Network Working Group
Internet-Draft
draft-riikonen-silc-spec-09.txt
Expires: 15 July 2007

P. Riikonen

15 January 2007

Secure Internet Live Conferencing (SILC), Protocol Specification <draft-riikonen-silc-spec-09.txt>

Status of this Draft

Abstract

This memo describes a Secure Internet Live Conferencing (SILC) protocol which provides secure conferencing services over insecure network channel. SILC provides advanced and feature rich conferencing services with security as main design principal. Strong cryptographic methods are used to protect SILC packets inside the SILC network. Three other specifications relates very closely to this memo; SILC Packet Protocol [SILC2], SILC Key Exchange and Authentication Protocols [SILC3] and SILC Commands [SILC4].

Table of Contents

List of Figures

1. Introduction

This document describes a Secure Internet Live Conferencing (SILC) protocol which provides secure conferencing services over insecure network channel. SILC can be used as a secure conferencing service that provides rich conferencing features. Some of the SILC features are found in traditional chat protocols such as IRC [IRC] but many of the SILC features can also be found in Instant Message (IM) style protocols. SILC combines features from both of these chat protocol styles, and can be implemented as either IRC-like system or IM-like system. Some of the more advanced and secure features of the protocol are new to all conferencing protocols. SILC also supports

multimedia messages and can also be implemented as a video and audio conferencing system.

Strong cryptographic methods are used to protect SILC packets inside the SILC network. Three other specifications relates very closely to this memo; SILC Packet Protocol [SILC2], SILC Key Exchange and Authentication Protocols [SILC3] and SILC Commands [SILC4].

The protocol uses extensively packets as conferencing protocol requires message and command sending. The SILC Packet Protocol is described in [SILC2] and should be read to fully comprehend this document and protocol. [SILC2] also describes the packet encryption and decryption in detail. The SILC Packet Protocol provides secured and authenticated packets, and the protocol is designed to be compact. This makes SILC also suitable in environment of low bandwidth requirements such as mobile networks. All packet payloads in SILC can be also compressed.

The security of SILC protocol sessions are based on strong and secure key exchange protocol. The SILC Key Exchange protocol is described in [SILC3] along with connection authentication protocol and should be read to fully comprehend this document and protocol.

The SILC protocol has been developed to work on both TCP/IP and UDP/IP network protocols. However, typical implementation would use only TCP/IP with SILC protocol. Typical implementation would be made in client-server model.

1.1 Requirements Terminology

The keywords MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

2. SILC Concepts

This section describes various SILC protocol concepts that forms the actual protocol, and in the end, the actual SILC network. The mission of the protocol is to deliver messages from clients to other clients through servers and routers in secure manner. The messages may also be delivered from one client to many clients forming a group, also known as a channel.

This section does not focus to security issues. Instead, basic network concepts are introduced to make the topology of the SILC network clear.

2.1 SILC Network Topology

SILC network forms a ring as opposed to tree style network topology that conferencing protocols usually have. The network has a cells which are constructed from a router and zero or more servers. The servers are connected to the router in a star like network topology. Routers in the network are connected to each other forming a ring. The rationale for this is to have servers that can perform specific kind of tasks what other servers cannot perform. This leads to two kinds of servers; normal SILC servers and SILC router servers.

A difference between normal server and router server is that routers knows all global information and keep the global network state up to date. They also do the actual routing of the messages to the correct receiver within the cell and between other cells. Normal servers knows only local information and receive global information only when it is needed. They do not need to keep the global network state up to date. This makes the network faster and scalable as there are less servers that needs to maintain global network state.

This, on the other hand, leads into a cellular like network, where routers are in the center of the cell and servers are connected to the router.

The following diagram represents SILC network topology.

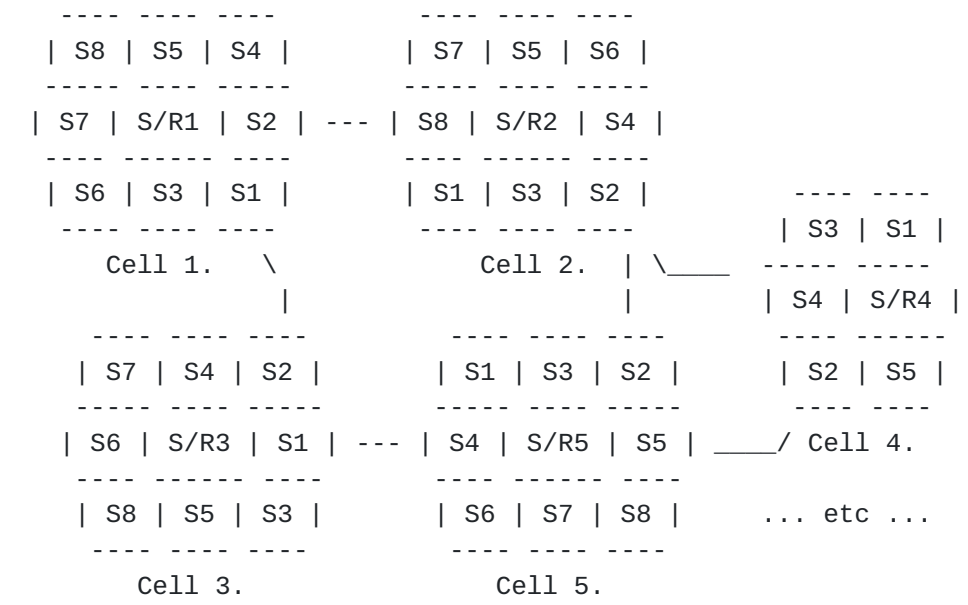

Figure 1: SILC Network Topology

A cell is formed when a server or servers connect to one router. In

SILC network normal server cannot directly connect to other normal server. Normal server may only connect to SILC router which then routes the messages to the other servers in the cell. Router servers on the other hand may connect to other routers to form the actual SILC network, as seen in above figure. However, router is also able to act as normal SILC server; clients may connect to it the same way as to normal SILC server. This, however is not a requirement and if needed router servers may be hidden from users by not allowing direct client connections. Normal server also cannot have active connections to more than one router. Normal server cannot be connected to two different cells. Router servers, on the other hand, may have as many router to router connections as needed. Other direct routes between other routers is also possible in addition of the mandatory ring connections. This leads into a hybrid ring-mesh network topology.

There are many issues in this network topology that needs to be careful about. Issues like routing, the size of the cells, the number of the routers in the SILC network and the capacity requirements of the routers. These issues should be discussed in the Internet Community and additional documents on the issue may be written.

2.2 Communication Inside a Cell

It is always guaranteed that inside a cell message is delivered to the recipient with at most two server hops. A client which is connected to server in the cell and is talking on channel to other client connected to other server in the same cell, will have its messages delivered from its local server first to the router of the cell, and from the router to the other server in the cell.

The following diagram represents this scenario:

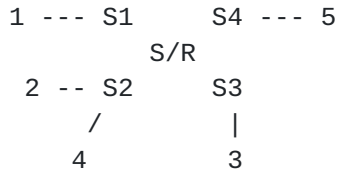

Figure 2: Communication Inside cell

Example: Client 1. connected to Server 1. send message to Client 4. connected to Server 2. travels from Server 1. first to Router which routes the message to Server 2. which then sends it to the Client 4. All the other

servers in the cell will not see the routed message.

If the client is connected directly to the router, as router is also normal SILC server, the messages inside the cell are always delivered only with one server hop. If clients communicating with each other are connected to the same server, no router interaction is needed. This is the optimal situation of message delivery in the SILC network.

2.3 Communication in the Network

If the message is destined to client that does not belong to local cell the message is routed to the router server to which the destination client belongs, if the local router is connected to destination router. If there is no direct connection to the destination router, the local router routes the message to its primary route. The following diagram represents message sending between cells.

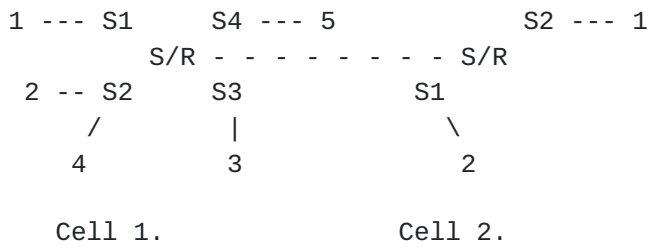

Figure 3: Communication Between Cells

Example: Client 5. connected to Server 4. in Cell 1. sends message to Client 2. connected to Server 1. in Cell 2. travels from Server 4. to Router which routes the message to Router in Cell 2, which then routes the message to Server 1. All the other servers and routers in the network will not see the routed message.

The optimal case of message delivery from the client point of view is when clients are connected directly to the routers and the messages are delivered from one router to the other.

2.4 Channel Communication

Messages may be sent to group of clients as well. Sending messages to many clients works the same way as sending messages point to point, from message delivery point of view. Security issues are another matter which are not discussed in this section.

Router server handles the message routing to multiple recipients. If any recipient is not in the same cell as the sender the messages are routed further.

Server distributes the channel message to its local clients which are joined to the channel. Router also distributes the message to its local clients on the channel.

2.5 Router Connections

Router connections play very important role in making the SILC like network topology to work. For example, sending broadcast packets in SILC network require special connections between routers; routers must be connected in a specific way.

Every router has their primary route which is a connection to another router in the network. Unless there is only two routers in the network must not routers use each other as their primary routes. The router connections in the network must form a ring.

Example with three routers in the network:

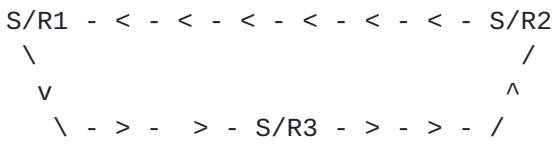

Figure 4: Router Connections

Example: Network with three routers. Router 1. uses Router 2. as its primary router. Router 2. uses Router 3. as its primary router, and Router 3. uses Router 1. as its primary router. When there are four or more routers in th enetwork, there may be other direct connections between the routers but they must not be used as primary routes.

The above example is applicable to any amount of routers in the network

except for two routers. If there are only two routers in the network both routers must be able to handle situation where they use each other as their primary routes.

The issue of router connections are very important especially with SILC broadcast packets. Usually all router wide information in the network is distributed by SILC broadcast packets. This sort of ring network, with ability to have other direct routes in the network can cause interesting routing problems. The [SILC2] discusses the routing of packets in this sort of network in more detail.

3. SILC Specification

This section describes the SILC protocol. However, [SILC2] and [SILC3] describes other important protocols that are part of this SILC specification and must be read.

3.1 Client

A client is a piece of software connecting to SILC server. SILC client cannot be SILC server. Purpose of clients is to provide the user interface of the SILC services for end user. Clients are distinguished from other clients by unique Client ID. Client ID is a 128 bit ID that is used in the communication in the SILC network. The client ID is based on the user's IP address and nickname. User use logical nicknames in communication which are then mapped to the corresponding Client ID. Client IDs are low level identifications and should not be seen by the end user.

Clients provide other information about the end user as well. Information such as the nickname of the user, username and the host name of the end user and user's real name. See section 3.2 Server for information of the requirements of keeping this information.

The nickname selected by the user is not unique in the SILC network. There can be 2^8 same nicknames for one IP address. As for comparison to IRC [IRC] where nicknames are unique this is a fundamental difference between SILC and IRC. This typically causes the server names or client's host names to be used along with the nicknames on user interface to identify specific users when sending messages. This feature of SILC makes IRC style nickname-wars obsolete as no one owns their nickname; there can always be someone else with the same nickname. Also, any kind of nickname registering service becomes obsolete. See the section 3.13.1 for more information about nicknames.

3.1.1 Client ID

Client ID is used to identify users in the SILC network. The Client ID is unique to the extent that there can be 2^128 different Client IDs, and IDs based on IPv6 addresses extends this to 2^224 different Client IDs. Collisions are not expected to happen. The Client ID is defined as follows.

```
128 bit Client ID based on IPv4 addresses:

32 bit  Server ID IP address (bits 1-32)
 8 bit  Random number or counter
88 bit  Truncated MD5 hash value of the nickname

224 bit Client ID based on IPv6 addresses:

128 bit  Server ID IP address (bits 1-128)
  8 bit  Random number or counter
 88 bit  Truncated MD5 hash value of the nickname
```

- Server ID IP address - Indicates the server where this client is coming from. The IP address hence equals the server IP address where the client is connected.

