Signed Timestamped Open Badges

Het ‘wanneer’ van een digitaal certificaat.

Waar te vinden in vorig ‘Potentieel doelen overzicht’?

Te implementeren onderdelen in grote lijnen	Wel/Niet	Prioriteits advies	Must have	Nice to have	Non-goal	Bron
Signing met 1 signature mechanism (makkelijk te veranderen tegen de tijd dat QC een probleem wordt)	x	1	x			Surf
Secure key storage (hardware modules)	x	1	x			Rapport RR 7.3
Secure key generation (hardware module en/of anders)	x	1	x			Rapport RR 7.5
Timestamping (secure, trustless (onafhankelijk te raadplegen))	x	1	x			Nieuw
Validatie van signed open badge (met uitgever pub-key)	x	1	x			Surf
Meerdere signing **keys** (multisig) (docent + Examen Commissie bijv)	x	2		x		Rapport RR 7.4
Encryptie van signed open badge (met houder pub-key)	x	3		x		Surf
Ontvanger publickey/address implementatie (alternatief EduId), handig voor revocation.	x	4		x		Nieuw
Meerdere signature **mechanisms**		999			x	Rapport RR 7.1
Re-signing van open badges		999			x	Rapport RR 7.2
Quantum resistent algoritme (waar nog geen consensus over is) implementeren.		999			x	Rapport RR 7.1

Waarom Timestamping als oplossing?

Een open badge biedt het **‘aan wie’** en het **‘wat’** van een (onderwijs) claim.

Digitaal ondertekenen biedt het **‘van wie’**.

Nog op te lossen is het **‘wanneer’**.

- Dit **‘wanneer’** is moeilijk te garanderen in de toekomst.
- Een robuuste techniek is dus nodig om het **‘wanneer’** van een badge uitgifte te garanderen.
- Timestamping kan een veilige, lange termijn techniek zijn die het **‘van wie’**, **‘aan wie’** en **‘wat’** van een open badge door de tijd heen garandeert.

Waar komt dat in de praktijk op neer?

- **Altijd zekerheid over moment van uitgifte Open Badge.**
  - Dus zekerheid dat uitgifte van badges die zijn gedaan ná verlies van signing key niet geldig zijn.
  - Te zien welke keys (docent) op een moment een uitgifte heeft gedaan. (Frauduleuze uitgifte is te vinden.)

- **Huidige, niet QC resistente cryptografie, is te gebruiken.**
  - Makkelijk om te wisselen over 10-20 jaar waarbij historische badges nog intact blijven.

- **Timestamping proces is al (deels) in open source code beschikbaar.**

- **Weinig infrastructuur nodig.**
  - Badges zelf zijn stateless.
  - Geen re-signing (contactmomenten).
  - Maar één signature mechanisme implementeren i.p.v. 2 of 3.

- **Quantum resistente cryptografie wordt onderzocht door grotere organisaties.**
  - Resources van SURFnet hoeven daar niet op te focussen.

Welke manier van Timestamping wordt veel gebruikt?

Timestamping via Trusted Timestamping Authorities (TSA's)

EN

Timestamping via een immutable Blockchain (Bitcoin)

Beide manieren bieden timestamping, echter ligt in elk van de manieren het vertrouwen op een andere plek.

TSA's:
Hier ligt het vertrouwen in een (consortium van) 3e partij(en).

Bitcoin:
Hier ligt het vertrouwen in de praktische onmogelijkheid van het aanpassen van de opgeslagen data.

Voor de uiteindelijke implementatie is het zeker belangrijk om hier een goede keus in te maken.

Voor de theoretische 'high-level-view' kunnen we nu eerst even verder.

Wat technische details van zowel TSA’s als Bitcoin

- Hoe ziet een TSA certificaat eruit?
- Hoe werkt timestamping via een TSA?

- Hoe ziet een Bitcoin (timestamp) proof eruit?
- Hoe wordt een proof opgebouwd (merkle tree)?

