Blockchain Stacking Fraud and Deterrence

Tong Cao, Man Ho Au, Xiapu Luo
The Hong Kong Polytechnic University

***Abstract*—As crypto-assets become increasingly integrated into global financial systems, they are now employed to demonstrate solvency, establish creditworthiness, and satisfy wealth verification requirements for various financial and regulatory purposes. A fundamental challenge in these applications is proving that a given wallet address is controlled exclusively by a single legal entity, as fraudulent multiple claims can undermine the integrity of asset-based verification systems.**

In this paper, we propose a binding of wallet address to legal identity that makes conflicting claims cryptographically inconsistent. In the main design, claimants post to a public, privacy-preserving bulletin board a zkSNARK that proves knowledge of both the private key and the address–identity binding without revealing either; proofs are structured so that duplicate claims for the same address are publicly detectable. When a shared bulletin board is not available, we introduce a self-deterrence proof: a zero-knowledge protocol in which producing two valid proofs for the same address under different legal identities would enable efficient extraction of the private key, creating a strong economic disincentive against double claiming. These cryptographic primitives provide effective countermeasures against blockchain stacking fraud attacks and enhance the reliability of crypto-assets in traditional financial applications, contributing to the broader integration of digital assets into established economic systems.

I. Introduction

Over the past decade the population of crypto-asset holders has grown dramatically. In 2025, roughly more than 820 Million[1] cryptocurrency wallets collectively control crypto-assets exceeding USD 4 Trillion[2] threshold for the first time. These crypto-assets facilitate integration with traditional financial systems by enabling holders to leverage their blockchain wealth for real-world economic activities. Specifically, owners can utilize crypto-assets to demonstrate solvency during regulatory compliance audits, establish creditworthiness through on-chain transaction histories when applying for unsecured loans, and satisfy wealth verification requirements for investment-based immigration and residency programs.

As with conventional collateral in traditional finance, clients who want to obtain services backed by crypto-assets must first demonstrate unequivocal ownership of those assets. To prove the ownership of crypto-assets, the holders must: 1), prove control of the relevant blockchain wallet address, typically by signing a challenge supplied by the financial service provider and broadcasting an authenticated transaction to the blockchain network; and 2), prove the quantity of assets held at the wallet address via providing a snapshot that offers the momentary assurance, while the on-chain address balance monitoring is needed in the service provider's side to check if the assets are in the wallet address continuously.

[1]https://coinlaw.io/cryptocurrency-wallet-adoption-statistics/
[2]https://coinmarketcap.com/

In conventional finance, assets, such as, bank deposits, equities, bonds, are issued or recorded by a trusted central entity (a bank, transfer agent, or central securities depository). Because the registry itself binds each unit of value with a legal person (using legal identity), that issuer or custodian can certify exclusive ownership whenever required. Native crypto-assets, by contrast, are bearer instruments whose state is defined solely by a decentralized ledger. *There is no authoritative registry that maps an on-chain address to a real-world identity*. The wallet address holder can demonstrate momentary control by signing a challenge with the corresponding private key, yet this proof has two important limitations: 1), the signature only shows that the prover can use the corresponding private key to generate the digital signature, where the knowledge of private key might be encumbered [1]; 2), the signature may be copied and shared, and transferred among a group of friends [2]. Thus, a valid signature does not guarantee that the wallet address is solely owned by a legal entity. Such digital asset wallet address ownership proving schemes lead to double claim problem: *the real holder can use the private key to help friends pass through the ownership verification processes*.

Existing blockchain wallet address ownership tests fall into two broad categories: 1), proof of complete knowledge of the private key [1]; and 2), simple challenge–response signatures. However, neither eliminates the double claim problem in which two or more parties assert control over the same address. In the former case, it can require the claimant to produce an interactive zero-knowledge proof of full possession of all bits of the private key. Because the key becomes completely usable by whomever receives it, this requirement discourages commercial key-rental schemes: a rational key owner is unlikely to hand over the entire secret to a stranger. However, it does not rule out cooperative sharing among trusted acquaintances. Two friends can exchange the full key material, back up each other's claims, and later present independent proofs that appear equally valid; no external verifier can tell which party is the real owner. In the latter case, auditors commonly ask the client to sign a nonce with the target wallet address's private key. If the address is public and remains unchanged, the auditor can pair that one-time proof with ongoing balance monitoring to obtain a reasonable collateral guarantee. However, the method fails when the client uses a rotating or undisclosed address set for privacy.

Because a blockchain wallet address's private key is generated independently of any legal identifier, certifying that a given address is controlled by exactly one real-world entity is challenging. Our approach obliges the private key holder to bind the address to a single identity once and then makes any conflicting claim cryptographically self-defeating. Specifically, we publish each claim on a public, privacy-preserving

bulletin board: the claimant posts a zkSNARK proof that proves knowledge of both the private key and an binding of wallet address with a legal ID simultaneously, while revealing neither. The resulting proofs are unique, so duplicate claims for the same address are immediately detectable by any verifier. Where a shared bulletin board is infeasible, we introduce a *self-deterrence* proof: a zero-knowledge proof protocol invented such that two valid proofs for the same address but different legal IDs allow efficient extraction of the underlying private key, thereby creating a strong economic disincentive to double claim. The same self-deterrence primitive can be repurposed to halt blockchain loan stacking, preventing a borrower from pledging identical collateral to multiple lenders in the blockchain ecosystem, which we show as an independent interest and will tackle it in the future work.

II. Blockchain Stacking Fraud Attack

Blockchain Stacking Fraud Attack: using the same wallet addresses to obtain real-world financial services from multiple marketplaces simultaneously. The economic effect would be the potential loss of bust out.

A. Multi-entities Stacking: Double Claim of Ownership

As shown in Figure. 1, we consider two real-world digital asset related activities: Asset Holder leverages her digital assets in a wallet address to apply for financial services based on a third-party verification report, and Asset Holder's friend utilizes the digital assets held in the same wallet address as verification resources for the regulatory-mandated proof of solvency requirements. The asset holder uses the corresponding secret key of the wallet address *addr. x* to generate $claim, \sigma$ (σ can be a digital signature scheme) to convince "Checker A" that she owns *addr. x*, and generate $claim', \sigma'$ to convince "Checker B" that her friend owns *addr.x* at the same time. A double claim attack means the asset holder gets the financial service offering from the FSP and her friend successfully completed the verification process by claiming the ownership of same wallet address. A double claim would succeed when "Checker A, B" confirm the validity of $claim, \sigma$ and $claim', \sigma'$ and "Checker A, B" are isolated.

