Parâmetro state

Adiciona proteção cross-site

↳ Se um atacante aceder ao callback do cliente, o cliente envia o código (code) ao Authorization Server

⇒ o atacante pode obter authorization tokens ou esgotar recursos do cliente e do servidor, ou levar a que o cliente solicite um token que nunca pediu.

Ver utilização do parâmetro state no código do exemplo anterior

- outros fluxos de autorização (client credentials, password, implicit)
  ↳ ver diapositivos

Tokens estruturados: JSON Web Token (JWT)

No exemplo anterior, o Authorization Server (AS) e o Protected Resource (PR) comunicavam os Access Token (AT) (para efeitos de validação) através de uma base de dados partilhada.

O JWT é um tipo de token (estruturado) que contém toda a informação necessária dentro dele, permitindo que o AS comunique indiretamente com o PR através do token JWT.

↳ o JWT permanece opaco aos clientes

Estrutura dum JWT

Um JWT é um objeto JSON formatado de forma própria para ser transmitido na rede.

Tem três partes:

- Header (Algorithm e Token Type)
- Payload (Dados)
- Assinatura (opcional)

Exemplo de JWT:

eyJ(...)Ino . eyJzdwI(...)dwVq . (vazio)

exemplo de header:

```
{
  "typ": "JWT",
  "alg": "none"
}
```

"none" → significa que é "not signed"

pode ser:

"HS256" → JWT Signed (symmetric)

"RS256" → JWT RSA signed

...

