

Extensions

draft-hendrickson-privacypass-public-metadata-issuance

draft-wood-privacypass-auth-scheme-extensions

draft-hendrickson-privacypass-expiration-extension

Issuance Protocol Properties

Private Token types determine issuance protocol properties, including:

- Challenge and redemption structures (TokenChallenge and Token)

- Public verifiability support

- Public metadata support

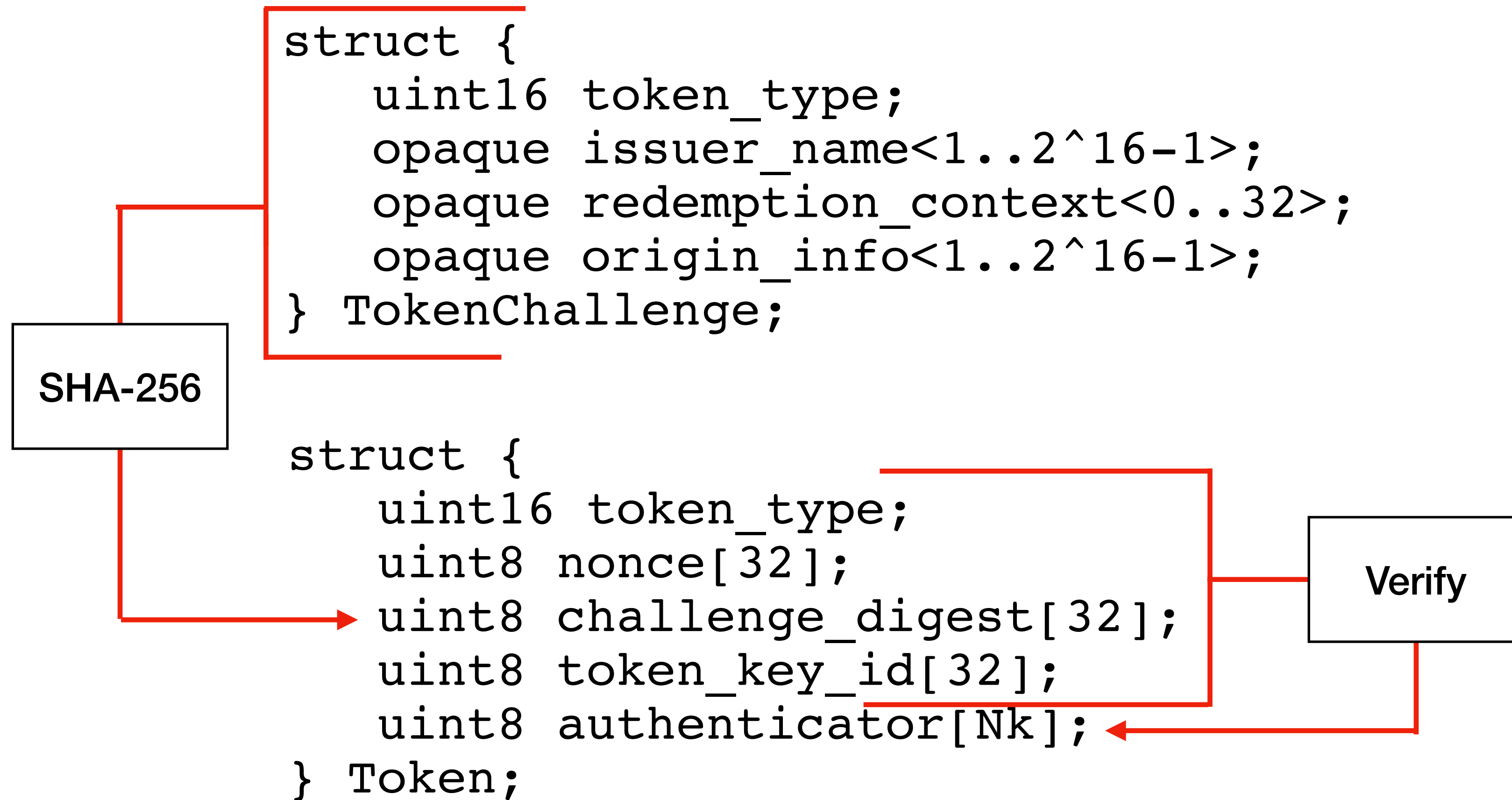
- Private metadata support

- ...

