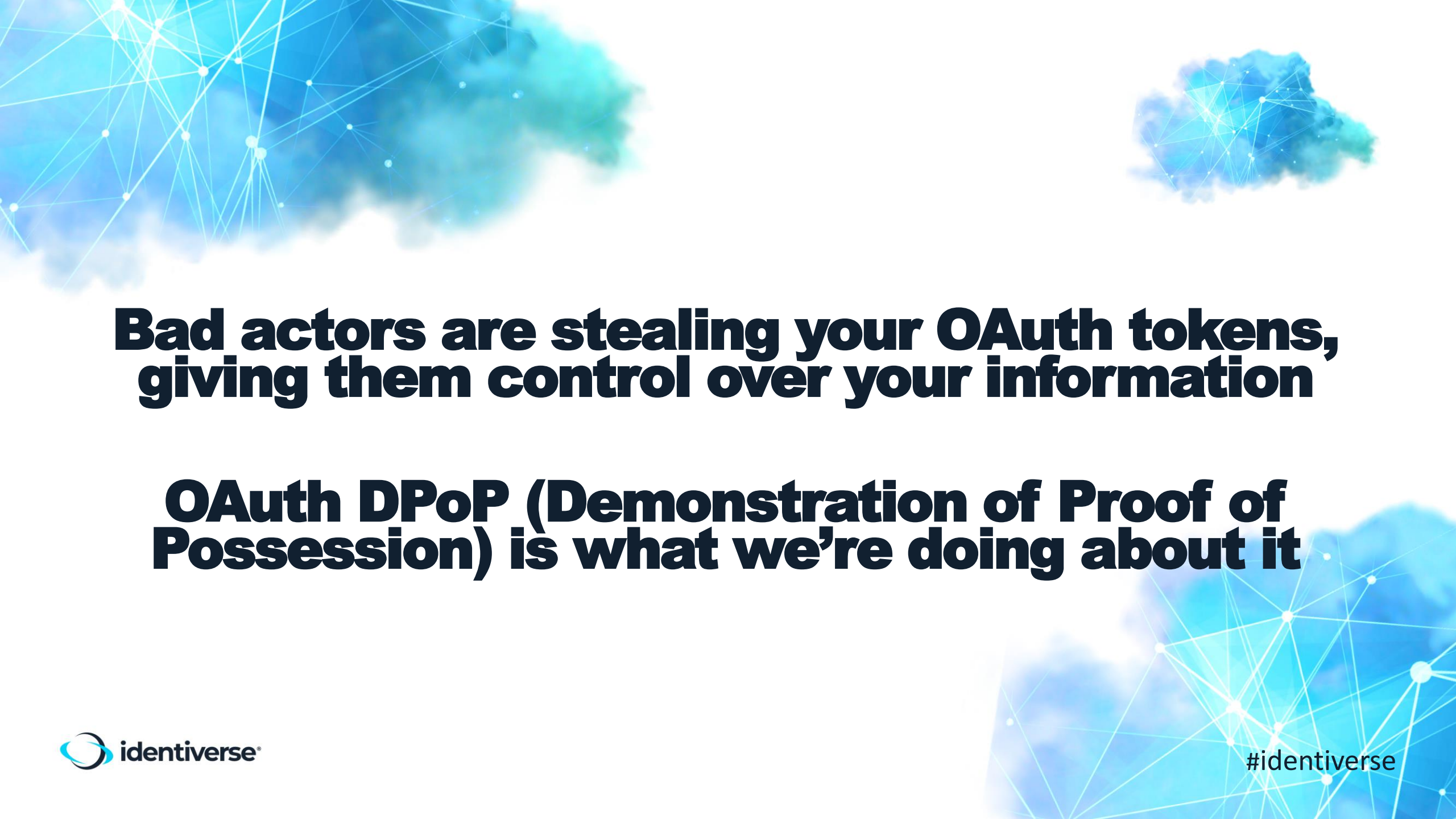




**Bad actors are stealing your OAuth tokens,
giving them control over your information**

**OAuth DPoP (Demonstration of Proof of
Possession) is what we're doing about it**



**Michael B.
Jones**

Identity Standards Architect
Microsoft



**Pieter
Kasselmann**

Identity Standards Architect
Microsoft



Come Together

In the next 25 minutes...

Token Theft is Real (Mike)

Proof of Possession to the Rescue (Mike)

- The Journey We've Been On
- Where We Are Today (DPoP)

