INTERNET-DRAFT
Intended Status: Informational
Expires: September 15, 2013

L. Hitt
21CT, Inc.

March 14, 2013

ZSS Short Signature Scheme
draft-irtf-cfrg-zss-00

Abstract

This document describes the ZSS Short Signature Scheme for implementation from bilinear pairings on supersingular elliptic curves. The ZSS Short Signature Scheme uses general cryptographic hash functions such as SHA-1 or SHA-2 and is efficient in terms of pairing operations.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Table of Contents

1 Introduction

This document describes the ZSS Short Signature Scheme (designed by Zhang, Safavi-Naini, and Susilo) for implementation from bilinear pairings [ZSS]. It does not require any special hash function such as MapToPoint [B-F], which is still probabilistic and generally inefficient, but rather can use cryptographic hash functions such as SHA-1 or SHA-2.

This document is restricted to implementation of ZSS on a particular family of supersingular elliptic curves, though the scheme is valid on other elliptic curve groups. This supersingular family offers efficiency and simplicity advantages when computing the pairing, which is the most time consuming procedure in pairing-based cryptography. These advantages are important since short signatures are needed in low-bandwidth communication environments.

The scheme is constructed from the Inverse Computational Diffie-Hellman Problem (Inv-CDHP) on bilinear pairings (see Section 1.2 below for a discussion of Inv-CDHP). The security of the scheme is based on the assumed hardness of this problem (which is widely accepted), which means there is no polynomial time algorithm to solve it with non-negligible probability. Bilinear pairings have been used to construct Identity (ID)-Based cryptosystems [B-F], so that the identity information of a user functions as his public key. The signing process in a short signature scheme can be regarded as the private key extract process in the ID-based public key setting from bilinear pairings. Therefore, the ZSS signature scheme can be regarded as being derived from Sakai-Kasahara's ID-based encryption scheme with pairing [S-K, RFC6508].

The algorithm is for use in the following context:

* where there are two parties, a Signer and a Verifier;

* where a message is to be signed and then verified (e.g., for authenticating the initiating party during key establishment);

* where a Certificate Authority (CA) or Trusted Third Party (TTP) within a traditional Public Key Infrastructure (PKI) provides a root of trust for both parties.

1.1 Bilinear Pairings

Let G_1 be a cyclic additive group generated by P, whose order is a prime q. Let G_2 be a cyclic multiplicative group with the same order q. Let Z_q be the additive group of integers modulo q.

Let <,>: G_1 X G_1 --> G_2 be a map with the following properties.

1. Bilinearity: <aP,bQ>=<P,Q>^(ab) for all P, Q elements of G_1 and a, b elements of Z_q.

2. Non-degeneracy: There exists P, Q elements of G_1 such that <P,Q> != 1. In other words, the map does not send all pairs in G_1 X G_1 to the identity in G_2.

3. Computability: There is an efficient algorithm to compute <P,Q> for all P, Q in G_1.

In our setting of prime order groups, non-degeneracy is equivalent to <P,Q> != 1 for all nontrivial P, Q elements in G_1. So, when P is a generator of G_1, <P,P> is a generator of G_2. Such a bilinear map is called a bilinear pairing.

1.2 Discrete Logarithm Problem and Diffie-Hellman Problems

We consider the following problems in the additive group (G_1;+).

Discrete Logarithm Problem (DLP): Given two group elements P and Q, find an integer n in (Z_q)*, such that Q=nP whenever such an integer exists.

Decision Diffie-Hellman Problem (DDHP): For a,b,c in (Z_q)*, given P, aP, bP, cP decide whether c is congruent to ab mod q.

Computational Diffie-Hellman Problem (CDHP): For a,b in (Z_q)*, given P, aP, bP, compute abP.

Inverse Computational Diffie-Hellman Problem (Inv-CDHP): For a in (Z_q)*, given P, aP, compute [a^(-1)]P.

Square Computational Diffie-Hellman Problem (Squ-CDHP): For a in (Z_q)*, given P, aP, compute [a^2]P.

