Ondertekenen en adresseren in REST

Een profiel voor Edukoppeling

Versie 0.2 (Concept)
Marc Fleischeuers, Kennisnet

Versiehistorie

Versie	Datum	Auteur	Beschrijving
0.0	17-12-2018	M. Fleischeuers	Eerste opzet
0.1	20-12-2018	M. Fleischeuers	Commentaar development team
0.2	7-2-2019	M. Fleischeuers	Verduidelijking ondertekening en plaatjes. Commentaar.

Goedkeuring

Versie	Datum	Naam	Functie

1 Inleiding

<doelgroep: ontwerpers, architecten>
<scope: profielen voor adresseren en routeren van integere en versleutelde communicatie equivalent aan Edukoppeling>

2 Beveiliging van berichtverkeer

Communicatie op basis van REST-gebaseerde protocollen kan worden beveiligd door berichten te ondertekenen (dat wil zeggen, voorzien van een cryptografisch sterke handtekening) of door te ondertekenen en versleutelen. Er zijn een aantal standaarden die beschrijven hoe een en ander in zijn werk gaat. Binnen deze standaarden zijn voor bepaalde maatregelen meerdere varianten beschikbaar. Hiervoor moeten keuzes worden gemaakt en vastgelegd.

Afspraken voor gebruik van ondertekening of encryptie en ondertekening in berichtverkeer in de onderwijsketen volgen uit de BIV classificatie van het certificeringsschema voor dit berichtverkeer. Daarnaast nog additionele afspraken worden gemaakt, per toepassing of per keten, bijvoorbeeld ondertekenen van berichten als er via brokers of intermediairs gerouteerd wordt.

Toepassing van ondertekening en / of versleuteling in REST-gebaseerde services

REST-GEN -001	De aanbieder van de dienst bepaalt of zijn dienst gebruikt maakt van *ondertekening* van de invoer of uitvoer van de dienst.	De aanbieder van de dienst gebruikt de gevoeligheid van de gegevens van de dienst om te bepalen of ondertekenen van objecten een passende maatregel is.
REST-GEN -002	De aanbieder van de dienst bepaalt of zijn dienst gebruik maakt van *encryptie* van de invoer of uitvoer van de dienst.	De aanbieder van de dienst gebruikt de gevoeligheid van de gegevens van de dienst om te bepalen of versleutelen van objecten een passende maatregel is.
REST-GEN -003	De afnemer valideert de ondertekening van de uitvoer van een dienst als de aanbieder van de dienst de uitvoer ondertekent.	Als een dienst zijn uitvoer ondertekent, zal de afnemer deze ondertekening valideren. Als een bericht van deze dienst geen ondertekening bevat, zal de afnemer het bericht afkeuren.
REST-GEN -004	Voor ondertekening en / of versleuteling van REST api's zullen aanbieders en afnemers de JOSE-standaarden (zie hieronder) en JWT gebruiken.	Er zijn meerdere standaarden die gebruikt kunnen worden. Voor deze afspraken beperken we ons tot JOSE, JWS, JWE, JWK, JWA en JWK, en maken hierbij gebruik van JWT.
REST-GEN -005	Voor toepassingen van machine-machine communicatie met signing of encryptie en signing van berichten binnen de onderwijsketen zullen de nadere voorschriften uit deze standaard gebruikt worden	De standaarden zijn generiek opgezet en ondersteunen veel verschillende use cases. Per toepassing moet worden bepaald welke variaties veilig gebruikt kunnen worden. In dit document wordt de selectie beschreven voor de toepassing in het onderwijs.

3 Overzicht van relevante standaarden en referenties

JOSE	https://tools.ietf.org/html/rfc7520	The JSON Object Signing and Encryption (JOSE) technologies -- JSON Web Signature [JWS], JSON Web Encryption [JWE], JSON Web Key [JWK], and JSON Web Algorithms [JWA] -- can be used collectively to encrypt and/or sign content using a variety of algorithms.
JWS	https://tools.ietf.org/html/rfc7515	JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JSON-based data structures.
JWE	https://tools.ietf.org/html/rfc7516	JSON Web Encryption (JWE) represents encrypted content using JSON-based data structures.
JWK	https://tools.ietf.org/html/rfc7517	A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) data structure that represents a cryptographic key. This specification also defines a JWK Set JSON data structure that represents a set of JWKs.
JWA	https://tools.ietf.org/html/rfc7518	This specification registers cryptographic algorithms and identifiers to be used with the JSON Web Signature (JWS), JSON Web Encryption (JWE), and JSON Web Key (JWK) specifications. It defines several IANA registries for these identifiers.
JWT	https://tools.ietf.org/html/rfc7519	JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties.
IANA JOSE registry	https://www.iana.org/assignments/jose/jose.xhtml	Registratie van parameters en algoritmen voor gebruik bij Json Object Signing and Encryption
IANA JWT Registry	https://www.iana.org/assignments/jwt/jwt.xhtml	Registratie van claims in JWT tokens

<TO DO: aanvullen met delen van het REST rapport>

4 Ondertekening

Ondertekening van berichten wordt gebruikt voor <...>

4.1 Alternatieven en keuzes

4.1.1 Gebruik van JWS en JWT

JWS is de standaard om cryptografische ondertekening aan te brengen en te valideren in een JSON bericht. JWT is een JSON gegevensstructuur waar ondertekening(en) van een JSON bericht in kunnen worden ondergebracht.

Een JWT (uitgesproken als *jot*) bvat 3 of 5 velden met precies beschreven inhoud, die per toepassing (hier: ondertekening en/of versleuteling) verschilt. Voor een JWT zijn er drie verschillende representaties (formaten voor weergave of uitwisseling) gedefinieerd:

- *Compacte representatie*: base64url weergave van de drie of vijf onderdelen van een JWT, geschikt voor opname in een url of http header
- *JWS JSON representatie*: gebruik van een JWT als JSON weergaveobject, voor additionele flexibiliteit bijvoorbeeld meerdere ondertekeningen bij encryptie
- *Platte JWS JSON representatie*: vereenvoudigde variant van de JWS JSON representatie

Een JWT wordt van client naar server overgebracht in de vorm van

- Een URL parameter: http://server.io/parm1=value1&parm2=[jwt token]
- Een http “Authorization: bearer” header variabele, als in OAuth2
- Een zelf-gedefinieerde http header variabele
- Een http cookie

Voor de keuze van de JWT-representatie en –vorm is het van belang dat het niet interfereert met bestaande oplossingen. Het gebruik van een Authorization:bearer kan een risico worden, dit is namelijk de header die gebruikt wordt door OAuth. Omdat het hier in eerste instantie om routeringsinformatie gaat, zou dat niet direct in deze header te verwachten zijn. De manier waarop cookies worden gezet en verwijderd lijkt niet te passen met dit gebruik, bovendien zouden cookies eigenlijk betekenisloos moeten zijn. Opname in een URL kan problemen geven als het token te groot wordt, bijvoorbeeld als het publieke deel van het certificaat onderdeel is van de header van het token. Daarmee is een eigen header variabele de meest voor de hand liggende manier om JWTs uit te wisselen.

JWT’s kunnen JWT’s bevatten, bijvoorbeeld bij het doorgeven van JWT tokens van een systeem naar een ander systeem. Deze werkwijze is vergelijkbaar met het doorgeven van SOAP berichten als attachment bij een SOAP bericht. De complexiteit en voorspelbaarheid van het berichtverkeer wordt hiermee nadelig beïnvloed.

