Efficient and Universally Composable Single Secret Leader Election from Pairings

Dario Catalano[1], Dario Fiore[1,2], Emanuele Giunta[2]

PKC'23

University of Catania, Italy.

IMDEA Software Institute, Madrid, Spain.

Universidad Politecnica de Madrid, Spain.

Consensus from Probabilistic Leader Election

Leader Election $\longrightarrow$ Consensus

Consensus from Probabilistic Leader Election

Probabilistic **Secret** Leader Election → Consensus

Consensus from Probabilistic Leader Election

Probabilistic **Secret** Leader Election → Consensus

election protocol | potential leaders elected | tie breaker

Issues: Potential **forks** and **wasted effort**.

Single Secret Leader Election

[BEHG20] proposed Single Secret Leader Election protocols (SSLE).

✅ **Advantage**: Harder to attack [AC21].

❗ **Disadvantage**: Less efficient than probabilistic elections.

Game-Based Definition

Uniqueness: Each election can have at most one leader.

Fairness: All users have the same probability of winning an election.

Unpredictability: No one can guess the leader identity before she reveals better than randomly.

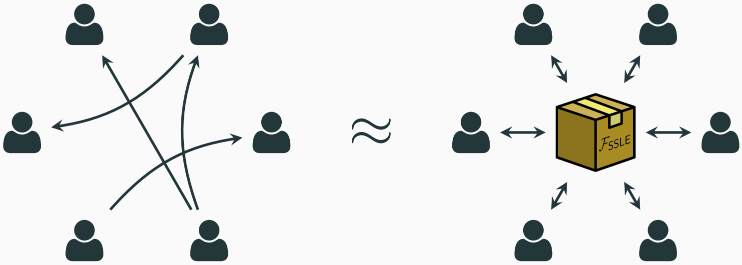
Universal Composability

Game-based security [BEHG20] may fail under composition.

Universal Composability

Game-based security [BEHG20] may fail under composition.

We propose a stronger definition in the **UC model** [Can00].



Real Protocol

Ideal Protocol

Previous Work

	Based on	Off-chain	On-chain	Security	Corruption
[BEHG20]	iO	$O(1)$	$O(1)$	Game-Based*	Static
[BEHG20]	TFHE	$O(t)$	$O(t)$	Game-Based	Static
[BEHG20]	DDH	$O(N)$	$O(N)$	Game-Based	Static
[CFG22]	DDH	$O(N)$	$O(N)$	UC	Adaptive
[LOS22]	DDH	$O(N)$	$O(N)$	Game-Based	Adaptive
[NNHP22]	MPC	$O(N^2)$	$O(1)$	UC	Adaptive
This Work	SXDH	$O(N)$	$O(\log^2 N)$	UC	Static

Our Construction

Predicate Encryption for Keyword Search

A **PEKS** is a Functional Encryption scheme where a key sk_y allows to test if a ciphertext encrypts y or not.

$$\mathsf{Dec}(\mathsf{sk}_y, \mathsf{Enc}(x)) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

Predicate Encryption for Keyword Search

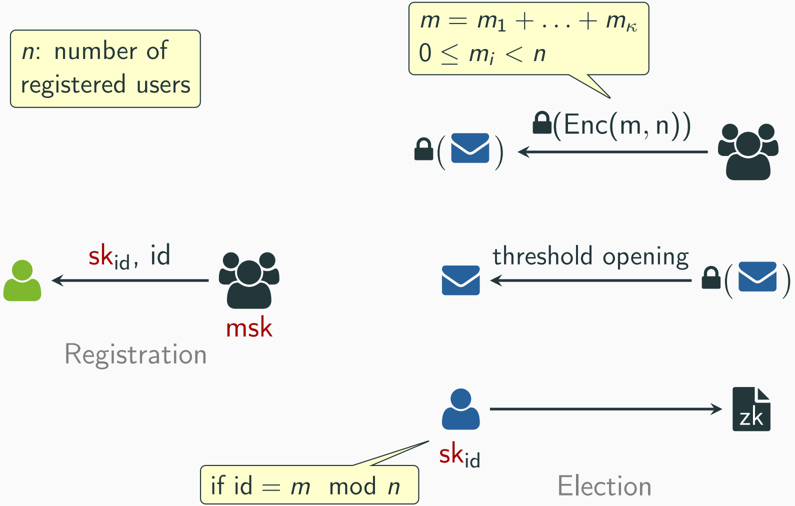
A **PEKS** is a Functional Encryption scheme where a key sk_y allows to test if a ciphertext encrypts y or not.

