

Internet Research Task Force  
Internet-Draft  
Intended status: Informational  
Expires: July 6, 2017

D. Harkins  
HP Enterprise  
January 2, 2017

**Public Key Exchange**  
**draft-harkins-pkex-03**

Abstract

This memo describes a password-authenticated protocol to allow two devices to exchange "raw" (uncertified) public keys and establish trust that the keys belong to their respective identities.

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 <http://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 July 6, 2017.

Copyright Notice

Copyright (c) 2017 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 (<http://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.

## Table of Contents

|                             |                                                         |                    |
|-----------------------------|---------------------------------------------------------|--------------------|
| <a href="#">1.</a>          | Introduction . . . . .                                  | <a href="#">2</a>  |
| <a href="#">1.1.</a>        | Requirements Language . . . . .                         | <a href="#">3</a>  |
| <a href="#">1.2.</a>        | Notation . . . . .                                      | <a href="#">3</a>  |
| <a href="#">2.</a>          | Properties . . . . .                                    | <a href="#">4</a>  |
| <a href="#">3.</a>          | Assumptions . . . . .                                   | <a href="#">5</a>  |
| <a href="#">4.</a>          | Cryptographic Primitives . . . . .                      | <a href="#">5</a>  |
| <a href="#">5.</a>          | Protocol Definition . . . . .                           | <a href="#">6</a>  |
| <a href="#">5.1.</a>        | Exchange Phase . . . . .                                | <a href="#">6</a>  |
| <a href="#">5.2.</a>        | Commit/Reveal Phase . . . . .                           | <a href="#">7</a>  |
| <a href="#">6.</a>          | IANA Considerations . . . . .                           | <a href="#">8</a>  |
| <a href="#">7.</a>          | Security Considerations . . . . .                       | <a href="#">8</a>  |
| <a href="#">8.</a>          | References . . . . .                                    | <a href="#">9</a>  |
| <a href="#">8.1.</a>        | Normative References . . . . .                          | <a href="#">9</a>  |
| <a href="#">8.2.</a>        | Informative References . . . . .                        | <a href="#">10</a> |
| <a href="#">Appendix A.</a> | Role-specific Elements . . . . .                        | <a href="#">10</a> |
| <a href="#">A.1.</a>        | ECC Role-specific Elements . . . . .                    | <a href="#">11</a> |
| <a href="#">A.1.1.</a>      | Role-specific Elements for NIST p256 . . . . .          | <a href="#">11</a> |
| <a href="#">A.1.2.</a>      | Role-specific Elements for NIST p384 . . . . .          | <a href="#">12</a> |
| <a href="#">A.1.3.</a>      | Role-specific Elements for NIST p521 . . . . .          | <a href="#">13</a> |
| <a href="#">A.1.4.</a>      | Role-specific Elements for brainpool p256r1 . . . . .   | <a href="#">15</a> |
| <a href="#">A.1.5.</a>      | Role-specific Elements for brainpool p384r1 . . . . .   | <a href="#">15</a> |
| <a href="#">A.1.6.</a>      | Role-specific Elements for brainpool p512r1 . . . . .   | <a href="#">16</a> |
| <a href="#">A.2.</a>        | FFC Role-specific Elements . . . . .                    | <a href="#">17</a> |
| <a href="#">A.2.1.</a>      | Role-specific Elements for 2048-bit FFC group . . . . . | <a href="#">18</a> |
| <a href="#">A.2.2.</a>      | Role-specific Elements for 3072-bit FFC group . . . . . | <a href="#">19</a> |
| <a href="#">A.2.3.</a>      | Role-specific Elements for 4096-bit FFC group . . . . . | <a href="#">21</a> |
| <a href="#">A.2.4.</a>      | Role-specific Elements for 8192-bit FFC group . . . . . | <a href="#">24</a> |
|                             | Author's Address . . . . .                              | <a href="#">30</a> |

**[1.](#) Introduction**

Many authenticated key exchange protocols allow for authentication using uncertified, or "raw", public keys. Usually these specifications-- e.g. [\[RFC7250\]](#) for TLS and [\[RFC7670\]](#) for IKEv2-- assume keys are exchanged in some out-of-band mechanism.

[\[RFC7250\]](#) further states that "the main security challenge [to using 'raw' public keys] is how to associate the public key with a specific entity. Without a secure binding between identifier and key, the protocol will be vulnerable to man-in-the- middle attacks."

The Public Key Exchange (PKEX) is designed to fill that gap: it establishes a secure binding between exchanged public keys and identifiers, it provides proof-of-possession of the exchanged public keys to each peer, and it enables the establishment of trust in

Harkins

Expires July 6, 2017

[Page 2]

public keys that can subsequently be used to facilitate authentication in other authentication and key exchange protocols.

### **1.1. Requirements Language**

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](#)].

### **1.2. Notation**

This memo describes a cryptographic exchange using sets of elements called groups. Groups can be either traditional finite field or can be based on elliptic curves. The public keys exchanged by PKEX are elements in a group. Elements in groups are denoted in upper-case and scalar values are denoted with lower-case. The generator of the group is  $G$ .

When both the initiator and responder use a similar, but unique, datum it is denoted by appending an "i" for initiator or "r" for responder, e.g. if each side needs an element  $C$  then the initiator's is  $C_i$  and the responder's is  $C_r$ .

During the exchange, one side will generate data and the other side will attempt to reconstruct it. The reconstructed data is "primed". That is, if the initiator generates  $C$  then when responder tries to reconstruct it, the responder will refer to it as  $C'$ . Data that is directly sent and received is not primed.

The following notation is used in this memo:

$$C = A + B$$

The "group operation" on two elements,  $A$  and  $B$ , that produces a third element,  $C$ . For finite field cryptography this is the modular multiplication, for elliptic curve cryptography this is point addition.

$$C = A - B$$

The "group operation" on element  $A$  and the inverse of element  $B$  to produce a third element,  $C$ . Inversion is defined such that the group operation on an element and its inverse results in the identity element, the value one (1) for finite field cryptography and the "point at infinity" for elliptic curve cryptography.

$$C = a * B$$

This denotes repeated application of the group operation to  $B$ -- i.e.  $B + B$ --  $(a - 1)$  times.



$a = H(b)$

A cryptographic hash function that takes data  $b$  of indeterminate length and returns a fixed sized digest  $a$ .

$a = F(B)$

A mapping function that takes an element and returns a scalar. For elliptic curve cryptography,  $F()$  returns the x-coordinate of the point  $B$ . For finite field cryptography,  $F()$  is the identity function.

$a = \text{KDF-b}(c, d)$

A key derivation function that derives an output key  $a$  of length  $b$  from an input key  $c$  and context  $d$ .

$a = \text{HMAC}(b, c)$

A keyed MAC function that produces a digest  $a$  using key  $b$  and text  $c$ .

$a \parallel b$

Concatenation of data  $a$  with data  $b$ .

$\{a\}_b[c]$

Authenticated-encryption of data  $a$  with key  $b$  and associated data (AD) that is authenticated but not encrypted  $c$ .

## 2. Properties

Subversion of PKEX involves an adversary being able to insert its own public key into the exchange without the exchange failing, resulting in one of the parties to the exchange believing the adversary's public key actually belongs to the protocol peer.

PKEX has the following properties:

- o An adversary is unable to subvert the exchange without knowing the password.
- o An adversary is unable to discover the password through passive attack.
- o The only information exposed by an active attack is whether a single guess of the password is correct or not.
- o Proof-of-possession of the private key is provided.
- o At the end of the protocol, either trust is established in the peer's public key and the public key is bound to the peer's identity, or the exchange fails.



### 3. Assumptions

Due to the nature of the exchange, only DSA ([\[DSS\]](#)) and ECDSA ([\[X9.62\]](#)) keys can be exchanged with PKEX.

PKEX requires fixed elements that are unique to the particular role in the protocol, an initiator-specific element and a responder-specific element. They need not be secret. It is assumed that both parties know the role-specific elements for the particular group in which their key pairs were derived. Techniques to generate role-specific elements, and generated elements for popular groups, are listed in [Appendix A.1](#) and [Appendix A.2](#).

The authenticated-encryption algorithm provides deterministic "key wrapping". To achieve this the AE scheme used in PKEX is AES-SIV as defined in [\[RFC5297\]](#).

The KDF provides for the generation of a cryptographically strong secret key from an "imperfect" source of randomness. To achieve this the KDF used in PKEX is the unsalted version of [\[RFC5869\]](#).

The keyed MAC function is HMAC per [\[RFC2104\]](#).

The following assumptions are made on PKEX:

- o Only the peers involved in the exchange know the password.
- o The peers' public keys are from the same group.
- o The discrete logarithms of the public role-specific elements are unknown, and determining them is computationally infeasible.

### 4. Cryptographic Primitives

HKDF and HMAC require an underlying hash function and AES-SIV requires a key length. To provide for consistent security the hash algorithm and key length depend on the group chosen to use with PKEX.

For ECC, the hash algorithm and key length depends on the size of the prime defining the curve,  $p$ :

- o SHA-256 and 256 bits: when  $\text{len}(p) \leq 256$
- o SHA-384 and 384 bits: when  $256 < \text{len}(p) \leq 384$
- o SHA-512 and 512 bits: when  $384 < \text{len}(p)$



For FFC, the hash algorithm depends on the prime,  $p$ , defining the finite field:

- o SHA-256 and 256 bits: when  $\text{len}(p) \leq 2048$
- o SHA-384 and 384 bits: when  $2048 < \text{len}(p) \leq 3072$
- o SHA-512 and 512 bits: when  $3072 < \text{len}(p)$

## 5. Protocol Definition

PKEX is a balanced PAKE. The identical version of the password is used by both parties.

PKEX consists of two phases: exchange and commit/reveal. It is described using the popular protocol participants, Alice (an initiator of PKEX), and Bob (a responder of PKEX).

We denote Alice's role-specific element  $a$   $P_i$  and Bob's as  $P_r$ . The password is  $pw$ . For simplicity, Alice's identity is "Alice" and Bob's identity is "Bob". Alice's public key she wants to share with Bob is  $A$  and her private key is  $a$ , while Bob's public key he wants to share with Alice is  $B$  and his private key is  $b$ .

### 5.1. Exchange Phase

The Exchange phase is essentially the SPAKE2 key exchange. The peers derive ephemeral public keys, encrypt, and exchange them. Each party hashes a concatenation of his or her identity and the password and operates on the role-specific element to obtain a secret encrypting element. The group operation is then performed with the ephemeral key and the secret encrypting element to produce an encrypted ephemeral key.

|                                                                                                                                                  |                                                                                                                       |
|--------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <p>Alice:</p> <p>-----</p> <p><math>x, X = x * G</math></p> <p><math>Q_a = H(\text{Alice}   pw) * P_i</math></p> <p><math>M = X + Q_a</math></p> | <p>Bob:</p> <p>----</p> <p><math>y, Y = y * G</math></p> <p><math>Q_r = H(\text{Bob}   pw) * P_r</math></p>           |
|                                                                                                                                                  | <p><math>M \text{ -----&gt;}</math></p>                                                                               |
|                                                                                                                                                  | <p><math>Q_a = H(\text{Alice}   pw) * P_i</math></p> <p><math>X' = M - Q_a</math></p> <p><math>N = Y + Q_r</math></p> |
|                                                                                                                                                  | <p><math>&lt;----- N</math></p>                                                                                       |
| <p><math>Q_r = H(\text{Bob}   pw) * P_r</math></p> <p><math>Y' = N - Q_r</math></p>                                                              |                                                                                                                       |



Both M and N MUST be verified to be valid elements in the selected group. If either one is not valid the protocol fails.

At this point the peers have exchanged ephemeral elements that will be unknown except by someone with knowledge of the password. Given our assumptions that means only Alice and Bob can know the elements X and Y.

The secret encrypting elements Qa and Qb are irretrievably deleted at this point.

## **5.2. Commit/Reveal Phase**

In the Commit/Reveal phase the peers commit to the particular public key they wish to exchange and reveal it to the peer. Proof-of-possession of the private key is accomplished by "signing" the public key, the identity to which the public key is bound, the recipient's ephemeral public key, and the sender's ephemeral public key.

