Policy-CRDT: Conflict-Free Replicated Data Type with Remove-Wins Strategy for Convergent Access Control in Asynchronous Environments

Mahamdou Sidibe *

Posted Date: 18 December 2025

doi: 10.20944/preprints202512.1467.v1

Keywords: conflict-free replicated data types; CRDT; access control; remove-wins; strong eventual consistency; zero trust; multi-cloud systems

 doi:10.20944/preprints202512.1467.v1

Article

Policy-CRDT: Conflict-Free Replicated Data Type with Remove-Wins Strategy for Convergent Access Control in Asynchronous Environments

Mahamadou Sidibe

Faculty of Secure Information Technologies, ITMO University, Saint Petersburg, Russia; sidibemahamadou@windowslive.com

Abstract

Modern multi-cloud and edge-cloud systems replicate both data and access control policies across geographically distributed nodes under weak consistency models. In asynchronous environments with possible network partitions, policy updates (additions and revocations of rules, delegation and revocation of privileges) may occur concurrently, causing conflicts and potential privilege escalation when naïve conflict resolution schemes such as last-writer-wins (LWW) or add-wins are used. This paper proposes a formal model of *Policy-CRDT*, a conflict-free replicated data type (CRDT) for sets of access control policies with a *remove-wins* strategy, based on the two-phase set (2P-Set) and a join-semilattice structure on replica states. At the CRDT abstraction level, each replica state is represented by a pair of monotonically growing sets of added and revoked policy identifiers, and state merging is defined as a commutative, associative, and idempotent union operator. We show that the proposed data type satisfies the standard Strong Eventual Consistency (SEC) conditions for state-based CRDTs: replica states form a join-semilattice, local updates are monotone, and the merge function computes least upper bounds, which ensures convergence of replicas once they have received the same set of updates. We formally prove that the remove-wins strategy guarantees inevitable suppression of any policy for which at least one revocation exists in the global history, in contrast to LWW and add-wins schemes that can admit "dangerous" states with excessive permissions. We further propose an architecture for deploying Policy-CRDT in a distributed PDP/PEP infrastructure in the spirit of Zero Trust and NIST SP 800-207/800-207A, and we present an analytical evaluation of convergence latency and the probability of potentially dangerous states compared to alternative strategies. The results demonstrate that Policy-CRDT provides formally grounded convergence of access control policies at reasonable overhead and is semantically safe in multi-cloud and edge deployment scenarios.

Keywords: conflict-free replicated data types; CRDT; access control; remove-wins; strong eventual consistency; zero trust; multi-cloud systems

1. Introduction

Distributed applications deployed across multiple cloud providers and edge locations increasingly rely on replication of both data and access control policies to reduce latency and improve availability and resilience [3,6]. In such environments, consistency models are often weakened to eventual consistency, and access control policies (RBAC/ABAC, XACML) are replicated across multiple policy decision points (PDPs) and policy enforcement points (PEPs) spread over regions and administrative domains [11,13]. In the presence of asynchronous communication and potential network partitions, policy updates—additions of allowing and denying rules, delegation and revocation of privileges—can occur concurrently, leading to temporary divergence and conflicts.

Conflict-free replicated data types (CRDTs) provide a principled way to achieve Strong Eventual Consistency (SEC) without coordination by ensuring that replicas converge to a unique state when they have observed the same set of updates [1,5]. For sets of elements, several CRDT models have

 doi:10.20944/preprints202512.1467.v1

been proposed, including grow-only sets, two-phase sets (2P-Sets), observed-remove sets (OR-Sets), and variants with add-wins or remove-wins semantics [3,8]. However, specialising these models to replication of access control policies in security-critical Zero Trust environments [14,15] requires careful design of conflict resolution.

In the context of access control, an error in the direction of granting extra permissions is much more severe than a temporary error that denies legitimate access. Naïve schemes such as LWW or add-wins, when applied to policy identifiers under weak consistency, can lead to states where revoked policies remain effectively active because their last update or add operation dominates a concurrent revocation in the chosen conflict resolution semantics [2,10]. This is fundamentally at odds with the principles of Zero Trust and the requirements of security standards and regulations.

Goal. The goal of this paper is to define and formally justify a CRDT for access control policies, called *Policy-CRDT*, that:

- provides Strong Eventual Consistency (SEC) for replicated policy sets;
- enforces a safety monotonicity property: the presence of a revocation cannot reduce the effective strictness of the resulting policy set compared to any scenario without this revocation;
- is compatible with existing ABAC/XACML policy languages and Zero Trust architectures in multi-cloud and microservice environments [11,13,15].

