Architecture Document

Brendan McMillion
IETF 123 / July 23, 2025

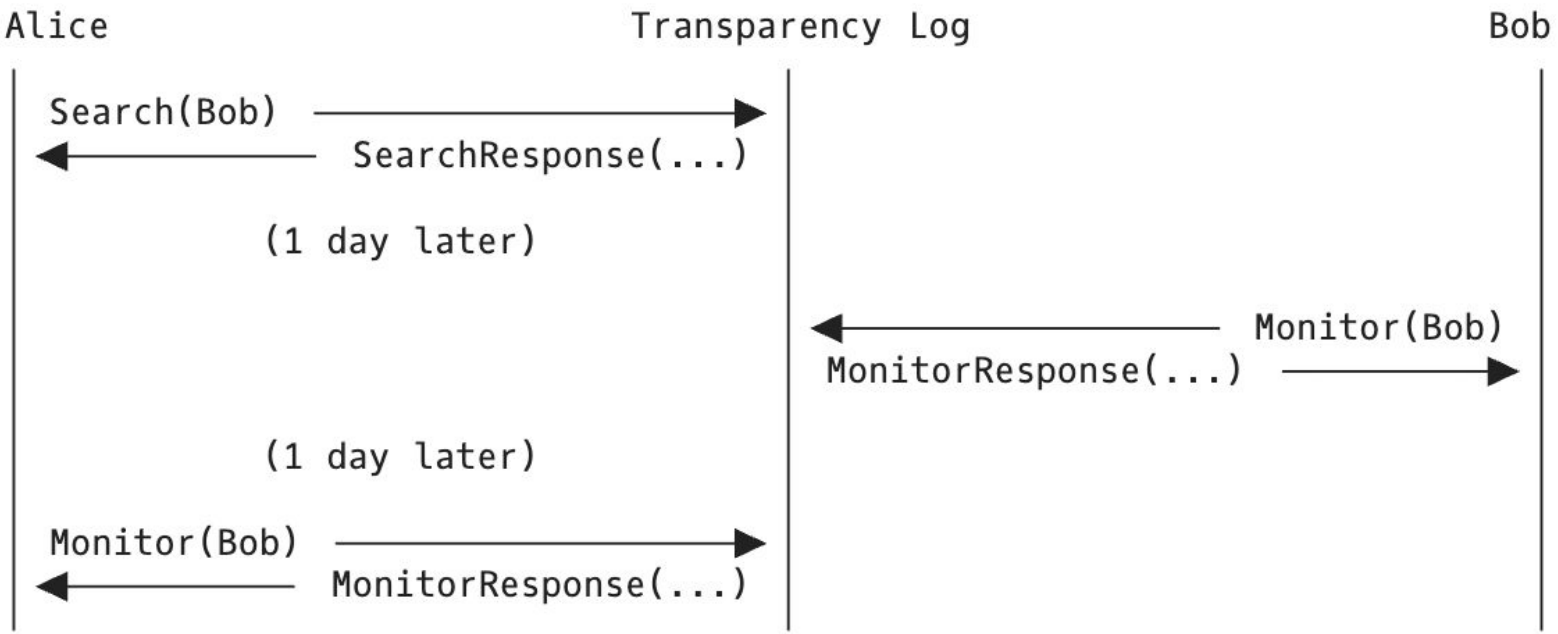
Automated Updates

“Users are considered to **own** a label if they are understood to either initiate all changes to the label's value, **or if they must be informed of all changes to the label's value.**”

