# SECURITY ANALYSIS OF SEAGKAP

## 1. INTRODUCTION

In this section, we prove the correctness of the protocol and provide a formal security proof in the Random Oracle Model (ROM) [15]. Authentication is proven secure under the EUF-CMA [16], and confidentiality is proven secure under the Real-Or-Random (ROR) model [17]. In addition, we conduct an informal security analysis on replay resistance, forward and backward secrecy.

## 2. FORMAL PROOF

### 2.1. Authentication Proof

Let $\mathbb{G}$ be a cyclic group of prime order $q$ generated by $G$. Each participant $P_i$ holds a long-term signing key pair $(\mathrm{sk}_i, \mathrm{pk}_i = \mathrm{sk}_i G)$. The protocol consists of two authenticated message rounds as follows.

(a) **Multicast first-round message**. Each $P_i$ samples randomness $r_i$, computes $R_i = r_i G$, and forms $h_i = \mathrm{hash}(ID_i \parallel R_i \parallel ID\_list \parallel Time)$, $\quad s_i = \mathrm{sk}_i h_i + r_i \pmod q$. The message $m_i = (ID_i, R_i, s_i)$ is multicast to neighbors. Verification in the enclave is compressed into a single neighbor-check equation $r_i S_i \cdot G \overset{?}{=} Y_i + X_i$, where $S_i = s_{i+1} - s_{i-1}$, $Y_i = r_i h_{i+1} \cdot pub_{i+1} - r_i h_{i-1} \cdot pub_{i-1}$, and $X_i = r_i \cdot (R_{i+1} - R_{i-1})$.

(b) **Broadcast second-round message**. Each $P_i$ computes $X_i$, then $H_i = hash(ID_i \parallel X_i \parallel R_i \parallel ID\_list \parallel Time)$, $\quad t_i = sk_i H_i + r_i$. The broadcast message is $M_i = (ID_i, X_i, R_i, t_i)$. Applications aggregate all but one party's signatures into $T_{-i} = \sum_{\substack{j=1 \\ j\neq i}}^{n} T_j$, $R_{-i} = \sum_{\substack{j=1 \\ j\neq i}}^{n} R_j$, and $W_{-i} = \sum_{\substack{j=1 \\ j\neq i}}^{n} H_j \cdot pub_j$.

We prove that accepting forged messages in either round is equivalent to forging Schnorr signatures.

The adversary $\mathcal{A}$ is polynomial-time, controls the communication channel, and may inject, modify, or replay protocol messages. Hash functions are modeled as random oracles.

(a) **Forge**$_1$: An honest instance accepts in Round 1 while at least one neighbor's message is a forgery.

(b) **Forge**$_2$: An honest instance accepts in Round 2 while the aggregate set includes at least one forged signature.

Our goal is to bound $\Pr[\text{Forge}_1]$ and $\Pr[\text{Forge}_2]$.

**Lemma 1 Neighbor Check Completeness.** The compressed verification equation in Round 1 is equivalent to verifying both Schnorr signatures from the two neighboring participants individually. Therefore, except with negligible probability in the ROM, acceptance of the compressed equation implies that both neighbors' signatures are valid.

*Proof.* The compressed verification equation in Round 1 can be algebraically reduced to the standard Schnorr verification equations of the two neighbors, $s_{i+1} \cdot G = R_{i+1} + h_{i+1} \cdot pk_{i+1}$ and $s_{i-1} \cdot G = R_{i-1} + h_{i-1} \cdot pk_{i-1}$.

Thus, acceptance of the compressed check is equivalent to accepting both neighboring signatures simultaneously. The only possible discrepancy arises if the outputs of the random oracle queries $h_{i+1}$ and $h_{i-1}$ happen to cancel in such a way that an invalid signature is nevertheless accepted. However, since the random oracle produces independent and uniformly distributed outputs, the probability of such a collision is negligible in the security parameter. □

**Lemma 2 Aggregate Check Completeness.** The Round 2 aggregate verification equation is equivalent to the standard Schnorr aggregate signature verification. Therefore, except with negligible probability in the ROM, acceptance of the aggregate equation implies that all individual signatures are valid.