- Random number or counter - Random number to further randomize the Client ID. Another choice is to use a counter starting from the zero (0). This makes it possible to have 2^8 same nicknames from the same server IP address.

- MD5 hash - MD5 hash value of the case folded nickname is truncated taking 88 bits from the start of the hash value. This hash value is used to search the user's Client ID from the ID lists. Note that the nickname MUST be prepared using the stringprep [RFC3454] profile described in the Appendix A before computing the MD5 hash. See also the section 3.13.1 for more information.

Collisions could occur when more than 2^8 clients using same nickname from the same server IP address is connected to the SILC network. Server MUST be able to handle this situation by refusing to accept anymore of that nickname.

Another possible collision may happen with the truncated hash value of the nickname. It could be possible to have same truncated hash value for two different nicknames. However, this is not expected to happen nor cause any serious problems if it would occur. Nicknames are usually logical and it is unlikely to have two distinct logical nicknames

produce same truncated hash value. Use of MD5 in nickname hash is not a security feature.

3.2 Server

Servers are the most important parts of the SILC network. They form the basis of the SILC, providing a point to which clients may connect to. There are two kinds of servers in SILC; normal servers and router servers. This section focus on the normal server and router server is described in the section 3.3 Router.

Normal servers MUST NOT directly connect to other normal server. Normal servers may only directly connect to router server. If the message sent by the client is destined outside the local server it is always sent to the router server for further routing. Server may only have one active connection to router on same port. Normal server MUST NOT connect to other cell's router except in situations where its cell's router is unavailable.

3.2.1 Server's Local ID List

Normal server keeps various information about the clients and their end users connected to it. Every normal server MUST keep list of all locally connected clients, Client IDs, nicknames, usernames and host names and user's real name. Normal servers only keeps local information and it does not keep any global information. Hence, normal servers knows only about their locally connected clients. This makes servers efficient as they do not have to worry about global clients. Server is also responsible of creating the Client IDs for their clients.

Normal server also keeps information about locally created channels and their Channel IDs.

Hence, local list for normal server includes:

server list - Router connection
- o Server name
- o Server IP address
- o Server ID
- o Sending key
- o Receiving key
- o Public key

client list - All clients in server
- o Nickname
- o Username@host
- o Real name

- o Client ID
- o Sending key
- o Receiving key
- o Public key

channel list - All channels in server

- o Channel name
- o Channel ID
- o Client IDs on channel
- o Client ID modes on channel
- o Channel key

3.2.2 Server ID

Servers are distinguished from other servers by unique 64 bit Server ID (for IPv4) or 160 bit Server ID (for IPv6). The Server ID is used in the SILC to route messages to correct servers. Server IDs also provide information for Client IDs, see section 3.1.1 Client ID. Server ID is defined as follows.

64 bit Server ID based on IPv4 addresses:

```
32 bit  IP address of the server
16 bit  Port
16 bit  Random number
```

160 bit Server ID based on IPv6 addresses:

```
128 bit  IP address of the server
 16 bit  Port
 16 bit  Random number
```

- o IP address of the server - This is the real IP address of the server.
- o Port - This is the port the server is bound to.
- o Random number - This is used to further randomize the Server ID.

Collisions are not expected to happen in any conditions. The Server ID is always created by the server itself and server is responsible of distributing it to the router.

3.2.3 SILC Server Ports

The following ports has been assigned by IANA for the SILC protocol:

```
silc            706/tcp    SILC
silc            706/udp    SILC
```

If there are needs to create new SILC networks in the future the port numbers must be officially assigned by the IANA.

Server on network above privileged ports (>1023) SHOULD NOT be trusted as they could have been set up by untrusted party.

3.3 Router

Router server in SILC network is responsible for keeping the cell together and routing messages to other servers and to other routers. Router server may also act as normal server when clients may connect to it. This is not requirement and router servers may be hidden from clients.

However, router servers have a lot of important tasks that normal servers do not have. Router server knows everything and keeps the global state. They know all clients currently on SILC, all servers and routers and all channels in SILC. Routers are the only servers in SILC that care about global information and keeping them up to date at all time.

3.3.1 Router's Local ID List

Router server as well MUST keep local list of connected clients and locally created channels. However, this list is extended to include all the informations of the entire cell, not just the server itself as for normal servers.

However, on router this list is a lot smaller since routers do not need to keep information about user's nickname, username and host name and real name since these are not needed by the router. The router keeps only information that it needs.

Hence, local list for router includes:

```
server list        - All servers in the cell
   o Server name
   o Server ID
   o Router's Server ID
   o Sending key
   o Receiving key

client list        - All clients in the cell
   o Client ID
```

```
channel list       - All channels in the cell
   o Channel ID
   o Client IDs on channel
   o Client ID modes on channel
   o Channel key
```

Note that locally connected clients and other information include all the same information as defined in section section 3.2.1 Server's Local ID List. Router MAY also cache same detailed information for other clients if needed.

3.3.2 Router's Global ID List

Router server MUST also keep global list. Normal servers do not have global list as they know only about local information. Global list includes all the clients on SILC, their Client IDs, all created channels and their Channel IDs and all servers and routers on SILC and their Server IDs. That is said, global list is for global information and the list must not include the local information already on the router's local list.

Note that the global list does not include information like nicknames, usernames and host names or user's real names. Router does not need to keep these informations as they are not needed by the router. This information is available from the client's server which maybe queried when needed.

Hence, global list includes:

```
server list        - All servers in SILC
   o Server name
   o Server ID
   o Router's Server ID

client list        - All clients in SILC
   o Client ID

channel list       - All channels in SILC
   o Channel ID
   o Client IDs on channel
   o Client ID modes on channel
```

3.3.3 Router's Server ID

Router's Server ID is equivalent to normal Server ID. As routers are normal servers same types of IDs applies for routers as well. See section 3.2.2 Server ID.

3.4 Channels

A channel is a named group of one or more clients which will all receive messages addressed to that channel. The channel is created when first client requests JOIN command to the channel, and the channel ceases to exist when the last client has left it. When channel exists, any client can reference it using the Channel ID of the channel. If the channel has a founder mode set and last client leaves the channel the channel does not cease to exist. The founder mode can be used to make permanent channels in the network. The founder of the channel can regain the channel founder privileges on the channel later when he joins the channel.

Channel names are unique although the real uniqueness comes from 64 bit Channel ID. However, channel names are still unique and no two global channels with same name may exist. See the section 3.13.1 for more information about channel names.

Channels can have operators that can administrate the channel and operate all of its modes. The following operators on channel exist on the SILC network.

- Channel founder - When channel is created the joining client becomes channel founder. Channel founder is channel operator with some more privileges. Basically, channel founder can fully operate the channel and all of its modes. The privileges are limited only to the particular channel. There can be only one channel founder per channel. Channel founder supersedes channel operator's privileges.

  Channel founder privileges cannot be removed by any other operator on channel. When channel founder leaves the channel there is no channel founder on the channel. However, it is possible to set a mode for the channel which allows the original channel founder to regain the founder privileges even after leaving the channel. Channel founder also cannot be removed by force from the channel.

- Channel operator - When client joins to channel that has not existed previously it will become automatically channel operator (and channel founder discussed above). Channel operator is able to administrate the

channel, set some modes on channel, remove a badly behaving client from the channel and promote other clients to become channel operator. The privileges are limited only to the particular channel.

Normal channel user may be promoted (opped) to channel operator gaining channel operator privileges. Channel founder or other channel operator may also demote (deop) channel operator to normal channel user.

3.4.1 Channel ID

Channels are distinguished from other channels by unique Channel ID. The Channel ID is a 64 bit ID (for IPv4) or 160 bit ID (for IPv6), and collisions are not expected to happen in any conditions. Channel names are just for logical use of channels. The Channel ID is created by the server where the channel is created. The Channel ID is defined as follows.

64 bit Channel ID based on IPv4 addresses:

32 bit Router's Server ID IP address (bits 1-32)
16 bit Router's Server ID port (bits 33-48)
16 bit Random number or counter

160 bit Channel ID based on IPv6 addresses:

128 bit Router's Server ID IP address (bits 1-128)
16 bit Router's Server ID port (bits 129-144)
16 bit Random number or counter

o Router's Server ID IP address - Indicates the IP address of the router of the cell where this channel is created. This is taken from the router's Server ID. This way SILC routers know where this channel resides in the SILC network.

o Router's Server ID port - Indicates the port of the channel on the server. This is taken from the router's Server ID.

o Random number or counter - To further randomize the Channel ID. Another choice is to use a counter starting from zero (0). This makes sure that there are no collisions. This also means that in a cell there can be 2^16 different channels.

3.5 Operators

Operators are normal users with extra privileges to their server or router. Usually these people are SILC server and router administrators that take care of their own server and clients on them. The purpose of operators is to administrate the SILC server or router. However, even an operator with highest privileges is not able to enter invite-only channels, to gain access to the contents of encrypted and authenticated packets traveling in the SILC network or to gain channel operator privileges on public channels without being promoted. They have the same privileges as any normal user except they are able to administrate their server or router.

3.6 SILC Commands

Commands are very important part on SILC network especially for client which uses commands to operate on the SILC network. Commands are used to set nickname, join to channel, change modes and many other things.

Client usually sends the commands and server replies by sending a reply packet to the command. Server MAY also send commands usually to serve the original client's request. Usually server cannot send commands to clients, however there MAY be commands that allow the server to send commands to client. By default servers MAY send commands only to other servers and routers.

Note that the command reply is usually sent only after client has sent the command request but server is allowed to send command reply packet to client even if client has not requested the command. Client MAY choose to ignore the command reply.

It is expected that some of the commands may be misused by clients resulting various problems on the server side. Every implementation SHOULD assure that commands may not be executed more than once, say, in two (2) seconds. However, to keep response rate up, allowing for example five (5) commands before limiting is allowed. It is RECOMMENDED that commands such as SILC_COMMAND_NICK, SILC_COMMAND_JOIN, SILC_COMMAND_LEAVE and SILC_COMMAND_KILL SHOULD be limited in all cases as they require heavy operations. This should be sufficient to prevent the misuse of commands.

SILC commands are described in [SILC4].

3.7 SILC Packets

Packets are naturally the most important part of the protocol and the

packets are what actually makes the protocol. Packets in SILC network are always encrypted using, usually the shared secret session key or some other key, for example, channel key, when encrypting channel messages. It is not possible to send a packet in SILC network without encryption. The SILC Packet Protocol is a wide protocol and is described in [SILC2]. This document does not define or describe details of SILC packets.

3.8 Packet Encryption

All packets passed in SILC network MUST be encrypted. This section gives generic description of how packets must be encrypted in the SILC network. The detailed description of the actual encryption process of the packets are described in [SILC2].

Client and its server shares secret symmetric session key which is established by the SILC Key Exchange Protocol, described in [SILC3]. Every packet sent from client to server, with exception of packets for channels, are encrypted with this session key.

Channels have a channel key that are shared by every client on the channel. However, the channel keys are cell specific thus one cell does not know the channel key of the other cell, even if that key is for same channel. Channel key is also known by the routers and all servers that have clients on the channel. However, channels MAY have channel private keys that are entirely local setting for the client. All clients on the channel MUST know the channel private key beforehand to be able to talk on the channel. In this case, no server or router knows the key for the channel.

Server shares secret symmetric session key with router which is established by the SILC Key Exchange Protocol. Every packet passed from server to router, with exception of packets for channels, are encrypted with the shared session key. Same way, router server shares secret symmetric key with its primary router. However, every packet passed from router to other router, including packets for channels, are encrypted with the shared session key. Every router connection MUST have their own session keys.

3.8.1 Determination of the Source and the Destination

The source and the destination of the packet needs to be determined to be able to route the packets to correct receiver. This information is available in the SILC Packet Header which is included in all packets sent in SILC network. The SILC Packet Header is described in [SILC2].

