Use of the Hash-based Signature Algorithm with CBOR Object Signing and Encryption (COSE)
draft-ietf-cose-hash-sig-03

Abstract

This document specifies the conventions for using the HSS/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 November 11, 2019.

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 the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

This document specifies the conventions for using the HSS/LMS hash-based signature algorithm with the CBOR Object Signing and Encryption (COSE) [RFC8152] syntax. The Leighton-Micali Signature (LMS) system provides a one-time digital signature that is a variant of Merkle Tree Signatures (MTS). The Hierarchical Signature System (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.

Table of Contents

1. Introduction ........................................... 2
   1.1. Algorithm Security Considerations .............. 3
   1.2. Terminology ..................................... 4
2. LMS Digital Signature Algorithm Overview ............ 4
   2.1. Hierarchical Signature System (HSS) ............ 4
   2.2. Leighton-Micali Signature (LMS) ................. 5
   2.3. Leighton-Micali One-time Signature Algorithm (LM-OTS) .... 6
3. Hash-based Signature Algorithm Identifiers .......... 7
4. Security Considerations ................................ 8
   4.1. Implementation Security Considerations .......... 8
5. Operational Considerations ............................. 9
6. IANA Considerations .................................... 9
   6.1. COSE Algorithms Registry Entry ................... 9
   6.2. COSE Key Types Registry Entry .................... 10
7. References ............................................. 10
   7.1. Normative References .............................. 10
   7.2. Informative References ............................ 10
Appendix A. Examples ...................................... 11
   A.1. Example COSE Full Message Signature ............. 11
   A.2. Example COSE_Sign0 Message ....................... 17

Appendix B. Acknowledgements ............................. 22

Author’s Address ........................................ 22
1.1. Algorithm Security Considerations

There have been recent advances in cryptanalysis and advances in the development of quantum computers. Each of these advances pose a threat to widely deployed digital signature algorithms.

At Black Hat USA 2013, some researchers gave a presentation on the current state of public key cryptography. They said: "Current cryptosystems depend on discrete logarithm and factoring which has seen some major new developments in the past 6 months" [BH2013]. Due to advances in cryptanalysis, they encouraged preparation for a day when RSA and DSA cannot be depended upon.

Peter Shor showed that a large-scale quantum computer could be used to factor a number in polynomial time [S1997], effectively breaking RSA. If large-scale quantum computers are ever built, these computers 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 quantum computers that have more than a trivial number of quantum bits (qu-bits). It is open to conjecture when it will be feasible to build such computers; however, RSA, DSA, ECDSA, and EdDSA are all vulnerable if large-scale quantum computers come to pass.

The HSS/LMS signature algorithm does not depend on the difficulty of discrete logarithm or factoring, as a result these algorithms are considered to be post-quantum secure.

Hash-based signatures [HASHSIG] are currently defined to use exclusively SHA-256 [SHS]. An IANA registry is defined so that other hash functions could be used in the future. LM-OTS signature generation prepends a random string as well as other metadata before computing the hash value. The inclusion of the random value reduces the chances of an attacker being able to find collisions, even if the attacker has a large-scale quantum computer.

Today, RSA is often used to digitally sign software updates. This means that the distribution of software updates could be compromised if a significant advance is made in factoring or a large-scale quantum computer is invented. The use of HSS/LMS hash-based signatures to protect software update distribution, perhaps using the format that is being specified by the IETF SUIT Working Group, will allow the deployment of software that implements new cryptosystems.
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:

- Hierarchical Signature System (HSS) -- see Section 2.1;
- Leighton-Micali Signature (LMS) -- see Section 2.2; and
- Leighton-Micali One-time Signature Algorithm (LM-OTS) -- see Section 2.3.

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:

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

The elements of the HSS signature value for a tree with Nspk signed public keys can be summarized as:

```
  u32str(Nspk) ||
  signed_public_key[0] ||
  signed_public_key[1] ||
  ...
  signed_public_key[Nspk-2] ||
  signed_public_key[Nspk-1] ||
  lms_signature /* signature of message */
```

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)

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. This 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 LMS public key can be summarized as:

\[ u32str(lms\_algorithm\_type) \ || \ u32str(otstype) \ | I \ | T[1] \]

An 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:

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

### 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. This specification supports only SHA-256 [SHS], with \( n=32 \).