Contributions. The main contributions of this paper are:

N1 A formal definition of *Policy-CRDT* as a state-based CRDT (CvRDT) for sets of policy identifiers with remove-wins semantics, structured as a two-phase set (2P-Set).

N2 A rigorous specification of the partial order on states, merge operation, and update operations `addPolicy`/`removePolicy`, satisfying the standard CRDT criteria and SEC conditions [1,5].

N3 Proof that the merge operation is commutative, associative, idempotent and monotone, and that all local updates are monotone with respect to the state order, ensuring conflict-freedom.

N4 A convergence theorem stating that Policy-CRDT satisfies Strong Eventual Consistency under eventual delivery.

N5 A safety analysis of the remove-wins strategy: we prove that any policy for which at least one revocation occurs in the global history is inevitably deactivated on all replicas, and we compare this with LWW and add-wins using an analytic model of "dangerous" states (i.e., states with excessive permissions).

N6 An architecture for deploying Policy-CRDT in a distributed PDP/PEP infrastructure aligned with Zero Trust and NIST SP 800-207/800-207A, including global and local control planes and integration with service mesh [15].

N7 An analytic evaluation of convergence latency and the probability of dangerous states based on classic results on randomized rumor spreading in fully connected networks [16,17], with plots and tables computed from explicit formulas, ensuring full reproducibility.

The remainder of the paper is structured as follows. Section 2 reviews CRDT fundamentals and related work on distributed access control and policy replication. Section 3 introduces the formal model of access policies and Policy-CRDT. Section 4 presents the convergence and SEC theorem and the safety analysis of remove-wins. Section 5 describes a deployment architecture and pseudocode. Section 6 provides the analytical evaluation. Section 7 discusses limitations and future work. Section 8 concludes.

2. Preliminaries and Related Work

2.1. CRDT Model and Strong Eventual Consistency

Conflict-free replicated data types (CRDTs) were introduced in [1,2] as data structures replicated at multiple nodes such that:

(i) each replica can be updated locally without coordination;

(ii) state merge resolves conflicts automatically;

 doi:10.20944/preprints202512.1467.v1

(iii) replicas that have seen the same set of updates (possibly in different orders) converge to the same state.

For state-based CRDTs (also known as convergent replicated data types, CvRDTs) the state of each replica belongs to a join-semilattice $(L, \sqsubseteq, \sqcup)$, where $\sqcup$ is commutative, associative, idempotent and computes least upper bounds, and local updates are monotone with respect to $\sqsubseteq$ [1,3,5]. Replicas periodically exchange states and merge them using $\mathsf{merge}(x, y) = x \sqcup y$.

Definition 1 (Strong Eventual Consistency [1,5])**.** *A replicated object satisfies Strong Eventual Consistency (SEC) if:*

(a) Convergence: *any two replicas that have observed the same set of updates (in possibly different orders) are in equivalent states.*
(b) Eventual delivery: *every update is eventually delivered to all replicas (subject to the replication policy).*
(c) Termination: *every local operation eventually completes.*

It is shown in [1,5] that a state-based CRDT whose states form a join-semilattice, whose merge function is $\sqcup$, and whose local updates are monotone satisfies SEC under eventual delivery.

2.2. Set CRDTs: G-Set, 2P-Set, OR-Set

The simplest CRDT for sets is the grow-only set (G-Set), which allows only add operations; its state is a set S and merge is set union $S_1 \cup S_2$ [3,7]. To support removals, the two-phase set (2P-Set) represents the state as a pair of sets (A, R), where A contains elements that have been added and R contains elements that have been removed (tombstones). Membership is defined as $x \in A \setminus R$ [7]. The 2P-Set exhibits remove-wins semantics: in the presence of concurrent add and remove of the same element, the element eventually becomes absent; however, once removed, an element cannot be re-added with the same identity.

More sophisticated structures, such as observed-remove sets (OR-Sets) and their optimizations [8], use unique tags for each insertion, allowing elements to be logically re-added while tracking causal relationships, at the cost of more complex metadata.

2.3. Distributed Access Control And Policy Replication

Attribute-Based Access Control (ABAC) [11,12] defines attributes of subjects, objects, actions and context as primary building blocks of policies, without fixing specific combining algorithms. XACML 3.0 [13] provides an XML-based policy language and a reference architecture with PDPs and PEPs, together with combining algorithms such as *deny-overrides*, *permit-overrides* and others. However, these combining algorithms are defined for flat sets of policies and do not address weakly consistent replication across multiple regions.