*Proof.* The Round 2 aggregate equation is equivalent to the standard Schnorr aggregate verification

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} t_j \cdot G \overset{?}{=} \sum_{\substack{j=1 \\ j\neq i}}^{n} (R_j + H_j \cdot pk_j).$$

Thus, acceptance of the aggregate verification implies that the combined signatures correspond to valid individual Schnorr signatures. Suppose, for contradiction, that at least one term is forged. In that case, the left-hand side and right-hand side would only balance if the forged component were perfectly canceled by independent random oracle outputs. Since random oracle outputs are independent and uniformly distributed, the probability of such a cancellation is negligible in the security parameter. Hence, acceptance implies that all signatures are valid. □

**Theorem 1 authentication proof for first-round message.** If a PPT adversary $\mathcal{A}$ wins Forge$_1$ with non-negligible probability, there exists a PPT algorithm $\mathcal{F}$ that breaks EUF-CMA security of Schnorr signatures with non-negligible probability.

*Proof.* The proof follows a standard game-hopping argument.

Game $G_0$. Real protocol execution. Let probability of success be $\Pr_0$.

Game $G_1$. The reduction $\mathcal{F}$ randomly selects a target signer $P^*$ (one of the two neighbors) and sets the challenge public key as pk$^*$. All signature queries for $P^*$ are answered by forwarding them to the signing oracle provided by the EUF-CMA challenger, while signatures for other parties are generated honestly using their known secret keys.

The random oracle $H(\cdot)$ is simulated via lazy sampling: whenever $\mathcal{M}$ queries $H(x)$ for a new input $x$, $\mathcal{F}$ returns a freshly chosen random value and records it; repeated queries are answered consistently from the record. Since this simulation is indistinguishable from the real execution, the adversary's view remains unchanged and therefore the success probability satisfies $\Pr_1 = \Pr_0$.

Game $G_2$. By Lemma 1, acceptance implies that both neighbor signatures are valid. If Forge$_1$ occurs, then at least one valid signature $(M^*, s^*)$ for $P^*$ exists that was never produced via the signing oracle.

Reduction. The pair $(M^*, s^*)$ constitutes a valid EUF-CMA forgery. Thus, the success probability of $\mathcal{M}$ in forging the first-round message can be directly bounded by the advantage of $\mathcal{C}$ against the underlying Schnorr signature scheme:

$$\Pr[\text{Forge}_1] \leq (n \cdot S) \cdot \text{Adv}^{\text{EUF-CMA}}_{\text{Schnorr}} + \text{negl}(\lambda),$$

where $n$ is group size and $S$ is session bound. □

**Theorem 2. Authentication proof for second-round message.** If a PPT adversary $\mathcal{A}$ wins Forge$_2$ with non-negligible probability, there exists a PPT algorithm $\mathcal{F}$ that breaks EUF-CMA security of aggregate Schnorr signatures with non-negligible probability.

*Proof.* The proof also follows a standard game-hopping argument.

Game $H_0$. Real protocol execution. Let the probability of success be $\Pr'_0$.

Game $H_1$. The reduction $\mathcal{F}$ randomly selects a target signer $P^*$ and sets the challenge public key as pk$^*$. All signature queries for $P^*$ are answered by forwarding them to the signing oracle provided by the EUF-CMA challenger, while signatures for other parties are generated honestly using their known secret keys. The random oracle $H(\cdot)$ is again simulated via lazy sampling. Since this simulation is indistinguishable from the real execution, the adversary's view remains unchanged and therefore the success probability satisfies $\Pr'_1 = \Pr'_0$.

Game $H_2$. By Lemma 2, acceptance of the aggregate verification equation implies that all individual signatures are valid. If Forge$_2$ occurs, then there must exist a new message $M^*$ with a valid signature $t^*$ for $P^*$ that was never obtained via the signing oracle.

Reduction. The pair $(M^*, t^*)$ constitutes a valid EUF-CMA forgery against the aggregate Schnorr signature scheme. Thus, the success probability of $\mathcal{M}$ in forging the second-round message can be directly bounded by the advantage of $\mathcal{C}$ against the underlying aggregate Schnorr signature scheme:

$$\Pr[\text{Forge}_2] \leq (n \cdot S) \cdot \text{Adv}^{\text{EUF-CMA}}_{\text{Agg-Schnorr}} + \text{negl}(\lambda),$$

where $n$ is group size and $S$ is session bound. □

## 2.2. Confidentiality Proof

**Theorem 3. Confidentiality proof in the ROR model.** If a PPT adversary $\mathcal{M}$ wins Forge$_{\text{ror}}$ with non-negligible probability, there exists a PPT algorithm $\mathcal{F}$ that breaks the DDH assumption in $\mathbb{G}$ with non-negligible probability.

