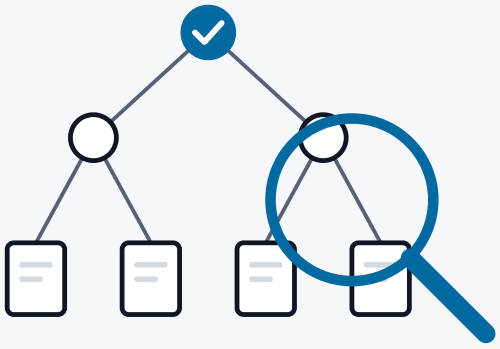
A-SIT Technology

E-Government Innovationszentrum

29. Oktober 2026 Konzept · Transparenz · Digitale Identität · 28 Min. Lesezeit

Verifizierbares Monitoring für Transparency Logs

Edona Fasllija



Autorinnen und Autoren	Edona Fasllija
Quelle	https://technology2.a-sit.at/veroeffentlichungen/verifizierbares-monitoring-fuer-transparency-logs/
Hinweis	Maschinell übersetzt.

Kernaussagen

- Transparency Logs liefern Inclusion- und Konsistenznachweise, aber keinen Nachweis, dass eine inhaltsbasierte Abfrage vollständig beantwortet wurde.
- Ein Verifiable Index (Log-Map-Log) bildet Schlüssel auf Positionen im Input Log ab; ein Output Log hält die Index-Revisionen fest.
- Bei etwa 16 Millionen Blättern dauert eine inhaltsbasierte Abfrage per Log-Scan 858,32 s, über den Index 0,047 ms (SMT), 0,100 ms (MPT) oder 0,179 ms (IAVL).
- Der Index kostet Aufbau und Speicher: etwa 1,54 GB (SMT), 2,60 GB (MPT), 2,94 GB (IAVL) plus rund 2,06 GB WAL; Inclusion Proofs gewinnen kaum.
- Für Certificate Transparency wird je Log eine eigene Index-Instanz vorgeschlagen; Benachrichtigungen werden durch periodische Heartbeats mit Nachweis ergänzt.

Transparenzsysteme wie Certificate Transparency und Binary Transparency ermöglichen es Clients, zu überprüfen, ob ein bestimmtes Objekt in einer öffentlichen, ausschließlich erweiterbaren Historie (*Append-Only-Log*) erfasst wurde und ob spätere Zustände (*Checkpoints*) auf früheren aufbauen. Was diese Systeme jedoch nicht von sich aus bieten, ist eine kryptografisch vollständige Antwort auf inhaltsbasierte Anfragen wie „Zeige mir alle Zertifikate für diese Domain“ oder „Zeige mir alle veröffentlichten Versionen dieses Moduls“. Nutzer müssen entweder das gesamte Log selbst auslesen und analysieren oder externen Such- und Indexierungsdiensten vertrauen, die möglicherweise Ergebnisse auslassen.Verifiable Indices setzen an dieser Lücke an und schlagen einen kontinuierlich gepflegten, durchgängig verifizierbaren Index vor, der auf einem bestehenden Append-Only-Log aufbaut. Der Index leitet inhaltsbasierte Schlüssel aus den Log-Einträgen ab, ordnet diese den entsprechenden Positionen im ursprünglichen Log zu und veröffentlicht kryptografisch authentifizierte Indexzustände, die unabhängig überprüft werden können. Dieser Bericht stellt das Modell und die Architektur von Verifiable Indices vor, fasst die Benchmark-Ergebnisse eines Prototyps zusammen und diskutiert deren Anwendung im Bereich Certificate Transparency.

1. Einleitung

Transparenzsysteme machen sicherheitsrelevante Ereignisse öffentlich überprüfbar, indem sie diese in ausschließlich erweiterbaren Logs (Append-Only-Logs) festschreiben. Dasselbe Prinzip findet sich bei der Ausstellung von Zertifikaten, bei Software-Prüfsummen, Build-Attestierungen, Firmware-Releases und in anderen Szenarien der Lieferkette. Certificate Transparency (CT) ist die wichtigste Umsetzung: Zertifikate und Vorzertifikate werden in unabhängig betriebenen Merkle-Tree-Logs erfasst, und die Logs liefern kompakte Nachweise, dass ein Eintrag vorhanden ist und dass ein späterer Zustand des Baums einen früheren erweitert. Diese Eigenschaften sind für die Integrität der Veröffentlichung sehr wirksam, stellen aber keine vollständige Monitoring-Lösung dar. Ein Domaininhaber kennt in der Regel die Log-Positionen der Zertifikate für seine Domains nicht. Er benötigt eine inhaltsbasierte Abfrage, etwa „Zeige mir jeden Eintrag, der example.com betrifft“. Diese Lücke wird heute üblicherweise von Monitoren geschlossen [1]. Ein Monitor sammelt neue Einträge, wertet sie aus, baut einen eigenen Suchindex auf und stellt Such- oder Benachrichtigungsdienste bereit. Das ist im Betrieb wirksam, schafft aber eine neue Vertrauensgrenze: Nutzer können zwar prüfen, dass ein zurückgegebenes Zertifikat tatsächlich geloggt wurde, aber nicht effizient überprüfen, ob der Monitor jedes passende Zertifikat zurückgegeben hat. Neuere Arbeiten zu CT-Monitoren von Drittanbietern haben gezeigt, dass Unvollständigkeit nicht nur theoretisch auftritt [2].

Bei chronologischen Logs ist effiziente Suche jedoch keine Grundoperation. Das Log ist nach Zeit bzw. Einfügeposition geordnet. Ein Client, der ein Blatt (Leaf) bereits kennt, kann einen Inclusion Proof anfordern, und ein Client, der zwei Checkpoints kennt, kann das ausschließlich erweiternde Wachstum überprüfen. Monitoring stellt eine andere Frage: Welche Einträge sind für mich relevant? Das ist in der Regel ein Prädikat über den Inhalt und nicht über die Position. Ein nicht indexiertes Log bietet daher Integrität der Veröffentlichung im Sinne eines Append-Only-Logs, aber keine verifizierbare Vollständigkeit bei prädikatbasierter Abfrage. Praktisch ergibt sich daraus ein wiederkehrender Zielkonflikt. Eine Stakeholderin oder ein Stakeholder kann das Log erneut abspielen und lokal indexieren. Das bewahrt das Ende-zu-Ende-

Vertrauen, verursacht aber erheblichen Speicher- und Verarbeitungsaufwand. Oder die Überwachung wird an Dritte ausgelagert. Das ist betrieblich attraktiv, macht Vollständigkeit aber zu einer Zusage auf Dienstebene und nicht zu einer kryptografisch überprüfbaren Eigenschaft.

Das zentrale Forschungsproblem besteht daher nicht darin, Append-Only-Logs zu ersetzen, sondern verifizierbare inhaltsbasierte Abfragen hinzuzufügen, während das Log die maßgebliche Quelle bleibt und das Monitoring praktikabel bleibt. Ein Verifiable Index fügt eine authentifizierte Abfrageschicht hinzu, ohne das ursprüngliche Log zu ersetzen. Der aktuelle Prototyp von transparency-dev beschreibt eine „Log-Map-Log"-Datenstruktur: Ein Input Log wird Blatt für Blatt verarbeitet, eine deterministische Mapping-Funktion wandelt jeden Eintrag in einen oder mehrere Lookup-Schlüssel um, eine verifizierbare Map ordnet diesen Schlüsseln Verweise zurück ins Input Log zu, und ein Output Log hält aufeinanderfolgende Revisionen der Map fest, sodass die Map historisch verifizierbar ist [3]. Dieses Projekt untersucht den Entwurf im Detail, einschließlich eines Write-Ahead-Transaktionslogs (WAL), mehrerer authentifizierter Map-Backends, APIs zur Auslieferung von Nachweisen und Leistungsmessungen auf Basis von SumDB-Daten.

2. Hintergrund

2.1. Append-Only-Logs

Chronologische Append-Only-Logs liefern Mitgliedschafts- und Konsistenznachweise, aber keine allgemeine durchsuchbare Datenbankschnittstelle. In CT kann ein Client ein SCT, einen Inclusion Proof und einen Konsistenznachweis zwischen Baumköpfen (STH) überprüfen. Ein Monitor kann neue STHs abrufen, neue Einträge beziehen, relevante Zertifikate klassifizieren und prüfen, ob der aus den abgerufenen Einträgen abgeleitete Baum zum neuen STH passt oder einen früheren STH konsistent erweitert. Das Protokoll definiert jedoch keinen Nachweis, der eine inhaltliche Abfrage beantwortet, etwa „alle für example.com relevanten Zertifikate". Vollständigkeit für diese Frage wird stattdessen im Betrieb erreicht, indem der Monitor jeden neuen Eintrag abruft und auswertet.

