Great—let's pin this down in math-tight, spec-ready terms using your **Engine / Verifier** split. I'll give you: formal model, constructions (incl. two non-reachability options), receipt format, verification equations, security goals, and drawing-ready figure list + captions. Minimal prose, maximal precision.

## 1) Formal model

Let:

- Directed, labeled multigraph $G = (V, E, \lambda_V, \lambda_E)$ with
  - $E \subseteq V \times \Sigma \times V$, edge label alphabet $\Sigma$.
  - $\lambda_V : V \to \{0,1\}^*$, $\lambda_E : E \to \{0,1\}^*$ (canonical encodings).
- Policy/path spec as a **finite automaton** $A = (Q, \Sigma, \delta, q_0, F)$ (DFA/NFA; regex compiles to DFA).
- **Relational reachability** under policy $A$: for $s, t \in V$,

$$\mathrm{Reach}_A(s,t) \ := \ \exists \,\mathrm{path}\ s \xrightarrow{w} t \ \mathrm{in}\ G \ \mathrm{with}\ w \in L(A).$$

Define **product graph** $H = V \times Q$ with edges

$$(u,q) \to (v,q') \iff \exists a \in \Sigma \ \mathrm{s.t.}\ (u,a,v) \in E \wedge q' = \delta(q,a).$$

Then $\mathrm{Reach}_A(s,t)$ iff $\exists q_f \in F$ with $(t, q_f) \in \mathrm{Post}^*\big(\{(s,q_0)\}, H\big)$.

## 2) Commitments and hashing

Let $H(\cdot)$ be a collision-resistant hash (e.g., SHA-256). Canonical element encodings:

- $\langle v \rangle :=$ CBOR$(\lambda_V(v))$,
- $\langle e \rangle :=$ CBOR$(u, a, v, \lambda_E(e))$,
- $\langle u, a, v \rangle$ canonical triple.

### 2.1 Dataset commitment (Engine)

Build a key-sorted **sparse Merkle map** (SMM) over:

- Node set $V$: keys $k_V = H(\texttt{"V"} \parallel \langle v \rangle)$, values $1$.
- Edge set $E$: keys $k_E = H(\texttt{"E"} \parallel \langle u,a,v \rangle)$, values $H(\lambda_E(e))$.
  Let $R_V, R_E$ be the roots. Define dataset root:

$$R_D = H(\texttt{"D"} \parallel R_V \parallel R_E).$$

### 2.2 Policy commitment

$$R_A = H\big(\texttt{"A"} \parallel \mathrm{CBOR}(Q, \Sigma, \delta, q_0, F)\big).$$

## 3) Receipts

A receipt proves statement $S = (s, t, A, \text{mode})$ at time $\tau$ relative to dataset commitment $R_D$.

### 3.1 Receipt structure (common)

cpp

Copy code

```cpp
Receipt = {
  version,                                // e.g., 1
  statement: {s_id, t_id, A_id, mode},    // ids or encodings
  RD, RA,                                  // commitments
  anchor: TSA_proof,                       // RFC 3161 blob or dual anchors
  witness: W,                              // differs by mode
  sig: Sig_SK( H( canonicalize(all_above) ) )
}
```

Engine signs with keypair $(\mathsf{SK}, \mathsf{PK})$; Verifier knows $\mathsf{PK}$.

### 3.2 Reachability witness $W^{\text{reach}}$

A **path witness:**

- Node/edge sequence $(v_0 = s, a_1, v_1, \ldots, a_m, v_m = t)$.
- For each used edge $(v_{i-1}, a_i, v_i)$: SMM inclusion proof $\pi_E^i$ w.r.t. root $R_E$.
- DFA trace $q_0 \xrightarrow{a_1} q_1 \cdots \xrightarrow{a_m} q_m$ with $q_m \in F$.

**Verifier checks:**

1. $\forall i$, $\mathsf{SMM.VerifyInclude}(R_E, k_E^i, H(\lambda_E(e_i)), \pi_E^i) = \mathsf{true}$.
2. $q_i = \delta(q_{i-1}, a_i)$ and $q_m \in F$.
3. Signature and anchor validate.

### 3.3 Non-reachability witnesses $W^{\text{nreach}}$ (two constructions)

#### (A) Product-state sparse map (PSSM) + cut certificate (practical, patentable)

Engine computes **closure bitmap** $B : V \times Q \to \{0,1\}$ for reachable product-states starting from $(s, q_0)$. Builds an SMM over keys
$k_H = H(\texttt{"H"} \parallel \langle v \rangle \parallel q)$ with value $B(v, q)$. Root $R_H$.
To avoid "trust me", include a **local cut certificate** $C$ consisting of a minimal **frontier set** $F^\star \subseteq V \times Q$ such that:

- Every path from $(s, q_0)$ to any $(t, q_f)$ must pass through $F^\star$.
- All $(x, q) \in F^\star$ are **closed**: for every outgoing transition $(x, q) \to (y, q')$ permitted by $E$ and $\delta$, we have $B(y, q') = 0$.

