

TOP-KT-020 - Uitwisseling publieke sleutels

▼ Versiegeschiedenis...

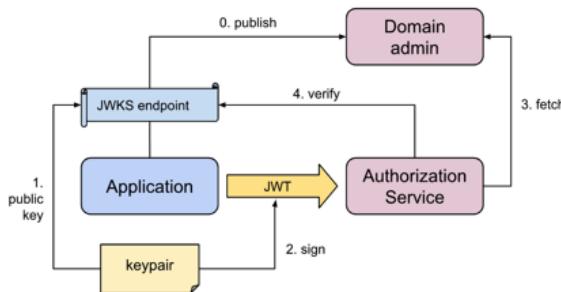
Versie	Datum	Status	Wijzigingen
1.0.1	8 Sept 2023	definitief	Verwijderen van mogelijkheid om publieke key te configureren. Deze mogelijkheid is niet in gebruik en tevens is dit niet gewenst.
1.0.0	10 Mar 2023	definitief	
0.1.0	1 Feb 2023	concept	

Beschrijving

Koppelstaal maakt bij zowel HTI, SMART on FHIR backend services, SMART on FHIR app launch als de token introspection gebruik van authenticatie op basis van publieke en private sleutels in asymmetrische sleutelparen. De uitwisseling van deze publieke sleutels is een terugkerend thema in Koppelstaal. De uitwisseling van publieke sleutels vormt in koppelstaal het netwerk van vertrouwen, omdat de publieke sleutels een centrale rol spelen in het authenticeren van de identiteit van de applicaties en andere systemen in het domein. Om de publieke sleutels uit te wisselen, wordt er gebruik gemaakt van een JWKS endpoint, waar in het JWK Set Formaat de publieke sleutels worden gedeeld. Publieke sleutels en asymmetrische sleutelparen spelen een rol in verschillende onderdelen binnen koppelstaal:

- Bij het ondertekenen en valideren van het HTI token.
- Bij het JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication (rfc7523), deze wordt gebruikt in:
 - Het benaderen van het token introspection endpoint.
 - Bij de SMART on FHIR backend services.
 - Bij de SMART on FHIR app launch.
- Bij het ondertekenen en valideren van het access_token van de SMART on FHIR backend services.
- Bij het ondertekenen en valideren van het id_token van het SMART on FHIR app launch framework.

In al deze gevallen wordt een JWT door de applicatie ondertekend met een eigen sleutelpaar. Het publieke deel van het sleutelpaar wordt via een JWKS endpoint beschikbaar gemaakt. Het JWKS endpoint is bij domeinbeheer bekend. De autorisatieservice haalt het adres van het JWKS endpoint op en valideert het JWT token met de publieke sleutel uit het JWKS endpoint. Een uitzondering is het JWKS endpoint van de autorisatieservice, deze wordt met het .well-known/smart-configuration (zie ook [TOP-KT-016 - SMART on FHIR Conformiteit](#)) bekend gemaakt. Verder is het van belang te noemen dat het introspection endpoint de validatie van JWT tokens voor zijn rekening neemt, applicaties hoeven dus niet zelfstandig de JWKS endpoints van elkaar te benaderen, dit doet de autorisatieservice voor de applicaties in het domein.



Overwegingen

Publieke sleutels: netwerk van vertrouwen

Omdat de publieke sleutel wordt gebruikt om de identiteit van een applicatie in het domein vast te stellen, worden de publieke sleutels ook wel aangeduid als de identiteit van de applicatie in een domein. Door de identiteit te publiceren, wordt het voor andere applicaties en componenten mogelijk de identiteit vast te stellen. Dit gebeurt door handtekeningen (signatures) van berichten van de applicatie te valideren met behulp van de publieke sleutel zoals die bekend is van de applicatie. Het systeem kan door het ondertekenen van een arbitrair bericht met de geheime sleutel aantonen dat het systeem de eigenaar is van deze geheime sleutel, door met de publieke sleutel de handtekening te valideren kan dit worden vastgesteld. Immers, enkel degene die de private sleutel heeft kan het bericht ondertekend hebben. Vertrouwen tussen systemen kan worden opgebouwd door elkaars publieke sleutels te accepteren.

