TLS Trust Expressions

draft-davidben-tls-trust-expr

David Benjamin Devon O'Brien Bob Beck

At a high level, TLS Trust Expressions defines:

A way for relying parties to succinctly convey trusted certification authorities to subscribers, and

Supporting mechanisms for subscribers to evaluate this information and select a trusted certificate path to serve, and

It does so in a way that's flexible enough to improve several real-world PKI use cases

Use Case: Say you want to rotate a root CA key...

Just make a new key and issue from it instead...

Difficult to transition smoothly:

- New relying parties might only trust the new key
- But old relying parties only trust the old key
- How do you satisfy both? ← long overlapping inclusions

The Web PKI has *25-year-old* root keys because this is really hard!

Single-Certificate Model

In practice, relying parties do not tell the subscriber what they trust

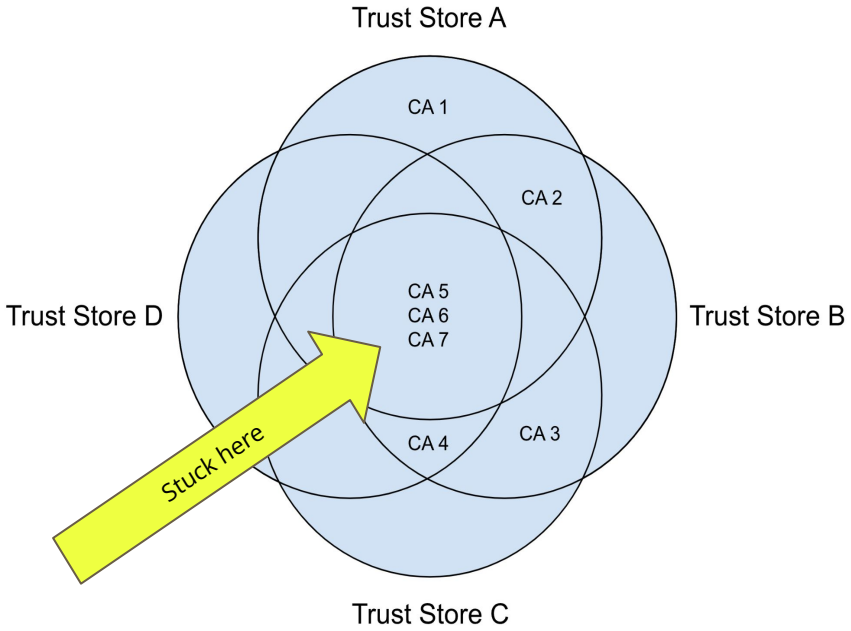
Different relying parties have different requirements:

- Also, different versions of a relying party
- Also, these requirements are not coordinated and can conflict

Subscribers need a single certificate bundle that is supported by all relying parties

→ **Subscriber choice is limited to the intersection of all supported relying parties**

Stuck in the Intersection



CA challenges:

- Only ubiquitous roots are useful for subscribers
- Difficult to support old and new relying parties across changes

Subscriber challenges:

- Limited choice
- Certificate changes for one relying party impacts support for another relying party
- Hard to predict what will work

Relying party Challenges:

- Policy changes to meet new security requirements for certificates burden the ecosystem — user security usually suffers

Multi-Certificate Model

Instead, use different certificate paths for different relying parties as needed

Requires two changes:

1. A certificate negotiation mechanism to select the right certificate path
2. An issuance mechanism for subscribers to easily obtain multiple paths

Certificate Negotiation

Trust store manifests

Trust store inclusions

Trust expressions

certificate_authorities?

certificate_authorities is a big list of X.509 names:

- X.509 names are inefficient
- Relying parties may trust many CAs

Chrome Root Program — 131 names totalling **13,104 bytes**

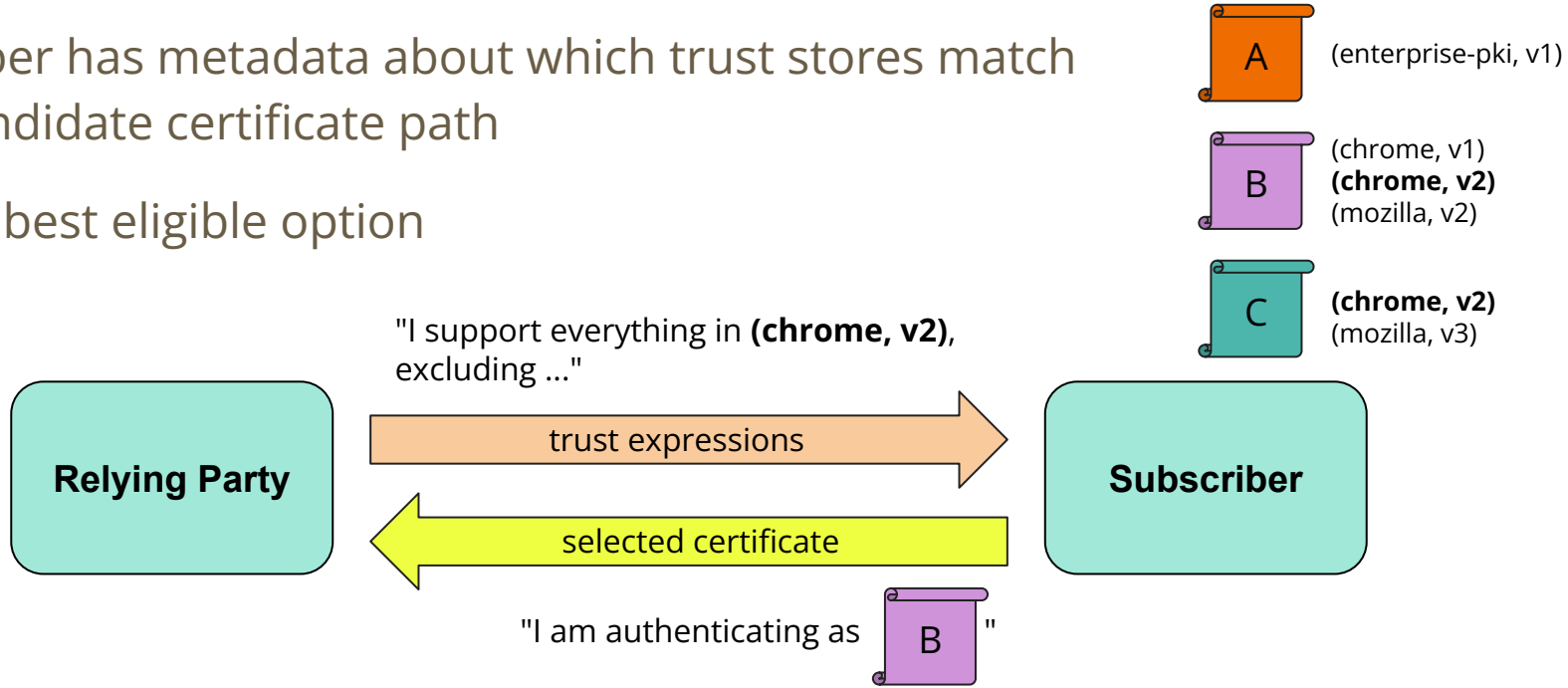
Mozilla CA Certificate Program — 144 names totalling **14,475 bytes**

Named Trust Stores

Relying party references versioned and named trust stores

Subscriber has metadata about which trust stores match each candidate certificate path