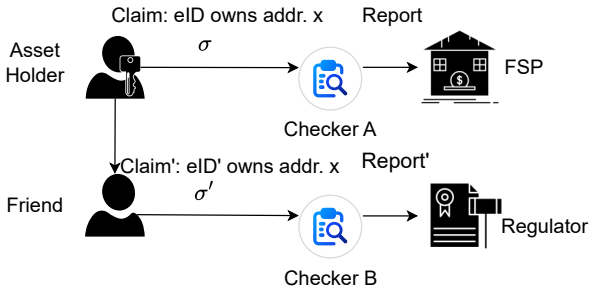

Fig. 1. Overview of double claim. "Checker" represents the trusted third party that can verify real-world entity identity eID, eID' and the digital signature σ, σ' of $claim$ and $claim'$. "FSP" denotes the financial service provider that incorporates digital asset into the service offerings. "Regulator" denotes the regional organization that is responsible for supervising all regional financial institutions and checking if they comply with regulations. The asset holder uses the secret key of wallet address "addr. x" to sign $claim$ and $claim'$ (the claim of a friend) and generate two signatures σ, σ'. The asset holder shares σ' with the friend. In this case, both asset holder and friend can benefit from the claim of owning "addr. x" (the asset holder gets financial service offering from the FSP, while the friend completes the proof of solvency audit process required by the regulartor).

B. Single-entity Stacking: Service Stacking

Double claiming of ownership is not the sole enabler of a stacking fraud attack. Even when an asset has a unique, uncontested owner, a malicious holder can still secure multiple loans or other credit facilities from independent financial-service providers (FSPs), thereby "stacking" credit lines against the same digital asset. Consider the scenario depicted in Figure 2. The asset holder proves control of a wallet by signing a message with the private key corresponding to the advertised address, and submits concurrent loan requests to FSP 1 and FSP 2. Checkers (Checker A for FSP 1 and Checker B for FSP 2) validate the signature, each generating an report that is forwarded to its respective FSP. Because the two checkers, and consequently the two FSPs, operate without synchronization, neither institution learns of the other's pending exposure. The borrower therefore succeeds in obtaining multiple credit lines secured by the same asset, setting the stage for a classic bust-out.

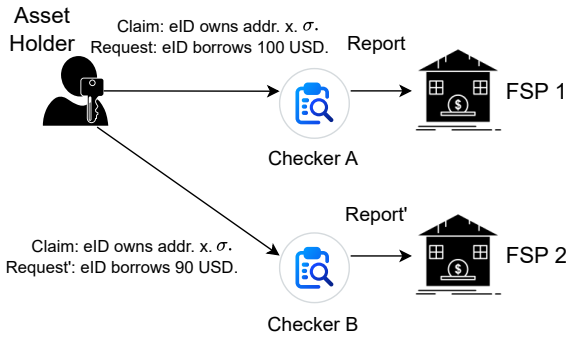

Fig. 2. Overview of service stacking. "Checker" represents the trusted third party that can verify real-world entity identity eID and the digital signature σ of $claim$. "FSP 1 and 2" denote two different financial service providers that incorporate digital asset into the service offerings. The asset holder uses the secret key of wallet address "addr. x" to sign $claim$ and generate signatures σ, and requests to borrow money from "FSP 1 and 2".

C. Case Study 1: Proof of Assets of Top 10 Exchanges

Our review of proof of solvency practices at the ten largest centralized cryptocurrency exchanges, as shown in Table II-C. In general, each proof of solvency is bifurcated into a proof of liabilities (aggregate customer balances) and a proof of assets (on-chain funds controlled by the exchange). The asset attestations themselves rely on proof of wallet address ownership coupled with periodic balance monitoring, but are typically published on a monthly or quarterly cadence, creating substantial windows in which the public view may diverge from reality. Notably, Kraken is the lone exchange that anchors

TABLE I
PROOF OF SOLVENCY SCHEMES OF TOP 10 EXCHANGES.

Exchanges	Proof of Solvency (PoS or PoR)[1]	Audit Report	Proof of Liabilities	Proof of Address Ownership	(Cabal Attack, Double Claim)[9]
Binance [3]	PoR	Mazars[2]in Nov., 2022	Maxwell-Todd [4]	Asset transfer	(❓, ✅)
Bybit [5]	PoR	Hacken[3]in Feb., 2025	Maxwell-Todd [4]	Asset Transfer	(❓, ✅)
Coinbase [6]	PoR	SEC Filings[4]in Mar., 2025	Audited Financials	Unknown	(❓, ❓)
OKX [7]	PoS	Hacken in Jan., 2025	Modified Merkle Tree	Asset transfer	(❓, ✅)
HTX [8]	PoR	Unknown	Maxwell-Todd [4]	Signature	(❓, ✅)
Bitget [9]	PoR	Self Audit[5]in Jan., 2025	Maxwell-Todd [4]	Signature	(❓, ✅)
Kraken [10]	PoR	The Network Firm LLP[6]in Sep., 2024	Maxwell+ [11]	Asset Transfer	(❌, ❌)
KuCoin [12]	PoR	Self Audit[7]in Feb., 2025	Maxwell-Todd [4]	Signature	(✅, ✅)
Gate.io [13]	PoR	Hacken in Jan., 2025,	Maxwell+ [11]	Asset transter	(❓, ✅)
Bitfinex [14]	PoR	AICPA[8]in Jan., 2024	Maxwell++ [15] and Provisions [2]	Asset Transfer	(❌, ✅)

[1] It was defined as Proof of Solvency in [2], [16]. However, some exchanges use Proof of Reserves instead of Proof of Solvency.
[2] Mazars ceased the audit service in Dec. 2022, and removed all proof-of-reserves report.
[3] https://www.bybit.com/app/user/audit-report.
[4] https://investor.coinbase.com/financials/sec-filings/default.aspx.
[5] https://www.bitget.com/blog/articles/bitget-2025-transparency-report.
[6] https://widget.pro.ledgerlens.io/kraken-platform-spot-futures.
[7] https://www.kucoin.com/proof-of-reserves. Note: the audit service was provided by Mazars before Dec., 2022.
[8] https://blog.bitfinex.com/media-releases/bitfinex-achieves-soc-2-type-2-security-compliance-setting-industry-gold-standard/.
[9] If the exchange is vulnerable to Cabal Attack or Homie Attack.

its asset proof to a fixed, publicly disclosed set of long-term wallet addresses. The remaining exchanges use wallet collections that do not inherently prevent double claim. These findings underscore the industry's continued reliance on partial disclosures and the absence of a rigorous standard that jointly certifies single entity address ownership.

