ParrotTalk
Internet Draft
Intended status: IETF Contribution
Expires: April 2018

R. Withers
Callisto House
November 1, 2017

ParrotTalk: Anonymous P2P encryption over IPv4/v6 no Certificates
draft-withers-parrot-talk-v36-00.txt

Abstract

I have developed the ParrotTalk Protocol, v3.6 and v3.7 documented in part here[1a][1b]. ParrotTalk is an encrypted connection framework. Currently allowing anonymous 2048-bit key negotiation to establish user-provided encryption cipher and user-provided encoding and decoding, both through a provided SessionAgentMap to a starting SessionAgent server. There is a 4-way negotiation, from ProtocolOffered/Accepted to Go/GoToo. ParrotTalk uses RSA 2048-bit signature validation and DH 2048-bit primes to establish the key used within the selected Cipher. The Cipher and Encoder are selected by name through the negotiation protocol. Currently three Ciphers are selectable: AESede, DESede, and DES. There are three encoders tested: asn1der, String and Bytes. This protocol is described here, in this document.

I have two implementations, though they are being reorganized: 1 in Squeak/Pharo [2] and the other in Java [3]. The particulars of MAC key and ivSequence derivation, as well as constrained traffic signing, are in the implementations. They will be added to this Internet-Draft.

[1a] - https://github.com/CallistoHouseLtd/ParrotTalk/blob/master/docs/ParrotTalkFrameDesign-3.6.pdf
[1b] - https://github.com/CallistoHouseLtd/ParrotTalk/blob/master/docs/ParrotTalkFrameDesign-3.7.pdf

[2a] - http://www.squeaksource.com/Cryptography/Cryptography-rww.115.mcz
[2b] - http://www.squeaksource.com/Cryptography/ParrotTalk-rww.25.mcz

[3a] - https://github.com/CallistoHouseLtd/ASN1
[3b] - https://github.com/CallistoHouseLtd/ParrotTalk

Status of this Memo

Copyright Notice

carefully, as they describe your rights and restrictions with respect to this document. All code is released under CHPNPBL.

Table of Contents

1. Introduction

ParrotTalk is an encrypted connection framework. Currently allowing anonymous 2048-bit key negotiation to establish user-provided encryption cipher and user-provided encoding and decoding, both through a provided SessionAgentMap to a starting SessionAgent server. Please look in the test case ThunkHelloWorldTest for building these maps and running a connection with data passing after encryption is established. There is a 4-way negotiation, from ProtocolOffered/Accepted to Go/GoToo. ParrotTalk uses RSA 2048 signature validation and DH 2048 primes to establish the key used within the selected Cipher. The Cipher and Encoder are selected by name through the negotiation protocol. Currently three Ciphers are selectable, AESede, DESede, and DES. There are two encoders tested, asn1der, and Bytes. This protocol is described here, in this document.

1.1. Frame Design

Frames are used in message pipeline, consisting of

- an 8 byte message specification,
- a msgType ASN1Choice Encoded header
- a possible data payload.

Frames are exchanged between layers, up & down the stack.

Each protocol frame transforms session state through the SessionOperations layer.

Each data layer transforms each frame by established session protocol.

As payload is transformed, header is transformed and re-encoded ASN1Der.

MsgSpec knows header & frame encoding specification.

Natural nested wrapping of data msgs, where an inner frame's messageSize removes down stack padding.

Protocol stack is established during session rendezvous with these data wrapping specifications:

Encoded – Primary payload

Encrypted – AES-256/CBC/PKCS7Padding with 128-bitblockSize & IV and a 256-bit key

MAC – 160-bit hmac hash

1.2. v3.6 Protocol Design

3-way rendezvous handshake protocol with Protocol pre-exchange

Protocol pre-exchange (ProtocolOffered/ProtocolAccepted)

VatId/Domain agreement (IWant/IAm)

2048-bit RSA PublicKey exchange (Iam/GiveInfo)

CryptoProtocol negotiation (ReplyInfo/Go/GoToo)