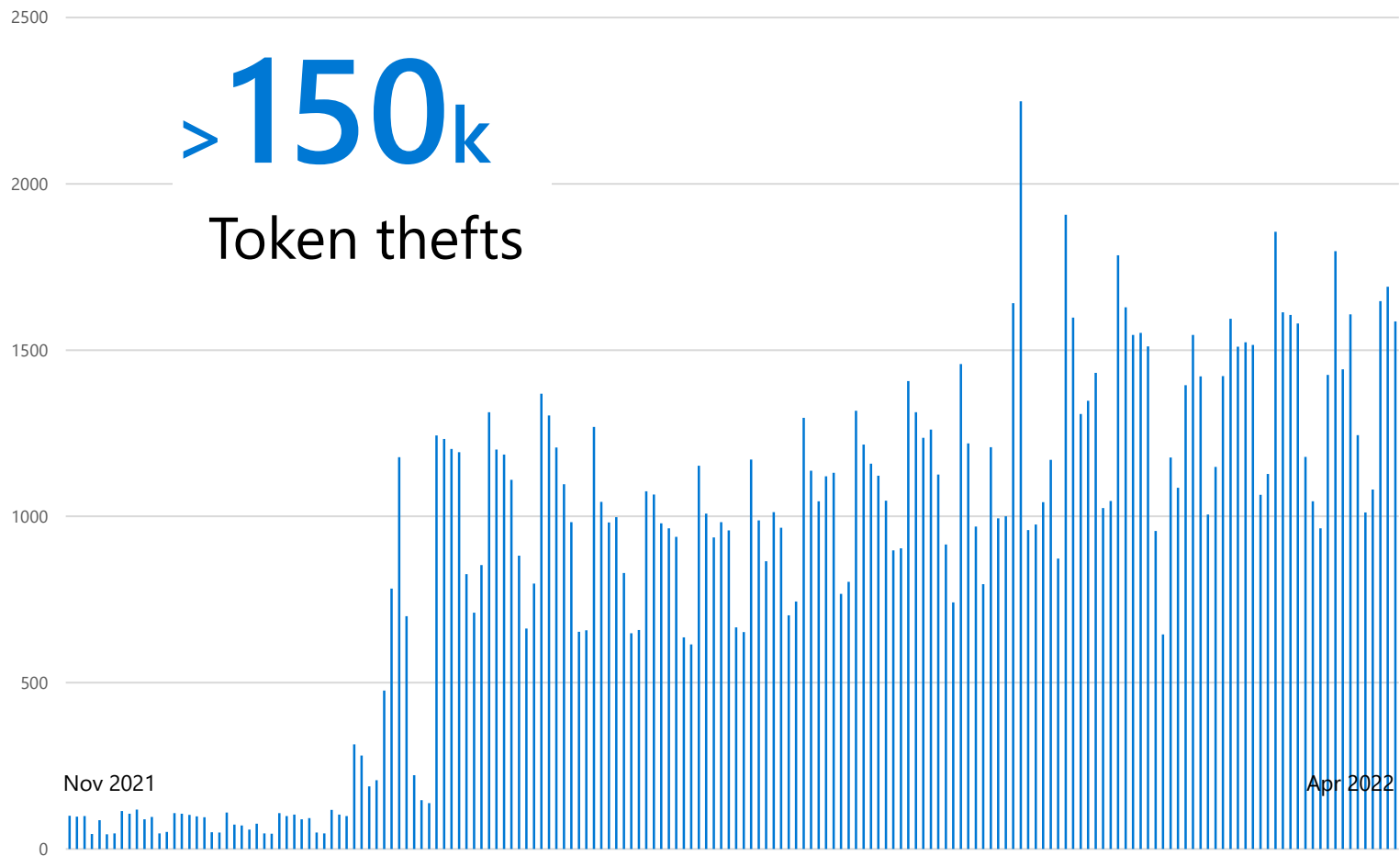
Two New Attacks (and how DPoP thwarts them) (Pieter)

- Pre-computed Proofs and the Server-Supplied Nonce
- Authorization Code Rebinding

Call to Action (Pieter)

Token Theft is Real

Detected over a six-month period



■ Token thefts detected

>1.7_M

Device malware infections



Content courtesy of @Alex_T_Weinert

OAuth 2.0 Tokens Typically Bearer Tokens

A Bearer Token is like cash

- If you have it, you can use it
- Even if you stole it

Theft of bearer tokens has become a significant problem



Proof-of-Possession Tokens a Safer Alternative

A Proof-of-Possession (PoP) Token is like your driver's license

- Only you can use it
- Even if someone steals it, they can't use it
- Using it requires proof that it's yours

The photo on your license is used for the proof



OAuth 2.0 Token Standards to Date

- Use of Bearer Tokens is defined by RFC 6750
- Currently, there's no OAuth 2.0 standard for non-bearer (proof-of-possession) tokens

[RFC Home] [TEXT] [PDF] [HTML] [Tracker] [IPR] [Errata] [Info page]	
Updated by: 8996	PROPOSED STANDARD
Internet Engineering Task Force (IETF)	Errata Exist
Request for Comments: 6750	M. Jones
Category: Standards Track	Microsoft
ISSN: 2070-1721	D. Hardt
	Independent
	October 2012

The OAuth 2.0 Authorization Framework: Bearer Token Usage

Abstract

This specification describes how to use bearer tokens in HTTP requests to access OAuth 2.0 protected resources. Any party in possession of a bearer token (a "bearer") can use it to get access to the associated resources (without demonstrating possession of a cryptographic key). To prevent misuse, bearer tokens need to be protected from disclosure in storage and in transport.

You Can't Do That (Proof of Possession to the Rescue)

What is Proof of Possession (PoP)?

What

- PoP demonstrates possession of cryptographic key material when performing an operation

How

- Typically with a signature in conjunction with a token that contains or references the PoP key used to sign

Why

- Preventing use of a leaked/stolen token

Magical Mystery Tour

- Proof-of-Possession (PoP)
- Holder-of-Key (HoK)
- Sender-Constrained
- Key-Constrained
- Certificate-Bound
- Key-Bound
- Key Confirmation
- etc.

The Long and Winding Road

- SAML 2.0 Holder-of-Key Assertion Profile – Not exactly OAuth
- OAuth 1.0 used PoP – But message signing too complicated
- OAuth 2.0 MAC draft – Used similarly complicated signing
- OAuth 2.0 HTTP Signing draft – Abandoned due to complexity
- TLS Token Binding – Some browsers declined to ship it
- OAuth 2.0 Mutual TLS – Client certs notoriously difficult to use *bringing us to...*
- OAuth 2.0 DPoP – Our current stab at solving this important problem

What is DPoP?

- Demonstrating Proof-of-Possession at the Application Layer
- Pragmatic application-level proof-of-possession mechanism for OAuth 2.0 access tokens and refresh tokens

The screenshot shows the IETF Data Tracker page for the document **draft-ietf-oauth-dpop-07**. The page title is "OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP)". The document is an "Active Internet-Draft (oauth WG)". The authors listed are Daniel Fett, Brian Campbell, John Bradley, Torsten Lodderstedt, Michael Jones, and David Waite. The last update was on 2022-04-04. The document replaces **draft-fett-oauth-dpop**. The stream is "Internet Engineering Task Force (IETF)". The intended RFC status is "(None)". The formats available are plain text, html, xml, htmlized, pdfized, and bibtex. The stream is "WG state" with a note: "WG Document (wg milestone: Jan 2022 - Submit 'OAuth 2.0 Pr...)". The document shepherd is Rifaat Shekh-Yusef.

The screenshot shows a Twitter thread by Brian Campbell (@_b_c) dated March 21. The first tweet says: "I found the right name for this on the S-Bahn last night @ve7jt #osw2019". It includes a photo of a poster for "DEUTSCHE POP Ausbildung & Studium" with a "DIPLOMA BACHELOR MASTER" badge. The poster features various images of people and text: "MUSIC & SOUND", "MARKETING & MANAGEMENT", "GAMES, FILM & PHOTOGRAPHY", "DESIGN, FASHION & MAKE-UP", and "FITNESS & LIFESTYLE". The second tweet says: "or DPoP with proper capitalization, which could also stand for Demonstrating Proof-of-Possession [at the application layer]". The third tweet says: "/cc @dfett42 @tlodderstedt".