D. Case Study 2: Crypto Unsecure Lending

Crypto markets have spawned a class of unsecured, credit-based lending protocols that extend credit without on-chain collateral, relying instead on off-chain identity or reputation scores of transaction history. These systems are acutely exposed to loan stacking attack (for the purpose of bust out), in which a malicious borrower simultaneously draws credit lines from multiple platforms and absconds before any single venue detects the aggregate exposure. Although individual protocols attempt to deter non-repayment through reactive measures penalty interest rates, linkage to real-world identities, or post-default debt auctions, such mechanisms activate only after default and therefore offer no ex-ante protection against loan stacking. The attacker can extract liquidity at virtually zero risk until the fraud is discovered. The prevalence of this attack surface underscores a critical structural vulnerability in the current crypto lending landscape and motivates the need for other preventive coordination mechanisms.

III. SYSTEM MODEL

A. System Overview

We model a system comprising three actors: the asset holder, one or more financial service providers (FSPs), and independent checkers, together with a public blockchain that serves as an immutable bulletin board. The asset holder controls a wallet derived by a private key. The FSP wishes to lend (or provide other service) against the assets in that wallet, and delegates to a checker the task of verifying ownership. While the blockchain irrevocably records addresses and transactions, it does not bind a real-world entity to a specific address, so nothing prevents the same party from "claiming" multiple identities or, conversely, several parties from asserting ownership of the same address. A naive remedy is to log each address-to-entity binding on-chain, but many users reject such disclosure for privacy reasons: even with zero-knowledge techniques, an on-chain footprint may compromise the clean transactional history or pseudonymity they value.

In this paper, we first propose a zkSNARK based address ownership protocol that achieves single-entity binding when every asset holder is willing to record a (or hidden) claim on-chain. To relax this strong adoption assumption, we introduce a self-deterrence protocol that penalizes double claim behavior immediately even when no global registry exists. Concretely, the asset holder proves in zero knowledge that a circuit, whose private input is the wallet's secret key, correctly computes the public address and a 128 entry list of selectively revealed secret-share pairs. The selection vector, called the indicator, is a 128-bit value computed as PRF(eID, addr), where eID is the asset holder's real-world identifier. Should two distinct entities claim the same address, their differing indicators cause them to reveal complementary subsets of shares. With probability of $1 - \frac{1}{2^{128}}$, these two disclosures suffice to reconstruct the secret key, enabling an extractor to sweep the funds instantaneously. The ever present threat of irrevocable asset loss provides a strong ex-ante deterrent, yet requires no universal blockchain participation, making the scheme deployable in practical financial service (e.g., lending) settings.

Address Ownership. In most blockchains, a wallet address is a deterministic function of its secret key, for example, Ethereum prepends the prefix 0x to the final 20 bytes of the Keccak-256 hash of the secp256k1 public key. While the chain immutably records every balance and transfer associated with that address, it remains silent on the real-world entity that controls the underlying key. Conventional ownership tests therefore rely on a digital signature: the putative owner signs a challenge and a verifier checks the signature against the

address. Such a test proves only momentary control, not durable ownership, and does nothing to stop multiple entities from asserting claims over the same address. We remedy this shortcoming in two stages. First, we introduce a zkSNARK construction that allows an address holder to post an ownership statement to a public bulletin board in zero knowledge, binding a legal identity to the address without leaking either the secret key or the identity itself. Second, recognizing that some users reject any on-chain footprint, we design a self-deterrence protocol that establishes ownership off-chain while preserving privacy.

Double Claim Prevention with a Bulletin Board. When timely access to a shared ledger is assumed, the simplest way to preclude double claiming is to record each ownership statement on that ledger: any subsequent, conflicting statement is immediately visible and can be rejected. Fig. 3 formalizes this approach with two zkSNARKs. The first proves knowledge of the secret key corresponding to the claimed address; the second proves possession of a verifiable legal identity. Separation of the two proofs guarantees that the checker learns nothing about the secret key when the legal identity needs to be provided. So long as every participant can read and write to the bulletin board without delay, double claim becomes provably impossible.

Network Partition and Forfeited Consistency. Real world deployments seldom enjoy such perfect synchrony: geographic latency, regulatory firewalls, or simple outages induce network partitions during which some parties cannot interact with the bulletin board. Under the CAP trilemma, we choose to preserve availability (financial services must remain live even if that means relinquishing strict consistency). Unchecked, however, this choice leaves the door to double claim. To counterbalance the risk, we layer the aforementioned self-deterrence protocol on top of the bulletin board. The protocol does not attempt to maintain a single global view. Instead, it threatens any asset holder who submits inconsistent statements with an irrevocable penalty.

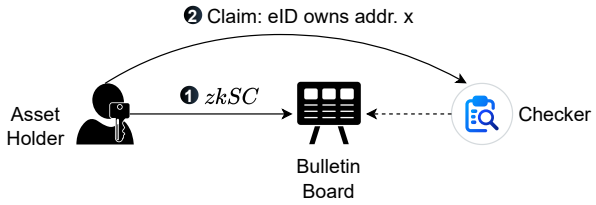

Fig. 3. Overview of preventing double claim using a bulletin board. A bulletin board can be instantiated as a trusted append-only log, BFT state machine, or a blockchain. The asset holder: ❶ uses zero knowledge Sound Claim protocol $zkSC$ to generate a zero knowledge proof for the statement "eID owns addr. x and knows the corresponding secret key" and publish the proof (inlcuding the necessary public parameters required for proof verification) and addr. x to the bulletin board; ❷ sends $claim$ to "Checker". By checking the record of addr. x in the bulletin board, "checker" can simply detect and prevent double claim.

