Use of the HSS/LMS Hash-based Signature Algorithm with CBOR Object Signing and Encryption (COSE)
draft-ietf-cose-hash-sig-09

Abstract

This document specifies the conventions for using the Hierarchical Signature System (HSS) / Leighton-Micali Signature (LMS) hash-based signature algorithm with the CBOR Object Signing and Encryption (COSE) syntax. The HSS/LMS algorithm is one form of hash-based digital signature; it is described in RFC 8554.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 13, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

This document specifies the conventions for using the Hierarchical Signature System (HSS) / Leighton-Micali Signature (LMS) hash-based signature algorithm with the CBOR Object Signing and Encryption (COSE) [RFC8152] syntax. The LMS system provides a one-time digital signature that is a variant of Merkle Tree Signatures (MTS). The HSS is built on top of the LMS system to efficiently scale for a larger numbers of signatures. The HSS/LMS algorithm is one form of hash-based digital signature, and it is described in [HASHSIG]. The HSS/LMS signature algorithm can only be used for a fixed number of signing operations. The number of signing operations depends upon the size of the tree. The HSS/LMS signature algorithm uses small public keys, and it has low computational cost; however, the signatures are quite large. The HSS/LMS private key can be very small when the signer is willing to perform additional computation at signing time; alternatively, the private key can consume additional memory and provide a faster signing time. The HSS/LMS signatures [HASHSIG] are currently defined to use exclusively SHA-256 [SHS].
1.1. Motivation

Recent advances in cryptanalysis [BH2013] and progress in the development of quantum computers [NAS2019] pose a threat to widely deployed digital signature algorithms. As a result, there is a need to prepare for a day that cryptosystems such as RSA and DSA that depend on discrete logarithm and factoring cannot be depended upon.

If large-scale quantum computers are ever built, these computers will have more than a trivial number of quantum bits (qubits) and they will be able to break many of the public-key cryptosystems currently in use. A post-quantum cryptosystem [PQC] is a system that is secure against such large-scale quantum computers. It is open to conjecture when it will be feasible to build such computers; however, RSA [RFC8017], DSA [DSS], ECDSA [DSS], and EdDSA [RFC8032] are all vulnerable if large-scale quantum computers come to pass.

Since the HSS/LMS signature algorithm does not depend on the difficulty of discrete logarithm or factoring, the HSS/LMS signature algorithm is considered to be post-quantum secure. The use of HSS/LMS hash-based signatures to protect software update distribution will allow the deployment of future software that implements new cryptosystems. By deploying HSS/LMS today, authentication and integrity protection of the future software can be provided, even if advances break current digital signature mechanisms.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. LMS Digital Signature Algorithm Overview

This specification makes use of the hash-based signature algorithm specified in [HASHSIG], which is the Leighton and Micali adaptation [LM] of the original Lamport-Diffie-Winternitz-Merkle one-time signature system [M1979][M1987][M1989a][M1989b].

The hash-based signature algorithm has three major components:
As implied by the name, the hash-based signature algorithm depends on a collision-resistant hash function. The hash-based signature algorithm specified in [HASHSIG] currently makes use of the SHA-256 one-way hash function [SHS], but it also establishes an IANA registry to permit the registration of additional one-way hash functions in the future.

2.1. Hierarchical Signature System (HSS)

The hash-based signature algorithm specified in [HASHSIG] uses a hierarchy of trees. The Hierarchical N-time Signature System (HSS) allows subordinate trees to be generated when needed by the signer. Otherwise, generation of the entire tree might take weeks or longer.

An HSS signature as specified in [HASHSIG] carries the number of signed public keys (Nspk), followed by that number of signed public keys, followed by the LMS signature as described in Section 2.2. The public key for the top-most LMS tree is the public key of the HSS system. The LMS private key in the parent tree signs the LMS public key in the child tree, and the LMS private key in the bottom-most tree signs the actual message. The signature over the public key and the signature over the actual message are LMS signatures as described in Section 2.2.

The elements of the HSS signature value for a stand-alone tree (a top tree with no children) can be summarized as:

```c
u32str(0) ||
lms_signature /* signature of message */
```

where, the notation comes from [HASHSIG].

The elements of the HSS signature value for a tree with Nspk signed public keys can be summarized as:
where, as defined in Section 3.3 of [HASHSIG], a signed_public_key is the lms_signature over the public key followed by the public key itself. Note that Nspk is the number of levels in the hierarchy of trees minus 1.

