Signing Open Badges

OpenFortress*
digital signatures

Abstract:

This report reviews cryptographic practices for Signed Open Badges. The intent of this is to replace database-driven lookups of stored Open Badges by a form that can self-validate. This report was sponsored under the Edubadges & microcredentialing project of the Educational Innovation with IT program of SURFnet.

Document descriptive information:

Title:	Signing Open Badges
Author:	Dr.ir. Rick van Rein
Version:	1.0
Date:	October 15, 2018
ASN.1 oid:	N/A
Release:	Distribution as SURFnet sees fit
Legal status:	Informational, OpenFortress assumes no liability

Contents

Management Summary

During a proof of concept, SURFnet and its affiliated educational partners have concluded that Open Badges are not only a possible, but even a highly desirable concept for acknowledging (partial) educational results. The topic of this report is how to employ digital signatures to permit validation of these credentials without a need to access an institution's database.

The desired longevity for such Signed Open Badges is about 50 years. Under a *business as usual* assumption we could estimate key sizes suitable for such a period, but the expectations for quantum computers indicate that all currently used algorithms will be broken in 10 to 15 years. So not the keys, but the algorithms will need to adapt. Since no firm choices for alternative algorithms have been made yet, we see a need for re-signing badges to resolve this.

Instead of trying to avoid re-signing, this report emphasises verifiability of Open Badges even after the cracks of public key algorithms have arrived, so as to avoid a rush — and people being too late to renew their badges. We achieve this by adding content that is validating in an internal manner, and that can be used to accommodate automated re-signing of Open Badges. This is possible with the described use and interpretation of the Open Badge and JOSE specifications.

We also discuss the possibility of concealed field values in badges; this can be achieved by storing salted and hashed content in fields, which can be verified against voluntarily provided data by the party holding the badge and seeking approval with another party.

This report details the cryptographic structures and procedures for realising these facilities, in ways that SURFnet may incorporate in services and/or software.

Initial advice specific to a pilot is included in Section 7.

OpenFortress* specialises on crypgtographic aspects of computer systems and networks, with an aim to make security and privacy easily accessible to general users.

1 Predicting the Future

Once granted, an Open Badge should be valid for years to come. Specifically when adding digital signatures, the period over which this is true is a concern because cryptographic mechanisms loose their strength. Given that badges may be granted to someone at age 15 and last until they are about 65, we are facing Open Badge lifetimes of about 50 years.

The current-day practice is to aim for a 128-bit security level, basically meaning that an unstructured search space with 2^{128} values is so large that it cannot be practically iterated for a brute force attack by anyone. The numbers are so high that nobody, not even cryptographers, doubt this. Since computational power develops akin to Moore's law, we should consider larger search spaces for future practice. According to various estimations, we should aim for 256-bit security levels to be good for the coming 50 years; this matches the most conservative estimations, and the more liberal ones are not far off.

By way of relativation, it may be noted that an Open Badge is likely to be less important when it is 50 years old; its holder is likely to have developed additional skills and not rely on older ones just as heavily. But, as will be shown, the security level of 256 bits is easily achieved with current-day algorithms, so it feels like false economy to look for lower security levels.

A security level of 256 bits directly applies to symmetric encryption algorithms which are designed to resemble random permutations, so AES-256 would be a good choice; for hashes we should double the bits to protect against birthday attacks that reduce the search space to its square root, so SHA-512 would be a possibility; for public key algorithms the mathematical structure is subject to additional analysis and we should look for more specific advise.

As it turns out, we cannot rely on the most-used public key algorithms anymore, as a direct result of the rise of *quantum computers* (see Appendix A). This is no longer science fiction, mostly due to Shor's algorithm that lets a quantum computer factor integers in reasonable (polynomial) time. This is expected to affect all popular public-key algorithms in use today: RSA, DSA, DH, ECDSA, ECDH. We need to replace these algorithms with ones based on other hard-to-reverse-without-prior-knowledge computations, but no single mechanism has been selected yet; such algorithms do exist, but reaching consensus on the most likely ones is work in progress.

For long-lived signatures, it is crucial to understand that quantum computing is not an esoteric domain. The mechanisms have been established, small-scale computers are operational today. Generally, quantum computing has entered the phase of technical refinement, where it is a matter of grinding and polishing to get sharper results and larger bit sizes. And we all know how fast that can work; in this case particularly, there is sufficient funding to advance the area, in part because of their cryptographic challenge, and in part because quantum computers can implement associative searches.