Double Claim Deterrence. The self-deterrence scheme forces every asset holder to embed, within the ownership proof, a cryptographic "booby trap". Specifically, the proof commits to a 128-bit indicator computed as PRF(eID, addr),

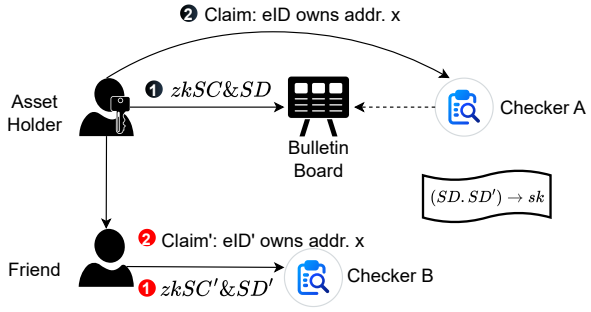

Fig. 4. Overview of deterring double claim in the presence of network partition. In order to discourage double claim, "Checker" requests the prover (asset holder or the friend) to submit a self-deterrence carrying proof SD together with proof of sound claim SC. In this case, when: ❶ the asset holder uses the secret key to generate self-deterrence carrying proof SD' with zero knowledge proof of sound claim $zkSC'$ for the friend, and ❷ the friend makes a claim of owning addr. x to "Checker B", anyone can extract the secret key from SD, SD'. This would pose a risk for the asset holder if the wallet address is claimed and satisfied the requirements of sound claim and self-deterrence carrying more than once, which deters double claim attack.

where eID is the asset holder's real world identity. From this indicator the circuit reveals a complementary subset of 128 secret-share pairs tied to the private key. As shown in Fig. 4, if two distinct identities ever claim the same address, their indicators differ, the revealed subsets mesh, and an extractor can reconstruct the secret key and immediately sweep the funds. Auditors, who receive the proofs asynchronously and share them at unpredictable intervals, act as the extractors of last resort. An attacker therefore faces a risky choice: transfer the funds prematurely and be flagged by the auditors, or wait and risk instant forfeiture once the contradictory proofs meet. Because auditor coordination schedules are opaque to the asset holders, successfully timing a bust-out becomes vanishingly unlikely, rendering double claim economically irrational.

B. Assumptions.

In this paper, we make the following assumptions:

- the PRF used to derive the 128 bit indicator is secure. Formally, no PPT adversary can distinguish its output from uniform with non-negligible advantage;
- the checker (verifier) does not collude with the financial service provider;
- the bulletin board can guarantee safety and liveness;
- the real world identities are unique, and non-forgeable. Each asset holder possesses a globally unique identifier, e.g., a national ID number, driver license number, social security number, or other government issued registry entry number, that can not be duplicated or forged. We assume that the checker is able to verify the correctness of the real world identifiers.

C. Participants

- Asset holder. A real-world entity can be a regional citizen with a legal identity number, or a regional organization with a legal registration number.

- Verifier. A trusted third party that can verify the legal ID of asset holder, and retrieve the data from the bulletin board.
- Bulletin board. A bulletin board can be instantiated as a trusted append-only log, BFT state machine replicas, or a blockchain.
- Financial service provider (FSP). A financial service provider can be any company in a region that provides financial services (e.g., lending, investment migration) to their customers.
- Regulator. A regional regulator is responsible for supervising all regional financial institutions and checking if they comply with regulations. For instance, exchanges are requested to submit proof of reserves to the regulator in United States.

D. Formalization

***Definition** 1:* (Address Generation). *Let sk be the secret key, we say that function $F_{gen}(sk) = addr$ is an address generation, where $addr$ denotes the corresponding wallet address.*

***Definition** 2:* (Claim). *Let eID be the legal identity number (e.g., social security number) of the entity, $addr$ be the corresponding wallet address, we say that function $F_{claim}(eID, addr) = e$ is a claim of "eID owns $addr$" iff. for all probabilistic polynomial-time adversary $\mathcal{A}$: $\Pr[\mathcal{A}(1^\lambda) \rightarrow (eID, eID') : eID \neq eID' \wedge (e, e') : e = e' \leq negl(\lambda)]$, where λ denotes the security parameter.*

***Definition** 3:* (zkSNARK). *Let $\mathcal{P}$ be the prover, $\mathcal{V}$ be the verifier, and $\mathbf{C}$ be a NP arithmetic circuit, a tuple of algorithms (Setup, Prove, Verify):*

- *Setup. On input an arithmetic circuit $\mathbf{C}$, a trusted party uses $S(\mathbf{C};r) = (pp, vp)$ to generate public parameter pp for prover, and vp for verifier.*
- *Prove. $Prove((x, y), w, pp) \rightarrow \pi$.*
- *Verify. $Verify(\pi, (x, y), vp) \stackrel{?}{=} 1$.*

is a zero-knowledge succinct non-interactive argument of knowledge for $\mathbf{C}$ iff.:

- *Completeness. $Prove((x, y), w, pp) \rightarrow \pi$, $((x, y); w) \in \mathbf{C}$, $\forall(pp, vp)$: $\Pr[Verify(\pi, (x, y), vp) = 1] = 1$;*
- *Knowledge Soundness. For $\mathcal{P}^*$, there exists an extractor $\mathcal{E}$ (i.e., probabilistic expected polynomial-time machine) that can extract w given the description of $\mathcal{P}^*$, and $\Pr[((x, y); w) \in \mathbf{C} \wedge verify(\pi, (x, y), vp) = 1] \geq 1 - negl(\lambda)$;*
- *Zero Knowledge. There exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for $((x, y); w) \in \mathbf{C}$, and auxiliary input $z \in \{0, 1\}^*$, the verifier's view during the execution $view_{\mathcal{V}}(\mathcal{P}((x, y), w, pp), \mathcal{V}(x, y), z, vp)$ and $\left\{\mathcal{S}(x, pp, z)\right\}$ are two perfectly distinguishing distributions;*
- *Succinctness. The total communication between $\mathcal{P}$ and $\mathcal{V}$ and verification time are $ploy(\lambda, |x|, log|size(\mathbf{C})|)$.*

***Definition** 4:* ((t,n)-Threshold Secret Share). Let $\mathcal{S}$ be a finite set of secrets (in a finite field $\mathbb{F}_p$ for some prime $p > 0$), a t-out-of-n secret sharing scheme for $\mathcal{S}$ is a pair of probabilistic algorithms:

- Distribute. A probabilistic algorithm shares a secret $s \in \mathbb{F}_p$. $Distribute(s) \rightarrow (s_1, s_2, ..., s_n)$.
- Reconstruct. A probabilistic algorithm recovers the secret s. $Reconstruct(\{s_i\}_{i\in[l]} | l \geq t, s_i \in (s_1, s_2, ..., s_n)) \rightarrow s$.

satisfying the following properties:

- Correctness. For every secret $s \in \mathcal{S}$, $\Pr[Reconstruct(\{s_i\}_{i\in[l]} | l \geq t, s_i \in (s_1, s_2, ..., s_n)) \rightarrow s] = 1$.
- Privacy. Any adversary obtains fewer than t shares of the secret who has zero knowledge about the secret.