The messages exchanged in the Commit/Reveal phase are encrypted and authenticated with AES-SIV using a key derived from the SPAKE2 key exchange in [Section 5.1](#). Successful construction and validation of these messages authenticates the SPAKE2 exchange by proving possession of the SPAKE2 shared secret and therefore knowledge of the password. A single octet of the value zero (0) is used as AD when encrypting Alice's message to Bob and a single octet of the value one (1) is used as AD when constructing Bob's response. The AD is not transferred as part of the either message.



```

Alice:                                     Bob:
-----                                     ----
z = KDF-n(F(x*Y'), Alice | Bob |
          F(M) | F(N) | pw)
u = HMAC(F(a*Y'), Alice | F(A) |
          F(Y') | F(X))

          {A, u}z[0] ----->

                                z = KDF-n(F(y*X'), Alice | Bob |
                                F(M) | F(N) | pw)
                                if (SIV-decrypt returns fail) fail
                                if (A not valid element) fail
                                u' = HMAC(F(y*A), Alice | F(A) |
                                F(Y) | F(X'))
                                if (u' != u) fail
                                v = HMAC(F(b*X'), Bob | F(B) |
                                F(X') | F(Y))

                                <----- {B, v}z[1]

                                if (SIV-decrypt returns fail) fail
                                if (B not valid element) fail
                                v' = HMAC(F(x*B), Bob | F(B) |
                                F(X) | F(Y))
                                if (v' != v) fail

```

where 0 and 1 are single octets of the value zero and one, respectively, n is the key length from [Section 4](#), and both the KDF and HMAC use the hash algorithm from [Section 4](#).

If the parties didn't fail they have each other's public key, knowledge that the peer possesses the corresponding private key, and trust that the public key belongs to the peer's stated identity.

## 6. IANA Considerations

This memo could create a registry of the fixed public elements for a nice cross section of popular groups. Or not. Once published this document will be a stable reference and a registry might not be needed.

## 7. Security Considerations

The encrypted shares exchanged in the Exchange phase MUST be ephemeral. Reuse of these keys, even with a different password, voids the security of the exchange.



If fixed elements other than those in [Appendix A.1](#) and [Appendix A.2](#) are used, their discrete logarithm MUST not be known. Knowledge of the discrete logarithm of either of the fixed elements voids the security of the exchange.

The public keys exchanged in PKEX are never disclosed to an attacker, either passive or active. While they are, as the name implies, public, PKEX provides for secrecy of the exchanged keys for any protocol that might need such a capability.

PKEX has forward secrecy in the sense that exposure of the password used in a previous run of the protocol will not affect the security of that run. This also means that once PKEX has finished, the password can be exposed to a third party with out loss of security-- the public keys exchanged are still trusted and still bound to the entities that performed the exchange originally.

There is no proof of security of PKEX at this time but the Exchange phase is SPAKE2 and the security proof for that protocol can be used to help prove the security of PKEX.

## **[8.](#) References**

### **[8.1.](#) Normative References**

- [DSS] U.S. Department of Commerce/National Institute of Standards and Technology, "Digital Signature Standard (DSS)", Federal Information Processing Standards FIPS PUB 186-4, July 2013.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", [RFC 2104](#), DOI 10.17487/RFC2104, February 1997, <<http://www.rfc-editor.org/info/rfc2104>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", [RFC 3526](#), DOI 10.17487/RFC3526, May 2003, <<http://www.rfc-editor.org/info/rfc3526>>.
- [RFC5297] Harkins, D., "Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)", [RFC 5297](#), DOI 10.17487/RFC5297, October 2008, <<http://www.rfc-editor.org/info/rfc5297>>.



- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", [RFC 5869](#), DOI 10.17487/RFC5869, May 2010, <<http://www.rfc-editor.org/info/rfc5869>>.
- [X9.62] American National Standards Institute, "X9.62-2005", Public Key Cryptography for the Financial Services Industry (ECDSA), 2005.

## **8.2. Informative References**

- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", [RFC 7250](#), DOI 10.17487/RFC7250, June 2014, <<http://www.rfc-editor.org/info/rfc7250>>.
- [RFC7670] Kivinen, T., Wouters, P., and H. Tschofenig, "Generic Raw Public-Key Support for IKEv2", [RFC 7670](#), DOI 10.17487/RFC7670, January 2016, <<http://www.rfc-editor.org/info/rfc7670>>.

## **Appendix A. Role-specific Elements**

Role-specific elements for six popular elliptic curves and four popular modp groups from [[RFC3526](#)] have been generated using the following technique.

A loop is performed to generate role-specific elements by generating a candidate, testing the candidate, and exiting the loop once the test succeeds. A single octet counter is incremented each time through the loop (first time through the loop, the counter is one).

To find a candidate, a hash of an identifier (the concatenation of the ASN.1 of the OID of the curve or the name of the FFC group), a constant string, and the counter is produced. If the length of the hash's digest is less than the desired bits, the digest is pre-pended to the inputs and the result is fed back into the hash (this time it is a hash of a concatenation of the old digest, asn.1, constant string, counter) to produce the next length-of-digest bits. This is repeated until the number of bits has been produced. Excess octets are stripped off. The resulting string is interpreted as an integer with the first octet of (the first) hash being the high-order octet of the integer. For curves whose prime is not an integral number of octets, the bitstring is right-shifted, pre-pending with zero bits, in order to make a big-endian bitstring of the appropriate length. If that resulting number is larger than the prime defining the group the counter is incremented and the loop continues. Once an candidate



has been produced it is checked to see whether it is a valid element in the group (for ECC it is treated as an x-coordinate and checked whether it produces a valid point on the curve, for FFC it is checked by seeing whether the exponentiation of the candidate to the power of the prime minus one divided by the order, modulo the prime, is one). If it is not, the counter is incremented and the whole loop is performed again. This process is repeated until an element is found. The hash algorithm used to generate candidates is determined by [Section 4](#).

The loop is performed twice for each elliptic curve and FFC group to produce initiator- and responder-specific elements. The string passed for the initiator-specific element is "PKEX Initiator", the string passed for the responder-specific element is "PKEX Responder".

For FFC groups, the identifier is "group X" (including the space character and excluding the quotation marks) where X is the id assigned to the group, e.g. the 2048-bit group is named "group 14". For ECC groups, the identifier is the DER-encoded ASN.1 representation of the OID of the curve.

#### **[A.1](#). ECC Role-specific Elements**

##### **[A.1.1](#). Role-specific Elements for NIST p256**



```
unsigned char nist_p256_initiator_x_coord[32] = {
0x56, 0x26, 0x12, 0xcf, 0x36, 0x48, 0xfe, 0x0b,
0x07, 0x04, 0xbb, 0x12, 0x22, 0x50, 0xb2, 0x54,
0xb1, 0x94, 0x64, 0x7e, 0x54, 0xce, 0x08, 0x07,
0x2e, 0xec, 0xca, 0x74, 0x5b, 0x61, 0x2d, 0x25
};
unsigned char nist_p256_initiator_y_coord[32] = {
0x3e, 0x44, 0xc7, 0xc9, 0x8c, 0x1c, 0xa1, 0x0b,
0x20, 0x09, 0x93, 0xb2, 0xfd, 0xe5, 0x69, 0xdc,
0x75, 0xbc, 0xad, 0x33, 0xc1, 0xe7, 0xc6, 0x45,
0x4d, 0x10, 0x1e, 0x6a, 0x3d, 0x84, 0x3c, 0xa4
};

unsigned char nist_p256_responder_x_coord[32] = {
0x1e, 0xa4, 0x8a, 0xb1, 0xa4, 0xe8, 0x42, 0x39,
0xad, 0x73, 0x07, 0xf2, 0x34, 0xdf, 0x57, 0x4f,
0xc0, 0x9d, 0x54, 0xbe, 0x36, 0x1b, 0x31, 0x0f,
0x59, 0x91, 0x52, 0x33, 0xac, 0x19, 0x9d, 0x76
};
unsigned char nist_p256_responder_y_coord[32] = {
0x26, 0x04, 0x09, 0x45, 0x0a, 0x05, 0x20, 0xe7,
0xa7, 0x27, 0xc1, 0x36, 0x76, 0x85, 0xca, 0x3e,
0x42, 0x16, 0xf4, 0x89, 0x85, 0x34, 0x6e, 0xd5,
0x17, 0xde, 0xc0, 0xb8, 0xad, 0xfd, 0xb2, 0x98
};
```

#### [A.1.2.](#) Role-specific Elements for NIST p384



```
unsigned char nist_p384_initiator_x_coord[48] = {
0x95, 0x3f, 0x42, 0x9e, 0x50, 0x7f, 0xf9, 0xaa,
0xac, 0x1a, 0xf2, 0x85, 0x2e, 0x64, 0x91, 0x68,
0x64, 0xc4, 0x3c, 0xb7, 0x5c, 0xf8, 0xc9, 0x53,
0x6e, 0x58, 0x4c, 0x7f, 0xc4, 0x64, 0x61, 0xac,
0x51, 0x8a, 0x6f, 0xfe, 0xab, 0x74, 0xe6, 0x12,
0x81, 0xac, 0x38, 0x5d, 0x41, 0xe6, 0xb9, 0xa3
};
unsigned char nist_p384_initiator_y_coord[48] = {
0x89, 0xd0, 0x97, 0x7b, 0x59, 0x4f, 0xa6, 0xd6,
0x7c, 0x5d, 0x93, 0x5b, 0x93, 0xc4, 0x07, 0xa9,
0x89, 0xee, 0xd5, 0xcd, 0x6f, 0x42, 0xf8, 0x38,
0xc8, 0xc6, 0x62, 0x24, 0x69, 0x0c, 0xd4, 0x48,
0xd8, 0x44, 0xd6, 0xc2, 0xe8, 0xcc, 0x62, 0x6b,
0x3c, 0x25, 0x53, 0xba, 0x4f, 0x71, 0xf8, 0xe7
};

unsigned char nist_p384_responder_x_coord[48] = {
0xad, 0xbe, 0xd7, 0x1d, 0x3a, 0x71, 0x64, 0x98,
0x5f, 0xb4, 0xd6, 0x4b, 0x50, 0xd0, 0x84, 0x97,
0x4b, 0x7e, 0x57, 0x70, 0xd2, 0xd9, 0xf4, 0x92,
0x2a, 0x3f, 0xce, 0x99, 0xc5, 0x77, 0x33, 0x44,
0x14, 0x56, 0x92, 0xcb, 0xae, 0x46, 0x64, 0xdf,
0xe0, 0xbb, 0xd7, 0xb1, 0x29, 0x20, 0x72, 0xdf
};
unsigned char nist_p384_responder_y_coord[48] = {
0x54, 0x58, 0x20, 0xad, 0x55, 0x1d, 0xca, 0xf3,
0x1c, 0x8a, 0xcd, 0x19, 0x40, 0xf9, 0x37, 0x83,
0xc7, 0xd6, 0xb3, 0x13, 0x7d, 0x53, 0x28, 0x5c,
0xf6, 0x2d, 0xf1, 0xdd, 0xa5, 0x8b, 0xad, 0x5d,
0x81, 0xab, 0xb1, 0x00, 0x39, 0xd6, 0xcc, 0x9c,
0xea, 0x1e, 0x84, 0x1d, 0xbf, 0xe3, 0x35, 0xf9
};
```

