

The Consensus Number of Untraceable Cryptocurrencies

Christian Cachin  

Institute of Computer Science, University of Bern, Switzerland

David Lehnherr  

Institute of Computer Science, University of Bern, Switzerland

Juan Villacis  

Institute of Computer Science, University of Bern, Switzerland

François-Xavier Wicht  

Institute of Computer Science, University of Bern, Switzerland

Abstract

Privacy-preserving cryptocurrencies hide which account funds a transfer among a set of candidate accounts, called its *masking set*. To prevent the hidden account from being spent twice, classical schemes retain the candidates and append a *nullifier* that marks the spent account without revealing it. Their ledgers therefore grow with every transfer. Constant-state schemes instead consume and replace the entire masking set. Here, we study the synchronization cost of this second design.

We introduce the *constant untraceable asset transfer* (CUAT) object. A CUAT transfer atomically replaces every account in its masking set; hence two transfers with intersecting sets cannot both succeed. We capture this contention by a conflict graph on transfer invocations, whose edges join invocations with intersecting masking sets. In one-round protocols, the relevant sets form a clique at a critical configuration. In general protocols, only sets from opposite valency classes must intersect.

Under weak untraceability, which considers one transaction in isolation, CUAT has unbounded consensus number even for one-round protocols. Under strong untraceability, which considers the complete history, accounts in a common masking set must have equal incidence. This condition makes both the one-round consensus power and CUAT's consensus number grow quadratically with the masking-set size. Cyclic and projective-plane constructions establish the corresponding one-round lower bounds, while a recursive grid construction shows that CUAT's consensus number is exactly the square of the masking-set size. CUAT is also not starvation-free.

The full version studies the complementary linear untraceable asset transfer (LUAT) object, which retains its masking set and records a nullifier. Its state grows, but its consensus number is two for every masking-set size and under either untraceability notion, and it is starvation-free. Thus LUAT pays for untraceability in storage, whereas CUAT pays in synchronization and fairness.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Concurrent algorithms; Security and privacy → Pseudonymity, anonymity and untraceability

Keywords and phrases Consensus number, untraceability, privacy-preserving cryptocurrencies, wait-freedom, masking sets, concurrent objects

Digital Object Identifier 10.4230/LIPIcs.AFT.2026.5

Related Version *Full Version:* <https://eprint.iacr.org/2026/1503> [10]

Funding This work was supported by the Swiss National Science Foundation (SNSF) under grant agreement Nr. 219403 (Emerging Consensus) and by the Initiative for Cryptocurrencies and Contracts (IC3).



© Christian Cachin, David Lehnherr, Juan Villacis, and François-Xavier Wicht;
licensed under Creative Commons License CC-BY 4.0

8th Conference on Advances in Financial Technologies (AFT 2026).

Editors: Aggelos Kiayias and Maria Kyropoulou; Article No. 5; pp. 5:1–5:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Cryptocurrencies enable peer-to-peer value transfer without trusted intermediaries. They maintain a replicated ledger that records transactions and the current state of all accounts. Public verifiability makes this ledger auditable, but also exposes a permanent record of users' financial activity. Pseudonymous addresses provide limited protection because transactions can be linked to one another and combined with off-ledger information. The resulting transaction graph may reveal payments, balances, business relationships, or personal associations. Privacy-preserving cryptocurrencies seek to retain public verification without making this financial history attributable to individual users.

One central goal is sender untraceability, which hides the sender of a transfer among a set of candidate accounts, its *masking set*. The difficulty is to prevent the hidden account from being spent twice. Classical systems retain every candidate and append a nullifier for the spent account. This preserves the masking set but makes the ledger grow with every transfer.

Another approach, used by *Quisquis* and related systems [16, 3, 29], replaces all accounts in a fixed-size masking set after every transfer. Only the current accounts and balances remain, but account replacement creates contention: two transfers using a common account cannot both succeed. Does this contention require consensus? How does the privacy requirement constrain its synchronization power? And can an eligible transfer be excluded indefinitely?

This tension is easy to miss when looking only at the cryptographic protocol. Both approaches hide one account among the same number of candidates, but they use those candidates differently. In the classical approach, the masking set is evidence used by a proof and remains available after the transfer. In the constant-state approach, it is also mutable ledger state: a successful transfer invalidates every account in the set and renders every proof about the old state stale. The mechanism that bounds storage thereby turns privacy choices into conflicts between otherwise independent users. We therefore ask:

What synchronization and progress costs arise when a cryptocurrency consumes its masking sets in order to keep the ledger state bounded?

1.1 Our Contribution

We answer these questions in Herlihy's consensus hierarchy [23]. The consensus number of an object is the largest number of processes that can solve wait-free consensus using that object and atomic registers. We introduce the *constant untraceable asset transfer* (CUAT)¹ object, extending the asset-transfer abstraction of Guerraoui *et al.* [21]. A λ -CUAT transfer atomically replaces all λ accounts in its masking set while hiding the sender and the balance update. The object isolates the synchronization caused by masking-set replacement from the surrounding cryptocurrency protocol.

Consensus numbers measure the synchronization power inherent in the object, not the number of validators that every transaction must contact. A high consensus number means that a sufficiently contended use of the object can solve agreement among many processes. Disjoint operations may still proceed without coordination. This distinction is important for CUAT: the cost is determined by the intersections among masking sets, rather than by the total workload or the size of the validator set.

We distinguish two adversarial views. Weak untraceability hides the sender of each transfer in isolation. Under this notion, CUAT has unbounded consensus number, even when every process invokes only one transfer. Strong untraceability instead protects against

¹ Pronounced *Coo-at*.

an observer of the complete history. We characterize this notion by a uniform-incidence condition: accounts that occur in a common masking set must appear equally often in the history. This condition limits the conflict structures that a private execution may expose.

We determine the resulting consensus power in two protocol regimes. A *one-round* protocol models a payment attempt whose masking set is fixed before submission. For every $\lambda \geq 3$, its consensus number lies between $\lfloor \lambda^2/2 \rfloor$ and $\lambda^2 - \lambda + 1$; the upper bound is attained whenever $\lambda - 1$ is a prime power. General protocols may invoke CUAT repeatedly and choose later masking sets based on earlier outcomes. Their additional flexibility closes the gap: under strong untraceability, CUAT has consensus number exactly λ^2 for every $\lambda \geq 2$. Thus the full synchronization power of the object can be strictly larger than the contention exposed by a single payment attempt.

Finally, we connect these worst-case consensus bounds to execution behavior. For a fixed workload, the conflict graph determines both its minimum sequential depth and its maximum one-round throughput. The same conflict mechanism also precludes starvation-freedom under an unfair asynchronous scheduler: a process may complete every invocation yet never succeed. This establishes a formal denial-of-service limitation for constant-state masking-set replacement.

1.2 Technical Approach

Our analysis starts from the *conflict graph*. Its vertices are transfer invocations, and two vertices are adjacent when their masking sets share an account index. Non-adjacent transfers commute; adjacent transfers cannot both succeed. This graph separates the cryptographic content of a transaction from the concurrency structure relevant to consensus: once masking sets are fixed, only their intersections matter. We connect these intersections to consensus through the standard critical-state argument. At a critical configuration, every next step leads to a univalent configuration, and the processes split according to the decision value forced by that step. Pending transfers of opposite valency must conflict. Otherwise, the two steps commute and lead to indistinguishable configurations that must decide different values, which is impossible.

The resulting graph condition depends on the protocol. In a *one-round* protocol, each process invokes one transfer with a fixed masking set. Applying the critical-state argument with each process as the distinguished process forces all masking sets to pairwise intersect. Conversely, any such clique gives an attempt-and-adopt consensus protocol: the first transfer succeeds, every other transfer fails, and the losers adopt the winner's registered input. Thus one-round consensus power equals the largest admissible clique (Theorem 11). In a general protocol, processes may invoke several transfers and adapt their sets. The critical configuration then guarantees only a complete bipartite intersection between the two valency classes. This weaker structure permits more processes than any one-round protocol.

This distinction separates two interpretations of CUAT. The one-round regime models a payment attempt whose masking set is chosen before submission: each user gets one chance to contend on the object. The general regime asks for the full position of CUAT in the consensus hierarchy: a protocol may use several objects, observe earlier outcomes, and choose later masking sets adaptively. The two regimes are governed by different subgraphs (cliques in the first, complete bipartite graphs in the second) and consequently have different bounds.

We first consider weak untraceability. It considers one transaction in isolation and places no global constraint on masking-set reuse. We put one hub account in arbitrarily many masking sets. They form an arbitrarily large clique, while the cryptographic proof hides the sender within each individual set. The attempt-and-adopt construction therefore solves consensus for any number of processes, already in one round: $\text{cons}(\text{CUAT}_{\lambda}^-) = \infty$.

The hub construction also explains why the weak notion is insufficient for a history-aware observer. The hub appears in every masking set and therefore becomes recognizable as a likely decoy, even though each zero-knowledge proof reveals nothing about the sender of its transaction in isolation.

Strong untraceability changes the picture because it lets the adversary observe the complete history. Repeated appearances then reveal information that is invisible in an isolated transaction. For an adversary that treats account appearances as independent, we show that it holds exactly when two accounts in a common masking set have the same incidence in the history (Theorem 14); the direction our bounds use, that strong untraceability forces equal incidence, holds against any stronger adversary. For a connected masking-set family, every occurring account consequently has the same incidence (Corollary 15). This converts a probabilistic privacy condition into a combinatorial constraint on the conflict graph, and is what makes the consensus bounds finite.

For one-round protocols, the masking sets are both uniformly incident and pairwise intersecting. If every account has incidence m , distinct ownership and double counting give $m \leq \lambda$. Fixing one masking set, each of its λ accounts occurs in at most $\lambda - 1$ other sets; hence the clique has size at most $1 + \lambda(\lambda - 1) = \lambda^2 - \lambda + 1$. A projective plane meets this bound when $\lambda - 1$ is a prime power. For every $\lambda \geq 3$, a cyclic difference cover gives a clique of size $\lfloor \lambda^2/2 \rfloor$.