2.2. Leighton-Micali Signature (LMS)

Subordinate LMS trees are placed in the the HSS structure discussed in Section 2.1. Each tree in the hash-based signature algorithm specified in [HASHSIG] uses the Leighton-Micali Signature (LMS) system. LMS systems have two parameters. The first parameter is the height of the tree, h, which is the number of levels in the tree minus one. The [HASHSIG] includes support for five values of this parameter: h=5; h=10; h=15; h=20; and h=25. Note that there are 2^h leaves in the tree. The second parameter is the number of bytes output by the hash function, m, which is the amount of data associated with each node in the tree. The [HASHSIG] specification supports only SHA-256, with m=32. An IANA registry is defined so that other hash functions could be used in the future.

The [HASHSIG] specification supports five tree sizes:

LMS_SHA256_M32_H5;
LMS_SHA256_M32_H10;
LMS_SHA256_M32_H15;
LMS_SHA256_M32_H20; and
LMS_SHA256_M32_H25.

The [HASHSIG] specification establishes an IANA registry to permit the registration of additional hash functions and additional tree sizes in the future.

The [HASHSIG] specification defines the value I as the private key identifier, and the same I value is used for all computations with the same LMS tree. The value I is also available in the public key. In addition, the [HASHSIG] specification defines the value T[i] as the m-byte string associated with the ith node in the LMS tree, where the nodes are indexed from 1 to 2^(h+1)-1. Thus, T[i] is the m-byte string associated with the root of the LMS tree.
The LMS public key can be summarized as:

\[
u32str(lms_algorithm_type) \ || \ u32str(otstype) \ || \ I \ || \ T[1]
\]

As specified in [HASHSIG], the LMS signature consists of four elements: the number of the leaf associated with the LM-OTS signature, an LM-OTS signature as described in Section 2.3, a typecode indicating the particular LMS algorithm, and an array of values that is associated with the path through the tree from the leaf associated with the LM-OTS signature to the root. The array of values contains the siblings of the nodes on the path from the leaf to the root but does not contain the nodes on the path itself. The array for a tree with height h will have h values. The first value is the sibling of the leaf, the next value is the sibling of the parent of the leaf, and so on up the path to the root.

The four elements of the LMS signature value can be summarized as:

\[
u32str(q) \ || \ \text{ots\_signature} \ || \ u32str(type) \ || \ \text{path}[0] \ || \ \text{path}[1] \ || \ ... \ || \ \text{path}[h-1]
\]

2.3. Leighton-Micali One-time Signature Algorithm (LM-OTS)

The hash-based signature algorithm depends on a one-time signature method. This specification makes use of the Leighton-Micali One-time Signature Algorithm (LM-OTS) [HASHSIG]. An LM-OTS has five parameters:

- **n** - The number of bytes output by the hash function. For SHA-256 [SHS], n=32.

- **H** - A preimage-resistant hash function that accepts byte strings of any length, and returns an n-byte string.

- **w** - The width in bits of the Winternitz coefficients. [HASHSIG] supports four values for this parameter: w=1; w=2; w=4; and w=8.

- **p** - The number of n-byte string elements that make up the LM-OTS signature.

- **ls** - The number of left-shift bits used in the checksum function, which is defined in Section 4.5 of [HASHSIG].

The values of p and ls are dependent on the choices of the parameters n and w, as described in Appendix B of [HASHSIG].
The [HASHSIG] specification supports four LM-OTS variants:

- LMOTS_SHA256_N32_W1;
- LMOTS_SHA256_N32_W2;
- LMOTS_SHA256_N32_W4; and
- LMOTS_SHA256_N32_W8.

The [HASHSIG] specification establishes an IANA registry to permit the registration of additional hash functions and additional parameter sets in the future.

Signing involves the generation of C, which is an n-byte random value.

The LM-OTS signature value can be summarized as the identifier of the LM-OTS variant, the random value, and a sequence of hash values (y[0] through y[p-1]) as described in Section 4.5 of [HASHSIG]:

\[ \text{u32str(otstype) || C || y[0] || ... || y[p-1]} \]

### 3. Hash-based Signature Algorithm Identifiers

The CBOR Object Signing and Encryption (COSE) [RFC8152] supports two signature algorithm schemes. This specification makes use of the signature with appendix scheme for hash-based signatures.

The signature value is a large byte string as described in Section 2. The byte string is designed for easy parsing. The HSS, LMS, and LMOTS components of the signature value format include counters and type codes that indirectly provide all of the information that is needed to parse the byte string during signature validation.