#### [A.1.3.](#) Role-specific Elements for NIST p521



```
unsigned char nist_p521_initiator_x_coord[66] = {
0x00, 0x16, 0x20, 0x45, 0x19, 0x50, 0x95, 0x23,
0x0d, 0x24, 0xbe, 0x00, 0x87, 0xdc, 0xfa, 0xf0,
0x58, 0x9a, 0x01, 0x60, 0x07, 0x7a, 0xca, 0x76,
0x01, 0xab, 0x2d, 0x5a, 0x46, 0xcd, 0x2c, 0xb5,
0x11, 0x9a, 0xff, 0xaa, 0x48, 0x04, 0x91, 0x38,
0xcf, 0x86, 0xfc, 0xa4, 0xa5, 0x0f, 0x47, 0x01,
0x80, 0x1b, 0x30, 0xa3, 0xae, 0xe8, 0x1c, 0x2e,
0xea, 0xcc, 0xf0, 0x03, 0x9f, 0x77, 0x4c, 0x8d,
0x97, 0x76
```

```
};
```

```
unsigned char nist_p521_initiator_y_coord[66] = {
0x01, 0x4c, 0x71, 0xfd, 0x1b, 0xd5, 0x9c, 0xa6,
0xed, 0x39, 0xef, 0x45, 0xc5, 0x06, 0xfd, 0x66,
0xc0, 0xeb, 0x0f, 0xbf, 0x21, 0xa3, 0x36, 0x74,
0xfd, 0xaa, 0x05, 0x6e, 0x4e, 0x33, 0x95, 0x42,
0x1a, 0x9d, 0x3f, 0x3a, 0x1c, 0x5e, 0xa8, 0x60,
0xf7, 0xe5, 0x59, 0x1d, 0x07, 0xaa, 0x6f, 0x40,
0x0a, 0x59, 0x3c, 0x27, 0xad, 0xe0, 0x48, 0xfd,
0xd1, 0x83, 0x37, 0x4c, 0xdf, 0xe1, 0x86, 0x72,
0xfc, 0x57
```

```
};
```

```
unsigned char nist_p521_responder_x_coord[66] = {
0x00, 0x79, 0xe4, 0x4d, 0x6b, 0x5e, 0x12, 0x0a,
0x18, 0x2c, 0xb3, 0x05, 0x77, 0x0f, 0xc3, 0x44,
0x1a, 0xcd, 0x78, 0x46, 0x14, 0xee, 0x46, 0x3f,
0xab, 0xc9, 0x59, 0x7c, 0x85, 0xa0, 0xc2, 0xfb,
0x02, 0x32, 0x99, 0xde, 0x5d, 0xe1, 0x0d, 0x48,
0x2d, 0x71, 0x7d, 0x8d, 0x3f, 0x61, 0x67, 0x9e,
0x2b, 0x8b, 0x12, 0xde, 0x10, 0x21, 0x55, 0x0a,
0x5b, 0x2d, 0xe8, 0x05, 0x09, 0xf6, 0x20, 0x97,
0x84, 0xb4
```

```
};
```

```
unsigned char nist_p521_responder_y_coord[66] = {
0x01, 0xb9, 0x9c, 0xc6, 0x41, 0x32, 0x5b, 0xd2,
0x35, 0xd8, 0x8b, 0x2b, 0xe4, 0x6e, 0xcc, 0xdf,
0x7c, 0x38, 0xc4, 0x5b, 0xf6, 0x74, 0x71, 0x5c,
0x77, 0x16, 0x8a, 0x80, 0xa9, 0x84, 0xc7, 0x7b,
0x9d, 0xfd, 0x83, 0x6f, 0xae, 0xf8, 0x24, 0x16,
0x2f, 0x21, 0x25, 0x65, 0xa2, 0x1a, 0x6b, 0x2d,
0x30, 0x62, 0xb3, 0xcc, 0x6e, 0x59, 0x3c, 0x7f,
0x58, 0x91, 0x81, 0x72, 0x07, 0x8c, 0x91, 0xac,
0x31, 0x1e
```

```
};
```

Harkins

Expires July 6, 2017

[Page 14]

#### [A.1.4.](#) Role-specific Elements for brainpool p256r1

```
unsigned char brainpool_p256r1_initiator_x_coord[32] = {
0x46, 0x98, 0x18, 0x6c, 0x27, 0xcd, 0x4b, 0x10,
0x7d, 0x55, 0xa3, 0xdd, 0x89, 0x1f, 0x9f, 0xca,
0xc7, 0x42, 0x5b, 0x8a, 0x23, 0xed, 0xf8, 0x75,
0xac, 0xc7, 0xe9, 0x8d, 0xc2, 0x6f, 0xec, 0xd8
};

unsigned char brainpool_p256r1_initiator_y_coord[32] = {
0x16, 0x30, 0x68, 0x32, 0x3b, 0xb0, 0x21, 0xee,
0xeb, 0xf7, 0xb6, 0x7c, 0xae, 0x52, 0x26, 0x42,
0x59, 0x28, 0x58, 0xb6, 0x14, 0x90, 0xed, 0x69,
0xd0, 0x67, 0xea, 0x25, 0x60, 0x0f, 0xa9, 0x6c
};

unsigned char brainpool_p256r1_responder_x_coord[32] = {
0x90, 0x18, 0x84, 0xc9, 0xdc, 0xcc, 0xb5, 0x2f,
0x4a, 0x3f, 0x4f, 0x18, 0x0a, 0x22, 0x56, 0x6a,
0xa9, 0xef, 0xd4, 0xe6, 0xc3, 0x53, 0xc2, 0x1a,
0x23, 0x54, 0xdd, 0x08, 0x7e, 0x10, 0xd8, 0xe3
};

unsigned char brainpool_p256r1_responder_y_coord[32] = {
0x2a, 0xfa, 0x98, 0x9b, 0xe3, 0xda, 0x30, 0xfd,
0x32, 0x28, 0xcb, 0x66, 0xfb, 0x40, 0x7f, 0xf2,
0xb2, 0x25, 0x80, 0x82, 0x44, 0x85, 0x13, 0x7e,
0x4b, 0xb5, 0x06, 0xc0, 0x03, 0x69, 0x23, 0x64
};
```

#### [A.1.5.](#) Role-specific Elements for brainpool p384r1



```
unsigned char brainpool_p384r1_initiator_x_coord[48] = {
0x0a, 0x2c, 0xeb, 0x49, 0x5e, 0xb7, 0x23, 0xbd,
0x20, 0x5b, 0xe0, 0x49, 0xdf, 0xcf, 0xcf, 0x19,
0x37, 0x36, 0xe1, 0x2f, 0x59, 0xdb, 0x07, 0x06,
0xb5, 0xeb, 0x2d, 0xae, 0xc2, 0xb2, 0x38, 0x62,
0xa6, 0x73, 0x09, 0xa0, 0x6c, 0x0a, 0xa2, 0x30,
0x99, 0xeb, 0xf7, 0x1e, 0x47, 0xb9, 0x5e, 0xbe
};
unsigned char brainpool_p384r1_initiator_y_coord[48] = {
0x54, 0x76, 0x61, 0x65, 0x75, 0x5a, 0x2f, 0x99,
0x39, 0x73, 0xca, 0x6c, 0xf9, 0xf7, 0x12, 0x86,
0x54, 0xd5, 0xd4, 0xad, 0x45, 0x7b, 0xbf, 0x32,
0xee, 0x62, 0x8b, 0x9f, 0x52, 0xe8, 0xa0, 0xc9,
0xb7, 0x9d, 0xd1, 0x09, 0xb4, 0x79, 0x1c, 0x3e,
0x1a, 0xbf, 0x21, 0x45, 0x66, 0x6b, 0x02, 0x52
};

unsigned char brainpool_p384r1_responder_x_coord[48] = {
0x03, 0xa2, 0x57, 0xef, 0xe8, 0x51, 0x21, 0xa0,
0xc8, 0x9e, 0x21, 0x02, 0xb5, 0x9a, 0x36, 0x25,
0x74, 0x22, 0xd1, 0xf2, 0x1b, 0xa8, 0x9a, 0x9b,
0x97, 0xbc, 0x5a, 0xeb, 0x26, 0x15, 0x09, 0x71,
0x77, 0x59, 0xec, 0x8b, 0xb7, 0xe1, 0xe8, 0xce,
0x65, 0xb8, 0xaf, 0xf8, 0x80, 0xae, 0x74, 0x6c
};
unsigned char brainpool_p384r1_responder_y_coord[48] = {
0x2f, 0xd9, 0x6a, 0xc7, 0x3e, 0xec, 0x76, 0x65,
0x2d, 0x38, 0x7f, 0xec, 0x63, 0x26, 0x3f, 0x04,
0xd8, 0x4e, 0xff, 0xe1, 0x0a, 0x51, 0x74, 0x70,
0xe5, 0x46, 0x63, 0x7f, 0x5c, 0xc0, 0xd1, 0x7c,
0xfb, 0x2f, 0xea, 0xe2, 0xd8, 0x0f, 0x84, 0xcb,
0xe9, 0x39, 0x5c, 0x64, 0xfe, 0xcb, 0x2f, 0xf1
};
```

#### [A.1.6.](#) Role-specific Elements for brainpool p512r1



```
unsigned char brainpool_p512r1_initiator_x_coord[64] = {
0x4c, 0xe9, 0xb6, 0x1c, 0xe2, 0x00, 0x3c, 0x9c,
0xa9, 0xc8, 0x56, 0x52, 0xaf, 0x87, 0x3e, 0x51,
0x9c, 0xbb, 0x15, 0x31, 0x1e, 0xc1, 0x05, 0xfc,
0x7c, 0x77, 0xd7, 0x37, 0x61, 0x27, 0xd0, 0x95,
0x98, 0xee, 0x5d, 0xa4, 0x3d, 0x09, 0xdb, 0x3d,
0xfa, 0x89, 0x9e, 0x7f, 0xa6, 0xa6, 0x9c, 0xff,
0x83, 0x5c, 0x21, 0x6c, 0x3e, 0xf2, 0xfe, 0xdc,
0x63, 0xe4, 0xd1, 0x0e, 0x75, 0x45, 0x69, 0x0f
};
unsigned char brainpool_p512r1_initiator_y_coord[64] = {
0x5a, 0x28, 0x01, 0xbe, 0x96, 0x82, 0x4e, 0xf6,
0xfa, 0xed, 0x7d, 0xfd, 0x48, 0x8b, 0x48, 0x4e,
0xd1, 0x97, 0x87, 0xc4, 0x05, 0x5d, 0x15, 0x2a,
0xf4, 0x91, 0x4b, 0x75, 0x90, 0xd9, 0x34, 0x2c,
0x3c, 0x12, 0xf2, 0xf5, 0x25, 0x94, 0x24, 0x34,
0xa7, 0x6d, 0x66, 0xbc, 0x27, 0xa4, 0xa0, 0x8d,
0xd5, 0xe1, 0x54, 0xa3, 0x55, 0x26, 0xd4, 0x14,
0x17, 0x0f, 0xc1, 0xc7, 0x3d, 0x68, 0x7f, 0x5a
};
unsigned char brainpool_p512r1_responder_x_coord[64] = {
0x2a, 0x60, 0x32, 0x27, 0xa1, 0xe6, 0x94, 0x72,
0x1c, 0x48, 0xbe, 0xc5, 0x77, 0x14, 0x30, 0x76,
0xe4, 0xbf, 0xf7, 0x7b, 0xc5, 0xfd, 0xdf, 0x19,
0x1e, 0x0f, 0xdf, 0x1c, 0x40, 0xfa, 0x34, 0x9e,
0x1f, 0x42, 0x24, 0xa3, 0x2c, 0xd5, 0xc7, 0xc9,
0x7b, 0x47, 0x78, 0x96, 0xf1, 0x37, 0x0e, 0x88,
0xcb, 0xa6, 0x52, 0x29, 0xd7, 0xa8, 0x38, 0x29,
0x8e, 0x6e, 0x23, 0x47, 0xd4, 0x4b, 0x70, 0x3e
};
unsigned char brainpool_p512r1_responder_y_coord[64] = {
0x2a, 0xbe, 0x59, 0xe6, 0xc4, 0xb3, 0xd8, 0x09,
0x66, 0x89, 0x0a, 0x2d, 0x19, 0xf0, 0x9c, 0x9f,
0xb4, 0xab, 0x8f, 0x50, 0x68, 0x3c, 0x74, 0x64,
0x4e, 0x19, 0x55, 0x81, 0x9b, 0x48, 0x5c, 0xf4,
0x12, 0x8d, 0xb9, 0xd8, 0x02, 0x5b, 0xe1, 0x26,
0x7e, 0x19, 0x5c, 0xfd, 0x70, 0xf7, 0x4b, 0xdc,
0xb5, 0x5d, 0xc1, 0x7a, 0xe9, 0xd1, 0x05, 0x2e,
0xd1, 0xfd, 0x2f, 0xce, 0x63, 0x77, 0x48, 0x2c
};
```