The problem of maintaining replicated authorizations and policies has been studied both in the context of distributed databases [9] and weakly consistent data stores. Samarati et al. [9] analyse replicated authorizations with optimistic replication and highlight difficulties with revocations. Wobber et al. [10] propose policy-based access control for weakly consistent replication, where policies are treated as data that co-travels with collection contents, and show that careless handling of revocations can lead to persistent divergence.

Zero Trust Architecture (ZTA) [14] and its cloud-native extension [15] emphasise continuous verification, least privilege and uniform policy enforcement across locations. In these architectures, a global control plane provides high-level policies, while local enforcement is performed by PEPs (often implemented as service-mesh sidecars) and PDPs within each cluster. A key challenge is to replicate policies in a way that prevents privilege escalation in the presence of concurrent updates and network partitions.

In this work, we specialise the 2P-Set remove-wins semantics to the domain of access control policies and show how to obtain a Policy-CRDT that provides strong guarantees on revocations and convergence.

 doi:10.20944/preprints202512.1467.v1

3. Formal Model of Policy-CRDT

3.1. Access Policy Model

Let Q be the set of access requests. Each $q \in Q$ is described by attributes of the subject, object, action and context, in the spirit of ABAC [11,12]. Let $\mathsf{Eff} = \{\mathsf{permit}, \mathsf{deny}\}$ be the set of effects.

Definition 2 (Access policy)**.** *An access policy is a triple*

$$p = (\mathrm{id}(p), \varphi_p, \mathrm{eff}(p)),$$

where $\mathrm{id}(p)$ *is a globally unique policy identifier,* $\varphi_p : Q \to \{\mathsf{true}, \mathsf{false}\}$ *is a predicate describing applicability of the policy to a request, and* $\mathrm{eff}(p) \in \mathsf{Eff}$ *is the effect of the policy.*

Let $\mathcal{P}$ be the set of all possible policies, and let $\mathcal{I}$ be the set of identifiers, with $\mathrm{id} : \mathcal{P} \to \mathcal{I}$. We assume that any modification to the content of a policy uses a new identifier, which matches common practice in industrial PDPs and supports auditability [10].

3.2. Replica State and Partial Order

Let R be a finite set of replicas, each storing the replicated state of active policy identifiers. At the CRDT level, we only track identifiers; the full policy bodies are stored in a separate repository indexed by identifiers.

Definition 3 (Policy-CRDT state)**.** *The state of a replica is a pair*

$$S = (A, T),$$

where $A \subseteq \mathcal{I}$ *is the set of policy identifiers that have ever been added, and* $T \subseteq \mathcal{I}$ *is the set of identifiers that have been revoked (tombstones).*

The set of *active* policies at state S is

$$\mathrm{Active}(S) = \{\, i \in A \mid i \notin T \,\}.$$

Definition 4 (Partial order on states)**.** *On the set of states* $\Sigma = \{(A, T) \mid A, T \subseteq \mathcal{I}\}$ *we define a partial order by*

$$(A_1, T_1) \sqsubseteq (A_2, T_2) \quad \Longleftrightarrow \quad A_1 \subseteq A_2 \ \wedge \ T_1 \subseteq T_2.$$

Intuitively, a larger state contains more information about additions and revocations.

Definition 5 (Merge operation)**.** *The merge of two states* $S_1 = (A_1, T_1)$ *and* $S_2 = (A_2, T_2)$ *is*

$$S_1 \sqcup S_2 = (A_1 \cup A_2,\ T_1 \cup T_2).$$

Lemma 1. *The structure* $(\Sigma, \sqsubseteq, \sqcup)$ *is a join-semilattice:* $\sqcup$ *is commutative, associative and idempotent, and* $S_1 \sqcup S_2$ *is the least upper bound of* S_1 *and* S_2 *with respect to* $\sqsubseteq$*.*

Proof. Commutativity and associativity follow from the corresponding properties of set union: $A_1 \cup A_2 = A_2 \cup A_1$, $(A_1 \cup A_2) \cup A_3 = A_1 \cup (A_2 \cup A_3)$, and similarly for T_1, T_2, T_3. Idempotence holds since $A \cup A = A$, $T \cup T = T$.