With the earliest expectations of quantum computers that can crack today's algorithms to arrive in 10 years, and without consensus on replacing algorithms, the conclusion must be that re-signing of Open Badges cannot be reasonably avoided if their statements are to last for 50 years. What we can aim to avoid is that the occurrence of quantum computers renders all prior Open Badges untrustworthy; we can prepare for future re-signing by additional validation mechanisms, so that the introduction of quantum computers breaks the normal signature but leaves alternative methods of validation.

The basic choice to incorporate additional means of validation is also generally helpful when problems occur with the cryptographic mechanisms used; a single signature based on a mathematical structure would have to stand for 50 years without showing cracks, and this is a difficult guess to make with the limited experience that the field of cryptography has with this. This means that a design that incorporates re-signing is much more reliable in terms of algorithm stability than anything that predicts the future based on today's knowledge of one particular algorithm.

2 Cryptographic Mechanisms for Signed Open Badges

This section introduces a number of cryptographic mechanisms that may be used in an infrastructure for Signed Open Badges.

2.1 Certification

A certificate delivers a cryptographic bond between an identity and a public key. It is used to state that the public key is believed to belong to the stated identity, and thus that any authentications, signatures or decryptions can be trusted to be only achievable for the holder of that identity.

Trust in a signature implies the question *according to whom?* which is the main point of a certificate; the cryptographic bond between identity and public key is made by a digital signature by some party, and it is only meaningful when that party is considered a *trusted party* by anyone who wants to use the certificate to establish certainty about the identity of a public key owner (or, more accurately, of the owner of the adjoining private key).

This results in a public-key infrastructure. JOSE [RFC7165,RFC7520], including its signing branch JWS [RFC7515], works with keys but refrains from forming such an infrastructure [RFC7515,RFC7518], so it ends up using existing ones, most notably X.509 or OpenPGP.

2.2 Alternative Signatures

The most important mechanism to use is to have multiple signatures. This is possible with Open Badges, but the desired longevity of an Open Badge is

only realistically achievable when some support for re-signing exists, and/or validation based on alternative signature schemes.

Signatures are always applied to certain data portions. Alternative Signatures should process the exact same data, just through one or more alternative mechanisms. With most technologies, only a single signature would be supported and so the resulting alternative signatures would have to be inserted into an extension field. This would be technically complicating, but under the JOSE framework used for Signed Open Badges it is possible to add multiple signatures, provided that the JSON Web Signture is written in its JSON serialisation rather than the URL-safe compact serialisation.

2.3 Privacy of Certified Content

Some uses of Open Badges may call for privacy. Not all fields need to shout out their contents; instead it would be possible to store cryptographic codes that allow the validation of its contents at the same time that the Open Badge is being validated. And if the actual contents are not available, all the validation service would state is that *some value* for a given field exists, without showing it.

The content of a field can be easily concealed if it is stored as a salted hash. This means that not the actual contents are shown, but a random string (the salt) along with the outcome of hashing the salt with the concealed data. Normally, this means that the field content cannot be reconstructed, but when the holder of the Open Badge presents acclaimed contents for such a field, it is possible for the verifier to perform the same salted computation and verify that the outcome matches the hash outcome. The result is controllable release of information.

The purpose of the salt is to make each issued field different, and thereby, thwart rainbow tables that hold precomputed hashes of certain expected fields that can than be compared to large numbers of Open Badges. The longest salt that is useful has an amount of entropy (or 'surprise') that equals the bit size of the hash outcome. Given a proper random source, each random bit can be assumed to hold one bit of entropy.

Entropy may still pose a problem for the concealed content. For instance if it constitutes a grade in the range 1 to 10, one only needs to compute ten hashes to find the concealed content. In such cases, it may be desirable to incorporate more entropy in the content itself (such as a random string or personal description, or a string of digits behind a decimal dot, or combine a number of fields that are protected as one unit) and supply that to the validation process. When the entropy is of sufficient size, the procedure can replace a salt being stored alongside the hash outcome. Doing this is probably less practical than encrypting the field's contents with a (password-derived) key. This salt needs to be passed to validating parties, so it will look as unfriendly in the process as a key, but decryption can at least produce the field contents, rather than requiring these to be entered (and accurately matched) too.

