Welk signing algoritme?

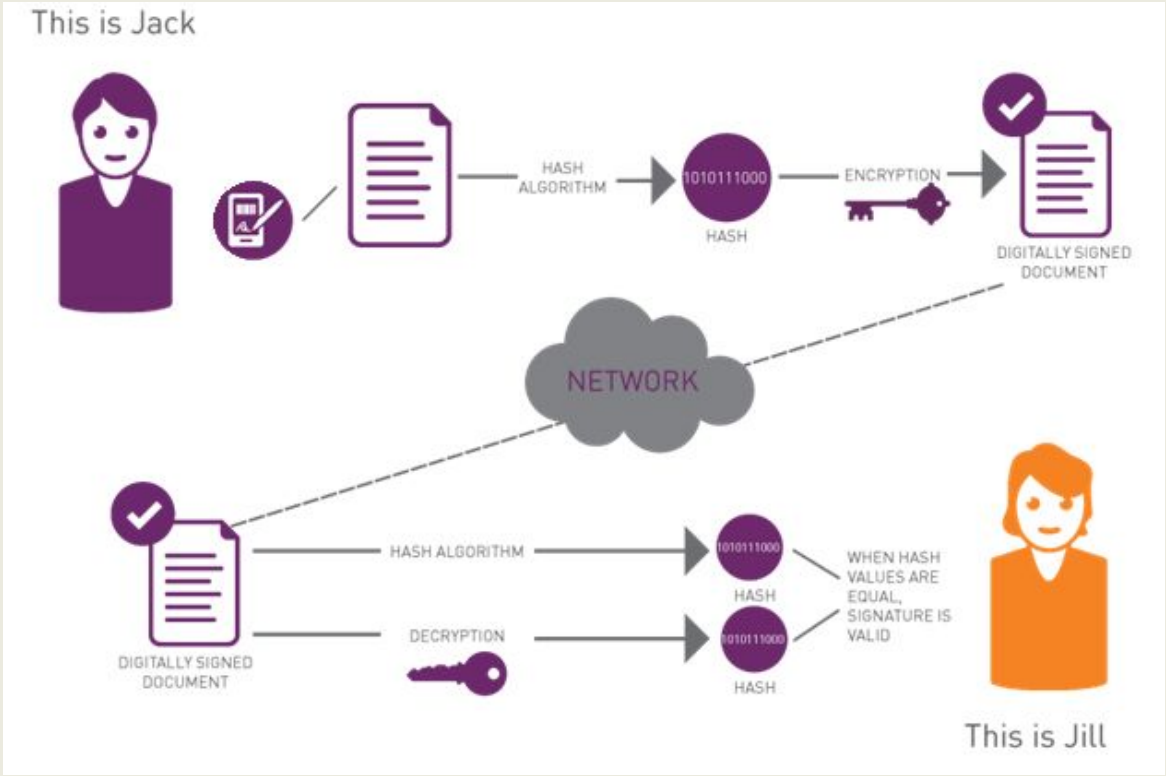
Een vergelijking van verschillende signing algoritmes voor Signed Open Badges

Het ‘wie’ 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

Hoe werkt digitaal ondertekenen ook alweer?



Bron: https://blog.gemalto.com/security/2017/12/19/lets-talk-pki/

Wat voor eigenschappen hebben signing algoritmes?

1. **Key size** (in bytes) van (a) public & (b) private keys.
2. **Signature size** (in bytes).
3. Key **generation speed** (in ms).
4. **Signing speed** (in ms).
5. **Verification speed** (in ms).
6. Mogelijkheid van **opslag** op een Hardware Security Module (**HSM**).
7. **Security** van het algoritme (**in bits**).
8. **Implementation difficulty** (hoe moeilijk en foutgevoelig is een implementatie (te maken)?)
9. **Exposure** to side channel **attacks**.
10. **Collision resistance** (wanneer er een slechte random # generator(RNG) wordt gebruikt, wat is dan de kans op collision?)
11. **Batch verification** optie (meerdere digitale handtekeningen tegelijk controleren).
12. **Quantum resistance** (presumably).