- **H** - A preimage-resistant hash function that accepts byte strings of any length, and returns an \( n \)-byte string. This specification supports only SHA-256 [SHS].

- **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 A 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:

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

### 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. The byte string is designed for easy parsing, and it includes a counter 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 present, and it 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

4.1. Implementation Security Considerations

Implementations must protect the private keys. Use of a hardware security module (HSM) is one way to 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, care must be taken to preserve confidentiality and integrity.

When a LMS key pair is generating a LMS key pair, an implementation must must generate the key pair and the corresponding identifier 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. [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 (PRNGs) to generate these values is much less severe than 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.

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 has the following columns:

Name:  HSS-LMS

Value:  TBD (Value to be assigned by IANA)

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 has the following columns:

Name: HSS-LMS

Value: TBD (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)

7. References

7.1. Normative References


7.2. Informative References


Appendix A.  Examples

This appendix provides an example of a COSE full message signature
and an example of a COSE_Sign0 message.

The programs that were used to generate the examples can be found at

A.1.  Example COSE Full Message Signature

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

{...}
"title":"HSS LMS Hash based signature - hsssig-01",
"input":{
  "plaintext":"This is the content.",
  "sign":{
    "protected":{
      "ctyp":0
    },
    "signers":{
      "key":{
        "kty":"HSS-LMS",
        "kid":"ItsBig",
        "comment":"1 level key - LM_SHA256_MD32_H10 + \LMOTS_SHA256_N32_W4 ",
        "public":"0000000100000006000000003d08fabd4a20 91ff0a8cb4ed834e7453432a58885cd9ba0431235 \466bf9651c6c92124404d45fa53cf161c28f1ad5a8e",
        "private":"1|6|3|558b8966c48ae9cb898b423c8344 \3a4e01a72f1b1ab5cc85cf1d892903b5439|1|d0 \8fabd4a2091ff0a8cb4ed834e74534"
    },
    "unprotected":{
      "kid":"ItsBig"
    },
    "protected":{
      "alg":"HSS-LMS"
    }
  }
},
"rng_description":"Random value for signature",
"rng_stream":{
  "ACFC5C7377D45C969DF7D289882A48C1A0E5C48B6E29DF5018D \3E683E36BC5"
},
"intermediates":{
  "signers":{