IV. Preventing Double Claim Using a Bulletin Board

This section first introduces sound claim *SC* protocol and a Bulletin Board that is instantiated by a smart contract *BB Contract*. We then show the approaches to use *SC* and *BB Contract* to achieve proof of single-entity address-ownership **SEAO** [17] in the network where everyone can use *BB Contract* without any restriction.

A. Intuition

In general, a sound claim of address ownership can be made if asset holder can use cryptographic primitives to bind a wallet address with a real-world entity identity, and prove that she can use the secret key of the same wallet address as the input to a protocol. However, this does not prevent double claim attack as shown in Figure. 1.

In the presence of a bulletin board, the asset holder can publish the proofs and relevant records to the bulletin board to convince others that the ownership of the wallet address has been claimed by a real-world entity with a unique identity. The deduplication function can be easily accomplished in a smart contract to detect and prevent double claim.

Highlight. Our sound claim protocol SC consists of two zkSNARKs, which enables the asset holder to convince the users of bulletin board that a wallet address has been bound with a unique real-world entity identity eID without releasing any information about eID. In particular, SC allows the asset holder to present eID to a third party checker for the purpose of identity verification, which does not affect the security of the secret key.

B. Sound Claim

The goal of a sound claim protocol *SC* is to enable the assets holder to generate zero knowledge proofs to convince others that she has bound a entity identity eID with a wallet address $addr$ and she can use the secret key sk of $addr$. As shown in Figure. 5, We use wallet address generation function f to build the computational statement that $\mathcal{P}$ *knows the secret key sk of function $f(sk) = addr$*, and Poseidon hash function based pseudorandom function as the binding function (i.e., claim that eID owns $addr$ by binding $eID, addr$ via a binding function) to build the computational statement that $\mathcal{P}$ *knows the entity identity eID of function $PRF(eID, addr) = e$*. We use Circom to specify the arithmetic circuits of f and

PRF, and compile them to R1CS. As defined in Section III-D, we use algorithms of $(Setup, Prove, Verify)$ of Groth16 to build public parameters, generate proof, and verify the proof.

Computations:
$PRF : \{0,1\}^* \to \mathbb{Z}_q$ be the pseudorandom function
$f : x \to y$ be the wallet address generation function
1. *zkSC-1*
Arithmetic circuit: $\mathbf{C_1} \xleftarrow{circom} PRF$
Public input: $addr$
Prover $\mathcal{P}$'s private input: eID,
Public output: e
Proof: π_1
$setup(\mathbf{C_1}) \to ppk_1, pvk_1$
$prove(\mathbf{C_1}, ppk_1, (addr, e), eID) \to \pi_1$
$verify(\mathbf{C_1}, pvk_1, (addr, e), \pi_1) \stackrel{?}{=} 1$
2. *zkSC-2*
Arithmetic circuit: $\mathbf{C_2} \xleftarrow{circom} f$
Prover $\mathcal{P}$'s private input: sk,
Public output: $addr$
Proof: π_2
$setup(\mathbf{C_2}) \to ppk_2, pvk_2$
$prove(\mathbf{C_2}, ppk_2, addr, sk) \to \pi_2$
$verify(\mathbf{C_2}, pvk_2, addr, \pi_2) \stackrel{?}{=} 1$

Fig. 5. Zero Knowledge Proof of Sound Claim.

In *SC*, the constructions of *zkSC-1* and *zkSC-2* are separated, releasing eID of *zkSC-1* does not affect zero knowledge property of *zkSC-2*. This allows the asset holder to present the eID to a trusted third party for identity verification without compromising the safety of sk. In real-world, asset holder who wants to apply for the traditional financial service offerings is required to complete identity verification, where the asset holder must present herself and identity certificate to a trusted third party.

C. Bulletin Board

We use a smart contract *BB Contract* to instantiate the bulletin board. The smart contract is settled on a blockchain, which guarantees:

- **Safety.** If a transaction is recorded in the chain and finalized by a correct consensus participant, then the transaction is recorded into the chain and finalized by all correct consensus participants.
- **Liveness.** For any correct transaction, it will be eventually recorded and finalized by all correct consensus participants.

Asset holder can publish proofs with corresponding public input and output to *BB Contract* so that others can verify proofs and check the claim of address ownership. For instance, a record of claim of address ownership in SC is denoted by tuple $(\pi_1, addr, e)$ ($addr, e$ are respectively the public input and output of *zkSC-1*) and tuple $(\pi_2, addr)$ ($addr$ is the public output of *zkSC-2*).

As shown in Figure. 6, *BB Contract* includes the following features:

- **New Record Checker.** When a record of claim of address ownership (which is represented by $(\pi_1, addr, e)$ and $(\pi_2, addr)$) is submitted, *mapping* is used to check if the record is new. For each wallet address, only the first valid record can be stored by *BB Contract*.
- **Consistency Checker.** If a record of claim of address ownership is recognized as new, *BB Contract* checks whether wallet addresses of *zkSC-1* and *zkSC-2* are same. This would ensure that *zkSC-1* and *zkSC-2* are consistently proving the claim of address ownership and knowledge of secret key for the same wallet address.
- **Proof Verification.** If a record of claim of address ownership has been recognized as new and passes through the consistency check, *verifyProof* is used to verify proofs π_1, π_2. In our case, the $Verify$ algorithm of Groth16 is hardcoded in the script, and *BB Contract* is as the verifier of the zkSNARK.
- **Storing the Record.** After the record passes through the proof verification, it is stored by the smart contract in the blockchain.
- **Data Retrieval.** As long as the record is stored in the blockchain, every blockchain user can retrieve the record.

D. PoSEAO With a Bulletin Board

By publishing the record of *SC* to a bulletin board *BB Contract*, we show that proof of single-entity address-ownership [17] can be achieved among all users of the bulletin board who publish their records on the bulletin board in time. In this case, whenever a claim of address ownership is sent from the asset holder to the third party checker, the checker can retrieve the address' record from the bulletin board. If there is no any record about the wallet address, the third party checker can request the asset holder to store a record of claim of address ownership to the bulletin board. If there is a record about the wallet address, and the result of claim function PRF matches with e in the record, then the third party checker can confirm that the asset holder is the unique owner of the wallet address among all users of the bulletin board. If there is a record about the wallet address, but the result of claim function PRF does not match with e in the record, the third party checker can reject the request.