Bilinear Diffie-Hellman problem (BDHP): Given (P, aP, bP, cP) for some a,b,c in (Z_q)*, compute v in G_2 such that v = <P,P>^(abc).

The CDHP, Inv-CDHP, and Squ-CDHP are polynomial time equivalent. The DLP, CDHP, Inv-CDHP, Squ-CDHP, and BDHP are assumed to be hard, which means there is no polynomial time algorithm to solve any of them with non-negligible probability. Therefore, the security of pairing based cryptosystems are typically based on these problems. A Gap Diffie-Hellman (GDH) group is a group in which the DDHP can be efficiently solved but the CDHP is intractable. The bilinear pairing gives us such a group, found on supersingular elliptic curves or hyperelliptic

curves over finite fields. The bilinear pairings can be derived from the Weil or Tate pairing, as in [B-F, Cha-Cheon, Hess]. The ZSS scheme works on any GDH group, but in this document we focus on a particular family of supersingular elliptic curves described in Section 3.4 and the pairing described in Appendix A.2.

1.3 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2 Architecture

We consider the situation where one entity (the Signer) wishes to sign a message that it is sending to another entity (the Verifier).

As in a traditional Public Key Infrastructure (PKI), a Certificate Authority (CA) or Trusted Third Party (TTP) provides assurance of a signer's identity, which is bound to the signer's public key. The CA may generate a public key and private key (a key pair) or the signer may generate their own key pair and register the Signer Public Key (SPK) with a CA.

The mechanism by which a secret key is transported MUST be secure, as the security of the authentication provided by ZSS signatures is no stronger than the security of this supply channel.

During the signing process, once the Signer has formed its message, it signs the message using its Signer Secret Key (SSK). It transmits the Signature with the message. The Verifier MUST then use the message, Signature, and SPK in verification.

This document specifies

* an algorithm for creating a Signature from a message, using an SSK;

* an algorithm for verifying a Signature for a message, using an SPK.

This document does not specify (but comments on)

* how to choose a valid and secure elliptic curve;

* which hash function to use.

3 Notation, Definitions and Parameters

3.1 Notation

n A security parameter; n should be at most half the bit size of q.

p A prime, of size at least 2n bits, which is the order of the finite field F_p. In this document, p is always congruent to 3 modulo 4.

F_p The finite field of order p (i.e., field with p elements). All elliptic curve points will be defined over F_p.

F* The multiplicative group of the non-zero elements in the field F; e.g., (F_p)* is the multiplicative group of the finite field F_p.

q An odd prime that divides p + 1. To provide the desired level of security, lg(q) MUST be greater than 2*n.

E An elliptic curve defined over F_p, having a subgroup of prime order q. In this document, we use supersingular curves with equation y^2 = x^3 - 3 * x modulo p.

E(F) The additive group of points of affine coordinates (x,y) with x, y in the field F, that satisfy the curve equation for E.

P A point of E(F_p) that generates the cyclic subgroup of order q.

0 The null element of any additive group of points on an elliptic curve, also called the point at infinity.

F_p^2 The extension field of degree 2 of the field F_p. In this document, we use a particular instantiation of this field; F_p^2 = F_p[i], where i^2 + 1 = 0. It is for this reason that we choose p congruent to 3 modulo 4.

PF_p The projectivization of F_p. We define this to be (F_p^2)*/(F_p)*. Note that PF_p is cyclic and has order p + 1, which is divisible by q.

G[q] The q-torsion of a group G. This is the subgroup generated by points of order q in G.

< , > A version of the Tate-Lichtenbaum pairing. In this document, this is a bilinear map from E(F_p)[q] x E(F_p)[q] onto the subgroup of order q in PF_p. A full definition is given in Appendix A.2.

g g = <P,P>. Having this pre-computed value allows the Verifier to only perform one pairing operation to verify a signature.

H A cryptographic hash function. [FIPS180-3] contains NIST approved hash functions.

lg(x) The base 2 logarithm of the real value x.

3.2 Definitions

Certificate Authority (CA) - The Certificate Authority is a trusted third party who provides assurance that the SPK belongs to the signer and verified proof of the signer's identity when the signer registered the SPK.