    "ToBeSign_hex":"85695369676E617475726543A103004AA10 1674853532D4C4D53405454686973206973207468652063 \6F6E74656E742E"
  }
},
"output":{
  "cbor_diag":"98\{\text{h'A10300'}, \}, \text{h'54686973206973207468652063 \0636F6E74656E742E', \text{\{h'A101674853532D4C4D53', \"}}
cbor": "D8628443A10300A05454686973206973207468520636F6E7 \ 4656E742E8134AA10167485352324C5D3A1044694774326967 \ 5909D0000000000100000391291DE76CE6E24D1E2A9B6 \ 0266519BC8CE889F8114EB0F0C00ED3129DE3A9BFC05A46923 \ 923AA3209BF9E1480AB789067949280AD36001C182CB3349C \ 0035FEE2E4228770D22F85687A18AEE76CD2F8E8F40043B14 \ A6E8729F679F7E3A5A34594763EEB70E840FBBFDAQ9BEC59BF \ 0236FDD34ACE31C1EAD1BD2B0213A0946160F30168A6E193C5 \ 7C32BB017C22529CE3760F93358633D5A69F7F0850BD720E72F \ 75B18D4E72D11B6321BDF1859102E723A3B1FA1E5BC53EC8 \ 732FF1B2C4D384137E8EEC94804CB47C823C0B141E28B178E1 \ F5A904CF7592AACE820C97E7714B9FCA4BAE978548400A705 \ CAE7B9112D182C21BCE3F10E0A7C0324F6674927961A3477B03 \ E3622169438C27CD46FCAD769D010D0B13A06F5CD0D93A2EE2B \ B0E25FFFD2A08C8DD0F65315187BFD3B346EB566B7A5B0421DF8 \ 7F7FD18080B5D36D347472D90F9F89495C775BF5930956E0DB7 \ F4D5F6B265C3196F9FB660949B2315B4CE7A0DF5E9083B42A9 \ 082D8FD1E2F62770EBCDE42A4A504488515C7BA81FED2BC8FC \ 1B63213254E9E160D0DD79CA4D4F677485BB501AD4DF46D0 \ 45942B935C7D435B34D09D3BC8F87A196B85C0E98091C8787 \ E193C84EC365555807A17CE1BBF8304DB7D63646EF8F1CFE9E \ 9071BB67A123F6C3F7638CDA04F4B038407406917101C4670 \ 163B9266253DE3B0D79170B0297CD4B88005743B6F3826B2 \ 491CD1E1F6069B306C4EF8AE4CF78F3EB320A20460A7FF8F84BF \ 22AD876B4661ED5D3BF3559162F1D536C1BD472B18D93E9D3A \ 1ABC8D5CEA706B7D7BFA5105E8D31C1AA6BD252071D68C9DCC \ 1EDF8EA2C326C564F7578FC8936515A34413C39F2AF8164AF299 \ 9C6994C54F042933F0E6ED84E846C14B234EAD8B7ABE3AC935BD9 \ 682E6D12150611692847327E147E7C7CD9BD436E7A274664 \ B21BE6D3046C615E53579775196AA4CD59ED5F5534E4C9579D \ F17873BB5E47827D83A4BDB22849DA579C9A106672BDEBE25 \ 6B2D5C66E327005130C0FC04F26549A6FC5DBB5634C0BFB66F1F \ AAAAD6692D32BA3B36F62C6DD607DAA867A8923C4CF1CC8AB847 \ F33DF12617C38A7B1DB13E6B26DE23F5A9F8E706966B717A5B4 \ 92C6A25920E8F67F666D602222CD488527D5BD5ADA2BF2892D2 \ BB7EC0B83334BBEE806A54B1DE69388530BF96696F71245 \ 7AF44705040DF843FABF01076DF607520845044ACCEA54DC487 \ 2A97CF2DCB0A996801CB7C22FD93987A6630PF4A825A405A3 \ E5BA148F5ADF72A87FD4B24DCE4E65C1A2B2AD2F2B19A74CA \ 3027383B318866E85A169923D3B4C4135E47220182C1FB74 \ 45037C3524614CF37006D27C0C4A314E778BADCF7D961551A9 \ FCB95A8D162A94110BC703F5F49CB8532CA9007322F2DBEE5D5C \ 237FAED2B2F9C953EF9E5EBBDC0C505BFB69475A873234A95E2C6 \ 6012B0CCE1E507AD10FC01D7E00FBB5565E9A8C761B812357F0BE \ 844DF14C35C82E80BEB1603D87C0CA6E9F67EB8891A4581122F3 \ 0D0DCCF997F6F7CB63758C70DCE7353DA1403BAAAC8DB3749211 \ 98081D77F52DB3F6B28147353BB11C6D563DD7336322021F584E \ 207FD61222C4FFPEAA742147C6348B6717C95905CA059A3686A7
A.2. Example COSE_Sign0 Message