DPoP Overview

- DPoP Proof JWT sent as an HTTP header
 - Demonstrates proof-of-possession in the context of the request
 - Sent the same way with (mostly) the same syntax and semantics for both:
 - token requests to the AS's token endpoint
 - protected resource requests
 - AS uses the proof to bind tokens
 - RS uses the proof to verify bound tokens

```
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkdVMjU2IiwiaWdrIjp7Imt0eSI6IkV  
VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUKRZ0XpDa0RscEJoRjQyVVFVZldwQVdCR  
nMiLCJ5IjoioVZFNGpmX09rX282NHpiVFRsY3V0SmFqSG10NnY5VERWclUwQ2R2R1JE  
QSIIsImNydiI6IiAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZzhY2MybFRjIiwiaHRtIj  
oiUE9TVCIsImh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia  
WF0IjoxNTYyMjYyNjE2fQ.2-GxA6T8lP4vfrg8v-FdWP0A0zdrj8igiMLvqRMUvwnQg  
4PtFLbdLXiOSsX0x7NVY-FNyJK70nfbV37xRZT3Lg
```



Anatomy of a DPoP Proof JWT

Explicitly typed

```
{  
  "typ": "dpop+jwt",
```

Asymmetric signature
algorithms only

```
  "alg": "ES256",
```

The public key to verify
the signature and
proof-of-possession is
being demonstrated

```
  "jwk":
```

```
  {
```

```
    "kty": "EC", "crv": "P-256",
```

```
    "x": "18tFrhx-34tV3hRICRDY9zCkDlpBhF42UQUfWVAWBFs",
```

```
    "y": "9VE4jf_0k_o64zbTT1cuNJajHmt6v9TDVrU0CdvGRDA"
```

```
  }
```

```
  }.
```

```
  {
```

```
    "jti": "-BwC3ESc6acc2lTc",
```

```
    "htm": "POST",
```

```
    "htu": "https://rs.example.com/important/stuff",
```

```
    "iat": 1637259115,
```

```
    "ath": "fUHy02r2Z3DZ53EsNrWBb0xWXoaNy59IiKCAqksmQEo",
```

```
    "nonce": "aiJ6sfh_zG.e.yffs0Z-Hv4w-7v"
```

```
  }
```

Unique identifier for
replay checking

Minimal info about
the HTTP request

Only valid for a limited
time window relative
to creation time

Hash of the access token
(only for protected
resource access)

Server-provided nonce
value (when previously
provided by the server)
for freshness & time

Access Token Request

POST /token HTTP/1.1

Host: server.example.com

Content-Type: application/x-www-form-urlencoded

DPoP: eyJ0eXAiOiJKcG9wK2p3dCI6ImFsZyI6IkdVMTJ0IiwiaWdrIjp7Imt0eSI6IkV
VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
nMiLCJ5IjoioVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
QSI6ImNydiI6IiAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIj
oiUE9TVCI6Imh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia
WF0IjoXNTYyMjYyNjE2fQ.2-GxA6T81P4vfrg8v-FdWP0A0zdrj8igiMLvqRMUvwnQg
4PtFLbdLXiOSsX0x7NVY-FNyJK70nfbV37xRZT3Lg

DPoP proof
JWT in HTTP
header

grant_type=authorization_code

&code=Sp1x10BeZQQYbYS6WxSbIA

&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb

&code_verifier=bEaL42izcC-o-xBk0K2vuJ6U-y1p9r_wW2dFWIWgjz-

Access Token Response

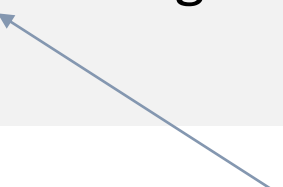
```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store
```

```
{
  "access_token": "Kz~8mXK1Ea1YznwH-LC-1fBAo.4Ljp~zsPE_Ne0.gxU",
  "token_type": "DPoP",
  "expires_in": 3626,
  "refresh_token": "Q..Zkm29lexi8VnWg2zPW1x-tgGad0Ibc3s3EwM_Ni4-g"
}
```

Token type
indicates that
the **access
token** is bound
to the DPoP
public key



Refresh token is bound to
the DPoP public key for a
public client



Refresh Token Request

POST /token HTTP/1.1

Host: server.example.com

Content-Type: application/x-www-form-urlencoded

DPoP: eyJ0eXAiOiJkcG9wK2p3dCI6ImFsZyI6IktVMjU2IiwiaWdrIjp7Imt0eSI6IkV
VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
nMiLCJ5IjoioVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
QSI6ImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIj
oiUE9TVCI6Imh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia
WF0IjoXNTYyMjY1Mjk2fQ.pAqut2IRDm_De6PR93SYmGBPxpwrAk90e8cP2hjiaG5Qs
GSuKDYW7_X620BxqhvYC8ynrrvZLTk41mSRroapUA

grant_type=refresh_token

&refresh_token=Q..Zkm29lexi8VnWg2zPW1x-tgGad0Ibc3s3EwM_Ni4-g

DPoP proof
JWT in HTTP
header



Protected Resource Request

GET /protectedresource HTTP/1.1

Host: resource.example.org

Authorization: DPoP Kz~8mXK1Ea1YznwH-LC-1fBAo.4Ljp~zsPE_Ne0.gxU

DPoP: eyJ0eXAiOiJkcG9wK2p3dCI6ImFsZyI6IkVTMjU2IiwiaWdrIjp7Imt0eSI6IkVDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUkRZOXpDa0RscEJoRjQyVVFVZlJdWQVdCRnMiLCJ5IjoioVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JEQSIsImNydiI6IiAtMjU2In19.eyJqdGkiOiJlMwozVl9iS2ljOC1MQUVCIiwiaHRtIjoiaR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZW50ZWRyZXNvdXJjZSIsIm1hdCI6MTU2MjI2MjYxOH0.1NhmpAX1WwmpBvwhok4E74kWCiGBNdavjLAeevGy32H3dbF0Jbri69Nm2ukkb-uyUI4AUg1JSskfWIyo4UCbQ

DPoP-bound
(reference style)
access token

Token is
bound to
the key
in the
proof

DPoP
proof

401 with WWW-Authenticate Challenge

Example Response to a Protected Resource Request Without a Token

```
HTTP/1.1 401 Unauthorized  
WWW-Authenticate: DPoP algs="ES256 PS256"
```