Public parameters - The public parameters are a set of parameters that are held by all users of the system. Each application of ZSS MUST define the set of public parameters to be used. The parameters needed are p, q, E, P, < , >, g, H, and n.

Signer Public Key (SPK) - The Signer's Public key is used to verify the signature of the entity whose SSK corresponds to the SPK. It is a point on the elliptic curve E.

Signer Secret Key (SSK) - The Signer's Secret Key is used to generate a signature and must not be revealed to any entity other than the trusted third party and the authorized signer. It is a value between 2 and q-1.

3.3 Representations

This section provides canonical representations of values that MUST be used to ensure interoperability of implementations. The following representations MUST be used for input into hash functions and for transmission. In this document, concatenation of octet strings s and t is denoted s || t.

Integers Integers MUST be represented as an octet string, with bit length a multiple of 8. To achieve this, the integer is represented most significant bit first, and padded with zero bits on the left until an octet string of the necessary length is obtained. This is the octet string representation described in Section 6 of [RFC6090].

F_p elements Elements of F_p MUST be represented as integers in

	the range 0 to p-1 using the octet string representation defined above. Such octet strings MUST have length L = Ceiling(lg(p)/8).
F_p^2 elements	The elements of F_p^2 = F_p[i] are represented as x_1 + i * x_2, where x_1 and x_2 are elements of F_p. It is for this reason that we choose p congruent to 3 modulo 4.
PF_p elements	Elements of PF_p are cosets of (F_p)* in (F_p^2)*. Every element of F_p^2 can be written unambiguously in the form x_1 + i * x_2, where x_1 and x_2 are elements of F_p. Thus, elements of PF_p (except the unique element of order 2) can be represented unambiguously by x_2/x_1 in F_p. Since q is odd, every element of PF_p[q] can be represented by an element of F_p in this manner. Elements of PF_p MUST be represented as an element of F_p using the algorithm in Appendix A.2. They are therefore represented as octet strings as defined above and are L octets in length. Representation of the unique element of order 2 in PF_p will not be required. This representation of elements in PF_p[q] allows efficient implementation of PF_p[q] group operations, as these can be defined using arithmetic in F_p. If a and b are elements of F_p representing elements A and B of PF_p[q], respectively, then A * B in PF_p[q] is represented by (a + b)/(1 - a * b) in F_p.
Points on E	Elliptic curve points MUST be represented in uncompressed form as defined in Section 2.2 of [RFC5480]. For an elliptic curve point (x,y) with x and y in F_p, this representation is given by 0x04 || x' || y', where x' is the octet string representing x, y' is the octet string representing y, and || denotes concatenation. The representation is 2*L+1 octets in length.

3.4 Arithmetic

ZSS relies on elliptic curve arithmetic. The coordinates of a point P on the elliptic curve are given by P = (P_x,P_y), where Px and Py are the affine coordinates in F_p satisfying the curve equation.

The following conventions are assumed for curve operations:

Point addition - If P and Q are two points on a curve E, their sum is denoted as P + Q.

Scalar multiplication - If P is a point on a curve, and k an integer, the result of adding P to itself a total of k times is denoted [k]P.

In this document, we use supersingular curves with equation y^2 = x^3 - 3 * x modulo p. This curve is chosen because of the efficiency and simplicity advantages it offers. The choice of -3 for the coefficient of x provides advantages for elliptic curve arithmetic that are explained in [P1363]. A further reason for this choice of curve is that Barreto's trick [Barreto] of eliminating the computation of the denominators when calculating the pairing applies.

4 The ZSS Cryptosystem

This section describes the ZSS short signature scheme [ZSS].

4.1 Parameter Generation

The following static parameters are fixed for each implementation. They are not intended to change frequently, and MUST be specified for each user community.

The system parameters, in general, are {G1, G2, q, P, <,>, g, H}. In this document, G1 will be E(F_p) and G2 is PF_P=(F_p^2)*/(F_p)*, so the specific parameters to be generated for a given security parameter n are {p, E, q, P, <,>, g, H}. These are known by the Sender and the Verifier.