When using a COSE key for this algorithm, the following checks are made:
The 'kty' field MUST be 'HSS-LMS'.

If the 'alg' field is present, it MUST be 'HSS-LMS'.

If the 'key_ops' field is present, it MUST include 'sign' when creating a hash-based signature.

If the 'key_ops' field is present, it MUST include 'verify' when verifying a hash-based signature.

If the 'kid' field is present, it MAY be used to identify the top of the HSS tree. In [HASHSIG], this identifier is called 'I', and it is the 16-byte identifier of the LMS public key for the tree.

4. Security Considerations

The Security considerations from [RFC8152] and [HASHSIG] are relevant to implementations of this specification.

There are a number of security considerations that need to be taken into account by implementers of this specification.

Implementations MUST protect the private keys. Compromise of the private keys may result in the ability to forge signatures. Along with the private key, the implementation MUST keep track of which leaf nodes in the tree have been used. Loss of integrity of this tracking data can cause a one-time key to be used more than once. As a result, when a private key and the tracking data are stored on non-volatile media or stored in a virtual machine environment, failed writes, virtual machine snapshotting or cloning, and other operational concerns must be considered to ensure confidentiality and integrity.

When generating an LMS key pair, an implementation MUST generate each key pair independently of all other key pairs in the HSS tree.

An implementation MUST ensure that a LM-OTS private key is used to generate a signature only one time, and ensure that it cannot be used for any other purpose.

The generation of private keys relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate these values can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality
random numbers is difficult, and [RFC4086] offers important guidance in this area.

The generation of hash-based signatures also depends on random numbers. While the consequences of an inadequate pseudo-random number generator (PRNG) to generate these values is much less severe than in the generation of private keys, the guidance in [RFC4086] remains important.

5. Operational Considerations

The public key for the hash-based signature is the key at the root of Hierarchical Signature System (HSS). In the absence of a public key infrastructure [RFC5280], this public key is a trust anchor, and the number of signatures that can be generated is bounded by the size of the overall HSS set of trees. When all of the LM-OTS signatures have been used to produce a signature, then the establishment of a new trust anchor is required.

To ensure that none of tree nodes are used to generate more than one signature, the signer maintains state across different invocations of the signing algorithm. Section 12.2 of [HASHSIG] offers some practical implementation approaches around this statefulness. In some of these approaches, nodes are sacrificed to ensure that none are used more than once. As a result, the total number of signatures that can be generated might be less than the overall HSS set of trees.

A COSE Key Type Parameter for encoding the HSS/LMS private key and the state about which tree nodes have been used is deliberately not defined. It was not defined to avoid creating the ability to save the private key and state, generate one or more signatures, and then restore the private key and state. Such a restoration operation provides disastrous opportunities for tree node reuse.

6. IANA Considerations

IANA is requested to add entries for hash-based signatures in the "COSE Algorithms" registry and hash-based public keys in the "COSE Key Types" registry.

6.1. COSE Algorithms Registry Entry

The new entry in the "COSE Algorithms" registry [IANA] has the following columns:
Name:  HSS-LMS

Value:  TBD1 (Value between -256 and 255 to be assigned by IANA, with a preference for -46)

Description:  HSS/LMS hash-based digital signature

Reference:  This document (Number to be assigned by RFC Editor)

Recommended:  Yes

6.2. COSE Key Types Registry Entry

The new entry in the "COSE Key Types" registry [IANA] has the following columns:

Name:  HSS-LMS

Value:  TBD2 (Value to be assigned by IANA)

Description:  Public key for HSS/LMS hash-based digital signature

Reference:  This document (Number to be assigned by RFC Editor)

6.3. COSE Key Type Parameters Registry Entry

The new entry in the "COSE Key Type Parameters" registry [IANA] has the following columns:

Key Type:  TBD2  (Value to be assigned above by IANA)

Name:  pub

Label:  TBD3 (Value to be assigned by IANA)

CBOR Type:  bstr

Description:  Public key for HSS/LMS hash-based digital signature

Reference:  This document (Number to be assigned by RFC Editor)

7. References

7.1. Normative References

7.2. Informative References


[DSS] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", FIPS Publication 186-6, 2013.


Appendix A. Examples

This appendix provides a non-normative example of a COSE full message signature and an example of a COSE_Sign1 message. This section is formatted according to the extended CBOR diagnostic format defined by [RFC8610].

The programs that were used to generate the examples can be found at https://github.com/cose-wg/Examples.
A.1. Example COSE Full Message Signature

This section provides an example of a COSE full message signature.