REST-USE-001	De compacte representatie van JWT wordt gebruikt, en de JWS JSON representatie als de additionele functionaliteit van die representatie nodig is. De platte JWS JSON representatie wordt niet gebruikt.	Additionele functionaliteiten in JWE JSON zijn: een bericht encrypten voor meerdere geadresseerden, en unprotected headers.
REST-USE-0	Een JWT in compacte representatie	Deze keuze vermijdt mogelijke toekomstige

02	wordt van client naar server doorgegeven in de vorm van een zelf-gedefinieerde http header variable '`edustd-jwt`'.	conflicten met OAuth, waarvoor ook JWT's gebruikt kunnen worden, maar dan met andere inhoud. Een URL parameter wordt al snel heel lang, en het nadeel van een cookie is [XXX]
REST-USE-003	Gebruik geen geneste JWT's.	In elk geval niet totdat er een duidelijke noodzaak voor is, en er duidelijkheid is over de gevolgen voor interoperabiliteit.

4.1.2 Ondertekening

Gebruik van PKIo certificaten, met als belangrijke eigenschap de hoge zekerheid van identificatie van de partij die het certificaat gebruikt, wordt ook ingezet bij ondertekening van REST berichtverkeer. In het huidige SOAP verkeer wordt het publieke deel van het certificaat opgenomen in het bericht. Dit is ook mogelijk in JWT tokens. Er is een tweede mogelijkheid, dat partijen het publieke deel van hun certificaat publiceren in een (statische) url, en deze url opnemen in de JWT tokens. Met een url in plaats van een publiek deel van een certificaat is een JWT token duidelijk kleiner, ten koste van een iets groter risico dat het certificaat niet beschikbaar is om de ondertekening van het bericht te valideren. Partijen kunnen zelf deze afweging maken, de standaard staat beide vormen toe.

De payload van een JWT token bevat *claims*, naam- en waarde paren waarvan de authenticiteit door de ondertekening van het token wordt bewezen. Een token bevat *registered claims*, die onderdeel zijn van de standaard. Voor verschillende toepassingen van JWT tokens zijn er daarnaast nog sets van *public claims*, hiervan zijn de namen vastgesteld bij IANA. Als laatste is het ook nog mogelijk om eigen, *private claims* op te nemen in de payload. Het is dus in principe mogelijk om een JWT token te gebruiken als 'envelop' voor de uitwisseling van applicatie-specifieke berichten. Dit is echter niet de intentie van dit profiel. Het voorstel is om de informatie over de ondertekening (en straks ook de encryptie) van berichten op te nemen in het JWT token, en dit token voegen bij een verder ongewijzigd bericht. Hiermee wordt ook de parallel met de werkwijze van het SOAP profiel van Edukoppeling in stand gehouden.

Het voorstel is hiermee om een (in eerste instantie private) claim af te spreken die gebruikt wordt om het feitelijke bericht mee te ondertekenen, gebruik makend van een cryptografisch sterk hashing algoritme, en alleen deze claim op te nemen in de payload van een JWT token. Met deze claim wordt de integriteit van het bericht geborgd. Het JWT token wordt vervolgens op de standaard manier ondertekend met het PKIo certificaat. Hiermee is de integriteit van het JWT token geborgd.

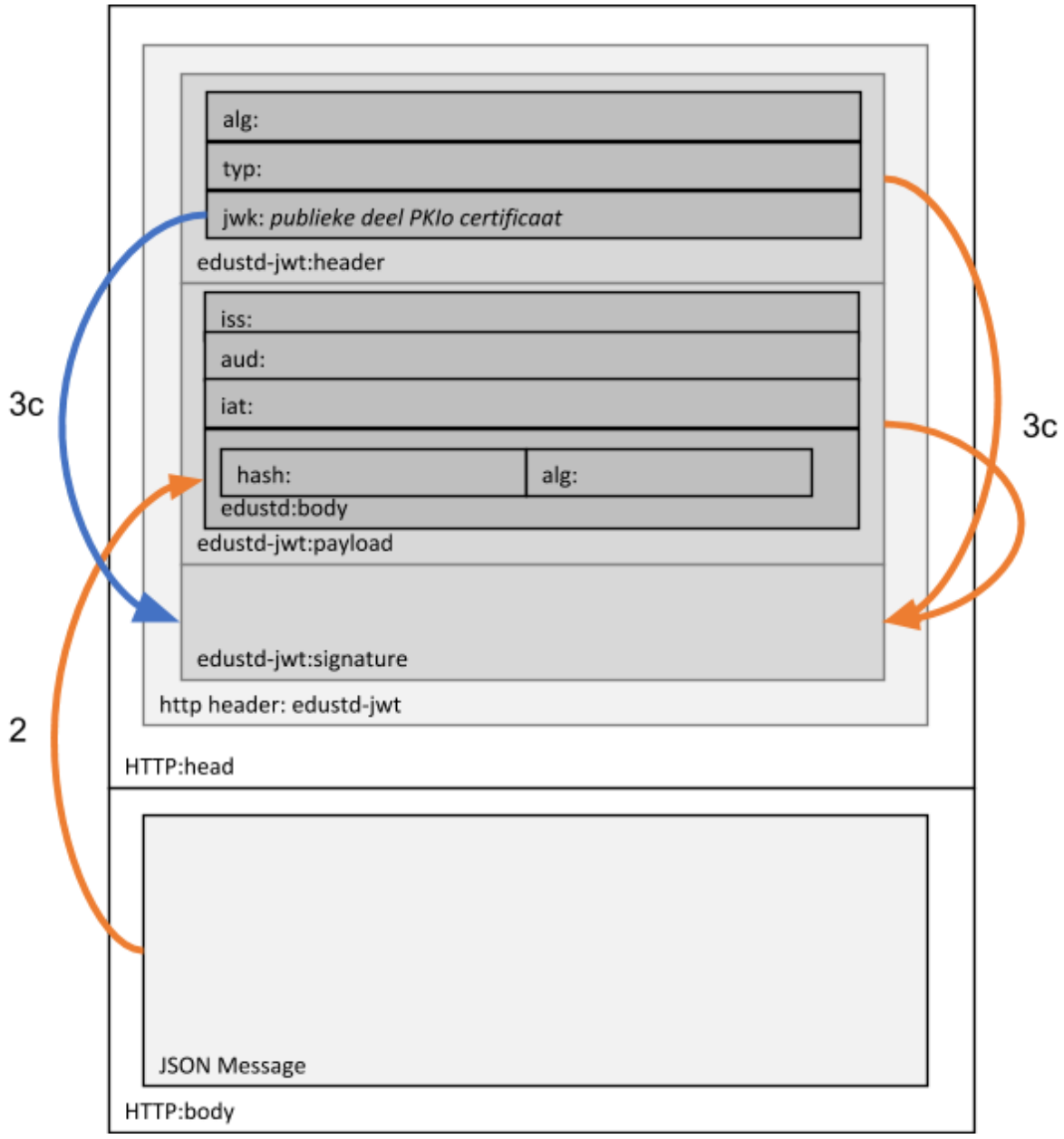
REST-SIG-001	REST berichtuitwisseling maakt gebruik van PKIo certificaten voor ondertekening en encryptie volgens de werkwijze die nu ook voor SOAP berichten wordt gebruikt.	De logica van ondertekening en encryptie bij REST is analoog aan de bekende werkwijze, en dat is belangrijk omdat partijen hier al ervaring mee hebben.
REST-SIG-002	Voor ondertekening van berichten wordt het publieke deel van het PKIo certificaat (of een verwijzing daar naartoe) opgenomen in de header van een JWT token en een claim met de ondertekeing opgenomen in de payload.	
REST-SIG-003	De werkwijze van Edukoppeling met SOAP, waarbij adressering, routering en ondertekening en encryptie van het bericht gescheiden is van de feitelijke	

	inhoud vna het bericht, wordt ook gehandhaafd in de werkwijze voor REST.	