## [A.2.](#) FFC Role-specific Elements



**A.2.1. Role-specific Elements for 2048-bit FFC group**

```

unsigned char group14_initiator[256] = {
0x97, 0x15, 0x52, 0x05, 0x89, 0xdf, 0xeb, 0x3d,
0xd6, 0x50, 0xe1, 0x96, 0xd4, 0x30, 0x81, 0x04,
0x3a, 0x4d, 0x6c, 0xae, 0xe5, 0x81, 0xb6, 0x1d,
0x53, 0xca, 0x65, 0xfa, 0x19, 0x59, 0xfd, 0xe4,
0xa4, 0xf0, 0x7b, 0xc5, 0xc7, 0xc4, 0xa9, 0xd6,
0xdf, 0x79, 0x54, 0x04, 0x5d, 0x64, 0xdc, 0x3c,
0xec, 0x0a, 0xa0, 0xd3, 0x2f, 0xef, 0xf3, 0xf5,
0x2c, 0x51, 0xe6, 0x6d, 0x1c, 0xdc, 0xac, 0x09,
0x3c, 0x00, 0x62, 0x41, 0xe7, 0x0b, 0x0d, 0xba,
0x1b, 0xf2, 0xb9, 0x22, 0xe9, 0x11, 0xea, 0xc7,
0xb1, 0xb2, 0x15, 0xc5, 0x19, 0x46, 0x2e, 0x15,
0x88, 0x41, 0xe0, 0x24, 0x16, 0x13, 0xc1, 0x0e,
0x27, 0xd3, 0x5f, 0x71, 0x12, 0xc6, 0x6f, 0x75,
0x1e, 0xe8, 0xa4, 0xaa, 0x57, 0xa3, 0x22, 0x32,
0xdc, 0xd4, 0xe3, 0xb5, 0xa3, 0xd0, 0x87, 0xb1,
0x3d, 0x1c, 0x2a, 0xcf, 0xe5, 0x87, 0x1f, 0xea,
0x98, 0xdd, 0xd6, 0x8f, 0xce, 0x0b, 0xdd, 0xba,
0xae, 0x85, 0x81, 0xd6, 0x89, 0x27, 0x71, 0xc9,
0x9f, 0xf9, 0xea, 0x5a, 0x89, 0xa6, 0xaa, 0x0a,
0xc2, 0x76, 0xd6, 0x6d, 0x89, 0xd3, 0xd2, 0x4c,
0xc0, 0xad, 0xb0, 0xf6, 0x4d, 0x2b, 0x7c, 0xbf,
0xd2, 0x4e, 0xe3, 0x2b, 0x4f, 0xd7, 0xc5, 0x3a,
0x4b, 0x1c, 0xc0, 0x17, 0xbe, 0x1e, 0x7b, 0x81,
0x1c, 0x77, 0xd4, 0xc2, 0xf6, 0xca, 0xb8, 0x51,
0xa8, 0x94, 0xe9, 0xe5, 0xe9, 0xa5, 0x46, 0x60,
0xb5, 0x36, 0x10, 0xcf, 0xc8, 0x2a, 0xe7, 0x7f,
0x94, 0x49, 0x96, 0xeb, 0xff, 0x7b, 0x62, 0xa9,
0x95, 0x35, 0x5c, 0xad, 0xcd, 0x52, 0x06, 0xa1,
0x9c, 0xa1, 0xb7, 0xe0, 0xcd, 0xd3, 0x13, 0x43,
0xee, 0xe7, 0xf1, 0xb1, 0x4d, 0x76, 0x51, 0x53,
0x24, 0x4f, 0xb4, 0xbd, 0x8b, 0x0c, 0x73, 0xd1,
0x6f, 0xf5, 0x27, 0x3f, 0xb9, 0x60, 0xa6, 0x17
}

```

```

unsigned char group14_responder[256] = {
0x3c, 0x86, 0x96, 0x8a, 0xb0, 0x4b, 0x14, 0xdd,
0x27, 0xf0, 0x6f, 0x51, 0xa2, 0xb2, 0xbd, 0x7d,
0x27, 0x34, 0xf2, 0x21, 0x3c, 0x6a, 0x63, 0xcf,
0x15, 0xd8, 0xeb, 0x21, 0xc2, 0x2d, 0xe5, 0x75,
0xbb, 0x7e, 0x09, 0x2e, 0xba, 0xa2, 0xd7, 0x04,
0x7c, 0x81, 0x82, 0x07, 0x09, 0x15, 0x28, 0x35,
0xf7, 0x0a, 0xd6, 0xa9, 0xaf, 0x0d, 0xb1, 0x97,
0x71, 0x76, 0x74, 0x66, 0xa3, 0x54, 0x9e, 0xd9,
0xa9, 0x13, 0xa8, 0xcf, 0xec, 0xce, 0x60, 0xd8,
0xea, 0x18, 0x67, 0xb7, 0x15, 0x17, 0xf7, 0xe6,

```



```

0x23, 0xc8, 0x30, 0x02, 0xb9, 0x9e, 0x0e, 0xe6,
0x64, 0x3e, 0x1a, 0x61, 0xb9, 0xbf, 0xd8, 0x7a,
0x0e, 0xbe, 0x1e, 0x58, 0xdc, 0xba, 0x9e, 0x31,
0xab, 0xcc, 0x6b, 0x03, 0xcc, 0x08, 0xf8, 0xa2,
0xcb, 0x9d, 0xd5, 0x1a, 0xb8, 0x6a, 0x1f, 0x4b,
0xab, 0xe8, 0x2a, 0x0c, 0x44, 0xde, 0x2a, 0xfd,
0x0f, 0x6d, 0x2f, 0xe4, 0xc4, 0x41, 0x61, 0xed,
0x4a, 0x85, 0x2a, 0x06, 0x9d, 0x3a, 0x27, 0xf0,
0x30, 0x6e, 0xf8, 0xb1, 0xc8, 0xde, 0x1f, 0xe0,
0xfb, 0xb6, 0xd0, 0x51, 0xee, 0x7d, 0x3a, 0x05,
0x0c, 0xbb, 0xa0, 0x41, 0xc5, 0x5d, 0x25, 0xb4,
0x48, 0xd6, 0x4f, 0x08, 0x85, 0x96, 0xa3, 0xa6,
0xf5, 0x1b, 0xa1, 0xb3, 0x13, 0x50, 0x06, 0xb2,
0xef, 0xf8, 0x2f, 0xe8, 0x7b, 0xe0, 0x5a, 0xe1,
0x42, 0x16, 0xfc, 0xdf, 0xad, 0x53, 0x95, 0x43,
0xb0, 0x73, 0x33, 0xa9, 0x08, 0x29, 0xcd, 0x6c,
0x14, 0x98, 0x5e, 0x98, 0xe6, 0xca, 0x92, 0x55,
0xd9, 0x3e, 0xc3, 0x51, 0x83, 0xda, 0x1e, 0x6d,
0x16, 0x88, 0x3f, 0xd1, 0xa5, 0xd1, 0xc5, 0x43,
0xcf, 0x8a, 0xd7, 0x29, 0xaa, 0xa6, 0x4f, 0x6b,
0x4f, 0xee, 0x36, 0x65, 0xd6, 0x71, 0xef, 0x71,
0xc5, 0x5b, 0x7c, 0x6d, 0x90, 0x9d, 0xf5, 0x74
}

```

#### **A.2.2. Role-specific Elements for 3072-bit FFC group**

```

unsigned char group15_initiator[384] = {
0x12, 0xce, 0x47, 0xcf, 0xa9, 0xc4, 0xfc, 0x5e,
0x99, 0xf6, 0xd2, 0x43, 0x5c, 0x60, 0x39, 0xb0,
0x06, 0xe9, 0x4a, 0xec, 0x21, 0x60, 0x9f, 0x5a,
0x25, 0xb1, 0x22, 0xf2, 0x53, 0x2b, 0x44, 0xe3,
0x6c, 0xfb, 0x9c, 0x46, 0x3c, 0x8f, 0x88, 0xaa,
0x60, 0xfd, 0x3a, 0x51, 0xf1, 0x19, 0x8d, 0x88,
0xee, 0xa4, 0xc2, 0x21, 0x3c, 0xbb, 0xc5, 0x53,
0x12, 0x16, 0xfb, 0xd3, 0xa9, 0x4d, 0x85, 0x5d,
0x17, 0x9d, 0x92, 0x15, 0x30, 0xc7, 0x97, 0x0c,
0x68, 0x62, 0x91, 0xff, 0xce, 0x81, 0x97, 0x25,
0x54, 0x94, 0x0e, 0x3a, 0x14, 0x36, 0x4e, 0xc2,
0xda, 0xc2, 0xaa, 0xa3, 0x58, 0x49, 0xca, 0xa4,
0xa4, 0x0b, 0x2a, 0x26, 0x35, 0x0d, 0x72, 0x4f,
0x10, 0x3c, 0x5f, 0x4d, 0xbc, 0x7c, 0x09, 0xcb,
0xef, 0x99, 0xdd, 0x73, 0x1c, 0x23, 0x69, 0xa7,
0xc9, 0xc4, 0x1a, 0x4c, 0x7c, 0xf2, 0xca, 0x48,
0x15, 0xf4, 0xd6, 0x30, 0x25, 0x44, 0x9f, 0xcd,
0xc0, 0x23, 0x72, 0x4a, 0x4f, 0x83, 0x3c, 0xba,
0x88, 0x1c, 0x5a, 0xcc, 0x3f, 0xf6, 0x5e, 0x68,
0x5a, 0x38, 0x10, 0xa1, 0xd2, 0x99, 0x5f, 0x4d,
0x48, 0xec, 0xb1, 0x2f, 0x9a, 0x08, 0xcf, 0x59,

```

Harkins

Expires July 6, 2017

[Page 19]

```

0x0f, 0xeb, 0x35, 0x0b, 0xf5, 0xab, 0x6e, 0xc9,
0x69, 0x44, 0xbd, 0x9a, 0x62, 0x3b, 0x53, 0x4e,
0x59, 0xc9, 0x38, 0x1d, 0x9e, 0x61, 0x5b, 0xdb,
0x72, 0x4e, 0xb9, 0x35, 0xb5, 0xc3, 0x9f, 0x47,
0x4f, 0x70, 0xfa, 0xff, 0x95, 0x45, 0xf9, 0x4a,
0xf4, 0xc3, 0xcc, 0x8e, 0xf4, 0x89, 0x0b, 0x73,
0x08, 0x97, 0x0d, 0x22, 0xe2, 0x97, 0xc8, 0xf8,
0x45, 0x83, 0x8f, 0xea, 0x68, 0x4b, 0xe0, 0xed,
0x71, 0xdb, 0x73, 0x62, 0x57, 0xab, 0x03, 0x69,
0x93, 0x66, 0x0e, 0xc1, 0x29, 0x2d, 0x01, 0x7c,
0x7d, 0x50, 0x14, 0x03, 0x9f, 0xdb, 0x5c, 0x4c,
0xf4, 0xdf, 0xa4, 0x9c, 0xda, 0x80, 0xd9, 0xbe,
0x0d, 0xdc, 0xb8, 0x0b, 0xc2, 0x19, 0x28, 0xaf,
0xf0, 0x98, 0x1a, 0xec, 0x26, 0xf0, 0x15, 0x1b,
0xa1, 0xa1, 0x11, 0x8f, 0x9f, 0x5a, 0x1e, 0x8a,
0x8e, 0x57, 0x84, 0x60, 0xc5, 0xda, 0xa2, 0x74,
0x3d, 0xe8, 0xc0, 0x08, 0x0f, 0x7e, 0xdd, 0x11,
0xd6, 0xbf, 0x5b, 0x2e, 0xde, 0x81, 0x1a, 0xfd,
0x33, 0x9c, 0x07, 0xcc, 0x1d, 0x0f, 0x63, 0xc8,
0x3e, 0x1d, 0xbb, 0x16, 0x5e, 0x70, 0x4c, 0x82,
0x8e, 0x72, 0xb6, 0x35, 0x69, 0xc6, 0xe4, 0xa7,
0xae, 0x6e, 0xa2, 0x23, 0xe8, 0x86, 0x99, 0x3a,
0x0b, 0x64, 0xec, 0xe2, 0xdb, 0xb2, 0xaa, 0xc4,
0x59, 0xe1, 0x23, 0x3d, 0xa5, 0x46, 0x92, 0x8f,
0x04, 0x34, 0x5f, 0x7a, 0x13, 0x55, 0x75, 0xd5,
0x6d, 0x0f, 0x5a, 0xc2, 0x0d, 0x16, 0xf9, 0xc0,
0xf3, 0xac, 0x0a, 0xa8, 0x62, 0x20, 0x09, 0x4e
}

