

Architecture Document

Brendan McMillion
IETF 125 / March 18, 2026



State Loss

State loss is a security issue if the user has consumed a version of a label that was:

- In Contact Monitoring mode:
 - Inserted within the Reasonable Monitoring Window, or
 - Inserted in a non-gossipped portion of the log.
- In Third-Party Auditing mode:
 - Inserted within the auditor's Maximum Acceptable Lag

State Loss

Recommendation:

- Set the relevant parameters (RMW, OOB frequency, MAL, ...) to be much less than the expected duration between state loss events.
- If state loss could be triggered adversarially (e.g., web context):
 - Use Third-Party Management mode.
 - If Contact Monitoring or Third-Party Auditing is necessary for some reason, don't allow users to consume labels inserted recently.

Pruning & Privacy Law Compliance

- Transparency Logs needs to comply with deletion requests: GDPR / California Delete Act / etc.
- **Previously:** To handle this, the draft recommended “anonymizing” prefix tree entries by trimming down VRF outputs to be as short as possible without breaking anything

Pruning & Privacy Law Compliance

- Transparency Logs needs to comply with deletion requests: GDPR / California Delete Act / etc.
- **Previously:** To handle this, the draft recommended “anonymizing” prefix tree entries by trimming down VRF outputs to be as short as possible without breaking anything
- **Now:** Draft recommends just deleting affected VRF outputs from the most recent version of prefix tree. All versions of prefix tree with those VRF outputs will be flushed within the Maximum Lifetime

Pruning & Privacy Law Compliance

- Transparency Logs needs to comply with deletion requests: GDPR / California Delete Act / etc.
- **Previously:** To handle this, the draft recommended “anonymizing” prefix tree entries by trimming down VRF outputs to be as short as possible without breaking anything
- **Now:** Draft recommends just deleting affected VRF outputs from the most recent version of prefix tree. All versions of prefix tree with those VRF outputs will be flushed within the Maximum Lifetime

* This implies Maximum Lifetime \leq compliance timeline

WGLC!

Thank you everyone :)

Protocol Document



Fixed-Version Search

Previously:

- When search hit an expired log entry:
 - We would provide a binary ladder
 - If search would continue into the expired portion of the tree => **abort**
 - If search would continue away from the expired portion => **continue**

Fixed-Version Search

Previously:

- When search hit an expired log entry:
 - We would provide a binary ladder
 - If search would continue into the expired portion of the tree => **abort**
 - If search would continue away from

Pro: A label version expires when the log entry it was inserted in passes its Maximum Lifetime

Con: Requires keeping around (log) number of expired log entries

Fixed-Version Search

Previously:

- When search hit an expired log entry:
 - We would provide a binary ladder
 - If search would continue into the expired portion of the tree => **abort**
 - If search would continue away from

Pro: A label version expires when the log entry it was inserted in passes its Maximum Lifetime

Con: Requires keeping around (log) number of expired log entries

Now:

- When search hits an expired log entry: proceed to right child regardless
- Once at a terminal log entry:
 - Determine if our terminal log entry is to the **right** of any unexpired distinguished log entry
 - If so, return success. Else, return error.

Fixed-Version Search

Previously:

- When search hit an expired log entry:
 - We would provide a binary ladder
 - If search would continue into the expired portion of the tree => **abort**
 - If search would continue away from

Pro: A label version expires when the log entry it was inserted in passes its Maximum Lifetime

Con: Requires keeping around (log) number of expired log entries

Now:

- When search hits an expired log entry: proceed to right child regardless
- Once at a terminal log entry:

Pro: Much simpler impl. of pruning

Pro: Can guarantee when things will be deleted (re: compliance)

Con: Label versions expire when they're to the **left** of the leftmost unexpired distinguished log entry

All other algorithms are unchanged since IETF 124

Minor wire format change #1

```
~~~ tls-presentation  
struct {  
+ Configuration config;  
  opaque label<0..2^8-1>;  
  uint32 version;  
  opaque value<0..2^32-1>;  
} UpdateTBS;  
~~~
```

Minor wire format change #2

```
~~~ tls-presentation  
struct {  
    opaque opening[Nc];  
    opaque label<0..2^8-1>;  
+   uint32 version;  
    UpdateValue update;  
} CommitmentValue;  
~~~
```

Minor wire format change #3

1299	~~~ tls-presentation	1300	~~~ tls-presentation
1300	struct {	1301	struct {
1301	select (Configuration.mode) {	1302	select (Configuration.mode) {
1302	case thirdPartyManagement:	1303	case thirdPartyManagement:
1303	opaque signature<0..2^16-1>;	1304	opaque signature<0..2^16-1>;
1304	};	1305	};
1305	- } UpdatePrefix;	1306	+ } UpdateSuffix;
1306		1307	
1307	struct {	1308	struct {
1308	- UpdatePrefix prefix;		
1309	opaque value<0..2^32-1>;	1309	opaque value<0..2^32-1>;
		1310	+ UpdateSuffix suffix;
1310	} UpdateValue;	1311	} UpdateValue;
1311	~~~	1312	~~~

Next Steps

- Validate through implementation before doing WGLC as well

Implementation Report



Published Implementations

GitHub	What's implemented?	Language
https://github.com/Bren2010/katie	Server+Client+Auditor	Go
https://github.com/signalapp/key-transparency-server	Server	Go
https://github.com/signalapp/libsignal/tree/main/rust/keytrans	Client	Rust
https://github.com/signalapp/key-transparency-auditor	Auditor	Java
https://github.com/trailofbits/signal-auditor	Auditor	Rust
https://github.com/guyfischman/RuKT	Server+Client	Rust

Bren2010/Katie (Go)

- Maybe ~95% complete
- Matches current draft on:
 - Log Tree structure
 - Prefix Tree structure
 - Algorithms for interacting with the Combined Tree
 - Tree mutations (adding & removing labels)
- Biggest missing pieces:
 - Proof verification
 - Credentials

Bren2010/Katie (Go)

- Maybe ~95% complete
- Matches current draft on:
 - Log Tree structure
 - Prefix Tree structure
 - Algorithms for interacting with the Combined Tree
 - Tree mutations (adding & removing labels)
- Biggest missing pieces:
 - Proof verification
 - Credentials

This code is particularly tricky and copying / referencing is encouraged

- Supports versioned operations
- Minimizes db lookups & writes

Bren2010/Katie (Go)

Throughput:

- **Write path (tree mutations):** Depends on exact deployment but 1,000+ label value changes per second reasonably achievable
- **Read path (producing proofs):** Horizontally scalable

Bren2010/Katie (Go)

- Greatest-Version Search proof Size:

	Stateless Client	Stateful Client
No Distinguished Log Entry	~4.8 kB	~3.4 kB
Recent Distinguished Log Entry	~2.8 kB	~1.3 kB

- Owner Monitoring proof size: expect ~1 kB per Distinguished Log Entry

The End

Questions?