

Server Monitoring

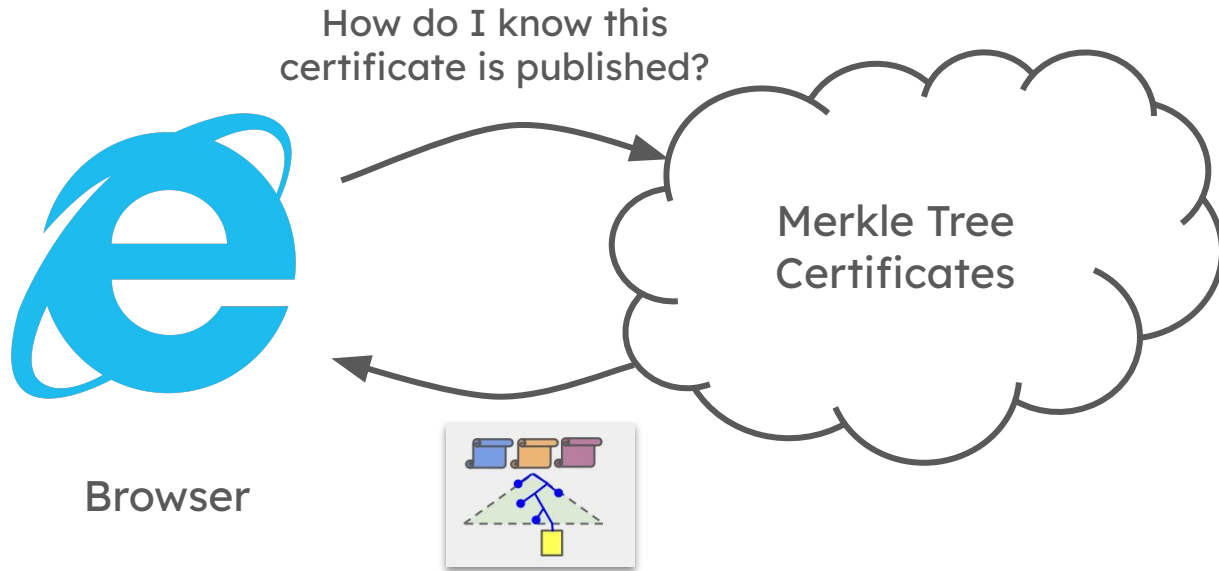
IETF 125 / Brendan McMillion

What is the security guarantee we want to provide?

At a high level: Browsers want to know that they're not accepting mis-issued certificates

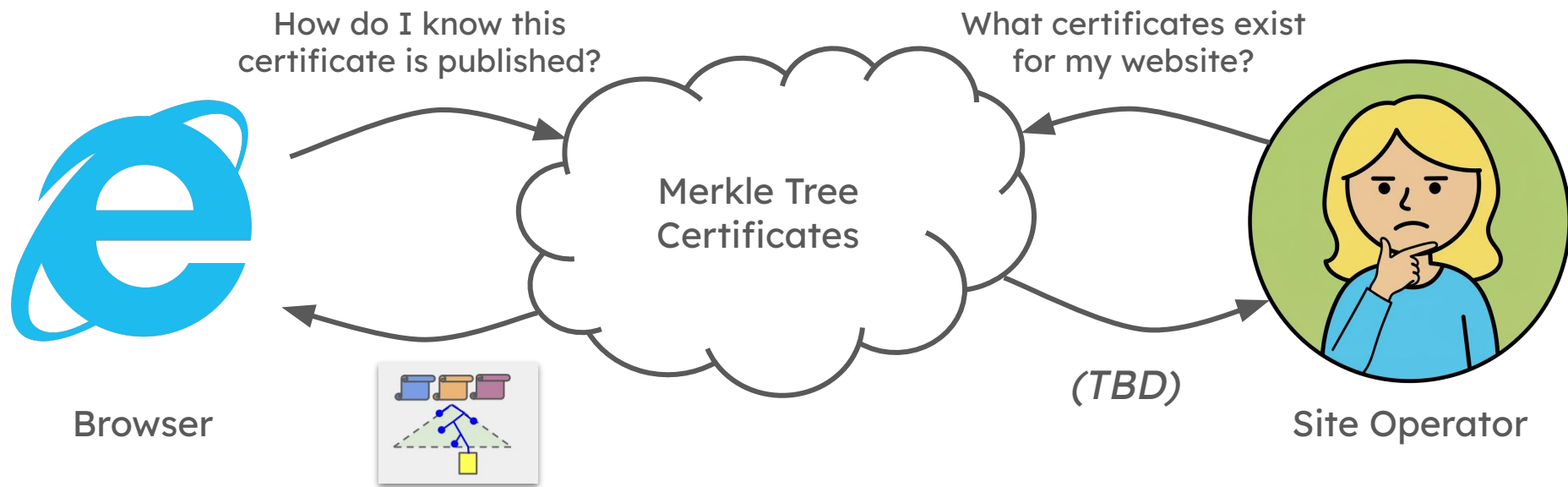
What is the security guarantee we want to provide?