These constructions also explain the shape of the bound. Projective planes distribute intersections perfectly: every pair of masking sets meets once and every account occurs equally often. Cyclic difference covers retain uniform incidence for every λ , at the cost of a smaller family.

The general case uses the weaker bipartite intersection exposed by a critical configuration. Let the two valency classes have sizes s_0 and s_1 . Every one of the $s_0 s_1$ cross-class pairs must share an account. If an account occurs i_a times on one side and j_a times on the other, it covers $i_a j_a$ pairs. Uniform incidence fixes $i_a + j_a = m \leq \lambda$. Summing over accounts and applying Cauchy–Schwarz yields $s_0 + s_1 \leq \lambda m \leq \lambda^2$. We match this bound recursively. The base protocol lets λ processes contend on one masking set. To combine protocols for λu and λw processes, we place the two groups on the rows and columns of a $\lambda \times \lambda$ account grid. Every row meets every column, each account has incidence $u + w$, and the winning group carries one value from its recursive protocol. Taking $u + w = \lambda$ gives $\text{cons}(\text{CUAT}_\lambda^+) = \lambda^2$ for every $\lambda \geq 2$.

The grid construction is recursive because a process first agrees with the members of its own group and then uses a row or column to choose between the two group values. The groups need not have equal size. This is essential when λ is odd and is why the exact quadratic bound holds without a parity or prime-power restriction.

Finally, we translate the conflict structure back to system behavior. The consensus number is a worst-case measure of contention, not a requirement to order every transfer. Disjoint masking sets commute and can be processed independently. For a fixed workload, the conflict graph makes this precise: its chromatic number is the minimum sequential depth, and its independence number is the maximum one-round throughput. Conflict also prevents fairness. An unfair scheduler can repeatedly commit one process's zero-value transfer before a competitor on the same masking-set lineage, making every invocation of the competitor fail. Thus CUAT is not starvation-free, formalizing a conjectured denial-of-service behavior of *Quisquis* [9].

Together, these results identify a three-way connection between privacy, storage, and synchronization. Increasing λ enlarges the anonymity set, but under strong untraceability it also permits a conflict structure with quadratic consensus power. This is a worst-case

statement: it does not force disjoint transfers through consensus. It does show that a constant-state design must control how masking sets are selected and how conflicting transactions are scheduled. Fair ordering, unpredictable set selection, and contention control therefore complement the cryptographic privacy mechanism rather than merely optimize it.

The full version [10] studies the complementary linear untraceable asset transfer (LUAT) object, which retains masking sets and grows the ledger. LUAT has consensus number 2 for every masking-set size and is starvation-free. The comparison isolates the design cost: LUAT pays for untraceability in storage, whereas CUAT pays in synchronization and fairness.

Section 2 discusses related work, and Section 3 gives the model. Section 4 defines CUAT and proves the consensus bounds. Section 5 studies parallelism and starvation; Section 6 concludes.

2 Related Work

Our analysis of CUAT's synchronization power connects three distinct research areas: consensus numbers of cryptocurrency objects and privacy-preserving primitives, quorum systems and combinatorial design theory, and multi-word compare-and-swap primitives.

Guerraoui *et al.* [21] challenge the assumption that consensus is necessary to transfer assets. By modeling asset transfers as concurrent objects, they show the consensus number of standard asset transfer is 1, which means that transactions can be processed asynchronously without total ordering. The result concerns the asset-transfer object alone, and not a cryptocurrency as a whole: issuance mechanisms such as mining, as well as smart-contract state, lie outside the object and need not have consensus number 1. Building on this insight, modern cryptocurrencies like Sui [8], Libra/Diem, and Zef [5] adopt their k -shared asset transfer model, where only the k users sharing an account need consensus, thereby reducing synchronization overhead. Beyond basic transfers, Ethereum's ERC20 token standard introduces additional complexity. As Alpos *et al.* [2] observe, ERC20 tokens allow account owners to approve multiple spenders, which in turn creates a dynamic consensus number where synchronization requirements depend on the number of authorized spenders. Importantly, this demonstrates that consensus requirements in smart contracts are state-dependent rather than fixed, thereby enabling more efficient synchronization mechanisms.

Privacy-preserving systems, however, face distinct synchronization challenges. Frey *et al.* [18] analyze the stronger allow-list and deny-list objects, showing that while allow-lists need no synchronization, deny-lists require consensus among verifiers performing set-non-membership proofs. More broadly, their framework applies to decentralized identity management and e-voting.

CUAT's pairwise-intersecting masking sets bear structural similarities to quorum systems [33, 20]. Both use intersections among sets of elements to provide synchronization in distributed systems. The key distinction is that quorum systems are designed to tolerate process failures and must satisfy constraints derived from failure assumptions, whereas masking sets in CUAT need not account for such failures. Despite this difference, both serve as synchronization abstractions. We show that CUAT requires additional conditions beyond pairwise intersection. First, to implement consensus, the intersections between masking sets must be specifically structured to identify the winner of a transfer operation. Second, to maintain sender untraceability across multiple transactions, we require uniformity, where each account must appear in the same number of masking sets.

CUAT's transfer operation exhibits structural similarities to multi-word compare-and-swap (MCAS) primitives. Harris *et al.* [22] introduce practical MCAS implementations using single-word CAS operations. Like MCAS, CUAT transfers atomically update multiple

memory locations (account representations in a masking set) and succeed only if all locations match expected values. However, CUAT differs fundamentally due to its privacy requirements and the conditions to maintain sender untraceability.

Christ and Bonneau [12] introduce revocable proof systems (RPS) as a unifying abstraction for authenticated data structures, including stateless blockchains and set commitments. They prove a fundamental trade-off: any scheme with sublinear global state must incur near-linear proof-update costs when statements are revoked (e.g., when coins are spent). This information-theoretic lower bound focuses on storage and witness maintenance rather than concurrency.

3 Preliminaries

This section establishes the theoretical foundations for our analysis. We present the system model, consensus numbers (which measure synchronization power of shared objects), and the consensus hierarchy theorem.

Notation. We use $\leftarrow$ for assignment in algorithms, $=$ for equality testing, $:=$ for definitions, and standard set notation ($\in$ for membership, $\subseteq$ for subset, $\cap$ for intersection, $\cup$ for union).

System Model. We consider N asynchronous sequential processes $p_1, \dots, p_N$ in a set $\mathcal{P}$ that may crash (crash-failure model). Throughout, Π denotes a protocol; the process set is $\mathcal{P}$. Processes interact through shared objects supporting operations that produce responses. Objects satisfy *linearizability* [25], where operation histories appear as if executed in a single, consistent order respecting real-time constraints. A *register* is a shared-memory abstraction supporting *read* and *write* operations. We assume *atomic registers* with linearizable semantics, where each operation appears to take effect instantaneously between invocation and response.

Wait-Freedom. An implementation of a shared object is *wait-free* if every operation invoked by a correct process completes in a finite number of steps, regardless of the behavior of other processes.

Adversarial Model. External adversary $\mathcal{E}$ can *eavesdrop* on network ciphertexts, record shared object states, and analyze process interactions to infer relationships. Though computationally unbounded, the adversary faces idealized cryptographic primitives secure against all attacks.

Consensus Numbers and the Consensus Hierarchy. The consensus number $\text{cons}(O)$ of an object O is the largest number n such that it is possible to wait-free implement a consensus object from atomic registers and objects of type O , in a system of n processes [23]. The consensus hierarchy shows that objects with higher consensus numbers are strictly more powerful and cannot be implemented from objects with lower consensus numbers. For an individual protocol Π we write $\text{cn}(\Pi)$ for the largest n such that Π solves n -consensus.

Valency. Following the standard terminology [23], a configuration C of a binary consensus protocol is *v-valent* (with $v \in \{0, 1\}$) if every execution extending C decides v , *univalent* if it is v -valent for some v , and *bivalent* if both decision values are still reachable from C . We write $C \cdot p$ for the configuration reached when process p takes one step from C , and chain steps as $C \cdot p \cdot q$ for the configuration reached when p then q each take one step. A bivalent configuration is *critical* if every immediate successor $C \cdot p$ is univalent.

Ownership. Both objects we study attach an owner to each account through a map $\mu : \mathcal{A} \rightarrow 2^{\mathcal{P}} \setminus \{\emptyset\}$, and only a process in $\mu(a)$ may spend a or read its balance. An account a is k -shared if $|\mu(a)| = k$, and *unshared* if $|\mu(a)| = 1$; an object is unshared if all of its accounts are. Following Guerraoui *et al.* [21], sharing is the mechanism by which several processes contend for one account. The CUAT_λ objects considered below are unshared (Definition 4). Since a process funds its transfer from an account of its own, at most λ processes can then carry the same masking set.

4 Constant Untraceable Asset Transfer

Quisquis and similar systems [16, 3, 29] address the monotonic growth problem through account replacement: each transfer invalidates all accounts in a masking set and creates new ones, while cryptographic proofs hide the sender. For contrast, the full version [10] describes the linear untraceable asset transfer (LUAT) object, which uses the classical growing allow-set and deny-set approach. We model these constant-state systems through the *constant untraceable asset transfer* (CUAT) object, a shared-memory abstraction building on the asset transfer formalism of Guerraoui *et al.* [21]. CUAT maintains sender untraceability through a masking-set mechanism where each transfer invalidates old accounts and creates new accounts for all members within a fixed-size set. Before presenting the formal model, we first define the untraceability notions that CUAT must satisfy.

4.1 Untraceability

Untraceability requires the adversary’s posterior on the sender to remain uniform over the masking set. We write $P(a_i \mid \cdot)$ for the posterior probability that a_i funded the transfer under consideration; the underlying probability model is specified in Section 4.3.4. We consider both the standard single-transaction guarantee [31, 6] and protection against history-aware deanonymization [28, 32, 15].

► **Definition 1** (Weak untraceability). CUAT_λ satisfies weak untraceability if, for every transfer τ with masking set S and every $a_i \in S$, $P(a_i \mid \tau) = 1/|S| = 1/\lambda$.