Welke zijn belangrijk voor Signed Open Badges? - **In het begin:** 1a, 2, 6, 7, 8. **Later ook:** 4, 5, 9, 10, 11

Aan de hand van de vorige lijst eigenschappen een keuze om de volgende vier signing algoritmen te vergelijken:

EdDSA (ed25519 curve)

ECDSA (secp256k1 curve)

RSA 3072

SPHINCS-256 with SHA512

Een vergelijking van de signing algoritmes gebaseerd op de hiervoor beschreven eigenschappen.
Focus op **EdDSA**

COMPARISON OF PKI TYPES (FOR SIGNED OPEN BADGES)	Privatekey size (bytes)	Publickey size (bytes)	Signature size (bytes)	Keygeneration speed (ms)	Signature speed (ms)	Verification speed (ms)	HS module possible	Security (bits)	Implementation difficulty	Exposure to side channel attacks	Collission resistance	Batch verification	Presumably QC resistant
EdDSA - ed25519 curve	48	44	64	0,293	0,097	0,200	Yes	128	Low	Low	Yes	Fast	No
ECDSA - secp256k1 curve	144	88	71	0,162	0,331	0,189	Yes	128	High	High	No	Slow	No
RSA 3072	1794	422	384	1964,065	21,46	0,413	Yes	128	Medium	High	No	Medium	No
SPHINCS-256 with SHA512	1135	1097	41000	8,201	137,762	1,593	No	128	High	Medium	Yes	Unknown	Yes

- **EdDSA (ed25519)**
  - Op dit moment het snelst groeiende asymmetrische algoritme in gebruik (bijvoorbeeld aangeraden bij Gitlab).
  - Dit is niet vreemd, aangezien het op zo goed als alle fronten uitblinkt.
  - Het heeft (op quantum computing resistentie na) de voordelen maar niet de nadelen van andere algoritmes.
  - Maar omdat het relatief nieuw is (in het langzaam ontwikkelende domein van de cryptografie) is het nog niet gestandaardiseerd.

Een vergelijking van de signing algoritmes gebaseerd op de hiervoor beschreven eigenschappen.
Focus op **ECDSA**

COMPARISON OF PKI TYPES (FOR SIGNED OPEN BADGES)	Privatekey size (bytes)	Publickey size (bytes)	Signature size (bytes)	Keygeneration speed (ms)	Signature speed (ms)	Verification speed (ms)	HS module possible	Security (bits)	Implementation difficulty	Exposure to side channel attacks	Collission resistance	Batch verification	Presumably QC resistant
EdDSA - ed25519 curve	48	44	64	0,293	0,097	0,200	Yes	128	Low	Low	Yes	Fast	No
ECDSA - secp256k1 curve	144	88	71	0,162	0,331	0,189	Yes	128	High	High	No	Slow	No
RSA 3072	1794	422	384	1964,065	21,46	0,413	Yes	128	Medium	High	No	Medium	No
SPHINCS-256 with SHA512	1135	1097	41000	8,201	137,762	1,593	No	128	High	Medium	Yes	Unknown	Yes

- **ECDSA (secp256k1)**
  - Dit algoritme wordt gebruikt in Bitcoin, wat extra/andere use cases mogelijk maakt voor signed open badges.
  - ECDSA is gestandaardiseerd en al langere tijd in gebruik.
  - Al is de secp256k1 curve uniek voor Bitcoin, vanwege een theoretische onveiligheid i.v.m. de ontwikkelaars (NSA) van een andere veel gebruikte curve in ECDSA.
  - Eigen implementatie (t.o.v. een bestaande library) is wel een stuk meer ingewikkeld dan een aantal andere algoritmen (mede door complexiteit van het algoritme en de noodzaak van een sterk RNG).