DataEncoder negotiation (ReplyInfo/Go/GoToo)

2048-bit prime/secret Diffie-Hellman parameter exchange (Go/GoToo)

Prior protocol traffic 2048-bit RSA Signature authentication (Go/GoToo)

QuadScopeInfrastrucure 4,5,6, , ,9:

4: Goose – routing

5: Parrot – session

6: Raven – presentation

7: Pidgeon - App DSL

8: Vulture - Container DSL

9: Eagle - meta

• Diffie-Hellman prime is the 2048-bit https://tools.ietf.org/html/rfc3526#page-3

• Diffie-Hellman generator is 2 from the same source

1.3. v3.6 Protocol Design

8 message rendezvous handshake protocol with Protocol pre-exchange

Protocol pre-exchange (ProtocolOffered/ProtocolAccepted)

VatId/Domain agreement (Hello_v3_7/Response_v3_7)

2048-bit RSA PublicKey exchange (Hello_v3_7/Response_v3_7)

CryptoProtocol negotiation (Hello_v3_7/Response_v3_7)

DataEncoder negotiation (Hello_v3_7/Response_v3_7)

2048-bit prime/secret Diffie-Hellman parameter exchange (Hello_v3_7/Response_v3_7)

Prior protocol traffic 2048-bit RSA Signature authentication (Response_v3_7/Signature_v3_7)

1.4. Protocol Headers

1.4.1. RSA Public Key

```
RSA DEFINITIONS ::= BEGIN

    RSAPublicKey::= SEQUENCE {
        modulo    Modulus,
        exponent  PublicExponent
    }

    Modulus ::= INTEGER.
    PublicExponent ::= INTEGER.

END
```

1.4.2. Layer 4: Goose – routing

```
Session DEFINITIONS ::= BEGIN

    ProtocolOffered ::= SEQUENCE {
        offered   UTF8String,
        preffered  UTF8String
    }
    ProtocolAccepted ::= SEQUENCE {
        accepted  UTF8String
    }
```

1.4.3. Base Layer 5: Parrot – session

```
Session DEFINITIONS ::= BEGIN

    <4> RawData ::= SEQUENCE {
    }

    <5> Encoded ::= SEQUENCE {

    }


    <6> Encrypted ::= SEQUENCE {
        ivSequence    OCTET STRING
     }

    <7> MAC ::= SEQUENCE {
        mac    OCTET STRING
     }

    <14> DuplicateConnection ::= SEQUENCE {
     }

    <15> NotMe ::= SEQUENCE {
    }
```

1.4.4. Version 3.6 Layer 5: Parrot – session

```
Session DEFINITIONS ::= BEGIN

    <8> IWant ::= SEQUENCE {
       vatId  UTF8String,
       domain UTF8String
    }

    <9> IAm ::= SEQUENCE {
       vatId  UTF8String,
```

```
      domain UTF8String,
      publicKey RSAPublicKey
   }



   <10> GiveInfo ::= SEQUENCE {
      vatId  UTF8String,
      domain UTF8String,
      publicKey RSAPublicKey
   }

   <11> ReplyInfo ::= SEQUENCE {
      cryptoProtocols   SEQUENCE OF STRING,
      dataEncoders      SEQUENCE OF STRING,
   }

   <12> Go ::= SEQUENCE {
      cryptoProtocol              UTF8String,
      dataEncoder                 UTF8String,
      diffieHellmanParameter      OCTET STRING,
      signature                   OCTET STRING
   }

   <13> GoToo ::= SEQUENCE {
      cryptoProtocol              UTF8String,
      dataEncoder                 UTF8String,
      diffieHellmanParameter      OCTET STRING,
      signature                   OCTET STRING
   }
```