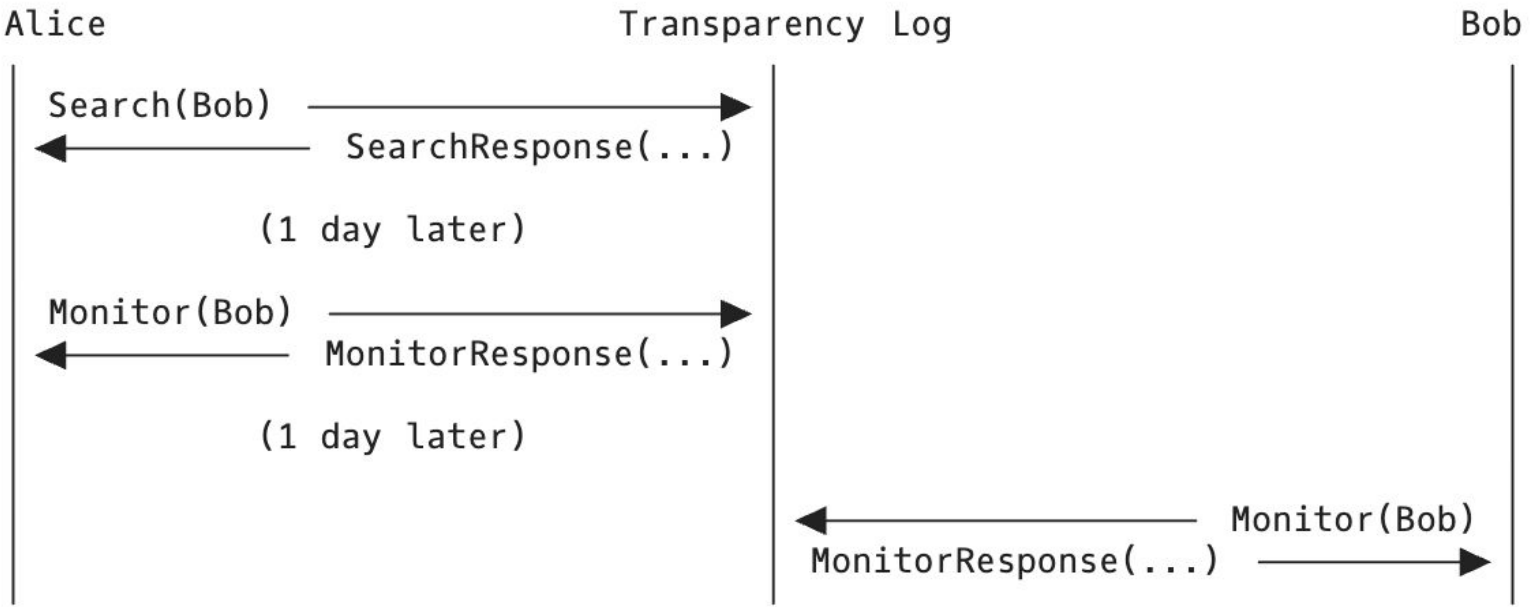
Use case:

- Service operator makes automated modifications to data.
- Users are informed after-the-fact to verify that changes were legal.

Contact Monitoring (before)



Contact Monitoring (now)



Third-Party Auditing

“Given the long-lived nature of transparency logs and the potentially short-lived nature of third-party auditing arrangements, **KT allows third-party auditors to start auditing a log at any arbitrary point.**”

+ Corresponding changes in protocol document to make this secure

From the top!

Interactive User Operations:
- Search
- Update
- Monitor

\+ *Non-interactive credentials*

Deployment Modes:
- Contact Monitoring
- Third-Party Auditing
- Third-Party Management

Non-binding guidance on:
- Combining multiple logs
- Understanding protocol security & privacy properties
- Pruning for efficiency vs compliance with privacy laws
- What to store in KT

Protocol Document

Third-Party Auditing

- Auditor receives an AuditorUpdate for each new log entry:

```
struct {
  uint64 timestamp;

  PrefixLeaf added<0..2^16-1>;
  PrefixLeaf removed<0..2^16-1>;

  PrefixProof proof;
} AuditorUpdate;
```

- Verifies that timestamp is >= last timestamp
- Verifies that each removed prefix tree leaf node was published in at least one distinguished log entry before being removed

Third-Party Management

- Updates are submitted signed by the Service Operator:

```
struct {
  UpdateRequest request;
  opaque signature<0..2^16-1>;
} ManagerUpdateRequest;
```

- Signature is verified by users in SearchResult:

```
struct {
  select (Configuration.mode) {
    case thirdPartyManagement:
      opaque signature<0..2^16-1>;
  };
} UpdatePrefix;

struct {
  UpdatePrefix prefix;
  opaque value<0..2^32-1>;
} UpdateValue;
```

Cipher Suites

Value	Name	R	Ref
0x0000	RESERVED	-	RFC XXXX
0x0001	KT_128_SHA256_P256	Y	RFC XXXX
0x0002	KT_128_SHA256_Ed25519	Y	RFC XXXX
0xF000 - 0xFFFF	Reserved for Private Use	-	RFC XXXX

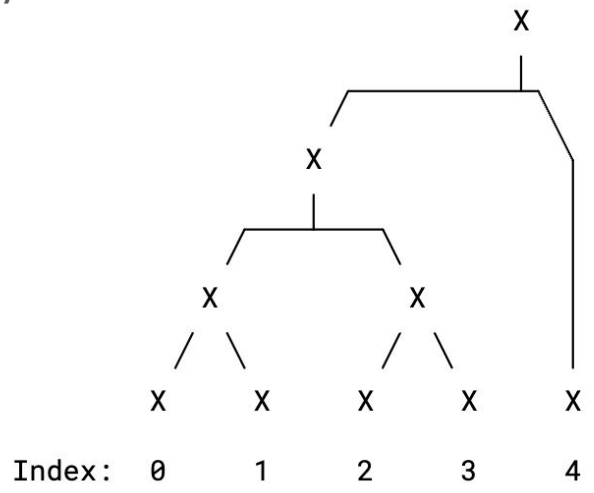
And that’s it! (RFC 9381 only specifies EC VRFs at classical 128-bit security level)

Specification Required for new entries

From the top!

