Peking University

北京大学
PEKING UNIVERSITY

Hades: Practical Decentralized Identity with Full Accountability and Fine-grained Sybil-resistance

Ke Wang, Jianbo Gao, Qiao Wang, Jiashuo Zhang, Yue Li,

Zhi Guan, Zhong Chen

wangk@pku.edu.cn

BG: What is the problem?

The ***permissionless*** nature of blockchain makes it difficult to link blockchain addresses to real-world identities.

Leads to:

- It's challenging for Dapps to implement access control based on ***identity attributes***. (e.g., age)
- Dapps face potential ***legal compliance risks***. (e.g., KYC compliance)
- Once a Dapp is attacked, it is difficult to ***trace the attacker***.
- Users can acquire disproportionate benefits by generating a multitude of addresses (***Sybil attack***)

Solutions

Naive solution: attach the user's wallet an on-chain credential issued by a Certificate Authority (CA).

the *openness* of blockchain leads to users being exposed to a significant risk of privacy leakage.

the most promising solutions:

- **decentralized identities (DIDs)** and **anonymous credentials**
- Basic idea: to allow the user to ***unlinkably*** show that they possess a credential authenticating her/his identity ***without disclosing the original credential***.
- Related works: Zebra, CanDiD, Coconut, BASS, etc.

Limitations & Challenges

#1 Insufficiency of Supporting Accountability.

Accountability is critical to

- *identify individuals* responsible for malicious behaviors (**auditability**)
- *retrieve all activities* of a suspect for investigations (e.g. anti-money laundering) (**traceability**)
- *revoke credentials* that are lost, stolen, or associated with malicious behaviors. (**Revocation**)

Unfortunately, none of the existing works can fully support all those accountability features.

The privacy-preserving requirement makes supporting traceability, auditability, and revocation challenging.

Limitations & Challenges

#2 Inability to resist Sybil attacks.

Sybil-resistance is extremely necessary in certain scenarios, such as anonymous voting, fair currency distribution (“airdrops”).

Unfortunately, Few previous works support traceability.

CanDID is the state-of-the-art DID system to support Sybil-resistance, *but*

- at the cost of compromising unlinkability.
- the Sybil-resistance process requires the participation of the committee

Implementing Sybil resistance while ensuring unlinkability is challenging because the application cannot determine whether the access comes from the same user.

Limitations & Challenges

#3 Inefficiencies of running on the blockchain.

Managing identity through smart contracts is desirable: the smart contracts of Dapps could directly call the identity management system

However, to ensure privacy, most previous works *rely on complex cryptographic* computations, resulting in enormous on-chain overhead.

Furthermore, due to the lack of an effective credential revocation mechanism, these cryptographic computations often need to be *re-executed multiple times*.

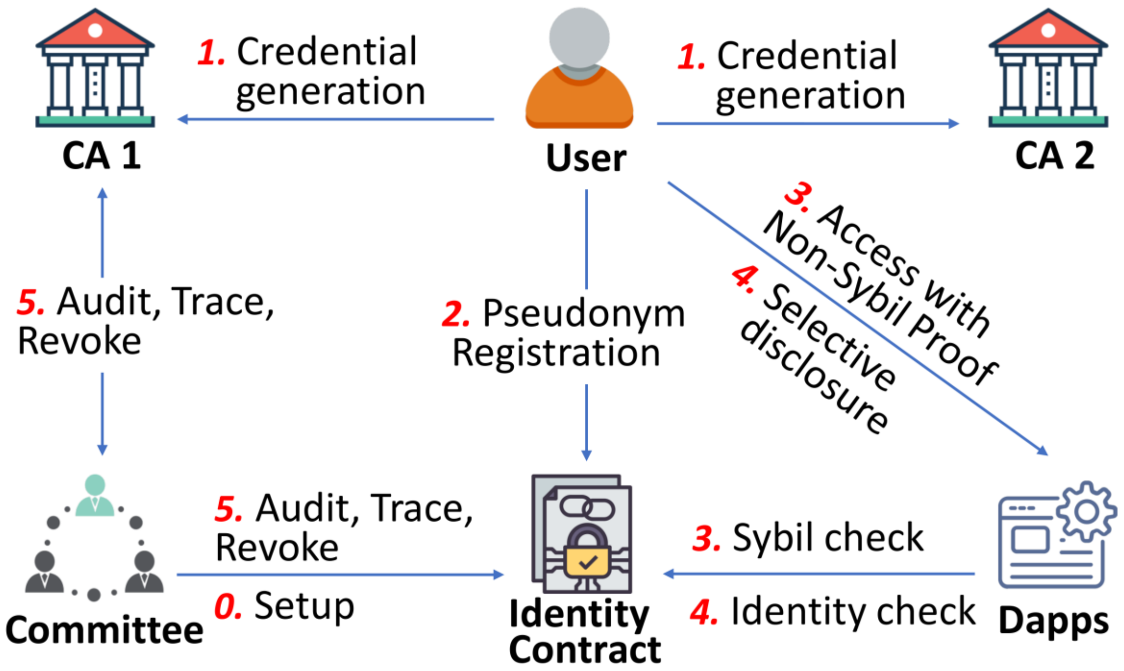
What is Hades?