Een vergelijking van de signing algoritmes gebaseerd op de hiervoor beschreven eigenschappen.
Focus op **RSA**

COMPARISON OF PKI TYPES (FOR SIGNED OPEN BADGES)	Privatekey size (bytes)	Publickey size (bytes)	Signature size (bytes)	Keygeneration speed (ms)	Signature speed (ms)	Verification speed (ms)	HS module possible	Security (bits)	Implementation difficulty	Exposure to side channel attacks	Collission resistance	Batch verification	Presumably QC resistant
EdDSA - ed25519 curve	48	44	64	0,293	0,097	0,200	Yes	128	Low	Low	Yes	Fast	No
ECDSA - secp256k1 curve	144	88	71	0,162	0,331	0,189	Yes	128	High	High	No	Slow	No
RSA 3072	1794	422	384	1964,065	21,46	0,413	Yes	128	Medium	High	No	Medium	No
SPHINCS-256 with SHA512	1135	1097	41000	8,201	137,762	1,593	No	128	High	Medium	Yes	Unknown	Yes

- **RSA (3072)**
  - Van de bekeken algoritmes is deze het langst in gebruik.
  - De voornaamste nadelen zijn grote keys en langzaam signen.
  - RSA is ook mogelijk met een grotere (veiligere) key, **RSA 4096** bijvoorbeeld (die is echter maar 1 bit meer veilig (129 vs 128) dan EdDSA).
  - Maar is minder practisch vanwege de nog grotere key size en signing snelheid.
  - Echter is RSA zeker een optie als key size en signing snelheid niet een limiterende factor zijn (bijvoorbeeld in een Pilot setting) of een gestandaardiseerd algoritme nodig is.

Een vergelijking van de signing algoritmes gebaseerd op de hiervoor beschreven eigenschappen.
Focus op **SPHINCS**

COMPARISON OF PKI TYPES (FOR SIGNED OPEN BADGES)	Privatekey size (bytes)	Publickey size (bytes)	Signature size (bytes)	Keygeneration speed (ms)	Signature speed (ms)	Verification speed (ms)	HS module possible	Security (bits)	Implementation difficulty	Exposure to side channel attacks	Collission resistance	Batch verification	Presumably QC resistant
EdDSA - ed25519 curve	48	44	64	0,293	0,097	0,200	Yes	128	Low	Low	Yes	Fast	No
ECDSA - secp256k1 curve	144	88	71	0,162	0,331	0,189	Yes	128	High	High	No	Slow	No
RSA 3072	1794	422	384	1964,065	21,46	0,413	Yes	128	Medium	High	No	Medium	No
SPHINCS-256 with SHA512	1135	1097	41000	8,201	137,762	1,593	No	128	High	Medium	Yes	Unknown	Yes

- **SPHINCS (256 SHA512)**
  - Dit is één van de post-QC opties die bruikbaar zou kunnen zijn voor signed open badges.
  - Dit is echter wel maar één van negen post-QC algoritmen die in de running zijn voor standaardisatie.
  - Er is nog geen consensus over welke het beste is (en geen bugs bevat) van de post-QC algoritmen.
  - Wat wel positief is aan dit algoritme is dat het state**LESS** is.
    - In vergelijking met andere post-QC algoritmen (zoals XMSS) biedt dit het voordeel dat alles binnen de badge contained kan worden..
  - Het algoritme is niet niet ‘battle tested’, maar heeft een goede kans om later (5 - 10 jaar) gebruikt te worden .