Let $S_3 = (A_3, T_3)$ with $S_1 \sqsubseteq S_3$ and $S_2 \sqsubseteq S_3$. Then $A_1 \subseteq A_3$ and $A_2 \subseteq A_3$, hence $A_1 \cup A_2 \subseteq A_3$; likewise $T_1 \cup T_2 \subseteq T_3$. Thus $S_1 \sqcup S_2 \sqsubseteq S_3$. On the other hand, $S_1 \sqsubseteq S_1 \sqcup S_2$ and $S_2 \sqsubseteq S_1 \sqcup S_2$ by construction, so $S_1 \sqcup S_2$ is a least upper bound. □

 doi:10.20944/preprints202512.1467.v1

3.3. Operations and Remove-Wins Semantics

Definition 6 (Policy-CRDT operations). *For a replica $r \in R$ with local state $S_r = (A_r, T_r)$ we define:*

- $\texttt{addPolicy}(i)$ *for* $i \in \mathcal{I}$*:*
$$S_r := (A_r \cup \{i\},\ T_r).$$
- $\texttt{removePolicy}(i)$*:*
$$S_r := (A_r,\ T_r \cup \{i\}).$$
- $\texttt{merge}(S_s)$ *for a remote state* $S_s = (A_s, T_s)$*:*
$$S_r := S_r \sqcup S_s.$$

Proposition 1 (Monotonicity of local updates). *For any state S and operation* $\mathrm{op} \in \{\texttt{addPolicy}(i), \texttt{removePolicy}(i)\}$ *we have* $S \sqsubseteq \mathrm{op}(S)$.

Proof. $\texttt{addPolicy}(i)$ adds i to A and leaves T unchanged, so $A \subseteq A \cup \{i\}$ and $T \subseteq T$. $\texttt{removePolicy}(i)$ adds i to T and leaves A unchanged, so $A \subseteq A$, $T \subseteq T \cup \{i\}$. □

Definition 7 (Remove-wins semantics). *Let $\mathcal{H}(i)$ be the set of all* $\texttt{addPolicy}(i)$ *and* $\texttt{removePolicy}(i)$ *operations in the global history. In the limit state $S_\infty = (A_\infty, T_\infty)$ after all updates have been delivered, the remove-wins semantics for identifier i is:*

$$i \in \mathrm{Active}(S_\infty) \iff i \in A_\infty \ \wedge\ i \notin T_\infty.$$

In particular, if $\mathcal{H}(i)$ contains at least one $\texttt{removePolicy}(i)$*, then $i \notin \mathrm{Active}(S_\infty)$.*

This is exactly the semantics of a 2P-Set [3,7] and is well suited to access control, where revocation is expected to dominate any prior additions with the same identifier.

4. Convergence and Strong Eventual Consistency

Lemma 2. *The merge operation $S_r := S_r \sqcup S_s$ is commutative, associative and idempotent. Repeated merge with a finite family of states is invariant under permutation of their order.*

Proof. Direct consequence of commutativity, associativity and idempotence of $\sqcup$ proven above. Any finite expression $S^{(0)} \sqcup S^{(1)} \sqcup \cdots \sqcup S^{(k)}$ depends only on the multiset $\{S^{(i)}\}_{i=0}^{k}$, not on the order or multiplicity. □

Theorem 1 (SEC for Policy-CRDT). *Assume that each replica of Policy-CRDT periodically sends its state to other replicas and that all messages are eventually delivered (no infinite losses). Then for any two replicas $r_1, r_2 \in R$ and any time t after which no new updates occur, there exists $t' \geq t$ such that $S_{r_1}(t') = S_{r_2}(t')$. In other words, Policy-CRDT satisfies Strong Eventual Consistency.*

Proof. The proof follows the standard argument for state-based CRDTs [1,5]. Let U be the set of all local updates (add/remove), and let $U_r \subseteq U$ be the subset delivered to replica r. Starting from an initial state $S_0 = (\emptyset, \emptyset)$, each update u is a monotone function $f_u : \Sigma \to \Sigma$, and the state of replica r can be written as

$$S_r = \bigsqcup_{u \in U_r} f_u(S_0).$$

By eventual delivery there exists a time t^* such that for all replicas r_1, r_2 we have $U_{r_1} = U_{r_2} = U^*$ for $t \geq t^*$. Then

$$S_{r_1} = \bigsqcup_{u \in U^*} f_u(S_0) = S_{r_2},$$

 doi:10.20944/preprints202512.1467.v1

because $\sqcup$ is commutative, associative and idempotent and the set of functions f_u is the same in both expressions. This implies convergence and thus SEC. □

Corollary 1 (Convergence of active policy sets)**.** *Under the assumptions of Theorem 1, there exists t' such that for all $r_1, r_2 \in R$,*

$$\text{Active}(S_{r_1}(t')) = \text{Active}(S_{r_2}(t')).$$

4.1. Safety of Remove-Wins for Access Control

Consider a fixed policy identifier $i \in \mathcal{I}$ with $\text{eff}(p) = \texttt{permit}$ for the corresponding policy. Let S_r^∞ be the limit state of replica r.