Weak untraceability restricts the adversary’s view to the transfer under consideration. Strong untraceability instead conditions on everything the adversary has recorded, including the masking sets of failed transfers.

► **Definition 2** (Strong untraceability). CUAT_λ satisfies strong untraceability if, for every transfer τ with masking set S in any history H and every $a_i \in S$, $P(a_i \mid H) = 1/|S| = 1/\lambda$.

4.2 Formal Model

We now formalize the λ -CUAT object. Each transfer atomically invalidates an old masking set and creates a new one of size λ , with a zero-knowledge proof witnessing ownership of the invalidated accounts.

► **Definition 3** (Masking sets and λ -masking). A masking set is any $S \subseteq \mathcal{A}$. The system is λ -masking if every masking set used by the protocol has size exactly λ .

With masking sets defined, we can specify the CUAT object. The definition below formalizes its state, operations, and transition rules.

► **Definition 4** (CUAT_λ object). CUAT_λ is a shared-memory asset transfer object specified by the tuple CUAT_λ = (Q, q₀, O, R, Δ, U) where:

- **States:** Q consists of pairs $q = (\mathcal{A}, \beta)$ where $\mathcal{A}$ is a finite set of currently valid accounts and $\beta : \mathcal{A} \rightarrow \mathbb{N}$ maps each account to its balance. Each account $a \in \mathcal{A}$ carries two fixed fields assigned at creation: an owner set $\mu(a) = \{p\}$ for a single process p , and an index $\text{idx}(a) \in \mathbb{N}$ also denoted by a subscript. The initial state is $q_0 = (\mathcal{A}_0, \beta_0)$ for some finite $\mathcal{A}_0$.
- **Operations:** The set O consists of: $\text{read}(a)$ for any account a , and $\text{transfer}(S_{\beta'}, \pi)$ where $S_{\beta'} = \{a'_1, \dots, a'_\lambda\}$ is a set of λ new accounts and π is a zero-knowledge proof. The transfer carries an implicit bijection $\phi : S_\beta \rightarrow S_{\beta'}$ between the old and new masking set that preserves both account fields: $\mu(\phi(a)) = \mu(a)$ and $\text{idx}(\phi(a)) = \text{idx}(a)$ for every $a \in S_\beta$.
- **Responses:** R consists of:
 - For $\text{read}(a)$: either $\beta(a)$ if the invoking process owns a , or $\perp$ otherwise.
 - For $\text{transfer}(S_{\beta'}, \pi)$: a pair $(\mathcal{A}, b)$ where $\mathcal{A}$ is the current account set and $b \in \{\text{TRUE}, \text{FALSE}\}$ indicates success or failure.
- **Transitions:** $\Delta \subseteq Q \times O \times R \times Q$ specifies allowed state transitions. For any $q = (\mathcal{A}, \beta)$ and $q' = (\mathcal{A}', \beta')$:
 - **Account read:** If $o = \text{read}(a)$ invoked by process p , then $q' = q$ and

$$r = \begin{cases} \beta(a) & \text{if } a \in \mathcal{A} \text{ and } p \in \mu(a), \\ \perp & \text{otherwise.} \end{cases}$$

- **Transfer success:** If $o = \text{transfer}(S_{\beta'}, \pi)$ with $S_\beta \subseteq \mathcal{A}$, $S_{\beta'} \cap \mathcal{A} = \emptyset$, $|S_{\beta'}| = \lambda$, and π is a proof demonstrating $\exists a_s, a_r \in S_\beta, v \in \mathbb{N}$ such that the invoking process lies in $\mu(a_s)$ and $\beta(a_s) \geq v$, then the new account set is

$$\mathcal{A}' = (\mathcal{A} \setminus S_\beta) \cup S_{\beta'},$$

and the new balance map $\beta' : \mathcal{A}' \rightarrow \mathbb{N}$ is

$$\beta'(a) = \begin{cases} \beta(\phi^{-1}(a)) - v & \text{if } a = \phi(a_s), \\ \beta(\phi^{-1}(a)) + v & \text{if } a = \phi(a_r), \\ \beta(\phi^{-1}(a)) & \text{if } a \in S_{\beta'} \setminus \{\phi(a_s), \phi(a_r)\}, \\ \beta(a) & \text{if } a \in \mathcal{A} \setminus S_\beta. \end{cases}$$

Return $r = (\mathcal{A}', \text{TRUE})$.

- **Transfer failure:** In all other cases of $\text{transfer}(\cdot)$, set $q' = q$ and $r = (\mathcal{A}, \text{FALSE})$.
- **Untraceability:** U specifies the untraceability notion required for the object. We consider two notions: weak untraceability (Definition 1), giving the object CUAT_λ[−], and strong untraceability (Definition 2), giving CUAT_λ⁺. We write CUAT_λ for either variant when the result or argument applies to both. The notion restricts the valid executions of the object, as some use patterns lead to untraceability loss, especially under strong untraceability.

The λ-CUAT models privacy-preserving cryptocurrencies through account invalidation and creation. A transfer takes a set $S_{\beta'} = \{a'_1, \dots, a'_\lambda\}$ of λ new accounts and a zero-knowledge proof π attesting ownership and valid balance updates. On success it atomically invalidates every account of S_β and creates its image under ϕ with updated balances, so old and new accounts occupy the same logical positions under different cryptographic representations. The

proof hides which accounts of S_β are the sender and the recipient, and balance confidentiality restricts balances to account owners. We write τ for the transfer operation and $\tau(S)$ for an invocation that consumes the masking set S ; the operation is invoked with the replacement accounts $S_{\beta'}$, while S_β is what it consumes and hence what governs its conflicts.

A transfer therefore blocks every concurrent transfer that uses any account of S_β , while transfers on disjoint masking sets proceed in parallel. This atomicity is what gives λ -CUAT non-trivial synchronization power, which the next section quantifies.

4.3 Consensus Number

This section characterizes the wait-free consensus number of λ -CUAT. Protocols may use any finite number of λ -CUAT instances together with atomic registers. By Definition 4, the bijection ϕ of every successful transfer preserves each account's index $\text{idx}(a)$, so two accounts with equal index represent the same logical position across rerandomizations; assuming distinct processes never produce identical cryptographic representations,² two masking sets conflict precisely when they share an idx value, and the synchronization structure of a protocol is captured by a graph on its masking sets (Definition 5 below). Two modeling choices distinguish us from earlier work on asset transfer. Guerraoui *et al.* [21] encode the winning process in account balances, a path that CUAT's consensus cannot take since only an account's owner may query its balance; synchronization therefore runs through the masking sets themselves. Untraceability is a runtime constraint [13]: we admit only executions consistent with the notion in force, and the consensus number is taken over that restricted family.

We distinguish protocols by the number of transfers each process may invoke. We first consider the one-round regime, which most directly models the use of CUAT in payment systems, and then turn to the more general multi-round regime.

Section 4.3.1 treats *one-round* protocols (one τ -invocation per process). This models how CUAT is used in payment systems [16, 3, 29]: a user submits a single transaction whose masking set is fixed before submission, and validators either commit it or reject it. The one-round upper bound therefore captures the synchronization cost faced inside one such transaction: validators process non-conflicting transfers independently and only need to coordinate when two transfers contend over a shared masking set, and the upper bound sizes the worst-case coordination slice.

Section 4.3.2 lifts the analysis to *multi-round* protocols, where CUAT is used as a synchronization primitive inside a layered agreement protocol whose outer rounds adapt their masking sets to earlier outcomes. This captures CUAT's theoretical synchronization power but is not the typical operational mode of payment systems. To start our analysis, we first formalize the conflict graph and its clique number.

► **Definition 5** (Conflict graph and clique number). *Two masking sets S and S' on a common CUAT_λ object conflict when there exist $a \in S$ and $a' \in S'$ with $\text{idx}(a) = \text{idx}(a')$. Let $\mathcal{S}(\Pi)$ denote the family of τ -invocations occurring in some execution of Π admissible under the untraceability notion in force, each carrying its masking set. The conflict graph G of Π has a vertex v_S for each invocation in $\mathcal{S}(\Pi)$, labelled by the masking set S it carries, and an edge $(v_S, v_{S'}) \in E$ whenever S and S' conflict; two invocations carrying the same masking set are distinct vertices, and they are adjacent. A clique in G is a set of pairwise-conflicting invocations, and the clique number $\omega(G)$ is the size of the largest clique.*

² Negligible in the security parameter for any concrete instantiation.

The family $\mathcal{S}(\Pi)$ captures every τ -invocation Π may make, including across executions where processes adapt their choices to the schedule. Any single execution realizes a subgraph of G , with clique number at most $\omega(G)$. The upper-bound arguments below inspect the invocation each process is about to make at a critical configuration; these invocations lie in $\mathcal{S}(\Pi)$ by definition, so the bounds hold whether Π chooses masking sets statically or dynamically. The matching lower bounds are realized by protocols whose masking-set assignment is fixed in advance. Alongside the conflict graph, a second invariant tracks how many masking sets contain a given account.

► **Definition 6** (Incidence number). *The incidence of an account $a \in \mathcal{A}$ is the number of vertices $v_S \in V(G)$ such that $a \in S$. The incidence number of Π is $r(\Pi) = \max_{a \in \mathcal{A}} \text{incidence}(a)$.*

Cliques in G implement consensus. When d invocations pairwise conflict, at most one of them can return **TRUE**: any pair shares an index, and the second transfer to commit finds that index already invalidated and fails. Losing transfers inspect the published CUAT state, locate the successful transfer by the new accounts it created, and adopt the value its issuer wrote to a single-writer register. Pairing each of d processes with one clique member yields a d -process consensus protocol that attempts the transfer, decides on success, and adopts on failure; hence $\omega(G)$ lower-bounds the consensus power of λ -CUAT.

The matching upper bounds, both at $\omega(G)$ and beyond, rest on a binary-input specialization of the standard critical-state construction [17, 23].