**Witness** $W^{\text{nreach}} = (R_H, \Pi_0, C, \Pi_C)$:

- $\Pi_0$: SMM non-inclusion proof for the target states: for each $q_f \in F$, SMM.VerifyZero$(R_H, k_H(t, q_f), \pi_{t,q_f})$.
- $C = \{(x_j, q_j)\}_{j=1}^{r}$ with:
  ♦ Inclusion proofs $\pi_{x_j,q_j}$ showing $B(x_j, q_j) = 1$.
  ♦ For each outgoing edge $(x_j, a, y)$ with $q' = \delta(q_j, a)$: proofs $\pi_{y,q'}$ showing $B(y, q') = 0$ (or, alternately, proofs of edge absence from $R_E$ if policy forbids it).
- $\Pi_C$: SMM proofs supporting the above claims.

**Verifier checks:**
♦ Non-inclusion at $(t, q_f)$ for all $q_f \in F$.
♦ Each frontier state is marked 1; each of its policy-consistent successors is 0 (or edge absent).
By **frontier minimality**, no accepting target is reachable ⇒ non-reachability holds.

*Note*: Frontier minimality can be encoded as: each $(x_j, q_j)$ has at least one predecessor in $B = 1$ (prove inclusion) and all successors are 0; the union of these cuts all $s \to F$ paths (Engine ensures; Verifier checks the local conditions).

#### (B) Edge-cut certificate over $G$ (small-footprint variant)

Provide a **cut set of edges** $K \subseteq E$ such that every $A$-conforming $s \to t$ path uses some edge in $K$.
Witness includes:

- Inclusion proofs that all edges in $K$ exist in $R_E$.
- Proofs (policy side) that removing $K$ destroys all automaton-conforming paths (encode each edge's label position in the automaton; Verifier simulates supergraph with edges in $K$ removed to confirm no residual accepting path exists using small "blocking lemmas" tied to $A$).
  This variant trades more policy algebra for fewer SMM proofs; include if you want a second embodiment.

### 4) Verification equations (succinct)

- **Dataset root**: $R_D \stackrel{?}{=} H(\texttt{"D"} \parallel R_V \parallel R_E)$.
- **Policy root**: $R_A \stackrel{?}{=} H(\texttt{"A"} \parallel \mathsf{CBOR}(A))$.
- **Anchor**: $\mathsf{RFC3161.Verify}(R_D \parallel R_A, \mathrm{TSA\_proof}) = \mathbf{true}$.
- **Signature**: $\mathsf{VRF} = H(\text{canonical Receipt minus sig}); \mathsf{Sig.Verify}(\mathsf{PK}, \mathsf{VRF}, \mathrm{sig}) = \mathbf{true}$.
- **Reachability**: $\bigwedge_i \mathsf{SMM.Incl}(R_E, k_E^i, H(\lambda_E(e_i)), \pi_E^i)$ and DFA trace acceptance.
- **Non-reachability (PSSM)**:

$$[\forall q_f \in F : \ \mathsf{SMM.Zero}(R_H, k_H(t, q_f), \pi)] \ \wedge \bigwedge_{(x,q)\in C} (\mathsf{SMM.Incl}(R_H, k_H(x,q), 1, \pi) \wedge \forall (x,q) \to (y,q') : \mathsf{SMM.Zero}(R_H, k_H(y,q'), \pi))\,.$$

## 5) Security goals

Assume collision-resistant $H$, EUF-CMA signature, and sound RFC-3161.

♦ **Receipt Unforgeability:** No PPT adversary can produce a valid Receipt asserting a false statement $S$ w.r.t. $(R_D, R_A)$ unless it:

- finds a Merkle collision (breaks CRH),
- forges signature,
- forges TSA proof.

♦ **Soundness (reachability):** A false positive requires including a non-existent edge (Merkle collision) or mis-simulating $A$.

♦ **Soundness (non-reachability)**: A false non-reach receipt requires either:

- marking an actually-reachable product-state as 0 (Merkle collision), or
- presenting a frontier $C$ where some successor is actually 1 (contradicted by inclusion proofs), or
- eliding an existing successor edge (contradicted by $R_E$ inclusion).

♦ **Completeness**: If statement is true, Engine can construct witnesses (path or valid frontier/cut) in $O(m \log n)$.

## 6) Complexity

Let $n = |V|, m = |E|, |Q| = q$.

♦ Path witness size: $O(\ell \cdot \log N)$ hashes, where $\ell$ is path length and $N$ SMM size. Verify time $O(\ell \log N)$.

