Workgroup: Network Working Group
Internet-Draft: draft-vanrein-sublime-01
Published: 11 September 2022
Intended Status: Standards Track
Expires: 15 March 2023
Authors: R. Van Rein
OpenFortress.nl

Subliminal Messaging in Codecs (SubliMe)

Abstract

The backbone for telephony consists of a digital network that is chiefly used for audio using the G.711 codec, which is also widely supported in VoIP telephony. This specification defines Subliminal Messaging as a general facility for opportunistic data exchange in the noise levels of audio codecs, with profiles for the G.711, G.722 and CLEARMODE codecs. The data exchange can be used to negotiate other codecs, UUID-identified services and call privacy/integrity.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 15 March 2023.

Copyright Notice

Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Telephony continues to be an area of communication that is mostly separate from the Internet. The addition of data transport, even if just opportunistic in nature, can resolve that. Telephony was traditionally an analog sound carrier and used analog modems to achieve this, but it has evolved into a digital backbone dedicated to the G.711 codec. Connectivity to this backbone is no longer real-time and impairs the old mechanisms for data transport. However, since codecs are digital, any part of its content is accurately replicated, allowing the opportunistic injection of data into codecs.

Codecs make use of sound properties to compress the audio signal, but the quality is often high enough to allow for bits that mostly reduce noise. These noise bits may alternatively be sacrificed to transfer digital data. Such a choice would be opportunistic, and care must be taken to validate any such data to distinguish it from noise from existing devices.

The G.711 codec passes floating-point numbers, and the volume of the mantisse bits are not always the same. A cut-off level may be defined, and bits under that level can be replaced with data. Such noise is audible, roughly to the level of a long-distance analog call, to yield bitrates around 28800 bits/second. For the higher quality G.722 codec, the lower 2 bits may be used for data to yield a constant 16000 bits/second raw data rate.

The design of Subliminal Messaging is based on HDLC frames, initially by stealing noise bearer bits from a codec and possibly later by stealing the complete byte flow of a codec. Checksums can detect compatibility beyond reasonable doubt. The address field in HDLC is used to dispatch data to various UUID-identified services. It is possible to transmit data in one direction only, such as in the early media of a remote ringing cadence or alongside a voice menu prompt.

HDLC starts under null encryption, but can run a key agreement service and then continue new service connections under an actual cryptographic mode. This adds authenticated encryption and service-closedown signatures.

Along a telephony path, codecs may be translated. A supportive translator may recognise Subliminal Messaging, take out the HDLC data and insert it into the translation. This generally destroys the ability to take over the entire codec byte stream, and there may be a need to use HDLC flow control on the translator. But when HDLC content is not otherwise modified, cryptographic assurances will

pass through, and provide end-to-end security for the data portions (but not the sound portions) of a call.

This approach for inclusion of data in a codec may be referred to as SubliMe, and it may be pronounced as "sublime". This includes future extensions that add bit-stealing and byte-stealing modes to more codecs.

2. HDLC framing

At the start of a connection, the channel is not assuming HDLC frames to be sent. For that to be considered, a first pattern to recognise is BREAK or 1111111, so seven of more consecutive 1 bits. Content is ignored until a FLAG marker 01111110 which starts the first HDLC frame. After the last FLAG, a BREAK can be sent without FLAG to return to the initial situation.

The switch from bit-stealing mode to byte-stealing mode or back is determined by HDLC commands, as specified below. The same is true for outer cryptography. Neither of these changes take immediate effect; instead, they are deferred until the next BREAK. After that has been sent, the channel makes the switch. A new BREAK should then be sent, which may be followed by a FLAG and the first HDLC frame in the new channel mode.

HDLC frames are inserted into codecs, initially using bit-stealing mode, but with the option to take over the complete codec byte stream in what will be called byte-stealing mode. The interpretation of HDLC frames is the same in both modes, but codecs may represent them completely differently.