► **Lemma 7** (Different valencies). *Let Π be a deterministic wait-free consensus protocol for $n \geq 2$ processes in the asynchronous shared-memory model. For any process p and any $v \in \{0, 1\}$, the input assignment I with $I(p) = v$ and $I(s) = 1 - v$ for $s \neq p$ admits a reachable critical configuration C such that $C \cdot p$ is v -valent and $C \cdot s$ is $(1 - v)$ -valent for every $s \neq p$.*

Proof. By symmetry of 0 and 1, take $v = 0$; process p then has input 0 and every other process has input 1. Let σ be the schedule in which p takes no steps and every other process runs to completion. By wait-freedom each $s \neq p$ decides along σ , and every such decision is 1: as p takes no step, σ is indistinguishable to those processes from σ run on the assignment that also gives p input 1, where validity forces 1. Hence σ ends in a 1-valent configuration. The initial configuration is bivalent (σ decides 1 while running p alone decides 0), so σ has a last bivalent prefix, and the standard critical-state construction extends that prefix p -silently to a reachable critical configuration C .

Fix $s \neq p$ and suppose, toward a contradiction, that $C \cdot s$ is 0-valent. Every extension of $C \cdot s$ then decides 0. Consider the extension that runs every process in $\mathcal{P} \setminus \{p\}$ to completion before p takes a step: wait-freedom guarantees termination, and the same indistinguishability argument forces the decision 1, since every process decides before p takes a step. This contradicts 0-valence, so $C \cdot s$ is 1-valent for every $s \neq p$. By bivalence of C , the remaining successor $C \cdot p$ is 0-valent. ◀

In each subsequent application, we extract a reachable critical configuration whose successors split processes by valency, then exploit the commutativity of λ -CUAT transfers on disjoint account indices: two masking sets on opposite valency sides must conflict, since otherwise their transfers commute and produce indistinguishable configurations forced to decide differently. The next lemma packages this technique.

► **Lemma 8** (Bipartite intersection at a critical configuration). *Let Π be a wait-free binary consensus protocol for $n \geq 3$ processes on CUAT_λ objects and atomic registers, and let C be a reachable critical configuration of Π . There exist a CUAT_λ object O and a non-trivial partition $\mathcal{P}_0 \sqcup \mathcal{P}_1 = \mathcal{P}$ such that every pending step at C is a τ -invocation on O , and, writing S_p for the masking set carried by p 's pending invocation:*

- (i) $C \cdot p$ is 0-valent for every $p \in \mathcal{P}_0$ and $C \cdot q$ is 1-valent for every $q \in \mathcal{P}_1$;
- (ii) $S_p \cap S_q \neq \emptyset$ on O for every $(p, q) \in \mathcal{P}_0 \times \mathcal{P}_1$.

Proof. At a critical configuration neither pending step is a register operation [23]: a read leaves the shared state unchanged, writes to distinct registers commute, and of two writes to one register the second overwrites the first, leaving a configuration indistinguishable to every process except the overwritten writer. Each case makes two oppositely valent successors indistinguishable to some process that decides in both. Every pending step at C is therefore a τ -invocation. Define $\mathcal{P}_0 = \{p \in \mathcal{P} : C \cdot p \text{ is 0-valent}\}$ and $\mathcal{P}_1 = \mathcal{P} \setminus \mathcal{P}_0$; both are non-empty by bivalency of C , and property (i) follows.

Fix $p \in \mathcal{P}_0$ and $q \in \mathcal{P}_1$, and suppose toward a contradiction that $\tau(S_p)$ and $\tau(S_q)$ are on distinct λ -CUAT objects, or on a common object with $S_p \cap S_q = \emptyset$. Each transfer's outcome is then unaffected by the other: the two masking sets are disjoint, and by account-representation uniqueness neither transfer's new accounts can appear in the other's masking set. The object states after the pair of steps therefore agree in either order, and neither step writes a register, so the shared state at $C \cdot p \cdot q$ equals that at $C \cdot q \cdot p$. The two configurations are not indistinguishable to p and q themselves, whose transfer responses carry the current account set and hence reveal the order; but they are indistinguishable to every other process. Since $n \geq 3$, choose a spectator $s \notin \{p, q\}$ and run it alone from each configuration. By wait-freedom s decides in both runs, and indistinguishability makes the two decisions equal; yet every extension of $C \cdot p \cdot q$ decides 0, as $C \cdot p$ is 0-valent, and every extension of $C \cdot q \cdot p$ decides 1, as $C \cdot q$ is 1-valent. This contradiction shows that p and q invoke τ on a common λ -CUAT object with $S_p \cap S_q \neq \emptyset$ on that object. Iterating over $(p, q) \in \mathcal{P}_0 \times \mathcal{P}_1$, all cross-valency pairs share a single object O ; transitivity through such pairs forces every process in $\mathcal{P}$ to invoke τ on O , and property (ii) holds with masking sets on O . ◀

In particular, applying the lemma to the critical configuration produced by Lemma 7 with (p, v) gives the *star* specialization, with $\mathcal{P}_v = \{p\}$ and $\mathcal{P}_{1-v} = \mathcal{P} \setminus \{p\}$: p 's pending masking set conflicts on a common object with every other process's pending masking set. Iterating over p is what the one-round upper bound relies on.

4.3.1 One-Round Protocols

Throughout this subsection, Π denotes a wait-free one-round consensus protocol on λ -CUAT objects and atomic registers: each process makes a single τ -invocation and identifies a contested winner from the published CUAT state. Before its τ -step a process has access only to the CUAT's initial state and its own identity, so its masking set is determined by the protocol from these two pieces of data alone. A process may also read registers, but atomic registers have consensus number 1 [23] and add no synchronization power on top of λ -CUAT. The analysis pairs a lower bound from constructive cliques in G with an upper bound from iterating Lemma 8 over each process.

Any clique of G admits a wait-free consensus protocol of matching size, by the standard attempt-and-adopt construction.

► **Lemma 9** (One-round lower bound from cliques). *Let G be a conflict graph admissible under the untraceability notion in force, and let $K = \{S_1, \dots, S_n\} \subseteq V(G)$ be a clique of size n whose members can be assigned to distinct processes so that S_i contains an account owned by p_i . Then there is a wait-free one-round n -consensus protocol Π over a CUAT_λ object in which process p_i uses masking set S_i . Consequently $\text{cons}(\text{CUAT}_\lambda) \geq \omega(G)$ whenever a largest clique admits such an assignment.*

Proof.

Construction. Assign S_i as p_i 's masking set. Each p_i writes its input to a single-writer register r_i (proposal) and invokes $\tau(S_i)$ on the λ -CUAT object. On success (τ returns **TRUE**), p_i decides its own input. On failure, p_i reads the published CUAT state to identify which transfer has been committed, reads the corresponding r_j , and decides that value. Algorithm 1 gives the full pseudocode.

Wait-freedom. The protocol contains no loops: τ is invoked once, and identification of the winner is a single inspection of the published state. Each process completes in a bounded number of steps regardless of the schedule.

Validity. The decided value is either p_i 's own input (on success) or r_{j^*} for some j^* , which is p_{j^*} 's input.

Agreement. Linearize the n τ -invocations on K and let the one issued by p_{j^*} be the first to return **TRUE**. For every other $S_i \in K$, $S_i \cap S_{j^*} \neq \emptyset$ by the clique property of K ; the account shared between S_i and S_{j^*} is invalidated when p_{j^*} commits, so $\tau(S_i)$ fails. Process p_i then identifies p_{j^*} 's transfer in the published state, by the new accounts it created, and decides r_{j^*} . Process p_{j^*} also decides r_{j^*} on success. Every process therefore decides the same value r_{j^*} . ◀

■ **Algorithm 1** Wait-free implementation of d -consensus using a λ -CUAT object.

Shared state: atomic registers $R[1..d]$, $A[1..d]$, a λ -CUAT object CUAT on a finite account set $\mathcal{A}_0$ with $\beta_0 \equiv 0$, a masking-set assignment $i \mapsto S_i \subseteq \mathcal{A}_0$ such that $|S_i| = \lambda$, $S_i \cap S_j \neq \emptyset$ for all $i \neq j$, and p_i owns some $a_i \in S_i$

```

1 function propose( $v$ ) at process  $p_i$ :
2    $R[i].\text{write}(v)$                                      // write proposal
3    $S_{\text{old}} \leftarrow S_i$                                // old masking set
4    $(S_{\text{new}}, \pi) \leftarrow \text{Compute}_{p_i}(S_{\text{old}}, a_i, a'_i, 0)$  // new accounts and proof
5    $A[i].\text{write}(S_{\text{new}})$                                // write new account set
6    $(\mathcal{A}, b) \leftarrow \text{CUAT.transfer}(S_{\text{new}}, \pi)$ 
7   if  $b = \text{TRUE}$  then
8     return  $v$                                            // won: decide own value
9   else
10     $S_{\text{winner}} \leftarrow \mathcal{A} \setminus \mathcal{A}_0$                 // new accounts in  $\mathcal{A}$ 
11    for  $j \in \{1, \dots, d\}$  do
12      if  $A[j].\text{read}() = S_{\text{winner}}$  then
13        return  $R[j].\text{read}()$                           //  $p_j$  won: adopt its value

```

Conversely, Lemma 8 turns wait-freedom into a clique witness in G .

► **Lemma 10** (One-round upper bound from pairwise intersection). *Let Π be a wait-free one-round consensus protocol on CUAT_λ objects and atomic registers, with G its conflict graph. If $\text{cn}(\Pi) \geq 3$, then $\text{cn}(\Pi) \leq \omega(G)$.*

Proof. Let $n = \text{cn}(\Pi) \geq 3$ and write S_i for the masking set carried by p_i 's single τ -invocation, which the protocol fixes from p_i 's identity and the initial state alone.

