I’m pessimistic that anyone made it this far, but if somebody is actually reading this - congrats! it’s pretty impressive to learn & implement this attack and even a variation of it so quickly :)

Again, you can find the article [here](https://www.usenix.org/system/files/usenixsecurity24-shagam.pdf)

Forget all optimizations. This time we tackle different difficulties - what if the oracle “lies”?

Formally speaking, whenever oracle replies whether some message is PCKS conf’ or not, there is a probability the answer is flipped.

The most native approach to tackle such a problem, is to repeat the oracle call, i.e. , k times for some k we choose. This indeed will reduce the chance, and at some point it would be negligible enough to just ignore.

However, such an approach isn’t very efficient, as this would take k times the original time. (for k=50, it would mean the program is 50 times slower !!)

So we’ll consider different ideas, and then build a different solution which takes ideas from both approaches.

As p is considerably small, we won’t mind missing some PCKS conf’ message, as this just requires more computation; but as p is small, the effect on the performance is negligible.

However, a “false positive” is critical, as we would choose the wrong range, one where m isn’t nesting at, and overall we won’t converge into an answer/converge into a wrong answer.

The researchers weaponized a statistical property of the attack - if the attack receives a “false positive”, thus moving into a wrong range, then it would (probably) get a 100+ continuous false answers from the oracle. On the other hand this “false streak” isn’t common in “true positives”.

So the idea is the following:

We’ll maintain “snapshots” - stages of the attack, by maintaining pairs

. notice one can continue an attack session based on such a pair.

When we encounter a “lose streak” of 100, roll back 15 snapshots.

Continue the attack.

Worth mentioning the goal of the attack isn’t to ensure a one successful attack, but instead running several attacks in a window of time, and to know most of them would succeed. Citing the researchers: “**We don’t want to ensure that all at-**

**tacks succeed (no false positive), but maximize the percentage**

**of attacks that finish successfully below a fixed query number**

**Threshold.”**

Thus, we recommend running your implementation 2-3 times in case you’re unlucky and the attack fails for reasons beyond your control.

Overall, the attacks differs from the original by:

We query the oracle 6 times, to enhance preciseness.

Maintain snapshots

Roll back 15 stages ago. (“going back in time”)

You’ll find pseudo next page.

For step 2.A - as this part is very time consuming and heavy on the computation, we’ll just ensure the message is PCKS conf’ by ensuring that out of 12 oracle calls to O, it succeeds in at least 10.

For step 2.B - make sure it succeeds in at least 2 out of 6.

For 2.C - query the oracle. if it succeeds in >= 2, if it fails all, reject and if it succeeds in exactly 1 re-query and ensure it succeeds in at least 2.

If at any point we encounter a loss streak of length 100, go back 15 snapshots ago, and set M and S\_i accordingly.