Dieser Unterschied ist die zentrale Einschränkung nicht indexierter chronologischer Logs. Sie liefern Nachweise für Aussagen der Form „Blatt l ist in Log n enthalten" und „Checkpoint n erweitert Checkpoint m", aber keine Nachweise für Aussagen der Form „diese zurückgegebene Menge ist die vollständige Menge aller Blätter, die das Prädikat P bis zum Präfix n erfüllen". Genau an diesem Punkt setzen Verifiable Indices an.

Für das Monitoring bedeutet die fehlende Eigenschaft, dass die inhaltliche Suche faktisch auf das vollständige erneute Abspielen des Logs hinausläuft. Der Monitor-Ablauf nach RFC 9162 verlangt wiederholtes Abfragen auf STH-Änderungen, das Abrufen aller neuen Blätter, die Prüfung, ob jeder Eintrag „von Interesse" ist, und anschließend entweder das Speichern des gesamten Baums oder genügend Aufwand, um die Konsistenz gegenüber den abgerufenen Einträgen und Nachweisen zu überprüfen. Bei Logs mit hohem Volumen ist das nicht nur unbequem, sondern der zentrale Engpass bei der Skalierung. Ein rohes chronologisches Log ist daher ein gutes Mittel zur Veröffentlichung, aber ein schlechtes Mittel für die Vollständigkeit von Abfragen.

Daraus ergibt sich ein wichtiger Zielkonflikt beim Vertrauen. Wer die Indexierung an einen gewöhnlichen Suchdienst auslagert, gewinnt Effizienz, verliert aber eine formale Vollständigkeitsgarantie. Der Monitor kann Ergebnisse versehentlich, unter Last oder böswillig auslassen, und die Nutzer haben keinen knappen

kryptografischen Beleg für die Auslassung. Die Motivation für den Verifiable Index ist genau, diesen Zielkonflikt zu vermeiden, indem ein „Stichwortverzeichnis“ eingeführt wird, dessen Antworten selbst durch kryptografische Nachweise abgedeckt sind.

Monitoring als dreistufige Pipeline

In der Praxis möchten nur wenige Domaininhaber jedes CT-Log erneut abspielen. Monitore von Drittanbietern sammeln daher Log-Einträge, bauen Datenbanken auf, stellen eine Domainsuche bereit und versenden Benachrichtigungen. Die Schwierigkeit besteht darin, dass CT-Nachweise den Log-Eintrag authentifizieren, nicht aber die Vollständigkeit der Datenbankabfrage eines Monitors. Sun et al. [2] beschreiben Monitore von Drittanbietern als wesentliche CT-Komponenten und berichten, dass verbreitete Dienste nicht immer vollständige Zertifikatsmengen zurückgeben. Ihr Watcher-Prototyp deckte in einem 52-tägigen Test mehrere Implementierungs- oder Entwurfsprobleme auf.

Die Folgen für die Sicherheit sind subtil. Ein Nutzer kann von einem Monitor ein Zertifikat erhalten und prüfen, dass es tatsächlich geloggt ist, das sagt aber nichts über ein ausgelassenes Zertifikat aus. Monitoring verlagert das Vertrauensproblem daher vom Log-Betreiber auf den Monitor, sofern der Nutzer das Log nicht selbst erneut abspielt oder die Indexierungsschicht nicht selbst verifizierbar wird.

Ein nützliches Betriebsmodell unterteilt das Monitoring in drei Stufen (Abbildung 1): Datenerfassung, Indexierung sowie Abfrage und Benachrichtigung. Die Datenerfassung ruft fortlaufend neu angehängte Einträge aus den relevanten Transparenz-Logs ab. Die Indexierung wertet diese Rohdaten aus und extrahiert Attribute, die zu Lookup-Schlüsseln werden. Die Abfrage stellt den indexierten Bestand den Nutzern zur Verfügung, während die Benachrichtigung hinterlegte Abfragen oder Abonnements proaktiv anwendet. Diese Aufteilung ist wertvoll, weil sie zeigt, wo Vertrauen entsteht.

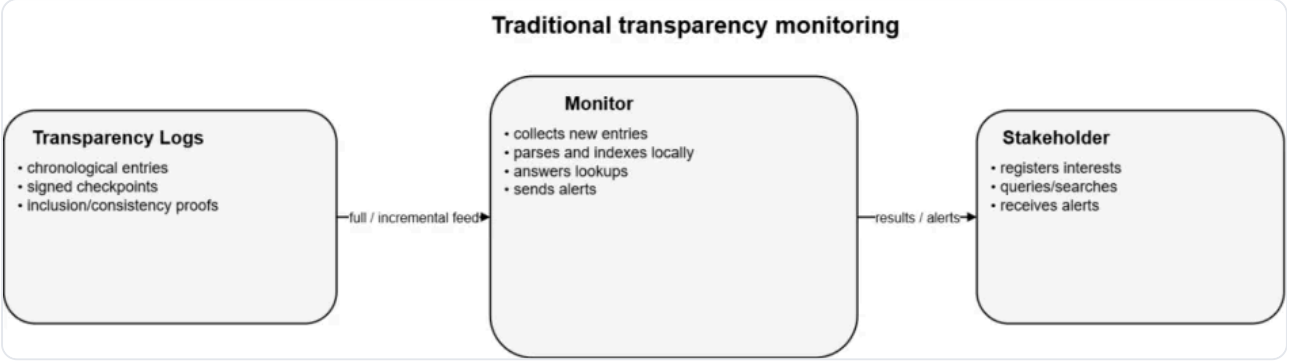


Abbildung 1: Traditionelles Monitoring

Die Datenerfassung muss mit jedem relevanten Input Log Schritt halten und sich nach Ausfallzeiten korrekt erholen. Die Indexierung muss Einträge deterministisch auswerten und darf fehlerhafte oder unerwartete Einträge nicht stillschweigend verwerfen. Die Abfrage muss die beabsichtigte Treffermenge zurückgeben, nicht bloß einzeln gültige Einträge.

Die Benachrichtigung hängt zusätzlich von der Verfügbarkeit des Dienstes und der Zustellung ab: „Keine Nachricht“ kann „kein Ereignis“ oder „der Monitor ist ausgefallen“ bedeuten. Die Suche wird vom Nutzer ausgelöst. Ist ein Abfragedienst nicht erreichbar oder liefert er einen Fehler, ist der Ausfall sofort sichtbar, und der Nutzer kann es erneut versuchen, den Anbieter wechseln oder das zugrunde liegende Log prüfen.

Die Benachrichtigung wird vom Dienst ausgelöst. Schweigen ist grundsätzlich mehrdeutig. Ein Monitor, der ausgefallen, verzögert oder falsch konfiguriert ist oder Ergebnisse gezielt unterdrückt, kann genau so aussehen wie ein Monitor, der schlicht nichts Neues gefunden hat.

2.2. Stand der Technik: Chronologische Transparenz-Logs und Monitoring

Die wichtigsten Transparenz-Einsätze lassen sich heute in zwei große Familien einteilen. Die erste Familie ist das klassische Append-Only-Log: CT[4], SumDB[5], [6], Rekor[7] und viele Software-Transparenz-Einsätze. Die zweite Familie ist das authentifizierte Verzeichnis bzw. die prüfbare Map, die Key-Transparency-Systemen zugrunde liegt: CONIKS [8], SeemLESS [9] und die AKD-basierten Einsätze von Meta [9] . Die erste Familie ist stark bei öffentlicher Veröffentlichung und Rechenschaft über ausschließlich erweiternde Historien, die zweite bei der Suche über Schlüssel und in vielen Fällen bei privater oder halbprivater Mitgliedschaft. Ein Verifiable Index wirkt als Brücke: Er legt über die erste Familie einen Teil der Abfragefähigkeit der zweiten Familie, ohne das Input Log als maßgebliche Quelle zu ersetzen.

Aus dieser Landschaft ergeben sich drei Entwurfsmuster. Erstens bieten Append-Only-Logs eine starke Integrität der öffentlichen Historie mit knappen Inclusion- und Konsistenznachweisen. Zweitens bieten authentifizierte Verzeichnisse eine starke Integrität bei der Suche über Schlüssel, oft mit Datenschutzmerkmalen wie VRF-abgeleiteten Positionen. Die gängige Praxis behandelt diese weiterhin als getrennte Teilsysteme. Was größtenteils fehlt, ist eine generische, wiederverwendbare Überlagerung, die Abfragesemantik in ein ansonsten gewöhnliches Append-Only-Log einbringt. Genau das ist die Rolle eines Verifiable Index.