Pick the best eligible option

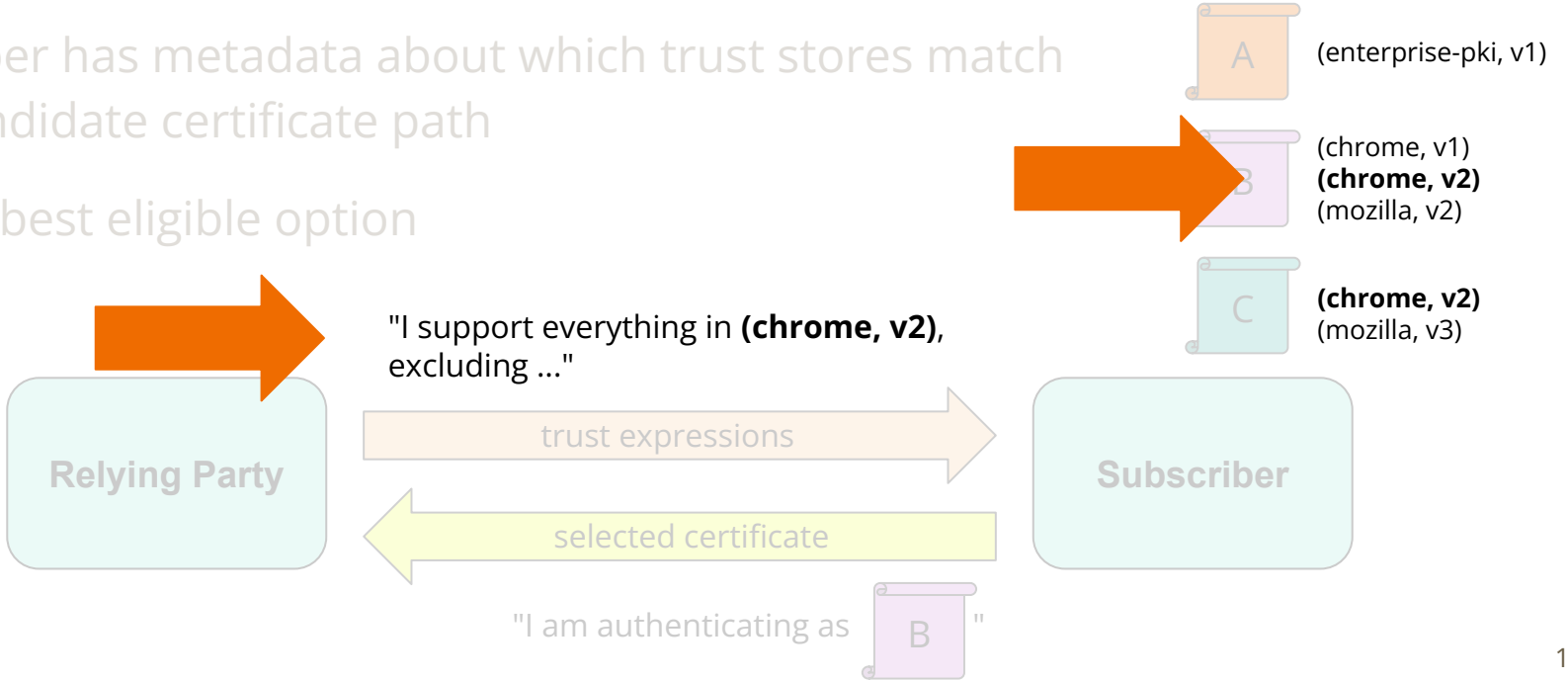


Where Does This Come From?