*Proof.* The proof proceeds via a sequence of games in the random oracle model (ROM).

Game $J_0$. Real ROR experiment. The adversary interacts with the protocol in the actual ROR experiment, including Send, Reveal, Corrupt, and Test queries. The probability of success in distinguishing the test key is denoted by $\text{Adv}_0$.

Game $J_1$. Simulation of the KDF as a random oracle. The key derivation function is replaced with a table-driven random oracle: on the first query of a new input, output a uniformly random $\kappa$-bit string and record it; on repeated queries, return the same value. This is perfectly consistent with the ROM assumption, and therefore the adversary's view is unchanged. Hence, $\text{Adv}_1 = \text{Adv}_0$.

Game $J_2$. Embedding an adjacent-DH sum challenge. By the telescoping identity, the key material is $K = \sum_{i=1}^{n} r_i r_{i+1} \cdot G$. The reduction $\mathcal{F}$ receives an ADS challenge $(G, \{R_i = r_i \cdot G\}_{i \in [n]}, T)$ where $T$ is either $\sum_{i=1}^{n} r_i r_{i+1} \cdot G$ or a uniform random element of $\mathbb{G}$. It embeds this $T$ as the session's $K$ and otherwise simulates the transcript faithfully (all signatures and random oracles are answered as in $J_1$). If $\mathcal{M}$ distinguishes $J_1$ from $J_2$, then $\mathcal{F}$ distinguishes the ADS challenge, breaking the ADS assumption. Hence,

$$|\text{Adv}_2 - \text{Adv}_1| \;\leq\; \text{Adv}^{\text{ADS}}_{\mathbb{G}}(\mathcal{F}).$$

Game $J_3$. Replacing the test key with uniform randomness. From Game $J_2$, the key material of the target session is $K = T$, where $T$ is either $\sum_{i=1}^{n} r_i r_{i+1} \cdot G$ or a uniform random element of $\mathbb{G}$. We keep simulating all messages and the random oracle as in $J_2$, and we *program* the KDF oracle at the (unique) target input $inp^\star = (K_i, len)$. We continue to answer KDF queries by lazy sampling and program the oracle at $inp^\star$. Conditioned on the event that the adversary never queries the KDF oracle at $inp^\star$, the test-session key can be replaced by a uniformly random $\kappa$-bit string without changing the adversary's view.

Let $\mathsf{Bad}_{J_3}$ denote the event that $\mathcal{M}$ queries the KDF oracle on $\mathsf{inp}^\star$. By the standard random-oracle argument (and the Fundamental Lemma of Game-Playing), the distinguishing gap between $J_2$ and $J_3$ is bounded by the probability of this bad event:

$$\big|\text{Adv}_3 - \text{Adv}_2\big| \;\leq\; \Pr[\mathsf{Bad}_{J_3}] \;\leq\; \frac{q_{\text{KDF}}}{2^\kappa},$$

where $q_{\mathrm{KDF}}$ is the number of queries that $\mathcal{M}$ issues to the KDF oracle and $\kappa$ is the KDF output length.

Reduction. If $\mathcal{M}$ distinguishes the real session key from random with non-negligible probability, then $\mathcal{F}$ can use $\mathcal{M}$ as a subroutine to solve the Adjacent-Diffie–Hellman Sum (ADS) problem in $\mathbb{G}$. Specifically, $\mathcal{F}$ embeds the ADS challenge element $T$ as the target session key material $K_i$. If $\mathcal{M}$ succeeds in distinguishing the session key, then $\mathcal{F}$ outputs that $T$ is the genuine ADS value; otherwise it outputs that $T$ is random. Therefore, the advantage of $\mathcal{M}$ in the ROR experiment is bounded by the advantage of $\mathcal{F}$ in solving ADS plus negligible terms.