Size of binary file is 2560 bytes.

{{ RFC Editor: This example assumes that -46 will be assigned for the HSS-LMS algorithm. If another value is assigned, then the example needs to be regenerated. }}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}

98{
    [ 
        / protected / h'a10300' / {
            \ content type \ 3:0
        } ,
        / payload / 'This is the content.',
        / signatures / [ 
            [ 
                / protected / h'a101382d' / {
                    \ alg \ 1:-46 \ HSS-LMS
                },
                / unprotected / {
                    / kid / 4:'ItsBig'
                }
            ]
        }
    ]
}
400f046191136c367038d6a9d0e0ae30ddc4737312cb5a2e35315eff5c1a7e08
5b68c5cf0c64c495df2ca6f030db04480a2ed4a04d0d62f929d463db9e41e346e49
c89d4de4339c834c4746309c886d6131f2f92155ca1160bac9610820a497b5a9a49
b35357d13df0df022a9e6be5f69129f68a5ed3a602146f6d00c4d9df0af9e0b9f961dc
9f727251d46899c28d87080ba2ead3e8193f51a789706ec32aaeeef94fb1eeca9a125
2fe89ab9ed3cc93d389aabbbe7d217948cae79c3e43bd7d7df67f50390ff95243ed36225b2
ac2ab57503c9e07fcf3d2cc32fa9a19797ece05c5ce5f0f59d5f1560f52d33f93166
af68a90293f5ad273a4ac4b3f05c05420bfa6262a7d52d2dafa63f7e2d2c005
f1ace0aa3cb62634ee341939361207b685006137a59e2f5cd5389265c4cf9ef1e177
02c791752e4f94676955bcbcf04dc4299a8a8827b9b95cb826e25ab6a9ebcf9bb60c91f2b06
oc69c111693eff242f241c51cb885541634b8a2142d3c9bada51d6ef3e08a486b74
3f2492dd9ad0af84f3ce3b08a5879a4762a7569b41f4bf8dfb71481303ba0d4a913
1882455ece23e503c253c9e2a17faeacf8e393ec2
6ebe08a5dbdf6314a38929799fb7feeebee202b9705e6ed17e4f3881e68930314d7
7e9c629c46df0db9257c5f5243f159d964961745cd46579f0d69694971c49cbbaf2af
8979a2c5ec8767619c7a7be516e66c941087441a81f119a838353bb24dcd725a819a1d
f662da2804c88d46c83e86205574282e4af23eam648fca79767af0b989f8d164567761
f539f3bcb762c93f17f6465cc0eef929b2b05ced7d76143c63dc7018a1b30c108e9a
01be32b9d91b6da13a15283c2a9694c899a772fe8fe0017eceb34e37377d72cba
06c5f5dacc9ad3df7e03cf196595a8c1d8b09302db13b3d4371239da988b67a176
6a3792452df716e91ab0af1e5275c9c1babca7b1a600ae6957f9b1b05e49e2
d181874ac6dd160dc717b73bd28ef5508d474665d5aef7f54814c7e206fa9e2e2c533
85d4d52f7679d95e05024ff202d7275b04d71d919e73bcb6e84f1fc05a1e78a1
1fd967906452562545bd1713c3cdd97661e9595ce47d6be3ead96ee6d006a5ce6f497
77fe2e3f91bebe877cacc8c648dfced0131dc71b9938759b981c5f5fe211de8b80
af4280e9e3d1717e3645b410a3cb51838ce3cd4bda1e355988061656668996d9a
8e3e56b4b132908306da0d5526c4a5e2591612a47b7e86550ade12807d1e651d44
96f9900a0a2979e42bcbccf32e3b1e0bf9f9aa4d352e0e0b3382242df2dced4e90
9c49f1a359e92ef971dc6d6b06047a333c2ebe827eb65df2811f0ebeb0f12f0b2094e
00dc8e418f3691a60cbe0cef6f45f47838b36db9205090e916267406dbfadb6e3bf
5e600a66580bdc839b5be4ee46294537a8788e86cc5e6150675db1a918d18eea6e
fe7d88ff57bf39b8610392811e097e091a61c481e0fbd3463edf65a64e24128cd9f13
02d2f2e7fdaaa8e0face73766c8ffef64f4699b5b5f4f3cd59c70fecedb60a25e82850417
157f4f3f3e72ca60f509641cd0f9d9a78130e1f7ba83333502ad40d3f64f41a43d0
35e2ac0edc3469a9ac504519c5c3fe6e567576896b602d46a910aad3ca400
24e3beb3cf32732d6e2886ad724dd56ea2858e4b60beec92d545d5d700387787b74552f
aef18f75497b20614161343b390f628522853265b1f7aeaf231673ad5a5e5e5edadd80
b0e999b26bc56b0de34f2b2216245676c28e5e7c834d3bcbd6b7db84552522f2
2e9744f6f3d7a16774898f6e9ab9092b2982c97db2d36349828c8f30170618
ecf14f41d8082897e338bf7fa837f674be5243f026c4af83d0154f452baf939b8
143a3a98f78fb8c981e00dbdbb7f07b712c45784f4d1c421465583ad5f5bce5f15edd60
067e0916086e532a770c7be366ee0c7e780f81892b00000000ed1e8c6e437
918d43fba7b953869541c4b0127303f6f7bf704edeedd9384ab6f8cb362c948646b3c984
8038e69d9a1f7d9367f709cdd355cd7063056fc03680900491bc4ebcbabec128
7c81a46e62a7b57640a818be1bcff7d9d9419a10dcd8686d16621a80816f5b5bc5
621172ca70b811f117d129529a7570cf79cfc52a7028a48538ecdd33b38d3d562d26
246595ce4fb73a525a5ed3c03524ebebdc8ce20c9bc4977c6898ff95fd3d310b0ba
e71696cef93ca65254656f9e69075383bb7543c82abfbcf7cddb8483b3276c
A.2. Example COSE_Sign1 Message