Voorbeeld van TSA Certificaat (verleend aan Lets Encrypt)

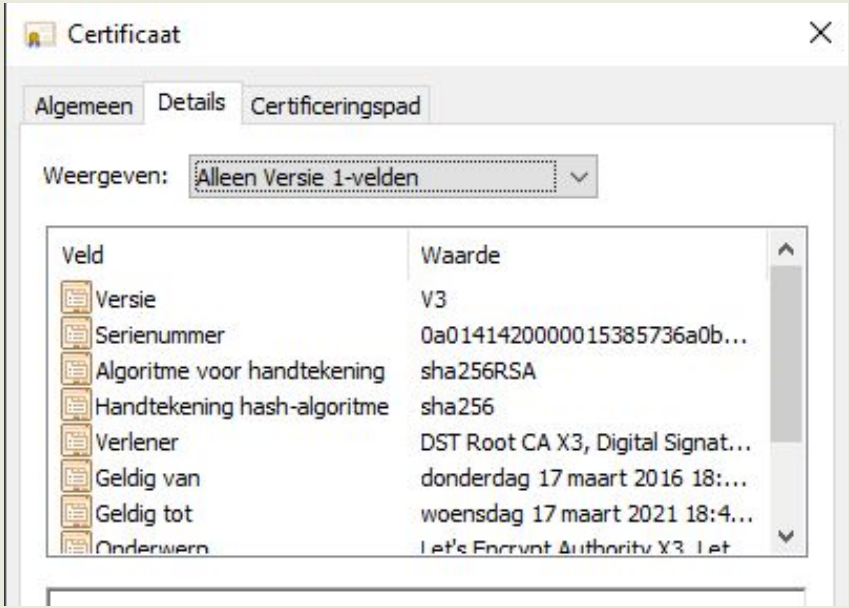
BASE64 ENCODED

-----BEGIN CERTIFICATE-----

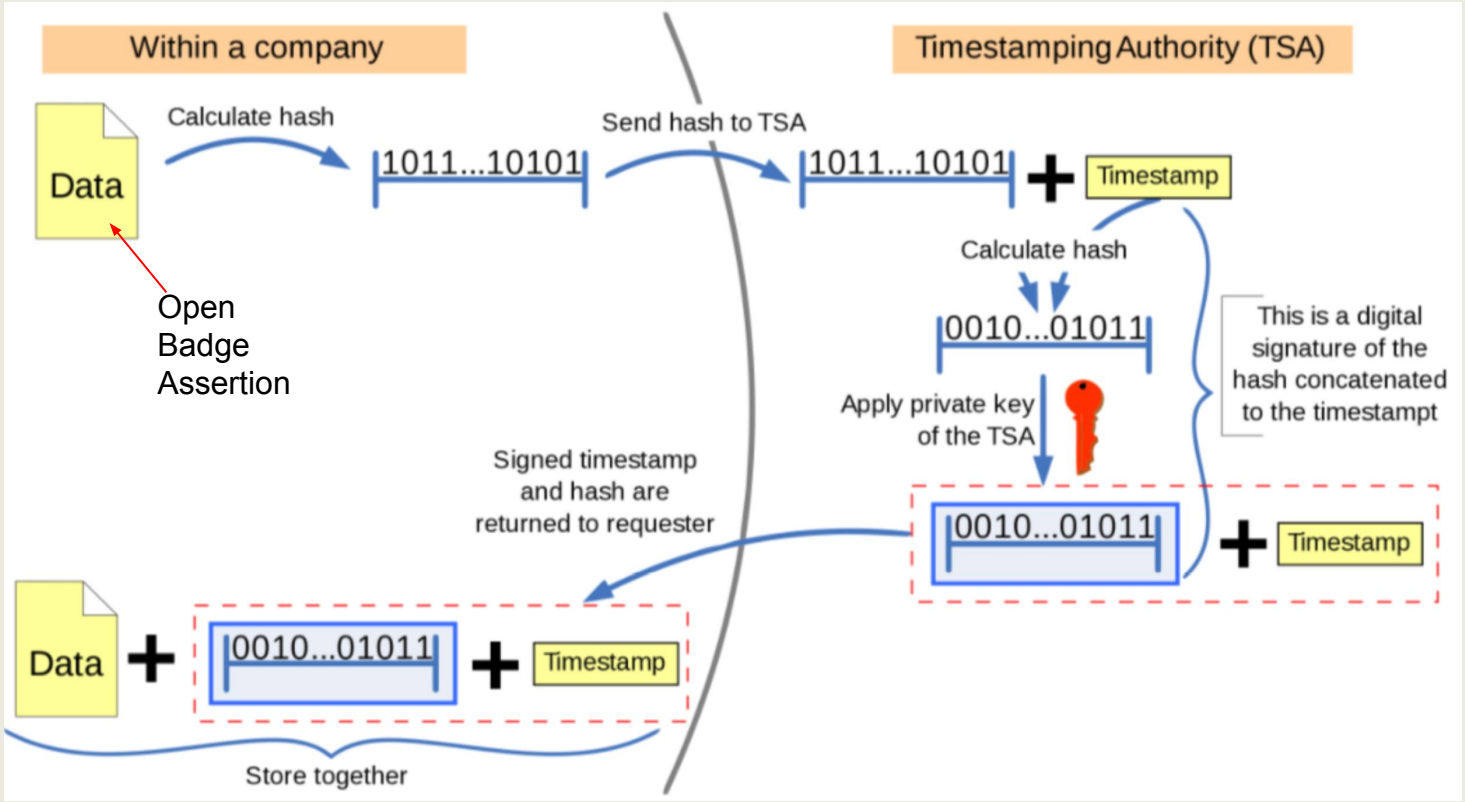
MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
DkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2NDA0NloXDTIxMDMxNzE2NDA0Nlow
SjELMAkGA1UEBhMCVVMxFjAUBgNVBAoTDUxldCdzIEVuY3J5cHQxIzAhBgNVBAMT
GkxldCdzIEVuY3J5cHQgQXV0aG9yaXR5IFgzMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAnNMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF
q6meNQhY7LEqxGiHC6PjdeTm86dicbp5gWAf15Gan/PQeGdxyGkOlZHP/uaZ6WA8
SMx+yk13EiSdRxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZ0
Z8h/pZq4UmEUEz9l6YKHy9v6Dlb2honzhT+Xhq+w3Brvaw2VFn3EK6BlspkENnWA
a6xK8xuQSXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD8bEfwzB/onkxEz0tNvjj
/PIzark5McWvxI0NHWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T
AQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEEczBxMDIG
CCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv
bTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k
c3Ryb290Y2F4My5wN2MwHwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAw
VAYDVR0gBE0wSzAIBgZngQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC
ARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNlbmNyeXB0Lm9yZzA8BgNVHR8ENTAz
MDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWDNDUkwu
Y3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQsF
AAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo
uM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/
wApIvJSwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUgwu
X4Po1QYz+3dszkDqMp4fklxBwXRsW10KXzPMTZ+sOPAveyxindmjkW8lGy+QsRlG
PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6
KOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==

-----END CERTIFICATE-----

.cer file (DECODED)



Hoe werkt Timestamping via een TSA?



Bron: https://en.wikipedia.org/wiki/Trusted_timestamping#Trusted_(digital)_timestamping