Das Input Log bleibt die maßgebliche Struktur für die Veröffentlichung, der Verifiable Index ist eine eigene Datenstruktur, und das Output Log macht die Historie des Index öffentlich überprüfbar. Das unterscheidet sich konzeptionell vom Aufbau eines vollständigen Key-Transparency-Dienstes vor den Clients, bei dem das Log in den Hintergrund tritt.

Der Entwurf ähnelt eher einer kontinuierlichen Überlagerung eines externen Log-Präfixes: Er fragt den neuesten Checkpoint des Input Logs ab, verarbeitet das Delta, schreibt eine neue Index-Wurzel fest und liefert nur den neuesten Zustand sowie inkrementelle Deltas pro Schlüssel aus. Die historische Korrektheit wird geprüft, indem die deterministische Mapping-Logik erneut über das Input Log ausgeführt und mit der Historie des Output Logs verglichen wird.

CT ist ein besonders starkes Beispiel, weil Monitoring für das angestrebte Sicherheitsergebnis zentral ist. RFC 9162 beschreibt den Zweck von CT darin, TLS-Serverzertifikate öffentlich zu loggen, damit verdächtige Ausstellungen bemerkt und die Tätigkeit von CAs geprüft werden können [10]. Das Log entscheidet nicht, ob ein Zertifikat für die genannte Organisation legitim ist. Diese Feststellung erfordert externe Ground Truth: Eigentum an oder betriebliche Kontrolle über die Domain, erwartete Aussteller, organisatorische Richtlinien, Zertifikatsinventar und manchmal den Einsatzkontext.

CT hat außerdem mehrere Merkmale, die die Indexierung erschweren. Das Ökosystem enthält mehrere unabhängige Logs. Einträge können Zertifikate oder Vorzertifikate sein. Dasselbe Zertifikat kann in mehreren Logs erscheinen, und für ein und dasselbe aus Sicht der Nutzer logische Zertifikatsereignis können mehrere Einträge entstehen. Ein Zertifikat kann mehrere DNS-Namen enthalten. Wildcard-Namen

bringen eine weitere Ebene an Abgleichssemantik mit sich. Diese Details sind beherrschbar, bedeuten aber, dass „Indexierung nach Domain“ präzise spezifiziert werden muss und nicht als informeller Auswertungsschritt behandelt werden darf.

3. Entwurf des Verifiable Index

Ein Verifiable Index ist eine authentifizierte, abfragbare Schicht über einem bestehenden Transparenz-Log. Konzeptionell verhält er sich wie ein Stichwortverzeichnis: Ein aussagekräftiger Schlüssel verweist auf Positionen im chronologischen Quell-Log. Der Unterschied zu einer normalen Suchdatenbank besteht darin, dass sowohl die Schlüsselsuche als auch der historische Zustand des Index kryptografisch festgeschrieben sind.

Der zentrale Entwurfsgedanke ist, drei Rollen zu trennen, die oft vermischt werden. Das Input Log bleibt die Quelle der Wahrheit für die Aufnahme und die Reihenfolge der Anhänge. Eine Mapping-Funktion ordnet Suchschlüsseln Posting-Listen von Positionen im Input Log zu. Ein Output Log schreibt im Zeitverlauf aufeinanderfolgende Index-Wurzeln fest und liefert dem Index so eine öffentliche, ausschließlich erweiterbare Nachweiskette. Die Map des Verifiable Index sitzt somit zwischen einem bestehenden Input Log und einem neu geschaffenen Output Log. Verifiable Indices beruhen auf einem kontinuierlichen Modell: Die Überlagerung wird aktualisiert, sobald neue Log-Einträge eintreffen, der Dienst antwortet nur gegen den neuesten festgeschriebenen Zustand, und die historische Absicherung entsteht durch die Prüfung des Output Logs, nicht dadurch, dass Clients beliebige historische Momentaufnahmen erhalten. [3] Wichtig: Der Verifiable Index entscheidet nicht, ob ein zurückgegebener Eintrag legitim ist. Er belegt, was das indexierte Log unter einem festgelegten Mapping enthält. Ein Stakeholder bzw. Monitor wendet weiterhin externe Richtlinien oder Ground Truth an, um zu entscheiden, ob diese Einträge akzeptabel sind.

3.1. Entitäten

Ein Verifiable-Index-Ökosystem umfasst mehrere Akteure, die jeweils eine eigene Rolle beim Erzeugen, Organisieren, Überprüfen und Nutzen von Transparenzinformationen haben.

Der erste Akteur ist der Claimant. Das ist die Entität, die eine Aussage erstellt, die öffentlich überprüfbar werden soll. In Certificate Transparency ist der Claimant typischerweise eine CA, die ein Zertifikat oder Vorzertifikat ausstellt. In einem Binary-Transparency-System kann es ein Software-Herausgeber sein, der eine neue Paketversion veröffentlicht. Allgemeiner lässt sich ein Claim als strukturiertes Objekt modellieren, das einen Typ, ein Subjekt, eine Nutzlast und zugehörige Metadaten enthält. Das Subjekt bezeichnet die Entität, auf die sich der Claim bezieht, die Nutzlast enthält die eigentliche Aussage.

Diese Claims werden an ein Append-Only-Log übermittelt. Das Log führt eine geordnete Folge von Einträgen und veröffentlicht in regelmäßigen Abständen signierte Checkpoints, die den aktuellen Zustand des Logs festschreiben. Jeder Checkpoint enthält die Identität des Logs, die aktuelle Größe des Logs, eine Merkle-Wurzel über alle Einträge bis zu dieser Größe und eine Signatur des Logs. Das Log unterstützt die üblichen Operationen eines Transparency Logs, etwa Inclusion Proofs für einzelne Einträge und Konsistenznachweise zwischen Checkpoints. So können Clients und Monitore überprüfen, dass ein bestimmter Eintrag enthalten ist und dass das Log im Zeitverlauf ausschließlich erweiternd gewachsen ist.

Der Indexer ist dafür zuständig, das rohe Log in eine abfragbare Struktur umzuwandeln. Er liest Log-Einträge, wendet eine deterministische Extraktionsfunktion an, um zu bestimmen, unter welchen Schlüsseln jeder Eintrag indexiert wird, und baut über diese Schlüssel-Wert-Zuordnungen einen authentifizierten Index auf. Ein Zertifikat kann beispielsweise unter jedem darin enthaltenen Domainnamen indexiert werden, während ein MTC-Eintrag unter seiner Domain, seinem Aussteller oder seinem Zertifikatstyp indexiert werden kann. Der Indexer veröffentlicht dann einen Index-Checkpoint, der die Index-Wurzel, den zugehörigen Checkpoint des Quell-Logs und die Indexierungsrichtlinie aneinander bindet. Wichtig: Der Indexer gilt nicht als vertrauenswürdig. Seine Antworten müssen von kryptografischen Nachweisen begleitet sein, und seine Index-Checkpoints müssen unabhängig überprüfbar sein.

Ein Monitor stellt diese unabhängige Überprüfung bereit. Seine Aufgabe ist es zu prüfen, dass der Index getreu aus dem zugrunde liegenden Log abgeleitet wird. Ein Monitor kann neue Log-Einträge abrufen, betroffene Teile des Index neu berechnen, Nachweise für Index-Aktualisierungen überprüfen und feststellen, ob abfragerelevante Einträge ausgelassen wurden. In stärkeren Einsatzformen können Monitore den vollständigen Index-Zustand selbst neu berechnen. In leichteren Einsatzformen können sie inkrementelle Aktualisierungen überprüfen und Warnungen auslösen, wenn der Indexer sich inkonsistent oder unvollständig verhält. Diese Rolle verallgemeinert die Funktion von Certificate-Transparency-Monitoren, erweitert sie aber vom bloßen Beobachten von Log-Einträgen auf die Prüfung der Korrektheit einer authentifizierten Abfrageschicht.

Ein Witness bzw. Cosigner kann eingesetzt werden, um die Überprüfung des Index für leichtgewichtige Clients praktikabel zu machen. Statt von jedem Client zu verlangen, jede Index-Aktualisierung zu überprüfen, verifizieren Witnesses die Aktualisierungen des Indexers und signieren anschließend die resultierenden Index-Checkpoints. Ein Client kann einen Index-Checkpoint akzeptieren, wenn er nach der lokalen Richtlinie ein ausreichendes Quorum an Witness-Signaturen trägt. Das verlagert aufwendige Überprüfungsarbeit auf eine kleinere Gruppe spezialisierter Akteure und gibt den Clients dennoch kryptografische Belege, dass der Index-Checkpoint von unabhängigen Parteien geprüft wurde. Witnesses verringern außerdem das Risiko von Equivocation, da ein Indexer, der widersprüchliche Index-Historien präsentiert, widersprüchliche Witness-Signaturen erlangen müsste oder Gefahr liefe, entlarvt zu werden.