Remark. There are two fundamental problems when we use a bulletin board to record the proofs of claim of address ownership and prevent double claim:

- finality time of the bulletin board affects the verification time of claim of address ownership.
- some asset holders are not willing to publish their records on the bulletin board. Despite the entity identity is not published, the owner of the wallet address that has a large mount of assets (the whale wallet address) generally does not want to show their wallet address' activities on the bulletin board.

In addition, due to the policy issues (e.g., some permissionless blockchains are forbidden in some regions), it is hard to

```
pragma solidity ^0.8.30;

import "./VerifierKey.sol";
import "./VerifierPRF.sol";

contract RecordProofs {
    VerifierKey public immutable keyVerifier;
    VerifierPRF public immutable prfVerifier;

    struct Record { bytes32 e; bool exists; }
    mapping(bytes32 => Record) private records;
    // key = addrBE

    event NewRecord(bytes32 indexed addrBE,
    bytes32 e);

    constructor(address _vkKey, address _vkPrf)
    {
        keyVerifier = VerifierKey(_vkKey);
        prfVerifier = VerifierPRF(_vkPrf);
    }

    function submit(
        VerifierPRF.Proof calldata  proof1,
        uint256[2]    calldata
        pubPrfSignals  // [addr,e]

        VerifierKey.Proof calldata  proof2,
        uint256[1]    calldata
        pubKeySignals, // [addr]
    ) external {
        /*  0. duplicate guard (cheap)  */
        bytes32 addrBE =
        bytes32(pubKeySignals[0]);
        require(!records[addrBE].exists, "addr
        has been recorded!");

        /*  1. public signals must match across
        circuits  */
        require(pubKeySignals[0] ==
        pubPrfSignals[0], "addr mismatch");

        /*  2. Groth16 verifications  */
        require(prfVerifier.verifyProof(proof1,
        pubPrfSignals), "bad proof1");
        require(keyVerifier.verifyProof(proof2,
        pubKeySignals), "bad proof2");

        /*  3. store & emit  */
        bytes32 e = bytes32(pubPrfSignals[1]);
        records[addrBE] = Record({ e: e, exists:
        true });
        emit NewRecord(addrBE, e);
    }

    function hasRecord(bytes32 addrBE) external
    view returns (bool) {
        return records[addrBE].exists;
    }

    function getRecord(bytes32 addrBE) external
    view returns (bytes32) {
        require(records[addrBE].exists, "record
        not found");
        return records[addrBE].e;
    }
}
```

Fig. 6. *BB Contract* to store unique record for each wallet address.

request world-wide users to use the same bulletin board. The third problem is:

- the world-wide users might publish their records on different bulletin boards. Without a consensus protocol among different bulletin boards, the double claim attack is feasible just like what we described in Section II.

V. Deterring Double Claim in the Presence of Network Partition

This section first introduces self-deterrence Σ-protocol to show the feasibility of constructing a self-deterrence proof to deter the double claim. To ensure that the asset holder furnish the same secret key as the private input of *SC* and self-deterrence protocol, this section then presents self-deterrence carrying proof protocol.

In practice, two idealized assumptions break down almost immediately: 1), not every asset holder will bother to post an ownership proof to a designated bulletin board, and 2), not every financial service provider will consult the same bulletin board, or even agree on which board is canonical. Either lapse fractures the communication fabric and, from a distributed systems standpoint, behaves like a network partition. Because the CAP theorem indicates that it is impossible to maintain both strong consistency and uninterrupted availability under partition, we need to settle for the next-best objective: keep the service live while making inconsistency prohibitively costly. The remainder of our design therefore focuses on replacing strict prevention with economic and cryptographic deterrence, so that any actor who attempts to exploit divergent views of the system faces an expected loss that outweighs the potential gain.

A. Intuition

We use cryptographic primitives to enable the asset holder to use her secret key and an indicator to generate a proof π_{SD}. This proof alone does not reveal anything about the secret key. If the asset holder uses the same secret key with a different indicator to generate alternative proof π'_{SD}, the secret key can be extracted from $\pi_{SD}.\pi'_{SD}$. We call this proof a self-deterrence proof.

We use zkSNARK to merge self-deterrence proof with SC to construct self-deterrence carrying proof protocol to deter double claim in the following scenarios: 1, record has been submitted to the bulletin board but has not been finalized; 2, some users do not publish their records on any bulletin board; 3, multiple inconsistent bulletin boards are used by world-wide users.

B. Self-deterrence Σ-protocol

To embed an economic "booby trap" inside every credit request we design a Σ-protocol, an interactive three-move proof of knowledge that can be compiled to a non-interactive zkSNARK with Fiat–Shamir protocol. The protocol lets a prover demonstrate, in zero knowledge, that the prover has revealed precisely the subset of secret-share pairs mandated by a public 128-bit indicator. Because each private key is

first split into 128 independent (2,2) Shamir shares, publishing any complementary subset inevitably leaks the entire key. A rational asset holder is therefore deterred from producing a second proof under a different indicator: doing so would allow any verifier to reconstruct the key and sweep the wallet.

Public Parameter:
$\mathbb{G}$ be a cyclic elliptic curve group of order q with generator g.
$H : \{0,1\}^* \to \{0,1\}^l$ be a cryptographic hash function.
$PRF : \{0,1\}^* \to \mathbb{Z}_q$ be the pseudorandom function.
$f : x \to y$ be the wallet address generation function.

zk-Self-deterrence:
Public Input: $eID, addr$
Private Input: sk

1, Self-deterrence Generation:
a, $\mathcal{P}$ computes:
$e = H(eID||addr)$;
$\{b_1, b_1, ..., b_l\}$ denote the bits of e.
b, $\mathcal{P}$ generates secret shares:
for $i \in \{1, ..., l\}$:
$S_{i,0} = PRF(sk, i, 0, addr)$
$S_{i,1} = sk - S_{i,0}$
c, Commitment:
for $i \in \{1, ..., l\}$:
$r, r_{i,0}, r_{i,1}, \rho_i \in_R \mathbb{Z}_q$
$T_i = h^{\rho_i}$
$C_{i,0} = g^{S_{i,0}} \cdot h^{r_{i,0}}$, $C_{i,1} = g^{S_{i,1}} \cdot h^{r_{i,1}}$, $C_{sk} = g^{sk} \cdot h^r$
d, Challenge:
$H(eID, addr, \{C_{i,0}, C_{i,1}\}_{i=1}^{l}, C_{sk}, \{T_i\}_{i=1}^{l}) = c$
e, Response.
for $i \in \{1, ..., l\}$:
$z_i = \rho_i - (r_{i,0} + r_{i,1} - r) \cdot c$
f, $\mathcal{P}$ generates proof of self-deterrence:
For $i \in \{1, ..., l\}$:
$\pi_{SD} = \{S_{i,b_i}, r_{i,b_i}\}_{i=1}^{l}$