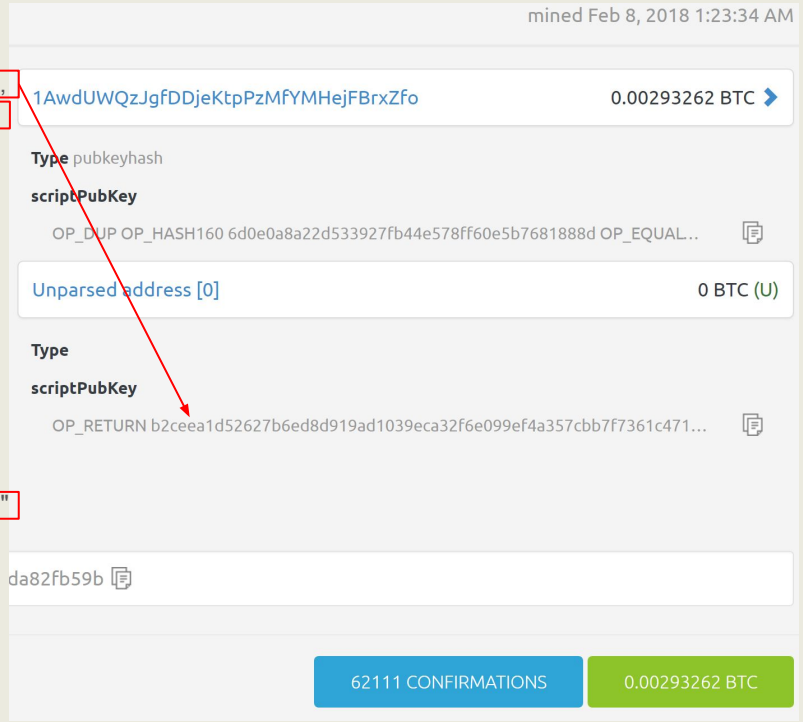
Voorbeeld van een Bitcoin Anchor Proof


```
"signature": {
  "type": [
    "MerkleProof2017",
    "Extension"
  ],
  "merkleRoot": "b2ceea1d52627b6ed8d919ad1039eca32f6e099ef4a357cbb7f7361c471ea6c8",
  "targetHash": "552f01d4fab7da1bce4315c134a1d46e9ef5968f49edf6f5de5d3a2776eea4fb",
  "proof": [
    {
      "right": "776aca4dc61d70480fb05e4d95aaedc719fedd752eab7d517e04af2f481f92af"
    },
    {
      "left": "6613ea6c78b0d93a45c725f3fcc9f2312181ffad0a6833299663d6aa1d7806a9"
    },
    {
      "right": "e780cd2fe41df361fa7c047191533fe6252ea20bcd0fc8b226da3656d1133e56"
    }
  ],
  "anchors": [
    {
      "sourceId": "2378076e8e140012814e98a2b2cb1af07ec760b239c1d6d93ba54d658a010ecd"
      "type": "BTCOpReturn",
      "chain": "bitcoinMainnet"
    }
  ]
}
```