Fix $p \in \mathcal{P}$ and apply Lemma 7 to the input assignment giving p input 0 and every other process input 1: it yields a reachable critical configuration C at which $C \cdot p$ is 0-valent and $C \cdot s$ is 1-valent for every $s \neq p$. The valency classes at C are therefore $\mathcal{P}_0 = \{p\}$ and $\mathcal{P}_1 = \mathcal{P} \setminus \{p\}$, the star specialization of Lemma 8: every process invokes τ on one common object O , and S_p conflicts with S_s on O for every $s \neq p$.

Iterating over $p \in \mathcal{P}$ makes every two of the n invocations conflict. Distinct processes make distinct invocations, so these are n distinct vertices of G , pairwise adjacent by Definition 5, hence an n -clique. Therefore $\text{cn}(\Pi) = n \leq \omega(G)$. ◀

Combining the two bounds gives equality.

► **Theorem 11** (One-round consensus number). *If Π is a one-round protocol on CUAT_λ with $\text{cn}(\Pi) \geq 3$, then $\text{cn}(\Pi) = \omega(G)$. Consequently, the largest n achievable by a one-round n -consensus protocol on CUAT_λ equals the supremum of $\omega(G)$ over conflict graphs admissible under the untraceability notion in force.*

Proof. Combine Lemma 9 and Lemma 10. ◀

Theorem 11 matches the one-round upper bound to $\omega(G)$, and the bound is the limit of the one-round regime: a committed τ permanently invalidates its masking set, and the surviving masking sets that conflicted with it can no longer fire, so the conflict graph is exhausted after one round of contention. Two follow-up questions remain. Section 4.3.2 relaxes the one-round restriction and asks how far multi-round protocols can push the consensus number past $\omega(G)$. Sections 4.3.3 and 4.3.4 take the orthogonal question of how large $\omega(G)$ can be made under each untraceability notion: weak untraceability leaves $\omega(G)$ unbounded, while strong untraceability caps it at $\lambda^2 - \lambda + 1$.

4.3.2 Multi-Round Protocols

A multi-round protocol lets each process invoke τ more than once and adapt later invocations to earlier outcomes. The structural gain over $\omega(G)$ is not larger cliques but a weaker requirement on the conflict graph: pairwise intersection at the critical configuration can be relaxed once the pending processes split by valency, so the next τ -round need only enforce conflict across the partition. Lemma 8 already extracts such a partition $\mathcal{P} = \mathcal{P}_0 \sqcup \mathcal{P}_1$ with a complete bipartite intersection between the two sides. At incidence 2 a λ -block biclique supports 2λ vertices, exceeding the one-round upper bound.

► **Theorem 12** (Upper bound). *A wait-free binary consensus protocol Π over CUAT_λ objects and atomic registers can solve consensus among at most $\max(2, 2\lambda(r(\Pi) - 1))$ processes.*

Proof. Suppose Π solves n -consensus, so $|\mathcal{P}| = n$; for $n \leq 2$ the bound is immediate, so assume $n \geq 3$. Take C , $\mathcal{P}_0$, $\mathcal{P}_1$ as in Lemma 8; the partition covers $\mathcal{P}$, and cross-valency pending masking sets conflict on a common CUAT_λ object.

Distinct processes make distinct invocations, so $\{v_{S_p} : p \in \mathcal{P}_0\}$ and $\{v_{S_q} : q \in \mathcal{P}_1\}$ are disjoint sets of vertices, of sizes $|\mathcal{P}_0|$ and $|\mathcal{P}_1|$, whether or not two processes happen to carry the same masking set. Property (ii) of Lemma 8 makes every cross pair adjacent, so the two sets span a complete bipartite subgraph $K_{|\mathcal{P}_0|, |\mathcal{P}_1|} \subseteq G$.

Fix $p \in \mathcal{P}_0$. Each account $a \in S_p$ has incidence at most $r(\Pi)$ in G , hence lies in at most $r(\Pi) - 1$ vertices distinct from v_{S_p} . Summing over the λ accounts of S_p , at most $\lambda(r(\Pi) - 1)$ distinct vertices in $V(G)$ are adjacent to v_{S_p} via a shared account. Since every v_{S_q} with $q \in \mathcal{P}_1$ is adjacent to v_{S_p} , we get $|\mathcal{P}_1| \leq \lambda(r(\Pi) - 1)$. By symmetry $|\mathcal{P}_0| \leq \lambda(r(\Pi) - 1)$, so $n = |\mathcal{P}_0| + |\mathcal{P}_1| \leq 2\lambda(r(\Pi) - 1)$. $\blacktriangleleft$

In the following two sections, we discuss how the incidence number $r(\Pi)$ relates to privacy and, through it, to the consensus number.

4.3.3 Weak Untraceability

Under weak untraceability the incidence number $r(\Pi)$ is unconstrained: an account may appear in every masking set without violating per-transaction anonymity. The hub construction places a shared hub account a_1 (owned by p_1) in all d masking sets, making the conflict graph a d -clique, and satisfies weak untraceability for any d . By Theorem 11 this yields $\text{cn}(\Pi) = d$, so the consensus number is unbounded.

► **Theorem 13** (Consensus number of CUAT_λ^-). *For every $d \geq 2$, CUAT_λ^- supports a wait-free one-round d -consensus protocol. Hence $\text{cons}(\text{CUAT}_\lambda^-) = \infty$.*

Proof. We instantiate Lemma 9 on the hub family and verify that the family satisfies weak untraceability.

Construction. Put $M = \max(d, \lambda)$ and take $\mathcal{A}_0 = \{a_1, \dots, a_{M+1}\}$ with $\mu(a_{i+1}) = \{p_i\}$ for $i \leq d$. Let S_i consist of the hub a_1 (owned by p_1) together with the $\lambda - 1$ accounts starting at a_{i+1} and taken cyclically through $a_2, \dots, a_{M+1}$, so that S_i contains p_i 's own account a_{i+1} . Since all d sets share a_1 , they pairwise intersect, so $K = \{S_1, \dots, S_d\}$ is a d -clique in the conflict graph G . Lemma 9 then implements wait-free d -consensus on this clique.

Weak untraceability. The cryptographic proof π of each τ -invocation is zero-knowledge: an observer of τ learns the masking set S_{old} and the published state, but obtains no advantage over the uniform distribution on S_{old} for the sender's identity. Hence $P(a_i \mid \tau) = 1/|S_{\text{old}}| = 1/\lambda$ for every $a_i \in S_{\text{old}}$, regardless of d or of the hub-account structure. Therefore Π achieves wait-free d -consensus and preserves weak untraceability for every $d \geq 2$. $\blacktriangleleft$

The hub construction leaves a_1 in every masking set, so a history-aware adversary identifies it as a near-certain decoy; this statistical leakage motivates strong untraceability.

4.3.4 Strong Untraceability

Strong untraceability protects against an adversary that observes the complete transaction history. We first show that it is equivalent to a local uniformity condition on account incidence. We then use this condition to bound both $\omega(G)$ and $r(\Pi)$. Each account $a \in \mathcal{A}$ has an unknown sender probability q_a and an unknown receiver probability r_a ; decoys are sampled uniformly. The prior is symmetric across accounts. We use the independent-inclusion model: account appearances are independent across accounts and transfers. Consequently, the likelihood of a history depends on each account only through its incidence $f_a(H)$. An adversary with additional information, such as account co-occurrences or side channels, is only stronger; the consensus upper bounds below use only the necessity direction of Theorem 14. Let $\mathcal{A}(H) = \{a : f_a(H) \geq 1\}$ be the accounts that occur in H .

► **Theorem 14** (Uniformity). *Denote by $f_i(H)$ the number of masking sets in which account a_i appears in the history of transactions H . Under the adversary model above, CUAT_λ satisfies strong untraceability on H if and only if $f_i(H) = f_j(H)$ for every two accounts a_i, a_j lying in a common masking set of H .*

Proof. Write $A = \mathcal{A}(H)$. An account occurs in a masking set as the sender, the receiver, or one of the $\lambda - 2$ uniformly sampled decoys. Under the independence assumption, the likelihood factorizes over accounts. Hence there is a function g , independent of the account identity, such that

$$\Pr [H \mid (q_a, r_a)_{a \in A}] \propto \prod_{a \in A} g(q_a, r_a, f_a(H)).$$

Thus $f_a(H)$ is sufficient for (q_a, r_a) . By symmetry of the prior, accounts with equal incidence are posterior-exchangeable. Accounts with different incidence are not: uniform decoy sampling contributes equally to all accounts, and $\mathbb{E}[q_a \mid H]$ is strictly increasing in $f_a(H)$. For sufficiency, suppose $f_a(H) = f_b(H)$ whenever a and b lie in a common masking set of H . Fix $\tau \in H$ and $a_i, a_j \in S_\tau$, and note $S_\tau \subseteq A$. Let π swap a_i and a_j and fix every other account. Since $f_{a_i}(H) = f_{a_j}(H)$ by hypothesis and no other count moves, π fixes S_τ and preserves the incidence vector, while mapping the event “ a_i sent τ ” to “ a_j sent τ ”. Posterior exchangeability gives the two events equal probability. Exactly one member of S_τ sent τ , so these $|S_\tau| = \lambda$ probabilities sum to 1 and each equals $1/\lambda$. Strong untraceability holds on H .

For necessity, suppose strong untraceability holds on H , and fix $\tau \in H$ with $a_i, a_j \in S_\tau$. Both have posterior probability $1/\lambda$ of having sent τ . Were $f_i(H) \neq f_j(H)$, strict monotonicity of $\mathbb{E}[q_a \mid H]$ in $f_a(H)$ would give the accounts different posterior sender probabilities for τ . Consequently, some $a^* \in S_\tau$ would satisfy $P(a^* \mid H) > 1/\lambda$. Hence $f_i(H) = f_j(H)$: incidence is constant on every masking set of H . ◀

The theorem gives a local condition, comparing accounts that occur in a common masking set. Connected masking-set families satisfy a global form of the same condition.

