2022-June-14: Draft DRP Security Model

Picture this: a covered business implements Data Rights Protocol and receives a request; how do they know that this request came from a trustworthy party? Online identity is still a fractured, incomplete system with varying levels of assurance, trust, and verification, and the DRP needs to define a system that can assert that identity attributes are accurate and that the request was affirmed by the user whose attributes are being bundled.

There are two/three questions this document seeks feedback on:

- How should the DRP network establish a chain of trust for identity JWTs?
- How should the DRP secure the API?
    - Client certificates, or API keys, or moving the JWT envelope to encapsulate the whole request
- How do these requirements overlap?

For the purpose of conversation here, the role of Privacy Infrastructure Provider is abstracted away. Inasmuch as a Covered Business is willing to trust the PIP to provide these services they can be managed therein, but ultimately it is up to the Covered Business to trust the Authorized Agent.

JWT Trust Chain Discussion

There are numerous paths to securing JWTs already encoded in the JWT-adjacent RFCS. These RFCs provide general and particular guidance on creating secure JWTs and the security of DRP depends on secure JWTs. Ultimately, though, the **trust** of the keys have to be established since they do not contain a signature chain like x.509 certificates.

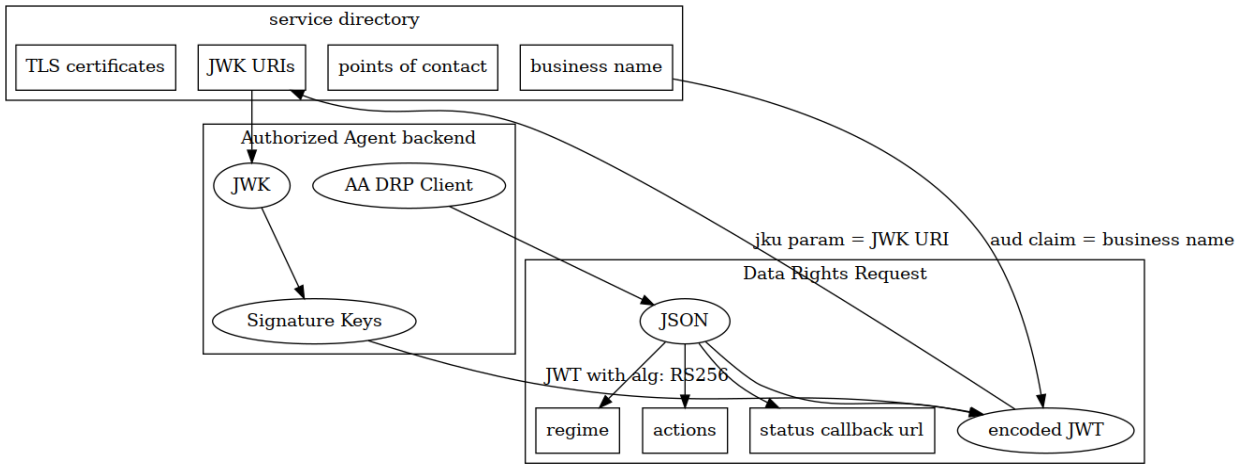
Background + Prior Work

- 7519 JWT is the specification of JSON Web Tokens, an URL-safe encoding of JSON objects.
- 7515 JWS is the specification of signed JWTs, both symmetric and asymmetric.
    - Header parameters for specifying signature algorithms and identifying which keys are used in alg and kid
    - Embedded JWK jwk claim or Reference to a JWK hosted on a remote server in jku
    - Embedded or referential x.509 certificates in x5u, x5c, x5t, etc
- 7516 describes the encryption envelope for JWTs, which is not being included in discussion right now in lieu of transport encryption.
- 7517 JWK specifies a JSON document format for representing keys which can be used in JWS.

    - Can represent both encryption + signing keys
    - Crypto primitives + "raw numbers"
    - Can also embed references to x.509 certs
- 7518 JWA outlines a set of crypto algorithms safe to use in JWTs.

The relationship follows: a JWT is signed using the JWS algorithm and encrypted using the JWE algorithm, using keys defined as a JWK using algorithms defined in JWA. Easy stuff, no?