Every HDLC frame is surrounded by FLAG markers. Two consecutive HDLC frames may share one FLAG marker 01111110 as their separator. It is also permitted to have no HDLC frame between to FLAG markers, so the FLAG can be used as filling when no HDLC frames are ready to be sent.

Before the first FLAG is accepted, the channel must send a BREAK marker 1111111. After this, the first FLAG marker may follow. After the last FLAG marker, another BREAK can be sent to return to the state where HDLC is essentially not transmitting. When new HDLC frames must be sent, another BREAK and FLAG can be sent. When BREAK occurs in the middle of an HDLC frame it is silently ignored.

2.1. HDLC in Bit-stealing Mode

Codecs that inject data bits for SubliMe into their media flow are said to work in bit-stealing mode. Encoding passes in a flow of bits, possibly with variable timing. The decoder produces the same flow of bits. The bits are not NRZI-encoded, because the codec does

not constitute the actual transmission channel. The codec may incorporate other encodings, though.

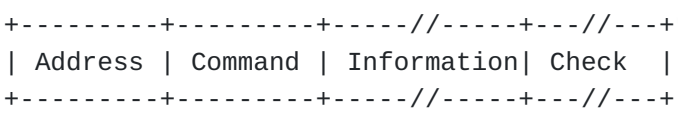
These bit flows are usually organised as synchronous HDLC, with bit-stuffing to turn data 11111 into 111110, thereby making 0111110 usable as a FLAG and 111111... as a BREAK marker.

2.2. HDLC in Byte-stealing mode

It is possible to completely switch the byte stream of a codec to a HDLC byte sequence. Any desire to pass audio must now be inserted via a HDLC service. This allows use of the full bandwidth for data, and only pass audio at a lower priority or in more compressed forms. The environment that negotiated the original codec is not made aware of this change; messaging with SubliMe is subliminal.

In byte-stealing mode, HDLC is organised as an asynchronous byte flow. The FLAG is the fixed byte 0x7e, and any occurrence of that byte in the data flow is escaped, as is the escape character and anything else deemed useful. Escaping inserts the byte value 0x7d and then passes the escaped byte XOR 0x40. Data bytes 0x7e are passed as 0x7d 0x3e and the data bytes 0x7d are passed as 0x7d 0x3d.

2.3. HDLC Frame Structure



HDLC frames send a one-byte Command to a one-byte Address. Depending on the Command field, there may be a use for a non-empty Information field. The frame ends with a Check. Since it is transmitted between FLAG sequences, the context provides framing and there is no need for an explicit Length field.

Addresses are used by SubliMe to determine which service shall process the HDLC frame. Addresses 0x00 through 0x7f are standardised in a register, and 0x80 through 0xfe can be called for with a service-specific UUID code. The othe end rejects unknown addresses. Address 0xff is generally used as "you-know-who" address for the remote end, but is more generally considered a broadcast address. Address 0x00 is used to communicate "meta" aspects of SubliMe.

Commands are standardised, and classify as I-frames that pass information with confirmation of reception, S-frames that pass supervisory information for confirmations and flow control, and U-frames that pass various kinds of unconfirmed information.

Certain commands carry details for windowing and flow control. I-frames include a modulo-8 frame number sent, while both I-frames and

S-frames send a modulo-8 frame number to indicate the next frame number that they expect to see from the other side. These are used to allow a small window of frames in transit, and regularly confirming them.

Information may be empty, and many HDLC frames require that. It is the carrier for user data, in a form that should be defined for the Address when registered, or for a UUID used to dynamically allocating an Address. There is a maximum size for the Information field, which means that fragementation may be needed when passing user data in HDLC frames.

Check (formally the Frame Check Sequence) validates whether the frame was transported without errors. The normal size of this field is 16 bits or 32 bits. SubliMe uses 16 bits for S-frames, and 32 bits for I-frames and most U-frames.