$$\mathsf{Dec}(\mathsf{sk}_y, \mathsf{Enc}(x)) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

Modular Keyword Search: sk_y reveals if $x = y$ modulo n

$$\mathsf{Dec}(\mathsf{sk}_y, \mathsf{Enc}(x, n)) = \begin{cases} 1 & \text{if } x = y \mod n \\ 0 & \text{otherwise} \end{cases}$$

High-level Construction

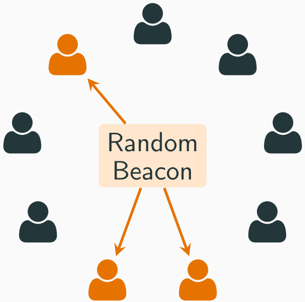


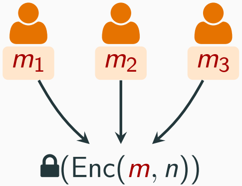
Election: Committee Selection

Each election begin by **publicly** selecting a **random** committee of κ users, which will produce a commitment to the challenge.

Election: Committee Selection

Each election begin by **publicly** selecting a **random** committee of κ users, which will produce a commitment to the challenge.





Election: Committed Challenge Ciphertext

The encryption of (m, n) in our Modular KS has the following form

$$\mathsf{Enc}(m, n) \;=\; \left(s \cdot \boxed{[\underline{a}]_1}\,,\; [\sigma \cdot \underline{x}]_1 + s \cdot \boxed{\left[\underline{a}^\top \mathsf{W}\right]_1} \right)$$

Public-Key
Elements

Where $\underline{x} = (m, -1, -n)$ and $\sigma, s \in \mathbb{F}_q$ are random.

Election: Committed Challenge Ciphertext

The encryption of (m, n) in our Modular KS has the following form

$$\mathsf{Enc}(m,n) \;=\; \Big(s \cdot \boxed{[\underline{\mathsf{a}}]_1}\,,\; [\sigma \cdot \underline{\mathsf{x}}]_1 + s \cdot \boxed{\left[\underline{\mathsf{a}}^\top \mathsf{W}\right]_1} \Big)$$

Public-Key
Elements

Where $\underline{\mathsf{x}} = (m, -1, -n)$ and $\sigma, s \in \mathbb{F}_q$ are random.

Main Challenge: computing $[\sigma \cdot \underline{\mathsf{x}}]_1$, which is non-linear in the secrets

Election: Committed Challenge Ciphertext

Solution: In the Random Oracle we generate the encryption of a random ElGamal ciphertext (of secret key z).

$$🔒(\mathsf{Enc}(m, n)) = \Big(\underset{\text{Random ElGamal Ciphertext}}{\underline{x} \cdot [\rho]_1}, \; \underset{\text{Public-Key Elements}}{s \cdot [\underline{a}]_1}, \; \underline{x} \cdot [\rho z + \sigma]_1 + s \cdot [\underline{a}^\top W]_1\Big)$$

Now the committed challenge is **linear** in the secrets $\underline{x}$ and s, and can be computed in **one round** with synchronous communication.

Election: Threshold Decryption & Leadership Claim

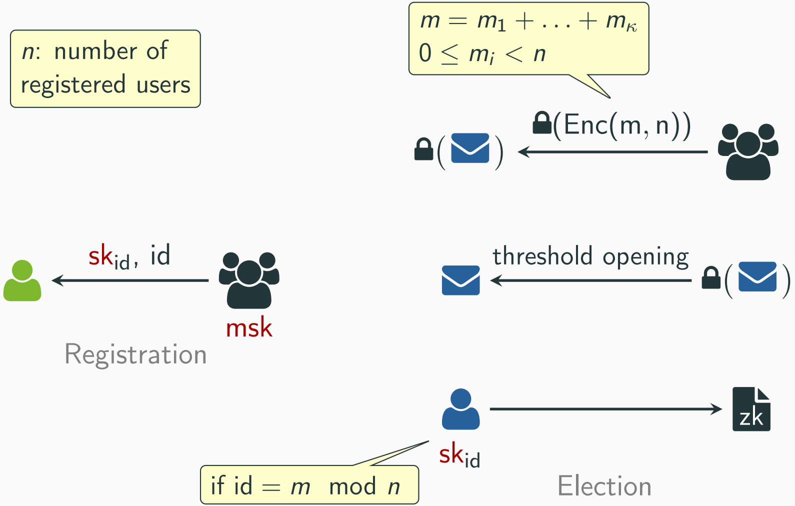
For the **threshold decryption** we assume:

- Less than $t \leq n/2$ corrupted users, i.e. **honest majority**
- Each party has z_i, a t-share of z.
- $(G_1, z \cdot G_1)$ and $z_i G_1$ is **public** for all user P_i

Parties then broadcast their decryption share and a zero knowledge proof to compute the challenge c.

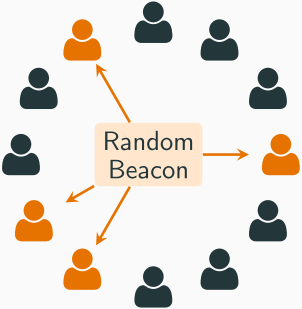
The leader eventually **claim victory** by **proving knowledge** of $\mathsf{sk}_{\mathsf{id}}$ which correctly decrypts c.

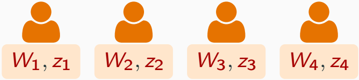
High-level Construction



Registration: Setup

At setup we select a committee which **shares** the secret z and computes new party's **secret keys**





$$W = W_1 + \ldots + W_4$$
$$z = z_1 + \ldots + z_4$$

Registration: Secret Key Generation

The secret keys in our construction are of the form

$$\mathsf{sk}_{\mathsf{id}} \;=\; \Big([r \cdot \underbrace{\mathsf{W}}_{\text{Master Secret Key}}\, \underline{\mathsf{y}}]_1,\ [r]_1\Big)$$

where $\underline{\mathsf{y}} = (1, \mathsf{id}, i)$ for $0 \leq i < \kappa$.

Registration: Secret Key Generation

The secret keys in our construction are of the form

$$\mathsf{sk}_{\mathsf{id}} \;=\; \Big([r \cdot \overset{\text{Master Secret Key}}{\mathsf{W}}\, \underline{\mathsf{y}}]_1,\; [r]_1 \Big)$$

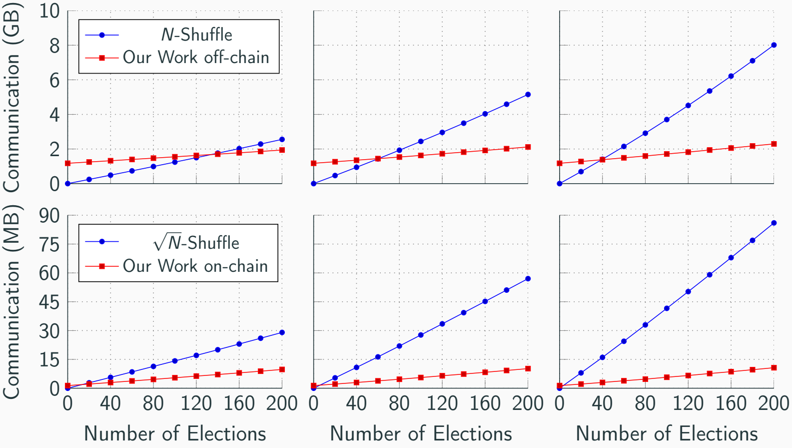
where $\underline{\mathsf{y}} = (1, \mathsf{id}, i)$ for $0 \leq i < \kappa$.

We remove the quadratic term on r and W by sampling $[r]_1$ with the **Random Oracle**:

$$\mathsf{sk}_{\mathsf{id}} \;=\; \Big(\overset{\text{Master Secret Key}}{\mathsf{W}}\, \underline{\mathsf{y}} \cdot \overset{\text{Random Oracle's Output}}{[r]_1},\; [r]_1 \Big)$$

Conclusions

Comparisons



Conclusion

We proposed a practical UC-secure **SSLE** achieving $O(\kappa \log n)$ on-chain communication from **standard pairing** assumptions.

Open problems:

- Reducing the setup cost
- Achieving Adaptive Security

Thanks for your attention!