1.4.5. Version 3.6 Layer 5: Parrot – session

```
Session DEFINITIONS ::= BEGIN

   <16> Hello_v3_7 ::= SEQUENCE {
      vatId  UTF8String,
      domain UTF8String,
      publicKey RSAPublicKey,
      cryptoProtocols   SEQUENCE OF STRING,
      dataEncoders      SEQUENCE OF STRING,
      diffieHellmanParameter      OCTET STRING
   }

   <17> Response_v3_7 ::= SEQUENCE {
      vatId  UTF8String,
      domain UTF8String,
      publicKey RSAPublicKey,
```

```
      cryptoProtocols   SEQUENCE OF STRING,
      dataEncoders      SEQUENCE OF STRING,
      diffieHellmanParameter       OCTET STRING,
      signature  OCTET STRING
   }


   <18> Signature_v3_7 ::= SEQUENCE {
      signature  OCTET STRING
   }
```

2. User Interface

VatID:Domain

Host:Port Internet Address

User-Defined CipherThunkMaker

^ CipherThunkMaker newName: 'AESede' cipherClass: Rijndael keySize: 32 blockSize: 16 hasIvParameter: true

User-Defined EncoderThunk

^ SessionAgentMap

newProtocol: (CipherThunkMaker newName: 'AESede' cipherClass: Rijndael keySize: 32 blockSize: 16 hasIvParameter: true)

encoder: (EncoderThunk

newName: 'String'

serializeThunk: [:payload | payload asByteArray]

materializeThunk: [:payload | payload asString])

3. Thunks

3.1. Thunk Stack

The ThunkStack is a stack of layers that get pushed and popped as the state machine changes shape. This really only happens in three places: initialized, encrypted, shutdown. There is a base control protocol and several callback methods implemented by the Thunks.

3.2. Anonymous Thunking

When the state machine connects past rendezvous, the state machine change allows thee SecurityOps to add three anonymous thunks and a user-provided encoderThunk, for immigration, user-provided decryption and customs into raw data through decoding. Please see SecurityOps>>#installOnSession.

3.3. Thunk Layers

Here is the rendezvous stack followed by the encrypted stack

3.3.1. Rendezvous:

Session

SendFramesBuffer

SessionOperations

ReceivingFrameBuffer

SocketThunk

3.3.2. Encrypted:

Session

EncoderThunk

Customs MAC validation thunk

CipherMakerThunk>>#makeThunk

Immigration MAC recording thunk

SessionOperations

ReceivingFrameBuffer

SocketThunk

4. Frames

4.1. 8-Byte Frame Specifications

1st-3rd Identity Specification

4th Byte Route Specification

5th-8th Message Size

4.2. Frame Phase Headers

4.3. Frame Payload

5. Headers

```
Session DEFINITIONS ::= BEGIN

    <16> PhaseHeader ::= CHOICE {
       duplicateConnection [15] EXPLICIT DuplicateConnection,
       notMe [14] EXPLICIT NotMe,
       rawData [4] EXPLICIT RawData,
       encoded [5] EXPLICIT Encoded,
       encrypted [6] EXPLICIT Encrypted,
       mac [7] EXPLICIT MAC,
       iwant [8] EXPLICIT IWant,
       iam [9] EXPLICIT IAm,
       giveInfo [10] EXPLICIT GiveInfo,
       replyInfo [11] EXPLICIT ReplyInfo,
       go [12] EXPLICIT Go,
       goToo [13] EXPLICIT GoToo,
       hello_v3_7 [16] EXPLICIT Hello_v3_7,
       response_v3_7 [17] EXPLICIT Response_v3_7,
       signature_v3_7 [18] EXPLICIT Signature_v3_7
    }
```

6. Security

6.1. Diffie Hellman Key Exchange

- Diffie-Hellman prime is the 2048-bit MODP Group

  https://tools.ietf.org/html/rfc3526#page-3

- Diffie-Hellman generator is 2 from the same source

6.2. MAC Key

There is a #hash:pad: message that preloads 16 bytes of the pad then adds the message then MD5 hashes the message.

Now we can create the MAC keys, wrapped in an SHA1HMAC.