Het uitwisselen en vertrouwen van publieke sleutels is geen nieuwe uitdaging; er wordt hard gewerkt aan nieuwe standaarden en toepassingen die het uitwisselen van identiteiten en vertrouwen mogelijk maken. Specifiek kijken we hier naar [DID](#) documenten. DID vormt een standaardformaat waarmee publieke sleutels en andere publieke informatie over personen of systemen kenbaar gemaakt kunnen worden en uitgewisseld kunnen worden. Daarnaast kijken we naar [Verifiable Credentials](#) om netwerken van vertrouwen op te bouwen. In die context is het van belang om [NUTS](#) en [Sovrin](#) te benoemen. We kiezen er voor dan ook in koppeltaal op deze ontwikkelingen voor te sorteren door in de technische keuzes die we maken dergelijke technologieën niet uit te sluiten.

Updates en rotatie

Publieke sleutels zijn onderdeel van sleutelparen. Deze sleutelparen worden gegenereerd, met voorkeur automatisch, waarna alleen het publieke deel wordt gedeeld. Het geheime deel moet juist niet worden gedeeld. Om deze reden vereisen we dat de leveranciers de private sleutel goed te beschermen. In het verlengde daarvan stimuleren we het gebruik van meerdere sleutels. Zo hoeft de sleutel niet over applicatie instanties, pods en/of servers gedeeld te worden. Tevens stimuleren we de snelle rotatie van sleutels, zo kan de lifecycle van de sleutels gekoppeld worden aan de lifecycle van de applicatiesinstanties. We adviseren leveranciers de sleutelparen bij het opstarten van de applicatie automatisch te genereren als onderdeel van de opstartprocedure.

JSON Web Keys

JWKS endpoints zijn geen standaard, maar komt voor in de OAuth 2.0 Authorization Server Metadata standaard als optionele URL. We kiezen er in koppelstaal voor om JWKS endpoints te gebruiken om elkaars publieke sleutels uit te wisselen. We kiezen er niet voor om de volledige OAuth 2.0 Authorization Server Metadata te ondersteunen, met name omdat de meeste applicaties niet de rol van autorisatie server vervullen. Van applicaties wordt verwacht dat zij een op een enkel endpoint al hun publieke sleutels die zijn actief gebruiken bij het ondertekenen van JWK tokens publiceren. Het JWKS endpoint bevat een JSON document met in de “keys” waarde een lijst van [JSON Web Key \(JWK\) \(rfc7517\)](#). De leverancier kan zelf de URL van het JWKS endpoint bepalen. De JWKS URL wordt per applicatie-instantie in domeinbeheer geconfigureerd.

Domain Hijacking

In de context van JWKS op basis van URLs is het van belang het concept van Domain Hijacking te benoemen. Indien een domein zijn DNS registratie verliest, wordt het voor de partij die het overneemt mogelijk om op de JWKS URL zijn eigen publieke sleutels te communiceren. Hiermee kan een eigen sleutelpaar gebuikt worden en dus een de rol van de applicatie in het domein worden overgenomen. Hoewel een klein risico is dit wel degelijk een reëel risico; zorgaanbieders hebben in het verleden hun domein verloren. Als Koppelstaal benoemen we dit risico. De mitigatie is mogelijk het gebruik van NUTS, aangezien hiermee de uitwisseling van de publieke sleutels gebeurt op basis van wederzijds vertrouwen door het accepteren van de publieke identiteit en het gebruik van vertrouwensnetwerken.

Caching van het JWKS endpoint

Validaties van ondertekende berichten vinden in een koppelstaal domein zeer regelmatig plaats, hierbij is een accurate en up-to-date situatie van de publieke sleutels van belang. Technisch gezien zou dus bij elke validatie het JWKS endpoint opnieuw benaderd moeten worden. Om eventuele druk op dit endpoint te verminderen en tevens het validatieproces te versnellen mag gebruikt worden van **Cache-Control** headers op het endpoint. Daarmee geeft de bron aan wat de geldigheidsduur van het JWKS document aan en kan de validerende partij voor die periode de bestaande gegevens gebruiken in plaats van steeds opnieuw de gegevens op te vragen.