Note that a re-signing procedure might be employed to add or remove a degree of protection and issue a modified Open Badge. This can be an automatic service because there is no change to the content, merely to its concealment. This can be a tool for Open Badge owners to produce variations at different levels of privacy; one with concealed fields to go online and another revealing all fields for direct communication. The validation that Open Badges are the same can be made, with the exception of fields that are concealed in both, following the same techniques as when field contents are explicitly shared.

2.4 Key Splitting

Cryptography hinges on the secrecy of keys. We generally speak of *secret keys* for use in symmetric algorithms such as AES and HMAC, or *private keys* to oppose the public keys used in asymmetric algorithms such as RSA and ECDSA.

To facilitate backup functions when faced with loss of a key, a term that is often used is key escrow. This is a rather blunt instrument that we should not have to use. Its main problem is that a backup site retains full control over one's tightly controlled secrets. The protection improves when such control can only be reconstructed when parties need to collaborate, for instance SURFnet, a notary-service lawyer and/or a backup university.

For such activities, *key splitting* is the solution. Think of a diagram $y = f(x)$ such that the key is the outcome of $f(0)$. In its simplest form, we can use $f(x) = a + b.x$ where a is the key. For b we choose a random number of as many bits as the key, and we usually make the computation modulo the maximum key value plus one. Then we choose two or more values (relative prime to the modulus) for x, and give each of our key sharing parties the pair $\langle x, f(x)\rangle$. One such party cannot derive a, but any two can. When 3 parties each hold a pair, we have 2-out-of-3 reconstruction possibilities. One party might be dismantled but the other two can still come together to reconstruct the key and perhaps divide it again with a new random b value, so the old reconstruction mechanism can no longer be combined with the new pairs.

A more general form of this mechanism is based on polynomials, where an *M-out-of-N* scheme requires a polynomial of degree $M - 1$ and N different pairs are supplied to as many parties. Alternatively, we could imagine a linear key splitting mechanism based on an affine space of $M - 1$ dimensions, for which M vectors in different directions would be required to span it, but as many as N vectors in different directions could be created for the key sharing parties. The key could be the length of a vector from origin to the affine space, either in a given direction different from all M vectors, or simply orthogonal to the affine space. The working principle for this method is that less than M vectors partially define the affine space, but in the defined dimensions it is like a fixed hanging point for a hinge and in any other dimension it is unclear what the rotation of that (multi-dimensional) hinge is.

Key splitting could even be applied recursively, allowing for example 2-out-of-3

for individuals within SURFnet. This is probably too much in practice.

Key splitting is ideally performed in isolation from the real world and detached from any network. Storage of parties' key fragments can be on any physical medium with suitable longevity, which can then be kept under a lock and chain.

3 Processing Open Badges

The procedures for Open Badge processing are at their core just signature creation and validation, with the possible addition or removal of partial concealment of field data. There are a few specific things worth noting, though.

3.1 Signing Open Badges

A signature on an Open Badge is like any other digital signature, except that we want to place multiple on the same data. This can be done with Open Badges, because they rely on JOSE for signing, also known as JWS.

JWS supports multiple signatures, each with its own headers specifying algorithm and key identity, but this is only possible in the JSON serialisation and not in the compact serialisation. Since Open Badges are described as JWS but the validation *example* describes the compact serialisation header.payload.signature as though it were the only serialisation form, it may come as a surprise to example-based coders that we need to use the general JSON serialisation for JWS and that the specification leaves us this room.

3.2 Validating Open Badges

The signing remarks for Open Badges also apply to validation of its multiple signatures. Specifically, the full JSON form must be recognised to accommodate this.

Instead of requiring a single signature to validate at full strength, we may consider adding the security levels of the various signatures, one at a time, until we reach the desired 256-bit security level that follows from the longevity of Open Badges. Such addition does require independent algorithms in the variants, so not RSA-SHA1 and ECDSA-SHA1, for instance. Also, composite algorithms such as RSA-SHA1 cannot give stronger levels than either of the independent algorithms SHA1 and RSA alone; it is common to match such algorithms for coinciding security strength but this balance shifts with the advent of quantum computing; a composition involving RSA will then fail as hard as RSA alone.

When we perform re-signing, we may introduce additional signatures without adding to the strength of a certificate. When the same hashing operation is used on re-signing, the warnings above already handle such situations. Removal of

a signature has no effect, because the certificate owner can easily add it back in, unless some information such as a serial number is changed on re-signing.

