@justin _ _ richer

https://bspk.io/

Fixing OAuth 2

- Avoid front channel
- Don’t assume access to web browsers
- Anticipate mobile, SPA, and other ephemeral clients
- Learn from OAuth 2’s vast deployment

Goals of XYZ

- Take advantage of today's technologies
- Allow rich resource requests
- Allow flexibility with interaction modes
- Allow flexibility with key presentation
- Allow OAuth 2 use cases but don't repeat it exactly

Polymorphic JSON

- Values can be of different types
  - "foo": "bar"
  - "foo": true
  - "foo": ["bar", "baz"]
  - "foo": {"bar": "baz"}

Philosophy of XYZ

- Simple and common use cases should be easy
  - Especially for client developers
- Extension points need to be clearly described
- Every element has a clear model and representation
- Close ties to underlying systems (HTTP, JSON)
- It's only real if it's implemented

Status

- Implementations in Java and Node.js
  - Redirect client
  - User code client
  - AS (transaction, interaction, and user code)
  - Several signing methods (JWS-D, DPoP, PoP, Cavage)
- ID in datatracker
- More full writeup on project website

@justin _ _ richer

https://bspk.io/

XYZ: Transactional Authorization

The proposal

OAuth 2 is one of the most successful security protocols in use today. Even so, in its wide use, the protocol has come up against some of its own limitations. This is a proposal for a transactional authorization protocol XYZ to address the things that OAuth 2 doesn't handle well on its own. Optimizations and decisions that made sense in OAuth 2 don't make as much sense today. We are in a different world, and perhaps it's time we started to look to a different protocol.

https://oauth.xyz/

But we're not trying to invent something in a vacuum. In particular, OAuth 2 has been extended to cover a wide variety of client applications, deployments, and use cases. While this flexibility and extensibility is one of OAuth 2's strengths, this has unfortunately led to a number of components that are almost--but not quite--solving the same problems in similar ways. PKCE, UMA, CIBA, OBUK, FAPI, and a host of other extensions to OAuth 2 make use of temporary credentials to let the protocol behave in new ways.

This protocol, which I'm calling "XYZ" for the moment, is based on a transactional model. The client of the API declares who it is and what it wants, the AS figures out what information it needs to fulfill that (which might include interacting with a user), and ultimately a token is produced. All along the way, components have the opportunity to bind keys to different parts of the transaction so that attackers can't take over. This intent-based system takes in experience and feedback from other similar projects and protocols, but in a way that pulls together many different aspects.

The XYZ protocol is *not intended to be directly compatible with OAuth 2*, much in the same way that OAuth 2 was not directly compatible with OAuth 1. However, the concepts and many of the goals should feel familiar to

Start a Transaction

Here's who I am and what I want...

Client → AS


```
{
    "resources": [
        {
            "actions": [
                "read",
                "write",
                "dolphin"
            ],
            "locations": [
                "https://server.example.net/",
                "https://resource.local/other"
            ],
            "datatypes": [
                "metadata",
                "images"
            ]
        }
    ],
    "keys": {
        "proof": "httpsig",
        "jwks": {
            "keys": [
                {
                    "kty": "RSA",
                    "e": "AQAB",
                    "kid": "xyz-1",
                    "alg": "RS256",
                    "n": "kOB5rR4Jv0GMeLaY6_It_r3ORwdf8ci_JtffXyaSx8xYJCCNaOKNJn_Oz0YhdHbXTeWO5Aoys
                }
            ]
        }
    },
    "interact": {
        "redirect": true,
        "callback": {
            "uri": "https://client.example.net/return/123455",
            "nonce": "LKLTI25DK82FX4T4QFZC"
        }
    },
    "display": {
        "name": "My Client Display Name",
        "uri": "https://example.net/client"
    }
}
```

```
{
    "resources": [
        {
            "actions": [
                "read",
                "write",
                "dolphin"
            ],
            "locations": [
                "https://server.example.net/",
                "https://resource.local/other"
            ],
            "datatypes": [
                "metadata",
                "images"
            ]
        }
    ]
}
```

```
{
    "display": {
        "name": "My Client Display Name",
        "uri": "https://example.net/client"
    }
}
```

```
{
    "keys": {
        "proof": "jwsd",
        "jwks": {
            "keys": [
                {
                    "kty": "RSA",
                    "e": "AQAB",
                    "kid": "xyz-1",
                    "alg": "RS256",
                    "n": "kOB5rR4Jv0GMeLaY6_It_r3ORwdf8ci_JtffXyaSx8xYJCCNaOKNJn_Oz0YhdHbXTeWO5Aoys
                }
            ]
        }
    }
}
```