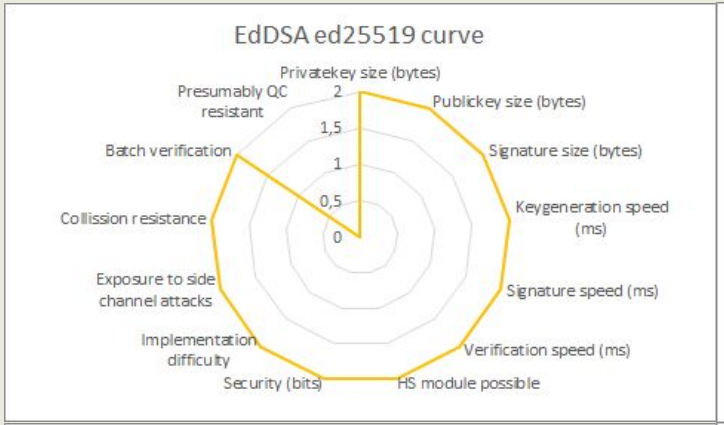
Voor de volledigheid

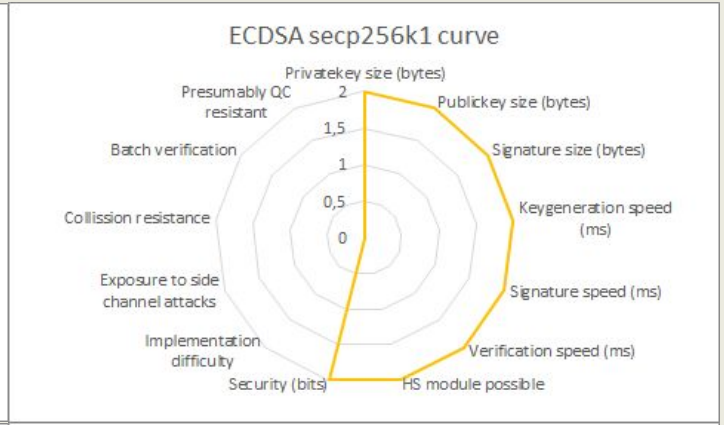
In dit schema zijn uiteraard niet alle PKI algoritmen opgenomen.

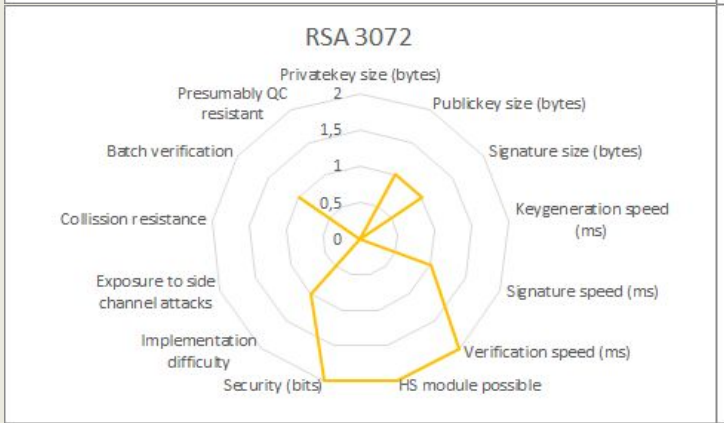
De keuze voor vergelijking van deze 4 is onder andere vanwege:

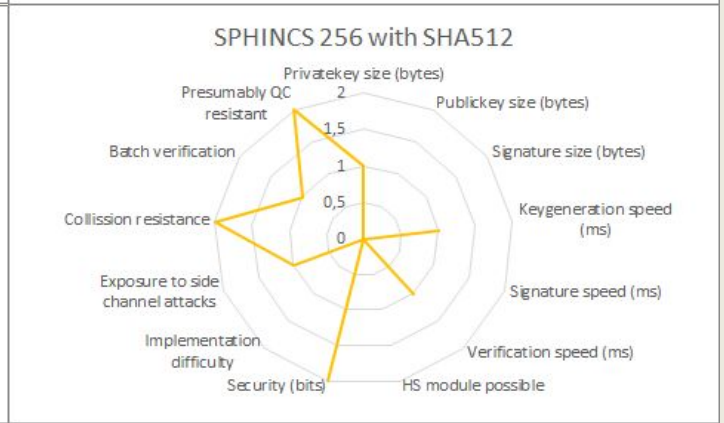
- Beste eigenschappen in algoritme-type (EdDSA) of,
- Unieke eigenschappen (ECDSA) of,
- Meest ver in ontwikkeling van het type (SPHINCS) of,
- Al lang in gebruik en gestandaardiseerd (RSA) en,
- Voldoen aan de benodigde eigenschappen.