Some mechanisms have been abolished and should not be assigned any security level (or 0 bits), simply because no further white-hat research is done on them. The best known example of this is MD5. Others are in the process of being abondoned, like SHA1, but might be guesstimated to support the remaining-secure bits left (divided by two, since it is a hash facing birthday attacks). However, note that there will soon be no further research on SHA1, so it will someday be dropped and therefore is not useful for long-term validation. Had we built an infrastructure around SHA1, then the gradual decay that we usually see for cryptographic algorithms would have provided a window of opportunity for re-signing with a new algorithm.

Before, we suggested algorithms such as SHA-512 and AES-256 for a 50-year period of validity. The actual validation moment determines whether such choices have indeed been made correctly, because it is only at this future point in time that the actual state of algorithms can be incorporated. This means that a software base should update its validators when new insights are found on the algorithms used, and installations should make a point to stay up to date with such updates. Validators might also reveal a more refined response than black-or-white, but that easily gets end users involved in cryptographic certainties, which might not be proper.

Keep in mind that validation of symmetric mechanisms requires access to the same key material as used for signing. This applies to additional mechanisms such as AES256-GCM and HMAC-SHA512 because they are built on top of symmetric algorithms that are assumed to be safe from quantum computers.

3.3 Re-Signing Open Badges

The process of re-signing is a relatively straightforward combination of a validation step and a signing step. The process of achieving this can be automated. The vital concern is, of course, that validated state must not pass through a simple JavaScript variable or HTTP cookie, but instead the two operations should be atomically combined into one process to avoid releasing signatures on forged content.

Signatures may be taken from the old Open Badge and added to the new one to suggest a higher security level than intended, but that can be easily stopped with minor changes to the signed data; for instance, a serial number or might be replaced or a date of issuance might be added or replaced.

The same process as used for re-signing and Open Badge can also be helpful in protection of privacy, because personal data can be dropped immediately. This allows the holder of an Open Badge to keep one without concealed fields, and add protection when desired; this avoids problems with holders loosing their decryption keys.

4 Cryptographic Key Infrastructures

Cryptography employs algorithms that operate based on keys; the algorithms are public and secret or private keys are to be protected; the result tends to be as strong as the way that these keys are dissiminated. For this reason, we should discuss infrastructures for key handling.

4.1 Public Key Infrastructures

Digital signatures are often made by certificates, which themselves are signed by trusted parties. A chain of trust can be followed along such signatures. The public key infrastructure determines the structures of such certificates and the chain of trust. Independently of how certificates are structured, they need to be located, which could be through an IRI or through some form of a distributed hash table[1].

Certificates under X.509 are often informally called 'SSL certificates', because that is how most people see them first. But X.509 is also usable as a signing infrastructure. It is most commonly used in constrained commercial environments, a common example being email signing (and encryption). A salient point of X.509 certificates is that they appoint a single trusted party, and as a result they tell you what party or chain of parties to trust. This works well in closed trust networks, but even the common use in TLS (formerly known as SSL and still widely but incorrectly referenced by that name) has shown the need for evermore additional validation mechanisms: CRL, OCSP, DANE and lesser known others have all been devised to resolve problems inherent in the X.509 mechanism.

The mechanism used more for everyday signing and encryption (of email) is OpenPGP. This is different in a few ways. First, a public key can be bound to one or more identities, and each identity can be certified by one or more parties. Most facilities for OpenPGP are standardised after X.509, but the model is more flexible and, perhaps of vital importance to Open Badges, there is no single trust path pointing up to one central controller from which all trust emenates. This not only spreads the risk and responsibility over a larger number of certification parties, but it also means that there is less opportunity for lock-in by certification vendors. This may be why OpenPGP is less well-supported in commercial initiatives. Another possible reason is that validation under OpenPGP involves more freedom of choice and, as a result thereof, the need to think more about trust.

For the purposes of Open Badges, the attachment of an OpenPGP key under

[1] Note that the Open Badge specification document actively suggests that only HTTP/HTTPS might be supported in actual software, which means that this standard dramatically reduces the willingness of imlpementers to look further. This is a bit in line with the focus of the specification on examples that suggest only one anticipated use, but overlooks design alternatives that could be meaningful in other use cases. Such forms tend to be frowned up if it were to occur as an RFC.

one or more domain names signed by DNSSEC may be all that we are looking for, and this would accommodate Western universities as well as smaller-scaled and more loosely organised educational efforts. Within a single nation, trust may well be a straight chain of trust, at least for formal education, but when crossing borders or in support of more loosely organised educational forms, there is a need for mutual recognition among peers, which is only difficult to establish with X.509. It is generally good to be able to evaluate *cryptographic* trust, independently of trust in the *educational reputation* assigned to the established identity of a signing party.