Example Response to a Protected Resource Request With an Invalid Token

```
HTTP/1.1 401 Unauthorized  
WWW-Authenticate: DPoP error="invalid_token",  
    error_description="Invalid DPoP key binding", algs="ES256"
```

Any Time at All

- Option for a server-provided nonce in the DPoP proof
 - “... DPoP-Nonce HTTP header in the response supplying a nonce value to be used when sending the subsequent request.”
 - Value is whatever the server chooses
 - Server contribution can ensure freshness of the proof JWT
 - Defeats pre-generation
 - Can help with significant client clock skew

Authorization Server nonce challenge

```
HTTP/1.1 400 Bad Request
DPoP-Nonce: 7c1P2Sczb-3yBH0-Z.Gv4w4xY
```

```
{
  "error": "use_dpop_nonce"
  "error_description": "AS needs nonce in DPoP proof"
}
```

Protected Resource nonce challenge

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: DPoP error="use_dpop_nonce",
  error_description="plz use nonce in next DPoP proof"
DPoP-Nonce: ab3d6eqG96ye0i0999Z_HX42t77x
```

Next nonce provided with successful response (no challenge)

```
HTTP/1.1 200 OK
DPoP-Nonce: l1m3n0peJ7S45zGPeYJbYuciQmJxj26
```

You Know My Name (Look Up The Number)

- Authorization Server Metadata
 - `dpop_signing_alg_values_supported`: Signals support for DPoP with the JWS `alg` values the AS supports for DPoP proofs
- Client Registration Metadata
 - `dpop_bound_access_tokens`: Indicates that the client always uses DPoP when requesting tokens from the authorization server

Binding an Authorization Code to a DPOP Key

- Enables end-to-end binding of the whole authorization flow
- Optional `dpop_jkt` authorization request parameter
 - SHA-256 JWK Thumbprint of the proof-of-possession public key
 - AS binds the issued authorization code to the thumbprint
 - AS checks the binding against the DPOP proof on code redemption
 - Can be used in conjunction with PKCE as-is
 - PKCE thwarts code injection
 - `dpop_jkt` thwarts authorization code rebinding
 - PAR can bind from the DPOP proof or `dpop_jkt` (need to be the same)

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&code_challenge=E9MeIhoa20wvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&dpop_jkt=NzbLsXh8uDCcd-6MNwXF4W_7noW XFZAfHkxZsRGC9Xs HTTP/1.1
Host: server.example.com
```

Ticket to Ride

- DPoP progressing on the journey to an RFC
- WGLC and shepherd review completed
- Promising, but potential challenges ahead remain
 - Adoption of new standards is never a certainty
 - Public key cryptography efficiency concerns
 - Optionality of some features
- Implementations appearing
- Endorsed by FAPI 2.0 draft as means of sender-constraining access tokens
- Interest from many parties and applications
- ***We're on the verge of an OAuth 2.0 PoP token standard!***