Der letzte Akteur ist der abfragende Client. Das kann ein Browser, ein Domaininhaber, ein Paketmanager, eine Wallet, eine Relying Party oder ein Monitoring-Dienst sein. Der Client sendet eine Abfrage an den Indexer und erhält eine Antwort zusammen mit einem Nachweis. Die Antwort kann passende Log-Einträge, Log-Indizes, Digests oder eine Projektion der relevanten Daten enthalten. Der Nachweis erlaubt dem Client zu überprüfen, dass die Antwort mit der authentifizierten Index-Wurzel konsistent ist und dass die Index-Wurzel an einen bestimmten Checkpoint des Quell-Logs gebunden ist. Je nach Richtlinie kann der Client vor der Annahme des Ergebnisses auch Witness-Signaturen, Aktualitätsanforderungen, Inclusion Proofs des Quell-Logs und Belege gegen Equivocation überprüfen.

Zusammen trennen diese Entitäten die Verantwortlichkeiten und vermeiden volles Vertrauen in einen einzelnen Suchdienst. Das Log sorgt für ausschließlich erweiternde Veröffentlichung, der Indexer für effizienten Abfragezugriff, Monitore und Witnesses für unabhängige Überprüfung, und Clients prüfen Nachweise, bevor sie sich auf Antworten verlassen. Die Sicherheit des Gesamtsystems hängt nicht nur von den authentifizierten Datenstrukturen des Indexers ab, sondern auch von der Entsprechung zwischen Index und Quell-Log, der Aktualität akzeptierter Checkpoints und den Mechanismen gegen Equivocation, die sicherstellen, dass alle Clients eine konsistente Index-Historie sehen.

3.2. Modell des Verifiable Index

Jede Instanz eines Verifiable Index wird durch ein Paar (L, μ) definiert. L bezeichnet das Input Log. μ bezeichnet die feste Mapping-Semantik: wie Rohblätter ausgewertet, kanonisiert, in Index-Schlüssel umgewandelt und als Werte dargestellt werden. Diese Unterscheidung ist wichtig, weil dasselbe chronologische Log unterschiedliche nützliche Indexstrukturen unterstützen kann, je nachdem, wie seine Einträge interpretiert werden und welche Abfragen der Index beantworten soll.

Der Parameter L bezeichnet die maßgebliche Quelle der Einträge. Ein VIndex verarbeitet die Einträge dieses Logs in ihrer ursprünglichen Anhängereihenfolge und pflegt einen authentifizierten Index über das bereits verarbeitete Präfix. Der Index ersetzt daher das Transparenz-Log nicht und wird auch nicht zu einer zweiten Quelle der Wahrheit. Das ursprüngliche Log bleibt maßgeblich für Existenz, Reihenfolge und Inhalt der Einträge, während der VIndex eine zusätzliche, inhaltsorientierte Sicht auf diese Einträge bietet. Ein bestimmter VIndex-Zustand sollte folglich einem bestimmten Checkpoint von L zugeordnet sein, damit ein Verifizierer genau feststellen kann, welches Log-Präfix der Index darzustellen beansprucht.

Der zweite Parameter, μ, legt fest, wie die Inhalte von L in indexierbare Information umgewandelt werden. Konzeptionell umfasst μ den vollständigen Regelsatz, der nötig ist, um aus einem rohen Log-Blatt ein oder mehrere Index-Updates zu erzeugen. Dazu gehört, wie ein Eintrag ausgewertet wird, wie relevante Felder kanonisiert werden, welche Index-Schlüssel ausgegeben werden und welche Werte diesen Schlüsseln zugeordnet werden. Ein Mapping kann beispielsweise einen Schlüssel mit den Positionen aller passenden Blätter im Input Log verknüpfen. In diesem Fall ist der einem Schlüssel zugeordnete Wert naturgemäß eine geordnete Posting-Liste von Input-Log-Indizes. Ein anderes Mapping könnte andere Felder extrahieren oder zusammengesetzte Schlüssel bilden und so eine andere authentifizierte Sicht über genau dasselbe zugrunde liegende Log erzeugen.

Es ist daher sinnvoll, μ als mehr als die Implementierung einer Mapping-Funktion zu betrachten. Es stellt die semantische Spezifikation des Index dar.

Diese Parametrisierung ist in zwei Richtungen relevant. Erstens kann ein Transparenz-Ökosystem mehrere unabhängige Input Logs enthalten. Wenn $L1$, ..., Lm verschiedene Logs sind und alle dieselbe Mapping-Semantik μ verwenden, ergeben sie unterschiedliche VIndex-Instanzen, $VL1,\mu$, $VL2,\mu$, ..., VLm,μ.

Ihre Positionen in den Posting-Listen sind lokal für das jeweilige Input Log und dürfen nicht global interpretiert werden. Ein Monitoring-Dienst, der das gesamte Ökosystem abdeckt, fragt daher mehrere VIndex-Instanzen ab und kombiniert die verifizierten Ergebnisse auf einer höheren Ebene.

Zweitens kann dasselbe Input Log mehrere VIndexes unterstützen, wenn unterschiedliche Indexierungssemantiken nützlich sind. Für ein Log L erzeugen die Mappings $\mu1$ und $\mu2$ unterschiedliche authentifizierte Sichten, $VL,\mu1$ und $VL,\mu2$.

Für Certificate Transparency könnte beispielsweise ein Mapping Einträge nach kanonischem DNS-Namen indexieren, während ein anderes zusammengesetzte Schlüssel wie Aussteller und DNS-Name verwenden könnte. Beide arbeiten über genau demselben CT-Log, beantworten aber unterschiedliche Klassen von Abfragen und haben daher unterschiedliche Vollständigkeitssemantik.

Das bedeutet auch, dass μ als Teil der Bedeutung einer authentifizierten VIndex-Wurzel behandelt werden muss. Eine Map-Wurzel allein schreibt nur eine Schlüssel-Wert-Struktur fest. Sie erklärt nicht, wie die Quelleinträge in diese Struktur umgewandelt wurden. Damit eine unabhängige Rekonstruktion funktioniert, müssen unterschiedliche Implementierungen dieselben deterministischen Regeln für Auswertung, Kanonisierung und Schlüsselerzeugung anwenden. Die Mapping-Semantik sollte daher stabil, versioniert und identifizierbar sein.

Konzeptionell lässt sich eine veröffentlichte VIndex-Revision so verstehen, dass sie (L, μ, Cn, Rn) aneinander bindet, wobei Cn der Checkpoint des verarbeiteten Input-Log-Präfixes und Rn die zugehörige VMap-Wurzel ist. Das später beschriebene Output Log liefert die dauerhafte Aufzeichnung dieser Bindungen. Ein Verifizierer kann daher nicht nur feststellen, welche Map-Wurzel verwendet wird, sondern auch, welches Input Log, welcher Zustand des Input Logs und welche Indexierungssemantik diese Wurzel repräsentiert.

Nach diesem Modell ist ein VIndex eine authentifizierte, kontinuierlich gepflegte Sicht auf ein bestimmtes Transparenz-Log unter einer bestimmten Mapping-Semantik. Das Paar (L, μ) identifiziert die VIndex-Instanz, während aufeinanderfolgende Wurzeln Rn die Entwicklung derselben Instanz darstellen, während das Quell-Log wächst.

3.3. Architektur des Verifiable Index

Die Architektur überträgt das konzeptionelle VIndex-Modell des vorherigen Abschnitts in ein einsetzbares System. Das Modell identifiziert eine VIndex-Instanz durch ein Input Log L, eine feste Mapping-Semantik μ und eine Folge authentifizierter Index-Zustände, die beim Wachsen des Logs abgeleitet werden. Die hier beschriebene Architektur definiert die Komponenten, die nötig sind, um diese Zustände kontinuierlich zu pflegen, zu veröffentlichen und verifizierbare Abfragen zu bedienen. Abbildung 2 zeigt die Hauptkomponenten.

3.3.1 Input Log

Das Input Log ist das bestehende chronologische Transparenz-Log L . Ein Batch Builder verfolgt seinen zuletzt verarbeiteten Checkpoint, ruft neu angehängte Blätter ab und überprüft, dass sie zum erwarteten Zustand des Logs gehören.