SubliMe makes a few modifications relative to the HDLC standard. Connections start with SABM and end with DISC, each causing a UA response on success; in SubliMe, the positive response to SABM is SABM and the response to DISC is DISC. SABM may carry salt or IV values in an Information field as defined by cryptography, and DISC may carry a long enough Check to enable cryptography to sign the entire connection I-frames as sent in its direction.

2.4. Windowing and Acknowledgement

The sender and receiver handle a window of up to 8 HDLC frames per Address, and indicate the progress on each end in various header fields. These window updates work as acknowledgements, and are sent in the header of an I-frame or S-frame. They are not included in U-frames.

After connecting to a SubliMe service Address in async/balanced mode, a receiver can actively send feedback. This is done with S-frames named RR (receiver ready), RNR (receiver not ready), REJ (reject) to request resends from the given window counter or SREJ (selective reject) to request that a given frame is sent once more.

Codec translators aware of SubliMe may take HDLC frames from one codec and inject it into another. This may lead to different bandwidth for HDLC, and they may need to inject RNR and RR signaling to Address 0x00, to pause and resume all flows of HDLC frames. This is done under null encryption.

Confirmation of arrival is only possible when a reverse channel of HDLC frames exists. This is not always the case, and it is still possible to do meaningful things as long as (1) it is possible to resend the data from the start because it is idempotent, and (2) there is no Poll bit anywhere. Some situations make it attractive to

send things blindly, for instance opportunistic attempts to share data.

2.5. Fragmentation of User Data

Aside from its tasks relating to data link management, HDLC carries user data. In doing so, it aims to respect the chunk sizes in which this occurs. The maximum size of the Information field may call for fragmentation of a chunk of user data over multiple HDLC frames.

2.6. Inner and Outer Stuff

There may be an outer codec and/or an inner codecs. Outer means that it is transported outside of HDLC (more accurately, that it contains the HDLC bits) and inner means that the codec is transported in HDLC frames.

There is a choice between outer and inner cryptography, again relative to the HDLC frames. Outer cryptography covers the entire byte stream and protects both the outer codec and the HDLC frames. Inner cryptography covers the contents of HDLC frames, including any inner codecs but excluding the outer codec, if any.

In bit-stealing mode, these differences can be meaningful. In byte-stealing mode, the difference is less pronounced, although HDLC headers are not fully protected by inner cryptography but they are with outer cryptography. But the more dramatic point is that inner cryptography does not protect an outer codec.

3. Communication Procedures

HDLC is expressive and, like most uses, SubliMe uses only a subset. Any commands not defined are ignored. In some cases where this benefits protocol understanding, this is detailed below.

3.1. XID :- Service Negotiation

XID frames send a tentative service configuration for the target Address. The peers locally combine these tentative frames to derive the actual service configuration for the Address. Two empty XID frames remove any existing service configuration from the Address.

When two tentative XID frames use a different UUID, then the lower one prevails and the higher one can try again on another Address. An empty XID in response to a tentative XID frame tells the recipient to stop trying this UUID.

XID frames can be sent before a connection. When they are sent during an open connection, they instantly update the interpretation, which would be error-prone with data in the window. It may however

be useful in the beginning of a connection, to benefit from its cryptographic mode.

XID frames may be sent to an Address that is or is not engaged in a connection. Connections may offer better cryptographic modes than null cryptography.

XID frames use a format local to SubliMe. They may be empty to reject a service negotiation attempt, or otherwise consist of the following byte sequence:

UUID: The service's UUID in 16 bytes. This identifies the service to run at the targeted Address, and defines the samantics of any data exchanged.

Service Version: One byte with the major version in the high nibble and the minor version in the low nibble, for the given UUID's service. These are protocol versions, not software versions, so they should be slow-changing. The major version signifies incompatibility and the minor version is backward compatible. Software may support multiple versions, as long as it avoids degredation to an insecure version.