Proving Key Posession

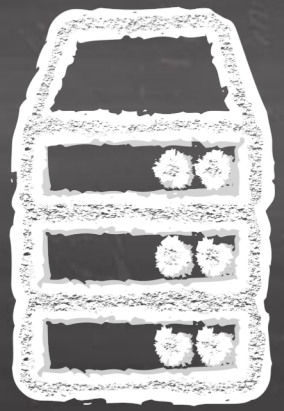
Client → AS

```
POST /transaction HTTP/1.1
Host: server.example.com
Content-Type: application/json
Detached-JWS: eyJiNjQiOmZhbHNlLCJhbGciOiJSUzI1NiIsImtpZCI6Inh5ei0xIn0..Y287HMtaY0EegEjoTd_04a4GC6qV

{
    "client": {
        "name": "My Client Display Name",
        "uri": "https://example.net/client"
    },
    "resources": [
        "dolphin-metadata"
    ],
    "interact": {
        "redirect": true,
        "callback": {
            "uri": "https://client.foo",
            "nonce": "VJLO6A4CAYLBXHTR0KRO"
        }
    },
    "keys": {
```

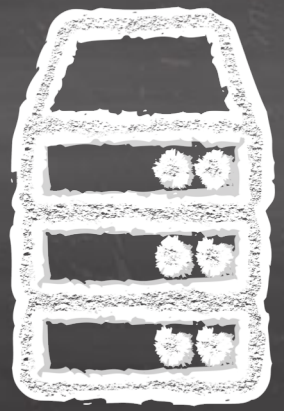
Client → AS

```
POST /transaction HTTP/1.1
Host: server.example.com
Content-Type: application/json
Content-Length: 716
Signature: keyId="xyz-client", algorithm="rsa-sha256", headers="(request-target) digest content-ler
Digest: SHA=oZz203kg5SEFAhmr0xEBbc4jEfo=

{
    "client": {
        "name": "My Client Display Name",
        "uri": "https://example.net/client"
    },
    "resources": [
        "dolphin-metadata"
    ],
    "interact": {
        "redirect": true,
        "callback": {
            "uri": "https://client.foo",
            "nonce": "VJLO6A4CAYLBXHTR0KRO"
        }
    },
    "keys": {
```

Maybe we can already issue an access token



Or:
"I need to talk to the user"



Client → AS

```
{
    "interact": {
        "redirect": true,
        "callback": {
            "uri": "https://client.example.net/return/123455",
            "nonce": "LKLTI25DK82FX4T4QFZC"
        },
        "user_code": true,
        "didcomm": true,
        "didcomm_query": true
    }
}
```

Interaction: Redirect With Callback

```
{
    "interact": {
        "redirect": true,
        "callback": {
            "uri": "https://client.example.net/return/123455",
            "nonce": "LKLTI25DK82FX4T4QFZC"
        }
    }
}
```