Proposition 2 (Anti-escalation of privileges)**.** *If there exists at least one $\texttt{removePolicy}(i)$ in the global history, then in the limit state we have $i \notin \text{Active}(S_r^\infty)$ for all replicas $r \in R$.*

Proof. By SEC all replicas converge to the same state $S_\infty = (A_\infty, T_\infty)$. Each $\texttt{removePolicy}(i)$ adds i to some local tombstone set; after convergence, $i \in T_\infty$. Active policies are $A_\infty \setminus T_\infty$, hence $i \notin \text{Active}(S_\infty)$ and thus not active on any replica. □

In contrast, in an LWW register or an add-wins set, the final state for i depends on the resolution of concurrent add/remove operations and on clock values used to break ties. Under realistic assumptions about clock skew and message reordering, revocations may fail to dominate additions, leading to states where a revoked policy remains active [2,10].

Thus, remove-wins provides a semantically conservative interpretation consistent with Zero Trust: uncertainty is resolved in favour of stricter access control (disabling potentially outdated policies) rather than silently granting permissions.

5. Architecture and Implementation

5.1. Zero Trust Deployment Architecture

Following NIST SP 800-207 and SP 800-207A [14,15], we consider an architecture with:

- a *global control plane* defining a unified set of access control policies and executing `addPolicy`/`removePolicy` operations;
- per-cluster PDPs and PEPs (e.g., in each cloud region, data center or edge site), integrated with a service mesh (such as Istio), where sidecar proxies serve as PEPs;
- a Policy-CRDT replica in each cluster, synchronised with others via state-based replication (gossip or periodic push/pull).

The global control plane publishes policy updates, which are applied to the local Policy-CRDT state and then propagated across replicas using merge messages. PEPs making run-time decisions consult the local set of active policy identifiers $\text{Active}(S_r)$ and evaluate only those policies (retrieved from a policy repository) using standard ABAC/XACML combining algorithms [11,13].

5.2. Pseudocode

We give pseudocode for Policy-CRDT operations on replica r.

Listing 1: Policy-CRDT operations on replica r

```
state S_r = (A_r, T_r)  // sets of identifiers

procedure addPolicy(id):
    // add a new policy with identifier id
    A_r := A_r ∪ { id }

procedure removePolicy(id):
    // revoke policy with identifier id
```

 doi:10.20944/preprints202512.1467.v1


```
    T_r := T_r ∪ { id }

procedure merge(S_s = (A_s, T_s)):
    // merge states of two replicas
    A_r := A_r ∪ A_s
    T_r := T_r ∪ T_s

function Active():
    // set of active policy identifiers
    return A_r \ T_r
```

The cost of `addPolicy` and `removePolicy` is $O(1)$ on average when sets are implemented as hash tables. The cost of `merge` is $O(|A_s| + |T_s|)$; known optimisations for state-based CRDTs (delta-CRDTs, digest-driven synchronisation) can reduce network overhead and merge costs [3].

5.3. Integration with PDP/PEP

Integration with ABAC/XACML-based PDPs and PEPs can be realised as follows:

- policy bodies (XACML rules, ABAC expressions) are stored in a durable repository keyed by identifiers $i \in \mathcal{I}$;
- Policy-CRDT manages only the set of active identifiers $\text{Active}(S_r)$;
- for each request q, the PDP fetches all policies whose identifiers are in $\text{Active}(S_r)$, evaluates them and combines their effects using standard combining algorithms (e.g., deny-overrides) [11,13].

Policy-CRDT is thus responsible for convergent control over which policy versions are considered active; the logic of policy evaluation and combination remains unchanged.

6. Analytical Evaluation

In this section we provide a reproducible analytical evaluation of Policy-CRDT in comparison with LWW and add-wins strategies. All numerical values are derived from explicit formulas;

6.1. Convergence Latency Under Gossip Replication

Consider N replicas in a fully connected network. Suppose that CRDT states are disseminated using a randomized gossip protocol (push or push-pull), as analysed in [16,17]. These works show that the expected number of rounds until all nodes are informed is $O(\log N)$.