The header MUST be encrypted with the session key of who is the next

receiver of the packet along the route. The receiver of the packet, for example a router along the route, is able to determine the sender and the destination of the packet by decrypting the SILC Packet Header and checking the IDs attached to the header. The IDs in the header will tell to where the packet needs to be sent and where it is coming from.

The header in the packet MUST NOT change during the routing of the packet. The original sender, for example client, assembles the packet and the packet header and server or router between the sender and the receiver MUST NOT change the packet header. Note however, that some packets such as commands may be resent by a server to serve the client's original command. In this case the command packet sent by the server includes the server's IDs as it is a different packet. When server or router receives a packet it MUST verify that the Source ID is valid and correct ID for that sender.

Note that the packet and the packet header may be encrypted with different keys. For example, packets to channels are encrypted with the channel key, however, the header is encrypted with the session key as described above. Most other packets have both header and packet payload encrypted with the same key, such as command packets.

3.8.2 Client To Client

The process of message delivery and encryption from client to another client is as follows.

Example: Private message from client to another client on different servers. Clients do not share private message delivery keys; normal session keys are used.

- Client 1 sends encrypted packet to its server. The packet is encrypted with the session key shared between client and its server.

- Server determines the destination of the packet and decrypts the packet. Server encrypts the packet with session key shared between the server and its router, and sends the packet to the router.

- Router determines the destination of the packet and decrypts the packet. Router encrypts the packet with session key shared between the router and the destination server, and sends the packet to the server.

- Server determines the client to which the packet is destined to and decrypts the packet. Server encrypts the packet with

session key shared between the server and the destination client, and sends the packet to the client.

- Client 2 decrypts the packet.

Example: Private message from client to another client on different servers. Clients have established a secret shared private message delivery key with each other and that is used in the message encryption.

- Client 1 sends encrypted packet to its server. The packet header is encrypted with the session key shared between the client and server, and the private message payload is encrypted with the private message delivery key shared between clients.

- Server determines the destination of the packet and sends the packet to the router. Header is encrypted with the session key.

- Router determines the destination of the packet and sends the packet to the server. Header is encrypted with the session key.

- Server determines the client to which the packet is destined to and sends the packet to the client. Header is encrypted with the session key.

- Client 2 decrypts the packet with the secret shared key.

If clients share secret key with each other the private message delivery is much simpler since servers and routers between the clients do not need to decrypt and re-encrypt the entire packet. The packet header however is always encrypted with session key and is decrypted and re-encrypted with the session key of next recipient.

The process for clients on same server is much simpler as there is no need to send the packet to the router. The process for clients on different cells is same as above except that the packet is routed outside the cell. The router of the destination cell routes the packet to the destination same way as described above.

3.8.3 Client To Channel

Process of message delivery from client on channel to all the clients on the channel.

Example: Channel of four users; two on same server, other two on different cells. Client sends message to the channel.

Packet header is encrypted with the session key, message data is encrypted with channel key.

- Client 1 encrypts the packet with channel key and sends the packet to its server.
- Server determines local clients on the channel and sends the packet to the Client on the same server. Server then sends the packet to its router for further routing.
- Router determines local clients on the channel, if found sends packet to the local clients. Router determines global clients on the channel and sends the packet to its primary router or fastest route.
- (Other router(s) do the same thing and sends the packet to the server(s).)
- Server determines local clients on the channel and sends the packet to the client.
- All clients receiving the packet decrypts it.

3.8.4 Server To Server

Server to server packet delivery and encryption is described in above examples. Router to router packet delivery is analogous to server to server. However, some packets, such as channel packets, are processed differently. These cases are described later in this document and more in detail in [SILC2].

3.9 Key Exchange And Authentication

Key exchange is done always when for example client connects to server but also when server and router, and router and another router connect to each other. The purpose of key exchange protocol is to provide secure key material to be used in the communication. The key material is used to derive various security parameters used to secure SILC packets. The SILC Key Exchange protocol is described in detail in [SILC3].

Authentication is done after key exchange protocol has been successfully completed. The purpose of authentication is to authenticate for example client connecting to the server. However, clients MAY be accepted to connect to server without explicit authentication. Servers are REQUIRED to use authentication protocol when connecting. The authentication may be based on passphrase (pre-shared secret) or public

key based on digital signatures. All passphrases sent in SILC protocol MUST be UTF-8 [RFC3629] encoded. The connection authentication protocol is described in detail in [SILC3].

3.9.1 Authentication Payload

Authentication Payload is used separately from the SKE and the Connection Authentication protocols. It can be used during the session to authenticate with a remote. For example, a client can authenticate itself to a server to become server operator. In this case, Authentication Payload is used.

The format of the Authentication Payload is as follows:

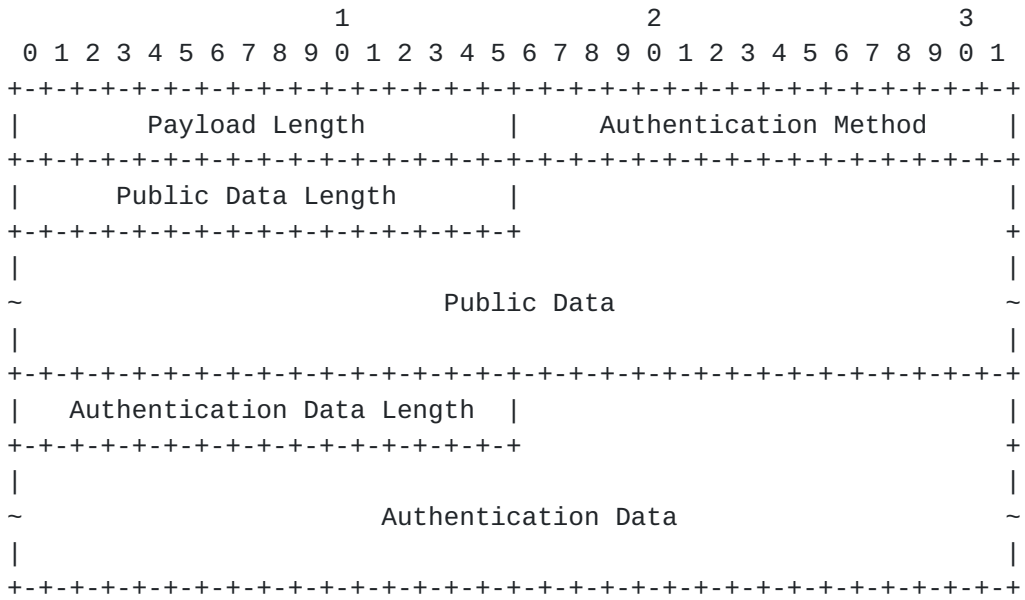

Figure 5: Authentication Payload

- Payload Length (2 bytes) - Length of the entire payload.

- Authentication Method (2 bytes) - The method of the authentication. The authentication methods are defined in [SILC2] in the Connection Auth Request Payload. The NONE authentication method SHOULD NOT be used.

- Public Data Length (2 bytes) - Indicates the length of the Public Data field.

- Public Data (variable length) - This is defined only if the authentication method is public key. If it is any other

this field MAY include random data for padding purposes. However, in this case the field MUST be ignored by the receiver.

When the authentication method is public key this includes 128 to 4096 bytes of non-zero random data that is used in the signature process, described subsequently.

- Authentication Data Length (2 bytes) - Indicates the length of the Authentication Data field. If zero (0) value is found in this field the payload MUST be discarded.

- Authentication Data (variable length) - Authentication method dependent authentication data.

If the authentication method is passphrase-based, the Authentication Data field includes the plaintext UTF-8 encoded passphrase. It is safe to send plaintext passphrase since the entire payload is encrypted. In this case the Public Data Length is set to zero (0), but MAY also include random data for padding purposes. It is also RECOMMENDED that maximum amount of padding is applied to SILC packet when using passphrase-based authentication. This way it is not possible to approximate the length of the passphrase from the encrypted packet.

If the authentication method is public key based (or certificate) the Authentication Data is computed as follows:

```
HASH = hash(random bytes | ID | public key (or certificate));
Authentication Data = sign(HASH);
```

The hash() and the sign() are the hash function and the public key cryptography function selected in the SKE protocol, unless otherwise stated in the context where this payload is used. The public key is SILC style public key unless certificates are used. The ID is the entity's ID (Client or Server ID) which is authenticating itself. The ID encoding is described in [SILC2]. The random bytes are non-zero random bytes of length between 128 and 4096 bytes, and will be included into the Public Data field as is.

The receiver will compute the signature using the random data received in the payload, the ID associated to the connection and the public key (or certificate) received in the SKE protocol. After computing the receiver MUST verify the signature. Also in case of public key authentication this payload is always encrypted. This payload is always sent as part of some other payload.

3.10 Algorithms

This section defines all the allowed algorithms that can be used in the SILC protocol. This includes mandatory cipher, mandatory public key algorithm and MAC algorithms.

3.10.1 Ciphers

Cipher is the encryption algorithm that is used to protect the data in the SILC packets. See [SILC2] for the actual encryption process and definition of how it must be done. SILC has a mandatory algorithm that must be supported in order to be compliant with this protocol.

The following ciphers are defined in SILC protocol:

aes-256-cbc	AES in CBC mode, 256 bit key	(REQUIRED)
aes-256-ctr	AES in CTR mode, 256 bit key	(RECOMMENDED)
aes-256-rcbc	AES in randomized CBC mode, 256 bit key	(OPTIONAL)
aes-192-<mode>	AES in <mode> mode, 192 bit key	(OPTIONAL)
aes-128-<mode>	AES in <mode> mode, 128 bit key	(RECOMMENDED)
twofish-256-<mode>	Twofish in <mode> mode, 256 bit key	(OPTIONAL)
twofish-192-<mode>	Twofish in <mode> mode, 192 bit key	(OPTIONAL)
twofish-128-<mode>	Twofish in <mode> mode, 128 bit key	(OPTIONAL)
cast-256-<mode>	CAST-256 in <mode> mode, 256 bit key	(OPTIONAL)
cast-192-<mode>	CAST-256 in <mode> mode, 192 bit key	(OPTIONAL)
cast-128-<mode>	CAST-256 in <mode> mode, 128 bit key	(OPTIONAL)
serpent-<len>-<mode>	Serpent in <mode> mode, <len> bit key	(OPTIONAL)
rc6-<len>-<mode>	RC6 in <mode> mode, <len> bit key	(OPTIONAL)
mars-<len>-<mode>	MARS in <mode> mode, <len> bit key	(OPTIONAL)
none	No encryption	(OPTIONAL)

The <mode> is either "cbc", "ctr" or "rcbc". Other encryption modes MAY be defined to be used in SILC using the same name format. The <len> is either 256, 192 or 128 bit key length. Also, additional ciphers MAY be defined to be used in SILC by using the same name format as above.

Algorithm "none" does not perform any encryption process at all and thus is not recommended to be used. It is recommended that no client or server implementation would accept "none" algorithm except in special debugging mode.

3.10.1.1 CBC Mode

The "cbc" encryption mode is the standard cipher-block chaining mode. The very first IV is derived from the SILC Key Exchange protocol. Subsequent IVs for encryption is the previous ciphertext block. The very

first IV MUST be random and is generated as described in [SILC3].

3.10.1.2 CTR Mode

The "ctr" encryption mode is Counter Mode (CTR). The CTR mode in SILC is stateful in encryption and decryption. Both sender and receiver maintain the counter for the CTR mode and thus can precompute the key stream for encryption and decryption. By default, CTR mode does not require plaintext padding, however implementations MAY apply padding to the packets. If the last key block is larger than the last plaintext block the resulted value is truncated to the size of the plaintext block and the most significant bits are used. When sending authentication data inside packets the maximum amount of padding SHOULD be applied with CTR mode as well.

In CTR mode only the encryption operation of the cipher is used. The decryption operation is not needed since both encryption and decryption process is simple XOR with the plaintext block and the key stream block.