```

```

unsigned char group15_responder[384] = {
0x9f, 0x17, 0xe0, 0xf9, 0x3d, 0x23, 0x36, 0x6e,
0x7d, 0xa6, 0x34, 0x75, 0xb7, 0xb4, 0x22, 0xb1,
0x87, 0x7a, 0x00, 0x4e, 0x02, 0x14, 0x4e, 0xe6,
0x96, 0xd4, 0x2c, 0x61, 0x00, 0x97, 0x7d, 0x99,
0xad, 0x18, 0x1c, 0xc4, 0x1b, 0xed, 0x6f, 0xd3,
0x9f, 0x85, 0xef, 0xfd, 0x1e, 0xcd, 0x13, 0xa0,
0x61, 0x2f, 0xf8, 0xa7, 0x11, 0xab, 0x83, 0xfc,
0xae, 0xad, 0xb6, 0xed, 0x6b, 0x7f, 0x34, 0x81,
0x30, 0xa2, 0x1c, 0x38, 0xb3, 0x31, 0x7b, 0x74,
0xc1, 0x0f, 0xf4, 0x29, 0x4b, 0xdd, 0x2b, 0x09,
0x32, 0xb7, 0x8f, 0x84, 0xab, 0x89, 0x9d, 0x64,
0xea, 0xec, 0x00, 0xa9, 0x0c, 0x82, 0x1c, 0x35,
0x75, 0x3b, 0x7f, 0x35, 0x28, 0xb8, 0xcc, 0xb1,
0x62, 0xc2, 0xd0, 0x74, 0x83, 0xb6, 0xf7, 0x76,
0x10, 0x2a, 0x7e, 0xce, 0xd0, 0x0a, 0x68, 0x46,
0xac, 0x78, 0x26, 0x90, 0xc7, 0x0e, 0xaa, 0x21,
0x59, 0xb1, 0x8d, 0x8e, 0xc8, 0xfb, 0x5c, 0x60,
0xec, 0x53, 0x9c, 0x40, 0xd9, 0x42, 0xab, 0xa3,

```



```

0xd8, 0x45, 0x81, 0x04, 0x6c, 0x13, 0x79, 0x66,
0x51, 0x1f, 0xa4, 0x81, 0x38, 0x0d, 0x48, 0x06,
0xef, 0x25, 0x89, 0x26, 0x5d, 0x09, 0x0e, 0xbe,
0xba, 0xe2, 0xca, 0x2c, 0xa6, 0x3e, 0x36, 0xd7,
0xef, 0x46, 0xe3, 0x8a, 0x1d, 0x85, 0x59, 0xc4,
0x89, 0x5e, 0x36, 0xea, 0xb2, 0x44, 0x79, 0xc8,
0x91, 0x80, 0x2c, 0x89, 0xfc, 0x55, 0x81, 0x62,
0x40, 0x82, 0x1a, 0x66, 0xf1, 0x1c, 0x10, 0xf0,
0x34, 0xdd, 0x52, 0x9f, 0xff, 0x63, 0x62, 0xe2,
0xec, 0x68, 0x75, 0xa0, 0x2e, 0x72, 0x44, 0xf3,
0x66, 0x33, 0x2a, 0x65, 0x61, 0x79, 0x91, 0x13,
0x4f, 0x73, 0x8f, 0x38, 0xef, 0xa6, 0x65, 0x88,
0xf9, 0x03, 0x35, 0x57, 0xed, 0xb9, 0x05, 0x7f,
0xe8, 0xfb, 0x90, 0xac, 0x19, 0x2e, 0xff, 0x9c,
0xaf, 0x76, 0x5f, 0x40, 0x74, 0x49, 0x11, 0xee,
0x18, 0xb5, 0x6e, 0xa4, 0x91, 0xbc, 0x42, 0x1c,
0x0e, 0x2e, 0x0d, 0x6f, 0xc3, 0x6a, 0x7b, 0x8b,
0xf4, 0x1a, 0x30, 0x52, 0x54, 0x99, 0xae, 0x8a,
0x8c, 0x33, 0x5e, 0x5c, 0xa8, 0xc2, 0x49, 0xf3,
0xbd, 0x0e, 0x85, 0x22, 0x9b, 0x5d, 0x92, 0xbc,
0x42, 0x8b, 0x60, 0x38, 0xd7, 0x14, 0x24, 0xaa,
0x36, 0xc7, 0x8f, 0xd7, 0xc2, 0x14, 0x20, 0x72,
0x0a, 0xba, 0x28, 0x95, 0x65, 0x53, 0x30, 0x0f,
0xc8, 0x17, 0xc2, 0x02, 0x58, 0x08, 0x7b, 0x93,
0x36, 0xcc, 0x80, 0x19, 0x9c, 0x1f, 0xad, 0x1f,
0x8e, 0x8e, 0x2e, 0x3a, 0xbf, 0x0f, 0xab, 0x76,
0x7a, 0xac, 0xce, 0x1a, 0x57, 0xe6, 0x7c, 0x64,
0x93, 0x5f, 0x92, 0x5d, 0xbe, 0xe2, 0x11, 0xf6,
0x58, 0x90, 0xd8, 0x87, 0xe4, 0x17, 0x8b, 0x61,
0xf6, 0x11, 0xe2, 0x0a, 0x99, 0xe8, 0x55, 0xcc
}

```

### [A.2.3.](#) Role-specific Elements for 4096-bit FFC group

```

unsigned char group16_initiator[512] = {
0x79, 0x8f, 0xfe, 0xed, 0x53, 0x08, 0x41, 0x73,
0xba, 0x89, 0x66, 0x8d, 0xf0, 0x18, 0xee, 0xe0,
0x76, 0xda, 0x5f, 0xf8, 0x55, 0x36, 0x53, 0x71,
0xd3, 0xfd, 0xf5, 0x30, 0xd9, 0xa0, 0xd2, 0x30,
0x2c, 0x16, 0x38, 0x0a, 0x2b, 0x91, 0x6e, 0x02,
0xc9, 0x27, 0x52, 0xf2, 0x51, 0x0e, 0xe3, 0x1f,
0xbb, 0x2b, 0x8b, 0xad, 0xa8, 0xc3, 0xf4, 0xc0,
0xba, 0x45, 0xf5, 0xf7, 0x4c, 0x91, 0x6f, 0x86,
0x9a, 0xb8, 0xb1, 0xea, 0x7d, 0x89, 0x91, 0x39,
0xd0, 0xb0, 0x95, 0x98, 0xf1, 0xa9, 0x03, 0x8d,
0xc5, 0x7a, 0x75, 0x36, 0x7a, 0xf8, 0x0b, 0xf6,
0xbf, 0x5c, 0x33, 0xde, 0x7f, 0xed, 0xec, 0x1a,
0x1e, 0xbc, 0x54, 0xf1, 0x5f, 0x5c, 0xfa, 0x2f,