Pharo/Squeak

makeHMAC

```
| sharedKey hashPadder macKey |
sharedKey := diffieHellman sharedKeyPadPositiveByteArray.
hashPadder := self class.
macKey := MD5 hashMessage: (
   (hashPadder hash: sharedKey pad: 16rCC),
   (hashPadder hash: sharedKey pad: 16rBB),
   (hashPadder hash: sharedKey pad: 16rAA),
   (hashPadder hash: sharedKey pad: 16r99)).
macKey := macKey, (MD5 hashMessage: (
   (hashPadder hash: sharedKey pad: 16r88),
   (hashPadder hash: sharedKey pad: 16r77),
   (hashPadder hash: sharedKey pad: 16r66),
   (hashPadder hash: sharedKey pad: 16r55))).
macKey := macKey, (MD5 hashMessage: (
   (hashPadder hash: sharedKey pad: 16r44),
   (hashPadder hash: sharedKey pad: 16r33),
   (hashPadder hash: sharedKey pad: 16r22),
   (hashPadder hash: sharedKey pad: 16r11))).
```

Java

```
private void generateMacKey(byte[] sharedKey) throws NoSuchAlgorithmException
{
    macBytes = md5Hash(ArrayUtil.concatAll(
```

```
                padAndHash(new byte[] { (byte)0xCC }, sharedKey),
                padAndHash(new byte[] { (byte)0xBB }, sharedKey),
                padAndHash(new byte[] { (byte)0xAA }, sharedKey),
                padAndHash(new byte[] { (byte)0x99 }, sharedKey)));
        macBytes = ArrayUtil.concatAll(macBytes, md5Hash(ArrayUtil.concatAll(
                padAndHash(new byte[] { (byte)0x88 }, sharedKey),
                padAndHash(new byte[] { 0x77 }, sharedKey),
                padAndHash(new byte[] { 0x66 }, sharedKey),
                padAndHash(new byte[] { 0x55 }, sharedKey))));
        macBytes = ArrayUtil.concatAll(macBytes, md5Hash(ArrayUtil.concatAll(
                padAndHash(new byte[] { 0x44 }, sharedKey),
                padAndHash(new byte[] { 0x33 }, sharedKey),
                padAndHash(new byte[] { 0x22 }, sharedKey),
                padAndHash(new byte[] { 0x11 }, sharedKey))));
    }
```

6.3. Cipher Key

```
Pharo/Squeak
cipherOnSecretBytes: secretBytes incoming: incoming mode: cryptMode

    | keyBytes cipher |
    keyBytes := (secretBytes size == keySize)
        ifTrue: [secretBytes]
        ifFalse: [keyBytes := secretBytes forceTo: keySize paddingWith:
16r98].
```

```
    cipher := cipherClass new.

    keySize ifNotNil: [[cipher keySize: keySize] on: Exception do:
[:v|]].

    cipher := (cipher key: keyBytes) cbc.

    self hasIvParameter

        ifTrue: [cipher initialVector: (self computeIv: keyBytes
incoming: incoming mode: cryptMode)].

    ^ cipher
```

Java

```
    private Cipher buildCipher(byte[] secretBytes, boolean incoming, int
cryptMode) {

        byte[] keyBytes;

        if(secretBytes.length >= keySize) {

            keyBytes = Arrays.copyOf(secretBytes, keySize);

        } else {

            keyBytes = Arrays.copyOf(secretBytes, keySize);

            Arrays.fill(keyBytes, secretBytes.length, keySize, (byte) 0x98);

        }

        secretKeySpec = new SecretKeySpec(keyBytes, fullCryptoProtocol.split("/")
[0]);

        Cipher cipher = null;

        try {

            cipher = Cipher.getInstance(fullCryptoProtocol);

        } catch (NoSuchAlgorithmException e) {

            e.printStackTrace();

        } catch (NoSuchPaddingException e) {

            e.printStackTrace();
```

```
        }
        if(hasIvParameter) {
            try {
                cipher.init(cryptMode, secretKeySpec, computeIVSpec(secretBytes,
   incoming, cryptMode));
            } catch (InvalidAlgorithmParameterException e) {
                e.printStackTrace();
            } catch (InvalidKeyException e) {
                e.printStackTrace();
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        } else {
            try {
                cipher.init(cryptMode, secretKeySpec);
            } catch (InvalidKeyException e) {
                e.printStackTrace();
            }
        }
        return cipher;
    }
```