The counter block is used to create the key for the CTR mode. The format of the 128 bit counter block is as follows:

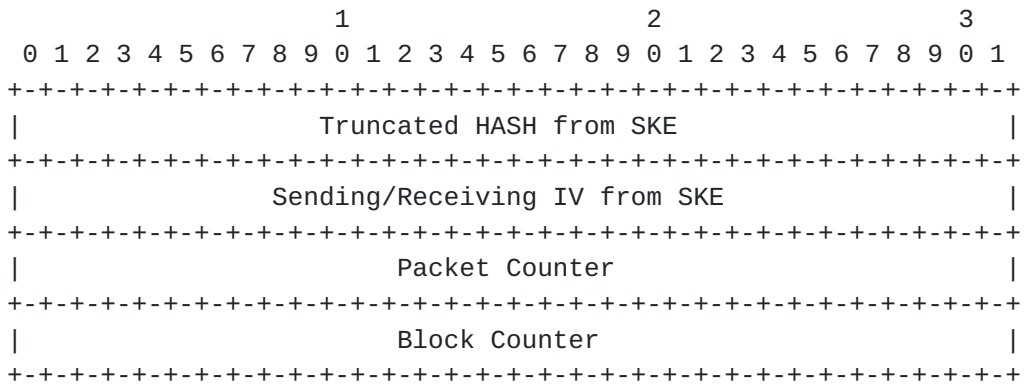

Figure 6: Counter Block

- Truncated HASH from SKE (4 bytes) - This value is the first 4 bytes from the HASH value that was computed as a result of SKE protocol. This acts as session identifier and each rekey MUST produce a new HASH value.

- Sending/Receiving IV from SKE (4 bytes) - If the CTR mode is fully stateful this field MUST include the first 4 bytes from the Sending IV or Receiving IV generated in SKE protocol. When this mode is used to encrypt sending traffic the Sending IV is used, when used to decrypt receiving traffic the Receiving IV is used. This assures that two parties of the protocol use different IV for sending

traffic. Each rekey MUST produce a new value.

If the IV Included flag is negotiated in SKE or CTR mode is used where the IV is included in the data payload, this field is the Nonce field from the IV received in the packet, defined below.

- Packet Counter (4 bytes) - This is MSB first ordered monotonically increasing packet counter. It is set value 1 for first packet and increases for subsequent packets. After rekey the counter MUST restart from 1.

  If the IV Included flag is negotiated in SKE or CTR mode is used where the IV is included in the data payload, this field is the Packet Counter field from the IV received in the packet, defined below.

- Block Counter (4 bytes) - This is an MSB first ordered block counter starting from 1 for first block and increasing for subsequent blocks. The counter is always set to value 1 for a new packet.

CTR mode MUST NOT be used with "none" MAC. Implementations also MUST assure that the same counter block is not used to encrypt more than one block. None of the counters must be allowed to wrap without rekey. Also, the key material used with CTR mode MUST be fresh key material. Static keys (pre-shared keys) MUST NOT be used with CTR mode. For this reason using CTR mode to encrypt for example channel messages or private messages with a pre-shared key is inappropriate. For private messages, the Key Agreement [SILC2] could be performed to produce fresh key material.

If the IV Included flag was negotiated in SKE, or CTR mode is used to protect channel messages where the IV will be included in the Message Payload, the Initialization Vector (IV) to be used is a 64-bit block containing randomness and packet counter. Also note, that in this case the decryption process is not stateful and receiver cannot precompute the key stream. Hence, the Initialization Vector (IV) when CTR mode is used is as follows.

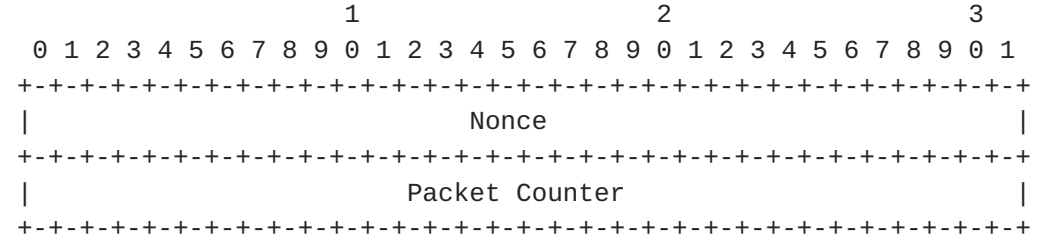

Figure 7: CTR Mode Initialization Vector

- Nonce (4 bytes) - This field should be random or otherwise not easily determinable and SHOULD change for each packet.

- Packet Counter (4 bytes) - This is MSB first ordered monotonically increasing packet counter. It is set value 1 for first packet and increases for subsequent packets. After rekey the counter MUST restart from 1.

When decrypting the packet the Counter Block is assembled by concatenating the truncated hash, with the received nonce and packet counter, and with the block counter. The Counter Block is then used to compute the key stream to perform the decryption.

3.10.1.3 Randomized CBC Mode

The "rcbc" encryption mode is CBC mode with randomized IV. This means that each IV for each packet MUST be chosen randomly. When encrypting more than one block the normal IV chaining is used, but for the first block new random IV is selected in each packet. In this mode the IV is appended to the ciphertext. If this mode is used to secure the SILC session, the IV Included flag must be negotiated in SILC Key Exchange protocol. It may also be used to secure Message Payloads which can deliver the IV to the recipient.

3.10.2 Public Key Algorithms

Public keys are used in SILC to authenticate entities in SILC network and to perform other tasks related to public key cryptography. The public keys are also used in the SILC Key Exchange protocol [SILC3].

The following public key algorithms are defined in SILC protocol:

```
rsa        RSA  (REQUIRED)
dss        DSS  (OPTIONAL)
```

DSS is described in [Menezes]. The RSA MUST be implemented according PKCS #1 [PKCS1]. When using SILC Public Key version 2 the PKCS #1 implementation MUST be compliant with PKCS #1 version 1.5. The signatures are computed with appendix; the hash OID is included in the signature. The user may always select the hash algorithm for the signatures. When using SILC Public Key version 1 the PKCS #1 implementation MUST be compliant with PKCS #1 version 1.5 where signatures are computed without appendix; the hash OID is not present in the signature. The hash algorithm used is specified separately or the default hash algorithm is used, as defined below.

Additional public key algorithms MAY be defined to be used in SILC.

When signatures are computed in SILC the computing of the signature is denoted as sign(). The signature computing procedure is dependent of the public key algorithm, and the public key or certificate encoding. When using SILC public key the signature is computed as described in previous paragraph for RSA and DSS keys. If the hash function is not specified separately for signing process SHA-1 MUST be used, except with SILC public key version 2 and RSA algorithm when the user MAY always select the hash algorithm. In this case the hash algorithm is included in the signature and can be retrieved during verification. When using SSH2 public keys the signature is computed as described in [SSH-TRANS]. When using X.509 version 3 certificates the signature is computed as described in [PKCS7]. When using OpenPGP certificates the signature is computed as described in [PGP] and the PGP signature type used is 0x00.

3.10.2.1 Multi-Precision Integers

Multi-Precision (MP) integers in SILC are encoded and decoded as defined in PKCS #1 [PKCS1]. MP integers are unsigned, encoded with the exact octet length of the integer. No extra leading zero octets may appear. The actual length of the integer is the bit size of the integer not counting any leading zero bits. The octet length is derived by calculating (bit_length + 7) / 8.

3.10.3 Hash Functions

Hash functions are used as part of MAC algorithms defined in the next section. They are also used in the SILC Key Exchange protocol defined in the [SILC3].

The following Hash algorithm are defined in SILC protocol:

sha1	SHA-1, length = 20 bytes	(REQUIRED)
sha256	SHA-256, length = 32 bytes	(RECOMMENDED)
md5	MD5, length = 16 bytes	(RECOMMENDED)

3.10.4 MAC Algorithms

Data integrity is protected by computing a message authentication code (MAC) of the packet data. See [SILC2] for details how to compute the MAC for a packet.

The following MAC algorithms are defined in SILC protocol:

```
   hmac-sha1-96     HMAC-SHA1, length = 12 bytes   (REQUIRED)
   hmac-sha256-96   HMAC-SHA256, length = 12 bytes (RECOMMENDED)
   hmac-md5-96      HMAC-MD5, length = 12 bytes    (OPTIONAL)
   hmac-sha1        HMAC-SHA1, length = 20 bytes   (OPTIONAL)
   hmac-sha256      HMAC-SHA256, length = 32 bytes (OPTIONAL)
   hmac-md5         HMAC-MD5, length = 16 bytes    (OPTIONAL)
   none             No MAC                         (OPTIONAL)
```

The "none" MAC is not recommended to be used as the packet is not authenticated when MAC is not computed. It is recommended that no client or server would accept none MAC except in special debugging mode.

The HMAC algorithm is described in [HMAC]. The hash algorithms used in HMACs, the SHA-1 is described in [RFC3174] and MD5 is described in [RFC1321]. The SHA-256 algorithm and its used with HMAC is described in [SHA256].

Additional MAC algorithms MAY be defined to be used in SILC.

3.10.5 Compression Algorithms

SILC protocol supports compression that may be applied to unencrypted data. It is recommended to use compression on slow links as it may significantly speed up the data transmission. By default, SILC does not use compression which is the mode that must be supported by all SILC implementations.

The following compression algorithms are defined:

```
   none        No compression               (REQUIRED)
   zlib        GNU ZLIB (LZ77) compression  (OPTIONAL)
```

Additional compression algorithms MAY be defined to be used in SILC.

3.11 SILC Public Key

This section defines the type and format of the SILC public key. All implementations MUST support this public key type. See [SILC3] for other optional public key and certificate types allowed in the SILC protocol. Public keys in SILC may be used to authenticate entities and to perform other tasks related to public key cryptography.

The format of the SILC Public Key is as follows:

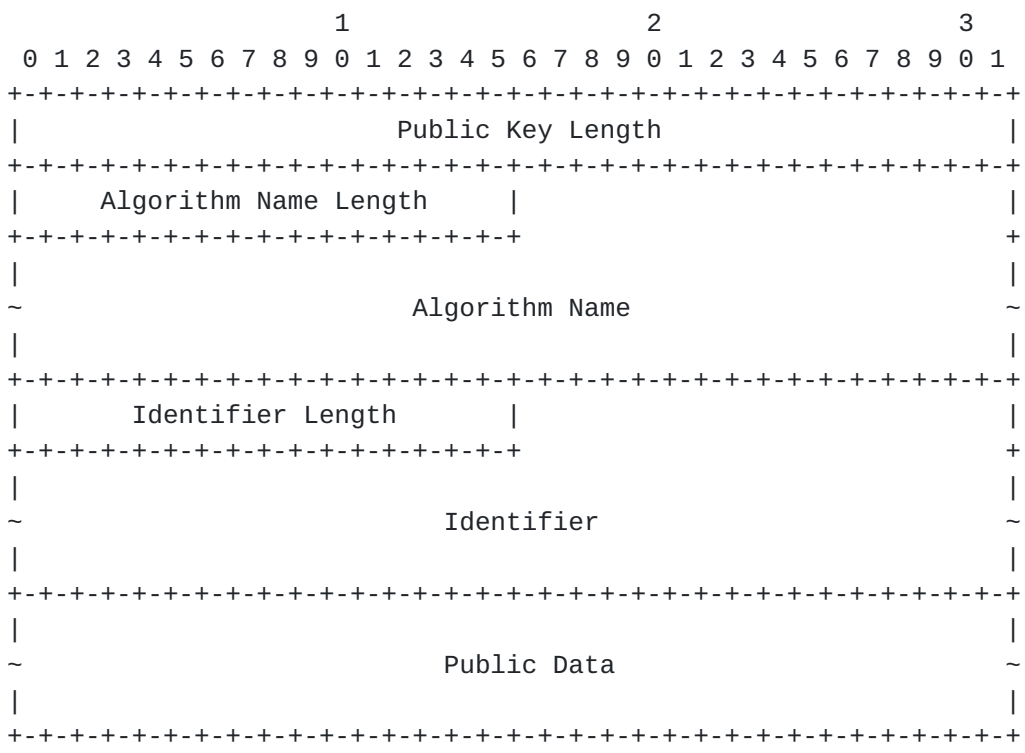

Figure 5: SILC Public Key

o Public Key Length (4 bytes) - Indicates the full length of the SILC Public Key, not including this field.

o Algorithm Name Length (2 bytes) - Indicates the length of the Algorithm Length field, not including this field.

o Algorithm name (variable length) - Indicates the name of the public key algorithm that the key is. See the section 3.10.2 Public Key Algorithms for defined names.

o Identifier Length (2 bytes) - Indicates the length of the Identifier field, not including this field.

o Identifier (variable length) - Indicates the identifier of the public key. This data can be used to identify the owner of the key. The identifier may be of the following format:

UN	User name
HN	Host name or IP address
RN	Real name
E	EMail address
O	Organization

```
   C      Country
   V      Version
```

Examples of an identifier:

```
  'UN=priikone, HN=poseidon.pspt.fi, E=priikone@poseidon.pspt.fi'

  'UN=sam, HN=dummy.fi, RN=Sammy Sam, C=Finland, V=2'
```

At least user name (UN) and host name (HN) MUST be provided as identifier. The fields are separated by commas (','). If comma is in the identifier string it must be escaped as '\,', for example, 'O=Company XYZ\, Inc.'. Other characters that require escaping are listed in [RFC2253] and are to be escaped as defined therein. The Version (V) may only be a decimal digit.

o Public Data (variable length) - Includes the actual public data of the public key.