We presented Hades, a DID system with

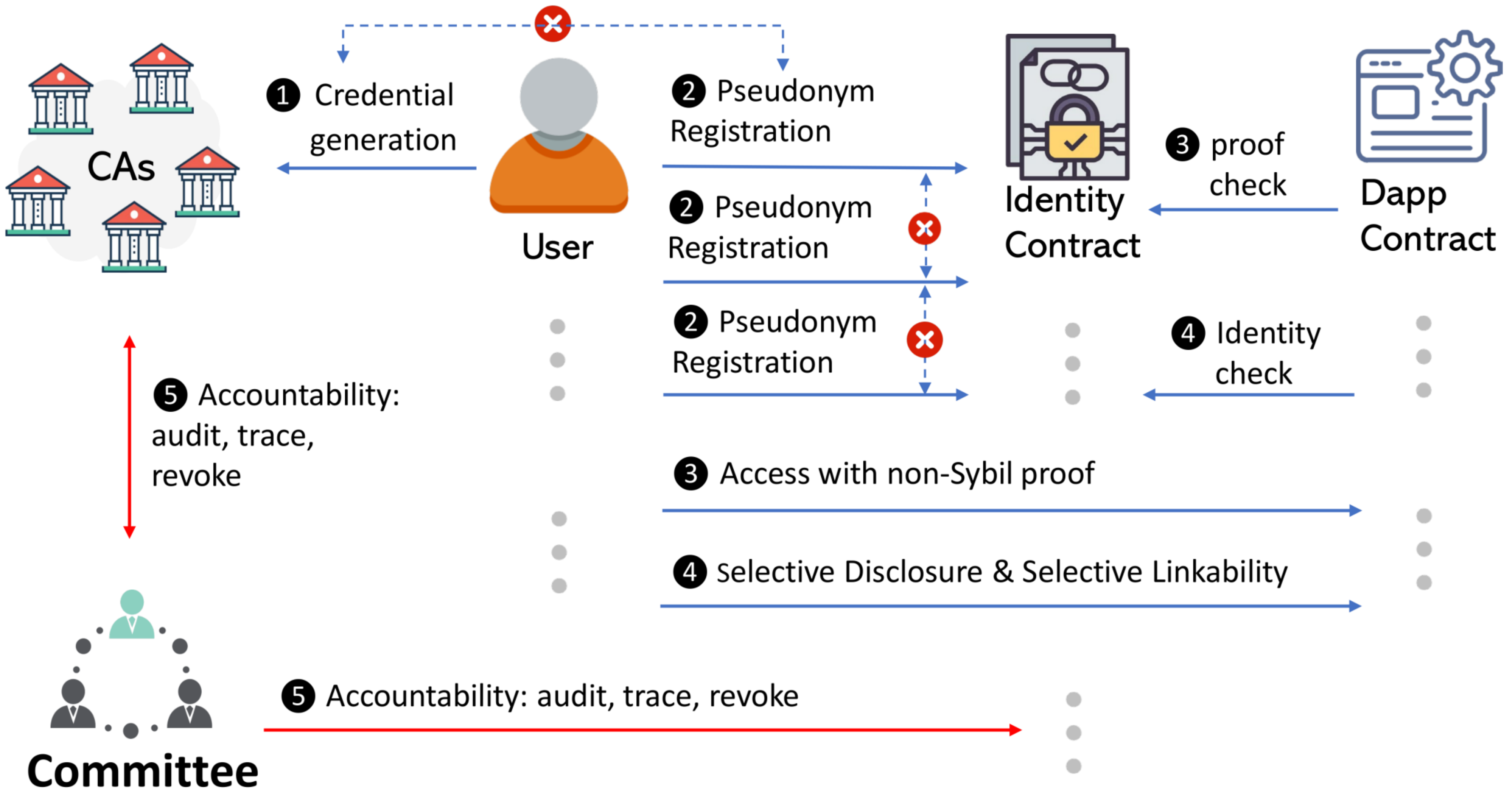
- ***full accountability.*** supporting traceability, auditability, and revocation.
- ***fine-grained Sybil-resistance.*** ① Sybil-resistance can be implemented based on user identity attributes (e.g. assigning different access limits for users of different age groups). ② does not require the assistance of a committee or a Certificate Authority (CA).
- ***Practical.*** ① has the lowest gas cost incurred on EVM as far as we know. ② An address only needs to be verified once during its validity period.
- ***privacy-preserving.*** ① The identity of the user and the issuer of the credentials are both concealed; ② pseudonyms can not be linked.