De claims zijn als volgt gedefinieerd:

- **JWT Header: jwk**, waarbij gebruik gemaakt wordt van de opties voor x509 certificaten in de standaard (JWK, https://tools.ietf.org/html/rfc7517)
- **edustd:body**
    - **alg:** bevat het label van het gebruikte hashing algoritme. Zie tabel 1 voor toegestane waarden.
    - **hash:** bevat de hashwaarde van het bericht volgens het opgegeven algoritme.

Een ondertekend JSON bericht is onderdeel van een http uitwisseling, waarbij het JWT token met de ondertekening is opgenomen in een http header variabele `edustd-jwt`. Deze structuur is weergegeven in figuur 1.



Figuur 1: Structuur van een JSON bericht met JWT ondertekening. De nummers bij de pijlen verwijzen naar acties in Verantwoordelijkheden van afzender.

De ondertekening van een JSON bericht is een hashwaarde die volgens een vastgesteld algoritme is berekend. De hash en het algoritme waarmee de hash is berekend worden beide opgenomen in de payload van het JWT token. Het algoritme om deze hash te berekenen gebruikt geen geheimen (*salt*) dus iemand die het bericht aanpast kan ook de hash opnieuw berekenen met het opgegeven algoritme. Het JSON bericht is nu dus nog niet digitaal ondertekend.

De header en payload van het JWT token worden ondertekend met het algoritme en bijbehorende paramers dat in de header van het JWT token is beschreven. In deze ondertekening wordt het PKIo certificaat gebruikt: de private sleutel van de afzender voor de ondertekening zelf, en de publieke sleutel (meegeleverd in de header of beschikbaar op een url) voor de validatie van de ondertekening. Met deze operatie wordt het JWT token digitaal ondertekend. Wijzigingen in het JSON bericht leiden

tot een ongeldige hash, en wijzigingen in de hash leiden tot een ongeldige ondertekening. Zonder private sleutel van het PKIo certificaat van de afzender kan er geen nieuwe geldige ondertekening worden gemaakt.

4.1.3 Algoritmen voor hashing en ondertekening

De algoritmen die we gebruiken voor de ondertekening van de JSON message zijn niet gestandaardiseerd in de RFCs of bij IANA. De lijst met toegestane algoritmen hiervoor zijn opgenomen in tabel 1.

alg	**Hashing- of MAC algoritme**	**Implementatie**
B64SHA256	BASE64URL geëncodeerde SHA-256 hash	Verplicht

Tabel 1 Lijst met algoritmen voor hashing van het JSON bericht. De termen onder implementatie (‘verplicht’, ‘aanbevolen’ en ‘optioneel’) gelden voor de ontvangers van berichten.

De JWA RFC (https://tools.ietf.org/html/rfc7518) beschrijft alle toegestane algoritmen voor ondertekening van het JWT (RFC 7518, par 3.1). Van deze lijst gebruiken wij:

alg	**Digitaal onderteken- of MAC algoritme**	**Implementatie**
RS256	RSASSA-PKCS1-v1_5 met SHA-256	Verplicht
RS384	RSASSA-PKCS1-v1_5 met SHA-384	Optioneel
RS512	RSASSA-PKCS1-v1_5 met SHA-512	Optioneel
ES256	ECDSA met P-256 en SHA-256	Aanbevolen
ES384	ECDSA met P-256 en SHA-384	Optioneel
ES512	ECDSA met P-256 en SHA-512	Optioneel
PS256	RSASSA-PSS met SHA-256 en MGF1 met SHA-256	Optioneel
PS384	RSASSA-PSS met SHA-384 en MGF1 met SHA-384	Optioneel
PS512	RSASSA-PSS met SHA-512 en MGF1 met SHA-512	Optioneel
none	Geen digitale ondertekening of MAC	Niet toegestaan

Tabel 2 Lijst met algoritmen voor ondertekening van het JWT token. De termen onder implementatie (‘verplicht’, ‘aanbevolen’ en ‘optioneel’) gelden voor de ontvangers van berichten.

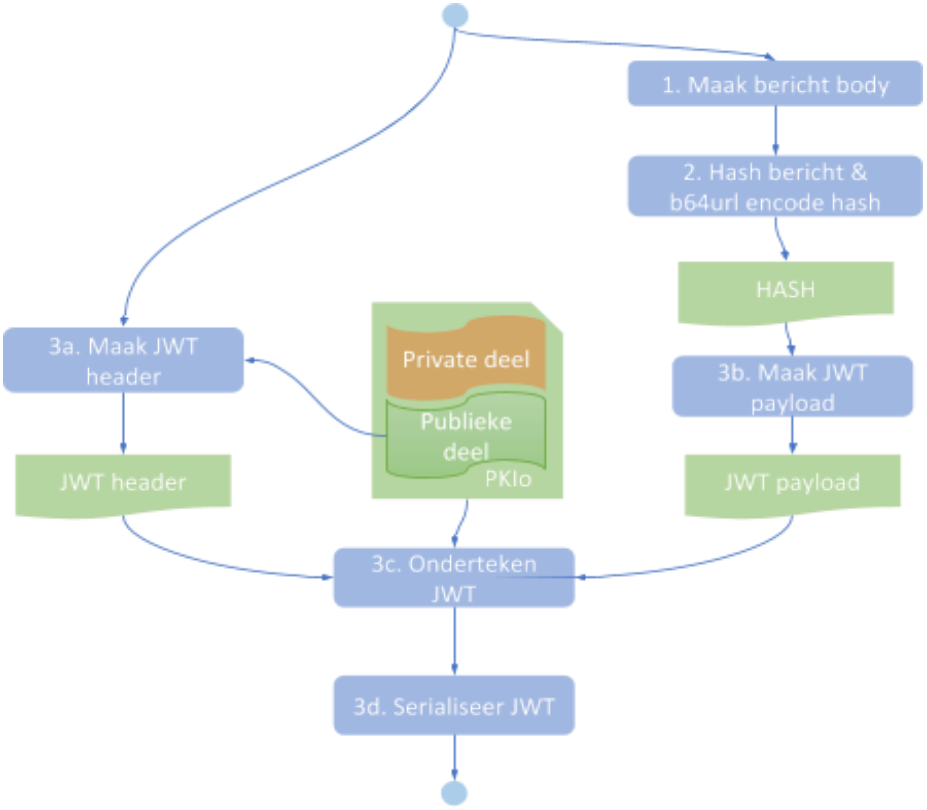
Ontvangers van ondertekende berichten moeten tenminste de als ‘verplicht’ aangemerkte onderteken-algoritmen ondersteunen (d.w.z. kunnen valideren). Afzenders van berichten met ondertekening kiezen bij voorkeur een van de ‘verplicht’ of ‘aanbevolen’ algoritmen in de tabel en gebruiken niet de ‘niet toegestaan’ algoritmen. Afzenders houden er rekening mee dat ontvangers niet alle optionele of aanbevolen algoritmes ondersteunen.