Basic issuance protocols in draft-ietf-privacypass-protocol support public and private verifiability with **no metadata support**

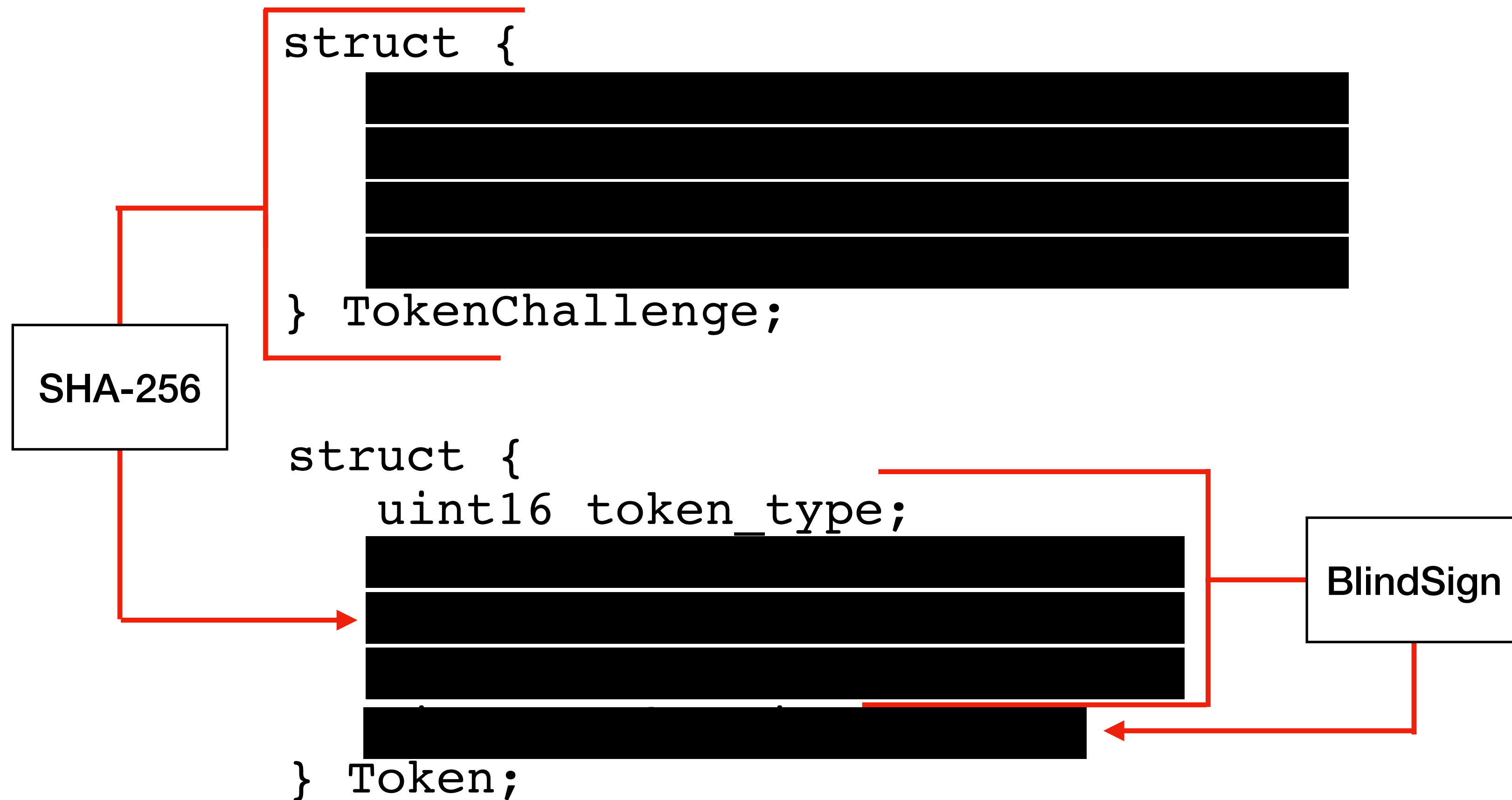
Basic Privacy Pass Tokens

Origin's perspective