The Overview of Hades

- ***Committee.*** a union of several distinct entities responsible for system management and identity accountability. *honest-majority*
- ***CA.*** an authorized organization that authenticates and stores users' identity attributes. *semi-honest*
- ***Identity Contract.*** a system contract that verifies, stores, and manages users' pseudonyms.
- ***Dapp.*** a series of smart contracts deployed on the blockchain.
- ***Users.*** access DApps using pseudonyms. *malicious*



The Workflow of Hades



Basic Ideas of Hades

- ***Practical.*** *zk-SNARKs* can be verified efficiently on EVM → building privacy-preserving properties on top of *zk-SNARKs*
- ***Decentralized accountability.*** All information required for accountability is encrypted using *threshold public-key encryption* → Accountability requires the consent of more than a certain number of committee members.
- ***Tracing.*** assign each pseudonym a unique *trapdoor-linkable identifier* → With the knowledge of the secret trapdoor, all relevant pseudonyms can be traced by their identifiers.
- ***Revocation.*** all pseudonyms of a user can be traced → can be revoked.
- ***Sybil-resistance.*** attach each access a unique *unlinkable* context-based access token → a user can generate limited numbers of access tokens for a given context.

Cryptographic Schemes

- ***Zero-knowledge proofs.*** Allow a user to prove in zero-knowledge that the secret values and all other public values satisfy some statements .
- ***Merkle trees:*** The Merkle tree allows a prover to commit to an arbitrary finite set S of values, and for any value x, reveal with a proof whether $x \in S$ or $x \notin S$
- ***Threshold public-key encryption:*** Threshold public-key encryption (TPKE) allows a set of users to decrypt a ciphertext if a predetermined threshold of authorized users cooperates
- ***Generalized Pedersen commitment***. In Hades, a generalized version of Pedersen commitment scheme is used to hide values of identity attributes into a commitment.

Credential Generation

$$\mathsf{PK}^{\mathsf{C}}, \mathsf{G}, (\mathsf{G}_1, \ldots, \mathsf{G}_n), \mathsf{CRS}_1$$

User: $\{a_j\}_{j\in S}$ | **CA**: $\mathsf{sk}_i^{\mathsf{A}}$

$sk^U \leftarrow_R \mathbb{Z}_p$ ← master key

$\beta \leftarrow_R \mathbb{Z}_p: \exists \alpha \in \mathbb{F}_q, (\alpha, \beta) \in \mathbb{G}_p$ ← a trapdoor for tracing

$PK^U \leftarrow x\mathsf{G}, B \leftarrow \beta\mathsf{G}$

$\psi^t \leftarrow \mathsf{Enc}(\mathsf{PK}^{\mathsf{C}}, \mathsf{Encode}(\beta), \mathsf{PK}_i^{\mathsf{A}})$ ← the TPKE encryption