Thus, the advantage of $\mathcal{M}$ in breaking confidentiality in the ROR model satisfies

$$\mathrm{Adv}_{\Pi}^{\mathrm{ror}}(\mathcal{M}) \;\le\; \mathrm{Adv}_{\mathbb{G}}^{\mathrm{ADS}}(\mathcal{F}) \;+\; \frac{q_{\mathrm{KDF}}}{2^{\kappa}} \;+\; \mathrm{negl}(\lambda).$$

□

## 3. INFORMAL SECURITY ANALYSIS

### 3.1. Replay Resistance

In each round of the protocol, a timestamp is included as part of the broadcast message. The timestamp ensures that a transmitted packet is valid only within a specific time window. If the timestamp embedded in the messages does not match the verifier's current time, the verification fails and the session is aborted. Hence, a man-in-the-middle adversary cannot simply replay previously intercepted packets to launch a replay attack. Furthermore, the timestamp is embedded within the Schnorr signatures of both rounds. Only a legitimate user holding the corresponding private key can generate a valid signature bound to a fresh timestamp. Since an adversary cannot forge or update the signature by modifying the timestamp, replay attempts are effectively prevented. Therefore, the proposed protocol resists replay attacks.

### 3.2. Forward and Backward Secrecy

In the proposed scheme, the session key incorporates ephemeral randomness generated inside the TEE by each participant. These random values are refreshed dynamically whenever the enclave is invoked. Because the adversary has no access to the internal state of the TEE, it cannot predict the generation or evolution of these random values. As a result, the session keys produced by the protocol are dynamic and protected by the TEE. Even if a man-in-the-middle adversary compromises one session key, it cannot infer past or future session keys established by the same parties. Thus, the proposed protocol achieves forward secrecy and backward secrecy.

### 3.3. Correctness

We now argue that the protocol achieves replay resistance. In the proposed protocol, each participant derives the final session key locally using the broadcast parameters and its own private randomness, eliminating the need for centralized coordination or reliance on a trusted third party. Signature verification in both rounds guarantees that all parties maintain a consistent view of $R_i$ and $X_i$, while any forged or inconsistent intermediate values are detected and discarded during the verification phase. Using the following key computation formula, it is straightforward to prove that the protocol ensures correctness. Consequently, under honest execution, all honest parties compute the same session key, which guarantees both correctness and key consistency.

$$\begin{aligned}
K_i &= nr_i \cdot R_{i-1} + (n-1)\cdot X_i + (n-2)\cdot X_{i+1} \\
&\quad + \cdots + X_{i-2} \\
&= r_i \cdot R_{i-1} + (r_i \cdot R_{i-1} + X_i) \\
&\quad + (r_i \cdot R_{i-1} + X_i + X_{i+1}) + \cdots \\
&\quad + (r_i \cdot R_{i-1} + X_i + \cdots + X_{i-2}) \\
&= r_i \cdot R_{i-1} + (r_i \cdot R_{i-1} + r_i \cdot R_{i+1} \\
&\quad - r_i \cdot R_{i-1}) \\
&\quad + (r_i \cdot R_{i-1} + r_i \cdot R_{i+1} - r_i \cdot R_{i-1} \\
&\quad + r_{i+1} \cdot R_{i+2} - r_{i+1} \cdot R_i) + \cdots \\
&\quad + (r_i \cdot R_{i-1} + r_i \cdot R_{i+1} - r_i \cdot R_{i-1} \\
&\quad + r_{i+1} \cdot R_{i+2} - r_{i+1} \cdot R_i + \cdots \\
&\quad + r_{i-2} \cdot R_{i-1} - r_{i-2} \cdot R_{i-3}) \\
&= r_i \cdot R_{i-1} + r_i \cdot R_{i+1} + r_{i+1} \cdot R_{i+2} \\
&\quad + \cdots + r_{i-2} \cdot R_{i-1} \\
&= r_i \cdot R_{i+1} + r_{i+1} \cdot R_{i+2} + \cdots \\
&\quad + r_{i-2} \cdot R_{i-1} + r_{i-1} \cdot R_i
\end{aligned}$$

where $X_i = r_i \cdot (R_{i+1} - R_{i-1})$, and indices are taken modulo $n$. The right-hand side is a cyclically invariant symmetric sum, and therefore it follows that $K_i = K_j$ for any $i, j$. By the deterministic nature of the key derivation function, it follows that $Key_i = Key_j$, that is, all honest parties derive the same session key.