ANNOUNCEMENTS

Cryptographic description of Zerocoin attack

By Reuben Yap | April 30, 2019 | No Comments

Following our earlier disclosure of the Zerocoin vulnerability that Zcoin discovered, to the best of our knowledge all major projects utilizing Zerocoin have either used sporks to disable Zerocoin or have rolled out an emergency patch to disable Zerocoin and as such this attack will no longer work on them.

We are releasing this information to assist those who wish to explore fixing Zerocoin and for projects to ascertain whether they've been an target of this attack.

Cryptographic Flaw Description

To spend Zerocoin you need to do the following:

1. Public value of minted coin `C = g^S * h^r (mod p)` is taken and two Pedersen commitments are made to `C` under different groups (`C1`

3. `P2` = ZK proof that `C2` is committed to the value that is contained in the RSA accumulator
4. `P3` = ZK proof that `C1` is committed to the value which itself is a commitment to the serial `S`
5. Output `S`, `C1`, `C2`, `P1`, `P2`, `P3`
6. For proof creations values `r`, `C`, `w` (witness) are used but not disclosed. The proof is tied to the openly known `A` (accumulator value at the time of the spend)

The weak link here is `P1` proof. Although it is an essential cryptographic building block of the Zerocoin protocol, its description and even existence is omitted from all the papers but it is described in the source code of libzerocoin since its inception.

It operates on two groups: {generators `g1`, `h1`, modulus `p1`, order `G1`} and {`g2`, `h2`, `p2`, `G2`}. This construct is supposed to prove that two commitments `A = g1^F1 * h1^R1 (mod p1)` and `B = g2^F2 * h2^R2 (mod p2)` are made to the same value i.e. `F1 = F2 = m`

Proof creation:

1. Create `r1`, `r2`, `r3` – random values much bigger than group orders and moduli
2. Create two helper commitments `T1 = g1^r1 * h1^r2 (mod p1)` and `T2 = g2^r1 * h2^r3 (mod p2)`
3. Calculate `challenge = Hash(A || B || T1 || T2)`
4. `S1 = r1 + (m * challenge)`, `S2 = r2 + (R1 * challenge)`, `S3 = r3 + (R2

Verification:

6. Calculate `T1 = g1^S1 * h1^S2 * A^(-challenge) (mod p1)` and `T2 = g2^S1 * h2^S3 * B^(-challenge) (mod p2)`

7. `hash = Hash(A || B || T1 || T2)`

8. Compare `hash` to `challenge`

The problem here is `m` used in step 4 can be relatively large but our group orders are smaller. So we can choose value of `F` satisfying the following conditions:

`F = F1 (mod G1)`

`F = F2 (mod G2)`

and use it as `m` in step 4. Then the proof will still verify despite two commitments are made to different values

Now to forge a coin attacker creates `P2` proof from a legitimately minted coin then mints a new coin off the books with different serial and uses it to create `P3` proof. Then forges `P1` to prove that `P2` and `P3` refer to the same coin even though they don't.

To sum it up: attacker needs at least one real coin but he can create as many spends as he wants out of it. These fake spends will be indistinguishable from genuine ones.

How to ascertain if your chain has been a target of this attack

bits and hence we were able to assess the maximum damage.

Again, we wish to highlight that **this attack can be done without leaving a fingerprint.**

Workaround for Discussion [NOT TO BE USED WITHOUT FURTHER ANALYSIS]

The workaround that we are proposing below is a band aid solution has not been subject to any sort of cryptanalysis. It is meant solely to stimulate discussion for those who wish to explore fixing Zerocoin. It should not be used unless it has been analyzed with proper scrutiny and we believe a better solution would be to replace the P1 proof entirely. We offer no warranties on this fix.

First let's modify the attack to make it even more powerful. Instead of using `F` we're using `F'` with the following properties:
`F' = F1 * challenge (mod G1)`
`F' = F2 * challenge (mod G2)`
and then calculate `S1 = r1 + F'`. You can calculate `F'` using Chinese Remainder Theorem and bit size of `F'` is the sum of bit sizes of two group orders: `bitsize(F') = bitsize(G1) + bitsize(G2)`. It is possible to bring down bit size of `F'` further by using brute force until required number of leading bits in `F'` turns to zero. Minimum possible bit size of `S1` is the same as `F'`

the following formula:

`bitsize(r1) = bitsize(max(p1, p2, G1, G2)) + bitsize(challenge) + K` where `K` is security parameter (512 in `libzerocoin`) used to mask non-even distribution of `S1` and thus make proof zero-knowledge. Also in current implementation verification allows `S1` to be up to 64 times bigger in bit size than `r1` (100000+ bits)

Suggested way to fix the problem is to allow only absolute minimum of bits in `S1`. First of all there is no need to get `r1` as big as group modulus, group order is enough. Next we can bring down parameter `K` just to few bits. And finally we can reduce number of bits in the hash function to reduce challenge size. That brings us to `bitsize(r1) = bitsize(max(G1, G2)) + bitsize(challenge) + K`

To be secure it has to be much smaller (at least 128 bits, close to 256 is more desirable) than size of `F'` used in the attack `max(bitsize(G1), bitsize(G2)) + bitsize(challenge) + K < bitsize(G1) + bitsize(G2)`

With current parameters it's impossible to implement because group orders are 1024 and 257 bits in size and challenge is 256 bits long. It becomes possible only if we change the parameters so group order `G2` is much bigger and close to 512 bits. Alternatively we can bring down number of bits in `challenge` to 128 although it's a worse solution. And `K` should be in the ballpark of 16-32.

Further Di

Author

Reuben Yap

OUR TECHNOLOGY

Zerocoin technology ensures that Zcoin's transactions are kept completely private

MORE ABOUT ZEROCOIN

JOIN US ON DISCORD

Join us on discord to stay updated on our project. Ask us any questions and learn about our bug bounty program.

JOIN US ON DISCORD

GET IN TOUCH

Have any questions? Want to get involved with the Zcoin Project? We'd love to hear from you.

TEAM@ZCOIN.IO

NEWSLETTER SUBSCRIPTION

If you want to subscribe to our newsletter, please submit the form below.

Email* :

SUBSCRIBE