$\Pi^c \leftarrow \mathsf{NIZK}^1\{\ldots\}^a$ ← A zero-knowledge proof

User → CA: $\xrightarrow[\{a_j\}_{j\in S}]{PK^U, B, \psi^t, \Pi^c}$

Verify identity, *abort* if failed

Verify Π^c, *abort* if failed

$A \leftarrow \sum_{j\in S} a_j G_j$

Choose e ← expiration time

$\sigma = \mathsf{Sign}(sk, (X||B||A||e))$

Store: $(PK^U, B, e, \psi^t, \sigma, \{a_j\}_{j\in S})$

CA → User: $\xleftarrow{e,\sigma}$

Verify σ, *abort* if failed

Store $(sk^U, \beta, e, \sigma, \{a_j\}_{j\in S})$

Output — User: $(sk^U, \beta, e, \sigma, \{a_j\}_{j\in S})$ | CA: $(PK^U, B, e, \psi^t, \sigma, \{a_j\}_{j\in S})$

$$\Pi^C \leftarrow \mathsf{NIZK}^1\{(\beta, k) : B = \beta G$$
$$\wedge\ \psi^t = \mathsf{Enc}(\mathsf{PK}^{\mathsf{C}}, \mathsf{Encode}(\beta), \mathsf{PK}_i^{\mathsf{A}})\}.$$

- We introduced a ***trapdoor*** for each credential, which can be used to trace all the pseudonyms associated with that credential.
- the user is required to provide a trace string ψ^t to the issuer, which is TPKE encryption of the trapdoor β

Pseudonym Registration

$\mathsf{PK}^{\mathsf{C}}, \mathsf{G}, \tau_r, \tau_c, \mathsf{CRS}_2$

User: $sk^U, \beta, \Gamma, \mathsf{pk}_i^{\mathsf{A}}, \xi^U$		**Identity Contract:**
Choose $e_s : e_t \le e$ ← expiration time		
$A_t \leftarrow A + r_t G, r_t \leftarrow_R \mathbb{Z}_p$ ← Pedersen commitment of Identity attributes		
Choose $m_0, m_1 : \lvert\mathbb{P}\rvert - w \le m_0 \le m_1 \le \lvert\mathbb{P}\rvert$		
Choose $\mathsf{nonce} : m_0 \le \mathsf{nonce} \le m_1$ ← This ensures that the range of nonce values is not too large		
$k \leftarrow \mathsf{Hash}(\beta \Vert \mathsf{nonce})$		
$\psi^a \leftarrow \mathsf{Enc}(\mathsf{PK}^{\mathsf{C}}, PK^U, (\mathsf{PK}_i^{\mathsf{A}} + \xi^U \mathsf{G}))$ with k ← Used for audit and tracing		
$\Pi^P \leftarrow \mathsf{NIZK}^2\{...\}^a$ ← A zero-knowledge proof to prove that all values are correctly generated	$\xrightarrow[\Pi^P, m_0, m_1, T_\xi]{A_t, e_t, \psi^a}$	Check: $\lvert\mathbb{P}'\rvert - w \le m_0 \le m_1 \le \lvert\mathbb{P}'\rvert$ *abort* if failed
		$\xi^U \leftarrow T_\xi.\mathsf{sender}()$ ← The address to be registered.
		Verify Π^P, *abort* if failed
if $Res \equiv 0$, *abort*	$\xleftarrow{Res}$	Store $(\xi^U, A_t, e_t, \psi^a)$
Store $(\xi^U, e_t, r_t, \mathsf{nonce})$		
$(\xi^U, e_t, r_t, \mathsf{nonce})$		$(\xi^U, A_t, e_t, \psi^a)$

- Instead of disclosing the credential, the user presents a zero-knowledge proof to the identity contract, *proving possession of a valid identity credential.*
- For auditing, the user is required to provide a trace string ψ^a to the contract, which is *TPKE encryption of identity information.*
- To enable tracing, users are required to employ trapdoor β to *deterministically produce the nonce k used in encryption*, making the ciphertext a unique identifier.
- A zero-knowledge proof ensures that all values are correctly generated.

Audit

If a pseudonym has shown malicious behavior, its identity-related information can be revealed by a threshold number of committee members.