The JOSE specifications, and notably the JSON Web Key specification, includes support for X.509 through a URL, inline certificate chain and certificate hashes; it does not yet define OpenPGP alternatives yet, but its JWK Parameters are registered at IANA with a relatively low bar for new identifiers set to 'specification required'. A proper form for that is a coherent Internet Draft, which anyone with sufficient knowledge can write and publish. So, as long as we define clearly what we think is best and certainly if we keep the general interest in mind, we should be able to satisfy the designated expert and add new identifiers matching our purposes.

4.2 Private Key Infrastructure

Private keys are paired to public keys under asymmetric algorithms such as RSA and ECDSA. The one to control the private key is the one to act as the party owning a (certified) public key. This is why the protection of private keys is paramount.

Consider the accidental loss of a private key. This means that no new signatures can be created that can be validated with the paired public key, nor can data encrypted to that public key be decrypted. Only the former of these problems can be addressed with a new key pair, whose public key is incorporated into a new certificate to state that the new public key represents the same identity. Both problems may be overcome if key splitting was used to make backups of the key.

Now consider that a private key gets stolen during a break-in on the machine that constructs Signed Open Badges. Now anyone can create signatures that can be validated with the public key. In other words, the identities certified for that public key can now be forged by the party taking ownership of the private key. This is much worse, and calls for the revocation of a certificate. This revocation implies a need to re-sign anything issued for it, or accept that previously made signatures have become invalid. In other words, it requires a database with private data of any Open Badge ever released. This is not always desirable. A mid-way solution that can overcome privacy concerns of such a database could be to only store secure hashes of the data, so as to recognise certifications whose re-siging is being requested but that still leaves concerns for robust and access-controlled data storage.

The most devastating thing that could happen is the undetected loss of a private key. This may be avoided almost completely by the use of a *hardware security module*, or HSM for short. This is a device with a standardised API called PKCS #11. The API is used (or more accurately, can be used and should be used) such that private keys are generated and stored behind the API, but can never be extracted. Especially when the actual private key is stored in separate hardware, usually constructed as a security bastion, may theft of the private key be considered a managed problem as it can avoid accidental and undetected retrieval.

Solutions exist that replicate HSM contents without exporting any key material; SURFnet actually uses such a facility for its DNSSEC setup.

4.3 Secret Key Management

The management of secret keys is mostly the same as that of private keys; the major difference is that the algorithms used are symmetric, so that the reverse algorithm uses the same keys. So, any validator should be aware of the same keys that were used to construct signatures. The result of this is that only known-good parties can validate such signatures. For most alogirhtms, secret keys can be generated as a random byte string with sufficient entropy, that is, sufficient 'surprising' bits to attain the desired security level.

Given that the same value is used for validation as for signing, it is also necessary to protect the keys in the same manner; when PKCS #11 is used to protect a secret key during signing, then PKCS #11 must also be used for validation. It would be quite awkward if validation of PKCS #11 protected signatures would be done with the same key built into a PHP script!

An extra option for secret keys is that they might be derived from a central master key, by mixing it with different salts through a hashing scheme. Such salts could safely be published. This might be used to, for example, generate a new secret key per batch of Open Badges that depended on secret keys. Care must be taken that derived keys cannot be extracted from the protective PKCS #11 interface, and this is not trivial with key derivation schemes.

5 Selection of Alternative Validators

Choosing a signature algorithm for current-day use is easy; one might go with RSA, ECDSA or both. But to overcome the predictable future hit by quantum computers (see Appendix A), we should add extra validation algorithms to Open Badges. These will provide a base of trust to a future re-signing service, so that it can produce a new primary signature on the Open Badge without the need to rely on material that may than have been cracked.

5.1 Database Records

A technically simple approach is to store every Open Badge in a database. Such long-term storage of education records is not the most desired option for two reasons:

1. The storage requirements may outlive the institution who granted them, or the Open Badge technology may be replaced or updated and it may not be desirable to maintain such long-term backward compatibility;
2. The privacy of the certified individuals may be at stake, and privacy laws may even require erasing their data after a certain data.