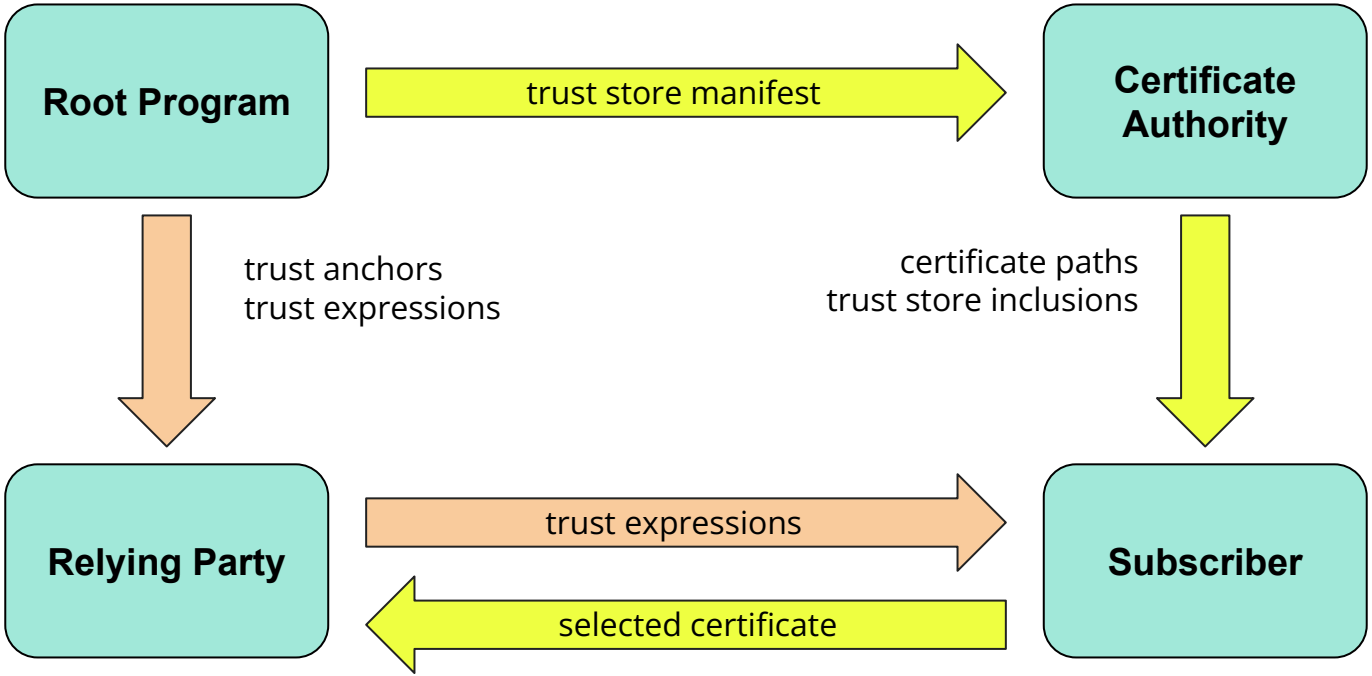
Relying party references versioned and named trust stores

Subscriber has metadata about which trust stores match each candidate certificate path

Pick the best eligible option



Overview


Root Program
trust store manifest
Certificate Authority
trust anchors
trust expressions
certificate paths
trust store inclusions
Relying Party
trust expressions
selected certificate
Subscriber