As a simple approximation we take

$$R(N) = \lceil \log_2 N \rceil$$

as a proxy for the expected number of communication rounds until convergence, assuming no new updates are introduced. The actual wall-clock time is proportional to $R(N)$ with a factor determined by average network latency.

Figure 1 shows $R(N)$ for $N \in \{2, 4, 8, 16, 32, 64\}$.

6.2. Model Of Conflicts And Dangerous States

We now consider an abstract probabilistic model of concurrency for a single policy identifier $i \in \mathcal{I}$:

- among all updates affecting i, a fraction $c \in [0, 1]$ participates in concurrent add/remove conflicts;
- for each such conflict, we are interested in whether the final state of i is *dangerous*, i.e., whether the policy remains active even though at least one revocation attempt occurred.

For simplicity we assume:

 doi:10.20944/preprints202512.1467.v1

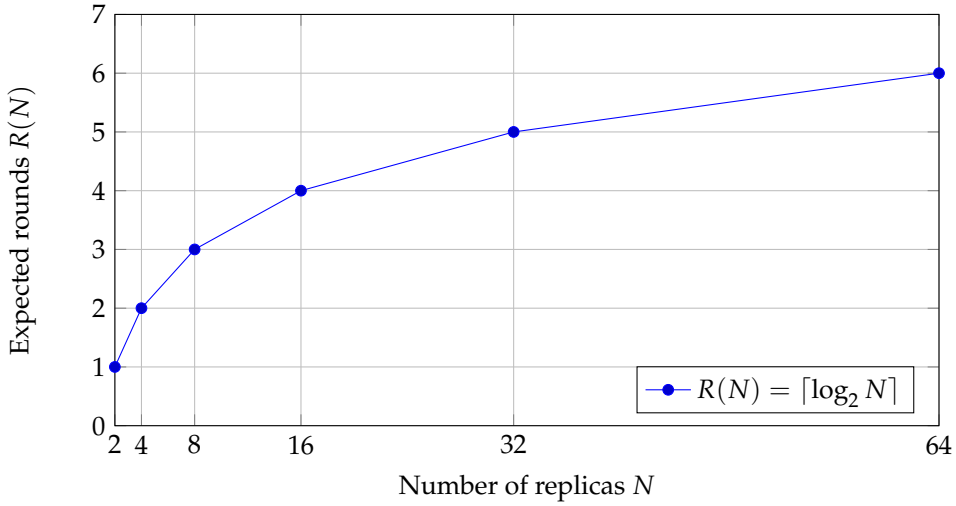

Figure 1. Approximate number of communication rounds to convergence in a gossip-based replication model, using $R(N) = \lceil \log_2 N \rceil$ as a simple proxy based on $O(\log N)$ bounds from [16,17].

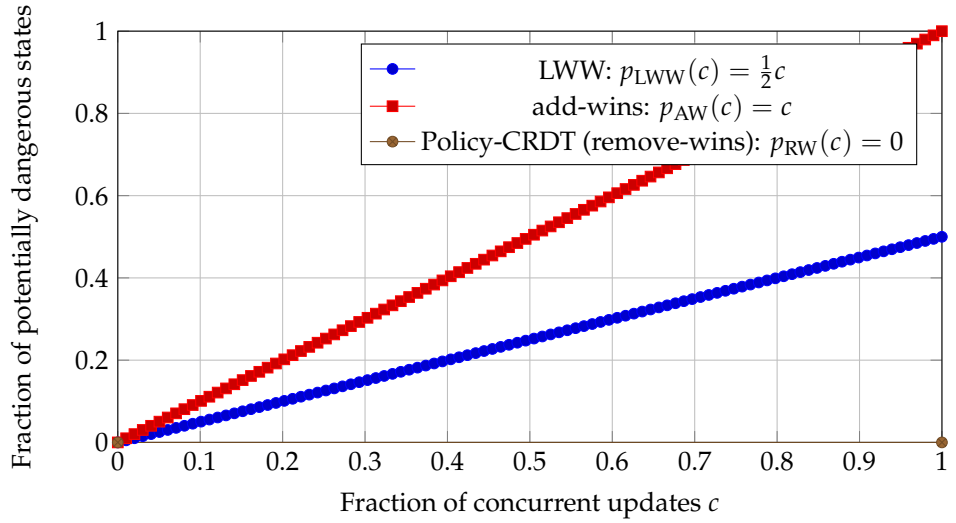

Figure 2. Comparison of the fraction of potentially dangerous states for different conflict resolution strategies in a simple model of concurrent updates.