4.2 Verzenden van berichten

4.2.1 Verantwoordelijkheden van afzender

De afzender stelt het bericht (JSON gegevensstructuur) samen dat naar de ontvanger moet worden gestuurd, maakt een ondertekening in een JWT token volgens onderstaande beschrijving en maakt een http bericht met het JWT token in de header en het bericht in de body en stuurt dit bericht naar de ontvanger met een http POST.

De werkwijze om een JWT token te genereren dat het bericht ondertekent is hieronder weergegeven. Deze werkwijze is ontleend aan de JWS RFC (https://tools.ietf.org/html/rfc7515) par 5.1.



Figuur 2. Stappen die een afzender zet om een JWT token te genereren dat een JSON bericht ondertekent.

Om een JWT token te maken dat een bericht ondertekekent volgt de afzender de volgende werkwijze:

1. Stel het bericht samen dat naar de ontvanger wordt gestuurd
2. Maak een hash voor het bericht aan en encodeer deze hash met base64url encoding, volgens de algoritmen in tabel 1.
3. Maak het JWS token
    a. Maak een JSON header met de claims. Neem hierin tenminste de verplichte claims voor de header (zie Claims in de header) op.
    b. Maak de content voor de JWS payload. Neem hierin tenminste de verplichte claims voor de payload (zie Claims in de payload) op.
    c. Kies een ondertekenalgoritme (*signing algorithm*) en het eigen PKIo certificaat (private en publieke deel). Zie tabel 2 voor een lijst van toegestane algoritmen. Bereken de ondertekening van het JWS token over de JWS header en payload: `JWS`

```
Ondertekening = Sign(ASCII(BASE64URL(UTF8(JWT Header)) || '.' ||
BASE64URL(JWT Payload)), [cert_priv.pem]).
```

d. Maak de geserialiseerde output (JWS compacte serialisatie): `BASE64URL(UTF8(JWT Header)) || '.' || BASE64URL(JWT Payload) || '.' || BASE64URL(JWS Ondertekening).`

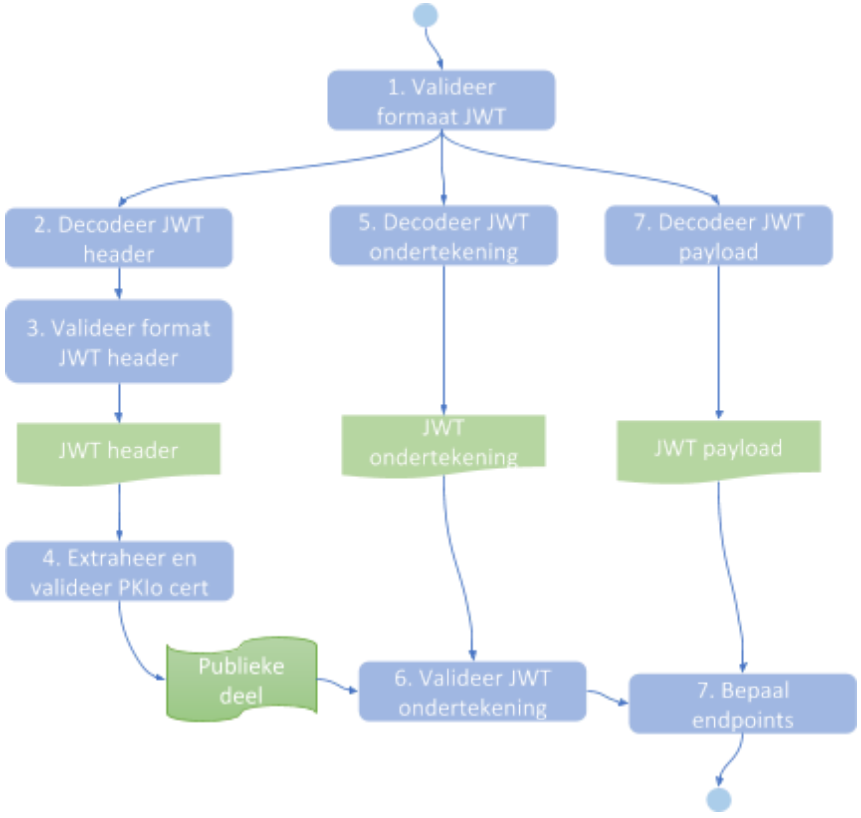
4. Bepaal de adressen van endpoint(s) of intermediair voor de geadresseerde(n) in het bericht en routeer het bericht naar deze ontvanger(s).

De onderstreepte onderdelen uit bovenstaande beschrijving worden door JWT library functies uitgevoerd.

4.2.2 Verantwoordelijkheden van transparante intermediair

Een transparante intermediair wordt gebruikt als een ondertekend JSON bericht niet rechtstreeks tussen afzender en geadresseerde kan worden gerouteerd, maar wel via een (reeks van) intermediairs. In dergelijke gevallen stelt de afzender het bericht op voor de beoogd ontvanger, maar levert het af bij de transparanter intermediair.

De intermediair heeft een verantwoordelijkheid voor het valideren van de ondertekening van het JWT token en voor het afleveren van het bericht, maar niet voor de validatie van inhoud of ondertekening van het JSON bericht. In onderstaand overzicht wordt de werkwijze weergegeven en toegelicht die een transparante intermediair volgt om een bericht te routeren (ontleend aan de JWS RFC (https://tools.ietf.org/html/rfc7515) par 5.1).



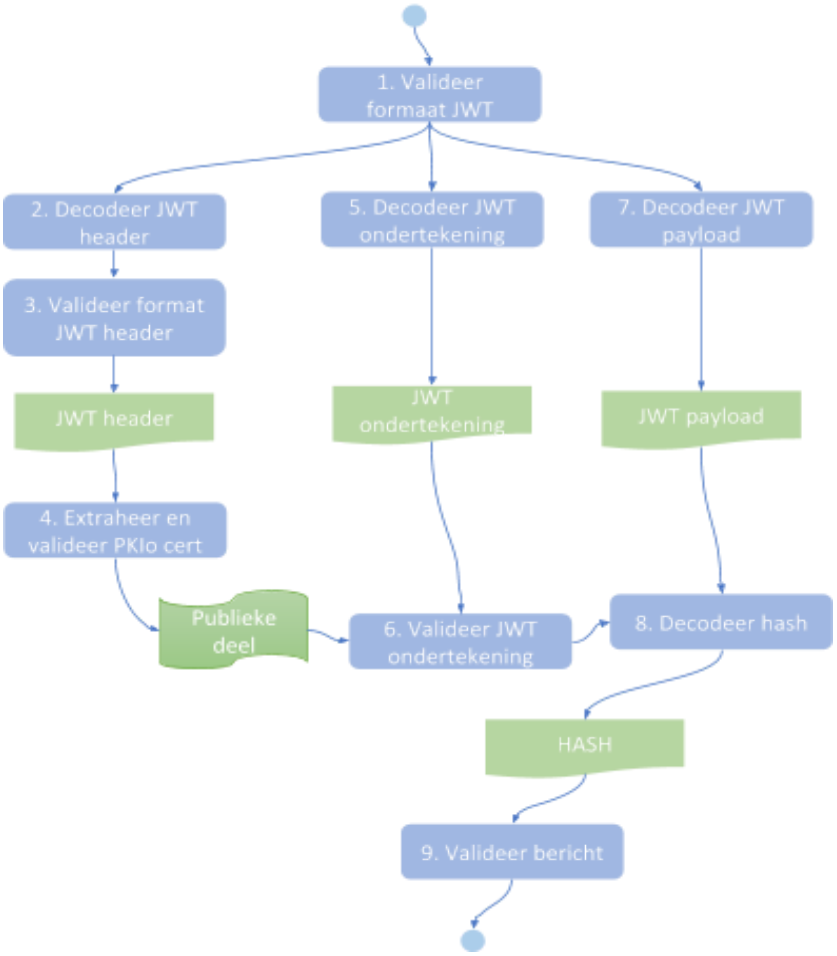
Elk van onderstaande stappen moet worden uitgevoerd. Als een van de stappen faalt, kan het token niet gevalideerd worden. Het bericht wordt in dat geval niet doorgestuurd en de afzender ontvangt een respons met een foutcode 400 (Bad Request) en een toepasselijke foutboodschap.