6.4. IV Sequence Key

Pharo/Squeak

computeIv: secretBytes incoming: incoming mode: cryptMode

```
      | hash receive send |

      hash := self computeIvHash: secretBytes.


      incoming

         ifTrue: [

            send := hash copyFrom: (self blockSize + 1) to: (self
   blockSize * 2).

            receive := hash copyFrom: 1 to: self blockSize]

         ifFalse: [

            send := hash copyFrom: 1 to: self blockSize.

            receive := hash copyFrom: (self blockSize + 1) to: (self
   blockSize * 2)].


      ^ (cryptMode == #ENCRYPT)

         ifTrue: [send]

         ifFalse: [receive].


   computeIvHash: secretBytes

      | opsHash |

      opsHash := SecurityOps hash: secretBytes pad: 16r33.

      [(blockSize * 2) > opsHash size] whileTrue: [opsHash := opsHash,
   (SecurityOps hash: secretBytes pad: 16r33)].

      ^ opsHash

   Java

      private IvParameterSpec computeIVSpec(byte[] secretBytes, boolean incoming,
   int cryptMode) throws NoSuchAlgorithmException {

         IvParameterSpec ivSpec = null;
```

```
            byte[] hash = computeIVHash(secretBytes);

            if (incoming) {

                if (cryptMode == Cipher.ENCRYPT_MODE) {

                    ivSpec = new IvParameterSpec(Arrays.copyOfRange(hash, blockSize,
    blockSize * 2));

                } else {

                    ivSpec = new IvParameterSpec(Arrays.copyOfRange(hash, 0,
    blockSize));

                }

            } else {

                if (cryptMode == Cipher.ENCRYPT_MODE) {

                    ivSpec = new IvParameterSpec(Arrays.copyOfRange(hash, 0,
    blockSize));

                } else {

                    ivSpec = new IvParameterSpec(Arrays.copyOfRange(hash, blockSize,
    blockSize * 2));

                }

            }

            return ivSpec;

        }

        private byte[] computeIVHash(byte[] secretBytes) throws
    NoSuchAlgorithmException {

            byte[] opsHash = SecurityOps.padAndHash(new byte[] { 0x33 }, secretBytes);

            int opsLength = opsHash.length;

            while((blockSize * 2) > opsHash.length) {

                byte[] bytes = new byte[opsHash.length + opsLength];

                System.arraycopy(opsHash, 0, bytes, 0, opsHash.length);
```

```
            opsLength = opsHash.length;
            opsHash = SecurityOps.padAndHash(new byte[] { 0x33 }, secretBytes);
            System.arraycopy(opsHash, 0, bytes, opsLength, opsHash.length);
            opsLength = opsHash.length;
            opsHash = bytes;
         }
         return opsHash;
      }
```

6.5. Signature Contents

The signature is computed over the ASN.1 DER encoded headers of the 4 remote traffic messages sent/received.

7. Operation

7.1. State Machine

The state machine maintains knowledge of what triggers lead to what actions. Triggers are entering expectation of a message and reception of the message.