- in an LWW scheme, add and remove are equally likely to "win" in a conflict (e.g., due to randomised tie-breakers), so each conflicting pair yields a dangerous outcome with probability $1/2$;
- in an add-wins set, any conflict leads to the policy being active, so each conflict is dangerous;
- in Policy-CRDT (remove-wins), any conflict eventually yields the policy inactive, so conflicts are never dangerous.

Let $p_{\mathrm{LWW}}(c)$, $p_{\mathrm{AW}}(c)$ and $p_{\mathrm{RW}}(c)$ be the expected fractions of dangerous states for a fixed c. With the assumptions above we obtain

$$p_{\mathrm{LWW}}(c) = \frac{1}{2}c, \qquad p_{\mathrm{AW}}(c) = c, \qquad p_{\mathrm{RW}}(c) = 0.$$

Figure 2 plots these functions.

6.3. Tabular Results

Combining the expressions for $R(N)$ and $p_{\mathrm{LWW}}(c)$, $p_{\mathrm{AW}}(c)$, $p_{\mathrm{RW}}(c)$, we can build a table for several representative values of N and c.

 doi:10.20944/preprints202512.1467.v1

All values in Table 1 follow directly from the formulas above, so the results are fully reproducible.

Table 1. Analytical estimates of convergence rounds $R(N)$ and fractions of dangerous states for different strategies at selected numbers of replicas N and concurrency fractions c.

N	$R(N) = \lceil \log_2 N \rceil$	c	$p_{\mathrm{LWW}}(c)$	$p_{\mathrm{AW}}(c)$	$p_{\mathrm{RW}}(c)$
4	2	0.1	0.05	0.10	0
8	3	0.1	0.05	0.10	0
16	4	0.5	0.25	0.50	0
32	5	0.5	0.25	0.50	0
64	6	0.9	0.45	0.90	0

7. Discussion and Limitations

7.1. Practical Benefits of Policy-CRDT

The analytical comparison indicates that Policy-CRDT with remove-wins semantics eliminates an entire class of dangerous states related to conflicting additions and revocations of policies:

- for any non-zero concurrency level $c > 0$, LWW and add-wins admit a non-zero fraction of dangerous states proportional to c;
- Policy-CRDT guarantees zero dangerous states in this model, because any revocation eventually suppresses the policy globally.

In a Zero Trust setting, where decisions are continuously enforced and do not rely on network locality for trust [14,15], this represents a meaningful safety improvement: in the presence of network delays or partitions the worst-case effect of using Policy-CRDT is temporary over-denial of some legitimate operations, whereas LWW/add-wins can silently permit operations that should be prohibited according to revocations.

7.2. Overheads And Tombstone Growth

A classical drawback of 2P-Set-based CRDTs, and thus of Policy-CRDT, lies in the unbounded growth of the tombstone set T: identifiers added to T are never removed, which can cause memory and bandwidth overhead. Possible mitigation strategies include:

- use of large random identifiers with negligible collision probability, enabling occasional "re-keying" or truncation of identifier spaces with carefully designed safety guarantees;
- periodic compaction of states using safe garbage-collection procedures based on globally agreed cut-offs (e.g., via durable logs or consensus) or on more advanced CRDT designs [3,8];
- partitioning of policies into domains and replication of separate CRDT instances per domain to bound state sizes.

Designing such mechanisms while preserving the security guarantees is non-trivial and is left for future work.

7.3. Policy Strictness and Lack of Some Invariants

Remove-wins semantics can lead to overly strict policies in scenarios where temporary revocations are quickly followed by re-authorisation intent. Under Policy-CRDT, a re-authorisation must use a new policy identifier, which is good for audit history but can complicate lifecycle management.

Moreover, while Policy-CRDT provides SEC for the set of active policies, it does not automatically enforce higher-level global invariants that depend on combinations of policies (e.g., caps on the total number of delegated privileges) [3,5]. Enforcing such invariants generally requires additional coordination or specialised coordination-free designs.

7.4. Future Work

Promising directions for future research include:

 doi:10.20944/preprints202512.1467.v1

- integration of Policy-CRDT with cryptographic enforcement mechanisms (e.g., CP-ABE or HABE), where keys and ciphertext policies are derived from replicated policy identifiers;
- extension of the model to temporal policies with time-bounded access and deadlines, which require consistent interpretation of time under weak consistency;
- formal specification and machine-checked verification of Policy-CRDT using existing frameworks for CRDT verification [18].