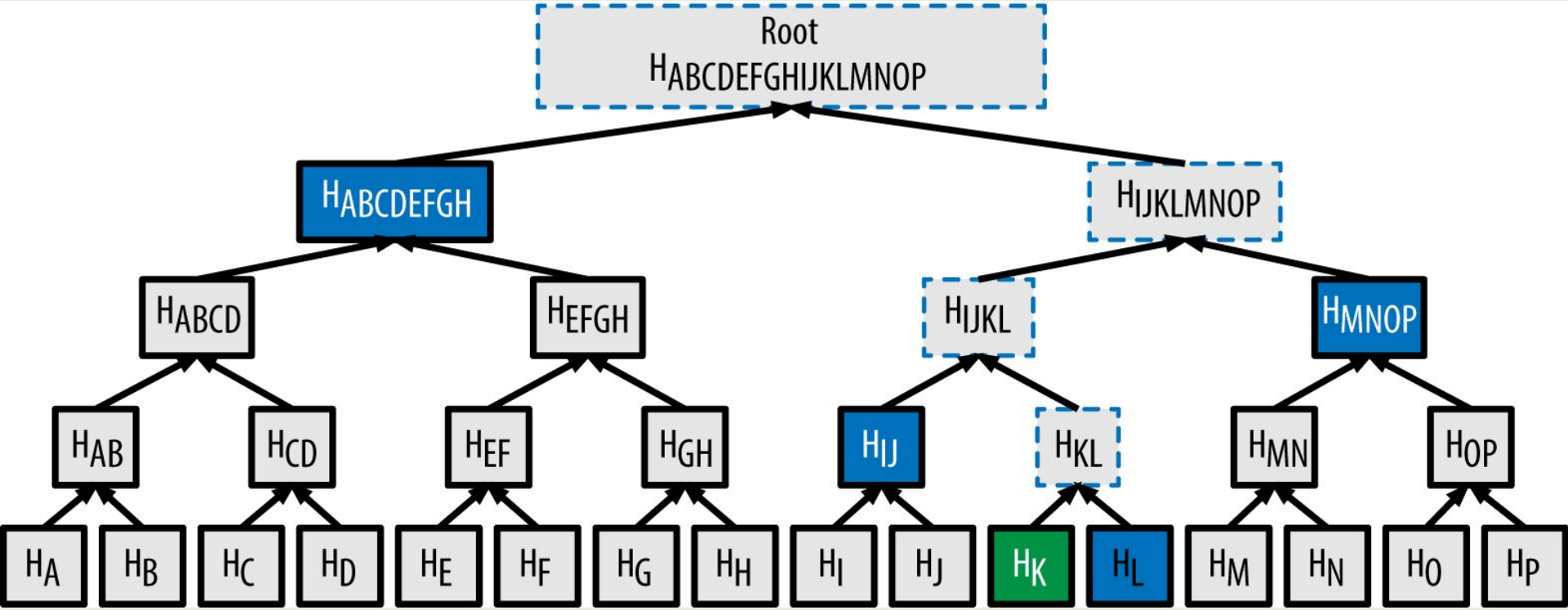

Assertion Hash

Op welk netwerk protocol de transactie staat

Transaction hash waarop te zoeken valt



Wat is een Merkle Tree?



Bron: https://github.com/chainpoint/whitepaper/issues/5

Assertion hash van 1 open badge

Een aantal belangrijke eigenschappen van Timestamping

- Onafhankelijk te verifiëren?
- Backdaten onmogelijk?
- Kosten van timestamp?
- Geldigheid timestamp?
- Extensie (voor Open Badge) beschikbaar?
- Extensie (voor Open Badge) nodig?
- Wachttijd timestamp?

Onafhankelijke verificatie

Voor verificatie van ‘wanneer’ de Assertion uitgegeven is, is contact met de Timestamping service nodig.

- In het geval van een TSA:
  - Moet de TSA nog bestaan en benaderbaar zijn.

- In het geval van Bitcoin:
  - Kan iedereen die een Bitcoin client draait (of één van de tientallen indexing websites) zelf verifiëren.

Is backdaten mogelijk?

Backdaten is het maken van een nep timestamp certificaat voor een bepaald stuk data.
Zo zou iemand een Open Badge kunnen timestampen met een datum van bijvoorbeeld 10 jaar terug als ze de TSA signature keys weten te bemachtigen.

- In het geval van een TSA:
  - Na verloop van tijd kan het zijn dat de signature keys van een TSA in verkeerde handen vallen.
  - Dan kunnen nep timestamps worden gemaakt over stukken data. (Als er daarna nog gesigned moet worden door een Issuer is het risico wel kleiner, want de Issuer, gaan we vanuit, controleert ook de datum van de timestamp op het moment van signen).
  - Een groter probleem is dat de historische badges niet meer te vertrouwen zijn bij een gekraakte TSA key.

- In het geval van een Bitcoin:
  - Backdaten is praktisch onmogelijk. De kosten daarvan zijn minimaal ~€1 miljoen voor 1 uur terugdraaien en ~€50 miljard voor een paar jaar terugdraaien (wat direct gezien wordt).

Wat zijn de kosten voor timestamping?

De kosten van Timestamping kunnen we onderverdelen in infrastructuur/implementatie kosten en kosten voor de timestamps zelf.

- In het geval van een TSA:
  - Implementatiekosten zijn hoger.
    - Er is nog geen extensie voor TSA certificaten in OB.
    - Er is nog geen validatie van TSA certificaten voor OB.
    - Implementatie met een TSA server.
  - Timestamp transactie kosten verschillen per TSA, maar dit is meestal in de enkele centen.

- In het geval van een Bitcoin:
  - Implementaties (open source) bestaan al. Deze zijn, helemaal voor een Pilot, te gebruiken.
  - Transactie kosten kunnen gratis zijn (er zijn een aantal calendar services).
  - Bij zelf transacties doen, kan het geaggregeerd worden. Dan is het fracties van centen.

Hoelang is een timestamp ‘geldig’?

Voor de toepassing van Open Badges is een geldigheid van > 50 jaar geprefereerd. Het verschilt per timestamping technologie wat mogelijk is.