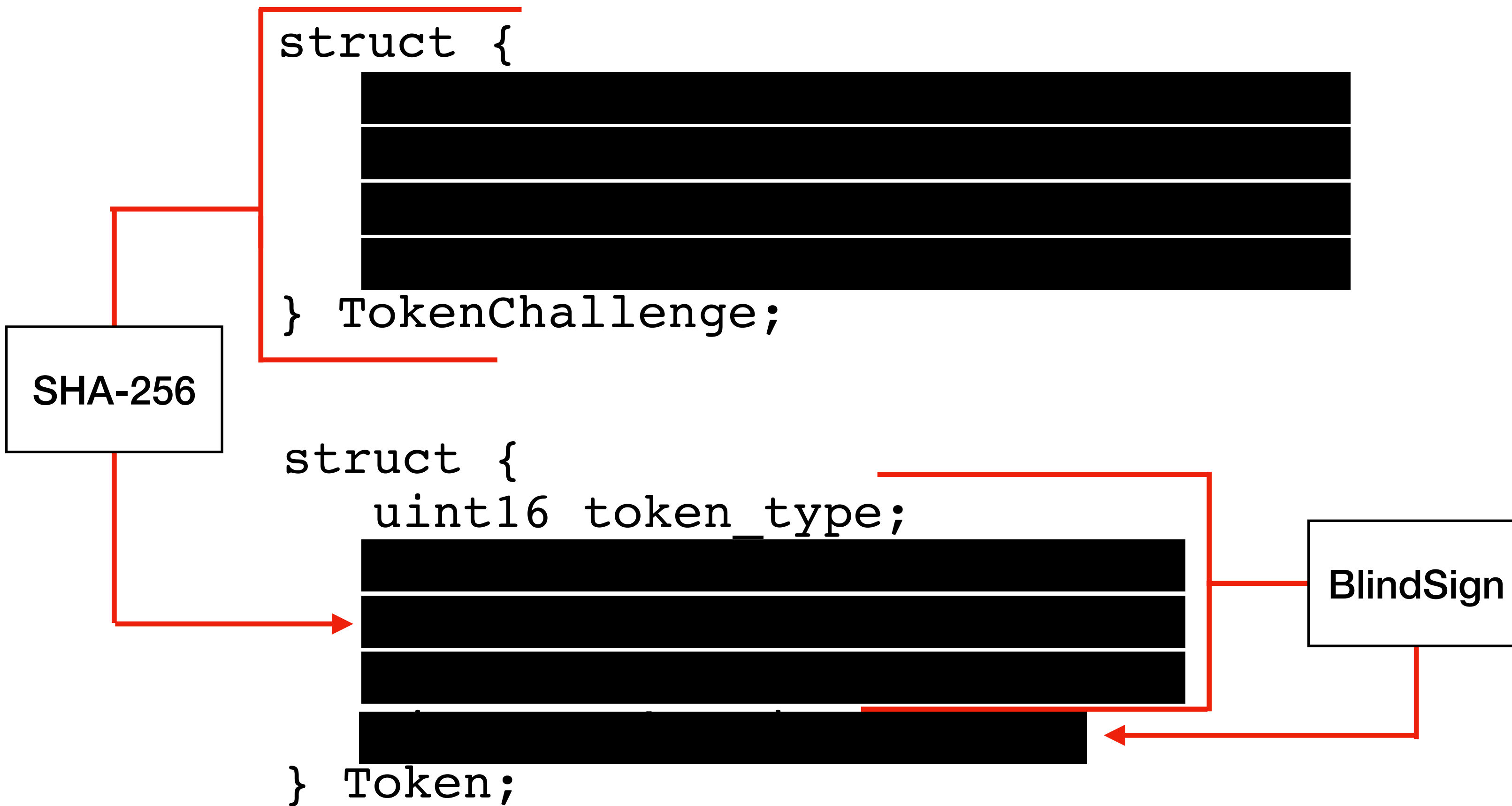
Basic Privacy Pass Tokens

Issuer's perspective



Basic Privacy Pass Tokens

Limitations



Issuer (and Attester) cannot see or enforce or communicate any sort of policy on token challenges

Examples:

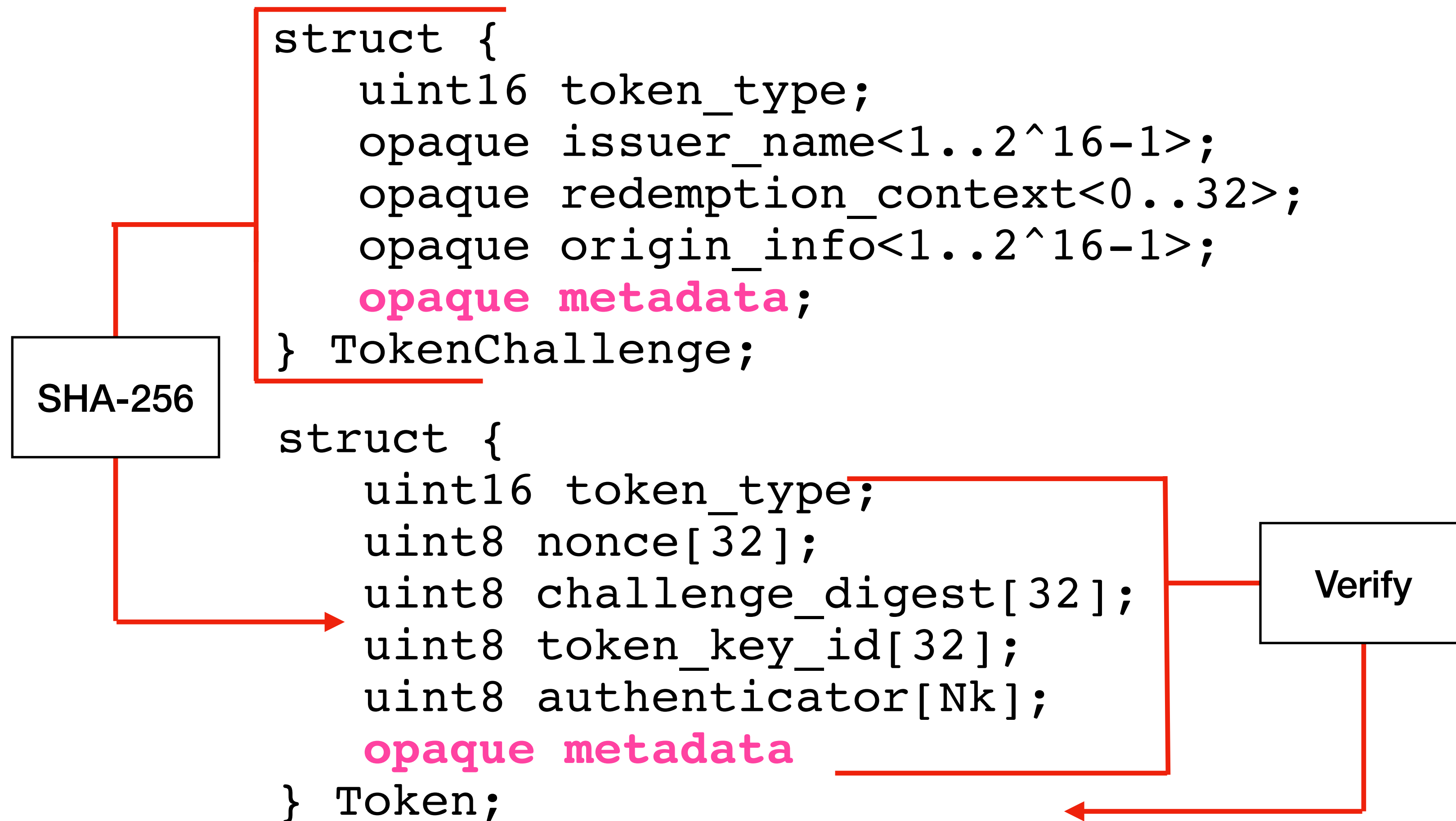
Token expiration information

Token geolocation metadata (see Private Relay)

...