It is not likely that this option will be chosen; the whole idea of using Signed Open Badges is to be able to distribute the knowledge of educational achievements from the accrediting institutions to the accredited individuals.

5.2 HMAC Signatures

The HMAC scheme builds on any given hash algorithm, and uses a secret key to derive an authentication code over a bit of data. One of its uses is as a key-based checksum, but it may also be used as a symmetric signature.

HMAC employs just a hash algorithm, and these are considered protected against the (mathematical) facilitation by quantum computers. In fact, HMAC obfuscates a hash so much more than the original hash that even the long-discarded MD5 hash may be secure to use in HMAC-MD5. We tend to avoid this to avoid confusion with algorithm users, but it is an indication of the added strength of HMAC.

On the other hand, HMAC may 'leak' tiny amounts of information about the secret key with every signature made. It is for that reason, as well as limitation of signature overhead, that HMAC signatures are usually cut a bit shorter than the signatures. We tend to generate fewer Open Badges than a protocol generates network packets, so we may be fine if we regularly rotate our keys.

Key rotation could be done per batch, or per period, of dissimination of HMAC signatures. The salt used in each rotation might be published so that parties holding key fragments could reproduce the series of keys as soon as they had reconstructed the secret key. In JOSE, the keys follow the JWK specification that includes a kid field holding a locally formatted key identity. This field can be made to hold a reference or identity for the master key along with the salt used for the batch at hand.

The validation operation requires the same HMAC computation as for signing, which is why this is called a symmetric signing algorithm. When a key rotation scheme is used, then the validation should follow the same rotation of keys to construct the secret key for validation of the batch or period at hand. It might look in the Open Badge for data that helps to select the key to derive.

Based on our desire for a 256-bit security level, we might consider an algorithm such as HMAC-SHA512. Recall that hash lenghts are always double that of the security level, to accommodate for the birthday attack.

5.3 Encryption

Another option that is expected to be protected from quantum computer attacks is symmetric encryption. What might be done here is the encryption of the data, and during valudation either allow its decryption or simply re-compute the encryption and check for the same outcome.

The latter scheme allows for reduced forms of stored data, such as the last block from a chained sequence of blocks, especially when this is an authenticating last block. As a better alternative, modern mechanisms for authenticated encryption can incorporate into the encrypted flow any amount of so-called *associated data*, which is plaintext data that is usually provided somewhere in the encryption context. This data would not expand the encrypted data but still impact the verification code at the end of the flow.

Based on our desire for a 256-bit security level, we might consider an algorithm such as AES-256. Though it shares a 128-bit block length with the other AES variants, it does use 256-bit keys to aid in the perturbation of these blocks of bits. A good choice for the reduced form would be to employ AES-GCM, where the same data as signed with the primary signature as associated data. There would be no use of a standardised secure hash for this construct, but instead a specific GMAC algorithm is used for AES-GCM. Without algorithm overlap, the opportunity of an independent validation path improves.

5.4 Post-Quantum Public-Key Algorithms

There are various candidate public-key algorithms about which consensus may be reached as replacements for our current uses of algorithms like RSA and ECDSA. With the formation of consensus still under consideration, we cannot predict what will happen.

Any of these mechanisms can still be useful as alternative signatures. They may not validate on current Open Badge software because the algorithms widely implemented yet, but at least locally we can use these algorithms as a secondary validation option. The advantages of this approach is that even external parties could validate the secondary signatures based on a trusted public key, so in the most lucky scenario we might not even have to resort to a re-signing service. But in the least lucky scenario we have chosen an algorithm that others choose to ignore, which would mean that only our re-signing services validate the alternative signatures, much like they do for symmetric signatures. This, combined with the properties such as storage size and updates to private keys may limit their usefulness as alternative signatures for Open Badges.

McEliece. This algorithm was suggested from government sources. They use it for a more limited purpose, namely for authentication during a key agreement process. Such authentication signatures can be made at the current desired security level of 128 bits, unlike our long-lived signatures that require a 256 bit security level. Already at the 128 bit level, McEliece needs to use blobs of 512 kB. This adds so much extra weight to an Open Badge that it appears to be unpractical.