```



0x98, 0x85, 0xe0, 0x6f, 0xb8, 0x2b, 0x16, 0xe9,  
0x48, 0x6e, 0xe4, 0xb3, 0x3f, 0x20, 0x66, 0x3b,  
0x3d, 0xcf, 0x62, 0xc6, 0xed, 0xe5, 0x2e, 0x7a,  
0xac, 0x0b, 0x15, 0x6d, 0x15, 0x4e, 0xcb, 0x23,  
0xcd, 0x5a, 0xed, 0x51, 0xf4, 0xbe, 0x52, 0x6b,  
0x55, 0x30, 0xe3, 0x57, 0x91, 0x1f, 0xf3, 0xbd,  
0x07, 0xd9, 0x8b, 0x32, 0x9d, 0xfb, 0x99, 0xbb,  
0x17, 0x81, 0x24, 0xd1, 0x82, 0x10, 0xce, 0x34,  
0x77, 0x4f, 0xbc, 0x4d, 0xe6, 0x23, 0xdf, 0x2c,  
0x24, 0x8d, 0xf5, 0xf5, 0xf9, 0x93, 0x3d, 0x08,  
0x55, 0x31, 0xc8, 0xe5, 0xf3, 0x5b, 0x4c, 0xe2,  
0x4a, 0xdf, 0x88, 0x83, 0xc7, 0x84, 0x1b, 0xfa,  
0x99, 0x72, 0x1b, 0x13, 0x9e, 0xf6, 0x76, 0xca,  
0xa9, 0xa2, 0x7f, 0xd8, 0xc0, 0x9a, 0x46, 0x15,  
0x0b, 0x20, 0x4e, 0x53, 0x09, 0xe5, 0x16, 0x67,  
0xaf, 0xe5, 0x07, 0xe9, 0x57, 0x2f, 0xdc, 0x38,  
0xde, 0x33, 0x19, 0x49, 0x08, 0x05, 0x6f, 0xb9,  
0xb0, 0xce, 0x97, 0xab, 0xb5, 0xac, 0x88, 0x4e,  
0x7a, 0xbe, 0xd0, 0xaa, 0x74, 0x67, 0x73, 0xe3,  
0xd5, 0x04, 0x18, 0x41, 0x51, 0xcd, 0xf5, 0x59,  
0x85, 0xb6, 0x4b, 0x63, 0x1f, 0x24, 0x27, 0x18,  
0x02, 0x1d, 0xb7, 0x95, 0x45, 0xae, 0x60, 0x3a,  
0xcc, 0xcb, 0xca, 0x6f, 0xb8, 0x6f, 0x1b, 0x95,  
0xba, 0x46, 0x76, 0x8d, 0xc3, 0x61, 0xd0, 0x86,  
0xc3, 0x6c, 0x9a, 0xd9, 0x33, 0xde, 0xce, 0x97,  
0x28, 0x69, 0xa3, 0xd4, 0xbc, 0x49, 0x6d, 0x10,  
0x8d, 0x31, 0x77, 0x53, 0x94, 0xe3, 0xc5, 0xf9,  
0xea, 0xcd, 0xb3, 0x5b, 0xbf, 0x51, 0x6f, 0x57,  
0x8b, 0x9f, 0x40, 0x8b, 0x47, 0xa8, 0x77, 0xbe,  
0x44, 0x3d, 0x8b, 0x54, 0x63, 0xc3, 0xbb, 0xdc,  
0xeb, 0x9e, 0xec, 0xf5, 0x7c, 0x31, 0x69, 0x45,  
0xf5, 0x48, 0x2c, 0x84, 0x57, 0x53, 0xe1, 0x20,  
0x25, 0x2b, 0x70, 0x3a, 0x4e, 0x9f, 0x36, 0xc3,  
0x16, 0xf8, 0x4f, 0xa3, 0x21, 0x66, 0x51, 0x49,  
0xc6, 0x6d, 0xac, 0xd0, 0x2a, 0x8e, 0xd4, 0x5d,  
0x55, 0x11, 0x1a, 0x31, 0x77, 0xdb, 0xa8, 0xf4,  
0x7a, 0xb9, 0x32, 0xe8, 0x54, 0xaf, 0xed, 0x55,  
0x6e, 0x87, 0xd0, 0x5c, 0x6d, 0xb9, 0x19, 0x0f,  
0xbf, 0x16, 0xa6, 0xc7, 0x7c, 0xa2, 0xd3, 0x95,  
0x34, 0xad, 0xfc, 0xc2, 0x0a, 0xfc, 0x23, 0x2f,  
0xc7, 0xe9, 0x98, 0x92, 0x90, 0x11, 0x1d, 0xd5,  
0xf5, 0xbc, 0xd2, 0x8f, 0x09, 0xb0, 0x43, 0x63,  
0x65, 0x51, 0x17, 0xfd, 0x11, 0xee, 0xd2, 0x23,  
0xff, 0x9f, 0x4e, 0x39, 0xda, 0xfe, 0x89, 0xc1,  
0x9d, 0xf3, 0xe0, 0x31, 0xdd, 0x74, 0x36, 0xb2,  
0x9f, 0xfa, 0xd2, 0xd0, 0xfc, 0xc4, 0x37, 0x83,  
0xf0, 0x13, 0x1f, 0x8d, 0x80, 0x47, 0x53, 0x93,  
0x0c, 0x84, 0x1f, 0x69, 0x08, 0x74, 0x33, 0x29,



```
0x59, 0x5f, 0x62, 0x7d, 0xe2, 0x59, 0x1c, 0x56,  
0x52, 0x75, 0xc2, 0x83, 0xad, 0xc0, 0xd4, 0x66,  
0x71, 0x6a, 0x1a, 0x61, 0x94, 0xa4, 0xa5, 0x73  
}
```

```
unsigned char group16_responder[512] = {  
0x06, 0x85, 0x8b, 0x2e, 0x0c, 0x05, 0xfd, 0x1b,  
0x1f, 0x93, 0xd3, 0xc2, 0xe6, 0x70, 0xc8, 0xe1,  
0x17, 0x39, 0xe6, 0x38, 0x75, 0xfd, 0xdd, 0xe6,  
0x4a, 0xfe, 0x95, 0x5e, 0xd6, 0x80, 0x17, 0x2c,  
0x1d, 0xbb, 0x8c, 0xf7, 0x2d, 0x9b, 0x17, 0x93,  
0x4d, 0x92, 0xd3, 0x57, 0xa0, 0xcd, 0x44, 0x37,  
0x1d, 0xdf, 0xd3, 0x80, 0x25, 0xa0, 0xa8, 0x51,  
0x13, 0xb9, 0x63, 0xec, 0xda, 0xa2, 0x8b, 0xdb,  
0x2e, 0x09, 0x9c, 0x93, 0x09, 0x02, 0x64, 0xb4,  
0xee, 0xa2, 0x3c, 0x75, 0x0e, 0xbb, 0x31, 0x44,  
0xff, 0xf0, 0x7e, 0x99, 0x86, 0x17, 0xe5, 0xc4,  
0xf9, 0x39, 0xe1, 0xec, 0xed, 0xd5, 0x13, 0xe9,  
0x97, 0xda, 0x2b, 0xb5, 0x1e, 0x23, 0x88, 0x1a,  
0xb5, 0x10, 0xda, 0x24, 0x05, 0xe7, 0xdf, 0xc3,  
0xc2, 0x24, 0xd2, 0xf4, 0x14, 0x6c, 0xfd, 0x2e,  
0x62, 0xa8, 0x00, 0x4e, 0xa9, 0x96, 0x3d, 0x48,  
0x4f, 0xcf, 0x62, 0xfe, 0x16, 0x06, 0x56, 0x81,  
0x1b, 0x58, 0xca, 0x84, 0xd2, 0x86, 0xad, 0xc6,  
0x66, 0xb4, 0x6f, 0x49, 0x91, 0x2a, 0xb5, 0x3b,  
0x39, 0xec, 0x88, 0xdf, 0x31, 0x24, 0x44, 0x04,  
0x30, 0x4e, 0x91, 0xdd, 0x1f, 0xf7, 0x62, 0x0c,  
0x8a, 0xf1, 0xcd, 0xf1, 0xcf, 0x56, 0x42, 0x1d,  
0x1b, 0xb2, 0x47, 0x7f, 0x4c, 0x82, 0x88, 0xbe,  
0x99, 0x31, 0x96, 0x6e, 0x5f, 0xa0, 0x6a, 0xa0,  
0x53, 0x63, 0xdb, 0xb9, 0xe0, 0xe4, 0x8f, 0xa9,  
0x44, 0x32, 0x2e, 0x05, 0x70, 0x3b, 0x6e, 0xc3,  
0x82, 0x36, 0x51, 0x4c, 0xbe, 0x38, 0x61, 0x54,  
0x66, 0x5c, 0x88, 0x42, 0x50, 0x84, 0x1a, 0x69,  
0xdf, 0xc5, 0x2b, 0x00, 0x3d, 0xdb, 0xe1, 0x92,  
0x69, 0xb4, 0xda, 0xfa, 0x87, 0x43, 0x9e, 0xdd,  
0x03, 0x29, 0xd4, 0x06, 0xef, 0x63, 0x60, 0xe3,  
0x83, 0xe3, 0x28, 0xd7, 0xa3, 0x47, 0xeb, 0xb7,  
0x0a, 0x20, 0x5a, 0x9e, 0x61, 0x68, 0xcc, 0x0b,  
0x39, 0xdc, 0x7b, 0x8c, 0x22, 0x0f, 0xc8, 0xd4,  
0x0c, 0x44, 0x9c, 0xa2, 0xb4, 0xd2, 0xf4, 0x71,  
0xeb, 0xc6, 0x75, 0xb8, 0x53, 0x8c, 0x93, 0x9a,  
0xf2, 0xd7, 0xba, 0x45, 0x40, 0xef, 0x56, 0xaf,  
0xdd, 0x1b, 0xcc, 0x0e, 0xe0, 0x3b, 0x2f, 0xd5,  
0xc5, 0xc7, 0x36, 0x37, 0xc0, 0x5e, 0xff, 0xb1,  
0x31, 0xce, 0xce, 0xc3, 0x28, 0xb7, 0x84, 0x88,  
0xe2, 0x7f, 0x11, 0x62, 0xb1, 0x14, 0x41, 0xe2,  
0x7e, 0xfb, 0x31, 0x0e, 0x5b, 0xba, 0x27, 0xf7,
```



```

0xa0, 0xce, 0xa0, 0xb8, 0xdd, 0xbe, 0xc4, 0x74,
0xe0, 0xc9, 0x50, 0x71, 0x16, 0x81, 0x42, 0x2a,
0xa9, 0xda, 0x2e, 0x3a, 0x85, 0x0b, 0x62, 0x5a,
0x55, 0x31, 0x31, 0xc4, 0xda, 0x4b, 0x36, 0x9a,
0xa6, 0x0b, 0x78, 0x51, 0x50, 0xea, 0x44, 0x07,
0x6d, 0xf7, 0x49, 0xb0, 0xea, 0x7e, 0x12, 0x92,
0x88, 0x5e, 0xb8, 0xee, 0x0b, 0xa9, 0xd8, 0x04,
0xbe, 0xd8, 0x5d, 0x8e, 0x0a, 0xea, 0x5c, 0xce,
0xf5, 0x2d, 0x80, 0xff, 0x57, 0x07, 0x0c, 0x06,
0x20, 0xaf, 0xb5, 0x32, 0x16, 0xb5, 0x79, 0x60,
0xce, 0x3b, 0xb8, 0x55, 0x4c, 0xf5, 0x58, 0x90,
0xeb, 0xae, 0x48, 0x04, 0x8b, 0x76, 0xfc, 0x66,
0x18, 0x70, 0x13, 0x5c, 0x85, 0xa4, 0x18, 0xf1,
0xbb, 0x06, 0x4e, 0x59, 0x19, 0x75, 0x4d, 0xaf,
0x97, 0x6b, 0x1c, 0x82, 0x2f, 0xbf, 0x91, 0xbb,
0xb7, 0x93, 0x21, 0xf2, 0xca, 0xf7, 0xec, 0x31,
0xf2, 0x05, 0x73, 0x1e, 0x69, 0x30, 0xd1, 0xd9,
0xef, 0x4f, 0x57, 0xc5, 0xf2, 0x46, 0xbf, 0xe8,
0x81, 0xb6, 0x24, 0xdb, 0xf3, 0x67, 0x27, 0x97,
0xb8, 0x04, 0x8c, 0xa3, 0xfe, 0xda, 0x6b, 0x9d,
0x51, 0xba, 0xbd, 0xb5, 0xca, 0xfe, 0x73, 0xce,
0x48, 0x9d, 0x28, 0x30, 0x7d, 0x53, 0x63, 0xc7
}

```

#### [A.2.4.](#) Role-specific Elements for 8192-bit FFC group

```

unsigned char group18_initiator[1024] = {
0x5b, 0x92, 0xc0, 0x76, 0x71, 0x48, 0x5e, 0x4d,
0x9d, 0xdf, 0xde, 0xc3, 0xcc, 0x32, 0x6e, 0xf1,
0x90, 0xef, 0x71, 0x86, 0xde, 0x55, 0x60, 0x3e,
0x24, 0x3e, 0xc2, 0x38, 0x09, 0x65, 0x56, 0xfb,
0x2b, 0x8b, 0x97, 0x07, 0x19, 0xac, 0xaf, 0x59,
0x32, 0xd7, 0x65, 0x19, 0xa2, 0x02, 0x83, 0xa3,
0x72, 0xc2, 0x2c, 0xe8, 0x15, 0x58, 0x0d, 0xb8,
0x1a, 0xad, 0xe3, 0xbc, 0xc0, 0x82, 0xab, 0x42,
0x9a, 0x7e, 0x0d, 0x39, 0x6a, 0x81, 0x3e, 0x72,
0xee, 0xba, 0x58, 0x01, 0xe6, 0x36, 0x9f, 0x82,
0x49, 0xa3, 0x6a, 0x88, 0x59, 0x0f, 0x77, 0x6b,
0x4d, 0x6a, 0x36, 0x25, 0xf0, 0xbc, 0x75, 0x53,
0x45, 0x1b, 0x02, 0x4c, 0x99, 0x5d, 0x51, 0x87,
0x41, 0x3c, 0xcc, 0x54, 0x6a, 0xdc, 0x6e, 0x22,
0xb5, 0x7d, 0xa9, 0x65, 0x90, 0xd3, 0x38, 0x4c,
0xa8, 0x26, 0x22, 0x57, 0xd8, 0x55, 0xc0, 0xc9,
0x9a, 0x62, 0x08, 0x71, 0x2b, 0x55, 0xde, 0x89,
0xd4, 0xf8, 0xab, 0x5f, 0x55, 0x42, 0xc0, 0x40,
0x60, 0x61, 0x1b, 0x68, 0x01, 0x80, 0x67, 0x27,
0x95, 0x8e, 0x6b, 0xcd, 0xc1, 0x04, 0x35, 0x96,
0x8e, 0x15, 0x1e, 0xd3, 0x01, 0x7d, 0x81, 0x38,