+ outras dimensões chaves/hashes

Exemplo de payload:

```
{ "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

claim

Representa o subject do token (usualmente representa o Resource Owner)

outros dados estruturados em JSON

JWT claims

Um JWT pode conter qualquer tipo de dados estruturados em JSON. Contudo, foram definidas claims fixas para utilizar entre diferentes aplicações.

Representam operações típicas

Alguns serviços (ex: OpenId Connect) definem um conjunto de claims JWT obrigatórias.

VER EXEMPLO NODE.JS

VER TABELA 1.1.

↓

JWT claims

Proteção Criptográfica de Tokens - JSON Object Signing and Encryption (JOSE)

↳ define algoritmos para assinatura simétrica / assimétrica, cifra simétrica / assimétrica, e key storage

↳ JSON Web Signatures (JWS)
↳ ex: "HS256", "RS256"

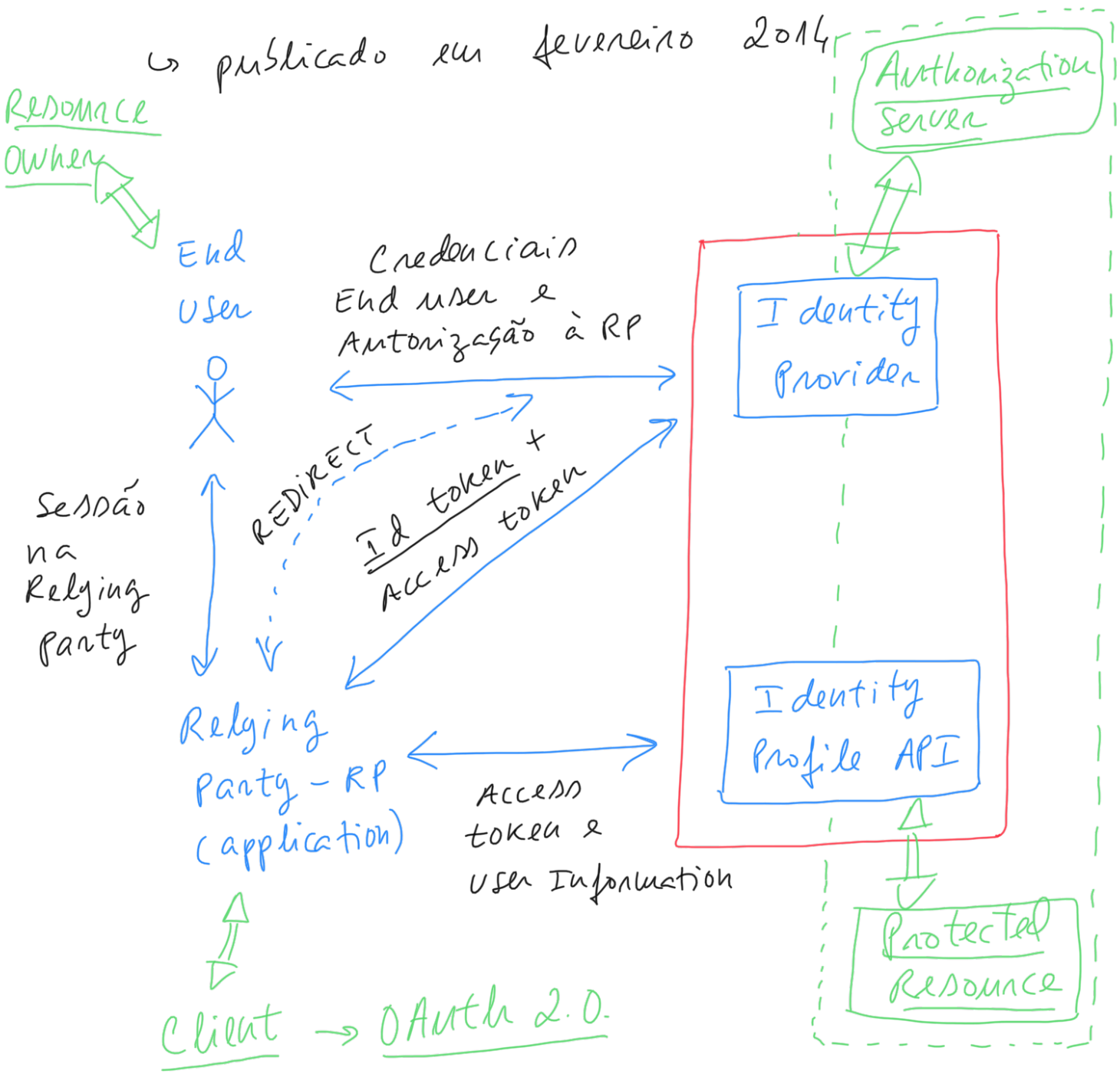
↳ JSON Web Encryption (JWE)

↳ JSON Web Keys (JWK)

OpenId Connect

Standard para autenticação e identidade sobre o OAuth 2.0.

↳ publicado em fevereiro 2014



End user delega à aplicação (Relying party) a sua própria informação de identidade, i.e., autoriza a RP a descobrir quem é o utilizador que está a iniciar sessão na RP.

O Id token é um signed JSON Web Token (JWT)

↓

semelhante a um cookie assinado

Ver Id token claims - Table 13.1 do pdf anexo