The format of this field for RSA algorithm is as follows:

```
   4 bytes            Length of e
   variable length    e
   4 bytes            Length of n
   variable length    n
```

The format of this field for DSS algorithm is as follows:

```
   4 bytes            Length of p
   variable length    p
   4 bytes            Length of q
   variable length    q
   4 bytes            Length of g
   variable length    g
   4 bytes            Length of y
   variable length    y
```

The variable length fields are multiple precession integers encoded as strings in both examples.

Other algorithms must define their own type of this field if they are used.

The SILC Public Key is version is 2. If the Version (V) identifier is not present the SILC Public Key version is expected to be 1. All new

implementations SHOULD support version 1 but SHOULD only generate version 2. In this case the Version (V) identifier MUST be present.

All fields in the public key are in MSB (most significant byte first) order. All strings in the public key MUST be UTF-8 encoded.

If an external protocol needs to refer to SILC Public Key by name, the names "silc-rsa" and "silc-dss" for SILC Public Key based on RSA algorithm and SILC Public Key based on DSS algorithm, respectively, are to be used. However, this SILC specification does not use these names directly, and they are defined here for external protocols (protocols that may like to use SILC Public Key).

A fingerprint from SILC Public Key is computed from the whole encoded public key data block. All fields are included in computation. Compliant implementations MUST support computing a 160-bit SHA-1 fingerprint.

3.12 SILC Version Detection

The version detection of both client and server is performed at the connection phase while executing the SILC Key Exchange protocol. The version identifier is exchanged between initiator and responder. The version identifier is of the following format:

```
SILC-<protocol version>-<software version>
```

The version strings are of the following format:

```
protocol version = <major>.<minor>
software version = <major>[.<minor>[.<build or vendor string>]]
```

Protocol version MUST provide both major and minor version. Currently implementations MUST set the protocol version and accept at least the protocol version as SILC-1.2-<software version>. If new protocol version causes incompatibilities with older version the <minor> version number MUST be incremented. The <major> is incremented if new protocol version is fully incompatible.

Software version MAY provide major, minor and build (vendor) version. The software version MAY be freely set and accepted. The version string MUST consist of printable US-ASCII characters.

Thus, the version strings could be, for example:

```
SILC-1.1-2.0.2
SILC-1.0-1.2
SILC-1.2-1.0.VendorXYZ
```

3.13 UTF-8 Strings in SILC

By default all strings that are sent in SILC protocol MUST be UTF-8 [RFC3269] encoded, unless otherwise defined. This means that any string sent inside for example, command, command reply, notify or any packet payload is UTF-8 encoded. Also nicknames, channel names, server names, and hostnames are UTF-8 encoded. This definition does not affect messages sent in SILC, as the Message Payload provides its own mechanism to indicate whether a message is UTF-8 text message, data message, which may use its own character encoding, or pure binary message [SILC2].

Certain limitations are imposed on the UTF-8 encoded strings in SILC. The UTF-8 encoded strings MUST NOT include any characters that are marked in the Unicode standard as control codes, noncharacters, reserved or private range characters, or any other illegal Unicode characters. Also the BOM (Byte-Order Mark) MUST NOT be used as byte order signature in UTF-8 encoded strings. A string containing these characters MUST be treated as malformed UTF-8 encoding.

The Unicode standard defines that malformed sequences shall be signalled by replacing the sequence with a replacement character. Even though, in case of SILC these strings may not be malformed UTF-8 encodings they MUST be treated as malformed strings. Implementation MAY use a replacement character, however, the character Unicode standard defines MUST NOT be used, but another character must be chosen. It is, however, RECOMMENDED that an error is returned instead of using replacement character if it is possible. For example, when setting a nickname with SILC_COMMAND_NICK command, implementation is able to send error indication back to the command sender. It must be noted that on server implementation if a character sequence is merely outside of current character subset, but is otherwise valid character, it MUST NOT be replaced by a replacement character.

On user interface where UTF-8 strings are displayed the implementation is RECOMMENDED to escape any character that it is unable to render properly. The escaping may be done for example as described in [RFC2253]. The escaping makes it possible to retrieve the original UTF-8 encoding. Alternatively, a replacement character may be used if it does not cause practical problems to the implementation.

3.13.1 UTF-8 Identifier Strings

Identifier strings are special strings in SILC protocol that require more careful processing, than the general UTF-8 strings described in the

previous section. These strings include the nicknames, server names, hostnames and some other identifier strings. These strings are prepared using the stringprep [RFC3454] standard. The Appendix A defines the stringprep profile for SILC identifier strings and conforming implementation MUST use the profile to prepare any identifier string.

The stringprep profile describes how identifier strings are prepared, what characters they may include, and which characters are prohibited. Identifier strings with prohibited characters MUST be treated as malformed strings.

The channel name is also special identifier strings with some slight differences to other identifier strings. The Appendix B defines the stringprep profile for the channel name strings and conforming implementation MUST use the profile to prepare any channel name string.

Because of the profile the identifier strings in SILC may generally include only letters, numbers, most punctuation characters, and some other characters. For practical reasons most symbol characters and many other special characters are prohibited. All identifier strings are case folded and comparing the identifier strings MUST be done as caseless matching.

In general, the identifier strings does not have a maximum length. However, the length of a nickname string MUST NOT exceed 128 bytes, and the length of a channel name string MUST NOT exceed 256 bytes. Since these strings are UTF-8 encoded the length of one character may be longer than one byte. This means that the character length of these strings may be shorter than the maximum length of the string in bytes. The minimum length of an identifier string MUST be at least one character, which may be one byte or more in length. Implementation MAY limit the maximum length of an identifier string, with exception of the nickname and channel name strings which has the explicit length definition.

3.14 Backup Routers

Backup routers may exist in the cell in addition to the primary router. However, they must not be active routers or act as routers in the cell. Only one router may be acting as primary router in the cell. In the case of failure of the primary router one of the backup routers becomes active. The purpose of backup routers are in case of failure of the primary router to maintain working connections inside the cell and outside the cell and to avoid netsplits.

Backup routers are normal servers in the cell that are prepared to take over the tasks of the primary router if needed. They need to have at least one direct and active connection to the primary router of the cell.

This communication channel is used to send the router information to the backup router. When the backup router connects to the primary router of the cell it MUST present itself as router server in the Connection Authentication protocol, even though it is normal server as long as the primary router is available. Reason for this is that the configuration needed in the responder end requires usually router connection level configuration. The responder, however must understand and treat the connection as normal server (except when feeding router level data to the backup router).

Backup router must know everything that the primary router knows to be able to take over the tasks of the primary router. It is the primary router's responsibility to feed the data to the backup router. If the backup router does not know all the data in the case of failure some connections may be lost. The primary router of the cell must consider the backup router being an actual router server when it feeds the data to it.

In addition to having direct connection to the primary router of the cell, the backup router must also have connection to the same router to which the primary router of the cell is connected. However, it must not be the active router connection meaning that the backup router must not use that channel as its primary route and it must not notify the router about having connected servers, channels and clients behind it. It merely connects to the router. This sort of connection is later referred to as being a passive connection. Some keepalive actions may be needed by the router to keep the connection alive.

It is required that other normal servers have passive connections to the backup router(s) in the cell. Some keepalive actions may be needed by the server to keep the connection alive. After they notice the failure of the primary router they must start using the connection to the first backup router as their primary route.

Also, if any other router in the network is using the cell's primary router as its own primary router, it must also have passive connection to the cell's backup router. It too is prepared to switch to use the backup router as its new primary router as soon as the original primary router becomes unresponsive.

All of the parties of this protocol know which one is the backup router of the cell from their local configuration. Each of the entities must be configured accordingly and care must be taken when configuring the backup routers, servers and other routers in the network.

It must be noted that some of the channel messages and private messages may be lost during the switch to the backup router, unless the message flag SILC_MESSAGE_FLAG_ACK is set in the message. The announcements

assure that the state of the network is not lost during the switch.

It is RECOMMENDED that there would be at least one backup router in the cell. It is NOT RECOMMENDED to have all servers in the cell acting as backup routers as it requires establishing several connections to several servers in the cell. Large cells can easily have several backup routers in the cell.

The order of the backup routers are decided at the local configuration phase. All the parties of this protocol must be configured accordingly to understand the order of the backup routers. It is not required that the backup server is actually an active server in the cell. The backup router may be a redundant server in the cell that does not accept normal client connections at all. It may be reserved purely for the backup purposes.

If also the first backup router is down as well and there is another backup router in the cell then it will start acting as the primary router as described above.

3.14.1 Switching to Backup Router

When the primary router of the cell becomes unresponsive, for example by sending EOF to the connection, all the parties of this protocol MUST replace the old connection to the primary router with first configured backup router. The backup router usually needs to do local modifications to its database in order to update all the information needed to maintain working routes. The backup router must understand that clients that were originated from the primary router are now originated from some of the existing server connections and must update them accordingly. It must also remove those clients that were owned by the primary router since those connections were lost when the primary router became unresponsive.

All the other parties of the protocol must also update their local database to understand that the route to the primary router will now go to the backup router.

Servers connected to the backup router MUST send SILC_PACKET_RESUME_ROUTER packet with type value 21, to indicate that the server will start using the backup router as primary router. The backup router MUST NOT allow this action if it detects that primary is still up and running. If backup router knows that primary is up and running it MUST send SILC_PACKET_FAILURE with type value 21 (4 bytes, MSB first order) back to the server. The server then MUST NOT use the backup as primary router, but must try to establish connection back to the primary router. If the action is allowed type value 21 is sent back to the server from the backup router. It is RECOMMENDED that implementations use the

SILC_COMMAND_PING command to detect whether primary router is responsive. If the backup router notices that the primary router is unresponsive it SHOULD NOT start sending data to server links before the server has sent the SILC_PACKET_RESUME_ROUTER with type value 21.

The servers connected to the backup router must then announce their clients, channels, channel users, channel user modes, channel modes, topics and other information to the backup router. This is to assure that none of the important notify packets were lost during the switch to the backup router. The backup router must check which of these announced entities it already has and distribute the new ones to the primary router.

The backup router too must announce its servers, clients, channels and other information to the new primary router. The primary router of the backup router too must announce its information to the backup router. Both must process only the ones they do not know about. If any of the announced modes do not match then they are enforced in normal manner as defined in section 4.2.1 Announcing Clients, Channels and Servers.

3.14.2 Resuming Primary Router

Usually the primary router is unresponsive only a short period of time and it is intended that the original router of the cell will resume its position as primary router when it comes back online. The backup router that is now acting as primary router of the cell must constantly try to connect to the original primary router of the cell. It is RECOMMENDED that it would try to reconnect in 30 second intervals to the primary router.

When the connection is established to the primary router the backup resuming protocol is executed. The protocol is advanced as follows:

1. Backup router sends SILC_PACKET_RESUME_ROUTER packet with type value 1 to the primary router that came back online. The packet will indicate the primary router has been replaced by the backup router. After sending the packet the backup router will announce all of its channels, channel users, modes etc. to the primary router.

   If the primary knows that it has not been replaced (for example the backup itself disconnected from the primary router and thinks that it is now primary in the cell) the primary router send SILC_PACKET_FAILURE with the type value 1 (4 bytes, MSB first order) back to the backup router. If backup receives this it MUST NOT continue with the backup resuming protocol.