```



0xe8, 0xbf, 0xd1, 0xa8, 0x31, 0xd5, 0x49, 0x11,  
0x0d, 0x78, 0x2f, 0x61, 0x31, 0xfc, 0xcc, 0x11,  
0x4f, 0x09, 0xa1, 0x13, 0x2b, 0x0e, 0x73, 0xbc,  
0x1f, 0xef, 0x01, 0x7d, 0xd3, 0x26, 0xe3, 0xda,  
0xa9, 0xaa, 0x15, 0xaf, 0x47, 0xe5, 0xc0, 0x39,  
0x29, 0xa3, 0x68, 0xfc, 0x03, 0xaa, 0x20, 0xc0,  
0xf8, 0xc5, 0xe2, 0xe3, 0x03, 0x0d, 0x3c, 0x20,  
0x7c, 0xbf, 0xa5, 0x1b, 0xd9, 0x92, 0x2d, 0x79,  
0x42, 0x42, 0xe8, 0x92, 0x25, 0x9a, 0x94, 0x54,  
0x40, 0xec, 0x8d, 0x55, 0x26, 0x71, 0xb3, 0x58,  
0x2c, 0x0b, 0x81, 0x4d, 0x53, 0xb8, 0x52, 0xf9,  
0x1b, 0xb1, 0x75, 0x60, 0xd4, 0x4b, 0x45, 0x72,  
0xa6, 0x61, 0x20, 0x96, 0xaa, 0x3b, 0xb9, 0x50,  
0x81, 0xe3, 0x93, 0xde, 0x4b, 0x80, 0xa2, 0xbd,  
0x20, 0x64, 0x63, 0xe2, 0x48, 0xc8, 0xec, 0x82,  
0x07, 0xa1, 0x7b, 0x45, 0x2a, 0xfb, 0xe9, 0x2f,  
0xa1, 0xf0, 0x69, 0x36, 0x2d, 0x4f, 0x1a, 0x85,  
0xf3, 0x58, 0x34, 0xe6, 0x0a, 0x9e, 0xe9, 0x9a,  
0x77, 0xe5, 0xf9, 0xa4, 0xc4, 0x14, 0xa2, 0x43,  
0xdd, 0xaa, 0x03, 0x17, 0x71, 0x55, 0x62, 0xf4,  
0xf5, 0x9c, 0x5f, 0x2f, 0xe7, 0x6f, 0xde, 0xa4,  
0x7a, 0xbb, 0x9d, 0xb5, 0x8d, 0xc3, 0x95, 0xf9,  
0x54, 0x06, 0xba, 0xd1, 0x31, 0xcf, 0x03, 0x6c,  
0x7a, 0x53, 0xd5, 0x76, 0x97, 0x4c, 0xbd, 0x23,  
0x59, 0xff, 0xfe, 0xea, 0xd3, 0xd1, 0x86, 0x10,  
0x2c, 0xf9, 0x9f, 0xc8, 0xd3, 0x45, 0x44, 0x2f,  
0x5a, 0xcf, 0x86, 0x8e, 0x1c, 0xc2, 0xb7, 0x04,  
0x75, 0x74, 0x78, 0xdf, 0x7a, 0x6f, 0xaf, 0x56,  
0x03, 0x93, 0x19, 0x4f, 0x4d, 0x73, 0x11, 0xc9,  
0x34, 0x90, 0x1a, 0x76, 0x18, 0x76, 0xa7, 0x19,  
0xe4, 0x5e, 0x66, 0x10, 0x2e, 0x0a, 0xbe, 0x7c,  
0x64, 0xd2, 0xd3, 0xbb, 0x18, 0x58, 0x86, 0xd9,  
0x54, 0x58, 0xf5, 0xeb, 0x86, 0xac, 0x61, 0x48,  
0xbc, 0x95, 0x1e, 0x13, 0xab, 0xef, 0x6e, 0xdf,  
0xbc, 0xa5, 0x78, 0x10, 0x87, 0x43, 0x9a, 0xd6,  
0xd6, 0x10, 0x30, 0xc0, 0xf5, 0x9b, 0x09, 0xc4,  
0x2c, 0xed, 0x8b, 0xeb, 0xc7, 0x3c, 0x12, 0xc5,  
0x1c, 0xf1, 0x88, 0xfd, 0x15, 0x45, 0xdb, 0xb3,  
0x35, 0x87, 0x40, 0xf8, 0x8a, 0xd1, 0x07, 0x32,  
0x2b, 0xf7, 0x4a, 0x77, 0xb5, 0x69, 0x4a, 0x20,  
0xdd, 0x69, 0x1e, 0x38, 0xac, 0x0b, 0x31, 0xda,  
0x43, 0x5d, 0xf0, 0x94, 0x22, 0x8d, 0x4a, 0x26,  
0xda, 0x91, 0xdf, 0xb7, 0xdd, 0xfb, 0x97, 0x88,  
0x7a, 0x43, 0x5e, 0xf3, 0x36, 0xbd, 0xef, 0xc0,  
0xe6, 0x7f, 0xd1, 0x81, 0x5b, 0xd6, 0x1b, 0x01,  
0x89, 0x19, 0x1d, 0x0e, 0xd0, 0x1a, 0x3a, 0x56,  
0x82, 0xf6, 0x2c, 0xdf, 0x6a, 0x42, 0xf5, 0x44,  
0x57, 0x61, 0x95, 0xdc, 0x9d, 0x4c, 0x15, 0xc0,



0x29, 0x42, 0x55, 0x77, 0x28, 0xc8, 0x7c, 0xe2,  
0xc8, 0x44, 0xbd, 0xdd, 0x8e, 0xe1, 0xb8, 0xa3,  
0xa6, 0xb1, 0xa5, 0xfc, 0x9f, 0xed, 0x5f, 0xd7,  
0x58, 0xee, 0xe9, 0xa8, 0x1e, 0x11, 0x1a, 0x8c,  
0xf6, 0xea, 0x45, 0x8f, 0x41, 0x4d, 0xb9, 0x7f,  
0xe9, 0xd2, 0x90, 0x8c, 0xed, 0x0a, 0xd1, 0x12,  
0x48, 0x9b, 0x5e, 0x8a, 0x98, 0xc8, 0x0e, 0x71,  
0xff, 0x35, 0xa3, 0xc0, 0x17, 0x1c, 0x29, 0xe7,  
0x30, 0x47, 0x5f, 0x22, 0x62, 0x1b, 0xde, 0xb9,  
0xeb, 0x20, 0xcc, 0xeb, 0xfc, 0x7d, 0x38, 0x1e,  
0xce, 0x8a, 0x4e, 0xa8, 0xfe, 0xba, 0xa9, 0xfc,  
0x44, 0x8a, 0xcf, 0x0a, 0xe3, 0xf1, 0x91, 0x63,  
0xaf, 0xf2, 0x7d, 0x52, 0x2a, 0x6d, 0x38, 0xcf,  
0x10, 0xd5, 0xa3, 0xa1, 0xb0, 0xcc, 0x74, 0x08,  
0xe9, 0x97, 0xe1, 0x7e, 0xd8, 0xd1, 0x3f, 0x4e,  
0x8d, 0x6d, 0x4e, 0x3e, 0x33, 0xc7, 0xae, 0x28,  
0xb6, 0x6a, 0xd0, 0x15, 0xf3, 0xd6, 0xfd, 0x11,  
0x3e, 0xbc, 0x65, 0xe6, 0xf7, 0xb4, 0xfe, 0x55,  
0x03, 0x4d, 0x1f, 0x4f, 0x8b, 0xef, 0x8d, 0x11,  
0xa2, 0x9b, 0x42, 0x58, 0xf4, 0xdf, 0x0c, 0xf2,  
0x80, 0x0e, 0x02, 0xff, 0xe8, 0x46, 0x0d, 0xae,  
0x50, 0x41, 0x14, 0x37, 0x5d, 0x82, 0x26, 0x96,  
0x9f, 0x1d, 0xff, 0x9e, 0xe0, 0x01, 0x24, 0x19,  
0xe8, 0xca, 0xe7, 0x7b, 0xaa, 0xef, 0x05, 0x4a,  
0x8a, 0xdd, 0x3b, 0xe8, 0x93, 0xf0, 0x21, 0xab,  
0x7f, 0x77, 0xcd, 0xc1, 0x71, 0x9f, 0x6b, 0x2b,  
0x64, 0xfb, 0x43, 0x9e, 0x92, 0x33, 0x68, 0xe6,  
0x51, 0xc1, 0x16, 0x3d, 0xde, 0xf8, 0x85, 0x8a,  
0xb6, 0x6c, 0x96, 0x7a, 0x6b, 0x12, 0xd8, 0x18,  
0x30, 0x5e, 0x0c, 0x82, 0xff, 0xff, 0xd0, 0xf7,  
0x9c, 0x23, 0x30, 0x61, 0x80, 0x5b, 0xde, 0xd9,  
0x35, 0x16, 0xdd, 0x6a, 0x5b, 0xbe, 0x5a, 0x1d,  
0x77, 0x37, 0x2b, 0xee, 0x00, 0x61, 0xef, 0xfe,  
0xa8, 0x3f, 0x2e, 0xd6, 0x8e, 0x3a, 0x0f, 0x03,  
0xae, 0x46, 0x01, 0x0c, 0x75, 0x8c, 0x42, 0x9b,  
0x24, 0x02, 0x4b, 0xdb, 0xb8, 0x98, 0x31, 0xc7,  
0xd0, 0xd9, 0xb1, 0x89, 0x4a, 0x54, 0x74, 0xc0,  
0x68, 0x5d, 0xe7, 0x62, 0xe4, 0x44, 0x44, 0x5e,  
0x17, 0x75, 0x34, 0x96, 0xbe, 0xf1, 0x94, 0x49,  
0x14, 0xe7, 0x17, 0x79, 0xf6, 0xab, 0xe8, 0xf4,  
0x47, 0x77, 0x74, 0x10, 0x51, 0x3b, 0x30, 0x8e,  
0x8e, 0x00, 0x4f, 0x0e, 0x75, 0x03, 0xc7, 0x48,  
0xfb, 0xf8, 0x50, 0xc7, 0xe5, 0xfc, 0xe2, 0x7d,  
0x07, 0x90, 0xd5, 0x91, 0x6a, 0xce, 0x14, 0x12,  
0xab, 0xe6, 0x65, 0x64, 0xfb, 0x03, 0xce, 0xdf,  
0xf1, 0x0b, 0x11, 0x82, 0x5a, 0x11, 0xf6, 0xf9,  
0xd6, 0xf2, 0xfe, 0xd4, 0x72, 0x60, 0x80, 0xd4,  
0x53, 0x86, 0xe6, 0xfc, 0xb7, 0xc0, 0x03, 0x4d,



```
0x3c, 0x32, 0xe9, 0xfd, 0x46, 0x9e, 0x81, 0x6a,  
0x72, 0xd6, 0x9c, 0x14, 0x70, 0x47, 0xbe, 0x35,  
0xef, 0xb2, 0xbb, 0x0a, 0xca, 0x84, 0xc9, 0x15,  
0xac, 0x83, 0x6b, 0x83, 0xe8, 0x36, 0x5d, 0x27,  
0xc0, 0x25, 0xc8, 0x92, 0x69, 0x6b, 0x51, 0x6a,  
0xc2, 0x8f, 0x9c, 0x8b, 0xf5, 0x35, 0x1e, 0x31,  
0x4b, 0xf7, 0xd6, 0x40, 0xa7, 0x1c, 0xe4, 0xa2,  
0x00, 0x3c, 0x78, 0x78, 0x8f, 0x27, 0x24, 0x7e,  
0x0b, 0x7f, 0xf6, 0xb0, 0x66, 0xf4, 0x79, 0x46,  
0x2e, 0x5b, 0x11, 0xca, 0x9b, 0x93, 0x4e, 0x99,  
0x2c, 0xd2, 0x2d, 0x88, 0x84, 0xf1, 0x1e, 0x0e  
}
```

```
unsigned char group18_responder[1024] = {  
0xdf, 0x63, 0xdc, 0x3a, 0x74, 0xbd, 0x83, 0x8e,  
0x72, 0x90, 0x81, 0xf0, 0x85, 0x65, 0x86, 0x07,  
0x06, 0x05, 0xed, 0x93, 0x73, 0xa5, 0xbb, 0x88,  
0x42, 0x17, 0x8a, 0x10, 0x33, 0xc3, 0x6e, 0x9b,  
0x3b, 0x68, 0x33, 0x30, 0xa2, 0x3f, 0xb2, 0xa3,  
0x2d, 0x6b, 0xec, 0x34, 0x5a, 0xe7, 0x6a, 0xc1,  
0x11, 0xc8, 0xa5, 0x3d, 0x6c, 0xc5, 0xf7, 0x48,  
0xca, 0xac, 0xa6, 0x10, 0x0b, 0x7c, 0x93, 0xe9,  
0x45, 0x4d, 0xa7, 0x00, 0x30, 0xd2, 0xf5, 0xaf,  
0x93, 0xf1, 0xa5, 0x8a, 0x9f, 0x10, 0x14, 0xe6,  
0x6a, 0xe0, 0x5a, 0xe6, 0xea, 0x8b, 0x21, 0xbb,  
0x6a, 0x1f, 0x6c, 0x8c, 0x0b, 0x01, 0xda, 0xfd,  
0x4f, 0x0b, 0x7f, 0xe9, 0x46, 0x27, 0x8e, 0xaa,  
0x64, 0xd1, 0xd5, 0x40, 0xc9, 0xf7, 0x47, 0xef,  
0x9f, 0x7c, 0xce, 0x6c, 0x41, 0xd2, 0x9c, 0x47,  
0x09, 0x6e, 0xc7, 0xc2, 0xdc, 0x7d, 0x7e, 0xce,  
0x04, 0x6f, 0xf9, 0xc1, 0x86, 0x56, 0x1b, 0x88,  
0x7f, 0x62, 0x33, 0x3b, 0xca, 0xb9, 0xd4, 0x7d,  
0x24, 0xfa, 0x9f, 0xd8, 0xf8, 0x63, 0x91, 0x72,  
0x45, 0x82, 0x4d, 0x9f, 0xd7, 0x9d, 0xc8, 0x0b,  
0x4c, 0x6a, 0xc5, 0xf4, 0xec, 0x77, 0x3e, 0xfd,  
0xb7, 0x6b, 0xe0, 0x86, 0x32, 0x41, 0x25, 0xfe,  
0x43, 0x03, 0x0a, 0x07, 0x90, 0x75, 0xd5, 0xca,  
0x48, 0x23, 0xfb, 0x80, 0x5f, 0x22, 0xfe, 0x1c,  
0xba, 0x48, 0x2c, 0x28, 0x78, 0x5c, 0xc4, 0x98,  
0xad, 0xb7, 0xa6, 0x78, 0x5f, 0x84, 0x3a, 0xb6,  
0x96, 0xd5, 0xad, 0x88, 0x60, 0xb9, 0x09, 0x00,  
0xbb, 0x7d, 0x5c, 0xd6, 0x17, 0xfe, 0x18, 0x8e,  
0x07, 0x10, 0xc3, 0xe9, 0xd0, 0xb8, 0xe2, 0xfa,  
0x00, 0xae, 0xa1, 0xcd, 0x86, 0x33, 0xda, 0x4b,  
0x0c, 0x34, 0xa1, 0x6e, 0x19, 0x0b, 0xdb, 0xef,  
0xb0, 0xf6, 0x86, 0xc6, 0xe3, 0x8c, 0x9c, 0x53,  
0x15, 0x16, 0x04, 0xd9, 0xa8, 0xa7, 0x38, 0xc1,  
0x43, 0x9d, 0x7e, 0x33, 0x16, 0x1a, 0x8d, 0x33,  
}
```