1. Ga na dat het JWT token 3 uit door '.' gescheiden letterreeksen bestaat.
2. Decodeer de header, dat wil zeggen het eerste deel van het JWT token tot aan de eerste punt: `BASE64URLDecode(JWT header)`.
3. Valideer het formaat van de header:
    a. Stel vast dat de gedecodeerde header een UTF-8 representatie is van een geldig JSON object.
    b. Verifieer dat de header geen dubbele parameters bevat.
    c. Verifieer dat er geldige waarden voor het ondertekenalgoritme worden gebruikt.
    d. Ga na of alle verplichte velden in de header verwerkt kunnen worden, net als alle waarden van deze velden.
4. Extraheer en valideer het certificaat dat gebruikt is om het bericht te ondertekenen:
    a. Bepaal het certificaat in de header van het JWT token, ofwel aan de hand van de `x5c` claim ofwel op basis van de `x5u` claim in de header.
    b. Ga na dat
        i. Het een geldig PKIo certificaat betreft.
        ii. Het certificaat niet verlopen is.
        iii. Het certificaat niet is ingetrokken door de leverancier van het certificaat.
5. Decodeer de JWT ondertekening, dat wil zeggen het laatste deel van het JWT token, vakaf het eerste karakter na de eerste punt, tot aan het einde: `BASE64URLDecode(JWT signature)`.
6. Valideer de ondertekening van het token tegen de header en payload van het JWT token: `ASCII(BASE64URL(UTF8(JWT Header)) || '.' || BASE64URL(JWT Payload))` op de manier die voorgeschreven is voor het gebruikte ondertekenagoritme JWT header.alg.
7. Bepaal de endpoint(s) of intermediair voor de geadresseerde(n) in het bericht en routeer het bericht naar deze ontvangers.

De onderstreepte onderdelen uit bovenstaande beschrijving worden door JWT library functies uitgevoerd.

4.2.3 Verantwoordelijkheden van ontvanger

De ontvanger valideert het JWS token, en met de ondertekening in de payload van het token, het bericht zelf. De ontvanger voert daarvoor de volgende stappen uit. Als een van de stappen faalt, kan het token of het bericht niet gevalideerd worden en daarmee kan de authenticiteit van het bericht niet worden vastgesteld. Deze stappen gaan uit van een JWT in compacte serialisatie.



Elk van onderstaande stappen moet worden uitgevoerd. Als een van de stappen faalt, kan het token niet gevalideerd worden. Het bericht wordt in dat geval niet doorgestuurd en de afzender ontvangt een respons met een foutcode 400 (Bad Request) en een toepasselijke foutboodschap.

1. Ga na dat het JWT token 3 uit door '.' gescheiden letterreeksen bestaat.
2. Decodeer de header, dat wil zeggen het eerste deel van het JWT token tot aan de eerste punt: `BASE64URLDecode(JWT header)`.
3. Valideer het formaat van de header:
    a. Stel vast dat de gedecodeerde header een UTF-8 representatie is van een geldig JSON object.
    b. Verifieer dat de header geen dubbele parameters bevat.
    c. Verifieer dat er geldige waarden voor het ondertekenalgoritme worden gebruikt.
    d. Ga na of alle verplichte velden in de header verwerkt kunnen worden, net als alle waarden van deze velden.
4. Extraheer en valideer het certificaat dat gebruikt is om het bericht te ondertekenen:
    a. Bepaal het certificaat in de header van het JWT token, ofwel aan de hand van de `x5c` claim ofwel op basis van de `x5u` claim in de header.
    b. Ga na dat
        i. Het een geldig PKIo certificaat betreft.
        ii. Het certificaat niet verlopen is.
        iii. Het certificaat niet is ingetrokken door de leverancier van het certificaat.
5. Decodeer de JWT ondertekening, dat wil zeggen het laatste deel van het JWT token, vakaf het eerste karakter na de eerste punt, tot aan het einde: `BASE64URLDecode(JWT signature)`.

6. Valideer de ondertekening van het token tegen de header en payload van het JWT token: `ASCII(BASE64URL(UTF8(JWT Header)) || '.' || BASE64URL(JWT Payload))` op de manier die voorgeschreven is voor het gebruikte ondertekenagoritme.
7. Decodeer de JWT payload, dat wil zeggen het tweede deel van het JWT token, vanaf het eerste karakter na de eerste punt, tot aan de tweede punt: `BASE64URLDecode(JWT payload).`
8. Extraheer en decodeer de hash van het bericht: `BASE64URLDecode(urn:edustd:body:hash).`
9. Hash het bericht volgens het algoritme gegeven in de JWT payload `"urn:edustd:body:alg"` en vergelijk deze hash met die uit het bericht. Als beide waarden identiek zijn is het bericht valide.

De onderstreepte onderdelen uit bovenstaande beschrijving worden door JWT library functies uitgevoerd.

5 Versleuteling van berichten

<TO DO, aandachtspunten:

- publiek certificaat van ontvangende partij moet publiek gepubliceerd zijn, hiervoor hebben we infrastructuur (PKI! Functie voor het serviceregister?)
- Encryptie van JWT en JSON: nog niet helemaal uitgekristalliseerd. Parallel met werkwijze voor ondertekening zou zijn:
    - Encrypt het bericht met een symmetrisch algoritme, plaats de sleutel in een claim in de payload van het JWT token
    - Het JWT Token moet nu ook encrypted worden. Encryptie van het JWT token encrypt payload en signature; header blijft leesbaar
    - Voor routering is het nodig dat claims iss, aud en sub leesbaar zijn - deze moeten dus nu (ook) worden opgenomen in de JWT header

>

5.1 Alternatieven en keuzes

5.1.1 Versleuteling van payload, versleuteling van of bericht + payload

5.1.2 Algoritmen voor versleuteling

5.2 Verzenden van berichten

5.2.1 Verantwoordelijkheden van afzender

5.2.2 Verantwoordelijkheden van transparante intermediair

5.2.3 Verantwoordelijkheden van ontvanger

5.3 Ontvangen van berichten

5.3.1 Verantwoordelijkheden van afzender

5.3.2 Verantwoordelijkheden van transparante intermediair

5.3.3 Verantwoordelijkheden van ontvanger

6 Adresseren en routeren

6.1 Adresseren

Op basis van een aantal claims in het JWT token kan een bericht gerouteerd worden. Er zijn in de JWS RFC (https://tools.ietf.org/html/rfc7515) claims gedefinieerd voor de afzender van het bericht, de bestemming van het bericht, een unieke identifier van een bericht en een identifier van een eerdere uitwisseling waar een bericht aan gerelateerd is.