$\mathsf{G}, \mathsf{PK}^{\mathsf{C}}, \tau_r, \mathsf{IC}^{\mathrm{a}}$

Committee: $\{\mathsf{sk}^{\mathsf{C}}_1, \ldots, \mathsf{sk}^{\mathsf{C}}_n\}, t$ **CAs**: $\mathsf{DB}_{\mathsf{U}}{}^{\mathrm{a}}$

Audit: ξ^U

$\psi^a \leftarrow \mathtt{IC.info}(\xi^U)$ ← published in the identity contract when registering.

$d^a_j \leftarrow \mathtt{Dec}(\psi^a, \mathsf{sk}^{\mathsf{C}}_j)$ ← Threshold decryption

$(PK^U, M_2) \leftarrow \mathtt{Comb}(\{d^a_j\}_{j \in S \text{ s.t. } |S| > t})$

$\mathsf{PK}^{\mathsf{A}}_i = M_2 - \xi^U \mathsf{G}$ ← Public key of the issuer

$\xrightarrow[\text{to CA}(\mathsf{PK}^{\mathsf{A}}_i)]{PK^U}$ $\mathtt{info} \leftarrow \mathsf{DB}_{\mathsf{U}}.\mathtt{info}(PK^U)$ ← Query the CA for the identity details associated with this credential.

Reveal $\mathtt{info}$ $\xleftarrow{\mathtt{info}}$

- To register a pseudonym, an audit string ψ^a is submitted to the identity contract, which is TPKE encryption of the owner's public key PK^U and the issuer's public key PK^A
- t +1 of committee members can collaboratively decrypt the audit string to recover the public keys
- By querying the CA identified by PK^A with PK^U, the identity information associated with the pseudonym can be revealed.

Tracing

If a user has shown malicious behavior, all pseudonyms belong to him/her can be revealed by a threshold number of committee members.

$$\mathsf{G}, \mathsf{PK}^{\mathsf{C}}, \tau_r, \mathsf{IC}^{\mathrm{a}}$$

Committee: $\{\mathsf{sk}_1^{\mathsf{C}}, ..., \mathsf{sk}_n^{\mathsf{C}}\}, t$ — **CAs**: $\mathsf{DB_U}^{\mathrm{a}}$

Trace: ID^{U}

Threshold decryption

TPKE encryption of the trapdoor.

$d_j^t \leftarrow \mathsf{Dec}(\psi^t, \mathsf{sk}_j^{\mathsf{C}})$ $\xleftarrow{\psi^t}$ $\psi^t \leftarrow \mathsf{DB_U}.\mathrm{info}(\mathsf{ID}^{\mathsf{U}})$

$(\beta, \mathsf{PK}_j^{\mathsf{A}}) \leftarrow \mathsf{Comb}(\{d_j^t\}_{j\in S \text{ s.t. } |S|>t})$

the range of the nonce

Public ψ^t $\xrightarrow{\beta}$ $n_0 \leftarrow |\mathbb{P}_0| - w, n \leftarrow |\mathbb{P}|$

First element of ψ → $C_0 \leftarrow \{\mathsf{Hash}(\beta||j)G\}_{j\in[n_0,n]}$

Reveal $\{\xi^U\}$ $\xleftarrow[\text{if needed}]{\{\xi^U\}}$ $\{\xi^U\} \leftarrow \mathtt{IC.filter}(C_0)$

compare with the ψ recorded on the identity contract

- A trace string ψ^t was provided to the issuer when the user apply credential
- t +1 of committee members can *collaboratively decrypt the trace string* to recover the trapdoor
- With the trapdoor β, the authority can locally *calculate all the identifiers* that the user can currently use.
- With the identifiers recorded in the identity contract, the authority can identify all pseudonyms belong to the user.

Revocation

The credential, and the pseudonyms associated to the credential can be revocated.

G, PK^C, τ_r, IC^a

Committee: $\{sk_1^C, ..., sk_n^C\}, t$ — **CAs**: $DB_U{}^a$

Revoke: ξ^U, Γ

$\tau_r \leftarrow \mathtt{TreeAdd}(\Gamma.PK^U, \tau_r)$ ← Add the pubkey of the credential to the revoke tree

$\mathtt{IC.udateRoot}(\mathtt{TreeRoot}(\gamma_r))$

$\mathtt{IC.revokePseudonym}(\xi^U) \xrightarrow[\text{revoked}]{\Gamma} DB_U.\mathtt{revoke}(\Gamma)$ ← Mark the related pseudonyms as invalid on the blockchain.

