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.
#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

1. Credential generation
2. Pseudonym Registration
3. Proof check
4. Identity check
5. Accountability: audit, trace, revoke

Committee
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

$\text{User: } \{a_j\}_{j \in S}$  
$\text{CA: } sk^A_i$

$sk^U \leftarrow_R \mathbb{Z}_p$  
$\beta \leftarrow_R \mathbb{Z}_p: \exists \alpha \in \mathbb{F}_q, (\alpha, \beta) \in \mathbb{G}_p$  
$PK^U \leftarrow xG, B \leftarrow \beta G$

$\psi^t \leftarrow \text{Enc}(PK^C, \text{Encode}(\beta), PK^A_i)$

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

- We introduced a \textit{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 $\psi^t$ to the issuer, which is TPKE encryption of the trapdoor $\beta$.
Pseudonym Registration

<table>
<thead>
<tr>
<th>User: $s k^U, \beta, \Gamma, p k^A_i, \zeta^U$</th>
<th>Identity Contract:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose $e_s : e_t \leq e$</td>
<td><strong>expiration time</strong></td>
</tr>
<tr>
<td>$A_t \leftarrow A + r_t G, r_t \leftarrow R \mathbb{Z}_p$</td>
<td>Pedersen commitment of</td>
</tr>
<tr>
<td></td>
<td>Identity attributes</td>
</tr>
<tr>
<td>Choose $m_0, m_1 :</td>
<td>\mathcal{P}</td>
</tr>
<tr>
<td>Choose nonce : $m_0 \leq \text{nonce} \leq m_1$</td>
<td>nonce values is not too large</td>
</tr>
<tr>
<td>$k \leftarrow \text{Hash}(\beta</td>
<td></td>
</tr>
<tr>
<td>$\psi^a \leftarrow \text{Enc}(\text{PK}^C, \text{PK}^U, (\text{PK}^A_i + \zeta^U G))$</td>
<td>A zero-knowledge proof to</td>
</tr>
<tr>
<td></td>
<td>prove that all values are</td>
</tr>
<tr>
<td>$\Pi^P \leftarrow \text{NIZK}^2[...]^a$</td>
<td>correctly generated</td>
</tr>
<tr>
<td>$\frac{A_t, e_t, \psi^a}{\Pi^P, m_0, m_1, T_\xi}$</td>
<td>The address to</td>
</tr>
<tr>
<td>Check: $</td>
<td>\mathcal{P}'</td>
</tr>
<tr>
<td>abort if failed</td>
<td></td>
</tr>
<tr>
<td>$\xi^U \leftarrow T_\xi.\text{sender}()$</td>
<td></td>
</tr>
<tr>
<td>Verify $\Pi^P$, abort if failed</td>
<td></td>
</tr>
<tr>
<td>Store $(\xi^U, A_t, e_t, \psi^a)$</td>
<td></td>
</tr>
<tr>
<td>Store $(\xi^U, e_t, r_t, \text{nonce})$</td>
<td></td>
</tr>
<tr>
<td>$(\xi^U, e_t, r_t, \text{nonce})$</td>
<td></td>
</tr>
</tbody>
</table>

- 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 $\psi^a$ to the contract, which is **TPKE encryption of identity information.**
- To enable tracing, users are required to employ trapdoor $\beta$ 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.

\[
\text{Audit: } \xi^U
\]

\[
\psi^a \leftarrow \text{IC.info}(\xi^U)
\]

\[
d_j^a \leftarrow \text{Dec}(\psi^a, sk_j^C)
\]

\[
(PK^U, M_2) \leftarrow \text{Comb}({d_j^a})_{j \in S \text{ s.t. } |S| > t}
\]

\[
PK_i^A = M_2 - \xi^U G
\]

\[
\text{Public key of the issuer}
\]

\[
\text{Reveal info}
\]

\[
\text{Published in the identity contract when registering.}
\]

\[
\text{Threshold decryption}
\]

\[
\text{By querying the CA identified by } PK^A \text{ with } PK^U, \text{ the identity information associated with the pseudonym can be revealed.}
\]

- To register a pseudonym, an audit string \(\psi^a\) 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.
If a user has shown malicious behavior, all pseudonyms belong to him/her can be revealed by a threshold number of committee members.

\[
\begin{array}{c|c}
\text{Committee: } \{sk_1^C, ..., sk_n^C\}, t & \text{CAs: } DB_0^a \\
\hline
\text{Trace: } ID^U & \text{Threshold decryption} \\
\end{array}
\]