► **Corollary 15** (Uniform incidence). *Let H be a history whose masking sets are connected under intersection, meaning that their intersection graph is connected; this holds in particular when they pairwise intersect. Then CUAT_λ satisfies strong untraceability on H if and only if $f_i(H) = f_j(H)$ for all $a_i, a_j \in \mathcal{A}(H)$.*

Proof. If incidence is constant on $\mathcal{A}(H)$ it is constant on every masking set of H , and Theorem 14 gives strong untraceability. Conversely, let strong untraceability hold on H and let $a, b \in \mathcal{A}(H)$. The theorem makes incidence constant on each masking set of H . Each of a, b lies in some masking set, and by hypothesis there are masking sets $S^{(1)}, \dots, S^{(m)}$ of H with $a \in S^{(1)}$, $b \in S^{(m)}$ and $S^{(l)} \cap S^{(l+1)} \neq \emptyset$ for each $l < m$. Consecutive sets share an account at which their two constants agree, so one value runs along the whole chain and $f_a(H) = f_b(H)$. ◀

Connectedness is necessary. For example, two disjoint masking-set components may have different incidences while remaining uniform within each component. The consensus constructions below use connected families, so the corollary applies. We quantify only over $\mathcal{A}(H)$ because newly created accounts outside the history have incidence zero.

We next bound the clique number of a one-round protocol. Its masking sets are λ -uniform and pairwise intersect, and strong untraceability makes their account incidence uniform.

► **Theorem 16** (Clique bound). *If the masking-set family of a one-round protocol Π has uniform incidence, then $\omega(G) \leq \lambda^2 - \lambda + 1$.*

Proof. Let $F = \{S_1, \dots, S_n\}$ be a clique of G of size $\omega(G)$; by definition of G , the S_i pairwise intersect, and by hypothesis every account in $\bigcup_i S_i$ lies in a common number m of the invocations of F , each carrying λ accounts. A one-round protocol gives each process a single invocation, determined by its identity, so the n invocations of F belong to n distinct processes; each funds its transfer from an account of its own, and distinct processes own distinct accounts, so $n \leq |\bigcup_i S_i|$, and double counting $|\bigcup_i S_i| m = n\lambda$ gives $m \leq \lambda$. Now fix S_1 . Every other member of F meets it, and each of the λ accounts of S_1 lies in at most $m - 1 \leq \lambda - 1$ further members, so $n \leq 1 + \lambda(\lambda - 1)$. Hence $\omega(G) = n \leq \lambda^2 - \lambda + 1$; Füredi [19] and Ihringer and Kupavskii [26] characterize the families attaining this bound. ◀

Combining Lemma 10 with Theorem 16 gives the one-round consensus bound under strong untraceability.

► **Corollary 17** (One-round bound). *A wait-free one-round consensus protocol on CUAT_λ^+ can solve consensus among at most $\lambda^2 - \lambda + 1$ processes.*

Proof. Let Π be a d -consensus protocol of the class of Section 4.3.1, with $d \geq 3$. Lemma 10 gives $d \leq \omega(G)$: the d masking sets must lie on a common object and form a d -clique in G , hence pairwise intersect. Strong untraceability forces every account in the family to appear in a common number of masking sets, i.e., uniform incidence (Corollary 15). Theorem 16 then caps the clique number, $\omega(G) \leq \lambda^2 - \lambda + 1$. Combining, $d \leq \lambda^2 - \lambda + 1$. ◀

When $\lambda - 1$ is a prime power, the upper bound of Corollary 17 is matched by a projective-plane construction.

► **Theorem 18** (Projective plane). *If $\lambda - 1$ is a prime power, CUAT_λ^+ supports a wait-free one-round $(\lambda^2 - \lambda + 1)$ -consensus protocol, instantiated by attempt-and-adopt on the line set of $\text{PG}(2, \lambda - 1)$.*

Proof. Let $q = \lambda - 1$. The projective plane $\text{PG}(2, q)$ has $\lambda^2 - \lambda + 1$ points and the same number of lines; each line contains $\lambda = q + 1$ points, every two lines meet in exactly one point, and each point lies on exactly λ lines [26]. Interpreting points as accounts and lines as masking sets yields a λ -uniform pairwise-intersecting family of size $\lambda^2 - \lambda + 1$ over $\lambda^2 - \lambda + 1$ accounts.

To run the attempt-and-adopt protocol of Lemma 9 (Algorithm 1) on this family, each of the $\lambda^2 - \lambda + 1$ processes must be assigned a distinct masking set containing one of that process's accounts. This is a perfect-matching problem in the bipartite graph between accounts and masking sets, with an edge for each containment. Each account appears in exactly λ masking sets and each masking set contains exactly λ accounts, so the graph is λ -regular; König's theorem [34] yields a perfect matching. Lemma 9 then implements $(\lambda^2 - \lambda + 1)$ -consensus on this family. ◀

For general λ , a cyclic difference cover gives a one-round family of the size needed below.

► **Theorem 19** (Cyclic family). *For every $\lambda \geq 3$, CUAT_λ^+ supports a wait-free one-round $\lfloor \lambda^2/2 \rfloor$ -consensus protocol.*

Proof. Put $g = \lfloor \lambda/2 \rfloor$, $h = \lceil \lambda/2 \rceil$, and $n = \lfloor \lambda^2/2 \rfloor$, so $g + h = \lambda$ and $gh = n/2$. Identify the accounts and processes with $\mathbb{Z}_n$, let $\mu(t) = \{t\}$, and define

$$D = \{0, 1, \dots, g - 1\} \cup \{g, 2g, \dots, hg\}, \quad S_t = D + t$$

for $t \in \mathbb{Z}_n$. The two parts of D are disjoint, so $|S_t| = |D| = g + h = \lambda$. Moreover, $x \in D + t$ exactly when $t \in x - D$; hence every account lies in exactly λ translates, and the family has uniform incidence.

It remains to show that the translates pairwise intersect. We have $(D + t) \cap (D + t') \neq \emptyset$ exactly when $t' - t \in D - D$. The consecutive part of D contributes the differences $[-(g-1), g-1]$. For each $j \in \{1, \dots, h\}$, subtracting $\{0, \dots, g-1\}$ from hg contributes $[(j-1)g+1, jg]$. These intervals tile $[1, hg]$, and their negatives cover $[-hg, -1]$. Since $hg = n/2$, these differences cover all of $\mathbb{Z}_n$. Thus the S_t pairwise intersect.

Finally, the translates are distinct. If a nonzero subgroup H stabilized D , then, since $0 \in D$, we would have $H \subseteq D \subseteq [0, n/2]$. Its least positive generator d would satisfy $d \geq n/2$ because $n-d \in H$; the only remaining case $d = n/2$ would force D to be a union of pairs $\{x, x+n/2\}$, possible here only for $D = \{0, n/2\}$ and $\lambda = 2$, which is excluded. Therefore the stabilizer is trivial. Since $0 \in D$, process t owns $t \in S_t$, and Lemma 9 applied to the resulting n -clique gives the claim. $\blacktriangleleft$

Thus, for every $\lambda \geq 3$, the one-round consensus number lies between $\lfloor \lambda^2/2 \rfloor$ and $\lambda^2 - \lambda + 1$. The upper bound is tight when $\lambda - 1$ is a prime power. For general protocols, Lemma 8 requires intersection only across the two valency classes. Counting how many cross-class pairs each account can cover gives the following bound.

► **Theorem 20** (General upper bound). *For every $\lambda \geq 2$, a wait-free consensus protocol on CUAT_λ^+ can solve consensus among at most λ^2 processes.*

Proof. Suppose Π solves n -consensus; if $n \leq 2$ the bound holds trivially, since $\lambda^2 \geq 4$, so assume $n \geq 3$. Take C , $\mathcal{P}_0$, $\mathcal{P}_1$, and O as in Lemma 8, and write $s = |\mathcal{P}_0| + |\mathcal{P}_1|$. Uniformity first bounds the incidence of an account by λ : if d masking sets of size λ cover v accounts with common incidence m , then $vm = d\lambda$ by double counting, while $d \leq v$ because distinct processes own distinct accounts and each funds its transfer from an account of its own masking set; hence $m \leq \lambda$. For every account a on O , let i_a and j_a count its incidences among the pending masking sets of $\mathcal{P}_0$ and $\mathcal{P}_1$, respectively, and write $s_0 = |\mathcal{P}_0|$ and $s_1 = |\mathcal{P}_1|$, so $s = s_0 + s_1$. Every cross-class pair must share an account, so $s_0 s_1 \leq \sum_a i_a j_a$. Uniformity makes $i_a + j_a = m$ for every account occurring at C , and each of the s pending masking sets has λ members, so $\sum_a i_a = s_0 \lambda$, $\sum_a j_a = s_1 \lambda$, and the number of such accounts is $v = s\lambda/m$. Cauchy-Schwarz gives $\sum_a i_a^2 \geq (\sum_a i_a)^2/v$, whence

$$\sum_a i_a j_a = \sum_a i_a (m - i_a) = m s_0 \lambda - \sum_a i_a^2 \leq m s_0 \lambda - \frac{(s_0 \lambda)^2}{v} = s_0 \lambda m \frac{s_1}{s}.$$

Combining with $s_0 s_1 \leq \sum_a i_a j_a$ and dividing by $s_0 s_1 > 0$ gives $s \leq \lambda m \leq \lambda^2$. $\blacktriangleleft$

Equality requires $m = \lambda$, a constant split $i_a = c$ and $j_a = \lambda - c$ of every account between the classes, hence $s_0 = \lambda c$ and $s_1 = \lambda(\lambda - c)$, and every cross-pair covered exactly once. The following recursive construction realizes these conditions.

► **Lemma 21** (Coincident sets). *For every $\lambda \geq 2$, CUAT_λ^+ supports a wait-free one-round λ -consensus protocol in which every process carries the same masking set.*

Proof. Take an object with $\mathcal{A}_0 = T = \{a_1, \dots, a_\lambda\}$, $\mu(a_i) = \{p_i\}$ and $\beta_0 \equiv 0$, and let every p_i carry the masking set T and the value $v = 0$. Each p_i owns $a_i \in T$, and the λ invocations pairwise conflict, so they form a λ -clique of G and Lemma 9 applies: the first τ to linearize commits and replaces T by its new accounts, every later τ finds $T \not\subseteq \mathcal{A}$ and fails, and a loser identifies the committed transfer by the new accounts it created and adopts its issuer's proposal. All λ invocations carry T , so every account of T has incidence λ and Theorem 14 gives strong untraceability. $\blacktriangleleft$

A grid construction composes two such protocols while adding their sizes.