SPHINCS. The SPHINCS scheme delivers a 128-bit security level designed for the long term. This level should be sufficient for the period until quantum computers, but may not cover our intended period of 50 years. SPHINCS signatures are 41 kB in size, which may be doable for Open Badges. A useful property is that the signing process does not alter the state of the private key, but in return it must be painstakingly avoided that the same key ever signs two (different) data blocks. SPHINCS includes solutions to that end.

41 kB signatures, stateless (so fixed private keys).

XMSS. The XMSS scheme has been standardised in RFC 8391 and is attracting public attention and promotion. The signatures are small, but every signature made updates a state that impacts the private key, so as to avoid using the same key twice. Note that such updates to private keys are likely to complicate operational concerns such as HSM replication.

6 Extending Open Badges

The suggestions made in this report rely on a few extensions to the Open Badges specification. The standard is open to such extensions, and it is probably a good idea to publish them so others can benefit too.

6.1 Critical and Non-Critical Extensions

The current Open Badges specification does not distinguish between extensions that are critical or non-critical. This is a practice that would be good to replicate from X.509 certificates.

Critical extensions should be understood by any validating party, and incorporated into its judgement. Non-Critical extensions may in contrast be ignored when they are not recognised. The purpose of this is to distinguish informative fields that can simply be suppressed in validation output from instructions that impact the validation process directly. With critical extensions it is possible to further constrain what makes up a valid Open Badge; without it, validators have less clarity on what to implement for each and every extension.

6.2 Alternative Signature

The current Open Badges specification suggests through its example that a single signature on each Open Badge may be the only form. As explained when looking ahead in cryptography over a period of 50 years, we anticipate that this causes problems with the future validity of Open Badges.

Under JOSE and JWS, it is possible to sign more than once. This does assume that we literally read the SignedBadge Verification example in the specification as just an example, even if the wording 'A JWS has three components separated by dots' makes it sound as though signatures under the incorporated JWS specification would always be written in the compact serialisation. To be able to support multiple signatures, the JSON serialisation is required. Ideally, the text of the 2.0 proposed standard should reflect that.

This means that alternative signatures do not require an extension to the format. At most is there a need to add any extra algorithms to be used. Note however that IANA already specifies several that are symmetric in nature, notably including *HMAC using SHA-512* and *AES GCM using 256-bit keys*. Others that we did not define should be added by IANA as long as we provide a specification that passes the designated expert, since that is what the policy for this registry prescribes.

Since this is not an extension as intended by the OpenBadge framework, it is no question whether it is critical or non-critical.

Signatures made on concealed field content should be made on the protected form of the fields, leaving it to the field itself to securely bind into the authorisation that can be derived from the signature. The idea is that uncovering of the concealed field content should not be required for validation of an Open Badge. Furthermore, if the signature would have been placed on the plaintext version of these fields, it would turn the Open Badge into an oracle: a data structure to be taken offline at which arbitrarily many questions may be posed about the possible values of the concealed data.

7 Advice for Signed Open Badges

- Select 2 or 3 additional signature mechanisms, and use multiple signatures as supported by JWS. Be verbose towards the Open Badges community about the required use of JSON serialisation for JWS, rather than the URL-safe compact serialisation exemplified in the Open Badge specification. *For a pilot, we depend on existing software, so I recommend additional signatures based on HMAC-SHA512 and on AEAD in AES256-GCM.*

- Plan a re-signing service, and develop software for it. This would be an automated service, but it may need access to secret keys for the validation of the additional signatures. The service can be offered as soon as con-

sensus on a post-quantum public-key mechanism is a fact, and becomes required when validators begin to drop current signature mechanisms like RSA and ECDSA. *For a pilot, I recommend an automated combination flow that starts with a validator and continues to a release of a freshly Signed Open Badge with the same data.*

- Decide on the protection mechanisms for secret and private keys; where possible, PKCS #11 is preferred. Setting up a true HSM is non-trivial, so prepare well. *For a pilot, I recommend SoftHSMv2. Extra points for using a Remote PKCS #11 mechanism to separate out the process. A party already familiar with HSM operational aspects might choose to use that solution instead, but for anyone else the burden would be too great for a pilot. It would be false security to import SoftHSMv2 keys into a hardware HSM at a later time.*

- Decide on the desired sharing of secret and private keys; a minimum choice would be 2-out-of-3, where a party can stop being available without risk to the key material and their uses. *For a pilot, I recommend gaining experience with a 2-out-of-3 scheme. This can be implemented with a straightforward key splitting scheme based on a straight line. I recommend documenting and testing a procedure for key loss or, equivalently, the disappearance of one of the key sharing parties.*

- Document a procedure for the generation of private and secret keys, and sharing the key material produced among parties to facilitate backup. *For a pilot, I recommend parties to get together, detach from networks, generate a key and split it, encrypt fragments with their holder's password or PGP key and storing the result on USB devices to be securely stored by the respective fragment owners. Don't forget to import the key into the PKCS #11 solution. With a higher-end HSM, expect to find tooling in the operational flow for the device.*

A Quantum Computers

This report references quantum computers. But what are these?