This section provides an example of a COSE_Sign0 message.

```json
{
    "title":"HSS LMS Hash based signature - hssig-sig-01",
    "input":{
        "plaintext":"This is the content.",
        "sign0":{
            "key":{
                "kty":"HSS-LMS",
                "kid":"ItsBig",
                "comment":"1 level key - LM_SHA256_MD32_H10 + \
                LMOTS_SHA256_N32_W4 ",
                "public":"000000001000000006000003d08fabd4a2091ff0a \ 
                8cb4ed834e7453432a58885cd9ba0431235466bfff651c6 \ 
                c92124404d45fa53cf161c28f1ad5a8e",
                "private":"1|6|3|558B8966C48AE9C8B98B423C83443AAE01 \ 
                4A72F1B1AB5CC85CF1D892903B5439|0|d08fabd4a2091f \ 
                f0a8cb4ed834e74534"
            },
            "unprotected":{
                "kid":"ItsBig"
            },
            "protected":{
                "alg":"HSS-LMS"
            },
            "alg":"HSS-LMS"
        },
        "rng_description":"Random value for signature",
        "rng_stream":{
            "1D5112D38A1146402875B73BC8D4B59C845C6AE61D03A70ABAD09 \ 
            8AC05AD8297"
        },
        "intermediates":{
            "ToBeSign_hex":"846A5369676E6174757265314AA101674853532D4 \ 
            C4D5340545468697320697320746686520636F6674656E742E"
        },
        "output":{
            "cbor_diag":"18([h' \A101674853532D4C4D53', \ 
            4: h'497473426967'], \ 
            h'5468697320697320746686520636F6674656E742E', \ 
            h'0000000000000000000391291DE76CE624D1E2A9B60266 \ 
            DC1F084B1158208249F28F4F7C7E931BA7B3BD0D824A4570"
        }
    }
}
```
5C15306447892E72C8CF461E3DF57E696AF1780AAD0A4F847F8FD
342A8502F0C98A38C696ED74F8A300D171BBAE888B226498CF63
EBA035185457E3A552ED5DA56DD90C1372D82386D000CA5F424E
8BF1FCA88A7173ED185E6F1D1EF2BAE53D701D034BF3B44F08
42BED8126494A7FD2C7B1321A5527B78681B1D1302CE4DAE86C6
8DAB48151D857934250E6DBF99D37DA15735831C2FA31DD2AB81
F8F1F2DE8D890DF298ACC03D411353EAD09C075860F9843B14B
E940871A4E91821D3642E5B0126DB8F9417757A152DCB3EF633D6
FDC357C2E5984EDC9101AE78BBC6C3B30E6051ECE48391FE9
42BAE97EBC7270ECDCCD07989CAD42789422EDC1BD22E5191E
B984A92F89236286F6327084B09A95732EF82F27007D5DDA084
03E8E553E47E0C1E3A567B8850F8E4D004170995CE1FCC2284E
ACB197F001C9B0F2F2FB7C24C5774A935F96761FD52394ADE47
8965E520B0D11F26979F791953CF026590919A90804CA35A94D5
7A52020861ADC270845EEBEE08D85C442A0ECBB5C7865E11A5C0
71D5202817A3C8EE741B4B1E2807E30BD9A2A1D1FE5F0E387F250
1926D87F372D461Z79F8774E979A93363527C13948DBE17767BFB
8A6467C576402481704E4E056D67F00EEC4399CED2B8802CC89A788
27B12DFD5DB8A3A194A8D05B3D3CF3C87E91AE3DBB05B685CE
C129406FAA395B0A042D06E929D67E62C6ED0C01FC2A9ABA2C
44FD0D7847B86A544AB53BD22E49C2AC942330B44F95AEF50A1
E4476E12E52670DA2C72912541C712986851BF022932B70DE30
1B1E9988131C73BD0B4E748FF2C2FFD457DB9F9EF1FA579F02930
D3C3AAA3D46F50A695FC3E511E035B695342E855B7D756B001BA
792C8B6F452A14F49DBE647A37095A28482548E3C0E9DFBDC02540
BE2EB609863660B0E45FF8847BD7392ED101AA85589B36455B8
64A682B58A87CCE0BE917C83BE0295186178B1E2CD26404B8FF
4A300FEEAE2A007E75312500E0EBB776621CEECF255E08727BBBE1
178A3CF25395468F207249A97FE9A631D9C651FD36AD2C20B08
1666E6C89E6E72BCBC6149E9C97B4AC86F65F9B54A800C8104
5CC7A6D75040A872273F1C1A2BBA06E060F5F9C63AA4BBBCE
603BBABF9DE4E2B95F6ED96E444F4ABD20524A2D74AEE0AD427C5E
0D5DFDE3267DDD2D4256752FB643D36E8A1B7B5E4481A51763768
04FF849DF09FDA86836DB400000067B95DE44AF89161DF4B91A
4A9EB3F15A39A4660F98F06BF30176869DFC362C94664B3C9
884003E69D9BA1F7D3967F709CCD35DC77D60356FC03680890B4
91CB4ECBBAEBC128E7C81A46E62A67B57640A0A788E1CF077DD94
19A10C6866D16621A8016FDB5BCD56211D272CA70B811117D1
29529A7570CF79CF52A7082AA4853C38EB3D3D35D62D26264595
C4FB73A525ASE2C30524EB180C2CE9C197CB697789F89F5D
3D130B01531696CEF93C6A52546BF96E90D75E383BB754367C5
842B4AFBFC7CD8B483B3276C29D4F0A341C2D4064E04D465387E4D
045851AC6A0A0EA94C71BB05C4ED463EE8C107362F0FCD808C1
4D0AC495C1670326D14752FF341C1CD2C427581C18C2CF40E48E
9CE949BE7C888E9CAE6BE4A415E291FD109721DCF084B11582082
49F284F47C7E931BA73BD0D824A4570*
}  

Appendix B. Acknowledgements

Many thanks to Scott Fluhrer, 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