Components of a DRP Trust Network



Chain of trust

1: Service Directory is managed by DRP consortium

The **Service Directory** is managed by the DRP consortium and consists of a machine-legible – probably JSON file hosted on S3 or somewhere with HTTPS – containing for each Authorized Agent:

- the business name
- the business's DRP point of contact(s)
- mTLS certificate thumbprint
- JWK URI or the JWK itself
- probably also a cryptographic hash of the JWK

Covered Businesses will probably be represented in this directory as well, but 🤷

I don't know how to make **this** thing secure though, which is ironic since it ends up being the trust root…

My first intuition is a tightly-controlled Git repository with all the bells-and-whistles therein. No force-pushing, tools for updating the repo included within it or in GitHub Actions. A more-engineered solution could be to use something like Amazon QLDB or a similar

permissioned ledger but that is way outside of my experience and comfort to weigh in on at the moment.

2: Authorized Agent hosts JWK on TLS (HTTPS) endpoint

Authorized Agents will generate cryptographic secrets for use in signing JWTs, and present the public portions of that cryptographic secret in a JWK file hosted under HTTPS on their business domain. This secret may-or-may-not by the same keys as the API client authentication (see below…)

3: (optionally?) JWT is encrypted against the CB's JWK

If CBs participate in the Service Directory they could publish JWKs to direct Authorized Agents to encrypt the identity tokens. An JWT encoded as sign(encrypt(payload)) would technically prevent CBs from re-using tokens or leaking them to third parties. This is probably obviated or off-set by the DRP API security discussion below, however.

4: JWT is signed against the AA's JWK

Either the "raw" JWT or JWE is signed with AA's private key. CBs would fetch the JWK to validate the signature

5: JWT includes securing claims

A JWT must include the following claims to prevent re-use or modification:

- aud claim is the Covered Business's name
- iss claim is the Authorized Agent's name

6: The attributes inside of the tokens themselves are affirmatively verified

Attributes should be affirmatively verified to the extent possible. Levels of assurance of the attributes themselves are kind of outside of the scope of this document as we are only concerned with AA-CB chain of trust, but Consumer-AA trust is critical to all of this.

Discussion

I would love to be able to "just" use x.509 certificates here – JWT specifies x.509 support in some fashion but I have not seen a library in any language which actually implements these features. It seems that in the best case scenario, an x.509 PEM would be referenced in the JWT, and the Covered Business would be responsible for engaging in something like this to validate a signed JWT:

- Download the JWK from jku or extract from jwk header parameter
    - this includes the "raw" cryptographic primitives
- Download the PEM from x5u
    - extract cryptographic primitives

- Validate the chain of trust to either DRP-managed root or trusted webpki root
- Compare the primitives

The JWK does not include the same trust chain information but is responsible only for communicating the cryptographic primitives. I am … uncomfortable with asking folks to work "this close to the math" for limited gain.

The content and trust of the service directory ends up being the root of this trust chain either way, so I'd like to prefer to get this right. What information needs to be included to establish a trust root for the JWKs? How do we ensure that a random engineer or interloper within an Authorized Agent's network can swap the certificate and forge requests? If this does happen, what detection mechanisms can we insert to protect against this?

In my view, the service directory can be managed in a GitHub repository with strict ACLs against pushing un-approved changes, with approval chains that would require cross-organization collaboration, including checksums/hashes of the JWK. It may be more feasible for the JWK to be directly held within the Directory rather than relying on a hosted JWK, as well.