8. Conclusions

This paper presented a formal algebraic approach to replication of access control policies in asynchronous multi-cloud and edge environments based on a conflict-free replicated data type called Policy-CRDT with remove-wins semantics. The state of Policy-CRDT is defined as a 2P-Set over policy identifiers, forming a join-semilattice; local updates are monotone and the merge function computes least upper bounds, ensuring Strong Eventual Consistency under eventual delivery.

We proved that the remove-wins strategy enforces an anti-escalation property: any policy for which at least one revocation occurs is eventually inactive on all replicas. A simple analytical model shows that, unlike LWW and add-wins strategies, Policy-CRDT eliminates a family of dangerous states with excessive permissions for any level of concurrency in updates.

We described a deployment architecture integrating Policy-CRDT into a Zero Trust PDP/PEP infrastructure and provided a reproducible analytical evaluation of convergence latency and safety characteristics. Policy-CRDT thus offers a formally justified and practically relevant mechanism for convergent access control in asynchronous distributed systems and paves the way for systematic use of CRDT techniques in cybersecurity.

References

1. M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-free replicated data types," in *Stabilization, Safety, and Security of Distributed Systems (SSS 2011)*, LNCS, vol. 6976, Springer, 2011, pp. 386–400. doi: 10.1007/978-3-642-24550-3_29.
2. M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "A comprehensive study of convergent and commutative replicated data types," INRIA Research Report RR-7506, 2011.
3. N. Preguiça, "Conflict-free replicated data types: An overview," arXiv:1806.10254, 2018.
4. N. Preguiça, C. Baquero, and M. Shapiro, "Conflict-free replicated data types (CRDTs)," in *Encyclopedia of Big Data Technologies*, Springer, 2018. doi: 10.1007/978-3-319-63962-8_185-1.
5. S. Burckhardt, "Principles of eventual consistency," *Foundations and Trends in Programming Languages*, vol. 1, nos. 1–2, pp. 1–150, 2014. doi: 10.1561/2500000011.
6. Y. Saito and M. Shapiro, "Optimistic replication," *ACM Computing Surveys*, vol. 37, no. 1, pp. 42–81, 2005. doi: 10.1145/1057977.1057980.
7. "Conflict-free replicated data type," Wikipedia, the free encyclopedia, accessed 5 December 2025. URL: https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type.
8. A. Bieniusa, M. Zawirski, N. Preguiça, M. Shapiro, C. Baquero, V. Balegas, and S. Duarte, "An optimized conflict-free replicated set," arXiv:1210.3368, 2012.
9. P. Samarati, P. Ammann, and S. Jajodia, "Maintaining replicated authorizations in distributed database systems," *Data & Knowledge Engineering*, vol. 18, no. 1, pp. 55–84, 1996. doi: 10.1016/0169-023X(95)00000-I.
10. T. Wobber, D. Terry, and T. Rodeheffer, "Policy-based access control for weakly consistent replication," in *Proceedings of the 5th European Conference on Computer Systems (EuroSys 2010)*, ACM, 2010, pp. 293–306. doi: 10.1145/1755913.1755943.
11. V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*, NIST Special Publication 800-162, 2014. doi: 10.6028/NIST.SP.800-162.
12. V. C. Hu, D. Ferraiolo, and R. Kuhn, "Attribute-based access control," *IEEE Computer*, vol. 48, no. 2, pp. 85–88, 2015. doi: 10.1109/MC.2015.33.
13. OASIS, *eXtensible Access Control Markup Language (XACML) Version 3.0*, OASIS Standard, Jan. 2013. URL: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

 doi:10.20944/preprints202512.1467.v1

14. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, *Zero Trust Architecture*, NIST Special Publication 800-207, 2020. doi: 10.6028/NIST.SP.800-207.
15. R. Chandramouli and Z. Butcher, *A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments*, NIST Special Publication 800-207A, 2023. doi: 10.6028/NIST.SP.800-207A.
16. R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking, "Randomized rumor spreading," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*, IEEE, 2000, pp. 565–574. doi: 10.1109/SFCS.2000.892324.
17. B. Doerr and A. Kostrygin, "Randomized rumor spreading revisited," in *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, LIPIcs, vol. 80, 2017, pp. 138:1–138:14. doi: 10.4230/LIPIcs.ICALP.2017.138.
18. P. Zeller, A. Bieniusa, and P. Thiemann, "Formal specification and verification of CRDTs," in *Correct System Design*, LNCS, vol. 9360, Springer, 2015, pp. 33–54. doi: 10.1007/978-3-662-46681-0_3.