At a high level: Browsers want to know that they're not accepting mis-issued certificates

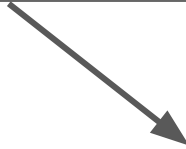
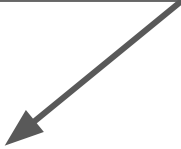


What is the security guarantee we want to provide?

At a high level: Browsers want to know that they're not accepting mis-issued certificates



Security of the whole system depends on both of these equally

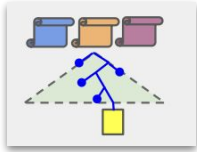


How do I know this certificate is published?

What certificates exist for my website?



Browser

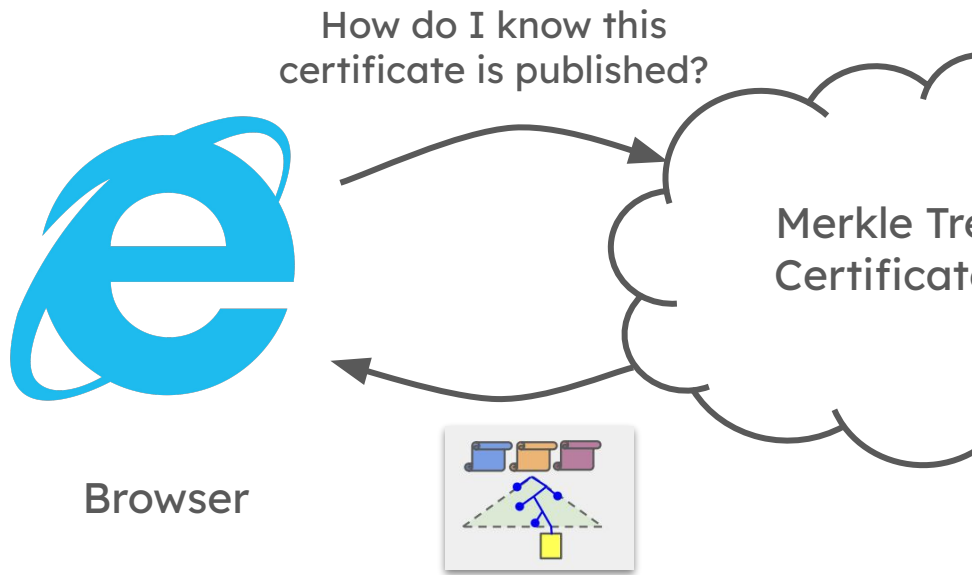


Site Operator

(TBD)

Why is the browser portion secure?