DRP API Security Discussion

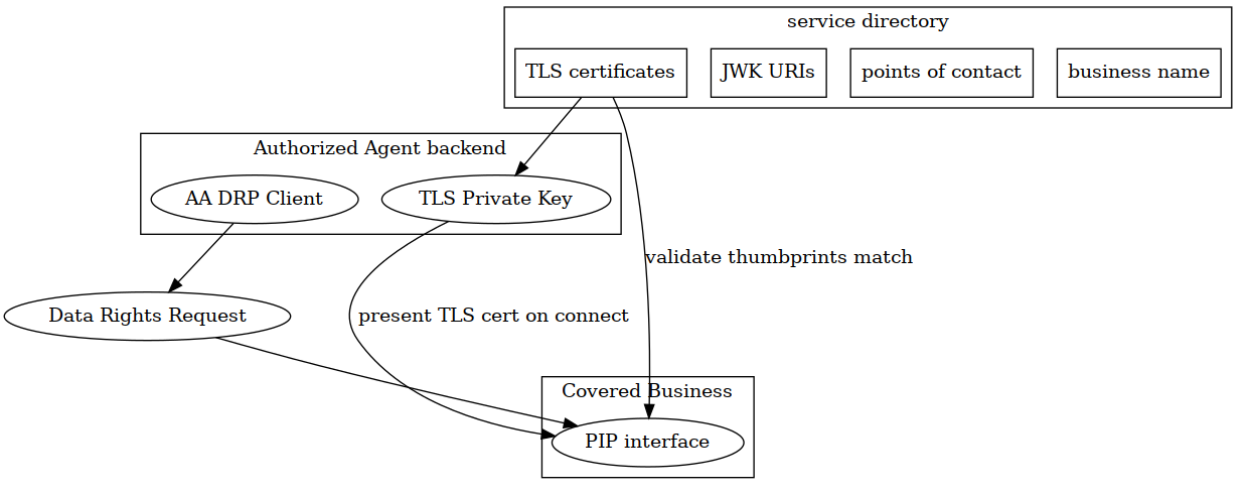
Initial drafts of the DRP protocol kindly gloss over this aspect, but there is a hard question here: How does the Covered Business know that the request is coming from an Authorized Agent, and **which** authorized agent? Building on prior art, we write in section 3.07 of the spec "the intention is to eventually leverage OAuth2 to secure these resources, either in concert with OIDC or out of band" but I am not so sure that will be viable in reality. These are fundamentally about **authorizing a known user to a system** and in DRP this is often not going to be the case. End Users may not have a login account or may be unaware of the account and this should not affect their fundamental rights. Without being able to engage in a full access/refresh token flow, the API will fall back to static client id + client secret pair crammed into an Authorization header. Managing these pairs would be a multiplicative weight of duty and probably is too onerous to sustain. Each AA would have to manage a key pair with each covered business in some "out of band" onboarding/rotation process.

There is a “shortcut” here where all the Covered Businesses under a single PIP can roll-up in to a single API key, so it’s not quite unscalable.

I propose here two alternative which may be better:

Client Certificate/mutual TLS

Sorry but the only thing worse than graphviz is every other diagramming tool…



Covered Businesses will continue to rely on web PKI certificates to secure their API to snoopers with HTTPS, but additionally Authorized Agents will present a certificate in their client connect process. The thumbprint of that certificate will be made available in the Service Directory. If no certificate is presented or if the certificate is not in the directory, the PIP would drop the connection and log it. These certificates could come from web PKI trusted roots or a "private" root managed by a vendor on behalf of the DRP consortium, but I don't think they're the same cryptographic primitives as in the JWK?

Alternative: Move everything in to JWT

Referring back to the Chain of Trust discussion, it seems reasonable to consider moving the entire request into the JWT. Rather than representing the identity of the user making a data rights action, the JWT would contain the entire request. This would conflict with using identity tokens provided by a Covered Business's IDp, however.

```
{
    "iss": "Permission Slip",
    "aud": "One Covered Business",
    "name": "James Consumer",
    "email": "james@consumer.org",
    "drp.regime": "ccpa",
    "drp.actions": ["deletion"],
    "drp.callback":
"https://permission-slip-api.consumer.org/drp/status"
}
```

Attack Vectors

- outside modification/replacement of cryptographic secrets in directory
- bulk requests for deletion or opt-out
- forged requests from backend
    - making a data rights request with 3rd party server for async updates
- leaked secrets from backend

Errata and Random

I also have been thinking about this, and I generally have reservations about being able to re-used tokens from Google or Facebook federated logins without them being able to offer custom scopes. I spent quite a lot of time looking for documentation on whether these scopes could be added, but so far it seems like we'd have to collaborate with those companies to extend their IDp servers with new claims or scopes

.