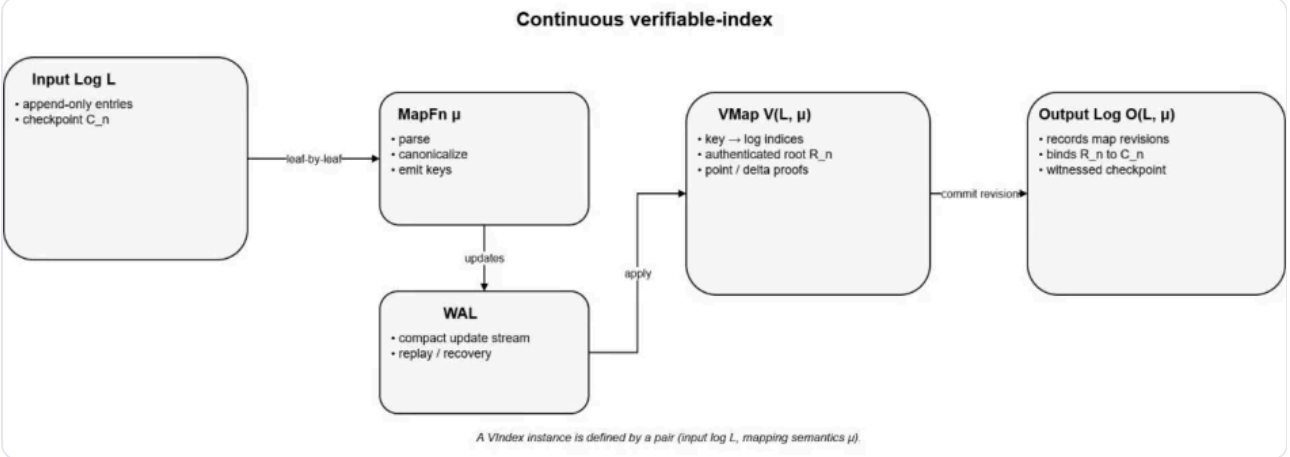


Abbildung 2: VIndex-Architektur

Das Input Log bleibt die Quelle der Wahrheit. Ein Batch Builder beobachtet neue Checkpoints und ruft die Einträge ab, die seit dem zuletzt verarbeiteten Checkpoint hinzugekommen sind. Diese Einträge werden durch die MapFn geleitet, die die Mapping-Semantik μ umsetzt. Die resultierenden Schlüssel-Updates werden in einem Write-Ahead-Map-Transaktionslog (WAL) persistiert und dann auf die Verifiable Map (VMap) angewendet. Periodisch wird die resultierende Map-Wurzel dem Checkpoint des Input Logs zugeordnet, aus dem sie abgeleitet wurde, und als neue Revision im Output Log veröffentlicht.

Jeder Batch ist somit an einen bestimmten Checkpoint Cn gebunden. Der Batch Builder übergibt die validierten Einträge und ihre stabilen Positionen im Quell-Log an die Mapping-Stufe.

Bei Systemen mit mehreren Logs wird jedes Log unabhängig verarbeitet. Die Blattposition i ist nur zusammen mit der Identität ihres Quell-Logs aussagekräftig.

3.3.2 MapFn

Die MapFn setzt die Mapping-Semantik μ um. Sie wandelt einen Eintrag des Quell-Logs deterministisch in null oder mehr Index-Schlüssel um:

$(ei, i) \longrightarrow \{(k1, i), \ldots, (kr, i)\}$.

Diese Stufe umfasst Auswertung, Kanonisierung, Schlüsselextraktion und den Umgang mit nicht unterstützten oder fehlerhaften Einträgen.

Die MapFn getrennt von der authentifizierten Map zu halten ist wichtig: Die Mapping-Semantik bestimmt, was der Index bedeutet, während das Map-Backend bestimmt, wie dieser Zustand authentifiziert und gespeichert wird. Der mitgelieferte Prototyp behandelt deterministische Indexierung ebenfalls als Entwurfsanforderung.

3.3.3 Write-Ahead-Map-Transaktionslog

Die Ausgabe der MapFn wird zuerst in einem Write-Ahead-Map-Transaktionslog (WAL) persistiert. Ein WAL-Datensatz ordnet eine Position im Quell-Log den für diesen Eintrag erzeugten Schlüsseln zu.

Das WAL dient zwei Zwecken. Erstens unterstützt es Wiederherstellung und Neuaufbau, ohne das gesamte Input Log erneut herunterzuladen und auszuwerten. Zweitens schafft es eine saubere Grenze zwischen der Interpretation der Einträge und der Pflege der authentifizierten Map.

In diesem Projekt nutzen wir diese Trennung, um sicherzustellen, dass unterschiedliche authentifizierte Map-Backends dieselben abgeleiteten Updates verarbeiten.

3.3.4 Verifiable Map

Die Verifiable Map speichert die inhaltsorientierte Sicht auf das Log. Logisch ordnet sie einem Index-Schlüssel k eine geordnete Posting-Liste von Positionen im Input Log zu:

$k \longmapsto P(k)$.

Zum Beispiel example . com $\longmapsto$[1842, 7719, 10552, …].

Die authentifizierte Map stellt eine Wurzel Rn bereit, die den vollständigen Index-Zustand nach der Verarbeitung des Logs bis zum Checkpoint Cn festschreibt.

Diese Schicht können verschiedene authentifizierte Datenstrukturen umsetzen. Wir evaluieren Sparse Merkle Trees (SMT), Merkle Patricia Trees (MPT) und IAVL-Backends hinter einer gemeinsamen VIndex-Abstraktion.

3.3.5 Output Log

Die VMap-Wurzel allein liefert keine öffentliche Historie der Index-Zustände. Das Output Log zeichnet daher die veröffentlichten VIndex-Revisionen auf.

Eine Revision bindet konzeptionell (L, μ, Cn, Rn). Das bedeutet, dass die Wurzel Rn das Input Log L unter der Mapping-Semantik μ nach der Verarbeitung des durch den Checkpoint Cn identifizierten Präfixes darstellt.

Das Output Log liefert die historische Rechenschaft für diese Map-Zustände und kann selbst durch Witnesses bestätigt werden. Der VIndex kann kontinuierlich aktualisiert werden, während Revisionen periodisch veröffentlicht werden, statt nach jedem einzelnen Blatt.

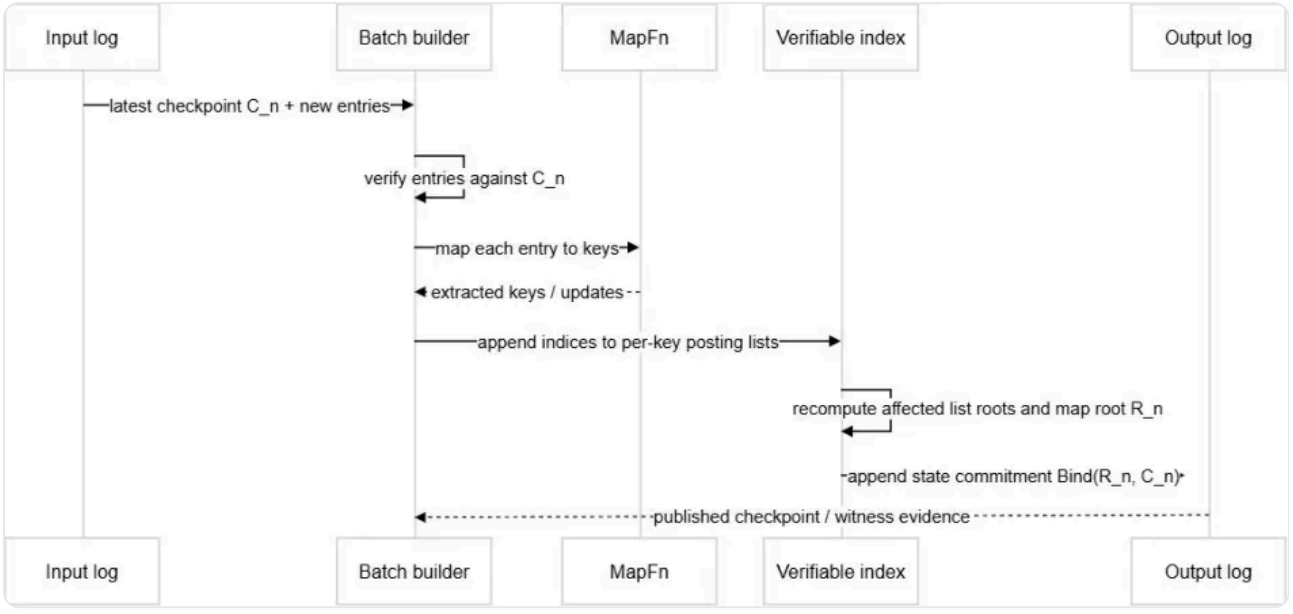


Abbildung 3: Sequenzdiagramm des Verifiable Index

4. Verifiable Index für Certificate Transparency

Certificate Transparency (CT) ist ein besonders relevanter Anwendungsfall für verifizierbare Indexierung. CT arbeitet in sehr großem Maßstab, stützt sich auf mehrere unabhängige Logs und verlangt von Monitoren die Verarbeitung komplexer X.509-Objekte mit mehreren Namen, Wildcard-Einträgen, Vorzertifikaten und wiederholten Beobachtungen über Logs hinweg. Zugleich ist die zentrale Monitoring-Frage naturgemäß inhaltsbasiert: Ein Domaininhaber möchte wissen, welche geloggten Zertifikate Namen betreffen, die unter seiner Kontrolle stehen.