AS → Client


```
{
    "interaction_url": "https://server.example.com/interact/4CF492MLVMSW9MKMXKHQ",
    "server_nonce": "MBDOFXG4Y5CVJCX821LH",
    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```

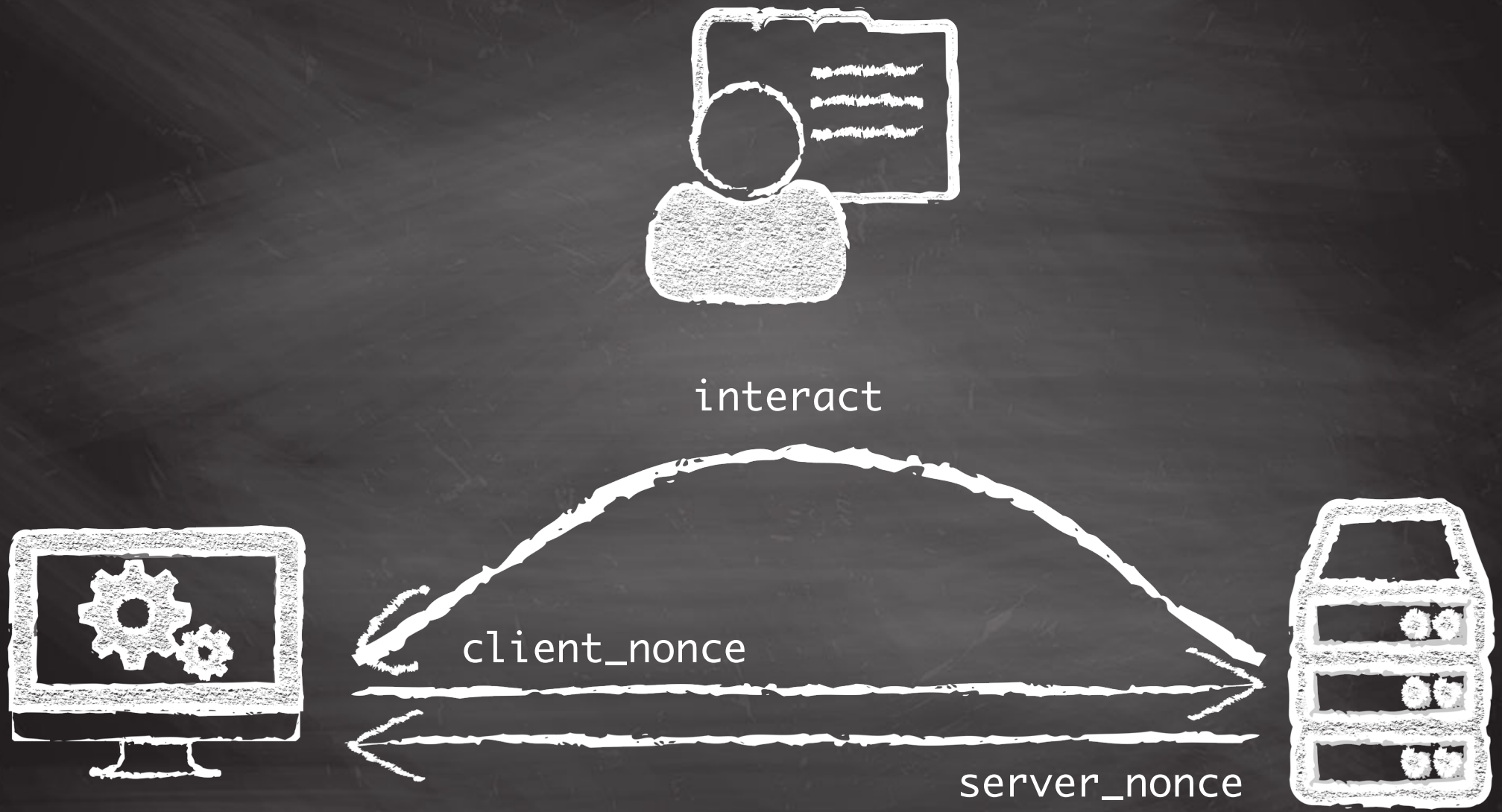
Client → Browser

```
HTTP 302 Found
Location: https://server.example.com/interact/4CF492MLVMSW9MKMXKHQ
```

AS → Browser

```
HTTP 302 Found
Location: https://client.foo/
  ?hash=p28jsq0Y2KK3WS__a42tavNC64ldGTBroywsWxT4md_jZQ1R2HZT8BOWYHcLmObM7XHPAdJzTZMtKBsaraJ64A
  &interact=4IFWWIKYBC2PQ6U56NL1
```

Connect the legs of the triangle



Client → AS

```
{
    "handle": "80UPRY5NM33OMUKMKSKU",
    "interact_ref": "4IFWWIKYBC2PQ6U56NL1"
}
```

Interaction: User Code

```
{
    "interact": {
        "user_code": true
    }
}
```

AS → Client


```
{
    "user_code": {
        "url": "https://server.example.com/interact/device",
        "code": "A1BC-3DFF"
    },
    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```

Client → AS

```
{
    "handle": "80UPRY5NM33OMUKMKSKU"
}
```

AS → Client


```
{
    "wait": 30,
    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```

Interaction: Scannable Code

```
{
    "interact": {
        "redirect": true,
        "user_code": true
    }
}
```

AS → Client


```
{
    "interaction_url": "https://server.example.com/interact/4CF492MLVMSW9MKMXKHQ",
    "user_code": {
        "url": "https://server.example.com/interact/device",
        "code": "A1BC-3DFF"
    },
    "handle": {
        "value": "80UPRY5NM33OMUKMKSKU",
        "type": "bearer"
    }
}
```


Access Tokens

```
{
    "access_token": {
        "value": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
        "type": "bearer"
    }
}
```