Quantum Computing is an emerging field of technology. It is vital to understand that it is no longer purely fundamental research, but rather technology being developed. The field has already shown the principles to work, and is working towards clearly set goals and has a good understanding of what aspects should be improved upon. One might say that a research road map exists. And like in other fields of technology, the numbers are going up rapidly and, with that, the power to do the inevitable to present-day cryptography.

To break an algorithm that is designed to be particularly difficult to crack, you almost need to change the rules of the game, and that is exactly what Quantum Computers do; they conduct a lot of computations in the same physical port, and given smart-enough algorithms, they can do surprising things.

To take a bit away from the mystique surrounding the field, think of a string on a guitar. When it is plucked, it imposes a diverse and random motion on the string, but the resonance of the string quickly dampens all frequencies except those that fit in the length of the string. The result is a sharp beginning full of unstructured noise and, after a while, a bright and harmonious tone. The selection process on each of the many initial frequencies happens completely independently in the same string. Now imagine passing some frequencies through to other strings, which also start vibrating and providing feedback on the resonance. And imagine building structures or perhaps even algorithms that could be built with this; some string parts would end up vibrating on one tone, others on another tone, and yet other frequencies from the original pluck are lost. This is not exactly how a Quantum Computer works, but it gives a reasonable impression.

Algorithms for these special mechanics have been devised to approach what we up to now thought uncrackable in practice in a time that is structurally faster. The unexpected edge that Quantum Computers provide stem from their ability to operate on multiple possible data values at the same time, and their ability to make these data values interact in ways that construct computations.

A.1 What gets broken?

In ten years from now, it is expected that all current public-key algorithms are broken. RSA, DSA, ECDSA, DH, ECDH, all our favourites. This is nothing short of devastating because the current Internet leans heavily on these algorithms to build trust with parties that were not previously introduced — based on their public-key-verifiable proof of ownership of a private key. Most notably, this includes HTTPS under any of the current forms of TLS. The game changes entirely when others can suddenly start claiming posession of the private key, or when we must assume that adverse third parties might be due to a crack made with a Quantum Computer.

There are some pieces that are expected to withstand a Quantum Computer though. We can continue to use symmetric (or 'bulk') ciphers such as AES, and we can still use hashing algorithms. Note that this means that Kerberos, on which we found most of our infrastructure, will continue to be safe, as long as it is not founded on the public-key setup protocol PKINIT (also known as *smart card logon*).

Work has started on devising new public-key crypto systems, and we should expect these to be commonplace when Quantum Computers hit the market. So, are we out of trouble? Not by a long shot, we do need to act well before that time. As in, now is the time to move.

A.2 What does this mean for Open Badges?

The Internet relies on public key cryptography for a number of things:

1. Signatures for authentication, such as in web traffic
2. Key encryption and key agreement, with or without Forward Secrecy
3. Signatures for legal authorisation, such as with Open Badges

All these uses will be broken with the current algorithms. But their impact varies.

Authentication is usually a thing at the present time; there is no damage from future cracks of an authentication key because we have no mechanism to jump back in time and perform false authentication. The other two use cases however, need to look into Quantum Computing for longevity of their security level.

Key encryption and key agreement are ways of establishing a 'bulk' encryption key for documents or network connections, including HTTPS. It is worryingly doable to capture and store all such traffic for future analysis. At this later moment, everything that was thought to be private traffic can be decrypted and looked through.

Legal authorisation is another big problem, and it is central to what Signed Open Badges are meant to do; looking at any legally binding signature made with current-day technology after the arrival of Quantum Computers is meaningless, as anyone could have forged the signature. This implies a requirement to refresh all signatures before that time; or, with the suggestions made in this document, to already have alternative signatures alongside the ones that can currently be validated.

Notes:

info@openfortress.nl

http://openfortress.nl

OpenFortress*
digital signatures