- In het geval van een TSA:
  - Er worden eigenlijk geen certificaten meer uitgegeven die langer dan 2 á 3 jaar geldig zijn.
  - Lange garantie van timestamp certificaten is ook moeilijk voor TSA’s.

- In het geval van een Bitcoin:
  - In principe zijn de timestamps geldig zolang er een significant gebruik ik van Bitcoin.
  - Mocht Bitcoin ooit niet meer gebruikt worden (vanwege overgaan op een andere (blockchain) technologie bijvoorbeeld) dan is de historische staat van Bitcoin (en dus de Open Badge timestamps daarin) alsnog op te slaan in een volgend systeem.

Zijn er extensies nodig voor de Open Badge?

Voor zowel TSA als Bitcoin timestamping van Signed Open Badges is een extensie (en de daarbij horende validatie) nodig.

- In het geval van een TSA:
  - Hier bestaat nog helemaal niets voor qua gebruik én implementatie.

- In het geval van een Bitcoin:
  - Er zijn al extensies gemaakt die voldoen aan de JSON-LD standaard die in gebruik zijn of gebruikt kunnen worden in een Open Badge structuur.

Wat is de wachttijd van het timestampen?

Het uitgeven van elke Open Badge kost een bepaalde hoeveelheid tijd. Dit verschilt per manier van implementatie. Het maken van de Assertion is hierin niet meegenomen. Dat is voor alle implementaties gelijk. Hier kijken we specifiek naar het timestampen.

- In het geval van een TSA:
  - Dit duurt hoogstens een paar seconden per Assertion.

- In het geval van een Bitcoin:
  - Er zijn calendar services (ook gebruikt door bijvoorbeeld Microsoft en Philips) die data vastleggen op Bitcoin binnen een paar seconden. Hierbij moet echter wel later op een andere (niet geprefereerde manier) de proof gevalideerd worden.
  - Diezelfde calendar service kan ook gecheckt worden na ~1 uur, waarna een normale proof + validatie opgenomen kan worden.
  - Zelf een transactie doen van (een batch van) Open Badges is ook mogelijk. Ook daarbij geldt ~1 uur als richttijd.

Een vergelijking van de meest belangrijke punten voor Timestamping van Signed Open Badges

	TSA	Bitcoin
Onafhankelijk te verifiëren	nee	ja
Backdaten onmogelijk	nee	ja
Kosten	paar cent	gratis of tot paar cent per badge
Geldigheid timestamp	< 3 jaar	praktisch oneindig
Extensie beschikbaar	nee	ja
Extensie nodig	ja	ja
Wachttijd timestamp	paar seconden	paar seconden (calendar service) tot ~1 uur

Advies voor Timestamping methode

Kijkend naar de hiervoor beschreven onderdelen van timestamping is het advies:

Gebruik Bitcoin als Timestamping implementatie voor Signed Open Badges.

Volgende stappen en onderzoek

Er zijn een aantal onderdelen die nu praktisch (Proof of Concept) onderzocht kunnen worden om een volle flow van Timestamping (Signed Open Badge) te laten zien:

1. Gebruiken van Bitcoin als timestamp anchor
2. JWS Compact Serialisation als signing mechanisme
3. RSA en/of EdDSA als signing algoritme

En er zijn uiteraard nog vragen die beantwoord moeten worden. Deze zijn in groepen in te delen (detail in verdere slides):

1. Key management (generen, opslaan, signen)
2. Validatie en publicatie van Open Badge en public keys
3. Badge beheer (voor badge ontvangers)
4. Encryptie en decryptie van (privacy gevoelige) assertion onderdelen
5. Overige vragen

Voorbeeld implementatiestappen in Proof of Concept

Om een voorbeeld van de (lineaire) flow van Timestamped Signed Open Badge uitgifte te laten zien zijn hiernaast wat (niet geïmplementeerde) functies te zien. →