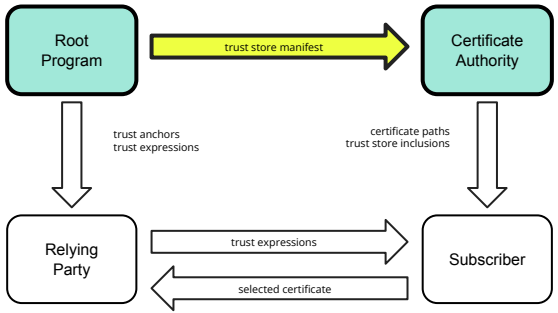
Trust Store Manifests

JSON document published by root program

Describes current and historical versions of a trust store

Also records integer "labels" and maximum leaf certificate lifetime

New versions defined over time



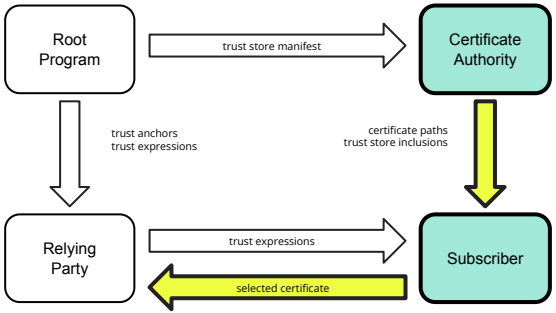

```
{
  "name": "example",
  "max_age": 864000,
  "trust_anchors": {
    "A1": {"type": "x509", "data": "..."},
    "A2": {"type": "x509", "data": "..."},
    ...
  },
  "versions": [
    {
      "timestamp": 1672531200,
      "entries": [
        {"id": "A1", "labels": [0, 100], "max_lifetime": 7776000},
        {"id": "A2", "labels": [1, 100], "max_lifetime": 7776000},
        {"id": "B1", "labels": [2, 101], "max_lifetime": 7776000},
        {"id": "B2", "labels": [3, 101], "max_lifetime": 7776000}
      ]
    },
    {
      "timestamp": 1675209600,
      "entries": [
        {"id": "A1", "labels": [0, 100], "max_lifetime": 7776000},
        {"id": "A2", "labels": [1, 100], "max_lifetime": 7776000},
        {"id": "C1", "labels": [4, 102], "max_lifetime": 7776000},
        {"id": "C2", "labels": [5, 102], "max_lifetime": 7776000}
      ]
    }
  ]
}
```

Trust Store Inclusions

CA collects manifests from each root program it participates in

Generates TrustStoreInclusionList for each issued certificate path

- Sent to subscriber at issuance
- Describes the path's matching trust store versions
- Assumed not updated until certificate renewal




```
enum {
    previous_version(0),
    latest_version_at_issuance(1)
} TrustStoreStatus;

struct {
    opaque name<1..2^8-1>;
    uint24 version;
} TrustStore;

struct {
    TrustStore trust_store;
    TrustStoreStatus status;
    uint24 labels<1..2^16-1>;
} TrustStoreInclusion;

TrustStoreInclusion
    TrustStoreInclusionList<1..2^16-1>;
```

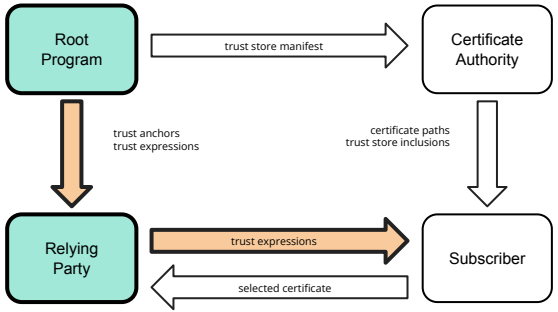
Trust Expressions

Root programs provide TrustExpressionList with trust anchor list

Each trust expression matches the referenced trust store, minus exclusions

A path is eligible if it matches any trust expression

Exclusions account for subscribers that predate the trust store (sections 4.1 and 6.5)




```
enum {
    trust_expressions(TBD), (2^16-1)
} ExtensionType;

struct {
    opaque name<1..2^8-1>;
    uint24 version;
} TrustStore;

struct {
    TrustStore trust_store;
    uint24 excluded_labels<0..2^16-1>;
} TrustExpression;

TrustExpression
    TrustExpressionList<1..2^16-1>;
```

Privacy

Only advertise trust anchors common to anonymity set

- E.g. browser vendor or OS root program

Other trust anchors continue to work

- If no trust expressions match, subscribers use previous behavior

ACME Extensions

Sending multiple certificate paths

Certificate properties

Sending Multiple Certificate Paths

ACME already has this

No more heuristic needed

Lift the same end-entity restriction

As long as CA is okay issuing and renewing all variants together

Future work: another mechanism for more complex cases? (Multiple orders?)

> The server MAY provide one or more link relation header fields [RFC8288] with relation "alternate". Each such field SHOULD express an alternative certificate chain starting with the same end-entity certificate. This can be used to express paths to various trust anchors. Clients can fetch these alternates and use their own heuristics to decide which is optimal.