CT lässt sich als Sammlung unabhängig indexierter Logs modellieren und nicht als eine einzige globale authentifizierte Map. Jedes CT-Log Li hat seine eigene Append-Only-Historie und damit seine eigene VIndex-Instanz, VLi,μ.

Das bewahrt die unabhängige Checkpoint-Folge jedes CT-Logs und vermeidet eine künstliche Ordnung zwischen Einträgen verschiedener Logs.

Ein CT-Monitoring-Dienst kann dann als Aggregationsschicht arbeiten. Für eine Domainabfrage fordert er von jedem abgedeckten VIndex ein verifiziertes Ergebnis an, überprüft die zugehörige Nachweiskette und führt die zurückgegebenen CT-Einträge zu einer kombinierten Sicht zusammen. Die Vollständigkeit bezieht sich daher auf die Menge der vom Monitor abgedeckten CT-Logs und auf die akzeptierten Checkpoints dieser Logs. Der Monitor kann nachweisen, dass er alle passenden Einträge aus den abgefragten VIndexes zurückgegeben hat, dies sollte aber nicht automatisch als Vollständigkeit in Bezug auf jedes mögliche CT-Log verstanden werden.

Mapping

Der direkteste Entwurf eines CT-Index wäre, DNS-Namen den Positionen der Zertifikats- oder Vorzertifikatseinträge zuzuordnen, die diese Namen enthalten. Zum Beispiel example . com ⟼[1250, 4821, 9317, …].

Die zugehörige MapFn wertet jedes CT-Blatt aus, extrahiert die relevanten DNS-Namen, kanonisiert sie und gibt für jeden Namen einen Schlüssel aus. Die Kanonisierungsregeln sollten ausdrücklich als Teil der Mapping-Semantik μ definiert werden. Im CT-Kontext gehört dazu, wie DNS-Namen hinsichtlich Groß- und Kleinschreibung und abschließender Punkte normalisiert werden, wie internationalisierte Domainnamen dargestellt werden und wie Wildcard-Namen behandelt werden. Die Spezifikation sollte außerdem festlegen, welche Zertifikatsfelder Namen zum Index beitragen, zum Beispiel, ob nur die Erweiterung Subject Alternative Name berücksichtigt wird oder auch ältere Common-Name-Felder einbezogen werden.

Die Darstellung des Index-Schlüssels beeinflusst auch, welche Art von Abfragen der VIndex unterstützen kann. Ein gehashter Schlüssel für den exakten Namen eignet sich gut für direkte Lookups und passt natürlich zu dünn besetzten (Sparse) authentifizierten Maps. Er unterstützt jedoch keine direkte Aufzählung von Suffixen oder Teilbäumen. Eine strukturierte Darstellung, etwa mit umgekehrten DNS-Labels, könnte Abfragen über Subdomains leichter unterstützen, würde aber ausdrücklichere Semantik für Ordnung und Bereichsnachweise erfordern.

Der natürliche Wert zu einem CT-Schlüssel ist eine geordnete Liste von Positionen im Quell-Log. Der VIndex sollte auf das CT-Log zurückverweisen, statt das vollständige Zertifikatsobjekt in der authentifizierten Map zu duplizieren. Für ein einzelnes CT-Log lässt sich eine Posting-Liste daher als $P(k)$ = [i1, i2, …, in] darstellen.

Duplikate und verwandte CT-Einträge

CT führt mehrere Formen von Duplikaten ein, die unterschieden und nicht innerhalb des authentifizierten Index zusammengelegt werden sollten.

Dasselbe Zertifikat kann in mehreren CT-Logs vorhanden sein. Ein Vorzertifikat und das zugehörige endgültige Zertifikat sind verwandt, stellen aber unterschiedliche CT-Einträge dar. Ein Zertifikat mit mehreren DNS-Namen erscheint berechtigterweise unter mehreren Index-Schlüsseln.

Aus diesem Grund sollte der VIndex nah an der Darstellung des Quell-Logs bleiben und jeden passenden Quelleintrag indexieren. Die Deduplizierung wird besser von der Monitoring-Schicht vorgenommen, wo die Richtlinie ausdrücklich festgelegt werden kann.

Eine Monitoring-Oberfläche kann beispielsweise dasselbe DER-Zertifikat, das in drei CT-Logs beobachtet wurde, als ein logisches Zertifikat anzeigen und dabei die drei zugrunde liegenden Log-Verweise beibehalten. Ebenso können ein Vorzertifikat und das endgültige Zertifikat für die Darstellung gruppiert werden, ohne eine der beiden authentifizierten Beobachtungen zu entfernen.

Als Entwurfsprinzip gilt: Der VIndex sollte die Belege des Quell-Logs bewahren, während richtlinienabhängige Gruppierung und Deduplizierung oberhalb des authentifizierten Index erfolgen sollten.

Alternative VIndex-Sichten

Ein einzelner Index für exakte Domains unterstützt möglicherweise nicht jeden CT-Monitoring-Anwendungsfall. Da ein VIndex sowohl durch das Quell-Log als auch durch die Mapping-Semantik parametrisiert wird, können bei Bedarf zusätzliche authentifizierte Sichten über demselben CT-Log aufgebaut werden.

Mögliche Beispiele sind:

VL, μ domain, VL, μ issuer+domain, VL, μ SPKI oder ein strukturierter Domain-Index, der für Suffix-Abfragen optimiert ist.

Diese sollten als optionale Sichten verstanden werden und nicht als Anforderung, jeden möglichen Index zu pflegen. Jeder zusätzliche VIndex erhöht Aufbau-, Speicher- und Betriebskosten.

Suche und Monitoring

Bei der interaktiven Suche übermittelt der Client eine Domain oder einen anderen unterstützten Schlüssel und erhält von jedem relevanten VIndex eine Antwort mit Nachweis. Da die Anfrage vom Nutzer ausgelöst wird, sind Fehler sofort sichtbar: Der Dienst kann nicht erreichbar sein, der Nachweis kann die Überprüfung nicht bestehen, oder die Antwort kann an einen unerwartet alten Checkpoint gebunden sein.

Die Suche nach exakten Namen ist daher der einfachste Anwendungsfall für einen CT-VIndex.

Die Benachrichtigung erfordert einen zusätzlichen Mechanismus für die Verfügbarkeit. Ein Domaininhaber erwartet möglicherweise nur dann eine Benachrichtigung, wenn ein neues passendes Zertifikat erscheint, aber Schweigen ist mehrdeutig: Es kann bedeuten, dass nichts Neues geloggt wurde oder dass der Monitoring-Dienst ausgefallen ist.

Ein praktischer VIndex-basierter Monitor kann zwei Funktionen trennen: (i) Ereignisbenachrichtigungen, die gesendet werden, wenn neue passende Einträge erscheinen, und (ii) periodische Heartbeats mit Nachweis, die auch dann gesendet werden, wenn kein neuer passender Eintrag existiert. Der Client speichert seinen zuletzt akzeptierten Zustand. Eine Ereignisbenachrichtigung enthält die neuen passenden Einträge

zusammen mit Belegen, die sie an die aktuelle VIndex-Revision binden. Ein periodischer Heartbeat bestätigt, dass der Monitor zu einem neueren authentifizierten Zustand fortgeschritten ist, auch wenn keine neuen Einträge zum Schlüssel des Abonnenten passen.