7.2. State Map

7.2.1. Squeak/Pharo stateMap

```
stateMap
   "(((SessionOperations stateMap compile)))"

   | desc |
   desc := ProtocolStateCompiler initialState: #initial.
   (desc newState: #initial -> (#processInvalidRequest: -> #dead))
      add: #answer -> (nil -> #receivingExpectProtocolOffered);
```

```
        add: #call -> (nil -> #receivingExpectProtocolAccepted).

     (desc newState: #connected -> (#processInvalidRequest: -> #dead))

        addInteger: 7 -> (#processBytes: -> #connected).

     (desc newState: #dead -> (#processInvalidRequest: -> #dead)).


     (desc newState: #receivingExpectProtocolOffered ->
  (#processInvalidRequest: -> #dead))

        addInteger: 1 -> (#processProtocolOffered: ->
  #receivingExpectIWant).

     (desc newState: #receivingExpectIWant -> (#processInvalidRequest:
  -> #dead))

        addInteger: 8 -> (#processIWant: -> #receivingExpectGiveInfo).

     (desc newState: #receivingExpectGiveInfo ->
  (#processInvalidRequest: -> #dead))

        addInteger: 10 -> (#processGiveInfo: -> #receivingExpectGo);

        addInteger: 14 -> (#processDuplicateConnection: -> #dead);

        addInteger: 15 -> (#processNotMe: -> #dead).

     (desc newState: #receivingExpectGo -> (#processInvalidRequest: ->
  #dead))

        addInteger: 12 -> (#processGo: -> #connected);

        addInteger: 14 -> (#processDuplicateConnection: -> #dead).


     (desc newState: #receivingExpectProtocolAccepted ->
  (#processInvalidRequest: -> #dead))

        addInteger: 3 -> (#processProtocolAccepted: ->
  #receivingExpectIAm).

     (desc newState: #receivingExpectIAm -> (#processInvalidRequest: ->
  #dead))

        addInteger: 9 -> (#processIAm: -> #receivingExpectReplyInfo);
```

```
        addInteger: 14 -> (#processDuplicateConnection: -> #dead);

        addInteger: 15 -> (#processNotMe: -> #dead).

    (desc newState: #receivingExpectReplyInfo ->
(#processInvalidRequest: -> #dead))

        addInteger: 11 -> (#processReplyInfo: ->
#receivingExpectGoToo);

        addInteger: 14 -> (#processDuplicateConnection: -> #dead).

    (desc newState: #receivingExpectGoToo -> (#processInvalidRequest:
-> #dead))

        addInteger: 13 -> (#processGoToo: -> #connected).

    ^desc.
```