Vergelijking algoritmes op alle eigenschappen.









Belangrijker: Vergelijking op **benodigde** eigenschappen.

Beste keus

Voor Pilot ook mogelijk

Vooral een optie bij een Blockcerts (-achtige) implementatie

Advies voor signature algoritmen

- Als Bitcoin adressen of keys nodig zijn:
  - gebruik dan **ECDSA secp256k1**.

- Als een gestandaardiseerd algoritme nodig is (of iets voor een Pilot):
  - gebruik dan **RSA 3072 of RSA 4096**

- Geen Bitcoin adres/key nodig maar wel in productie/dienst:
  - gebruik dan **EdDSA 25519**.
    - (https://ed25519.cr.yp.to/) ← referentie implementatie, onderzoek en docs.

Public key, private key en signature voorbeelden van verschillende algoritmes.

EdDSA private key voorbeeld

AAAAIADCUjFkToJxJNeCcv/NmBhKL8EuGBKgbBIT/tBO9AEU

EdDSA public key voorbeeld

AAAAC3NzaC1lZDI1NTE5AAAAIL1E9Vyn/dHY50wjXbFBDxp8JA/EdS8mS2NNy78TrQZ2

EdDSA Signature voorbeeld

3d550c158900b4c2922b6656d2f8057289de4ee65043745179685ae7d29b944d

ECDSA private key voorbeeld

MHgCAQEEIQCLMHboiZFqy72QNAXkwT3qbsh8TfB/dC0bBD9VmhUbe6AKBggqhkjOPQMBB6FEA0IABBxn80a1SDhk8wunswInh9t7fYwEPwbgqlivoqMp4hArG/GKnK+t//51/chgctuAAhvhzgiwYUa1GeuLf3yZsU8=

ECDSA public key voorbeeld

AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBxn80a1SDhk8wunswInh9t7fYwEPwbgqlivoqMp4hArG/GKnK+t//51/chgctuA
AhvhzgiwYUa1GeuLf3yZsU8=