Trifft der erwartete Heartbeat nicht innerhalb des vereinbarten Dienstintervalls ein, kann der Client den Monitoring-Dienst aktiv abfragen, einen anderen VIndex-Anbieter abfragen oder auf die direkte Prüfung der zugrunde liegenden CT-Logs zurückgreifen.

Insgesamt wirft die CT-Umsetzung auch mehrere Fragen auf, die bei einfacheren Transparenzsystemen weniger stark auftreten. Ein Einsatz muss festlegen, wie DNS-Namen kanonisiert werden sollen, wie Wildcard- und Subdomain-Abfragen dargestellt werden sollen, wie Vorzertifikate und endgültige Zertifikate auf der Monitoring-Ebene zueinander in Beziehung gesetzt werden sollen und wie verifizierte Ergebnisse aus mehreren Logs zusammengeführt werden sollen.

Diese Fragen sind wichtig, weil sie nicht nur die Leistung betreffen, sondern auch die Bedeutung eines „vollständigen“ Monitoring-Ergebnisses. Sie bilden daher einen zentralen Teil der Evaluierung und der offenen Forschungsfragen.

5. Evaluierung

Wir evaluieren den VIndex-Prototyp, um den Zielkonflikt zwischen schnelleren inhaltsbasierten Abfragen und dem zusätzlichen Aufwand für die Pflege eines authentifizierten Index zu beziffern. Die Experimente vergleichen ein herkömmliches Append-Only-Transparenz-Log mit drei VIndex-Backends: einem Sparse Merkle Tree (SMT), einem Merkle Patricia Trie (MPT) und IAVL. Alle VIndex-Backends verarbeiten dieselben abgeleiteten Schlüssel-Wert-Updates, sodass die Messungen vor allem Unterschiede zwischen den authentifizierten Datenstrukturen widerspiegeln und nicht Unterschiede bei Auswertung oder Datenaufbereitung.

Die Experimente verwenden reale Go-Modul-Daten aus SumDB, um Append-Only-Logs mit 2^16 bis 2^24 Blättern aufzubauen, und vergleichen einen Baseline-Log-Scan mit authentifizierten Indizes auf Basis von SMT, MPT und IAVL.

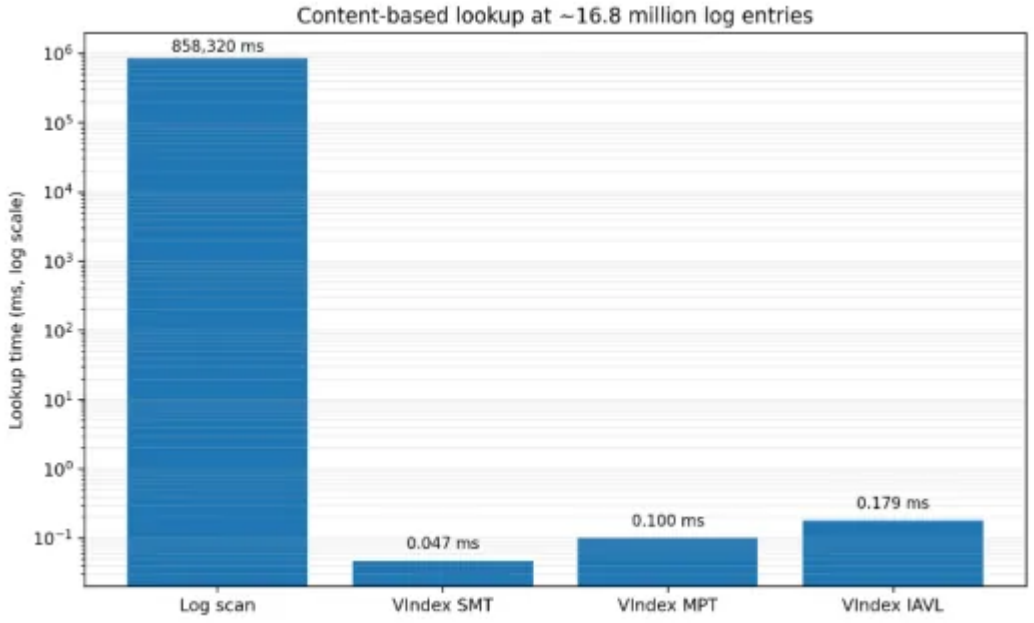
Die Kosten für WAL und VIndex-Aufbau werden getrennt gemessen.

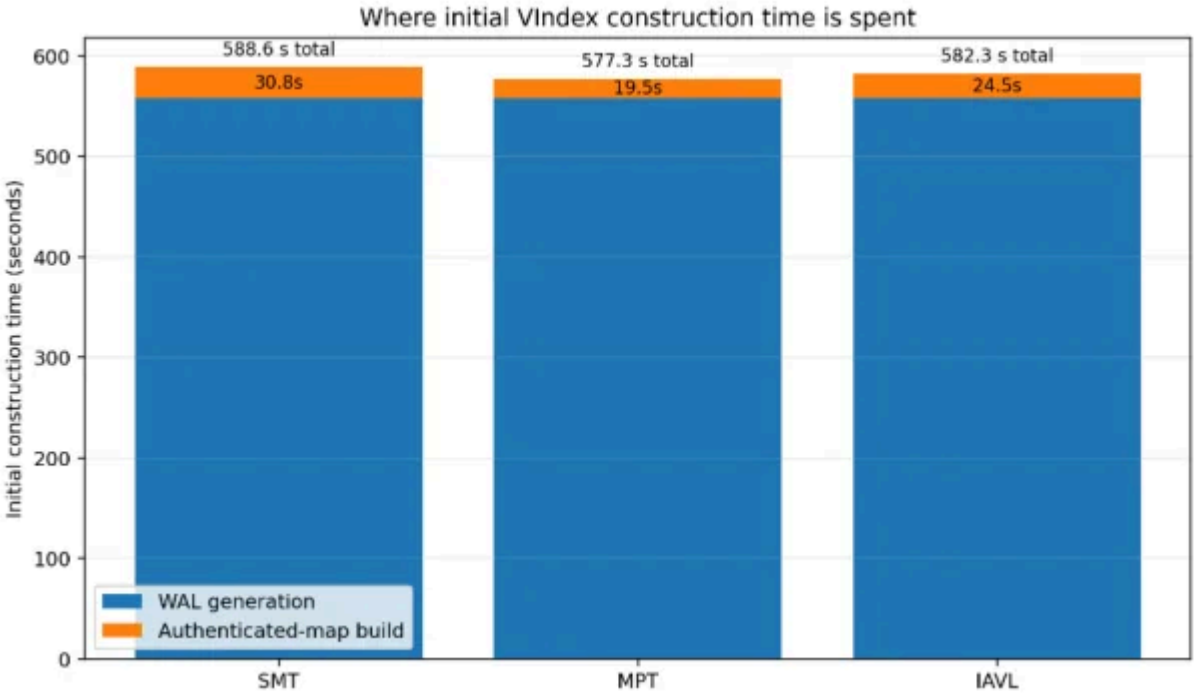
Benchmark	Berichtetes Ergebnis
Inhaltsbasierte Abfrage	Baseline-Scan: 858,32 s; SMT: 0,047 ms; MPT: 0,100 ms; IAVL: 0,179 ms
Inclusion-Proof-Ablauf	Baseline: 4,446 ms; bestes VIndex-Ergebnis: 3,874 ms
Aufbau des authentifizierten Index	MPT: 19,53 s; IAVL: 24,51 s; SMT: 30,84 s
Vollständige WAL-Erzeugung	Etwa 558 s
Speicherbedarf des Index	SMT: 1,54 GB; MPT: 2,60 GB; IAVL: 2,94 GB
WAL-Speicher	Etwa 2,06 GB

Die Benchmark-Ergebnisse zeigen, dass der Hauptnutzen des VIndex bei inhaltsbasierten Aggregationsabfragen liegt. Im Baseline-Entwurf erfordert der Abruf aller Einträge zu einem bestimmten Modul das Durchsuchen des chronologischen Logs. Bei etwa 16 Millionen Blättern dauert dieser Vorgang rund 858,32 Sekunden. Dieselbe Abfrage über den VIndex dauert dagegen 0,047 ms mit SMT, 0,100 ms mit MPT und 0,179 ms mit IAVL. Das entspricht einer Leistungsverbesserung um mehrere Größenordnungen und zeigt, dass der VIndex besonders wirksam ist, wenn Abfragen sonst einen großen Teil des Quell-Logs durchlaufen müssten.

