Vulnerability Report

Fireblocks' research team has discovered a vulnerability in the specification of two popular threshold signature schemes (TSS): GG18 and GG20. These schemes are considered pioneering for the MPC-wallet industry and are widely adopted by companies in the space. The vulnerability was found at the pseudocode level, and all vendors implementing the specification from the paper(s) should be considered vulnerable. The vulnerability can be exploited **to exfiltrate the key.**

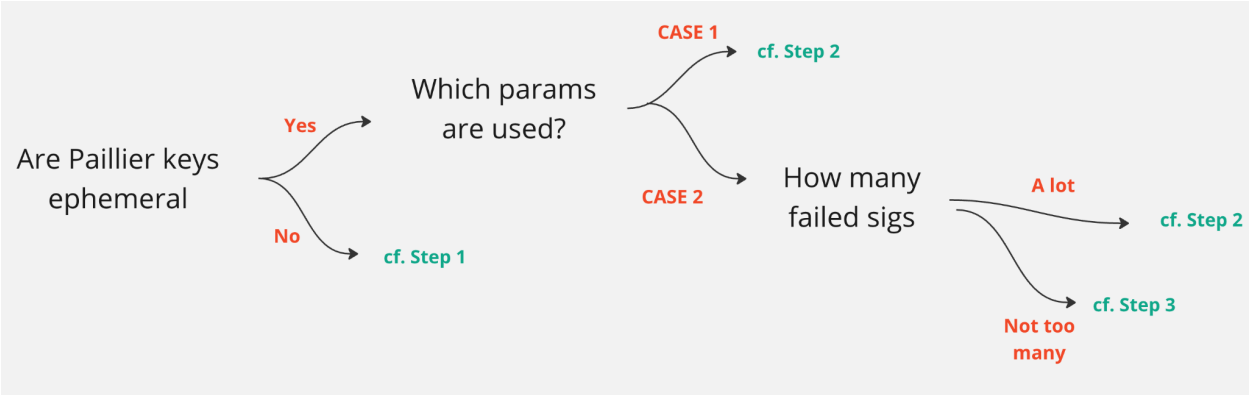
The origin of the vulnerability is that the parties do not check if the Paillier modulus, denoted N, has small factors (a prime of size less than 2**100 is considered small) or that it is a biprime (so it is the product of exactly two primes). If exploited, the vulnerability allows a threat actor interacting with the signatories in the TSS protocol to steal their secret shards and ultimately obtain the master secret key. The severity of the vulnerability depends on the implementation parameters, so different parameter choices give rise to different attacks with varying degrees of effort/resources required to extract the full key. Namely, the distinguishing parameter choices are as follows:

1. Beta Parameter in MTA ~ q^5 (where q is the size of the elliptic curve)
2. Beta Parameter in MTA ~ N
    a. N is not checked for small factors at all (not even 3,5,7,..)
    b. N is checked manually for smallish factors (< 2**12)

Case 1. By using a maliciously-crafted message, the attacker obtains leakage of the private key shards of the parties by interacting with the signatories in the first round of the TSS protocol. To obtain the full private key, it suffices to iterate the attack **sixteen** times. The malicious actor does not need to actively control a party or have prior knowledge of the key shards (though it requires some privileged access, e.g. a man-in-the-middle approach is possible for this attack).

Case 2. In this case, the attacker actively controls one of the signatories (so it also needs to "know" the corresponding secret shard) and it interacts with the signatories for the entire TSS protocol. By using a maliciously-crafted message, the attacker obtains leakage of the private key shards whenever the signature generation process terminates successfully. The total number of signature-generation attempts depends on the number of parties and the Paillier modulus N. For two parties, the number of (failed) signatures ranges from 200K (for case 2.a) to 10M (for case 2.b).

Forensic Analysis & Mitigation:



Step 1. Manually check that the Paillier key does not have small factors; sophisticated algorithms can find factors as big as 2**80 fairly easily. If such a factor is found, conclude that **the vulnerability has been exploited**. If no such factor is found, it can be reasonably assumed that no attack occurred.

Step 2. There is a possibility that the key has been exfiltrated. If possible, retrieve previous Paillier keys and follow Step 1.

Step 3. There is a possibility that the attack is in progress, or even that the key has been exfiltrated under a different attack (that eludes us). If possible, retrieve previous Paillier keys and follow Step 1.

Immediate mitigation moving forward. Follow Step 1. for all new Paillier keys.

Remediation:

To fix the issue, we recommend using a key-generation process that can detect subverted Paillier moduli. Specifically, we recommend implementing the ZK proofs from CMP20 (or CGGMP21) which detects moduli with factors smaller than 2**256.