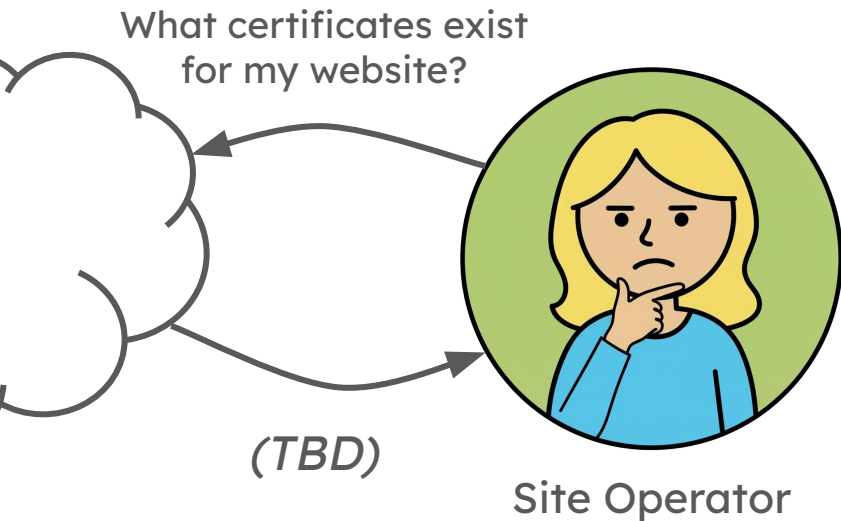
- The browser gets an inclusion proof in a transparency log
- **How do I know I'm seeing the same tlog as everyone else?**
The root hash is cosigned
- **Why are these cosigners trustworthy?**
Their public key is hardcoded into your browser



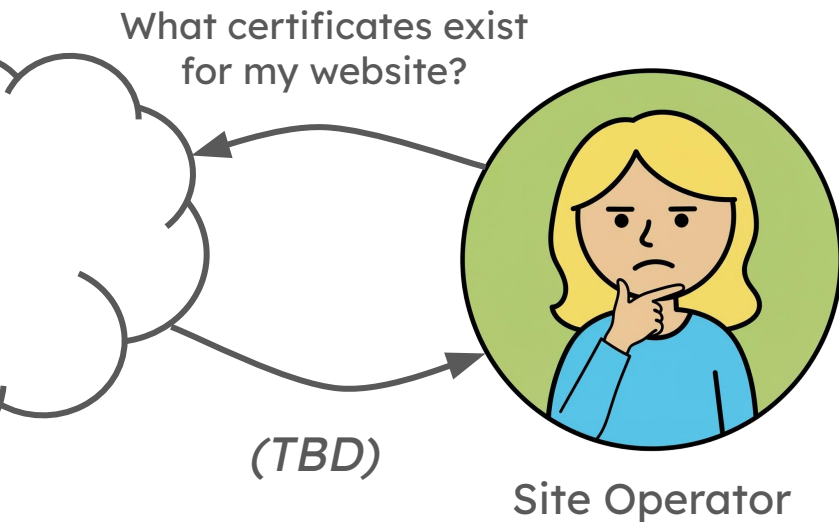
Why is the site operator portion secure?

Depends on how we fill in “(TBD)”

- Download the entire log?



Why is the site operator portion secure?

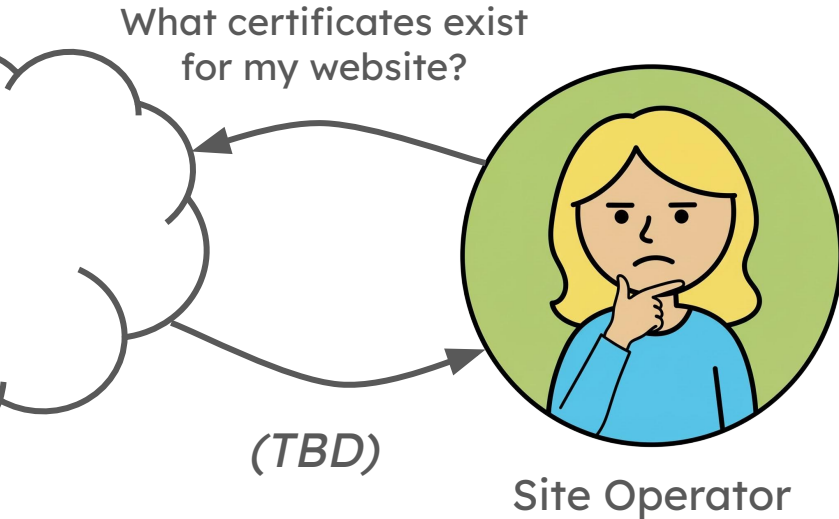


Depends on how we fill in “(TBD)”

- Download the entire log?

Way too inefficient

Why is the site operator portion secure?