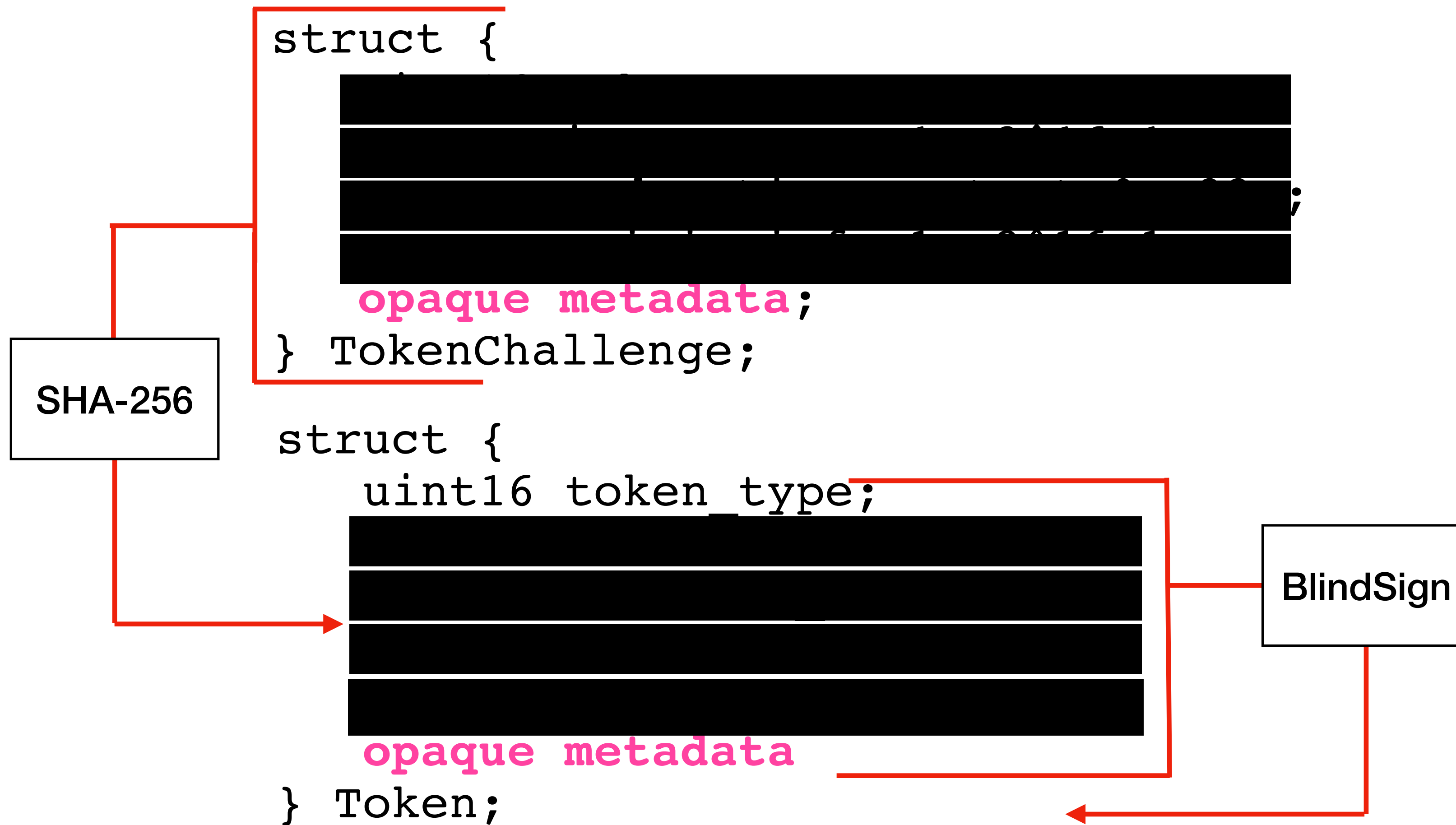
Privacy Pass Tokens with Public Metadata

Origin's perspective



Privacy Pass Tokens with Public Metadata

Issuer's perspective



Extensions as Metadata

draft-hendrickson-privacypass-public-metadata-issuance describes two issuance protocols that support public metadata

- POPRF variant

- Partially-blind RSA variant

draft-wood-privacypass-auth-scheme-extensions specifies a structure for metadata so different values can be TLV-encoded during issuance

Extensions

```
struct {  
    ExtensionType extension_type;  
    opaque extension_data<0..216-1>;  
} Extension;
```

```
struct {  
    reserved(0),  
    (65535)  
} ExtensionType;
```

```
struct {  
    Extension extensions<0..216-1>;  
} Extensions;
```

HTTP Authentication Extensions Parameter

Extensions are only present during the redemption path, so:

```
struct { ... } Token;
```

Authorization: PrivateToken token="abc.." extensions="def.."

```
struct { ... } Extensions;
```

Expiration Extension

The expiration extension is used to encode a token expiration

```
struct {
    uint64 timestamp_precision; // granularity
    uint64 timestamp;          // expiration timestamp
} ExpirationTimestamp;
```

Example timestamp for 1688583600, i.e., July 05, 2023 at 19:00:00 GMT+0000:

```
struct {
    uint64 timestamp_precision = 3600; // nearest hour
    uint64 timestamp = 1688583600;     // timestamp value
} ExpirationTimestamp;
```

Technical Questions

Should origins be able to ask clients to provide certain extensions?

Consider the interaction between **Accept-CH** and **Sec-CH** headers

Should we adopt QUIC-style encoding for extensions? Should any other encoding details change? (Bike sheds go here)

Privacy guardrails are determined by deployment, but should we strive for a better answer?

Procedural Questions

Is there interest in implementing and deploying these extensions?

If so, is the WG interested in adopting this series of work?