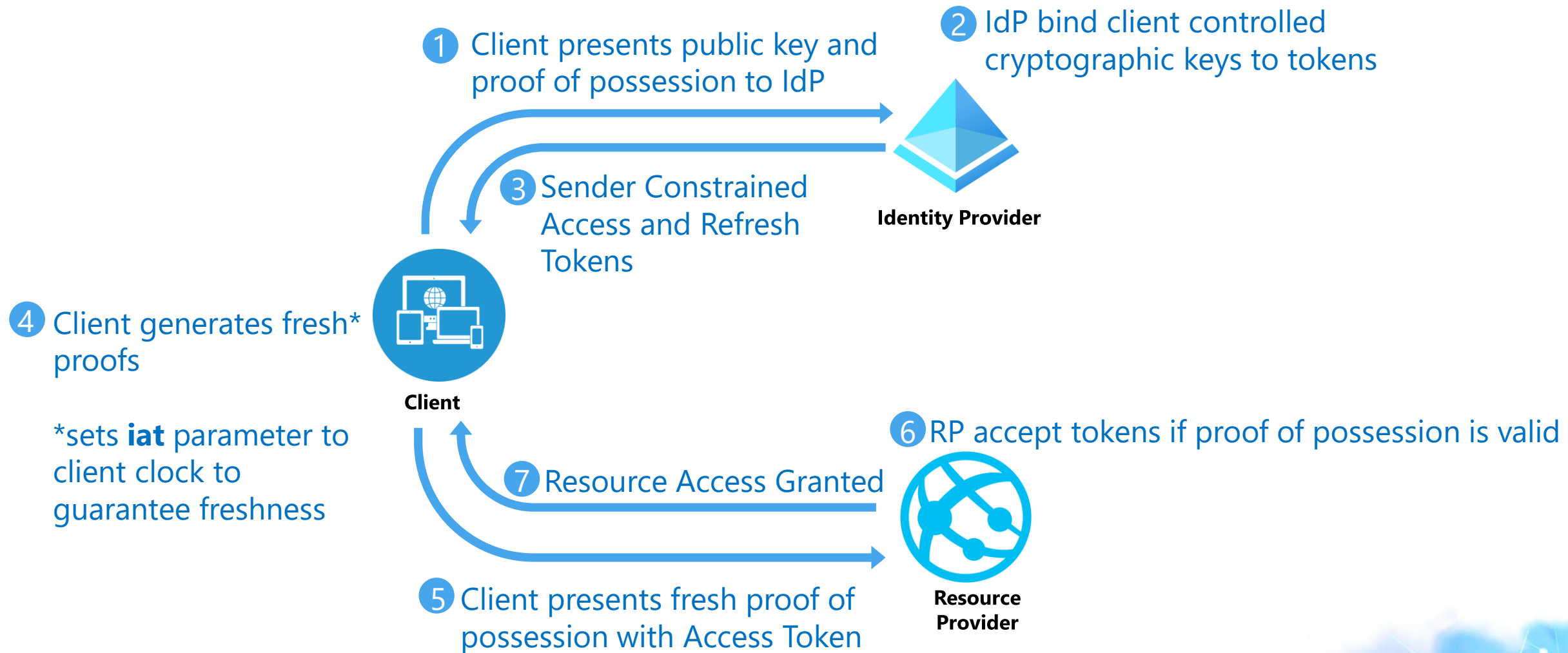
IETF 113
Vienna



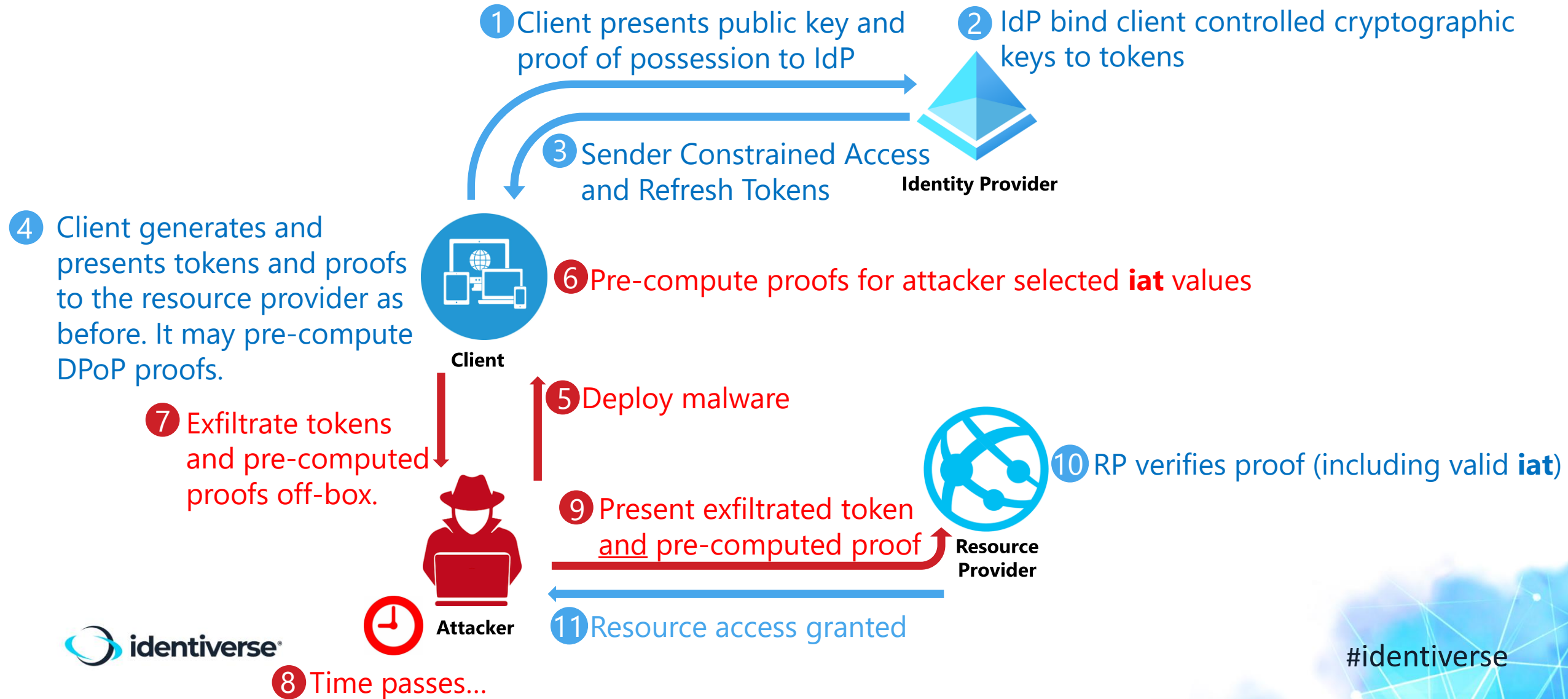
Two New Attacks (and how DPoP thwarts them)

