s_0 = X + Y$:

- $r_1 = ((X + Y) \cdot M + A) \bmod 2^{17}$
- $r_2 = ((X + Y) \cdot M^2 + A \cdot (M + 1)) \bmod 2^{17}$

Now, remember that $Y = 2^{35} \cdot w_h$, meaning, it's zero under $\bmod 2^{17}$. So, under r_1 and r_2 we can drop Y entirely:

- $r_1 = (X \cdot M + A) \bmod 2^{17}$
- $r_2 = (X \cdot M^2 + A \cdot (M + 1)) \bmod 2^{17}$

Notice that we have $X \cdot M + A$ and $X \cdot M^2 + A \cdot (M + 1)$ in the equations already. This means, we can apply our bitshift definition here to get rid of r_1 and r_2 terms. The second equation is a bit messy, we'll deal with it later. So, for the first equation:

$$((Y \cdot M) \bmod 2^{48}) + 2^{17} \cdot (((X \cdot M + A) \bmod 2^{48}) \gg 17) - 2^{48} \cdot c_1 = 2^{17} \cdot \Delta x \pmod{2^{17} \cdot \text{bound}}$$

We can tell that basically every term here has a factor of 2^{17} next to it. Except for the first one. But, knowing that $Y = 2^{35} \cdot w_h$, using our modulo factoring property, we'll claim that we can factor out 2^{17} out of the mod bracket. So, let's do that and immediately cancel out each factor of 2^{17} (I am running out of space here lmao)

$$((2^{18} \cdot w_h \cdot M) \bmod 2^{13}) + (((X \cdot M + A) \bmod 2^{48}) \gg 17) - 2^{31} \cdot c_1 = \Delta x \pmod{\text{bound}}$$

For the second equation:

$$((2^{18} \cdot w_h \cdot M^2) \bmod 2^{13}) + (((X \cdot M^2 + A \cdot (M + 1)) \bmod 2^{48}) \gg 17) - 2^{31} \cdot c_2 = \Delta z \pmod{\text{bound}}$$

where:

- $X = 2^{35} \cdot (D_h + c_0) + e_0$

So, for this condition, the only value that makes sense to bruteforce is e_0 . From e_0 , we can recover the low bits of the worldseed (let's call that w_l). Now, remember that:

$$e_0 = (w_l + D) \oplus M \pmod{2^{35}}$$

Let's apply XOR M to both sides:

$$e_0 \oplus M = w_l + D \pmod{2^{35}}$$

Now, we can get an expression for w_l :

$$w_l = (e_0 \oplus M) - D \pmod{2^{35}}$$

w_l is the same for each equation. So, when bruteforcing e_0 , to prematurely check our partial seeds, we will convert the first constraint's e_0 candidate to w_l and with that generate a corresponding e_0 for the rest of the constraints.

Conclusion

So, with this algebraic nightmare, we were able to split the structure seed into two parts, the highest 13 bits and the lowest 35. For the lowest 35 bits we were able to derive conditions to filter them out without having to reconstruct the full seed.

Now, separately, these two constraints are pretty useless. The weak condition doesn't filter out enough candidates and the strong condition introduces $c_0, c_1 \in \{0, 1\}$, which raises the search space by a factor of 4.

In combination though, if we first apply the weak condition and then the strong condition, we are effectively compensating for the weak condition's inability to filter out many candidates and for strong condition's increase of search space.

It also looks like we could take this approach a step further, to decompose the expression $s_0 = (w + D) \oplus M \pmod{2^{48}}$ into consecutive regions, where $w + D$ bits are flipped by the XOR and where they are not.