This section provides an example of a COSE_Sign1 message.

Size of binary file is 2552 bytes.

```plaintext
18(
   [  
      / protected / h’a101382d’ / { 
         / alg / 1:-46 \ HSS-LMS \  
      } / ,  
      / unprotected / { 
         / kid / 4:’ItsBig’  
      },  
      / payload / ’This is the content.’,  
      / signature / h’00000000000000000000000391291de76ce6e24d1e2a9b60 266519bc8ce889f814deb0fc00edd3129de3ab9ba5ac783bdf0fe689f57fb204 f1992db1ce2484f316c74be3f2094cafe8e96a4a9548cead0f78ee5d549510d1910 f647320448ae27ecee77249802a0c39c645bf8db08573af52c9391fd0e217f245c7 52c176b1514eb6e3067e0fb329225eaa88c7d21635e32ae84213f98018cb06f1b8 4e61ee4348b690d7c6265c19f9d868952d9982aecd417b5279dd674cd951c306016 cfe4fee3bfc5e5a5ad08b5bf453bc93995f26cfe7c0c1c5ba2574fc1f2d8470993 e8bd47ef9bcf309ef895226e92be60683459009611defbb9a43217956a0ab295bb da0fe4de39ed37e7c4a6cd8a5314d6b02b377406d5a5e589e91f9ea9f24ec1682ba1 f633c7784499323e40da651f7d3c19e38e3d4898b0c5038240bfefc7c5f00a8c014 b4af000a739f96cdadb94daf86ce80c76158d4f5c3cd2ba9f1393dfd4e556887f91 6854085242a05ec6bcb76659ec3d0d2fedae3fd1608a701c2265f5fd83c9b1ed3152 ddac7426c03e33902cbe8fd1da6174abe8d3568c9b76b149e0b77d61ac15b8fb118ce 5f9d1e448e216f375ef1f96a5239619459b131026143e8809bad40f5e6f66cd3da2 27431e68670c0b4b2c3801e19025be1ed218e0956967158ccc274c704acd8cc23 c14a99ed2a5478742dadc15f233484535e4021000b5d557313d4f271758680ed5e 7f6f81fd19f9bb9a748cabb2377aac1387f7db80e618e7d69a368729ca9a092af91e be1c58435fe62734d1d53d0b35d0d2093a201c889ad37a558b610f1ab00179a11f 881600e944edc47a7ae6d828009d7c51ffe9ad55a5406406e2e85dc056e47b5758 9eab1a8e792f4631af62d4588a1818167274273c69e7a0735be5dada7e224e3b170b
```
Appendix B. Acknowledgements

Many thanks to Roman Danyliw, Elwyn Davies, Scott Fluhrer, Ben Kaduk, Laurence Lundblade, John Mattsson, Jim Schaad, and Tony Putman for their valuable review and insights. In addition, an extra special thank you to Jim Schaad for generating the examples in Appendix A.

Author’s Address

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA 20170
US

Email: housley@vigilsec.com