Attack 1: Proof Pre-Computation Attack

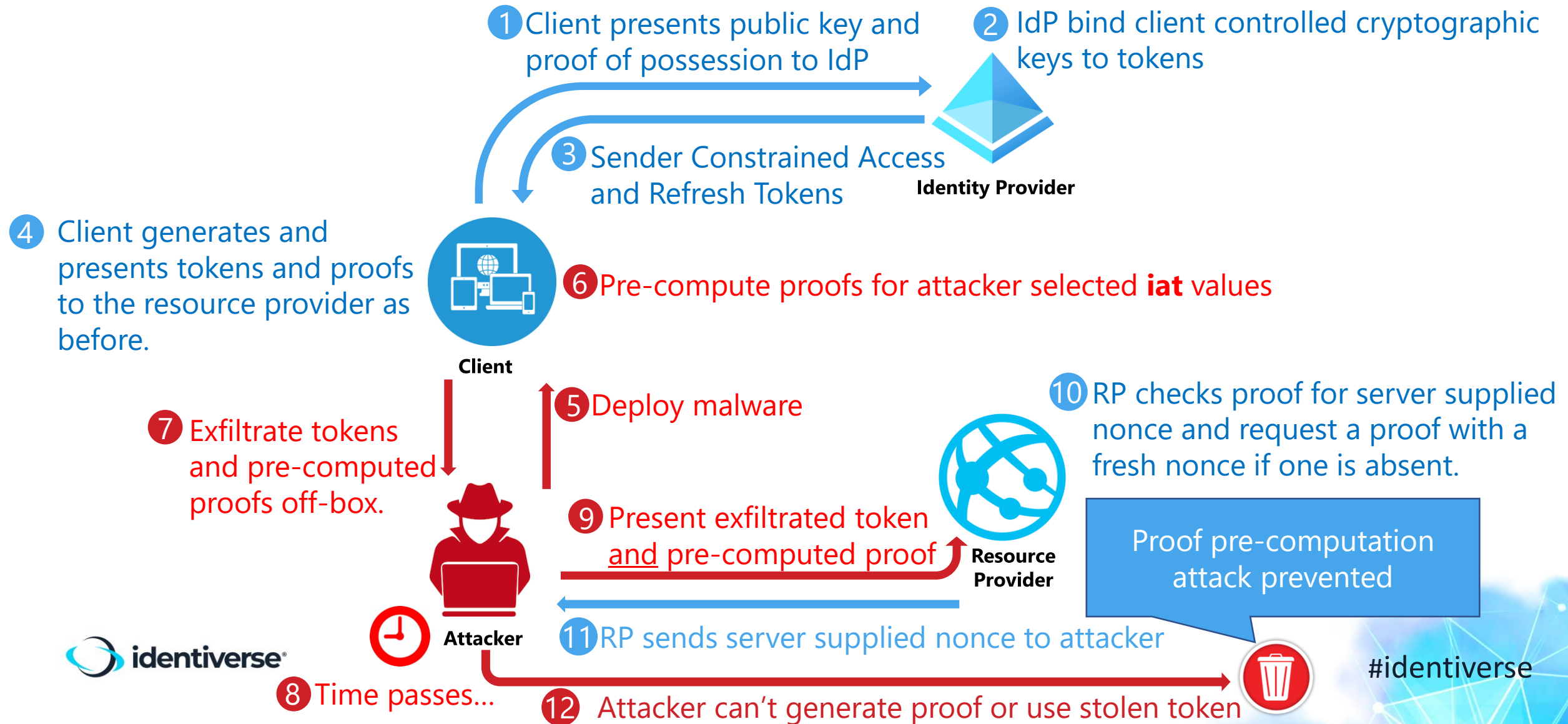
DPoP at a Glance



DPoP Proof Pre-computation Attack

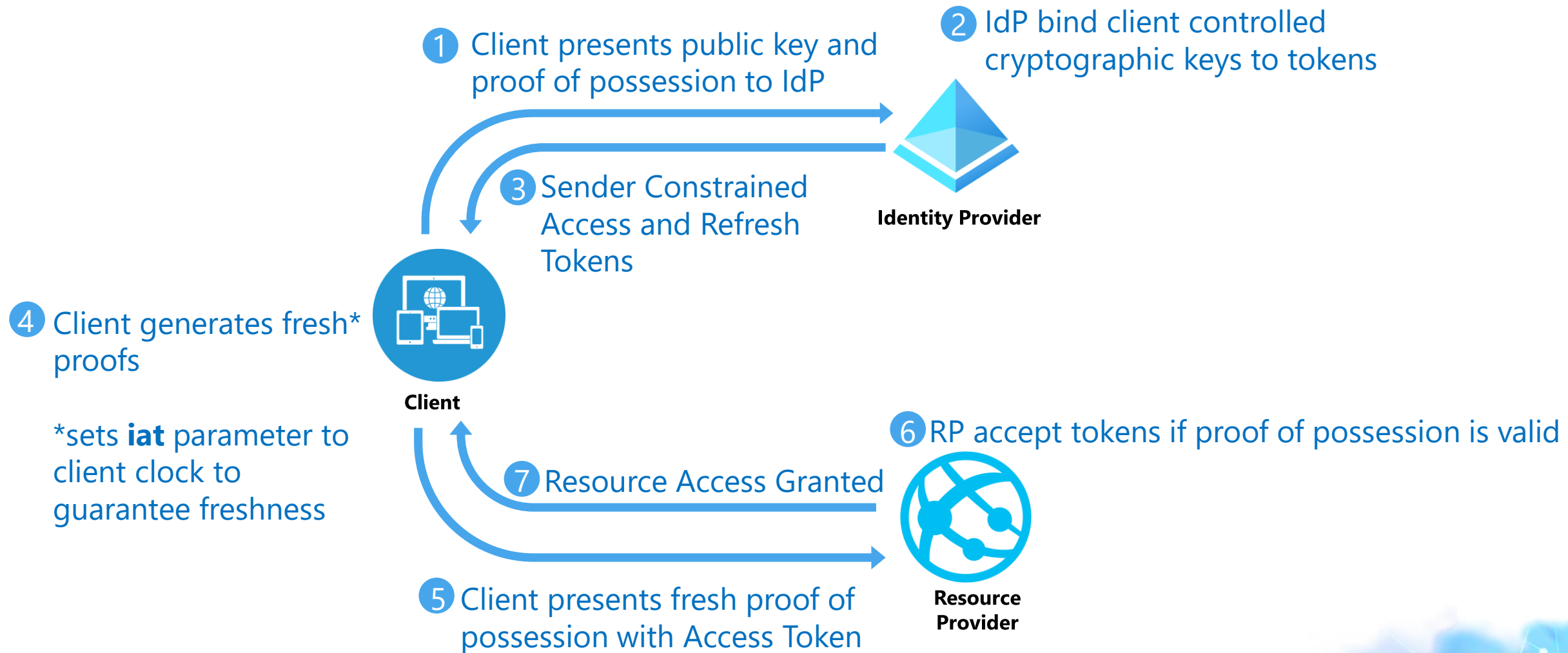


DPoP Proof Pre-computation Mitigation: Server Nonces

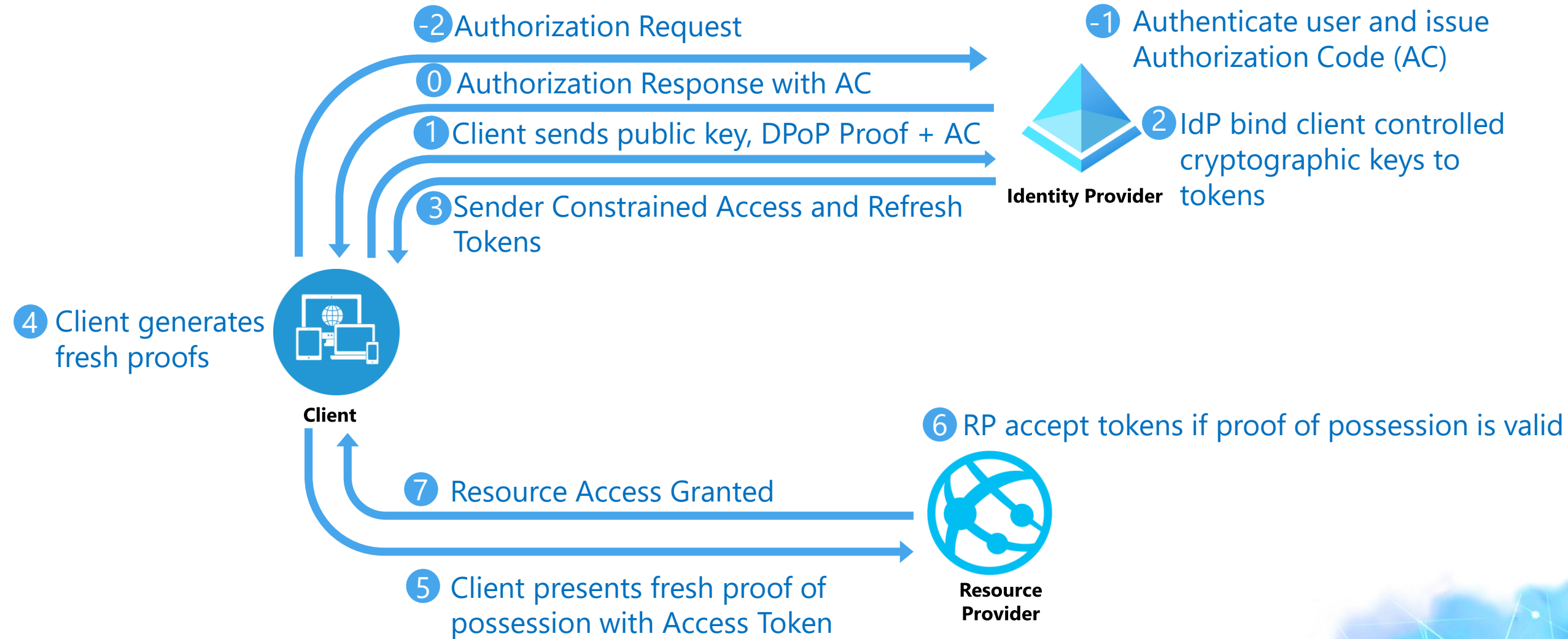


Attack 2: Authorization Code Rebinding Attack