2. Backup router sends SILC_PACKET_RESUME_ROUTER packet with type value 1 to its current primary router to indicate that it will resign as being primary router. Then, backup router sends the SILC_PACKET_RESUME_ROUTER packet with type value 1 to all connected servers to also indicate that it will resign as being primary router.

3. Backup router also send SILC_PACKET_RESUME_ROUTER packet with type value 1 to the router that is using the backup router currently as its primary router.

4. Any server and router that receives the SILC_PACKET_RESUME_ROUTER with type value 1 must reconnect immediately to the primary router of the cell that came back online. After they have created the connection they MUST NOT use that connection as active primary route but still route all packets to the backup router. After the connection is created they MUST send SILC_PACKET_RESUME_ROUTER with type value 2 back to the backup router. The session ID value found in the first packet MUST be set in this packet.

5. Backup router MUST wait for all packets with type value 2 before it continues with the protocol. It knows from the session ID values set in the packet when it has received all packets. The session value should be different in all packets it has sent earlier. After the packets are received the backup router sends the SILC_PACKET_RESUME_ROUTER packet with type value 3 to the primary router that came back online. This packet will indicate that the backup router is now ready to resign as being primary router. The session ID value in this packet MUST be the same as in the first packet sent to the primary router. During this time the backup router must still route all packets it is receiving from server connections.

6. The primary router receives the packet and send the packet SILC_PACKET_RESUME_ROUTER with type value 4 to all connected servers including the backup router. It also sends the packet with type value 4 to its primary router, and to the router that is using it as its primary router. The Session ID value in these packets SHOULD be zero (0).

7. Any server and router that receives the SILC_PACKET_RESUME_ROUTER packet with type value 4 must switch their primary route to the new primary router and remove the route for the backup router, since it is no longer the primary router of the cell. They must also update their local database to understand that the clients are not originated from the backup router but from the locally connected servers. After that they MUST announce their channels, channel users, modes etc. to the primary router. They MUST NOT use the

backup router connection after this and the connection is considered to be a passive connection. The implementation SHOULD be able to disable the connection without closing the actual link.

After this protocol is executed the backup router is now again a normal server in the cell that has the backup link to the primary router. The primary router feeds the router specific data again to the backup router. All server connections to the backup router are considered passive connections.

When the primary router of the cell comes back online and connects to its remote primary router, the remote primary router MUST send the SILC_PACKET_RESUME_ROUTER packet with type value 20 indicating that the connection is not allowed since the router has been replaced by an backup router in the cell. The session ID value in this packet SHOULD be zero (0). When the primary router receives this packet it MUST NOT use the connection as active connection but must understand that it cannot act as primary router in the cell, until the backup resuming protocol has been executed.

The following type values has been defined for SILC_PACKET_RESUME_ROUTER packet:

```
1    SILC_SERVER_BACKUP_START
2    SILC_SERVER_BACKUP_START_CONNECTED
3    SILC_SERVER_BACKUP_START_ENDING
4    SILC_SERVER_BACKUP_START_RESUMED
20   SILC_SERVER_BACKUP_START_REPLACED
21   SILC_SERVER_BACKUP_START_USE
```

If any other value is found in the type field the packet MUST be discarded. The SILC_PACKET_RESUME_ROUTER packet and its payload is defined in [SILC2].

4 SILC Procedures

This section describes various SILC procedures such as how the connections are created and registered, how channels are created and so on. The references [SILC2], [SILC3] and [SILC4] permeate this section's definitions.

4.1 Creating Client Connection

This section describes the procedure when a client connects to SILC server. When client connects to server the server MUST perform IP address lookup and reverse IP address lookup to assure that the origin

host really is who it claims to be. Client, a host, connecting to server SHOULD have both valid IP address and fully qualified domain name (FQDN).

After that the client and server performs SILC Key Exchange protocol which will provide the key material used later in the communication. The key exchange protocol MUST be completed successfully before the connection registration may continue. The SILC Key Exchange protocol is described in [SILC3].

Typical server implementation would keep a list of connections that it allows to connect to the server. The implementation would check, for example, the connecting client's IP address from the connection list before the SILC Key Exchange protocol has been started. The reason for this is that if the host is not allowed to connect to the server there is no reason to perform the key exchange protocol.

After successful key exchange protocol the client and server perform connection authentication protocol. The purpose of the protocol is to authenticate the client connecting to the server. Flexible implementation could also accept the client to connect to the server without explicit authentication. However, if authentication is desired for a specific client it may be based on passphrase or public key authentication. If authentication fails the connection MUST be terminated. The connection authentication protocol is described in [SILC3].

After successful key exchange and authentication protocol the client MUST register itself by sending SILC_PACKET_NEW_CLIENT packet to the server. This packet includes various information about the client that the server uses to register the client. Server registers the client and sends SILC_PACKET_NEW_ID to the client which includes the created Client ID that the client MUST start using after that. After that all SILC packets from the client MUST have the Client ID as the Source ID in the SILC Packet Header, described in [SILC2].

Client MUST also get the server's Server ID that is to be used as Destination ID in the SILC Packet Header when communicating with the server (for example when sending commands to the server). The ID may be resolved in two ways. Client can take the ID from an previously received packet from server that MUST include the ID, or to send SILC_COMMAND_INFO command and receive the Server ID as command reply.

Server MAY choose not to use the information received in the SILC_PACKET_NEW_CLIENT packet. For example, if public key or certificate were used in the authentication, server MAY use that information rather than what it received from client. This is a suitable way to get the true information about client if it is available.

The nickname of client is initially set to the username sent in the SILC_PACKET_NEW_CLIENT packet. User may set the nickname to something more desirable by sending SILC_COMMAND_NICK command. However, this is not required as part of registration process.

Server MUST also distribute the information about newly registered client to its router (or if the server is router, to all routers in the SILC network). More information about this in [SILC2].

Router server MUST also check whether some client in the local cell is watching for the nickname this new client has, and send the SILC_NOTIFY_TYPE_WATCH to the watcher.

4.2 Creating Server Connection

This section describes the procedure when server connects to its router (or when router connects to other router, the cases are equivalent). The procedure is very much alike to when a client connects to the server thus it is not repeated here.

One difference is that server MUST perform connection authentication protocol with proper authentication. A proper authentication is based on passphrase authentication or public key authentication based on digital signatures.

After server and router have successfully performed the key exchange and connection authentication protocol, the server MUST register itself to the router by sending SILC_PACKET_NEW_SERVER packet. This packet includes the server's Server ID that it has created by itself and other relevant information about the server. The router receiving the ID MUST verify that the IP address in the Server ID is same as the server's real IP address.

After router has received the SILC_PACKET_NEW_SERVER packet it distributes the information about newly registered server to all routers in the SILC network. More information about this is in [SILC2].

As the client needed to resolve the destination ID this MUST be done by the server that connected to the router, as well. The way to resolve it is to get the ID from previously received packet. The server MAY also use SILC_COMMAND_INFO command to resolve the ID. Server MUST also start using its own Server ID as Source ID in SILC Packet Header and the router's Server ID as Destination when communicating with the router.

4.2.1 Announcing Clients, Channels and Servers

After server or router has connected to the remote router, and it already has connected clients and channels it MUST announce them to the router. If the server is router server, also all the local servers in the cell MUST be announced.

All clients are announced by compiling a list of ID Payloads into the SILC_PACKET_NEW_ID packet. All channels are announced by compiling a list of Channel Payloads into the SILC_PACKET_NEW_CHANNEL packet. Channels' mode, founder public key, channel public keys, and other channel mode specific data is announced by sending the SILC_NOTIFY_TYPE_CMODE_CHANGE notify list.

The channel public keys that are announced are compiled in Argument List Payload where the argument type is 0x03, and each argument is Public Key Payload containing one public key or certificate.

Also, the channel users on the channels must be announced by compiling a list of Notify Payloads with the SILC_NOTIFY_TYPE_JOIN notify type into the SILC_PACKET_NOTIFY packet. The users' modes on the channel must also be announced by compiling list of Notify Payloads with the SILC_NOTIFY_TYPE_CUMODE_CHANGE notify type into the SILC_PACKET_NOTIFY packet.

The router MUST also announce the local servers by compiling list of ID Payloads into the SILC_PACKET_NEW_ID packet.

Also, clients' modes (user modes in SILC) MUST be announced. This is done by compiling a list of Notify Payloads with SILC_NOTIFY_UMODE_CHANGE notify type into the SILC_PACKET_NOTIFY packet. Also, channels' topics MUST be announced by compiling a list of Notify Payloads with the SILC_NOTIFY_TOPIC_SET notify type into the SILC_PACKET_NOTIFY packet. Also, channel's invite and ban lists MUST be announced by compiling list of Notify Payloads with the SILC_NOTIFY_TYPE_INVITE and SILC_NOTIFY_TYPE_BAN notify types, respectively, into the SILC_PACKET_NOTIFY packet.

The router which receives these lists MUST process them and broadcast the packets to its primary router. When processing the announced channels and channel users the router MUST check whether a channel exists already with the same name. If channel exists with the same name it MUST check whether the Channel ID is different. If the Channel ID is different the router MUST send the notify type SILC_NOTIFY_TYPE_CHANNEL_CHANGE to the server to force the channel ID change to the ID the router has. If the mode of the channel is different the router MUST send the notify type SILC_NOTIFY_TYPE_CMODE_CHANGE to the server to force the mode change to the mode that the router has.

The router MUST also generate new channel key and distribute it to the channel. The key MUST NOT be generated if the SILC_CMODE_PRIVKEY mode is set.

If the channel has a channel founder already on the router, the router MUST send the notify type SILC_NOTIFY_TYPE_CUMODE_CHANGE to the server to force the mode change for the channel founder on the server. The channel founder privileges MUST be removed on the server.

If the channel public keys are already set on the on router, the router MUST ignore the received channel public key list and send the notify type SILC_NOTIFY_TYPE_CUMODE_CHANGE to the server which includes the channel public key list that is on router. The server MUST change the list to the one it receives from router.

The router processing the channels MUST also compile a list of Notify Payloads with the SILC_NOTIFY_TYPE_JOIN notify type into the SILC_PACKET_NOTIFY and send the packet to the server. This way the server (or router) will receive the clients on the channel that the router has.

4.3 Joining to a Channel

This section describes the procedure when client joins to a channel. Client joins to channel by sending command SILC_COMMAND_JOIN to the server. If the receiver receiving join command is normal server the server MUST check its local list whether this channel already exists locally. This would indicate that some client connected to the server has already joined to the channel. If this is the case, the client is joined to the channel, new channel key is created and information about newly joined channel is sent to the router. The router is informed by sending SILC_NOTIFY_TYPE_JOIN notify type. The notify type MUST also be sent to the local clients on the channel. The new channel key is also sent to the router and to local clients on the channel.

If the channel does not exist in the local list the client's command MUST be sent to the router which will then perform the actual joining procedure. When server receives the reply to the command from the router it MUST be sent to the client which sent the command originally. Server will also receive the channel key from the server that it MUST send to the client which originally requested the join command. The server MUST also save the channel key.

If the receiver of the join command is router it MUST first check its local list whether anyone in the cell has already joined to the channel. If this is the case, the client is joined to the channel and reply is sent to the client. If the command was sent by server the command reply

is sent to the server which sent it. Then the router MUST also create new channel key and distribute it to all clients on the channel and all servers that have clients on the channel. Router MUST also send the SILC_NOTIFY_TYPE_JOIN notify type to local clients on the channel and to local servers that have clients on the channel.

If the channel does not exist on the router's local list it MUST check the global list whether the channel exists at all. If it does the client is joined to the channel as described previously. If the channel does not exist the channel is created and the client is joined to the channel. The channel key is also created and distributed as previously described. The client joining to the created channel is made automatically channel founder and both channel founder and channel operator privileges are set for the client.

If the router created the channel in the process, information about the new channel MUST be broadcast to all routers. This is done by broadcasting SILC_PACKET_NEW_CHANNEL packet to the router's primary route. When the router joins the client to the channel it MUST also send information about newly joined client to all routers in the SILC network. This is done by broadcasting the SILC_NOTIFY_TYPE_JOIN notify type to the router's primary route.