- A trace string \( \psi^t \) was provided to the issuer when the user apply credential.
- \( t + 1 \) of committee members can \textit{collaboratively decrypt the trace string} to recover the trapdoor.
- With the trapdoor \( \beta \), the authority can locally calculate \textit{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.

\[
\begin{align*}
\begin{array}{c}
d_j^t \leftarrow \text{Dec}(\psi^t, sk_j^C) \\
(\beta, PK_j^H) \leftarrow \text{Comb} \{(d_j^t)_{j \in S \text{ s.t. } |S|>t}\} \\
\text{Public } \psi^t & \xrightarrow{\beta} n_0 \leftarrow |P_0| - w, n \leftarrow |P| \\
\text{Reveal } \{\xi^U\} & \leftarrow \text{IC.filter}(C_0)
\end{array}
\end{align*}
\]
Revocation

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

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.

The revocation does not affect the validity of other users’ pseudonyms.
Sybil-resistance

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 $\rightarrow$ A Sybil attack can be identified by checking if the access token is duplicated.

All guaranteed by a zero-knowledge proof
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 $ID_U$
• The user sends the secret shares $[ID_U]$ of $ID_U$ to the committee, accompanied by a zero-knowledge proof, indicating that she/he has a credential that authenticated $ID_U$.
• The committee verifies the received proofs and executes an MPC protocol to compute the pseudorandom:

$$ID_{prf}^U = PRF([sk_{prf}^C],[ID_U])$$

• the user can generate access token as

$$\varphi = ID_{prf}^U Hash(ID_U||\xi||nonce) \text{ s.t. } 0 \leq nonce < \text{limit}(\{a_i\}).$$
• **privacy-preserving:**
  ① only the owner of the $ID_U$ can obtain the pseudorandom;
  ② the committee members might learn about the pseudorandom, but they remain unaware of $ID_U \Rightarrow \text{cannot link user by access tokens.}$
Selective Disclosure & Selective Linkability

Selective Disclosure.

- When a user registers a pseudonym, she/he needs to submit a Pedersen commitment, $A_t$, 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:

$$\Pi^D \leftarrow \text{NIZK}^A\{\{a_1, ..., a_n\}, r\} : A_t = \sum_{i=1}^{\text{n}} a_i G_i + r G$$

\[\land \text{ statements about } a_i \text{ for } 1 \leq i \leq n\]}
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 a 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

<table>
<thead>
<tr>
<th>Operation</th>
<th>#Constraints</th>
<th>Time[ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential generation</td>
<td>3,907</td>
<td>195</td>
</tr>
<tr>
<td>pseudonym registration</td>
<td>31,951</td>
<td>614</td>
</tr>
<tr>
<td>Sybil-resistance</td>
<td>4,291</td>
<td>245</td>
</tr>
<tr>
<td>selective disclosure</td>
<td>15,856</td>
<td>564</td>
</tr>
</tbody>
</table>

#2 The gas cost benchmark

<table>
<thead>
<tr>
<th>Operation</th>
<th>gas cost</th>
<th>input data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>pseudonym registration</td>
<td>339,978</td>
<td>320 bytes</td>
</tr>
<tr>
<td>Sybil-resistance</td>
<td>248,514</td>
<td>224 bytes</td>
</tr>
<tr>
<td>pseudonym revocation</td>
<td>~ 2,500/pr⁵</td>
<td>32 bytes/pr⁵</td>
</tr>
<tr>
<td>selective disclosure</td>
<td>232,857</td>
<td>192 bytes</td>
</tr>
</tbody>
</table>

⁵The 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>
<thead>
<tr>
<th>Technique</th>
<th>Gas cost</th>
<th>One-time cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hades</td>
<td>339 K</td>
<td>Yes</td>
</tr>
<tr>
<td>ZEBRA [35]</td>
<td>360 K</td>
<td>Yes</td>
</tr>
<tr>
<td>Coconut [40]</td>
<td>2,150 K</td>
<td>No</td>
</tr>
<tr>
<td>BASS [46]</td>
<td>1,585 K</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Selective-disclosure</th>
<th>Selective-linkability</th>
<th>Audit-ability</th>
<th>Trace-ability</th>
<th>Revocation</th>
<th>Sybil-resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hades</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>ZEBRA</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Coconut</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>BASS</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>CanDID</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

†●: 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 !