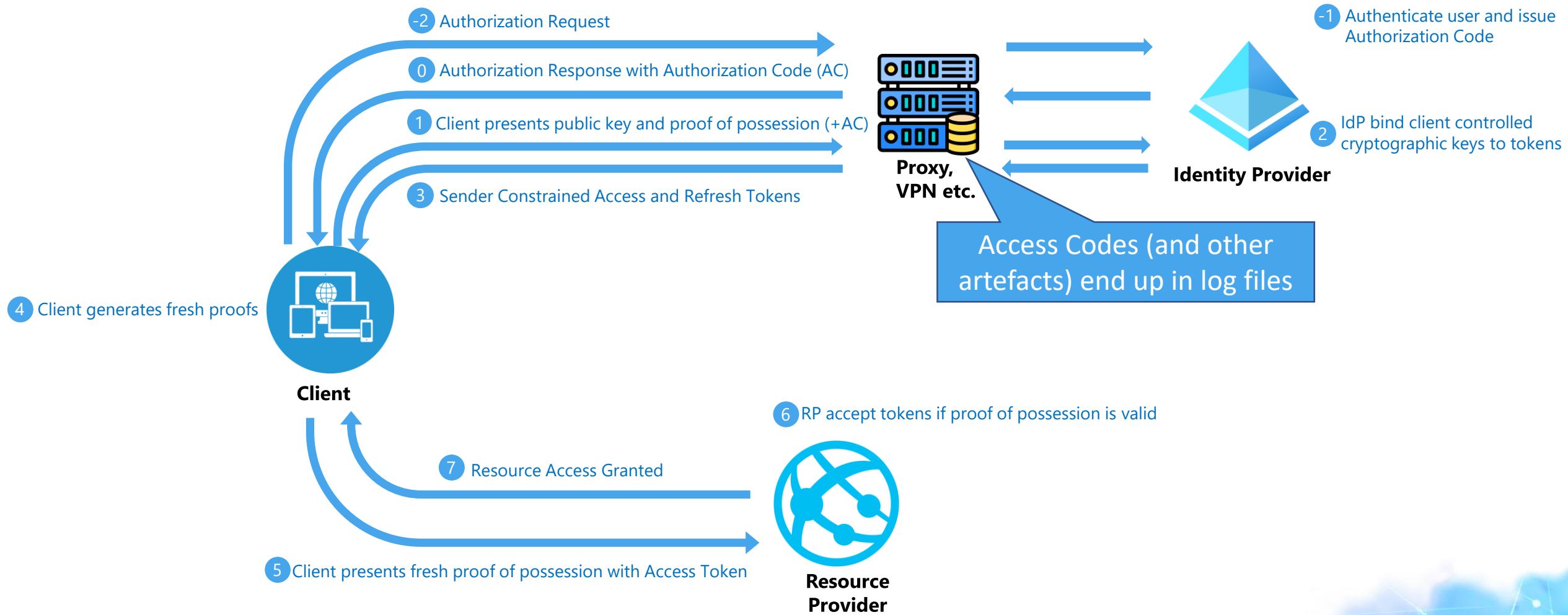
DPoP at a Glance



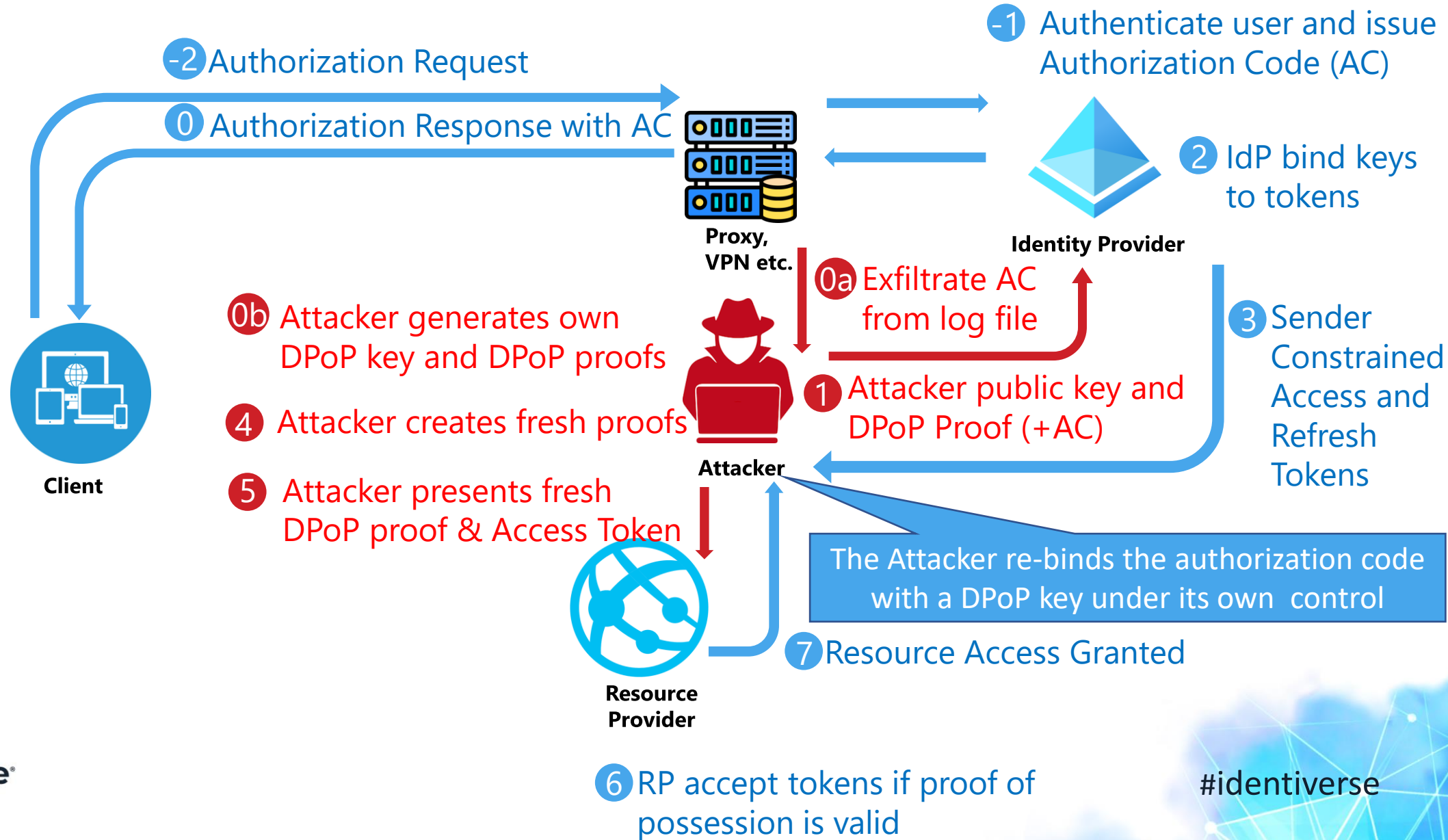
Let's take three steps back...



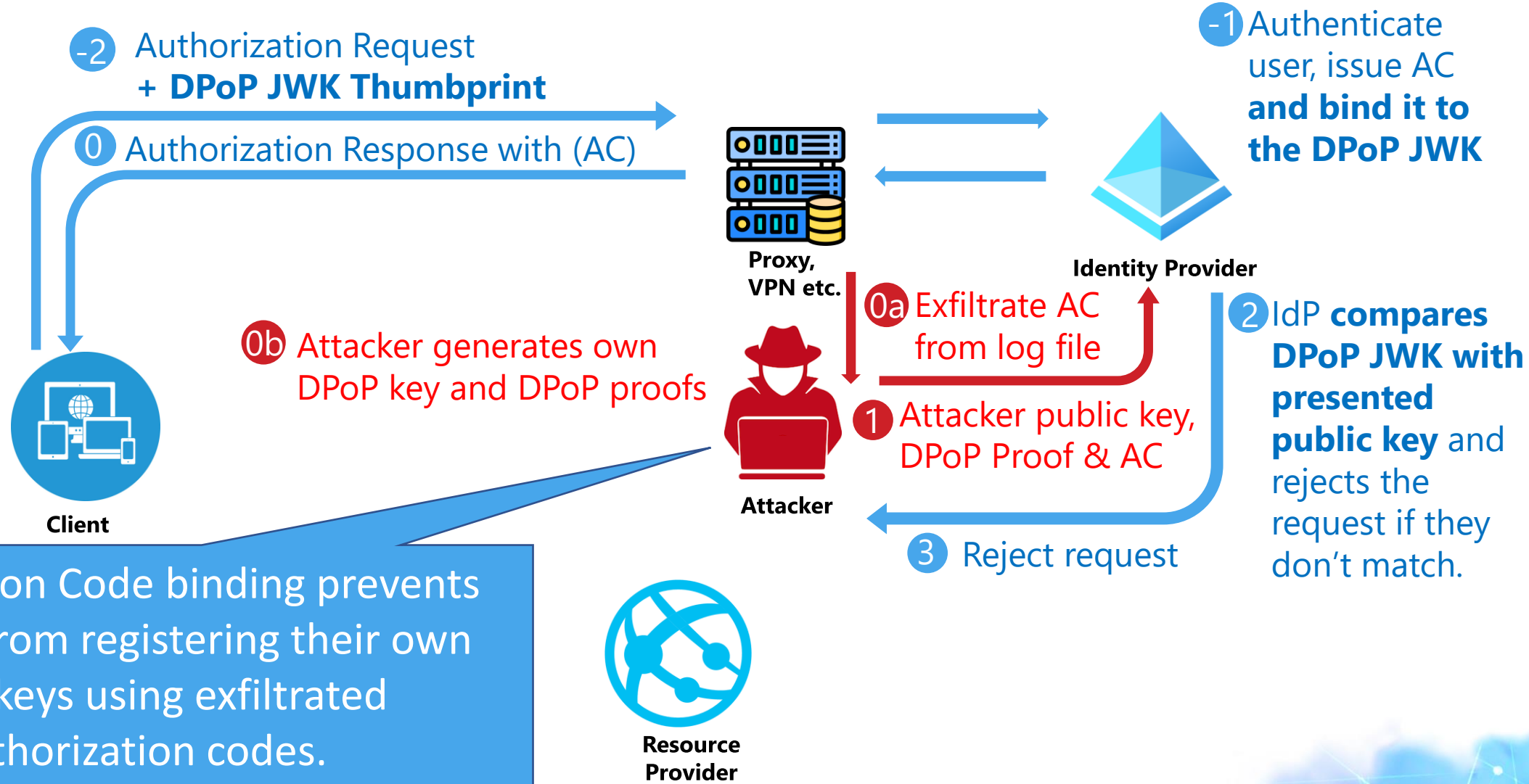
... and take a closer look...



Authorization Code Rebinding Attack



Authorization Code Rebinding Attack: Mitigation



Call to Action



Protect yourself from token theft with DPoP!

The background features three stylized blue clouds. Each cloud is overlaid with a network of white lines and dots, resembling a digital or data network. The clouds are positioned in the top-left, top-right, and bottom-right corners of the slide.

The End