De informatie in deze claims is vergelijkbaar met de WS-Addressing velden die ook in Edukoppeling met SOAP gebruikt worden. Het is verleidelijk om private claims te maken met namen die bekend zijn uit de WS-Addressing standaard. Hiermee wordt in elk geval de herkenbaarheid verbeterd. Echter, de meeste bibliotheken en toolkits die gebruikt worden om JWT tokens mee te maken, hebben standaard functies om afzender en ontvangers van berichten in te vullen en uit te lezen, en gebruiken hiervoor de registered JWT claims. In het kader van gemak en interoperabiliteit is het aan te raden om deze conventies over te nemen.

Om afzender en bestemming van het bericht uniek te kunnen identificeren worden OIN's en OIN's met administratienummers voor respectievelijk organisaties en administraties gebruikt. Voor het bewaken van de integriteit van het bericht is belangrijk dat deze properties wel onder de *ondertekening* van een het JWT vallen, en tegelijkertijd mogen deze properties niet *versleuteld* worden. Als een bericht versleuteld wordt, kan het betekenen dat het JWT ook versleuteld moet worden. Bij versleuteling van een JWT wordt de payload (en daarmee ook de claims iss, aud, sub) versleuteld terwijl de header van het JWT alleen ondertekend wordt, zodat de claims hierin leesbaar blijven. Volgens de JWT RFC (https://tools.ietf.org/html/rfc7519) par 5.3 is het voor dit geval mogelijk om de claims iss, aud en sub in de payload van een token te dupliceren naar de header van het token.

REST-ADR-001	De standaard claims "`iss`" en "`aud`" worden gebruikt om respectievelijk de oorspronkelijke afzender en beoogd ontvanger of ontvangers van berichten mee aan te duiden.	
REST-ADR-002	De claims voor routering worden opgenomen in de payload van het JWT token, en als het JSON bericht en het JWT token versleuteld wordt, ook in de header van het JWT token.	
REST-ADR-003	Voor de aanduiding van afzender en ontvanger wordt de notatie "`edustd:oin:[OIN]`" gebruikt, waarbij [OIN] wordt ingevuld met het 20-cijferig OIN dat aan de organisatie is toegekend.	
REST-ADR-004	Voor de aanduiding van administraties van afzender en ontvanger, in situaties dat er routering naar administraties binnen een organisatie nodig is, wordt de notatie "`edustd:ain:[AIN]`" gebruikt, waarbij [AIN] wordt ingevuld door het 20-cijferig OIN van de	

	organisatie inclusief het administratienummer.	
REST-ADR-005	Voor automatische routering op basis van het type berichtverkeer wordt de namespace van de berichtstandaard opgenomen in de “sub” claim van het bericht.	

Op basis van deze claims kunnen beslissingen over routering worden gemaakt. Een voorbeeld van een bericht met een header en een body:

```
{
	“alg”: “HS256”,
	“typ”: “JWT”
}
{
	“iss”: “edustd:ain:0000000700099AA00123”,
	“aud”: “edustd:oin:00000004012345567890”,
	“sub”: “http://las.vroegtijdigaanmelden.nl/api/v1”,
	“jti”: “B9963D3F-A57B-463A-BF1E-1716F3E2DE0F”,
	“iat”: 1544540142,
	“nbf”: 1544540142,
	“exp”: 1564543742
}
```

6.2 Routeren

<Serviceregister>

7 Bijlage: Claims

De volgende afspraken voor claims in de header en payload van een JWT token worden hierbij gemaakt:

7.1 Claims in de header

Onderstaande tabel toont de verplichte en optionele claims in de header bij gebruik van JWT in het onderwijs. De standaard zelf en de uitwisseling kunnen nog additionele eigen claims verplicht stellen.

#	Claim	V/O ?	Invulling
h1	alg	v	Algoritme voor ondertekening van het JWT token. Zie tabel 2.
h2	type	v	`JWT`
h3	jwk	v	Container voor velden h3.1 t/m h3.3, bevat het PKIo certificaat dat gebruikt is voor de ondertekening van het bericht, inclusief noodzakelijke metadata, formaat volgens RFC7517 (https://tools.ietf.org/html/rfc7517)
h3.1	kty	v	Type van het PKIo certificaat, bijvoorbeeld RSA
h3.2	n	v	Modulus van de publieke RSA sleutel van het x509 (PKIo) certificaat, geëncodeerd met base64url
h3.3	x5c *of* x5u	v	Ofwel de base64url-geëncodeerde pem representatie van de publieke sleutel van certificaat zelf, ofwel een url die naar de pem representatie van de publieke sleutel van het certificaat wijst

Alleen bij versleutelde berichten:

#	Claim	V/O ?	Invulling
h4	iss	v*	Afzender van het bericht (*issuer*), notatie `'urn:edustd:oin' ||` [de OIN van de afzenderorganisatie]. Als deze is opgegeven, is de waarde identiek aan de waarde in de payload.
h5	aud	v*	Geadresseerden van het bericht (*audience*). Hier kan een enkele geadresseerde worden ingevuld in een string-notatie, of een lijst van geadresseerden. De notatie voor geadresseerden is `'urn:edustd:oin' || [OIN van de geadresseerde]`, of `'urn:edustd:ain' || [OIN van de geadresseerde inclusief zijn administratie-kenmerk]`. Als deze is opgegeven, is de waarde identiek aan de waarde in de payload.
h6	sub	v*	Onderwerp van de berichtuitwisseling (*subject*). Hier is mogelijk om hier het service-version kenmerk (d.w.z. de namespace van de service) in te vullen, zodat deze informatie beschikbaar is om additionele gegevens van de service en zijn endpoint in het OSR op te zoeken. Als deze is opgegeven, is de waarde identiek aan de waarde in de payload.

*) Verplicht in het geval van versleutelde berichten, anders optioneel.

7.2 Claims in de payload

Onderstaande tabel vertoont de verplichte en optionele claims in de payload bij gebruik van JWT in het onderwijs. De standaard zelf en de uitwisseling kunnen nog additionele eigen claims verplicht stellen.

#	Claim	V/O?	Invulling
p1	iss	v*	Afzender van het bericht (*issuer*), notatie `'urn:edustd:oin' ||` [de OIN van de afzenderorganisatie] of `'urn:edustd:ain' ||` [de OIN van de afzenderorganisatie inclusief administratie-kenmerk]
p2	aud	v*	Geadresseerden van het bericht (*audience*). Hier kan een enkele geadresseerde worden ingevuld in een string-notatie, of een lijst van geadresseerden. De notatie voor geadresseerden is `'urn:edustd:oin' || [OIN van de geadresseerde]`, of `'urn:edustd:ain' || [OIN van de geadresseerde inclusief zijn administratie-kenmerk].`
p3	sub	v*	Onderwerp van de berichtuitwisseling (*subject*). Hier is mogelijk om hier het service-version kenmerk (d.w.z. de namespace van de service) in te vullen, zodat deze informatie beschikbaar is om additionele gegevens van de service en zijn endpoint in het OSR op te zoeken.
p4	iat	v	Timestamp (milliseconden sinds epoch) waarop het token is aangemaakt (*issued at*).
p5	nbf	o	Tijdstip (milliseconden sinds epoch) vanaf welke het JWT token geldig is (*not before*). Als niet opgegeven, gelijk aan 'iat'.
p6	exp	o	Tijdstip (milliseconden sinds epoch) tot welke het JWT token geldig is (*expires*). Als niet opgegeven, een uur (3.600.000 milliseconden) na iat.
p7	edustd: body	v	Container voor velden p7.1 en p7.2, bevat de ondertekening (hash) van het bericht
p7.1	hash	v	De base64url-geëncodeerde hashwaarde van het bericht zelf, `BASE64URLEncode(HASH(message body))`
p7.2	alg	v	Het algoritme waarmee de hashwaarde van het bericht is bepaald. Zie tabel 1 voor een lijst van toegestane algorigmen

*) Als het bericht versleuteld is en dit veld ook al in de header staat, wordt het toch verplicht in de body opgenomen.