4.2 Key Generation

To create signatures, each Signer requires an SSK and SPK. The SSK is an integer, and the SPK is an elliptic curve point. The SSK MUST be kept secret (to the Signer and possibly the CA), but the SPK need not be kept secret.

The Signer (or CA) MUST randomly select a value in the range 2 to q-1, and assigns this value to x, which is the SSK.

The Signer MUST derive its SPK, X, by performing the calculation X =[x]P.

If the signer generated the SPK, then it must be registered with a CA.

4.3 Signature Generation

Given the SSK x, and a message m, the Signer computes the signature S by performing the following steps:

1) Compute the hash of the message as a mod q value using the hash algorithm specified in the public parameters.

2) Compute (H(m)+x)^-1, where the inversion is performed modulo q.

3) Compute S = [(H(m)+x)^-1]P. The signature is S, and this is a point on the curve E.

The Signer sends m and S.

4.4 Signature Verification

Given the SPK X, a message m, and a signature S, the Receiver verifies that <[H(m)]P + X, S> = g, to ensure that the Signer is authentic and the message was not altered in transit. This is achieved by the Verifier performing the following steps:

1) Check that S is a point on the curve E, otherwise reject the signature.

2) Compute the hash of the message as a mod q value using the hash algorithm specified in the public parameters.

3) Compute the elliptic curve point [H(m)]P + X.

4) Compute the pairing <[H(m)]P + X, S>.

5) Verify that <[H(m)]P + X, S> = g; if not, reject the signature.

5 Security Considerations

This document describes the ZSS Short Signature Scheme. We assume that the security provided by this algorithm depends entirely on the secrecy of the secret keys it uses, and that for an adversary to defeat this security, he will need to perform computationally intensive cryptanalytic attacks to recover a secret key. Note that a security proof exists for ZSS in the Random Oracle Model [ZSS].

When defining public parameters, guidance on parameter sizes from [RFC4492] SHOULD be followed. Note that the size of the F_p^2 discrete logarithm on which the security rests is 2*lg(p). Table 1 shows bits of security afforded by various sizes of p. The order of the base point P used in ZSS MUST be a large prime q. If k bits of

security are needed, then lg(q) SHOULD be chosen to be at least 2*k. Similarly, if k bits of security are needed, then a hash with output size at least 2*k SHOULD be chosen.

Bits of Security	lg(p)
80	512
112	1024
128	1536
192	3840
256	7680

Table 1: Comparable Strengths, taken from [RFC4492]

Randomizing the messages that are signed is a way to enhance the security of the cryptographic hash function. [SP800-106] provides a technique to randomize messages that are input to a cryptographic hash function during the signature generation step. The intent of this method is to strengthen the collision resistance provided by the hash functions without any changes to the core hash functions and signature algorithms. If the message is randomized with a different random value each time it is signed, it will result in the message having a different digital signature each time.

Each user's SSK protects the ZSS communications it receives. This key MUST NOT be revealed to any entity other than the authorized user and possibly the CA (if the CA generated the key pair).

In order to ensure that the SSK is received only by an authorized entity, it MUST be transported through a secure channel. The security offered by this signature scheme is no greater than the security provided by this delivery channel.

The randomness of values stipulated to be selected at random, as described in this document, is essential to the security provided by ZSS. If the value of x used by a user is predictable, then the value of his SSK could be recovered. This would allow that user's signatures to be forged. Guidance on the generation of random values for security can be found in [RFC4086].

6 IANA Considerations

This memo includes no request to IANA.

7 References

7.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011.

[ZSS] Zhang, F., Safavi-Naini, R., and Susilo, W., "An Efficient Signature Scheme from Bilinear Pairings and Its Applications", PKC 2004, LNCS 2947, Springer-Verlag (2004), pp. 277-290.

7.2 Informative References

[Barreto] Barreto, P., Kim, H., Lynn, B., and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems", Advances in Cryptology - Crypto 2002, LNCS 2442, Springer-Verlag (2002), pp. 354-369.