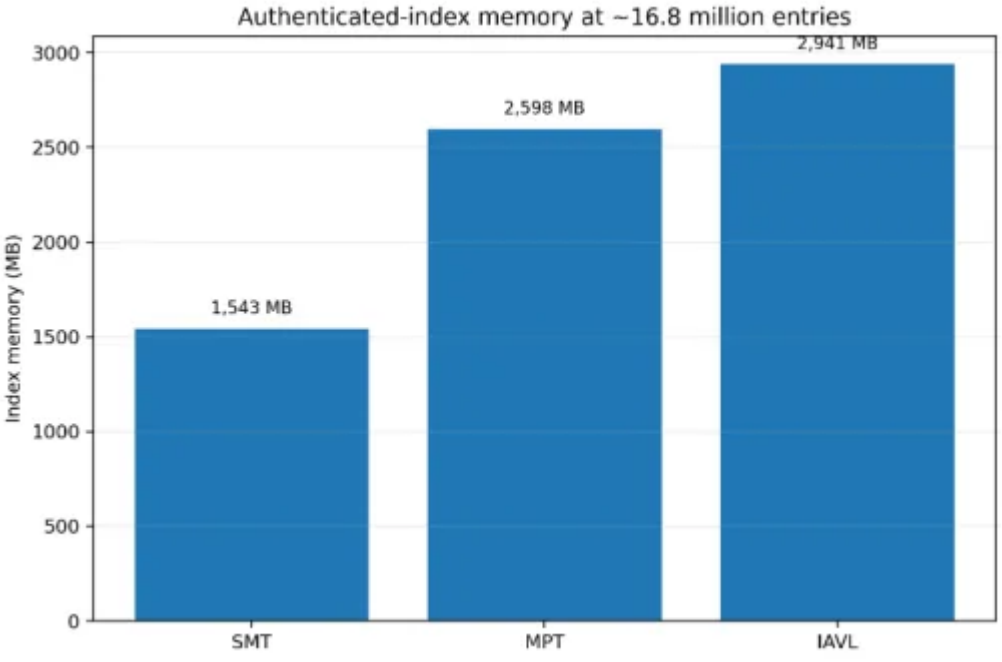
Die Verbesserung fällt bei Standard-Inclusion-Proofs deutlich geringer aus. Bei der größten evaluierten Baumgröße benötigt der Baseline-Nachweisablauf etwa 4,45 ms, verglichen mit etwa 3,87 ms für die leistungsstärkste VIndex-Konfiguration. MPT und IAVL zeigen ähnliche Ergebnisse. Das entspricht der beabsichtigten Rolle des VIndex: Sobald die Position eines relevanten Blatts bereits bekannt ist, bleibt der zugrunde liegende Merkle-Nachweis effizient. Der VIndex senkt vor allem den Aufwand, überhaupt erst das relevante Blatt oder die relevante Menge von Blättern zu finden.

Diese Verbesserungen bei der Abfrage gehen mit zusätzlichem Aufwand für Aufbau und Pflege des Index einher. Für den größten Datensatz benötigt die Erzeugung des vollständigen WAL etwa 558 Sekunden. Der Aufbau des authentifizierten Index aus dem WAL dauert dann etwa 30,84 Sekunden für SMT, 19,53 Sekunden für MPT und 24,51 Sekunden für IAVL. Unter den evaluierten Backends bietet MPT somit die schnellste Aufbauzeit, während die anderen Backends unterschiedliche Kompromisse bei Speicherbedarf und Abfrageverhalten zeigen.





Der VIndex bringt außerdem zusätzliche Ressourcenanforderungen mit sich. Bei etwa 16 Millionen Einträgen belegt der authentifizierte Index rund 1,54 GB mit SMT, 2,60 GB mit MPT und 2,94 GB mit IAVL, während das zugehörige WAL etwa 2,06 GB belegt. Diese Messungen zeigen, dass der VIndex einen Teil der Systemkosten von der Abfragezeit auf den kontinuierlichen Indexaufbau und die Speicherung verlagert.



Insgesamt stützen die Benchmark-Ergebnisse den vorgesehenen Einsatz des VIndex als zusätzliche Abfrageschicht über einem chronologischen Transparenz-Log. Sein Nutzen ist begrenzt bei Operationen, die im zugrunde liegenden Log bereits effizient sind, etwa Inclusion Proofs für bekannte Blattpositionen. Die größten Gewinne ergeben sich bei wiederholten inhaltsbasierten Abfragen, die sonst ein Durchsuchen des

Logs erfordern würden. Für Monitoring-Lasten ist dieser Zielkonflikt besonders relevant: Das System zahlt laufende Kosten für die Pflege des Index, vermeidet aber wiederholte vollständige Log-Scans, wenn dieselben oder ähnliche Prädikate ausgewertet werden, während das Log wächst.

6. Schlussfolgerungen:

Chronologische Transparenz-Logs lösen ein wichtiges Integritätsproblem: Sie machen veröffentlichte Ereignisse manipulationssicher erkennbar und im Zeitverlauf überprüfbar. Monitoring führt ein anderes Problem ein: jeden für einen Stakeholder relevanten Eintrag zu finden. Heute wird diese Fähigkeit normalerweise von vertrauenswürdigen Indizes Dritter geliefert. Verifizierbare Indexierung zielt darauf ab, die Effizienz dieser Indizes zu bewahren und die Transparenzkette zugleich durch die Abfrageschicht zu verlängern.

Das nützlichste Architekturmodell ist eine Log-Map-Log-Architektur pro Instanz. Ein Input Log bleibt maßgeblich, feste Mapping-Semantik μ leitet Schlüssel und Update-Operationen ab, ein kontinuierlich gepflegter Verifiable Index authentifiziert Posting-Listen von Schlüsseln zu Positionen, und ein Output Log hält Map-Revisionen fest, die an Checkpoints des Quell-Logs gebunden sind.

Die Evaluierungsergebnisse liefern starke Hinweise, dass der Zielkonflikt auf Systemebene eine weitere Untersuchung wert ist: Authentifizierte Indizes können wiederholte Aggregations-Scans um viele Größenordnungen verringern, wenngleich Aufnahme, Mapping, Speicherung und Indexaufbau nicht kostenlos sind. Die weitere Forschung betrifft daher, die Semantik reproduzierbar zu machen, die Wertstrukturen inkrementell zu gestalten, die schichtübergreifende Entsprechung effizient zu machen und das Monitoring-Erlebnis, einschließlich der Benachrichtigung, Ende-zu-Ende verifizierbar zu machen.

Referenzen

[1] “Monitors : Certificate Transparency.” Accessed: Sep. 01, 2026. [Online]. Available: https://certificate.transparency.dev/monitors/

[2] “Certificate Transparency Revisited: The Public Inspections on Third-party Monitors - NDSS Symposium.” Accessed: Sep. 01, 2026. [Online]. Available: https://www.ndsssymposium.org/ndss-paper/certificate-transparency-revisited-the-public-inspections-onthird-party-monitors/

[3] “incubator/vindex at main · transparency-dev/incubator · GitHub.” Accessed: Jun. 09, 2026. [Online]. Available: https://github.com/transparency-dev/incubator/tree/main/vindex

[4] “Certificate Transparency : Certificate Transparency.” Accessed: Nov. 11, 2024. [Online]. Available: https://certificate.transparency.dev/

[5] “Proposal: Secure the Public Go Module Ecosystem.” Accessed: Sep. 01, 2026. [Online]. Available: https://go.googlesource.com/proposal/+/master/design/25530-sumdb.md

[6] “Deploying key transparency at WhatsApp - Engineering at Meta.” Accessed: Sep. 01, 2026. [Online]. Available: https://engineering.fb.com/2023/04/13/security/whatsapp-keytransparency/

[7] “GitHub - sigstore/rekor: Software Supply Chain Transparency Log · GitHub.” Accessed: Sep. 01, 2026. [Online]. Available: https://github.com/sigstore/rekor

[8] M. S. Melara, A. Blankstein, P. University, J. Bonneau, E. W. Felten, and M. J. Freedman, “Open access to the Proceedings of the 24th USENIX Security Symposium is sponsored by USENIX CONIKS: Bringing Key Transparency to End Users CONIKS: Bringing Key Transparency to End Users,” p. 383, Accessed: Sep. 01, 2026. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technicalsessions/presentation/melara

[9] M. Chase, A. Deshpande, E. Ghosh, and H. Malvai, “SEEMless: Secure End-to-End Encrypted Messaging with less trust,” *Cryptology ePrint Archive*, 2018, doi: 10.1145/3319535.3363202.

[10] B. Laurie, E. Messeri, R. Stradling, B. Laurie, E. Messeri, and R. Stradling, “Certificate Transparency Version 2.0,” Dec. 2021, doi: 10.17487/RFC9162.