0xe8, 0x2b, 0x47, 0xaf, 0x4c, 0xd0, 0x96, 0x87,  
0x5b, 0x57, 0x27, 0xc3, 0x1a, 0xf7, 0x12, 0xfd,  
0x8a, 0x64, 0xa0, 0xc9, 0x51, 0xc9, 0x95, 0xcb,  
0x7b, 0x9d, 0x97, 0xda, 0x3c, 0xae, 0x87, 0x1c,  
0x08, 0xa8, 0xb2, 0x3b, 0x92, 0xb7, 0x52, 0x97,  
0x99, 0x9c, 0x52, 0x92, 0xcc, 0xba, 0xbe, 0x16,  
0xf7, 0xac, 0x08, 0xb5, 0x1a, 0x99, 0x99, 0xee,  
0x33, 0x3a, 0x2a, 0xea, 0x70, 0xbe, 0xcc, 0xfd,  
0xc7, 0x6b, 0xf3, 0xa9, 0x9d, 0x66, 0x87, 0x1b,  
0x26, 0x0e, 0xf5, 0x04, 0x33, 0x82, 0x25, 0x75,  
0x74, 0x37, 0xbd, 0xfe, 0x78, 0xa2, 0x4a, 0x77,  
0xda, 0xef, 0x84, 0xfd, 0x5c, 0x1d, 0x09, 0xc7,  
0xd8, 0xc2, 0x97, 0xf8, 0xd0, 0x96, 0x6f, 0xf0,  
0x54, 0xea, 0x5e, 0x14, 0xb6, 0x9c, 0x16, 0x96,  
0xcd, 0x56, 0x3e, 0x49, 0xef, 0x8f, 0xbf, 0xc1,  
0x59, 0xd6, 0x0c, 0xa0, 0xd6, 0x6b, 0xff, 0x34,  
0xe5, 0x68, 0x70, 0x6b, 0x00, 0x87, 0xac, 0x4e,  
0x7e, 0x44, 0xa2, 0x04, 0xa2, 0x96, 0xfc, 0x73,  
0xe7, 0xe2, 0x7f, 0xc6, 0x0b, 0x17, 0xdb, 0xa6,  
0x45, 0xa8, 0x72, 0x61, 0x08, 0xc8, 0x4c, 0x19,  
0x2d, 0x27, 0x0b, 0x4f, 0xb9, 0xc0, 0x55, 0xd6,  
0x6d, 0x11, 0xd7, 0x15, 0x0d, 0xef, 0x97, 0x08,  
0xd3, 0x22, 0xd8, 0x03, 0x7f, 0x91, 0xe0, 0x0e,  
0xe4, 0x70, 0x75, 0x47, 0x33, 0xd9, 0x80, 0x08,  
0x5f, 0xc9, 0xea, 0x91, 0xd4, 0x4e, 0x80, 0x08,  
0xb3, 0x83, 0x37, 0xba, 0xd2, 0xe0, 0x6d, 0x44,  
0x83, 0x9a, 0xf1, 0xa6, 0x83, 0xc4, 0x5a, 0xd2,  
0x3f, 0x50, 0x7a, 0x19, 0xed, 0x82, 0xda, 0x02,  
0x6f, 0xbe, 0x27, 0x91, 0xd6, 0x7e, 0x11, 0xc1,  
0x95, 0x3a, 0x6a, 0x61, 0x80, 0x6c, 0x23, 0x1f,  
0xcd, 0x8d, 0xac, 0x5c, 0x4c, 0x14, 0xee, 0xde,  
0x08, 0x15, 0x87, 0xae, 0x2d, 0xf4, 0x77, 0x81,  
0xcb, 0x39, 0x9a, 0x51, 0xb1, 0x3e, 0xc6, 0xd5,  
0xcd, 0xa2, 0x3a, 0x15, 0x6a, 0x73, 0x2e, 0x78,  
0x24, 0x16, 0xb8, 0xd2, 0x3e, 0xb4, 0x0c, 0xc2,  
0x0e, 0x67, 0xea, 0xe9, 0x9c, 0xce, 0x5c, 0x6a,  
0x29, 0x27, 0xe1, 0x3f, 0xac, 0x9d, 0x31, 0xb8,  
0xda, 0x2f, 0x94, 0x6e, 0xfc, 0x33, 0xab, 0x54,  
0xf9, 0x8e, 0xe8, 0xbf, 0x8e, 0x33, 0x62, 0xab,  
0x1f, 0xca, 0x17, 0x68, 0x89, 0x27, 0x19, 0x52,  
0xb1, 0x4c, 0xd1, 0x04, 0xb8, 0x8d, 0xfe, 0xc2,  
0xa4, 0xbf, 0xb8, 0x4e, 0x7c, 0x24, 0x2f, 0xaa,  
0x8c, 0x57, 0x3c, 0x60, 0xe8, 0xd5, 0x1e, 0x7f,  
0x8b, 0x85, 0x0e, 0xfb, 0x0e, 0xbd, 0xed, 0x75,  
0x3f, 0x35, 0xeb, 0xb7, 0x35, 0x60, 0x53, 0x8f,  
0x7d, 0x86, 0xbc, 0x98, 0x8b, 0x2e, 0x22, 0x33,  
0x5e, 0x60, 0x49, 0x6d, 0xc3, 0xe0, 0xaa, 0xec,  
0x5e, 0x67, 0x87, 0xd8, 0x20, 0x92, 0x71, 0x34,



```
0x46, 0xa4, 0xf4, 0x2d, 0x02, 0xaf, 0x64, 0x45,
0xef, 0xea, 0xb3, 0x84, 0x39, 0xd6, 0x6b, 0xaf,
0x7d, 0x63, 0xd3, 0x50, 0xf9, 0x95, 0xaf, 0xf7,
0xf3, 0x8a, 0x6f, 0x59, 0xf1, 0x32, 0x37, 0x3d,
0x6c, 0xd4, 0xaa, 0x17, 0xd7, 0x13, 0x19, 0xa7,
0x51, 0x98, 0x21, 0x89, 0x26, 0x12, 0xe1, 0xed,
0x78, 0x04, 0x33, 0xd7, 0xc6, 0xf4, 0xe2, 0x39,
0x5c, 0x37, 0xdb, 0x16, 0x42, 0xf9, 0x0a, 0x3d,
0xee, 0x4b, 0x96, 0xbd, 0x60, 0x1f, 0x36, 0xbb,
0xe6, 0x15, 0x95, 0x47, 0x7f, 0x8f, 0x2e, 0xff,
0x59, 0x9b, 0xfc, 0x99, 0x13, 0x87, 0xac, 0x85,
0xd9, 0x84, 0x0c, 0x99, 0x95, 0xe8, 0x2b, 0xdf,
0xae, 0x64, 0x7e, 0x24, 0x85, 0x67, 0x9c, 0x86,
0x51, 0x8a, 0x61, 0x6c, 0x17, 0x24, 0x89, 0xba,
0x2f, 0xfa, 0x9d, 0x3d, 0xa6, 0x51, 0xce, 0x85,
0xf8, 0x95, 0x78, 0xeb, 0x00, 0x51, 0x06, 0xb4,
0x8b, 0x02, 0x1b, 0x1c, 0xf7, 0x13, 0xcb, 0xee,
0x83, 0x98, 0xdc, 0xab, 0xed, 0x57, 0x62, 0x78,
0x1c, 0xc5, 0x5c, 0xac, 0xa6, 0x23, 0x68, 0xd0,
0xa5, 0xda, 0x43, 0x2d, 0x61, 0x73, 0x66, 0x03,
0xea, 0xc9, 0xad, 0x7e, 0xe3, 0x54, 0xa9, 0x53,
0x3e, 0x23, 0x4c, 0x6a, 0x15, 0x70, 0xa5, 0x2c,
0xee, 0xcd, 0x4d, 0x7e, 0x41, 0x6f, 0xa6, 0xc5,
0x1c, 0x24, 0x37, 0x58, 0x00, 0x81, 0xd9, 0xb2,
0xf7, 0x9a, 0x9c, 0xa3, 0xf5, 0xc6, 0x31, 0xc1,
0xb2, 0x8b, 0x3d, 0xec, 0xbe, 0x21, 0xe7, 0x53,
0x0f, 0xb8, 0x87, 0x76, 0xb5, 0x76, 0xcc, 0x50,
0x03, 0x51, 0x8a, 0xa5, 0xb9, 0x50, 0xc7, 0x38,
0xaf, 0x98, 0x01, 0xdf, 0x77, 0xb2, 0x9f, 0xe8,
0xa5, 0x5b, 0x9d, 0x48, 0x3a, 0x82, 0xe1, 0x10,
0xc8, 0x34, 0xc1, 0x07, 0x8f, 0x63, 0x60, 0x3e,
0x25, 0xb6, 0x33, 0xbc, 0x15, 0xce, 0x99, 0x39,
0x62, 0x83, 0x5b, 0xbc, 0x22, 0xb9, 0x0b, 0xd3,
0x97, 0x2b, 0x87, 0xee, 0x85, 0xd6, 0x72, 0x01,
0xb8, 0xdb, 0xe1, 0xdd, 0x5f, 0x61, 0x5f, 0x81,
0x44, 0xcc, 0x65, 0x71, 0x44, 0xb7, 0xca, 0x48,
0xd2, 0x33, 0x7b, 0x56, 0xe6, 0x07, 0xb8, 0xc5,
0x6c, 0xb0, 0xf6, 0x72, 0x69, 0x75, 0xf7, 0xfc,
0xd5, 0xab, 0xe0, 0xbb, 0x65, 0xcb, 0xd8, 0x4a,
0xae, 0x99, 0x58, 0x5c, 0xe3, 0x61, 0x38, 0x07,
0x97, 0xee, 0xa7, 0x67, 0x48, 0xb7, 0x9e, 0xc0,
0xe9, 0xf5, 0x3c, 0x18, 0x3d, 0xb2, 0x06, 0x0f,
0x75, 0x8d, 0xb8, 0x82, 0xfa, 0x6d, 0x30, 0x91,
0x8b, 0x3b, 0xee, 0xf8, 0x27, 0x40, 0xec, 0x26,
0x08, 0xd4, 0xca, 0x00, 0x8f, 0x28, 0xa2, 0x38,
0xd9, 0xa0, 0x42, 0xc4, 0x51, 0x8b, 0x6c, 0xce
}
```



Author's Address

Dan Harkins  
HP Enterprise  
1322 Crossman avenue  
Sunnyvale, California 94089  
USA

Phone: +1 415 997 9834  
Email: [dharkins@lounge.org](mailto:dharkins@lounge.org)