The revocation does not affect the validity of other users' pseudonyms

- To revoke a credential, the committee first *adds the credential's public key* (i.e., PK^U) *into the revocation tree* and updates the new tree root to the identity contract → proof of pseudonym registration using this credential will fail verification
- *Trace all pseudonyms* registered using this credential → marks these pseudonyms *"revoked"* in the identity contract.

Sybil-resistance

$$\mathsf{G}, \mathsf{PK}^{\mathsf{C}}, \mathsf{CRS}_3, \mathtt{limit}(\ldots), \zeta, \mathtt{IC}^{\mathrm{b}}$$

User: $sk^U, k, r, \{a_i\}$		**Application**:
$n \leftarrow \mathtt{limit}(\{a_i\})$		
$\mathtt{nonce} \leftarrow_R [0, n)$		
$\varphi = \mathtt{Hash}(sk^U \| \zeta \| \mathtt{nonce})$		
$\Pi^S \leftarrow \mathrm{NIZK}^3\{\ldots\}^{\mathrm{a}}$	$\xrightarrow{\xi^U, \varphi, \Pi^S}$	$(\psi^a, A_t) \leftarrow \mathtt{IC.info}(\xi^U)$
		Verify Π^S, *abort* if failed
if $\mathtt{Res} \equiv 0$, abort	$\xleftarrow{\mathtt{Res}}$	Store (ξ^U, φ)
Store $(\xi^U, \mathtt{nonce}, \varphi)$		
$(\xi^U, \mathtt{nonce}, \varphi)$		(ξ^U, φ)

$$\begin{aligned}\Pi^S \leftarrow \mathrm{NIZK}^3\{(sk^U, \mathtt{nonce}, k, \{a_i\}, r) : PK^U &= sk^U \mathsf{G} \\ \wedge\ \psi^a &\leftarrow \mathtt{Enc}(\mathsf{PK}^{\mathsf{C}}, PK^U, (\mathsf{PK}_i^{\mathsf{A}} + \xi^U \mathsf{G})) \\ \wedge\ \varphi &= \mathtt{Hash}(sk^U \| \zeta \| \mathtt{nonce}) \\ \wedge\ 0 \le \mathtt{nonce} &< \mathtt{limit}(PK^U, \{a_i\}) \\ \wedge\ A_t &= \sum a_i \mathsf{G}_i + r\mathsf{G}\ \}\end{aligned}$$

All guaranteed by a zero-knowledge proof

- We design an n-time access token generation scheme, which takes the credential's private key and Sybil-resistance instance ID as input, and outputs at most n distinct tokens.
- n is determined *by the identity attributes* of the user.
- Every new access must be accompanied by an access token → A Sybil attack can be identified by checking *if the access token is duplicated*

Sybil-resistance over identity identifiers.

- In the above, the Sybil-resistance process is conducted over credentials, considering ***each credential as a unique entity***.
- This may fail when users can apply credentials from ***multiple CAs***.
- A plausible approach is to employ Sybil-resistance over identity identifiers, such as Social Security Numbers (SSN)
- We can achieve this using a ***threshold pseudorandom function (PRF).***

Sybil-resistance over identity identifiers.

- The user first apply a credential that verifies his SSN $\boldsymbol{ID^U}$
- The user sends the *secret shares* [$\boldsymbol{ID^U}$] of $\boldsymbol{ID^U}$ to the committee, accompanied by a zero-knowledge proof, indicating that she/he has a *credential* that authenticated $\boldsymbol{ID^U}$.
- The committee verifies the received proofs and executes an MPC protocol to *compute the pseudorandom*:

$$\mathtt{ID}^{\mathtt{U}}_{\mathtt{prf}} = \mathtt{PRF}([\mathtt{sk}^{\mathtt{C}}_{\mathtt{prf}}], [\mathtt{ID}^{\mathtt{U}}])$$

- the user can generate access token as

$$\varphi = \mathtt{ID}^{\mathtt{U}}_{\mathtt{prf}}\mathtt{Hash}(\mathtt{ID}^{\mathtt{U}}||\zeta||\mathtt{nonce}) \text{ s.t. } 0 \leq \mathtt{nonce} < \mathtt{limit}(\{a_i\}).$$

- ***privacy-preserving:*** ① only the owner of the $\boldsymbol{ID^U}$ can obtain the pseudorandom; ② the committee members might learn about the pseudorandom, but they remain unaware of $\boldsymbol{ID^U}$ → *cannot link user by access tokens.*

Selective Disclosure & Selective Linkability

Selective Disclosure.