2, For $i \in \{1, ..., l\}$, $\mathcal{V}$ checks:
$C_{i,b_i} \stackrel{?}{=} g^{S_{i,b_i}} \cdot h^{r_{i,b_i}}$
$\frac{C_{i,0} \cdot C_{i,1}}{C_{sk}} \stackrel{?}{=} h^{r_{i,0} + r_{i,1} - r}$

Fig. 7. Self-deterrence Σ-protocol.

C. Self-deterrence Carrying Proof Construction

Self-deterrence carrying means a self-deterrence is contained in a sound claim. The prover must use the same secret key to generate self-deterrence proof and make a sound claim.

We use zkSNARK to construct self-deterrence carrying proof, which includes the following protocols:

- $zkSNARK(PRF(eID, addr) = e)$. Public input: $addr$, public output: e, private input: eID. $zkSNARK(PRF(eID, addr) = e) \to \pi_1$.
- $zkSNARK(f(sk) = addr \wedge \mathcal{SS}(sk) = 128 * (2,2)shares \wedge output(e, 128 * (2,2)shares) \to 128shares)$. Public input: e, public output: $addr, 128shares$, private input: sk. $zkSNARK(f(sk) = addr \wedge \mathcal{SS}(sk) = 128 * (2,2)shares \wedge output(e, 128 * (2,2)shares) \to 128shares) \to \pi_2$.

Consistency analysis. The adversary cannot use two different secret keys sk, sk' to generate proof of sound claim SC and self-deterrence carrying proof SD respectively. We use zkSNARK to combine multiple computations into one arithmetic circuit, which ensure the consistency between different computations with the same secret input.

Computations:
$f : x \to y$ be the wallet address generation function.
$\mathcal{SS} : (sk, i), i \in \{1, 2, ..., l\} \to s_{i,0}, s_{i,1}$ be the (2,2) threshold secret sharing function.
$\mathcal{SR} : (\{ind_i\}_{i=1}^{i=l}, \{s_{i,0}, s_{i,1}\}_{i=1}^{i=l}) \to \{s_{i,ind_i}\}_{i=1}^{i=l}$ be the selective release function.
3. *Self-deterrence Carrying*
Arithmetic circuit: $\mathbf{C_3} \xleftarrow{circom} f \wedge \mathcal{SS} * l \wedge \mathcal{SR}$
Public input: $\{ind_i\}_{i=1}^{i=l}$
Prover $\mathcal{P}$'s private input: sk,
Public output: $addr, \{s_{i,ind_i}\}_{i=1}^{i=l}$
Proof: π_3
$setup(\mathbf{C_3}) \to ppk_3, pvk_3$
$prove(\mathbf{C_3}, ppk_3, (\{ind_i\}_{i=1}^{i=l}, addr), sk) \to \pi_3$
$verify(\mathbf{C_3}, pvk_3, (\{ind_i\}_{i=1}^{i=l}, addr), \pi_3) \stackrel{?}{=} 1$

Fig. 8. Self-deterrence Carrying.

D. Effectiveness

When SD is requested by the trusted third party, the double claim attacker would face the risk:

- Win. The attacker transfers all funds from the claimed address to an unclaimed address after report is issued and before SD meets SD';
- Draw. The attacker transfers all funds from the claimed address to an unclaimed address before report is issued and SD meets SD';
- Lose. The attacker can not transfer all funds from the claimed address to an unclaimed address before SD meets SD'.

E. Preventing Service Stacking Attack

Publishing an address ownership claim on a *BB Contract* or any append-only blockchain log suffices to prevent the classic double-claim attack: conflicting claim is visible to everyone and can be rejected. Unfortunately, the same recipe breaks down for service or loan stacking. A debt position is not a simple, static attribute of an address. Borrowers are reluctant to advertise their liabilities, and service providers or lenders are disinclined to disclose proprietary credit information. Even

if we encrypted the loan details in a zero-knowledge record, hiding the principal defeats the very cross platform aggregation we need to spot excess leverage. Moreover, it is hard to require every service/lending venue to synchronize its customer registry with every other venue in real time.

In principle, one could try to: 1), force borrowers to log each new debt on-chain; 2),force lenders to log each loan they issue; or 3), leave the state off-chain but impose an immediate, catastrophic penalty on any borrower who is caught stacking loans or other services. Options 1 and 2 focus on the same adoption and anonymity barriers that plague the BB approach. We therefore pursue option 3: make service/loan stacking economically irrational by coupling every credit request with a self-deterrence (SD) proof that can be cross-checked, privately, by checkers. We describe the scheme as follows:

- 1. the borrower computes an SD proof by using $(sk, eID, serviceID)$, where sk is the wallet's secret key, eID is the borrower's real world identity, and $serviceID$ uniquely names the financial service provider's venue;
- 2. the borrower transmits SD to the prospective checker;
- 3. checkers exchange the SD over private, authenticated channels;
- 4. If the same address appears in two inconsistent SDs (i.e., tied to different serviceIDs or eIDs), the complementary shares embedded in the proofs enable any checker to reconstruct sk and immediately sweep the funds;
- 5. If no conflict is detected, the service provider proceeds with the usual credit assessment workflow and, if satisfied, disburses the financial service (e.g., loan);
- 6. Because a rational borrower cannot predict which financial service providers talk to which, the expected loss from loan stacking outweighs any temporary liquidity gain, restricting each address to one live, unsecured loan at a time.

The current construction is one-off: once an SD proof for a given address has been consumed, either because the service (e.g, loan is repaid) or because the deterrent trigger fires, the address cannot safely be used in a fresh financial service application. Enabling reusability without diluting the deterrent property is an open research question.

VI. Implementation and Evaluation

A. ZKP Implementation

Front-end. We use Circom language to specify the circuits, and compile the circuits to R1CS. The circuits of zero knowledge proof of sound claim and self-deterrence carrying proof have 107 handwritten lines of circom code.

