Response to Audit Findings (2024-08-27)

HOPR Association

1. [High] Incorrect implementation of ecAdd when a is not 0

- Status: Code Change
- Description of changes:
  - Updated the implementation of the calculation of the lambda value in point-doubling
  - Added a test for elliptic curve doubling when a is not zero (i.e. 2) and P is (0, 1)
- Commit hash (if applicable): 1e8b9cf68bbe638e6a4159646e7b2e20739d395f

To exploit this error and manipulate the ticket-winning probability in favor of the attacker, the attacker would need to brute-force a highly specific payload (i.e., the ticket information, particularly by mutating the ticket challenge value) during its creation. The goal would be to ensure that the result returned by the `hash_to_curve` function (after hashing with a domain separator) produces two identical scalars, leading to faulty point doubling in the VRF computation.

As a result, the ticket recipient might fail to redeem a ticket that they believed to be a winning one, thereby losing its value. This failed ticket could either be an individual ticket or part of an aggregated batch. However, the computational cost of such an attack outweighs the potential gain from a single ticket. To successfully attack aggregated tickets, the attacker would need to observe the victim's aggregation pattern and manipulate the packets so that the faulty ticket appears as the first one in the aggregation batch after mixers.

Exploiting this bug would require attackers to find a random “challenge” value that can be computed to a suitable ticket hash, which has a complexity of 2^160 since the challenge is 20 bytes long. Therefore, we recognize the risk posed by this issue

in the deployed contract, have introduced a fix in the source code, and will implement a fixed version for the next HOPR network deployment.

2. [Low] “Channels” does not completely follow ERC1820

- Status: Acknowledgment and documentation
- Description of changes:
  - Added code comments for the constructor function to specify that the ERC1820 implementation for the `ERC777TokensRecipient` is only for the Channels contract itself
- Commit hash (if applicable): d1aa68fef7e08f65afb1ece6e7a34464ab1daa5f

As stated in EIP-1820, the `ERC1820Implementer` contract **must** implement the `canImplementInterfaceForAddress` function if it serves as the implementer of an interface for any address other than itself. However, the `HoprChannels` contract is designed to serve only for itself. Therefore, calling `setInterfaceImplementer` is sufficient in this context.

To restrict the implementation strictly to itself while implementing `canImplementInterfaceForAddress`, the contract would also need to call `_registerInterfaceForAddress` for itself during deployment, which would increase the gas cost compared to the current implementation. However, by leaving `canImplementInterfaceForAddress` unimplemented, any calls originating from contracts other than itself will automatically revert, thanks to the code within the `setInterfaceImplementer` function:

```
Unset
if (_implementer != address(0) && _implementer != msg.sender) {
      require(
            ERC1820ImplementerInterface(_implementer).canImplementInterfaceFor
      Address(_interfaceHash, addr) == ERC1820_ACCEPT_MAGIC,
```

```
                "Does not implement the interface"
        );
    }
```

3. [Info] Cryptographic implementation states using p-1 but uses p-2

- Status: Code Change
- Description of changes:
    - Update the code comment. The inverse is calculated with `a ^ (P - 2) === a ^ (-1) mod P` in a finite field, according to Fermat's Little Theorem.
- Commit hash (if applicable): 1f70d634fa636f9b27ae5116db6acba7849a7b3b

4. [Info] Revoking a target does not delete its capabilities

- Status: Acknowledgment and documentation
- Description of changes:
    - Added code comments on the revokeTarget function in both `HoprCapabilityPermissions` library and HoprNodeManagementModule.
- Commit hash (if applicable): 1d62e269d58314b533957173a24ed2e3270728fc

5. [Info] `ensureNodeIsSafeModuleMember` can be tricked

- Status: Code Change
- Description of changes:
    - Removed ensureNodeIsSafeModuleMember function and their respective references
    - Added code comments in `addNodeSafe` and `deregisterNodeBySafe` to signify removing such a check.
    - Adjust unit test accordingly

- Commit hash (if applicable): f6a17004841555478c446baf5445e8ad23cac8fd

The original intent of including this check in the module was to prevent unexpected removal actions. However, it does not provide the anticipated benefits, as Safe operates under the assumption that it should be responsible for node admin transactions, given its privilege for advanced node management. With two issues[1][2] reported concerning this function, we have decided to remove it.

6. Optimisations and miscellaneous

- Status: Code Change
- Description of changes:
  - Replace `_setupRole` with `_grantRole`
- Commit hash (if applicable): 6334cfb3c5bbd2dc4e3b5f2f06c84ad6ff6ffe2a

[1] https://github.com/hats-finance/SafeStaking-by-HOPR-0x607386df18b663cf5ee9b879fbc1f32466ad5a85/issues/23

[2] https://github.com/hoprnet/hoprnet/issues/6466