It is important to note that new channel key is created always when new client joins to channel, whether the channel has existed previously or not. This way the new client on the channel is not able to decrypt any of the old traffic on the channel. Client which receives the reply to the join command MUST start using the received Channel ID in the channel message communication thereafter. Client also receives the key for the channel in the command reply. Note that the channel key is never generated or distributed if the SILC_CMODE_PRIVKEY mode is set.

4.4 Channel Key Generation

Channel keys are created by router which creates the channel by taking enough randomness from cryptographically strong random number generator. The key is generated always when channel is created, when new client joins a channel and after the key has expired. Key could expire for example in an hour.

The key MUST also be re-generated whenever some client leaves a channel. In this case the key is created from scratch by taking enough randomness from the random number generator. After that the key is distributed to all clients on the channel. However, channel keys are cell specific thus the key is created only on the cell where the client, which left the channel, exists. While the server or router is creating the new channel key, no other client may join to the channel. Messages that are sent

while creating the new key are still processed with the old key. After server has sent the SILC_PACKET_CHANNEL_KEY packet client MUST start using the new key. If server creates the new key the server MUST also send the new key to its router. See [SILC2] for more information about how channel messages must be encrypted and decrypted when router is processing them.

If the key changes very often due to joining traffic on the channel it is RECOMMENDED that client implementation would cache some of the old channel keys for short period of time so that it is able to decrypt all channel messages it receives. It is possible that on a heavy traffic channel a message encrypted with channel key that was just changed is received by client after the new key was set into use. This is possible because not all clients may receive the new key at the same time, and may still be sending messages encrypted with the old key.

When client receives the SILC_PACKET_CHANNEL_KEY packet with the Channel Key Payload it MUST process the key data to create encryption and decryption key, and to create the MAC key that is used to compute the MACs of the channel messages. The processing is as follows:

```
channel_key  = raw key data
MAC key      = hash(raw key data)
```

The raw key data is the key data received in the Channel Key Payload. It is used for both encryption and decryption. The hash() is the hash function used with the HMAC of the channel. Note that the server also MUST save the channel key.

4.5 Private Message Sending and Reception

Private messages are sent point to point. Client explicitly destine a private message to specific client that is delivered to only to that client. No other client may receive the private message. The receiver of the private message is destined in the SILC Packet Header as in any other packet as well. The Source ID in the SILC Packet Header MUST be the ID of the sender of the message.

If the sender of a private message does not know the receiver's Client ID, it MUST resolve it from server. There are two ways to resolve the client ID from server; it is RECOMMENDED that client implementations send SILC_COMMAND_IDENTIFY command to receive the Client ID. Client MAY also send SILC_COMMAND_WHOIS command to receive the Client ID. If the sender has received earlier a private message from the receiver it should have cached the Client ID from the SILC Packet Header.

If server receives a private message packet which includes invalid

destination Client ID the server MUST send SILC_NOTIFY_TYPE_ERROR notify to the client with error status indicating that such Client ID does not exist.

See [SILC2] for description of private message encryption and decryption process.

4.6 Private Message Key Generation

Private message MAY be protected with a key generated by the client. One way to generate private message key is to use static or pre-shared keys in the client implementation. Client that wants to indicate other client on the network that a private message key should be set, the client MAY send SILC_PACKET_PRIVATE_MESSAGE_KEY packet to indicate this. The actual key material has to be transferred outside the SILC network, or it has to be pre-shared key. The client receiving this packet knows that the sender wishes to use private message key in private message communication. In case of static or pre-shared keys the IV used in the encryption SHOULD be chosen randomly. Sending the SILC_PACKET_PRIVATE_MESSAGE_KEY is not mandatory, and clients may naturally agree to use a key without sending the packet.

Another choice to use private message keys is to negotiate fresh key material by performing the Key Agreement. The SILC_PACKET_KEY_AGREEMENT packet MAY be used to negotiate the fresh key material. In this case the resulting key material is used to secure the private messages. Also, the IV used in encryption is used as defined in [SILC3], unless otherwise stated by the encryption mode used. By performing Key Agreement the clients can also negotiate the cipher and HMAC to be used in the private message encryption and to negotiate additional security parameters. The actual Key Agreement [SILC2] is performed by executing the SILC Key Exchange protocol [SILC3], peer to peer. Because of NAT devices in the network, it might be impossible to perform the Key Agreement. In this case using static or pre-shared key and sending the SILC_PACKET_PRIVATE_MESSAGE_KEY to indicate the use of a private message key is a working alternative.

If the key is pre-shared key or other key material not generated by Key Agreement, then the key material SHOULD be processed as defined in [SILC3]. In the processing, however, the HASH, as defined in [SILC3] MUST be ignored. After processing the key material it is employed as defined in [SILC3]. If the SILC_PACKET_PRIVATE_MESSAGE_KEY was sent, then it defines the cipher and HMAC to be used. The hash algorithm to be used in the key material processing is the one that HMAC algorithm is defined to use. If the SILC_PACKET_PRIVATE_MESSAGE_KEY was not sent at all, then the hash algorithm to be used SHOULD be SHA1. In this case also, implementations SHOULD use the SILC protocol's mandatory cipher

and HMAC in private message encryption.

4.7 Channel Message Sending and Reception

Channel messages are delivered to a group of users. The group forms a channel and all clients on the channel receives messages sent to the channel. The Source ID in the SILC Packet Header MUST be the ID of the sender of the message.

Channel messages are destined to a channel by specifying the Channel ID as Destination ID in the SILC Packet Header. The server MUST then distribute the message to all clients, except to the original sender, on the channel by sending the channel message destined explicitly to a client on the channel. However, the Destination ID MUST still remain as the Channel ID.

If server receives a channel message packet which includes invalid destination Channel ID the server MUST send SILC_NOTIFY_TYPE_ERROR notify to the sender with error status indicating that such Channel ID does not exist.

See the [SILC2] for description of channel message routing for router servers, and channel message encryption and decryption process.

4.8 Session Key Regeneration

Session keys MUST be regenerated periodically, say, once in an hour. The re-key process is started by sending SILC_PACKET_REKEY packet to other end, to indicate that re-key must be performed. The initiator of the connection SHOULD initiate the re-key.

If perfect forward secrecy (PFS) flag was selected in the SILC Key Exchange protocol [SILC3] the re-key MUST cause new key exchange with SKE protocol. In this case the protocol is secured with the old key and the protocol results to new key material. See [SILC3] for more information. After the SILC_PACKET_REKEY packet is sent the sender will perform the SKE protocol.

If PFS flag was set the resulted key material is processed as described in the section Processing the Key Material in [SILC3]. The difference with re-key in the processing is that the initial data for the hash function is just the resulted key material and not the HASH as it is not computed at all with re-key. Other than that, the key processing it equivalent to normal SKE negotiation.

If PFS flag was not set, which is the default case, then re-key is done

without executing SKE protocol. In this case, the new key is created by providing the current sending encryption key to the SKE protocol's key processing function. The process is described in the section Processing the Key Material in [SILC3]. The difference in the processing is that the initial data for the hash function is the current sending encryption key and not the SKE's KEY and HASH values. Other than that, the key processing is equivalent to normal SKE negotiation.

After both parties have regenerated the session key, both MUST send SILC_PACKET_REKEY_DONE packet to each other. These packets are still secured with the old key. After these packets, the subsequent packets MUST be protected with the new key. Note that, in case SKE was performed again the SILC_PACKET_SUCCESS is not sent. The SILC_PACKET_REKEY_DONE is sent in its stead.

4.9 Command Sending and Reception

Client usually sends the commands in the SILC network. In this case the client simply sends the command packet to server and the server processes it and replies with command reply packet. See the [SILC4] for detailed description of all commands.

However, if the server is not able to process the command, it is sent to the server's router. This is case for example with commands such as SILC_COMMAND_JOIN and SILC_COMMAND_WHOIS commands. However, there are other commands as well [SILC4]. For example, if client sends the WHOIS command requesting specific information about some client the server must send the WHOIS command to router so that all clients in SILC network are searched. The router, on the other hand, sends the WHOIS command further to receive the exact information about the requested client. The WHOIS command travels all the way to the server which owns the client and it replies with command reply packet. Finally, the server which sent the command receives the command reply and it must be able to determine which client sent the original command. The server then sends command reply to the client. Implementations should have some kind of cache to handle, for example, WHOIS information. Servers and routers along the route could all cache the information for faster referencing in the future.

The commands sent by server may be sent hop by hop until someone is able to process the command. However, it is preferred to destine the command as precisely as it is possible. In this case, other routers en route MUST route the command packet by checking the true sender and true destination of the packet. However, servers and routers MUST NOT route command reply packets to clients coming from other servers. Client MUST NOT accept command reply packet originated from anyone else but from its own server.

4.10 Closing Connection

When remote client connection is closed the server MUST send the notify type SILC_NOTIFY_TYPE_SIGNOFF to its primary router and to all channels the client was joined. The server MUST also save the client's information for a period of time for history purposes.

When remote server or router connection is closed the server or router MUST also remove all the clients that was behind the server or router from the SILC Network. The server or router MUST also send the notify type SILC_NOTIFY_TYPE_SERVER_SIGNOFF to its primary router and to all local clients that are joined on the same channels with the remote server's or router's clients.

Router server MUST also check whether some client in the local cell is watching for the nickname this client has, and send the SILC_NOTIFY_TYPE_WATCH to the watcher, unless the client which left the network has the SILC_UMODE_REJECT_WATCHING user mode set.

4.11 Detaching and Resuming a Session

SILC protocol provides a possibility for a client to detach itself from the network without actually signing off from the network. The client connection to the server is closed but the client remains as valid client in the network. The client may then later resume its session back from any server in the network.

When client wishes to detach from the network it MUST send the SILC_COMMAND_DETACH command to its server. The server then MUST set SILC_UMODE_DETACHED mode to the client and send SILC_NOTIFY_UMODE_CHANGE notify to its primary router, which then MUST broadcast it further to other routers in the network. This user mode indicates that the client is detached from the network. Implementations MUST NOT use the SILC_UMODE_DETACHED flag to determine whether a packet can be sent to the client. All packets MUST still be sent to the client even if client is detached from the network. Only the server that originally had the active client connection is able to make the decision after it notices that the network connection is not active. In this case the default case is to discard the packet.

The SILC_UMODE_DETACHED flag cannot be set by client itself directly with SILC_COMMAND_UMODE command, but only implicitly by sending the SILC_COMMAND_DETACH command. The flag also cannot be unset by the client, server or router with SILC_COMMAND_UMODE command, but only implicitly by sending and receiving the SILC_PACKET_RESUME_CLIENT packet.

When the client wishes to resume its session in the SILC Network it connects to a server in the network, which MAY also be a different from the original server, and performs normal procedures regarding creating a connection as described in section 4.1. After the SKE and the Connection Authentication protocols has been successfully completed the client MUST NOT send SILC_PACKET_NEW_CLIENT packet, but MUST send SILC_PACKET_RESUME_CLIENT packet. This packet is used to perform the resuming procedure. The packet MUST include the detached client's Client ID, which the client must know. It also includes Authentication Payload which includes signature computed with the client's private key. The signature is computed as defined in the section 3.9.1. Thus, the authentication method MUST be based in public key authentication.

When server receive the SILC_PACKET_RESUME_CLIENT packet it MUST do the following: Server checks that the Client ID is valid client and that it has the SILC_UMODE_DETACHED mode set. Then it verifies the Authentication Payload with the detached client's public key. If it does not have the public key it retrieves it by sending SILC_COMMAND_GETKEY command to the server that has the public key from the original client connection. The server MUST NOT use the public key received in the SKE protocol for this connection. If the signature is valid the server unsets the SILC_UMODE_DETACHED flag, and sends the SILC_PACKET_RESUME_CLIENT packet to its primary router. The routers MUST broadcast the packet and unset the SILC_UMODE_DETACHED flag when the packet is received. If the server is router server it also MUST send the SILC_PACKET_RESUME_CLIENT packet to the original server whom owned the detached client.