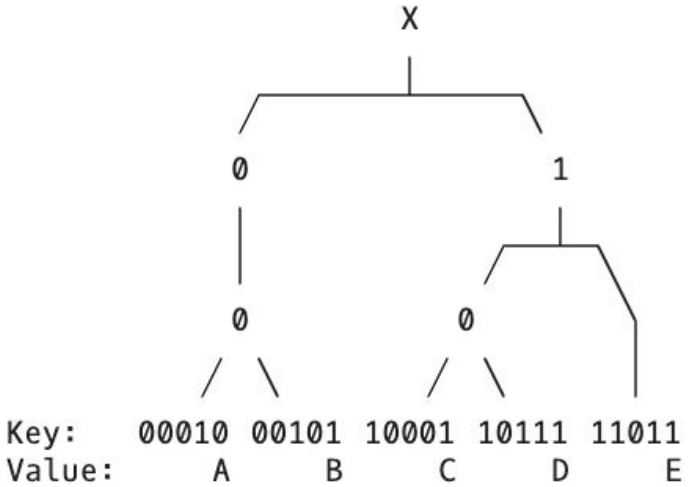
Log Tree

Left-Balanced Binary Tree: Every left subtree is the largest possible balanced binary tree



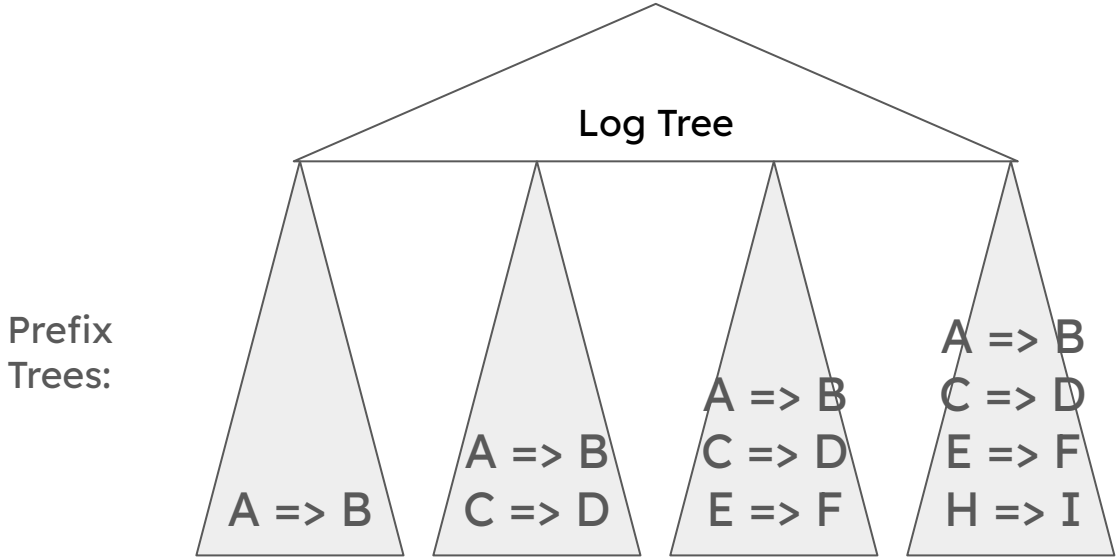
Prefix Tree

Trie: Every node represents a prefix that all strings stored below it share.



Combined Tree

- A prefix tree contains every label->value mapping stored in the Transparency Log
- A log tree contains the history of every version of the prefix tree



Working with a Combined Tree

Binary Ladder:

- Series of lookups in a specific version of the Prefix Tree
- Determines whether a given version of a label exists or not

Distinguished Log Entries:

- Log entries that are chosen at specific (time-based) interval
- Used to align consistency checks between users

Search

Fixed-Version Search:

- Binary search through log tree for first leaf where desired label-version exist
- Binary ladder from each log entry determines whether label-version exists or not

Greatest-Version Search:

- Find the greatest version of the label that exists in the rightmost distinguished log entry (with a binary ladder)
- Binary search to the rightmost log entry, repeating above step for each log entry

Monitor

- If we previously looked up a version of the label, and it was to the right of the rightmost distinguished log entry:
  - Check that a fixed-version or greatest-version search for the same version of the label would still converge back to what we saw

- **If you own the label:** for any new distinguished log entries:
  - Find the greatest version of the label that exists (with a binary ladder)
  - Check that the output is what you'd expect

Before Montreal

- Explicit Update operation to fix some bugs
- Random minor tweaks / bug fixes probably
- Fresh implementation

Please review & share thoughts on the mailing list!