8 Bijlage: Voorbeeld

<TODO bijwerken>
Een voorbeeld van een encoded JWT

eyJhbGciOiJSUzI1NiIsInR5cGUiOiJKV1QiLCJqd2siOnsia3R5IjoiUlNBIiwibiI6InRRcE16aXIzZTB4N0RqNklmczBKenN
JZkhxRWhTSUJTOV9wRjlVUXNHT2dfSlZRZjZuVkVkbFN5VDd2dmZOTjNXZV9ZN1VBdzkyUWxDRHlLaDM1YjQxVk
NGdmNEbHBGSW9jZFBpQk5mMG52Q0pPTmtXWVhUb3RlTWhsNmk4S3U3XzN2NVVWZi1QOGFLZjdZTmt2ZnV
PR093bTNJa3FHZzkweTliTktEczBSc1FPUFZyekwxWm1xXzhzek1iUlZ6MVNtWDRnUkJiMExTQzFPbHNubmRxUWt
2QnZGbDg1SzdMZ3Btd2NMTkJLejhDRkZac2MwcVhuV3VPWm4wM2t2NnNFSnlXNWVncDRSVFE5ZTIwaHI4cDJr
cW40b29ZNXllUmJkSkFqSVItSDB4RjVuSjJFLUxWR2Zfa3NyRUc3WngwWkJyVFp3anpjbkZqM3RhVG45WnlpUSIsI
mUiOiJBUUFCIiwieDVjIjpbIk1JSURyRENDQXBRQ0NRREIMRmtnWEZ1cWpUQU5CZ2txaGtpRzl3MEJBUXNGQURD
Qmx6RUxNQWtHQTFVRUJoTUNUa3d4RlRBVEJnTlZCQWdNREZwMWFXUXRTRzlzYkdGdVpERVRNQkVHQTFVRU
J3d0tXbTlsZEdWeWJXVmxjakVTTUJBR0ExVUVDZ3dKUzJWdWJtbHpibVYwTVJJd0VBWURWUVFMREFsRlpIVmpZ
WFJwYjI0eEVqQVFCZ05WQkFNTUNXeHZZMkZzYUc5emRERWdNQjRHQ1NxR1NJYjNEUUVKQVJZUmRXNXJibTkz
YmtCc2IyTmhiR2h2YzNRd0hoY05NVGd3T1RJMU1UTTFOekF6V2hjTk5EWXdNakE1TVRNMU56QXpXakNCbHpFT
E1Ba0dBMVVFQmhNQ1Rrd3hGVEFUQmdOVkJBZ01ERnAxYVdRdFNHOXNiR0Z1WkRFVE1CRUdBMVVFQnd3S1d
tOWxkR1Z5YldWbGNqRVNNQkFHQTFVRUNnd0pTMlZ1Ym1semJtVjBNUkl3RUFZRFZRUUxEQWxGWkhWallYUn
BiMjR4RWpBUUJnTlZCQU1NQ1d4dlkyRnNhRzl6ZERFZ01CNEdDU3FHU0liM0RRRUpBUllSZFc1cmJtOTNia0JzYjJO
aGJHaHZjM1F3Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRQzFDa3pPS3ZkN1RIc
09Qb2grelFuT3doOGVvU0ZJZ0ZMMytrWDFSQ3dZNkQ4bFZCL3FkVVIyVkxKUHUrOTgwM2RaNzlqdFFERDNaQ1V
JUElxSGZsdmpWVUlXOXdPV2tVaWh4MCtJRTEvU2U4SWs0MlJaaGRPaTE0eUdYcUx3cTd2L2UvbFJWLzQveG9wL
3RnMlM5KzQ0WTdDYmNpU29hRDNUTDFzMG9PelJHeEE0OVd2TXZWbWFyL3l6TXh0RlhQVktaZmlCRUZ2UXRJT
FU2V3llZDJwQ1M4RzhXWHprcnN1Q21iQndzMEVyUHdJVVZteHpTcGVkYTQ1bWZUZVMvcXdRbkpibDZDbmhGT
kQxN2JTR3Z5bmFTcWZpaWhqbko1RnQwa0NNaEg0ZlRFWG1jbllUNHRVWi8rU3lzUWJ0bkhSa0d0Tm5DUE55Y1
dQZTFwT2YxbktKQWdNQkFBRXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBRFNRMnp2SDRud1Q0Ny82WFkze
itSTlkvY0hFK1hjbHZzRm5wTXJSVzQrSlQrQU1LN3pQUWZiNDQ3WktOR3NVcWlQZllFOGo4TEI3TFVyejNsTkxQcF
NwemhHMkF5WVVGSXI5NERISnVWSWtWOHNxajRxZzVUR1QrTEN4c3dhM3E1dU1GSmR4MkpVR0VXSE1JUEF
WYUNZSm50UlJrbDhHM3ZTQnZXK2Z4UEJLWXJsZ0VjbloxT0J1Zjc2MTY3THJnZmNralUxaXFPbE1wQmdvelAzMX
dvMTBDZ0MrWGY2QXZIRllvQzdUM3F0RU1MTzJwUWdkU1BTNHhqYWdFa2NFRWJMekVYc0J6T3drQ0tITWQra
E1vSUdGWkFTa01PRW1RWDhxWXErQThwR1o2SHdTSGtvTzI2SUMzeU5qdDRLL0ZPUEVEUTczNGRtamI5azNUc
0h5LzIwPSJdLCJ4NXQiOiJHODBjY1NPYUw0d3hiRFhYSkhOdm83d3JBSUEiLCJ4NXQjMjU2IjoiNkhpWU1yaVZlNW5
1djViOS12dzZGM2xWVGszUWFLem9IV0NGWGFYdVk3ZyIsImtpZCI6Iktlbm5pc25ldCBzaWduaW5nIGNlcnRpZmlj
YXRlIiwiYWxnIjoiUlMyNTYiLCJ1c2UiOiJzaWcifX0.
eyJpYXQiOjE1NDQ1NDAxNDIsIm5iZiI6MTU0NDU0MDE0MiwiZXhwIjoxNTY0NTQzNzQyLCJhdWQiOiIwMDAwM
DAwMzI3MjQ0ODM0MDIwNCIsImlzcyI6IjAwMDAwMDAzMjcyNDQ4MzQwMTE2IiwiaGFzaCI6ImNZbXR2aWRna
Wh1dkdyOTR5dnBaaEUzSVdUb1p0WEZ6U3FQYWZmeFA5blE9In0.
Xol3X3miKfKLwu7MZf9QcGQ-PF464LJcU8R-oFK-RBc24uy2NZo7w05iuGZOMjFx9Mi5VZak_hdYX2cVHBLz7QUxq
ye4vDa-K4zGrkjKvAc84Rr1TpSl7VWX8TqJFfCiOHLCr2tl3-Gi-mxZ3S-1lllhnvffdVr1LlXvC6ZeRJHPjkib9hTNgFWEH2
BO2tW3T3txDha9vJJyDvGBSFPW2oLhXf3PqWZ2SZC601IVcMDmPKM0oESF81OUGiA7sr7AZbKXWhDvHOVUhW
YMwSHxpZBuSBkCyiUn6ZPZvyFTc01YUH5a51p3rx0t1YfJb1JUQpQHH7M6xq5ODDDtPIQlaw