► **Theorem 22** (Grid composition). *For every $\lambda \geq 2$ and every $m \in \{1, \dots, \lambda\}$, CUAT_λ^+ supports a wait-free λm -consensus protocol. In particular $\text{cons}(\text{CUAT}_\lambda^+) \geq \lambda^2$.*

Proof. Induction on m ; the case $m = 1$ is Lemma 21. For $m \geq 2$ write $m = u + w$ with $u, w \geq 1$ and split the λm processes into classes $\mathcal{P}_0$ of size λu and $\mathcal{P}_1$ of size λw .

Round 1. Each class runs the protocol supplied by the induction hypothesis, on its own objects. Every process of class v thereby holds a common *class value* w_v , which is the input of some process of that class.

Round 2. Partition $\mathcal{P}_0$ into λ blocks $X_1, \dots, X_\lambda$ of size u and $\mathcal{P}_1$ into λ blocks $Y_1, \dots, Y_\lambda$ of size w . On a new object with accounts $\{a_{ij}\}_{i,j \in [\lambda]}$, let every $p \in X_i$ carry the row $\{a_{ij} : j \in [\lambda]\}$ and every $q \in Y_j$ the column $\{a_{ij} : i \in [\lambda]\}$. Each masking set has λ accounts, and a_{ij} occurs in the invocations of the u processes of X_i and the w processes of Y_j , so incidence is $u + w = m$ on every account and Theorem 14 admits the round. Each process writes its class value to a single-writer register and invokes τ .

Row i and column j meet exactly in a_{ij} , so once some transfer commits, every transfer of the other class finds one of its accounts invalidated and fails; the committing transfers therefore all belong to one class, and at least one commits. A process that commits decides its own class value; one that fails identifies a committed transfer in the published state, reads off the class of its issuer, and adopts that class's value. Every process thus decides the winning class's value, which round 1 made common to that class and which is some process's input, so agreement and validity hold, and neither round loops.

Finally, the bipartite graph between the λm processes and the λ^2 accounts of the round-2 object has degree λ on the left and $m \leq \lambda$ on the right, so $|N(X)| \geq \lambda|X|/m \geq |X|$ for every set X of processes and Hall's theorem assigns each process a distinct account of its own masking set to fund from. Taking $m = \lambda$ gives λ^2 processes. ◀

The families of Lemma 21 and Theorem 22 satisfy more than uniform incidence: any two accounts of a masking set are exchanged by a symmetry of the whole family, a transposition of two grid columns or of two grid rows. Their sender posteriors are therefore uniform for every adversary whose prior is symmetric in the accounts, independently of the independent-inclusion idealization.

► **Theorem 23** (Consensus number of CUAT_λ^+). *For every $\lambda \geq 2$, $\text{cons}(\text{CUAT}_\lambda^+) = \lambda^2$.*

Proof. The upper bound is Theorem 20 and the lower bound is Theorem 22. ◀

For every $\lambda \geq 2$, $\lambda^2 > \lambda^2 - \lambda + 1$; hence multi-round protocols are strictly more powerful than one-round protocols under strong untraceability. The prime-power condition remains relevant only to attaining the one-round upper bound through a projective plane, the multi-round value carrying no arithmetic condition on λ .

Having characterized CUAT's consensus power through the conflict graph structure, we now examine how this structure affects parallelization opportunities and fairness guarantees.

5 Parallelization and Fairness

The consensus analysis uses the conflict graph at a critical configuration. For a fixed workload, the same graph determines sequential depth and parallel throughput. It also exposes a fairness limitation: an unfair scheduler can indefinitely prevent a conflicting transfer from succeeding.

5.1 Workload Parallelism

Fix a transfer workload $\mathcal{T}$. Its conflict graph $G_{\mathcal{T}} = (V, E)$ has one vertex per transfer, including distinct transfers with the same masking set, and an edge between transfers whose masking sets share an index. Adjacent transfers cannot both succeed in one round; non-adjacent transfers commute. Between rounds, a pending transfer may be *rebased* by replacing each account with the current representation of the same index.

► **Theorem 24** (Sequential depth). *Suppose there is no external interference and every transfer remains balance-eligible after rebasing. Then $\chi(G_{\mathcal{T}})$ is the minimum number of rounds in which all transfers in $\mathcal{T}$ can succeed.*

Proof. A proper k -coloring partitions V into independent sets $V_1, \dots, V_k$. Rebase and execute V_i in round i . Its transfers use disjoint indices and therefore all succeed. Conversely, color each transfer by its round in any successful k -round schedule. Transfers in one round cannot share an index, so this is a proper k -coloring. Hence $\chi(G_{\mathcal{T}}) \leq k$. ◀

The independence number gives the corresponding one-round throughput.

► **Theorem 25** (Parallel throughput). *The maximum number of transfers in $\mathcal{T}$ that can succeed in one round is $\alpha(G_{\mathcal{T}})$.*

Proof. The transfers in an independent set use disjoint indices, commute, and all succeed in one round. Thus $\alpha(G_{\mathcal{T}})$ is achievable. Conversely, transfers that all succeed in one round are pairwise non-conflicting and therefore form an independent set. ◀

5.2 Starvation

Wait-freedom guarantees that an invocation terminates, not that it succeeds. An unfair asynchronous scheduler may repeatedly resolve conflicts against the same process. This formalizes a denial-of-service behavior conjectured for *Quisquis* by Bünz *et al.* [9].

► **Definition 26** (Starvation-freedom). *An implementation of CUAT_{λ} is starvation-free if, for every correct process p , every infinite execution in which p invokes infinitely many eligible transfers contains infinitely many successful invocations by p . Eligibility is evaluated at invocation: the sender account is owned by p , its balance covers the chosen value, and the masking set is valid. Successive invocations may follow the current representations of the same logical accounts through rerandomization.*

This definition requires neither bounded waiting nor proportional success, only infinitely many successes for a process that remains active and eligible.

► **Theorem 27** (Starvation). *No CUAT_{λ} implementation is starvation-free under an asynchronous, unfair scheduler. A correct process may fail infinitely often while another correct process succeeds infinitely often on the same masking-set lineage. This holds even for zero-value transfers.*

Proof. Fix a masking set S containing accounts a and a^* , owned by processes p and p^* , respectively. Let $b, b^* \in S$ be arbitrary destinations. Both processes repeatedly invoke transfers on the current representations of these accounts. We construct an infinite execution in which p^* wins every conflict.

Round r . Process p invokes $\tau_r = \text{transfer}(S'_p, \pi_p)$ with $\pi_p = (S, a, b, v, p)$ for some $v \leq \beta(a)$. The transfer is eligible when invoked. The scheduler pauses p before its linearization point and lets p^* complete its zero-value transfer $\tau_r^* = \text{transfer}(S'_*, \pi^*)$ with witness $\pi^* = (S, a^*, b^*, 0, p^*)$. This invocation succeeds: S is valid and $\beta(a^*) \geq 0$ at its linearization point. It replaces S by S'_* and succeeds.

The scheduler then resumes p . Its old masking set is invalid, so τ_r returns **FALSE**.

Both processes rebase before the next round. Because p^* transfers value zero, its balance is unchanged and it remains eligible. Repeating the schedule makes every invocation of p fail while p^* succeeds once per round. ◀

5.3 Implications and Mitigations

The graph parameters give an operational interpretation of contention. $\chi(G_{\mathcal{T}})$ is the minimum latency, in sequential rounds, for a fixed eligible workload, whereas $\alpha(G_{\mathcal{T}})$ is its maximum one-round throughput. These quantities depend on the placement of masking sets, not only on their size. Two workloads with the same number of transfers and the same anonymity parameter may therefore have very different execution costs. They also concern only the conflicting portion of the workload: isolated vertices and independent components can be processed without a common order.

This observation exposes two denial-of-service surfaces. First, a sender that can influence its masking set can introduce edges incident to selected honest transfers, increasing sequential depth and reducing parallel throughput. Second, control over scheduling can repeatedly resolve an existing edge against the same transfer. Zero-value transfers make the second attack particularly cheap because the adversary can rerandomize the masking-set lineage without depleting its balance. The starvation theorem requires neither an invalid proof nor insufficient funds; every adversarial transfer is valid when it commits. These results apply when a transfer consumes its entire masking set. They do not apply when a transfer only references its anonymity set. Monero rings and Zcash note-commitment sets retain their decoys; a repeated key image or nullifier indicates a double spend rather than masking-set-wide contention. Anonymous Zether [9, 14] similarly avoids masking-set-wide invalidation by deriving the relevant nonce from the sender's secret key. These mechanisms are closer to LUAT, analyzed in the full version [10]. Sui [8] and Zef [5] similarly process disjoint public transfers independently and coordinate only contending operations.

Several complementary mechanisms can limit these attacks. First, the protocol can derive masking sets from public, unpredictable randomness rather than let senders choose them freely. A verifiable random function or randomness beacon can make the selection auditable while preventing an adversary from targeting a particular transfer [30]; the design must also limit grinding over candidate sets. This controls how adversarially the conflict graph can be formed, but does not resolve conflicts that occur naturally.

Second, admission control can raise the cost of repeated contention. Fees, rate limits, per-account quotas, or restrictions on zero-value transfers make an infinite sequence of interfering transfers expensive or impossible within a fixed interval. Such measures bound the rate of an attack rather than eliminate the underlying schedule, and they may also penalize honest users who retry after a conflict.