[B-F] Boneh, D., Franklin, M., "Identity-based encryption from the Weil pairing", Advances in Cryptology - Crypto 2001, LNCS 2139, Springer-Verlag (2001), pp. 213-229.

[Cha-Cheon] Cha, J.C., Cheon, J.H., "An identity-based signature from gap Diffie-Hellman groups", Public Key Cryptography - PKC 2003, LNCS 2139, Springer-Verlag (2003), pp. 18-3.

[FIPS180-3] Federal Information Processing Standards Publication (FIPS PUB) 180-3, "Secure Hash Standard (SHS)", October 2008.

[Hess] Hess, F., "Efficient identity based signature schemes based on pairings", SAC 2002, LNCS 2595, Springer-Verlag (2002), pp. 310-324.

[Miller] Miller, V., "The Weil pairing, and its efficient calculation", J. Cryptology 17 (2004), 235-261.

[P1363] IEEE P1363-2000, "Standard Specifications for Public-Key

Cryptography", 2001.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC6508] Groves, M., "Sakai-Kasahara Key Encryption (SAKKE)", RFC 6508, February 2012.

[S-K] Sakai, R., Ohgishi, K., and M. Kasahara, "ID based cryptosystem based on pairing on elliptic curves", Symposium on Cryptography and Information Security - SCIS, 2001.

[SP800-106] Dang, Q., "Randomized Hashing for Digital Signatures", NIST Special Publication 800-106, February 2009.

Appendix A. Elliptic Curves, Pairings and Supporting Algorithms

E is a supersingular elliptic curve (of j-invariant 1728). E(F_p) contains a cyclic subgroup of order q, denoted E(F_p)[q], whereas the larger object E(F_p^2) contains the direct product of two cyclic subgroups of order q, denoted E(F_p^2)[q].

P is a generator of E(F_p)[q]. It is specified by the (affine)coordinates (P_x,P_y) in F_p, satisfying the curve equation.

Routines for point addition and doubling on E(F_p) can be found in Appendix A.10 of [P1363].

A.1. E(F_p^2) and the Distortion Map

If (Q_x,Q_y) are (affine) coordinates in F_p for some point (denoted Q) in E(F_p)[q], then (-Q_x,iQ_y) are (affine) coordinates in F_p^2 for some point in E(F_p^2)[q]. This latter point is denoted [i]Q, by analogy with the definition for scalar multiplication. The two points P and [i]P together generate E(F_p^2)[q]. The map [i]: E(F_p)-> E(F_p^2) is sometimes termed the distortion map. This map is used to ensure the pairing is applied to independent points so that the pairing is not equal to 1.

A.2. The Tate-Lichtenbaum Pairings

As in [RFC6508], we describe the pairing < , > to be used in ZSS. We will need to evaluate polynomials f_R that depend on points on E(F_p)[q]. Miller's algorithm [Miller] provides a method for evaluation of f_R(X), where X is some element of E(F_p^2)[q] and R is some element of E(F_p)[q] and f_R is some polynomial over F_p whose divisor is (q)(R) - (q)(0). Note that f_R is defined only up to scalars of F_p.

The version of the Tate-Lichtenbaum pairing used in this document is given by <R,Q> = f_R([i]Q)^c / (F_p)*. It satisfies the bilinear relation <[x]R,Q> = <R,[x]Q> = <R,Q>^x for all Q, R in E(F_p)[q], for all integers x. Note that the domain of definition is restricted to E(F_p)[q] x E(F_p)[q] so that certain optimizations are natural.