Certificate Properties

CertificatePropertyList — extensible container for certificate metadata, e.g. trust store inclusions

New media type: `application/pem-certificate-chain-with-properties`

> Prepend a CERTIFICATE PROPERTIES block to existing ACME type

Use HTTP Accept header to negotiate

```
enum {
    trust_stores(0), (2^16-1)
} CertificatePropertyType;

struct {
    CertificatePropertyType type;
    opaque data<0..2^16-1>;
} CertificateProperty;

CertificateProperty
    CertificatePropertyList<0..2^16-1>;
```

```
-----BEGIN CERTIFICATE PROPERTIES-----
...
-----END CERTIFICATE PROPERTIES-----
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

CAs transparently issue multiple paths that *together* cover all relying parties

Root ubiquity problem is gone!

Multi-Certificate Examples

Key rotation

Eliding intermediates

Postquantum roots

Backup certificates

Key Rotation

CA operator generates new root key

Issue from both in parallel

New relying parties get new root, old ones get old root

Out-of-date relying parties work as long as old root key is in operation

No subscriber changes are required during a rotation

> Subscriber *does not need to know* why there are two paths, just which to send where

Eliding Intermediates

A predistributed intermediate is the same as a (short-lived) trust anchor

Can omit intermediates as in draft-ietf-tls-cert-abridge-00:

- CA sends short path and long path to subscribers
- Up-to-date relying parties get the short path from subscriber
- Older relying parties get the long path from subscriber

Postquantum roots

Postquantum roots can be gradually deployed

CAs may introduce postquantum roots at different times

Relying party may trust *some* postquantum roots, but not *a specific* postquantum root

Certificate negotiation fixes the root ubiquity problem

Backup Certificates

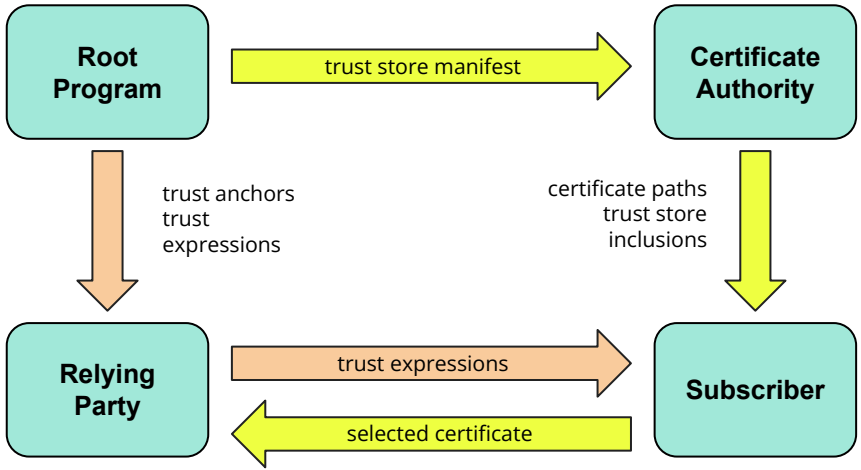
Subscriber can combine output from multiple CAs

- Same negotiation mechanism as paths from one CA, different deployment model

Backup if one CA is unreachable, compromised, or removed.

Can also be used to be compatible multiple ecosystems.

Recap



Trust expressions allow relying parties to compactly describe a trust anchor list

CA provisions subscriber with metadata to select certificates for each relying party

Enables a multi-certificate deployment model

PKI transitions can proceed smoothly without subscriber disruption

Questions?

https://github.com/davidben/tls-trust-expressions/

https://datatracker.ietf.org/doc/draft-davidben-tls-trust-expr/

Backup slide: Version Skew

TrustStoreInclusionList set at issuance, but relying party may be newer

Relying party says (example, v3), but subscriber knows up to (example, v2)

- latest_version_at_issuance entries match all subsequent versions
- If a CA was removed in v3, relying parties exclude it until last pre-v3 certificate expires

(Details in sections 4.1 and 6.5)

```
// (example, v2)
{
  "timestamp": 1672531200,
  "entries": [
    {"id": "A", "labels": [0, 2],
     "max_lifetime": 7776000},
    {"id": "B", "labels": [1, 2],
     "max_lifetime": 7776000}
  ]
},

// (example, v3)
{
  "timestamp": 1675209600,
  "entries": [
    {"id": "B", "labels": [1, 2],
     "max_lifetime": 7776000},
  ],
}
```