Depends on how we fill in “(TBD)”

- ~~Download the entire log?~~
- Third-Party Monitor?

Security comes from: trust in the monitor

So what's the security of the whole system?

Getting a mis-issued certificate without the site operator knowing =

Certificate Authority has to misbehave

AND (Quorum of cosigners misbehaves

OR

Our Third-Party Monitor misbehaves)

So what's the security of the whole system?

Getting a mis-issued certificate without the site operator knowing =

Certificate Authority has to misbehave

AND (Quorum of cosigners misbehaves

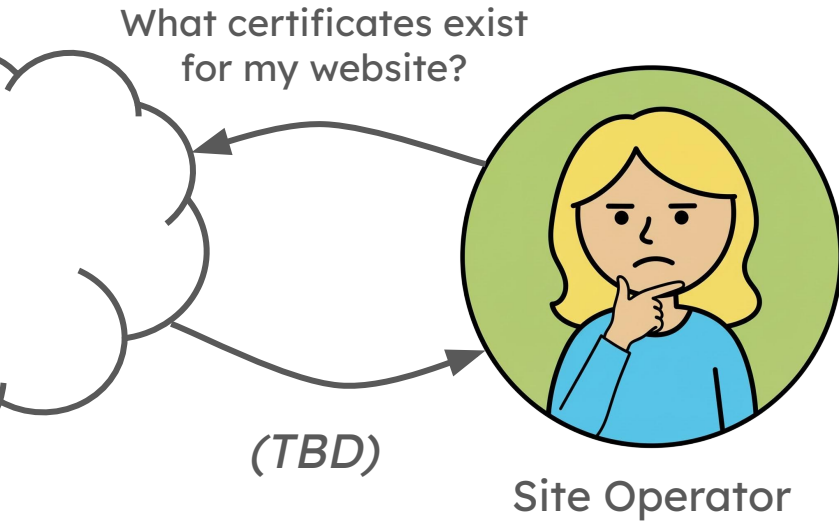
OR

Our Third-Party Monitor misbehaves)

Vetted by browser
Misbehavior is provable

None of the above :(

Why is the site operator portion secure?



Depends on how we fill in “(TBD)”

- ~~— Download the entire log?~~
- ~~— Third Party Monitor?~~
- Verifiable Index with Cosigners

So what's the security of the whole system?

Getting a mis-issued certificate without the site operator knowing =

Certificate Authority has to misbehave

AND (Quorum of cosigners misbehaves

OR

Verifiable Index **AND** Cosigners misbehave)

Better, but these
two are still
undercutting each
other

So what's the security of the whole system?

Getting a mis-issued certificate without the site operator knowing =

Certificate Authority has to misbehave

AND (Quorum of cosigners misbehaves

OR

VIndex **AND** SAME cosigners misbehave)

UNLESS we know
that the cosigners
are the exact same

= CA Misbehaves **AND** Quorum of cosigners Misbehave

This is the security
guarantee we want

How does the **server** know the browser's policy for cosigners?

- Force the server to track the development of every browser they care about?
- Add some protocol mechanism where browsers advertise/confirm cosigner public keys and policy?
- Add a new log entry type for verifiable index root hashes

How does the **server** know the browser's policy for cosigners?

- Force the server to track the development of every browser they care about?
- Add some protocol mechanism where browsers advertise/confirm cosigner public keys and policy?
- Add a new log entry type for verifiable index root hashes
If the client accepts the server's certificate, then:
=> The server knows it was cosigned correctly, which means:
=> These log entries are also cosigned correctly

How does the server find these log entries?*

*** Without introducing new trusted parties!**

- `accumulated_count` field contains sum of total `accumulated`-type entries logged so far
- Server can do verifiable search to find n^{th} such log entry

```
struct {
    uint64 accumulated_count;
    MerkleTreeCertEntryType type;
    select (type) {
        case null_entry: Empty;
+
        case tbs_cert_entry: opaque tbs_cert_entry;
+
        case accumulated:
            AccumulatedType accumulated_type;
            select(accumulated_type) {}
+
        /* May be extended with future types. */
    }
} MerkleTreeCertEntry;
```

PR for review

<https://github.com/davidben/merkle-tree-certs/pull/172>

Thanks :)