Understanding Optimized BLS Multisignatures on EVM

Arijit Dutta

May 15, 2023

Background

- Pairing: $e(\mathbb{G}_1, \mathbb{G}_2) \rightarrow \mathbb{G}_T$
- For $P, S \in \mathbb{G}_1$ and $Q, R \in \mathbb{G}_2$, we have

$$
\begin{aligned}
e(P, Q+R) &= e(P, Q) * e(P, R) \\
e(P+S, R) &= e(P, R) * e(S, R) \\
e(aP, bQ) &= e(P, Q)^{ab} = e(P, aQ)^{b} = e(bP, aQ)
\end{aligned}
$$

- Let $H \in \mathbb{G}_1, G \in \mathbb{G}_2$ are some generators, same scalar field $\mathbb{F}_r$
- Signer($sk \in \mathbb{F}_r, pk = skG \in \mathbb{G}_2$), message $= m \in \mathbb{F}_r$, $\mathbb{H}(m) \in \mathbb{G}_1$
- BLS signature: $sk\mathbb{H}(m) \in \mathbb{G}_1$
- Verification: $e(\sigma, -G) * e(\mathbb{H}(m), pk) \stackrel{?}{=} 1$
- Say $H(m) = xH$, first term $= e(H, G)^{-(sk*x)}$, second term $= e(H, G)^{sk*x}$

Background

- Pairing: $e(\mathbb{G}_1, \mathbb{G}_2) \to \mathbb{G}_T$
- For $P, S \in \mathbb{G}_1$ and $Q, R \in \mathbb{G}_2$, we have

$$e(P, Q + R) = e(P, Q) * e(P, R)$$
$$e(P + S, R) = e(P, R) * e(S, R)$$
$$e(aP, bQ) = e(P, Q)^{ab} = e(P, aQ)^{b} = e(bP, aQ)$$

- BLS Multisignature: all signers need to sign the same message m
- Let three signers: $(sk_1, pk_1), (sk_2, pk_2), (sk_3, pk_3)$, $pk_i = sk_i G \in \mathbb{G}_2, i \in [3]$
- The signers send their signatures: $\sigma_i = sk_i \mathbb{H}(m), i \in [3]$ to a submitter
- Submitter verifies/computes
  - proof of possession: $e(-\sigma_i, G) * e(\mathbb{H}_2(pk_i), pk_i) \stackrel{?}{=} 1, i \in [3]$
  - aggregated public key: $apk = pk_1 + pk_2 + pk_3$
  - signature: $\sigma = \sigma_1 + \sigma_2 + \sigma_3$
- Verification: $e(\sigma, -G) * e(\mathbb{H}(m), apk) \stackrel{?}{=} 1$

Optimised Multisignatures

- Verification equation

$$e(\sigma, -G) * e(\mathbb{H}(m), apk) \stackrel{?}{=} 1$$

- EVM suitable because:
  - $-G$ can be precomputed
  - $\mathbb{H}(m)$ is efficiently computed in $\mathbb{G}_1$ (cofactor $= 1$ in $\mathbb{G}_1$, around 254 bits in $\mathbb{G}_2$)
- Drawback: public keys are in $\mathbb{G}_2$, aggregation/addition is costly (30k gas as compared to 500 in $\mathbb{G}_1$)
- Proposal:
  - Do the costly addition in $\mathbb{G}_2$ off-chain
  - Verify it on-chain using a newly introduced public key in $\mathbb{G}_1$

Optimised Multisignatures (contd)

- One public key in $\mathbb{G}_1$, in addition to that in $\mathbb{G}_2$

$$pk_1 = sk * H \in \mathbb{G}_1$$
$$pk_2 = sk * G \in \mathbb{G}_2$$

- Three participants: Alice, Bob, Charlie

$$apk = pk_{2,\mathrm{Alice}} + pk_{2,\mathrm{Bob}} + pk_{2,\mathrm{Charlie}}$$
$$P_1 = pk_{1,\mathrm{Alice}} + pk_{1,\mathrm{Bob}} + pk_{1,\mathrm{Charlie}}$$

- apk is computed by the submitter (offchain computation)
- P_1 is computed by the contract (onchain computation)
- To check apk is correct,

$$e(P_1, G) * e(-H, apk) \stackrel{?}{=} 1,$$

$e(H, G)^{(sk_{\mathrm{Alice}} + sk_{\mathrm{Bob}} + sk_{\mathrm{Charlie}})}$ and its inverse in G_T

Protocol

- Check σ signed by *apk*, check the validity of *apk* with P_1, combine them with a random scalar
- $\alpha = \text{hash-to-scalar}(\sigma, m, P_1, apk)$, to check

$$
\begin{aligned}
& e(\sigma, -G) * e(\mathbb{H}(m), apk) * (e(P_1, G) * e(-H, apk))^{\alpha} = 1 \\
\implies & \underbrace{e(\sigma, -G)} * e(\mathbb{H}(m), apk) * \underbrace{e(\alpha P_1, G)} * e(-\alpha H, apk) = 1 \\
\implies & e(\sigma - \alpha P_1, -G) * e(\mathbb{H}(m) - \alpha H, apk) = 1
\end{aligned}
$$

- Further optimisation: Work with precomputed *apk* and subtract public keys corresponding to the absent signer

References

1. https://geometry.xyz/notebook/Optimized-BLS-multisignatures-on-EVM
2. https://eth2book.info/capella/part2/building_blocks/signatures/

Thank you for your attention. Questions?