ECDSA Signature voorbeeld

A704AC7E6586D21E310240D2860C5F9C7E7DFA13FD0FDC05B8C3BF250BA06CC7

RSA 3072 private key voorbeeld

AAABgHhpG+x/5k2A/DVe1QoRVNi1gJiq+URcFk064oAfGMOR3ywQ98/MSV6y5x6F
G5DPOJk0KqRY0JboNy1qx/PzOKAF2hPbJGdYPo0UD5NqgmNPlk0xoQnzMB2JtJqb
35RS1agx8XU0HnWGUmOZnZnmxLR86xvpsY8SA07xyKP+fMbapO89tA2ng3c9VXoq
L0LY6UtWL2kdGLF3pz2SptukB1+zyo9VlNv5XgtDauuNYBylJNG4showwm9pDfdz
fvkZi8o5aSk64nCIDZgOyIs/2pjA6iEr0S6TdaTu8aRRQ/ja4PqmZ+huok08skxk
10zd7EQLyZWvf/h6OCKrDMlqsUiAEuJDdHY2yPr4Z65ccTuR6ZoI72A4eBkaTtcv
bKSy0dC2BdYiWfO6FEgUxTdseYyx6ArVMgWV5TUeYJM8WDgHjl6wP8nlThDLVpng
llko1Cp72yWZyJl+CuNq+tQcLGrBR6QbqOJ+kiZnDCt2pjf2WVxWxw/AyOhXwUf5
EPeSdQAAAMEA02ZsUEpmCD3XYxmrgOu2GmMWFxT6Ndci4kjdO+YlVbFYz2LiluBj
Uw81wgqmWl8jEXRUHs7aFNVs48QiupBKlUqYcYnyl8Fw9Jndw3kLntbVxaU4RqiI
oHBTozCRnZUiBAkWB55SUR0JdBY45LxY1IpC5bn+jLTkyvoGBUGuuEHJI6ovHHl3
d2vqZluMwA8K0acEXJRIazUI+R5HWc7l5Py6gX2tlsnHONjDs4JQ1vXLZUVj3y8Q
zWnB3jGeexhfAAAAwQCuCUGQDDZn1JuSLuE/nv8X4tYbUuwR59HxW3NNWTtrUIc9
qjI6Evdyi7g6gVCO/sc4Ate7R04HAv+8Ji98oSOB9pgosjdS+r1wZP0I9tJxFb5D
aPuSlxIF8h04OLH3pwE+y6Touyohe3xt4kSSoLVNmXDExZtciAA//veHhqSPPtnA
BuZUGEAgw8DMvjRoRsttbY/b1tG6qfby0KZn0x86s2skHPZnHnrST6/lANkcyI+s
lz48fUYcXzlpI5ztFbUAAADAXFJ2nLAYr3cAYu0T2mq48S6imDXLnri24tmB/L4O
tg7A8cKJ48qZrRaJdK6/Wz9otoEtQOW0YsBn+uCqQmUu4CMswAad1Xv8SUkVe2Gy
QlYNWhgDhiF4bHvyb+RHcMe84VaP0mqx4AXjZ8WNmRFHUhlOeOAZku3mubeGu6j+
MtdaQTl+dMRMaRiCNbdz8VJjX5lAfVymxxRw9z30Hr8JtqBFmMPebGyzTOL4lB6B
dVNxTwUPsGJBUKCsBUWymIkx

RSA 3072 public key voorbeeld

AAAAB3NzaC1yc2EAAAABJQAAAYEAj7dCXFaXAap3WHlx4rnY4Z7TStyU3fpMKqEe
2vvTPACvhy0GvjXj9SgcHCtB7ubP8KmeQAbocgznBKi9Guh9ZCgEST9c/3nXC3sK
VRwG+qlQS55U2lPOEr0h4dLRC9688goPrOugObDV8r+bEocLzym9xnoFeTaIE+bG
Lxb4Bh28mWqtnK8pTEDyaITuFfqi1cn2hbdfiX5LvRpkDmjwBt7Y/ag1zL5G9KtO
EOKLfOSPCtQvaZUJGZ5rgi8C5z+HUcS3xznLvJPNgXs6C+dvx49FtNYA+zUmZvMN
2cFnjzBocgU9zTVWSxTsFaI8Awvzw/jaY3QC6ZOW2c9AaUNaAguZuM69LYGy62Ek
/6RlNf3UJAAXv7SNFu0yDcsoWc+AyFMlAHH4U25cBnRgmHttqQiO0DO81yFZtbom
fpR6gh5M0JP2OEBOk3Txh6nwefouUMEubY4BUK8+dz0Q39Qpnxr3BmDQqSR83ieS
oxkJHcNaXpZmaEgdanPJMeGZ+wYr

RSA 3072 Signature voorbeeld

1953f607aeac9b2a910cbf4dc6889c1a61b3e60fe25c90b6dc5a17e3f3d3c8
8f2127f970ff9c08ecc56c0d71b75f7b6a2349032e14756255853c1949a5f5
399b115568ef74aeb3eb1b2889e4f0ec9b289cdc8525c9eefcb9b27b35dc63
c085ef9c23b6500366b5096001b896a8d59b689ecc953d7ac294ee413db0fc
9384b65bcc86f5f10ee5a826f11d114333a5eb78c205cb4a6a22bf224da927
84b6e8b414d1508532beb091fd34f5aac33e6f450374bf6ee990d7e198c181

SPHINCS private key + public key

Voor dit algoritme lukte het niet om zo snel om een werkende implementatie te vinden.

Daarnaast, als karakters zou de signature niet eens op de slides passen , maar als puntjes...

SPHINCS Signature