```
{
    "access_token": {
        "value": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
        "jwks": {
            "keys": [
                {
                    "kty": "RSA",
                    "e": "AQAB",
                    "kid": "xyz-1",
                    "alg": "RS256",
                    "n": "kOB5rR4Jv0GMeLaY6_It_r3ORwdf8ci_JtffXyaSx8xYJCCNaOKNJn_Oz0YhdHbXTeWO5Aoys
                }
            ]
        }
    }
}
```

```
{
    "resources": {
        "token1": [
            {
                "actions": [
                    "read",
                    "write",
                    "dolphin"
                ],
                "locations": [
                    "https://server.example.net/",
                    "https://resource.local/other"
                ],
                "datatypes": [
                    "metadata",
                    "images"
                ]
            }
        ],
        "token2": [
            {
                "actions": [
                    "foo",
                    "bar",
                    "dolphin"
                ],
                "locations": [
                    "https://resource.other/"
                ],
                "datatypes": [
                    "data",
                    "pictures"
                ]
            }
        ]
    }
}
```

```
{
    "multiple_access_tokens": {
        "token1": {
            "value": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
            "type": "bearer"
        },
        "token2": {
            "value": "UFGLO2FDAFG7VGZZPJ3IZEMN21EVU71FHCARP4J1",
            "type": "bearer"
        }
    }
}
```

Handles

Client → AS

```
{
    "resources": [
        "dolphin-metadata"
    ],
    "keys": "7C7C4AZ9KHRS6X63AJAO",
    "display": "VBUEOIQA82PBY2ZDJW7Q"
}
```

```
{
    "resources": [
        "dolphin-metadata"
    ]
}
```

```
{
    "resources": [
        {
            "actions": [
                "read",
                "write",
                "dolphin"
            ],
            "locations": [
                "https://server.example.net/",
                "https://resource.local/other"
            ],
            "datatypes": [
                "metadata",
                "images"
            ]
        },
        "dolphin-metadata"
    ]
}
```

AS → Client


```
{
    "display_handle": {
        "value": "VBUEOIQA82PBY2ZDJW7Q",
        "type": "bearer"
    },
    "user_handle": {
        "value": "XUT2MFM1XBIKJKSDU8QM",
        "type": "bearer"
    },
    "resources_handle": {
        "value": "KLKP36N7GPOKRF3KGH5N",
        "type": "bearer"
    },
    "key_handle": {
        "value": "7C7C4AZ9KHRS6X63AJAO",
        "type": "bearer"
    },
    "claims_handle": {
        "value": "14XF3WKRPKW4RN9AROOC",
        "type": "bearer"
    }
}
```

Identity

```
{
    "claims": {
        "subject": true,
        "email": true,
        "phone": true,
        "auth_time": true
    }
}
```

AS ➡ Client


```
{
    "claims": {
        "subject": "I6W52R97IH",
        "email": "user@example.com",
        "phone": "555-USER",
        "updated_at": "2020-01-01T12:43:29+0000",
        "auth_time": "2020-02-17T21:23:39+0000"
    }
}
```

AS → Client

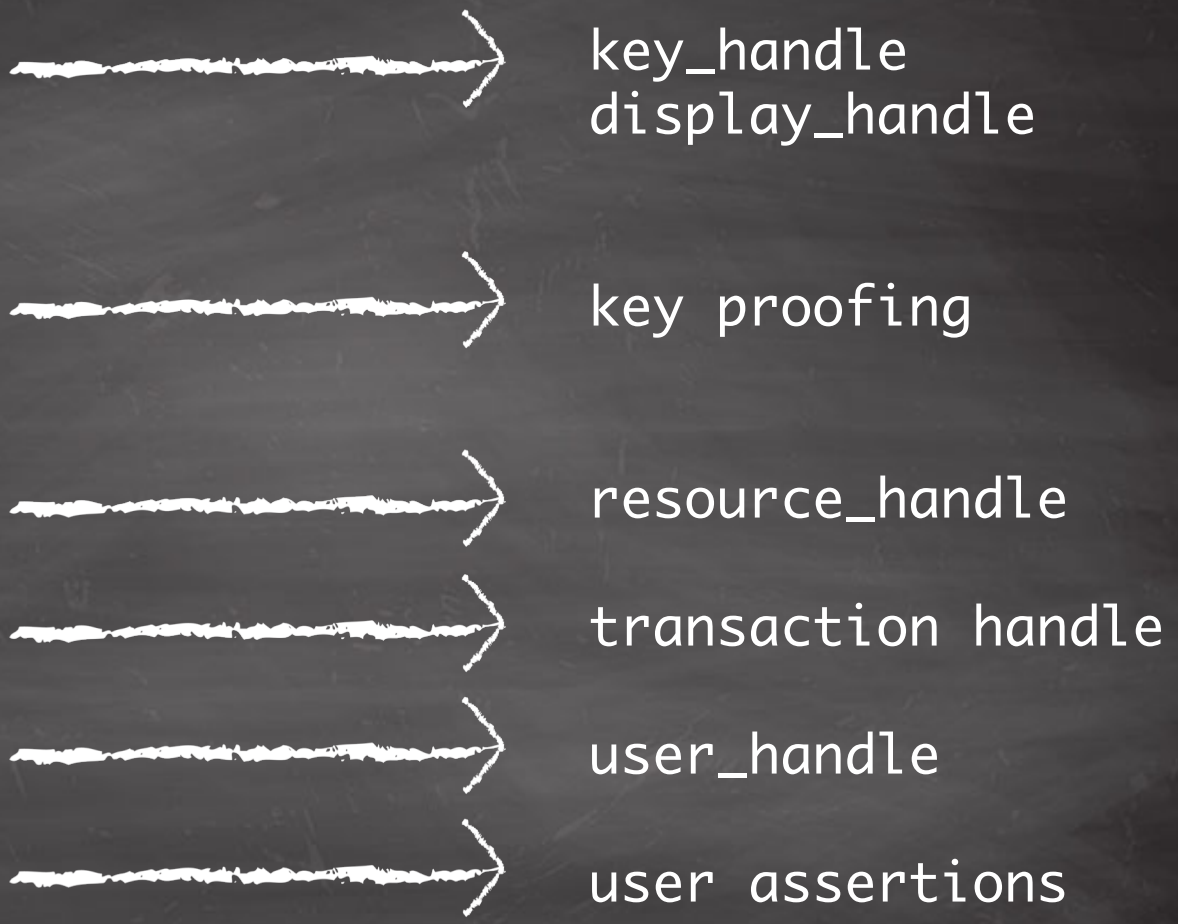