Back-end. We implement zero knowledge proof of sound claim and self-deterrence carrying proof by using $(Setup, Prove, Verify)$ of Groth16 in Snarkjs.

B. Bulletin Board Implementation

We instantiate the bulletin board as smart contract in Solidity, which can be implemented to any EVM-compatible chain. The smart contract can: 1), verify Groth16 proofs; 2), store the deduplicated records of *zkSC* and *SD-carrying-proof* into the blockchain.

C. Circuit Benchmarks

We evaluate the cost of circuits by implementing $(Setup, Prove, Verify)$ of Groth16 in Snarkjs on a Lenovo ThinkCentre M93p workstation equipped with an Intel Core i7-4790 CPU and 32 GB of RAM, running Ubuntu 22.04.5 LTS operating system.

- *zkSC-1* circuit. 600 constraints for a Poseidon based Pseudo-Random Function;
- *zkSC-2* circuit. 400,000 constraints for the wallet address generation function of Ethereum;
- *SD-carrying* circuit. 76,800 constraints for 128 unpaired secret shares. 400,000 contraints for the wallet address generation function of Ethereum.

D. Cost of Bulletin Board

We use smart contract to:

- verify Groth16 proofs. 420,000 gas.
- store a unique record for each wallet address. 20,000 gas.
- allow the subscribers of the bulletin board to retrieve the data. Off chain call data is free.

In total, posting a record of *zkSC* and *SD-carrying-proof* for a wallet address in Ethereum costs 445,000 gas.

VII. Related Works

A. zkSNARK of Digital Signatures

Although zkSNARKs can verify conventional signatures, using a sign-then-prove workflow to establish wallet ownership remains problematic. First, the verification logic inflates circuit size to hundreds of thousands of constraints, increasing proving time, gas costs, and hardware requirements far beyond those of hash-based statements. Second, designing an immediate and automatic penalty for double claim behavior is inherently hard: the wallet owner can sign two unrelated identity strings and post two independent proofs, while an on-chain verifier sees only opaque proofs and lacks the linkage information needed to detect or punish this equivocation in real time.

B. Loan Stacking

Loan stacking means the rapid, undisclosed accumulation of multiple credit lines, which has long been recognized as a blind spot in digital consumer lending. Classic P2P lending studies [18] showed that hidden indebtedness degrades screening accuracy when lenders rely on siloed credit files, and FinTech balance sheet lenders further amplified the problem with instant approval mobile apps whose underwriting pipelines do not synchronize in real time [19]. In the Web3 landscape, an increasing number of traditional financial institutions now accept digital assets as creditworthiness for real-world services. This convergence, however, introduces a new threat: fraudulent loan stacking, where a borrower pledges the same on-chain assets to obtain multiple, undisclosed credit lines for the purpose of bust-out[3]. The root cause is the lack of a unified registry that links blockchain activity with off-chain

[3]https://kount.com/glossary/bust-out

lending records. In this paper, we take an initial step toward closing that gap and propose a framework for preventing blockchain loan stacking attacks.

VIII. Conclusion

In this paper, we formally define blockchain stacking fraud attacks and conduct empirical case studies to identify conditions under which such attacks are feasible in the broader blockchain ecosystem. We propose schemes designed for two deployment scenarios (with and without a trusted bulletin board) to mitigate blockchain stacking fraud vulnerabilities.

References

[1] M. Kelkar, K. Babel, P. Daian, J. Austgen, V. Buterin, and A. Juels, "Complete knowledge: Preventing encumbrance of cryptographic secrets," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 2415–2429.

[2] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 720–731.

[3] Binance, "Binance," https://www.binance.com/en/my/wallet/verification, 2025, online; accessed 7 Mar. 2025.

[4] Z. Wilcox, "Proving your bitcoin reserves," https://bitcointalk.org/index.php?topic=595180.0, 2014, online; accessed 7 Mar. 2025.

[5] Bybit, "Bybit," https://www.bybit.com/app/user/audit-report, 2025, online; accessed 7 Mar. 2025.

[6] Coinbase, "Coinbase," https://www.coinbase.com/blog/how-crypto-companies-can-provide-proof-of-reserves, 2025, online; accessed 7 Mar. 2025.

[7] OKX, "Okx," https://hacken.io/case-studies/okx-por/, 2025, online; accessed 7 Mar. 2025.

[8] HTX, "Htx," https://www.htx.com/en-us/proof-of-reserve, 2025, online; accessed 7 Mar. 2025.

[9] Bitget, "Bitget," https://www.bitget.com/proof-of-reserves, 2025, online; accessed 7 Mar. 2025.

[10] Kraken, "Kraken," https://www.kraken.com/proof-of-reserves, 2025, online; accessed 7 Mar. 2025.

[11] K. Hu, Z. Zhang, and K. Guo, "Breaking the binding: Attacks on the merkle approach to prove liabilities and its applications," *Computers & Security*, vol. 87, p. 101585, 2019.

[12] KuCoin, "Kucoin," https://www.kucoin.com/proof-of-reserves, 2025, online; accessed 7 Mar. 2025.

[13] Gate.io, "Gate.io," https://www.gate.io/proof-of-reserves, 2025, online; accessed 7 Mar. 2025.

[14] Bitfinex, "Bitfinex," https://www.bitfinex.com/, 2025, online; accessed 7 Mar. 2025.

[15] K. Chalkias, K. Lewi, P. Mohassel, and V. Nikolaenko, "Practical privacy preserving proofs of solvency," *Amsterdam ZKProof Community Event*, 2019.

[16] V. Buterin, "Having a safe cex: proof of solvency and beyond," https://vitalik.eth.limo/general/2022/11/19/proof_of_solvency.html, 2022.

[17] J. Austgen, A. Fábrega, M. Kelkar, D. Vilardell, S. Allen, K. Babel, J. Yu, and A. Juels, "Liquefaction: Privately liquefying blockchain assets," *arXiv preprint arXiv:2412.02634*, 2024.

[18] Y. Li, K. Gao, Y. Duan, and W. Xu, "Octopus: Privacy-preserving collaborative evaluation of loan stacking," *arXiv preprint arXiv:2007.02234*, 2020.

[19] G. Buchak, G. Matvos, T. Piskorski, and A. Seru, "Fintech, regulatory arbitrage, and the rise of shadow banks," *Journal of financial economics*, vol. 130, no. 3, pp. 453–483, 2018.