Third, a fair broadcast or ordering layer can ensure that a transfer which remains pending is eventually placed before later conflicting transfers. This directly addresses starvation, but changes the assumptions of Theorem 27: the scheduler is no longer unfair, and the system may require synchrony, randomized leader election, or Byzantine agreement. Order-fair Byzantine consensus constrains adversarial transaction reordering [27, 11], while causal or

atomic group broadcast preserves dependencies between related messages [7, 4]. Causal order alone does not prevent starvation between concurrent transfers, so an inclusion or order-fairness condition is still needed. Randomized contention management offers a lighter alternative under a weaker progress condition [24], with probabilistic rather than deterministic success guarantees.

Finally, validators can use the conflict graph constructively. Independent sets can be batched and verified in parallel, while a coloring gives a schedule for rebasing and executing a contended batch. Overall, these measures call for a holistic design that jointly controls masking-set selection, admission, ordering, and execution.

6 Conclusion

We introduced λ -CUAT to study the synchronization cost of constant-state sender untraceability. Under weak untraceability, CUAT has unbounded consensus number, even for one-round protocols. Under strong untraceability, accounts in a common masking set must have uniform incidence. For one-round protocols, this condition bounds the consensus number by $\lambda^2 - \lambda + 1$; projective planes attain this bound whenever $\lambda - 1$ is a prime power, and cyclic constructions give the general lower bound $\lfloor \lambda^2/2 \rfloor$. For general protocols, our recursive grid construction matches the upper bound for every $\lambda \geq 2$, yielding $\text{cons}(\text{CUAT}_\lambda^+) = \lambda^2$.

The conflict graph connects these worst-case bounds to concrete workloads: its coloring and independent-set structure determine sequential depth and one-round throughput. It also exposes a progress cost, since CUAT cannot guarantee starvation-freedom under an unfair asynchronous scheduler. These results answer the question posed in the introduction. Consuming a masking set to bound ledger state turns a privacy mechanism into mutable shared state and thereby couples sender untraceability to synchronization and fairness. The consensus numbers measure the agreement power available in the worst case, not a coordination requirement for every transfer; transfers with disjoint masking sets commute and can be processed independently.

The full version [10] analyzes the complementary λ -LUAT abstraction, which captures the classical growing-state nullifier approach used by systems such as Monero [1] and Zcash [6]. It proves that LUAT has consensus number 2 for every masking-set size and under both untraceability notions, and that LUAT is starvation-free. It also studies partitioning objects and decoy-pruning mechanisms for reclaiming the state accumulated by LUAT. The comparison isolates the design trade-off: LUAT pays for untraceability through growing state, whereas CUAT keeps state constant at the cost of stronger synchronization and weaker fairness.

A central direction for future work is an end-to-end constant-state ledger that addresses privacy, progress, and efficiency jointly. Such a design should derive masking sets from unpredictable, publicly verifiable randomness while limiting grinding; guarantee fair inclusion of conflicting transfers under explicit scheduling and fault assumptions; and control repeated contention without creating a new denial-of-service vector. Compact proofs, aggregation, and conflict-aware batching may also be needed to keep communication practical as masking sets grow. Combining these mechanisms would clarify whether a ledger can bound its state without sacrificing fairness or practical throughput.

References

- 1 Kurt M. Alonso and The Monero Community. *Zero to Monero: Second Edition — Second Edition: First Principles and Cryptonote Cryptography*. The Monero Project, 2020. Version 2.0.0, April 2020. URL: <https://web.getmonero.org/library/Zero-to-Monero-2-0-0.pdf>.
- 2 Orestis Alpos, Christian Cachin, Giorgia Azzurra Marson, and Luca Zanolini. On the synchronization power of token smart contracts. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, pages 640–651. IEEE, 2021. doi:10.1109/ICDCS51616.2021.00067.
- 3 Jayamine Alupotha, Mathieu Gestin, and Christian Cachin. Nopenena untraceable payments: Defeating graph analysis with small decoy sets. *IACR Cryptol. ePrint Arch.*, page 903, 2024. URL: <https://eprint.iacr.org/2024/903>.
- 4 Mariarosaria Barbaraci and Christian Cachin. Byzantine reliable broadcast with causal ordering: (research statement). In Dennis Olivetti and Eric Ruppert, editors, *APPLIED*, pages 13:1–13:5. ACM, 2026. doi:10.1145/3820355.3820392.
- 5 Mathieu Baudet, Alberto Sonnino, Mahimna Kelkar, and George Danezis. Zef: Low-latency, scalable, private payments. arXiv:2201.05671, 2022. arXiv:2201.05671.
- 6 Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014. doi:10.1109/SP.2014.36.
- 7 Kenneth P. Birman, André Schiper, and Pat Stephenson. Lightweight causal and atomic group multicast. *ACM Trans. Comput. Syst.*, 9(3):272–314, 1991. doi:10.1145/128738.128742.
- 8 Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, and Lu Zhang. Sui lustris: A blockchain combining broadcast and consensus. arXiv:2310.18042, 2023. doi:10.48550/arXiv.2310.18042.
- 9 Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 423–443. Springer, 2020. doi:10.1007/978-3-030-51280-4_23.
- 10 Christian Cachin, David Lehnherr, Juan Villacis, and François-Xavier Wicht. The consensus number of untraceable cryptocurrencies. *Cryptology ePrint Archive*, Paper 2026/1503, 2026. URL: <https://eprint.iacr.org/2026/1503>.
- 11 Christian Cachin, Jovana Micic, Nathalie Steinhauer, and Luca Zanolini. Quick order fairness. In *Financial Cryptography*, pages 316–333, 2022. doi:10.1007/978-3-031-18283-9_15.
- 12 Miranda Christ and Joseph Bonneau. Limits on revocable proof systems, with implications for stateless blockchains. In Foteini Baldimtsi and Christian Cachin, editors, *Financial Cryptography and Data Security - 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1-5, 2023, Revised Selected Papers, Part II*, volume 13951 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2023. doi:10.1007/978-3-031-47751-5_4.
- 13 Eli Daian, Giuliano Losa, Yehuda Afek, and Eli Gafni. A wealth of sub-consensus deterministic objects. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.DISC.2018.17.
- 14 Benjamin E. Diamond. Many-out-of-many proofs and applications to anonymous zether. In *SP*, pages 1800–1817, 2021. doi:10.1109/SP40001.2021.00026.
- 15 Christoph Egger, Russell W. F. Lai, Viktoria Ronge, Ivy K. Y. Woo, and Hoover H. F. Yin. On defeating graph analysis of anonymous transactions. *Proc. Priv. Enhancing Technol.*, 2022(3):538–557, 2022. doi:10.56553/POPETS-2022-0085.

- 16 Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shihō Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 649–678. Springer, 2019. doi:10.1007/978-3-030-34578-5_23.
- 17 Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985. doi:10.1145/3149.214121.
- 18 Davide Frey, Mathieu Gustin, and Michel Raynal. The synchronization power (consensus number) of access-control objects: the case of allowlist and denylist. In Rotem Oshman, editor, *37th International Symposium on Distributed Computing, DISC 2023, October 10-12, 2023, L'Aquila, Italy*, volume 281 of *LIPICs*, pages 21:1–21:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.DISC.2023.21.
- 19 Zoltán Füredi. Maximum degree and fractional matchings in uniform hypergraphs. *Combinatorica*, 1(2):155–162, 1981. doi:10.1007/BF02579271.
- 20 David K. Gifford. Weighted voting for replicated data. In Michael D. Schroeder and Anita K. Jones, editors, *Proceedings of the Seventh Symposium on Operating System Principles, SOSOP 1979, Asilomar Conference Grounds, Pacific Grove, California, USA, 10-12, December 1979*, pages 150–162. ACM, 1979. doi:10.1145/800215.806583.
- 21 Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. The consensus number of a cryptocurrency. *Distributed Comput.*, 35(1):1–15, 2022. doi:10.1007/S00446-021-00399-2.
- 22 Timothy L. Harris, Keir Fraser, and Ian A. Pratt. A practical multi-word compare-and-swap operation. In Dahlia Malkhi, editor, *Distributed Computing, 16th International Conference, DISC 2002, Toulouse, France, October 28-30, 2002 Proceedings*, volume 2508 of *Lecture Notes in Computer Science*, pages 265–279. Springer, 2002. doi:10.1007/3-540-36108-1_18.
- 23 Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, 1991. doi:10.1145/114005.102808.
- 24 Maurice Herlihy, Victor Luchangco, and Mark Moir. Obstruction-free synchronization: Double-ended queues as an example. In *23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, pages 522–529. IEEE Computer Society, 2003. doi:10.1109/ICDCS.2003.1203503.
- 25 Maurice Herlihy and Jeannette M. Wing. Axioms for concurrent objects. In *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages, Munich, Germany, January 21-23, 1987*, pages 13–26. ACM Press, 1987. doi:10.1145/41625.41627.
- 26 Ferdinand Ihringer and Andrey Kupavskii. Regular intersecting families. *Discrete Applied Mathematics*, 270:142–152, 2019. doi:10.1016/j.dam.2019.07.009.
- 27 Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In *CRYPTO (3)*, pages 451–480, 2020. doi:10.1007/978-3-030-56877-1_16.
- 28 Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero’s blockchain. In *ESORICS (2)*, pages 153–173, 2017. doi:10.1007/978-3-319-66399-9_9.
- 29 Varun Madathil and Alessandra Scafuro. sfpri tt fhsefte: Achieving full-privacy in account-based cryptocurrencies is possible. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part VII*, volume 16251 of *Lecture Notes in Computer Science*, pages 199–229. Springer, 2025. doi:10.1007/978-981-95-5122-4_7.
- 30 Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999. doi:10.1109/SFFCS.1999.814584.

- 31 Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 397–411. IEEE Computer Society, 2013. doi:10.1109/SP.2013.34.
- 32 Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *Proc. Priv. Enhancing Technol.*, 2018(3):143–163, 2018. doi:10.1515/POPETs-2018-0025.
- 33 Robert H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4(2):180–209, 1979. doi:10.1145/320071.320076.
- 34 Douglas B West. *Introduction to graph theory*. Prentice Hall, Upper Saddle River, NJ, 2 edition, August 2000.