We provide pseudocode for computing <R,Q> with elliptic curve arithmetic expressed in affine coordinates. We make use of Barreto's trick [Barreto] for avoiding the calculation of denominators. Note that this section does not fully describe the most efficient way of computing the pairing; it is possible to compute the pairing without any explicit reference to the extension field F_p^2. This reduces the number and complexity of the operations needed to compute the pairing.

```
<CODE BEGINS>

/* Copyright (c) 2012 IETF Trust and the persons identified as
authors of the code.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, is permitted pursuant to, and subject to the license
terms contained in, the Simplified BSD License set forth in Section
4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info). */

     Routine for computing the pairing <R,Q>:

       Input R, Q points on E(F_p)[q];

       Initialize variables:
          v = (F_p)*;    // An element of PF_p[q]
          C = R;         // An element of E(F_p)[q]
          c = (p+1)/q;   // An integer

       for bits of q-1, starting with the second most significant
       bit, ending with the least significant bit, do

          // gradient of line through C, C, [-2]C.

          l = 3*( C_x^2 - 1 ) / ( 2*C_y );

          //accumulate line evaluated at [i]Q into v

          v = v^2 * ( l*( Q_x + C_x ) + ( i*Q_y - C_y ) );

          C = [2]C;

          if bit is 1, then

             // gradient of line through C, R, -C-R.

             l = ( C_y - R_y )/( C_x - R_x );

             //accumulate line evaluated at [i]Q into v

             v = v * ( l*( Q_x + C_x ) + ( i*Q_y - C_y ) );

             C = C+R;

           end if;

         end for;
```

```
            t = v^c;

            return representative in F_p of t;

        End of routine;

        Routine for computing representative in F_p of elements of
        PF_p:

          Input t, in F_p^2, representing an element of PF_p;

          Represent t as a + i*b, with a,b in F_p;  return b/a;

        End of routine;

    <CODE ENDS>
```

A.3. Hashing to an Integer Range

We use the function HashToIntegerRange(s, n, hashfn) to hash strings to an integer range. Given a string (s), a hash function (hashfn), and an integer (n), this function returns a value between 0 and n - 1.

Input:

* an octet string, s

* an integer, n <= (2^hashlen)^hashlen

* a hash function, hashfn, with output length hashlen bits

Output:

* an integer, v, in the range 0 to n-1

Method:

1) Let A = hashfn(s)

2) Let h_0 = 00...00, a string of null bits of length hashlen bits

3) Let l = Ceiling(lg(n)/hashlen)

4) For each i in 1 to l, do:

   a) Let h_i = hashfn(h_(i - 1))

b) Let v_i = hashfn(h_i || A), where || denotes concatenation

5) Let v' = v_1 || ... || v_l

6) Let v = v' mod n

Appendix B. Example Data

This appendix provides example data for the ZSS short signature scheme with the public parameters (n, E, p, P, q, g, H). Also found in [RFC6808,RFC6809].

n = 128

E: y^2 = x^3 -3x

p = 997ABB1F 0A563FDA 65C61198 DAD0657A
416C0CE1 9CB48261 BE9AE358 B3E01A2E
F40AAB27 E2FC0F1B 228730D5 31A59CB0
E791B39F F7C88A19 356D27F4 A666A6D0
E26C6487 326B4CD4 512AC5CD 65681CE1
B6AFF4A8 31852A82 A7CF3C52 1C3C09AA
9F94D6AF 56971F1F FCE3E823 89857DB0
80C5DF10 AC7ACE87 666D807A FEA85FEB

P = (Px,Py) where

Px = 53FC09EE 332C29AD 0A799005 3ED9B52A
2B1A2FD6 0AEC69C6 98B2F204 B6FF7CBF
B5EDB6C0 F6CE2308 AB10DB90 30B09E10
43D5F22C DB9DFA55 718BD9E7 406CE890
9760AF76 5DD5BCCB 337C8654 8B72F2E1
A702C339 7A60DE74 A7C1514D BA66910D
D5CFB4CC 80728D87 EE9163A5 B63F73EC
80EC46C4 967E0979 880DC8AB EAE63895