The servers and routers that receives the SILC_PACKET_RESUME_CLIENT packet MUST know whether the packet already has been received for the client. It is a protocol error to attempt to resume the client session from more than one server. The implementations could set internal flag that indicates that the client is resumed. If router receive SILC_PACKET_RESUME_CLIENT packet for client that is already resumed the client MUST be killed from the network. This would indicate that the client is attempting to resume the session more than once which is a protocol error. In this case the router sends SILC_NOTIFY_TYPE_KILLED to the client. All routers that detect the same situation MUST also send the notify for the client.

The servers and routers that receive the SILC_PACKET_RESUME_CLIENT must also understand that the client may not be found behind the same server that it originally came from. They must update their caches according to this. The server that now owns the client session MUST check whether the Client ID of the resumed client is based on the server's Server ID. If it is not it creates a new Client ID and send SILC_NOTIFY_TYPE_NICK_CHANGE to the network. It MUST

also send the channel keys of all channels that the client has joined to the client since it does not have them. Whether the Client ID was changed or not the server MUST send SILC_PACKET_NEW_ID packet to the client. Only after this is the client resumed back to the network and may start sending packets and messages.

It is also possible that the server did not know about the global channels before the client resumed. In this case it joins the client to the channels, generates new channel keys and distributes the keys to the channels as described in section 4.4.

It is an implementation issue for how long servers keep detached client sessions. It is RECOMMENDED that the detached sessions would be persistent as long as the server is running.

4.12 UDP/IP Connections

SILC protocol allows the use of UDP/IP instead of TCP/IP. There may be many reasons to use UDP, such as video and audio conferencing might be more efficient with UDP.

When UDP/IP is used, in the SILC Key Exchange protocol the IV Included flag MUST be set and the first 16-bits of the Cookie field in the Key Exchange Start Payload MUST include the port that the other end will use as the SILC session port. The port is in MSB first order. Both initiator and responder will set the port they are going to use and all packets after the SKE has been completed with the SILC_PACKET_SUCCESS packet MUST be sent to the specified port. Initiator will send them to the port responder specified and vice versa. When verifying the cookie for modifications the first two bytes are to be ignored in case IV Included flag has been set.

The default SILC port or port where the SILC server is listenning for incoming packets is used only during initial key exchange protocol. After SKE has been completed all packets are sent to the specified ports, including connection authentication packets and rekey packets even when PFS is used in rekey.

Changing the ports during SILC session is possible only by first detaching from the server (with client-server connections) and then performing the SILC Key Exchange protocol from the beginning and resuming the detached session.

Since the UDP is unreliable transport the SKE packets may not arrive to the recipient. Implementation should support retransmission of SKE packets by using exponential backoff algorithm. Also other SILC packets

such as messages may drop en route. With message packets only way to assure reliable delivery is to use message acking and retransmit the message by using for example exponential backoff algorithm. With SKE packets the initial timeout value should be no more than 1000 milliseconds. With message packets the initial timeout value should be around 5000 milliseconds.

5 Security Considerations

Security is central to the design of this protocol, and these security considerations permeate the specification. Common security considerations such as keeping private keys truly private and using adequate lengths for symmetric and asymmetric keys must be followed in order to maintain the security of this protocol.

Special attention must also be paid to the servers and routers that are running the SILC service. The SILC protocol's security depends greatly on the security and the integrity of the servers and administrators that are running the service. It is recommended that some form of registration is required by the server and router administrator prior to acceptance to the SILC Network. Even though the SILC protocol is secure in a network of mutual distrust between clients, servers, routers and administrators of the servers, the client should be able to trust the servers they are using if they wish to do so.

It however must be noted that if the client requires absolute security by not trusting any of the servers or routers in the SILC Network, it can be accomplished by negotiating private secret keys outside the SILC Network, either using SKE or some other key exchange protocol, or to use some other external means for distributing the keys. This applies for all messages, private messages and channel messages.

It is important to note that SILC, like any other security protocol, is not a foolproof system; the SILC servers and routers could very well be compromised. However, to provide an acceptable level of security and usability for end users, the protocol uses many times session keys or other keys generated by the servers to secure the messages. This is an intentional design feature to allow ease of use for end users. This way the network is still usable, and remains encrypted even if the external means of distributing the keys is not working. The implementation, however, may like to not follow this design feature, and may always negotiate the keys outside SILC network. This is an acceptable solution and many times recommended. The implementation still must be able to work with the server generated keys.

If this is unacceptable for the client or end user, the private keys negotiated outside the SILC Network should always be used. In the end

it is the implementor's choice whether to negotiate private keys by default or whether to use the keys generated by the servers.

It is also recommended that router operators in the SILC Network would form a joint forum to discuss the router and SILC Network management issues. Also, router operators along with the cell's server operators should have a forum to discuss the cell management issues.

6 References

[SILC2] Riikonen, P., "SILC Packet Protocol", Internet Draft, January 2007.

[SILC3] Riikonen, P., "SILC Key Exchange and Authentication Protocols", Internet Draft, January 2007.

[SILC4] Riikonen, P., "SILC Commands", Internet Draft, January 2007.

[IRC] Oikarinen, J., and Reed D., "Internet Relay Chat Protocol", RFC 1459, May 1993.

[IRC-ARCH] Kalt, C., "Internet Relay Chat: Architecture", RFC 2810, April 2000.

[IRC-CHAN] Kalt, C., "Internet Relay Chat: Channel Management", RFC 2811, April 2000.

[IRC-CLIENT] Kalt, C., "Internet Relay Chat: Client Protocol", RFC 2812, April 2000.

[IRC-SERVER] Kalt, C., "Internet Relay Chat: Server Protocol", RFC 2813, April 2000.

[SSH-TRANS] Ylonen, T., et al, "SSH Transport Layer Protocol", Internet Draft.

[PGP] Callas, J., et al, "OpenPGP Message Format", RFC 2440, November 1998.

[SPKI] Ellison C., et al, "SPKI Certificate Theory", RFC 2693, September 1999.

[PKIX-Part1] Housley, R., et al, "Internet X.509 Public Key Infrastructure, Certificate and CRL Profile", RFC 2459, January 1999.

[Schneier] Schneier, B., "Applied Cryptography Second Edition",

John Wiley & Sons, New York, NY, 1996.

[Menezes] Menezes, A., et al, "Handbook of Applied Cryptography", CRC Press 1997.

[OAKLEY] Orman, H., "The OAKLEY Key Determination Protocol", RFC 2412, November 1998.

[ISAKMP] Maughan D., et al, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[IKE] Harkins D., and Carrel D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[HMAC] Krawczyk, H., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[PKCS1] Kalinski, B., and Staddon, J., "PKCS #1 RSA Cryptography Specifications, Version 2.0", RFC 2437, October 1998.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

[RFC1321] Rivest R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC3174] Eastlake, F., et al., "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[PKCS7] Kalinski, B., "PKCS #7: Cryptographic Message Syntax, Version 1.5", RFC 2315, March 1998.

[RFC2253] Wahl, M., et al., "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

[RFC3454] Hoffman, P., et al., "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[SHA256] Eastlake 3rd, D., et al., "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

7 Author's Address

Pekka Riikonen
Helsinki
Finland

EMail: priikone@iki.fi

Appendix A

This appendix defines the stringprep [RFC3454] profile for string identifiers in SILC protocol. Compliant implementation MUST use this profile to prepare the identifier strings in the SILC protocol. The profile defines the following as required by [RFC3454].

- Intended applicability of the profile: the following identifiers in the SILC Protocol; nicknames, usernames, server names, hostnames, service names, algorithm names and other security property names [SILC3], and SILC Public Key name.

- The character repertoire that is the input and output to stringprep: Unicode 3.2 with the list of unassigned code points being the Table A.1, as defined in [RFC3454].

- The mapping tables used: the following tables are used, in order, as defined in [RFC3454].

  Table B.1
  Table B.2

  The mandatory case folding is done using the Table B.2 which includes the characters for the normalization form KC.

- The Unicode normalization used: the Unicode normalization form KC is used, as defined in [RFC3454].

- The prohibited characters as output: the following tables are used to prohibit characters, as defined in [RFC3454];

  Table C.1.1
  Table C.1.2
  Table C.2.1
  Table C.2.2
  Table C.3
  Table C.4
  Table C.5
  Table C.6

Table C.7
Table C.8
Table C.9

- Additional prohibited characters as output: in addition, the following tables are used to prohibit characters, as defined in this document;

  Appendix C
  Appendix D

- The bidirectional string testing used: bidirectional string testing is ignored in this profile.

This profile is to be maintained in the IANA registry for stringprep profiles. The name of this profile is "silc-identifier-prep" and this document defines the profile. This document defines the first version of this profile.

Appendix B

This appendix defines the stringprep [RFC3454] profile for channel name strings in SILC protocol. Compliant implementation MUST use this profile to prepare the channel name strings in the SILC protocol. The profile defines the following as required by [RFC3454].

- Intended applicability of the profile: channel names.

- The character repertoire that is the input and output to stringprep: Unicode 3.2 with the list of unassigned code points being the Table A.1, as defined in [RFC3454].

- The mapping tables used: the following tables are used, in order, as defined in [RFC3454].

  Table B.1
  Table B.2

  The mandatory case folding is done using the Table B.2 which includes the characters for the normalization form KC.

- The Unicode normalization used: the Unicode normalization form KC is used, as defined in [RFC3454].

- The prohibited characters as output: the following tables are used to prohibit characters, as defined in [RFC3454];

  Table C.1.1

Table C.1.2
Table C.2.1
Table C.2.2
Table C.3
Table C.4
Table C.5
Table C.6
Table C.7
Table C.8
Table C.9

- Additional prohibited characters as output: in addition, the following tables are used to prohibit characters, as defined in this document;

  Appendix D

- The bidirectional string testing used: bidirectional string testing is ignored in this profile.

This profile is to be maintained in the IANA registry for stringprep profiles. The name of this profile is "silc-identifier-ch-prep" and this document defines the profile. This document defines the first version of this profile.

Appendix C

This appendix defines additional prohibited characters in the identifier strings as defined in the stringprep profile in Appendix A.

Reserved US-ASCII characters
0021 002A 002C 003F 0040

Appendix D

This appendix defines additional prohibited characters in the identifier strings as defined in the stringprep profile in Appendix A and Appendix B. Note that the prohibited character tables listed in the Appendix A and Appendix B may include some of the same characters listed in this appendix as well.

Symbol characters and other symbol like characters
00A2-00A9 00AC 00AE 00AF 00B0 00B1 00B4 00B6 00B8 00D7 00F7
02C2-02C5 02D2-02FF 0374 0375 0384 0385 03F6 0482 060E 060F
06E9 06FD 06FE 09F2 09F3 09FA 0AF1 0B70 0BF3-0BFA 0E3F
0F01-0F03 0F13-0F17 0F1A-0F1F 0F34 0F36 0F38 0FBE 0FBF
0FC0-0FC5 0FC7-0FCF 17DB 1940 19E0-19FF 1FBD 1FBF-1FC1

1FCD-1FCF 1FDD-1FDF 1FED-1FEF 1FFD 1FFE 2044 2052 207A-207C
208A-208C 20A0-20B1 2100-214F 2150-218F 2190-21FF 2200-22FF
2300-23FF 2400-243F 2440-245F 2460-24FF 2500-257F 2580-259F
25A0-25FF 2600-26FF 2700-27BF 27C0-27EF 27F0-27FF 2800-28FF
2900-297F 2980-29FF 2A00-2AFF 2B00-2BFF 2E9A 2EF4-2EFF
2FF0-2FFF 303B-303D 3040 3095-3098 309F-30A0 30FF-3104
312D-3130 318F 31B8-31FF 321D-321F 3244-325F 327C-327E
32B1-32BF 32CC-32CF 32FF 3377-337A 33DE-33DF 33FF 4DB6-4DFF
9FA6-9FFF A48D-A48F A4A2-A4A3 A4B4 A4C1 A4C5 A4C7-ABFF
D7A4-D7FF FA2E-FAFF FFE0-FFEE FFFC 10000-1007F 10080-100FF
10100-1013F 1D000-1D0FF 1D100-1D1FF 1D300-1D35F 1D400-1D7FF

Other characters
E0100-E01EF

Full Copyright Statement