Uiteraard omvat het meer dan alleen dit (bijv. helper classes, aparte (deel) apps voor verschillende functionaliteit, etc.)

```
class TimestampedSignedOpenBadge:

    def create_assertion(self, base_context, student_info, issuer_public_key):...
    def hash_assertion(self, assertion):...
    def get_latest_merkle_tree(self):...
    def merkleize_assertion(self, latest_merkle_tree, assertion_hash, student_id):...
    def get_proof_path(self, assertion_hash):...
    def get_merkle_tree_root(self, latest_merkle_tree):...
    def store_merkle_tree_root_on_bitcoin(self):...
    def get_latest_transaction(self):...
    def check_confirmations(self, latest_transaction):...
    def create_anchor_proof(self, latest_transaction, proof_path):...
    def combine_assertion_and_anchor_proof(self, assertion, anchor_proof):...
    def create_jws_header(self, signing_algorithm):...
    def sign_header_and_payload(self, header, payload, private_key, algorithm):...
    def get_badge_image(self, assertion):...
    def bake_badge(self, jws_signature, badge_image):...
    def unpack_baked_badge(self, bake_badge):...
    def verify_jws_signature(self, jws_signature, public_key):...
    def verify_anchor_proof(self, unbaked_badge):...
    def verify_finished_badge(self, finished_badge, public_key, latest_transaction):...
    def add_public_key_to_issuer_web(self, public_key, issuer_url):...
    def send_finished_badge_to_recipient(self, student_info, finished_badge):...
```

Verder onderzoek (1/5)

- Key management:
  - Wie genereert de keys
  - Hoe worden de keys gegenereerd
  - Waar worden de keys gegenereerd
  - Waar worden keys opgeslagen
  - Wie managet de keys
  - Wie signed daadwerkelijk

Verder onderzoek (2/5)

- Validatie en publicatie van Open Badge en public keys
  - Waar wordt validatie gedaan
  - Hoe is volledig onafhankelijke validatie mogelijk
  - Wie publiceert de public keys
  - Waar worden de public keys gepubliceerd
  - Hoe wordt door verloop van tijd de public key list bijgehouden

Verder onderzoek (3/5)

- Badge beheer (voor Badge ontvangers)
  - Hoe kunnen ontvangers hun badge behouden
  - Hoe kunnen ontvangen hun badges delen
  - Hoe bewijst een badgehouder dat hij/zij ontvanger is

Verder onderzoek (4/5)

- Encryptie en decryptie van (privacy gevoelige) assertion onderdelen
  - Hoe kan een badge onderdeel ge-encrypt worden
  - Hoe kan een badge onderdeel gedecrypt worden
  - Wat is er van de ontvanger nodig om een badge te encrypten/decrypten

Verder onderzoek (5/5)

- Overige:
  - Wie doet het timestampen/anchoring
  - Hoe is multisig binnen JWS compact serialisation te implementeren
  - Hoe is een eduID te implementeren
  - Hoe wordt op termijn gekozen voor een QC resistent algoritme
  - Hoe groot kan/mag een signed open badge zijn (bytes)
  - Wat is er nodig voor issuers om volledig zelf te kunnen uitgeven
  - Hoe wordt revocation gedaan

Volgende stappen

Advies is om te vervolgen met twee hoofdonderdelen:

- Uitwerken Proof of Concept voor Timestamped Signed Open Badge.
- Onderzoeken en beantwoorden van verdere benodigde onderdelen.

Voorbeeld versimpelde openbare lijst van public keys

Van	Tot	Instelling	Key type	Public key
1-4-2019	2-4-2019	Rijks Universiteit Groningen	ecdsa-sha2-nistp256	ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNo.....FAdjMCgMAFPnbJx1cZTSORlzPQydBNN0rZI=
2-4-2019	3-4-2019	Hanze Hogeschool	ecdsa-sha2-nistp256	ecdsa-sha2-nistp256 AAAAE2VjZ.....X8iNQK4wtAiHvtbsvILCDgq+n5NQH7Y=
3-4-2019	4-4-2019	Universiteit Utrecht	ecdsa-sha2-nistp256	ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItb.....XYaxlE7AKliZr1A8/E/Uq/L5JH/KLQMsA7X1g=

De overheid van Malta doet zoiets al voor hun Timestamped Signed Open Badges:
https://education.gov.mt/en/Blockcerts/Pages/Blockcerts-Public-Key-Registry.aspx

Zoiets zou SURFnet of DUO (en op termijn de Uitgevers) ook kunnen doen.