♦ Non-reach (PSSM): frontier size $r$, out-degree $\Delta$: witness includes $r$ inclusions + $r\Delta$ zero-proofs ⇒ size $O((r + r\Delta) \log N)$. In many RBAC/data-lineage graphs, $r \ll n$.

♦ Microsecond-level verification on commodity CPUs ( $20$ SHA-256 for $\log_2 N \approx 20$).

## 7) Engine / Verifier separation (math view)

**Engine**

♦ Builds $R_V, R_E, R_D$; compiles $A \to R_A$.

♦ Reachability: extracts path $\pi$ + inclusion proofs.

♦ Non-reachability: computes closure $B$ over $H$; derives minimal frontier $C$; constructs $R_H$ and the proof set $\Pi$.

♦ Produces Receipt and $\mathsf{Sig}_{\mathsf{SK}}$.

**Verifier**

♦ Recomputes no closures; it **only**:

- checks Merkle (inclusion/zero) proofs against roots,
- checks DFA steps on provided symbols,
- checks TSA + signature.

## 8) Pseudocode (spec-ready)

pseudo

Copy code

```
function Engine.GenerateReceipt(G, A, s, t, mode, TSA, SK):
  (RV, RE) := SMM.BuildNodesEdges(G)
  RD := H("D" || RV || RE)
  RA := H("A" || CBOR(A))
  if mode == "reach":
     path := FindPath_A(G, A, s, t)         // any A-conforming path
     W := []
     for each edge e=(u,a,v) in path:
        k := H("E" || <u,a,v>)
        W.append( SMM.ProveInclude(RE, k, H(lambdaE(e))) )
     witness := {path_labels, W}
  else: // "nreach"
     B := ClosureBitmap(G, A, s)             // over product graph H
     (RH, MapH) := SMM.BuildBitmap(B)
     C := Frontier(B, A)                     // minimal frontier cut
     witness := {RH,
                 ZeroProofsForTargets(MapH, t, A.F),
                 ProofsForFrontier(MapH, C),
                 ZeroProofsForSuccessors(MapH, C, G, A) }
  anchor := TSA.Timestamp( RD || RA )
  payload := canonicalize(version, stmt, RD, RA, anchor, witness)
  sig := Sign(SK, H(payload))
  return Receipt(payload, sig)
```

```pseudo
function Verifier.Verify(Receipt, PK):
  parse (version, stmt, RD, RA, anchor, witness, sig)
  assert SigVerify(PK, H(canonicalize(...)), sig)
  assert RFC3161.Verify(RD || RA, anchor)
  if stmt.mode == "reach":
     for each edge proof in witness:
        assert SMM.VerifyInclude(RE_from_RD, key, val, proof)
     assert DFA_Accepts(stmt.A, witness.path_labels)
     return True
  else: // "nreach"
     assert forall qf in A.F:
        SMM.VerifyZero(RH_from_witness, k_H(t,qf), π)
     for each (x,q) in C:
        assert SMM.VerifyInclude(RH, k_H(x,q), 1, π)
        for each policy-consistent successor (y,q'):
           assert SMM.VerifyZero(RH, k_H(y,q'), π)
     return True
```

## 9) Figures (patent drawing set)

- ♦ **FIG. 1** System context: data sources → Engine → Receipt → Verifier/Auditor.
- ♦ **FIG. 2** Dataset commitment: (a) node SMM $R_V$; (b) edge SMM $R_E$; (c) combined $R_D$.
- ♦ **FIG. 3** Reachability receipt: path witness with per-edge Merkle proofs into $R_E$; DFA trace overlay.
- ♦ **FIG. 4** Product graph $H = V \times Q$: closure bitmap $B$; minimal frontier $C$; SMM $R_H$.
- ♦ **FIG. 5** Non-reachability verification flow: (i) zero-proofs at $(t, q_f)$; (ii) frontier inclusions; (iii) successor zero-proofs.
- ♦ **FIG. 6** Anchoring timeline: $R_D \| R_A$ → TSA stamp; receipt signing; audit timeline.
- ♦ **FIG. 7** Alternative embodiment: edge-cut certificate $K$ with inclusion proofs and automaton blocking lemmas.
- ♦ **FIG. 8** Receipt structure exploded view (fields, sizes, canonicalization order).

## 10) Claims-adjacent phrasing (for your spec)

- "A method for generating a cryptographic receipt certifying relational non-reachability, comprising: committing to a dataset with a collision-resistant Merkle map; compiling a path policy to a finite automaton; constructing a product-state closure over a graph of nodes and automaton states; deriving a frontier set that intercepts all policy-conforming paths; and producing Merkle inclusion and non-inclusion proofs sufficient for independent verification without recomputing the closure."
- "A verifier configured to validate said receipt by checking only commitment-consistent Merkle proofs, automaton transitions on provided labels, and an external timestamp, wherein no access to the underlying dataset is required."