7.2.2. Java stateMap

```
    public StateMachineConfig<State,Trigger> buildStateMachineConfig() {

        StateMachineConfig<State, Trigger> sessionConnectionConfig = new
StateMachineConfig<State, Trigger>();


        sessionConnectionConfig.configure(State.Initial)

            .permit(Trigger.Calling, State.CallInProgress)

            .permit(Trigger.Answering, State.AnswerInProgress);

        sessionConnectionConfig.configure(State.EncryptedConnected)

            .permit(Trigger.Disconnect, State.Closed);

        sessionConnectionConfig.configure(State.Closed)

            .onEntry(new Action() {

                public void doIt() {

                    session.stop();
```

```
            }});
        sessionConnectionConfig.configure(State.Startup)
            .permit(Trigger.SendBye, State.IdentifiedStartupSendingBye)
            .permit(Trigger.Disconnect, State.Closed);
        sessionConnectionConfig.configure(State.IdentifiedStartup)
            .permit(Trigger.SendBye, State.IdentifiedStartupSendingBye)
            .permit(Trigger.Disconnect, State.Closed);
        sessionConnectionConfig.configure(State.StartupSendingNotMe)
            .substateOf(State.Startup)
            .onEntry(new Action() {
                public void doIt() {
                    sendNotMe();
                }});
        sessionConnectionConfig.configure(State.IdentifiedStartupSendingBye)
            .substateOf(State.IdentifiedStartup)
            .onEntry(new Action() {
                public void doIt() {
                    stateMachine.fire(Trigger.Disconnect);
                }});


        /**
         * Calling states
         */
        sessionConnectionConfig.configure(State.CallInProgress)
            .substateOf(State.Initial)
```

```
            .onEntry(new Action() {
                public void doIt() {
                    sendProtocolOffered();
                }})
            .permit(Trigger.ExpectProtocolAccepted,
   State.CallReceiveProtocolAccepted);
        sessionConnectionConfig.configure(State.CallReceiveProtocolAccepted)
            .substateOf(State.CallInProgress)
            .permit(Trigger.ReceivedProtocolAccepted, State.StartupSendingIWant);
        sessionConnectionConfig.configure(State.StartupSendingIWant)
            .substateOf(State.Startup)
            .onEntry(new Action() {
                public void doIt() {
                    sendIWant();
                }})
            .permit(Trigger.ExpectIAm, State.StartupReceiveIAm);
        sessionConnectionConfig.configure(State.StartupReceiveIAm)
            .substateOf(State.Startup)
            .permit(Trigger.ReceivedIAm, State.StartupSendingGiveInfo);
        sessionConnectionConfig.configure(State.StartupSendingGiveInfo)
            .substateOf(State.Startup)
            .onEntry(new Action() {
                public void doIt() {
                    sendGiveInfo();
                }})
```

```
            .permit(Trigger.ExpectReplyInfo,
    State.IdentifiedStartupReceiveReplyInfo);

        sessionConnectionConfig.configure(State.IdentifiedStartupReceiveReplyInfo)
            .substateOf(State.IdentifiedStartup)
            .permit(Trigger.ReceivedReplyInfo, State.IdentifiedStartupSendingGo);
        sessionConnectionConfig.configure(State.IdentifiedStartupSendingGo)
            .substateOf(State.IdentifiedStartup)
            .onEntry(new Action() {
               public void doIt() {
                   sendGo();
               }})
            .permit(Trigger.SendBye, State.IdentifiedStartupSendingBye)
            .permit(Trigger.ExpectGoToo, State.IdentifiedStartupReceiveGoToo);
        sessionConnectionConfig.configure(State.IdentifiedStartupReceiveGoToo)
            .substateOf(State.IdentifiedStartup)
            .permit(Trigger.SendBye, State.IdentifiedStartupSendingBye)
            .permit(Trigger.ReceivedGoToo, State.IdentifiedStartupConnecting);
        sessionConnectionConfig.configure(State.IdentifiedStartupConnecting)
            .substateOf(State.IdentifiedStartup)
            .onEntry(new Action() {
               public void doIt() {
                   stateMachine.fire(Trigger.Connect);
               }})
            .permit(Trigger.Connect, State.EncryptedConnected);
```

```
        /**
         * Answering states
         */
        sessionConnectionConfig.configure(State.AnswerInProgress)
            .substateOf(State.Initial)
            .onEntry(new Action() {
                public void doIt() {
                    stateMachine.fire(Trigger.ExpectProtocolOffered);
                }})
            .permit(Trigger.ExpectProtocolOffered,
State.AnswerReceiveProtocolOffered);
        sessionConnectionConfig.configure(State.AnswerReceiveProtocolOffered)
            .substateOf(State.AnswerInProgress)
            .permit(Trigger.ReceivedProtocolOffered,
State.AnswerSendingProtocolAccepted);
        sessionConnectionConfig.configure(State.AnswerSendingProtocolAccepted)
            .substateOf(State.AnswerInProgress)
            .onEntry(new Action() {
                public void doIt() {
                    sendProtocolAccepted();
                }})
            .permit(Trigger.ExpectIWant, State.StartupReceiveIWant);
        sessionConnectionConfig.configure(State.StartupReceiveIWant)
            .substateOf(State.Startup)
            .permit(Trigger.SendNotMe, State.StartupSendingNotMe)
            .permit(Trigger.ReceivedIWant, State.StartupSendingIAm);
```

```
        sessionConnectionConfig.configure(State.StartupSendingIAm)
            .substateOf(State.Startup)
            .onEntry(new Action() {
                public void doIt() {
                    sendIAm();
                }})
            .permit(Trigger.ExpectGiveInfo, State.StartupReceiveGiveInfo);
        sessionConnectionConfig.configure(State.StartupReceiveGiveInfo)
            .substateOf(State.Startup)
            .permit(Trigger.ReceivedGiveInfo,
State.IdentifiedStartupSendingReplyInfo);
        sessionConnectionConfig.configure(State.IdentifiedStartupSendingReplyInfo)
            .substateOf(State.IdentifiedStartup)
            .onEntry(new Action() {
                public void doIt() {
                    sendReplyInfo();
                }})
            .permit(Trigger.ExpectGo, State.IdentifiedStartupReceiveGo);
        sessionConnectionConfig.configure(State.IdentifiedStartupReceiveGo)
            .substateOf(State.IdentifiedStartup)
            .permit(Trigger.SendBye, State.IdentifiedStartupSendingBye)
            .permit(Trigger.ReceivedGo, State.IdentifiedStartupSendingGoToo);
        sessionConnectionConfig.configure(State.IdentifiedStartupSendingGoToo)
            .substateOf(State.IdentifiedStartup)
            .onEntry(new Action() {
```

```
                public void doIt() {
                    sendGoToo();
                    stateMachine.fire(Trigger.Connect);
                }})
            .permit(Trigger.Connect, State.EncryptedConnected);

        return sessionConnectionConfig;
    }
```

8. Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

In this document, the characters ">>" preceding an indented line(s) indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the portions of this RFC covered by these keywords.

9. Security Considerations

INFO (REMOVE): Every draft MUST have a Security Considerations section.

<Add any security considerations>

10. IANA Considerations

INFO (REMOVE): Every draft MUST have an IANA Considerations section, although it may be removed prior to publication by the RFC Editor if null.

<Add any IANA considerations>

11. Conclusions

11.1. The Callisto House Profit Non-Profit Bipolar License 1.10 (CHPNPB 1.10)

11.1.1. The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

11.1.2. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

11.1.3. All use and study and learning of said software system, as well as any modifications thereof, are used for NON-PROFIT ACTIVITES, without charge.

11.1.4. All use and study and learning of said software system, as well as any modifications thereof, are used for PROFIT ACTIVITIES, require 2.5% royalty charges to CALLISTO ENTERPRISES, LLC.

12. References

INFO (REMOVE): Authors can use either the auto-numbered references OR the named references; typically, these would not be mixed in a single document. This template includes both examples for illustration of the two variations. Named references are preferred (e.g., [RFC2119] or [Fab1999].

12.1. Normative References

INFO (REMOVE): Normative refs are references to standards documents **required to understand this doc. These are usually Standards-track and BCP RFCs, or external (IEEE, ANSI, etc.) standards, but may include other publications.**

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

[2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

FC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

FC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

12.2. Informative References

INFO (REMOVE): Informative refs are those that are not standards or standards not required to understand this doc. These are usually informative RFCs, internet-drafts (avoid if possible), and other external documents.

[3] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

ab1999] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

13. Acknowledgments

murmur/whisper/hubbub/ParrotTalk would not be possible without the ideas, implementation, brilliance and passion of the erights.org community, which are this software's conceptual foundation and reference implementation. In particular, We would like to thank the following individuals:

Mark Miller

Marc Stiegler

Bill Franz

Tyler Close

Kevin Reid

murmur/whisper/hubbub/ParrotTalk would not be possible without the ideas, implementation, brilliance and passion of the Squeak/Pharo communities and the cryptography team and the virtual machine team, which are this software's implementation foundation. Thank you.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. <First Appendix>

INFO (REMOVE): Starts on a new page. These are optional.

INFO (REMOVE): Careful with headers in appendices - they won't renumber when moved in/out levels in outline mode. Only Headers 1-9 do that trick, as used in the body of the RFC!

A.1. <First Header level>

<Text>

A.2. <Second Header level 1>

INFO (REMOVE): The following 'abbreviated statement' or the subsequent 'full statement' are required if the document contains any code (see BCP 78 for further information). The first is an 'abbreviated statement':

INFO (REMOVE): or use this 'full statement' (through the remainder of this subsection):

- Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.2.1. <H2>

<Text>

A.2.1.1. <H3>

<Text>

A.2.1.1.1. <H4>

<Text>

A.2.1.1.1.1. <H5>

<Text>

Authors' Addresses

Robert Withers
Callisto House
Cackalacky, USA

Email: robert.withers@protonmail.com