```
{
    "access_token": {
        "value": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
        "type": "bearer"
    },
    "claims": {
        "subject": "I6W52R97IH",
        "email": "user@example.com",
        "phone": "555-USER",
        "updated_at": "2020-01-01T12:43:29+0000",
        "auth_time": "2020-02-17T21:23:39+0000"
    }
}
```

Client → AS


```
{
    "user": {
        "assertion": "eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUlMyNTYifQ.ewogImlzcyI6ICJodHRwOi8vc2VydmVyLm
        "type": "oidc_id_token"
    }
}
```

Mapping concepts to OAuth 2

Mapping to OAuth2



Mapping to OAuth2

```
{
    "display": "client_id"
    "resources": ["scope1", "scope2"],
    "key": "client_id"
}
```

Porting to/from OAuth 2

- PAR + RAR + JAR + JARM
- DPoP + PoP + HTTPSig + MTLS
- Auth Code, Device, Exchange, Refresh, Assertion, CIBA, OB/FAPI, Client Credentials, and UMA flows
- PKCE + State + Nonce
- This is unwieldy at best…

XYZ: Transactional Authorization

The proposal

OAuth 2 is one of the most successful security protocols in use today. Even so, in its wide use, the protocol has come up against some of its own limitations. This is a proposal for a transactional authorization protocol XYZ to address the things that OAuth 2 doesn't handle well on its own. Optimizations and decisions that made sense in OAuth 2 don't make as much sense today. We are in a different world, and perhaps it's time we started to look to a different protocol.

https://oauth.xyz/

But we're not trying to invent something in a vacuum. In particular, OAuth 2 has been extended to cover a wide variety of client applications, deployments, and use cases. While this flexibility and extensibility is one of OAuth 2's strengths, this has unfortunately led to a number of components that are almost--but not quite--solving the same problems in similar ways. PKCE, UMA, CIBA, OBUK, FAPI, and a host of other extensions to OAuth 2 make use of temporary credentials to let the protocol behave in new ways.

This protocol, which I'm calling "XYZ" for the moment, is based on a transactional model. The client of the API declares who it is and what it wants, the AS figures out what information it needs to fulfill that (which might include interacting with a user), and ultimately a token is produced. All along the way, components have the opportunity to bind keys to different parts of the transaction so that attackers can't take over. This intent-based system takes in experience and feedback from other similar projects and protocols, but in a way that pulls together many different aspects.

The XYZ protocol is *not intended to be directly compatible with OAuth 2*, much in the same way that OAuth 2 was not directly compatible with OAuth 1. However, the concepts and many of the goals should feel familiar to