↳ as claims a negrito são obrigatórias

O Id token é emitido no campo id_token na resposta recebida do OAuth token endpoint, i.e., vem em conjunto com o access token.

Ex:
```
{
  "access_token": "987 ... ghj",
  "token_type": "Bearer",
→ "id_token": "eyJ ... "
}
```

O Id token é assinado com a chave do Id Provider (ou Authorization Server).

↳ os campos c_hash e at_hash são assinaturas do authorization code e access token, respetivamente, para que o cliente possa verificar estas assinaturas mantendo o conteúdo do auth. code e access token opacos ao cliente. → visa evitar ataques de injeção

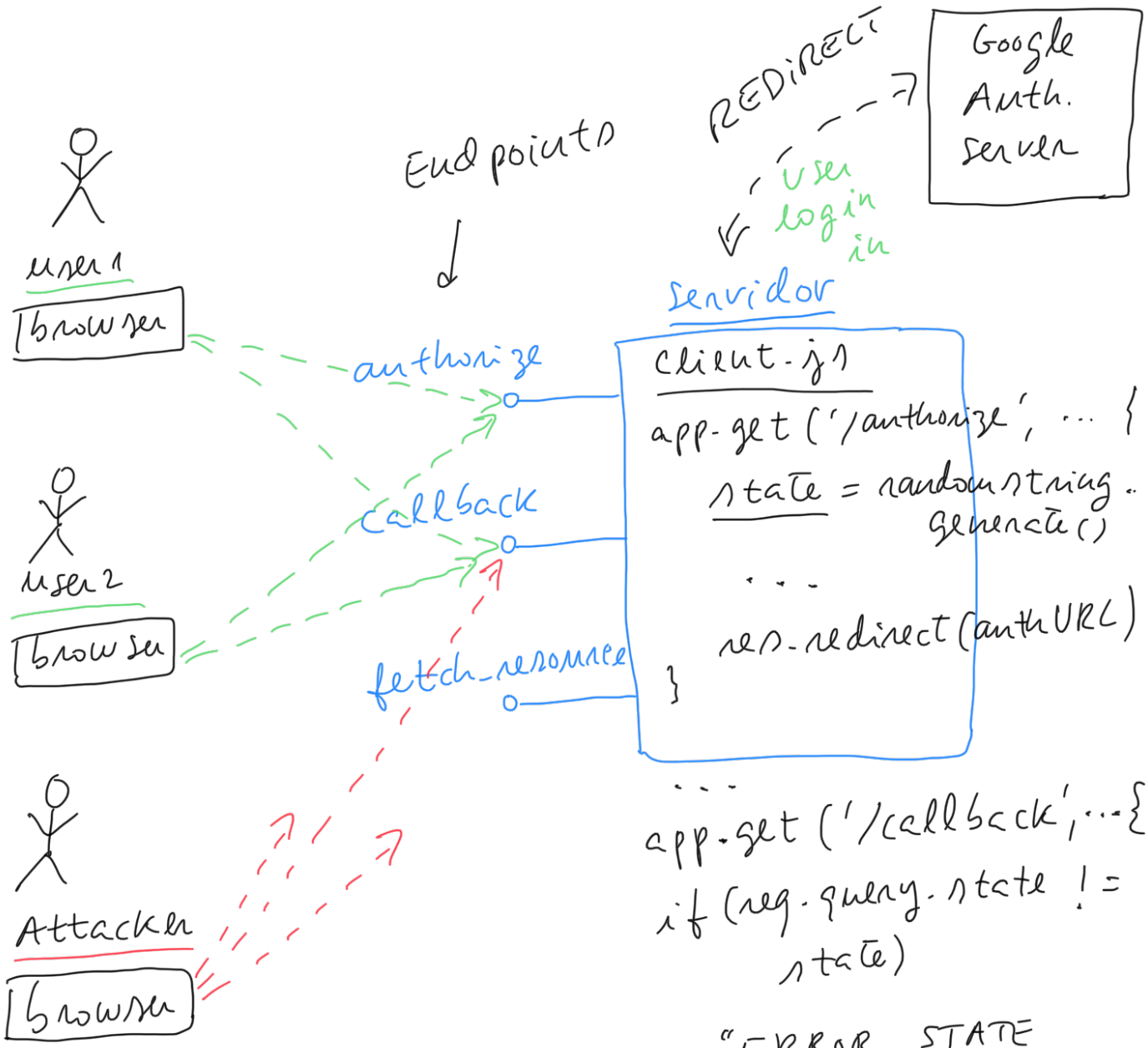
Parsing e validação:

- Parsing das secções header e payload
- validar assinatura do token (ex: assinatura digital assimétrica)
- verificar se ID Provider é trusted
- verificar se id client = audience field
- confirmar datas de expiração, iat, ...
- verificar se nonce é igual ao enviado
- validar hashes, se aplicável

UserInfo endpoint

O access token pode, opcionalmente, ser usado num Protected Resource que contém informação de perfil (profile) do utilizador
↳ userInfo endpoint

Parâmetro state (revisão)



DEMO: ex-2-OAuth-JWT

↳ Uncomentários "ADDED" em:

- authorization server.js
- ProtectedResource.js