Toepassing, restricties en eisen

JWKS endpoint

In koppelstaal wordt van elke applicatie en dienst in het domein die gebruik maakt van SMART on FHIR app launch, SMART on FHIR backend services en token introspection verwacht een JWKS endpoint te implementeren. Dit endpoint is een URL waar een document van mime-type application/json te vinden is, waar in een JSON map onder de sleutel “keys” een lijst aan JSON Web Keys (rfc7517) te vinden is zoals beschreven in het JWK Set Format.

```
1 {  
2   "keys": [  
3     ..  
4   ]  
5 }
```

JSON Web Keys

De individuele publieke sleutel wordt uitgewisseld in het JWK (rfc7517) formaat. In dit formaat kent de volgende velden in koppelstaal.

- kty (Key Type), verplichte key type
- kid (Key ID), verplichte key identifier
- use, optioneel, altijd sig
- key_ops, optioneel, moet verify

Afhankelijk van de key type (kty) zijn de velden van de public key verplicht. Deze staan beschreven in het JSON Web Algorithms (JWA) rfc7518 document. Bijvoorbeeld.

- RSA: n en e
- EC: crv , x , en eventueel y

De keys die beschikbaar worden gesteld het JSON Web Keys mogen enkel public keys zijn, de standaard geeft ook de mogelijkheid private keys uit te wisselen. Dit is uitdrukkelijk niet de bedoeling. De meeste software bibliotheken ondersteunen zowel publieke als private sleutels, dit is een punt van aandacht waar fouten potentieel gemakkelijk gemaakt worden.

Caching

De implementatie van het JWKS endpoint mag gebruik maken van Cache-Control headers. Aangezien de publieke sleutel niet geheim is, integendeel, mag deze ook op tussenliggende systemen, zoals proxy-servers, worden opgeslagen.

```
1 Cache-Control: public, max-age=60
```

We geven geen limiet aan de maximale leeftijd van het JWKS document. Echter, de maximale leeftijd bepaalt de tijd tussen publiceren van een nieuwe sleutel en het gebruik van de nieuwe sleutel. Wordt een nieuwe een sleutel eerder gebruikt, kan het zijn dat de verifiërende partij nog niet op de hoogte is van de nieuwe sleutel en de verificatie dus niet lukt. Dit geldt ook voor het intrekken van een sleutel, deze intrekking wordt pas geëffectueerd na het verlopen van de maximale leeftijd van het JWKS document zoals bepaald door de Cache-Control headers.

Voorbeelden

Request:

```
1 GET /.well-known/jwks.json HTTP/2
2 Host: authentication-service.koppelstaal.headease.nl
3 user-agent: curl/7.79.1
4 accept: application/json
```

Antwoord:

Irrelevante headers zijn weggelaten.

```
1 HTTP/2 200
```

```
2 date: Thu, 06 Oct 2022 07:42:38 GMT
3 content-type: application/json
4 content-length: 437
5
6 {
7     "keys": [
8         {
9             "e": "AQAB",
10            "kid": "J-1qj3T1WHijPpwHetreow3MQgbE_luA66NiIoHKoEo",
11            "kty": "RSA",
12            "n": "1Ec-LfmgoXo6cr-5wiCJ5qCIqS-
cWcND4QnANYM0pLYsfNfWTy2NwWNzMBYfHJfdGKdVjooyyKXLcEKpju15jc0bUyuGUo6dnb0bhWqkBeIC13UUJaUFJcuMM3lozSgGDBYMD
4a3zbGt_Pu1RYjBbnhuktXF8D-14lv9KCPvJonMsv8ckDgJGg0-rQ1CB9A0BZAr6_J1S7MkBN5nCnC05ba-
PWbCQmd4W74XZVSKmFrd8jfRpjvwIc-ZlaVK-
JigQPWdqvLeHLZ3zyQh_0FLonKKZdXP1kPB3yPZa13XKc2ZxwydxsbzdfgSWcKPhxQrRYzu4PIPL1YDmA89XEeXw"
13        }
14    ]
15 }
```

Links naar gerelateerde onderwerpen

In het topic [TOP-KT-008 - Beveiliging aspecten](#) wordt verder uitgeweid over de eisen aan de sleutels en ondertekening.

Gebruikte standaarden

[JSON Web Token \(JWT\) Profile for OAuth 2.0 Client Authentication \(rfc7523\)](#)

[JSON Web Key \(JWK\) \(rfc7517\)](#)

[JSON Web Algorithms \(JWA\) \(rfc7518\)](#)

[OAuth 2.0 Token Introspection \(rfc7662\)](#)

{}