(regeleinden toegevoegd voor de leesbaarheid)

Gedecodeerd is dit

Header:

```
{
  "alg": "RS256",
```

```
  "type": "JWT",
  "jwk": {
        "kty": "RSA",
        "n":
"tQpMzir3e0x7Dj6Ifs0JzsIfHqEhSIBS9_pF9UQsGOg_JVQf6nVEdlSyT7vvfNN3We_Y7UAw92QlCDyKh35b4
1VCFvcDlpFIocdPiBNf0nvCJONkWYXToteMhl6i8Ku7_3v5UVf-P8aKf7YNkvfuOGOwm3IkqGg90y9bNKDs0
RsQOPVrzL1Zmq_8szMbRVz1SmX4gRBb0LSC1OlsnndqQkvBvFl85K7LgpmwcLNBKz8CFFZsc0qXnWuOZn0
3kv6sEJyW5egp4RTQ9e20hr8p2kqn4ooY5yeRbdJAjIR-H0xF5nJ2E-LVGf_ksrEG7Zx0ZBrTZwjzcnFj3taTn9Zy
iQ",
        "e": "AQAB",
        "x5c": [

"MIIDrDCCApQCCQDILFkgXFuqjTANBgkqhkiG9w0BAQsFADCBlzELMAkGA1UEBhMCTkwxFTATBgNVBAg
MDFp1aWQtSG9sbGFuZDETMBEGA1UEBwwKWm9ldGVybWVlcjESMBAGA1UECgwJS2VubmlzbmV0MRI
wEAYDVQQLDAlFZHVjYXRpb24xEjAQBgNVBAMMCWxvY2FsaG9zdDEgMB4GCSqGSIb3DQEJARYRdW5rb
m93bkBsb2NhbGhvc3QwHhcNMTgwOTI1MTM1NzAzWhcNNDYwMjA5MTM1NzAzWjCBlzELMAkGA1UE
BhMCTkwxFTATBgNVBAgMDFp1aWQtSG9sbGFuZDETMBEGA1UEBwwKWm9ldGVybWVlcjESMBAGA1UE
CgwJS2VubmlzbmV0MRIwEAYDVQQLDAlFZHVjYXRpb24xEjAQBgNVBAMMCWxvY2FsaG9zdDEgMB4GCS
qGSIb3DQEJARYRdW5rbm93bkBsb2NhbGhvc3QwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBA
QC1CkzOKvd7THsOPoh+zQnOwh8eoSFIgFL3+kX1RCwY6D8lVB/qdUR2VLJPu+9803dZ79jtQDD3ZCUIPIqHf
lvjVUIW9wOWkUihx0+IE1/Se8Ik42RZhdOi14yGXqLwq7v/e/lRV/4/xop/tg2S9+44Y7CbciSoaD3TL1s0oOzR
GxA49WvMvVmar/yzMxtFXPVKZfiBEFvQtILU6Wyed2pCS8G8WXzkrsuCmbBws0ErPwIUVmxzSpeda45mf
TeS/qwQnJbl6CnhFND17bSGvynaSqfiihjnJ5Ft0kCMhH4fTEXmcnYT4tUZ/+SysQbtnHRkGtNnCPNycWPe1p
Of1nKJAgMBAAEwDQYJKoZIhvcNAQELBQADggEBADSQ2zvH4nwT47/6XY3z+RNY/cHE+XclvsFnpMrRW4+
JT+AMK7zPQfb447ZKNGsUqiPfYE8j8LB7LUrz3lNLPpSpzhG2AyYUFIr94DHJuVIkV8sqj4qg5TGT+LCxswa3q
5uMFJdx2JUGEWHMIPAVaCYJntRRkl8G3vSBvW+fxPBKYrlgEcnZ1OBuf76167LrgfckjU1iqOlMpBgozP31wo
10CgC+Xf6AvHFR/C7T3qtEMLO2pQgdSPS4xjagEkcEEbLzEXsBzOwkCKHMd+hMoIGFZASkMOEmQX8qYq+
A8pGZ6HwSHkoO26IC3yNjt4K/FOPEDQ734dmjb9k3TsHy/20="
        ],
        "x5t": "G80ccSOaL4wxbDXXJHNvo7wrAIA",
        "x5t#256": "6HiYMriVH5nuv5b9-vw6F3lVTk3QaKzoHWCFXaXuY7g",
        "kid": "Kennisnet signing certificate",
        "alg": "RS256",
        "use": "sig"
  }
}
```


Hierin staan de volgende velden:

1. alg: het gebruikte algoritme voor ondertekening van het JWT, hier RSASSA-PKCS1-v1_5 met SHA-256
2. type: Type van het token, altijd JWT
3. jwk: sectie voor het publieke deel van het certificaat dat de afzender heeft gebruikt voor de ondertekening van de payload. De rest van de velden hebben betrekking op dit certificaat.
    a. kty: type van het certificaat, hier RSA
    b. n: de digitale sleutel van het certificaat, deze wordt gebruikt voor de validatie van de ondertekening. NB deze sleutel is ook onderdeel van het certificaat, maar moet hier expliciet worden opgenomen.
    c. e: vaste string?
    d. x5c: het publieke deel van het certificaat zelf. Alternatief hiervoor is x5u met als waarde de url waar het certificaat kan worden opgehaald.

e. x5t: de fingerprint van het certificaat, op basis van het sha-1 algoritme
f. x5t#256: de fingerprint van het certificaat, op basis van het sha-256 algoritme
g. kid: uniek kenmerk van dit certificaat, waarmee het bijvoorbeeld in de keystore kan worden gelabeld.
h. alg: het algoritme voor de ondertekening die met het certificaat is gedaan, hier RSASSA-PKCS1-v1_5 met SHA-256
i. use: geeft aan of het certificaat bruikbaar is voor ondertekening (sig) of encryptie (enc)

Payload:

```
{
  "iat": 1544540142,
  "nbf": 1544540142,
  "exp": 1564543742,
  "aud": "urn:edustd:ain:0000000700099AA00123",
  "iss": "urn:edustd:oin:00000003272448340116",
   "urn:edustd:msh": "cYmtvidgihuvGr94yvpZhE3IWToZtXFzSqPaffxP9nQ="
}
```

Hierin staan de volgende velden:

1. iat: tijdstip waarop het token is aangemaakt (*issued at*)
2. nbf: vroegste tijdstip waarop dit token geldig is (*not before*)
3. exp: uiterste geldigheid van het certificaat (*expiration*)
4. aud: beoogde ontvanger van het bericht, kan zowel een individuele string zijn als een lijst van strings voor een bericht dat bestemd is voor meerdere ontvangers
5. iss: oorspronkelijke afzender van het bericht
6. urn:edustd:msh: hashwaarde van het eigenlijke bericht waar dit JWT token bij hoort (*message hash*)