- When a user registers a pseudonym, she/he needs to submit a *Pedersen commitment*, At, of her/his identity attribute values. This is recorded in the identity contract.
- The user can prove to the application that her/his identity attributes meet certain assertions, such as being over 18 years old, *with a zero-knowledge proof*:

$$\Pi^D \leftarrow \mathtt{NIZK}^4\{\{a_1, ..., a_n\}, r) : A_t = \sum_{i=1} a_i G_i + rG \wedge \text{statements about } a_i \text{ for } 1 \leq i \leq n\}$$

Selective Disclosure & Selective Linkability

Selective Linkability.

- Users can use the access token generation scheme to *prove the linkability of their pseudonyms* without revealing identity-related information.
- If two pseudonyms are registered using the same credential, then given the same instance ID, the owner will certainly be able *to generate an identical access token* for them.
- To prove the linkability of pseudonyms, the owner can generate an identical access token for the pseudonyms using the same context.
- Since the access token generation does not reveal any identity-related information, this selective linkability scheme is *privacy-preserving*.
- very useful in *pseudonym replacement* and *pseudonym revocation*

Implementation & Benchmark

- We implemented Hades using Rust and Solidity and published the code on GitHub as an open-source project: https://github.com/didnet/Hades
- We evaluated our implementation on machine equipped with an Intel Core i9-13900K@3.0GHz 16-Core (8P+16E) CPU and 64 GB of RAM. The identity contract was deployed on BSC Testnet.

#1 The zero-knowledge proof benchmark

Operation	#Constraints	Time[ms]
credential generation	3, 907	195
pseudonym registration	31, 951	614
Sybil-resistance	4, 291	245
selective disclosure	15, 856	564

#2 The gas cost benchmark

Operation	gas cost	input data size
pseudonym registration	339, 978	320 bytes
Sybil-resistance	248, 514	224 bytes
pseudonym revocation	∼ 2, 500/pr[a]	32 bytes/pr[a]
selective disclosure	232, 857	192 bytes

[a]The symbol '/pr' means 'per pseudonym revocation'.

Comparison

- We compared Hades with other identity management protocols in terms of gas cost, selective linkability, selective disclosure, audit, trace, revocation, and Sybil-resistance.
- Hades has the *lowest gas cost*.
- Hades is the first DID system that implemented *all of the features* listed.
- Hades is also the first DID system supporting *lightweight, fine-grained Sybil-resistance*

Table 3: Comparison with prior works in gas cost

Technique	Gas cost	One-time cost
Hades	339 K	Yes
ZEBRA [35]	360 K	Yes
Coconut [40]	2,150 K	No
BASS [46]	1,585 K	No

Property	Selective-disclosure	Selective-linkability	Audit-ability	Trace-ability	Revo-cation	Sybil-resistance
Hades	●	●	●	●	●	●
ZEBRA	○	◐	●	○	●	○
Coconut	●	◐	○	○	○	○
BASS	○	◐	●	○	●	○
CanDID	●	◐	○	●	●	◐

† ●: supports, ◐: partially supports ○: does not support.
‡ Partial support for selective linkability indicates only supporting unlinkability; partial support for Sybil-resistance refers to supporting Sybil-resistance with a single and fixed strategy.

Conclusion

- We presented Hades, a practical decentralized identity system that supports *full accountability* and *fine-grained Sybil-resistance*.
- Hades is the first DID system encouraging fine-grained Sybil-resistance through a *lightweight solution*.
- We implemented Hades, and our evaluation shows that Hades *has the lowest gas cost* incurred on EVM and is suitable for mobile devices and web plugins.

Thank You !