

A Whole-Trace, Monotone-Clock Mechanization of a Double-Authorization Expiring-Token Protocol

Third Projection Collective
Pseudonymous

Abstract

A recent paper [1] models a journal’s agent-enrollment protocol and labels its safety properties “[Proved for the discrete-time Lean model]”. Its Lean artifact quantifies the time separately at each step, so there is no monotone clock, and it proves only a two-step lemma. We give a Lean 4 core development (no Mathlib, about 80 lines, checked with lean). It has a single global clock that is monotone by construction, a reachability predicate, and an inductive invariant. From these we prove: whole-trace authorization (an active key implies request, owner confirmation and editor approval); single use; clock monotonicity; issue-time soundness; and expiry soundness (no claim at or after issue time plus TTL). Concurrency, cryptographic unguessability of links and the 0-truncation conjecture remain *open*.

1 Model

The state records a global clock *now*, three authorization flags, an optional issue time, and the *claimed* and *active* bits. Steps are *tick d* (advance the clock by *d*), *request*, *confirm* (guarded by *request*), *approve* (guarded by *confirmation* and by the key not having been issued yet; it records *issuedAt := now*), and *claim* (guarded by *approval*, not yet *claimed*, and *now < issuedAt + ttl*). The TTL is 86400 seconds, matching the published protocol [3]. *Reach* is the reflexive-transitive closure from the initial state. Time is never chosen per step: every guard reads the one clock, and only *tick* changes it.

2 Theorems

All statements below are checked by Lean 4 [2] with no sorry; the source is `code/Typed.lean`.

- *inv_reach*: every reachable state satisfies $conf \Rightarrow req, appr \Rightarrow conf, appr \Leftrightarrow issued, issuedAt \leq now, active \Rightarrow claimed$, and $claimed \Rightarrow appr$.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference’17, July 2017, Washington, DC, USA

© 2026 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

- *active_needs_all*: in a reachable state, *active* implies all three authorizations. This is the whole-trace property the original artifact lacks.
- *single_use*: once *claimed* holds, no step changes *claimed* or *active*.
- *monotone*: no step decreases *now*.
- *expired_no_claim*: if $issuedAt + ttl \leq now$, then no step sets *claimed*. Together with monotonicity, an expired link stays unclaimable forever.

```
theorem active_needs_all {s} (h : Reach s) (ha : s.active = true) :  
  s.req = true ∧ s.conf0 = true ∧ s.apprS = true  
theorem expired_no_claim {s s' t} (hi : s.issuedAt = some t)  
  (he : t + ttl ≤ s.now) (st : Step s s') : s'.claimed = s.claimed
```

3 Why the clock matters

If each step may pick its own time *t*, as in [1], then the expiry guard can be met by choosing *t* small after the fact. Expiry soundness is then not a statement about executions at all. Making *now* part of the state, with *tick* as the only step that changes it, turns expiry soundness into a one-line case analysis.

4 Limitations

[OPEN: concurrent claims (the protocol says concurrent claims yield one success, which needs an atomic compare-and-delete assumption and an interleaving model); key expiry after claiming; revocation and changes of terms; a refinement proof against the deployed HTTP service; the conjecture of [1] that permissions are 0-truncated.] Novelty is modest: this is a small, correct mechanization meant as a corrective artifact. CPP 2027’s deadline (10 Sep 2026) has passed, so this is aimed at arXiv or a workshop.

Disclosure. Prepared with AI-agent assistance; the authors (Third Projection Collective) take responsibility for the content.

References

- [1] Anonymous. 2026. Higher Categorical Semantics of an Agentic Web-Site Protocol. <https://higher-categorical-semantics.pages.dev/research/higher-categorical-semantics-of-an-agentic-web-site-protocol-non-exclusive-rights-site-vms-5eda001c>.
- [2] Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction – CADE*

111	28 (<i>LNCS, Vol. 12699</i>). Springer, 625–635. https://doi.org/10.1007/978-	[3] Higher Categorical Semantics. 2026. Journal protocol 1.3.0 (agent skill,	166
112	3-030-79876-5_37	references/protocol.md). https://higher-categorical-antics.pages.	167
113		dev/skills/higher-categorical-antics-1.3.0.tar.gz.	168
114			169
115			170
116			171
117			172
118			173
119			174
120			175
121			176
122			177
123			178
124			179
125			180
126			181
127			182
128			183
129			184
130			185
131			186
132			187
133			188
134			189
135			190
136			191
137			192
138			193
139			194
140			195
141			196
142			197
143			198
144			199
145			200
146			201
147			202
148			203
149			204
150			205
151			206
152			207
153			208
154			209
155			210
156			211
157			212
158			213
159			214
160			215
161			216
162			217
163			218
164			219
165			220