Py = 0A824906 3F6009F1 F9F1F053 3634A135
D3E82016 02990696 3D778D82 1E141178
F5EA69F4 654EC2B9 E7F7F5E5 F0DE55F6
6B598CCF 9A140B2E 416CFF0C A9E032B9
70DAE117 AD547C6C CAD696B5 B7652FE0
AC6F1E80 164AA989 492D979F C5A4D5F2
13515AD7 E9CB99A9 80BDAD5A D5BB4636
ADB9B570 6A67DCDE 75573FD7 1BEF16D7

q = 265EAEC7 C2958FF6 99718466 36B4195E
905B0338 672D2098 6FA6B8D6 2CF8068B

```
                BD02AAC9 F8BF03C6 C8A1CC35 4C69672C
                39E46CE7 FDF22286 4D5B49FD 2999A9B4
                389B1921 CC9AD335 144AB173 595A0738
                6DABFD2A 0C614AA0 A9F3CF14 870F026A
                A7E535AB D5A5C7C7 FF38FA08 E2615F6C
                203177C4 2B1EB3A1 D99B601E BFAA17FB

          g  = 66FC2A43 2B6EA392 148F1586 7D623068
                C6A87BD1 FB94C41E 27FABE65 8E015A87
                371E9474 4C96FEDA 449AE956 3F8BC446
                CBFDA85D 5D00EF57 7072DA8F 541721BE
                EE0FAED1 828EAB90 B99DFB01 38C78433
                55DF0460 B4A9FD74 B4F1A32B CAFA1FFA
                D682C033 A7942BCC E3720F20 B9B7B040
                3C8CAE87 B7A0042A CDE0FAB3 6461EA46

          H  = SHA-256 (defined in [FIPS180-3]).

     The SSK is:
          x  = AFF429D3 5F84B110 D094803B 3595A6E2 998BC99F

     The SPK is:
          X  = (Xx,Xy) where

          Xx = 5958EF1B 1679BF09 9B3A030D F255AA6A
                23C1D8F1 43D4D23F 753E69BD 27A832F3
                8CB4AD53 DDEF4260 B0FE8BB4 5C4C1FF5
                10EFFE30 0367A37B 61F701D9 14AEF097
                24825FA0 707D61A6 DFF4FBD7 273566CD
                DE352A0B 04B7C16A 78309BE6 40697DE7
                47613A5F C195E8B9 F328852A 579DB8F9
                9B1D0034 479EA9C5 595F47C4 B2F54FF2

          Xy = 1508D375 14DCF7A8 E143A605 8C09A6BF
                2C9858CA 37C25806 5AE6BF75 32BC8B5B
                63383866 E0753C5A C0E72709 F8445F2E
                6178E065 857E0EDA 10F68206 B63505ED
                87E534FB 2831FF95 7FB7DC61 9DAE6130
                1EEACC2F DA3680EA 4999258A 833CEA8F
                C67C6D19 487FB449 059F26CC 8AAB655A
                B58B7CC7 96E24E9A 39409575 4F5F8BAE

     Suppose H(m) = 3230 31312D30 32007465 6C3A2B34
                    34373730 30393030 31323300

     Signature S = (Sx,Sy) where
          Sx = 93AF67E5 007BA6E6 A80DA793 DA300FA4
                B52D0A74 E25E6E7B 2B3D6EE9 D18A9B5C
```

```
                    5023597B D82D8062 D3401956 3BA1D25C
                    0DC56B7B 979D74AA 50F29FBF 11CC2C93
                    F5DFCA61 5E609279 F6175CEA DB00B58C
                    6BEE1E7A 2A47C4F0 C456F052 59A6FA94
                    A634A40D AE1DF593 D4FECF68 8D5FC678
                    BE7EFC6D F3D68353 25B83B2C 6E69036B

               Sy = 155F0A27 241094B0 4BFB0BDF AC6C670A
                    65C325D3 9A069F03 659D44CA 27D3BE8D
                    F311172B 55416018 1CBE94A2 A783320C
                    ED590BC4 2644702C F371271E 496BF20F
                    588B78A1 BC01ECBB 6559934B DD2FB65D
                    2884318A 33D1A42A DF5E33CC 5800280B
                    28356497 F87135BA B9612A17 26042440
                    9AC15FEE 996B744C 33215123 5DECB0F5

          For verification of the signature:
               <H(m)P + X, S> = g
```

Author's Address

Laura Hitt
6011 W Courtyard Dr.
Building 5, Suite 300
Austin, TX 78730

EMail: Lhitt@21CT.com