General Flags: One byte with the following general flag bits, low to high:

*2 bits with the XID senders role (01=client, 10=server, 11=peer, 00=either)

*1 bit to request large Information frames (2048 bytes) instead of the default (256 bytes); not counting any bytes inserted by cryptographic modes

*4 bits reserved (send zero, do not interpret)

*1 bit fixated to 0 to signal a local XID format

MRU: Unsigned 16-bit integer in network byte order to indicate the maximum acceptable size for a user message to this service. When higher than the maximum Information field size, then reassembly will be done to form the user message. If not, bytes are sent to the service in a continuous flow.

Service Parameters: Every service may append to the XID bytes any further flags and parameters. These may depend on the Service Version, subject to compatibility concerns.

3.1.1. XID :- Bootstrapping SubliMe

SubliMe is an opportunistic protocol, so it must first be discovered. This begins with looking for HDLC in bit-stealing mode in the current codec, detecting a BREAK followed by a FLAG and valid HDLC frames interspersed with FLAG bytes. Each HDLC frame have good Check values. When this fails at any point, it needs to restart. Once established, a full restart is not necesssary.

As soon as HDLC framing are recognised, each party sends an XID frame to Address 0x00 to signal support for SubliMe.

```
Goal: SubliMe opportunistic service signaling
Name: sublime.0cpm.org
UUID: d14e63c6-3a3a-3b2b-8d9a-12140fa1b385
Pver: 1.0
Ustx: Full signatures on the most recent full second on the outer codec;
      UI commands may be transmitted to Address 0x00 without active conn
Parm: Service Parameters bytes follow
Info: The outer codec is split into 1 second worth of samples, rounded d
      This starts after the last BREAK.  Before the next second, the sig
      transmitted to allow the other side to verify outer codec integrit
```

Service Flags: One byte with Service Flags, from low to high:

- *1 bit to indicate support for outer cryptography; codec translators reset this bit
- *1 bit to indicate support for inner cryptography
- *1 bit to insist on outer cryptography; codec translators cannot be in the codec path
- *1 bit to request byte-stealing mode; codec translators may support this but may need active flow control with RR and RNR
- *4 bits reserved (send zero, do not interpret)

Cryptographic Modes: Listed in their one-byte representation. There must be at least one, even if it is just null cryptrography. The selected cryptographic mode for sending will be the first in this list that is mentioned in the Cryptographic Modes listed in the XID from the peer. The first of the Cryptographic Modes listed in the peer's XID and that occurs in this list will be used for reading.

3.2. SABM, DISC :- Service Connections

Service connections allow data communication for an Address and set the currently keyed cryptographic mode. Connections are opened with the SABM and closed with DISC. SABM is the first HDLC frame signed with the cryptographic mode that it installs.

Unlike standard HDLC, the confirming response to the SABM command is SABM. The negative response is DM. Standard SABM commands have no Information field, but the cryptographic mode may use it for parameters such as an initialisation vector. Both the cryptographic mode and its parameters are separate choices on each side.

Unlike standard HDLC, the confirming response to the DISC command is DISC. The negative response is DM. DISC commands have a longer Check field than standard, with the cryptographic mode's signature for the Information fields in preceding I commands.

Connection attempts with the other mode setting commands SM, SNRM, SARM, SNRME, SARME and SABME are ignored because they are not supported in this SubliMe version. The same applies to the RD command, which is unused because both ends send DISC.

3.3. SABM, DISC :- Switching between Stealing Modes

All codecs start in bit-stealing mode, but when the outer codec is not considered useful it may be beneficial to switch to byte-stealing mode. These modes are defined by the codec, and the byte-stealing mode can usually pass more, and never less, than the bit-stealing mode.

To switch from bit-stealing mode to byte-stealing mode, send SABM to Address 0x00; to switch back, send DISC to Address 0x00. Without confirmation, it cannot be certain that the other side will follow the change.

The actual change is not performed until a channel sends BREAK. The break condition involves 7 bits valued 1, and after the byte holding the last 1 bit, the channel switches to the other mode. The other mode also starts with at least 7 bits valued 1, followed by a FLAG and the next HDLC frame. Codecs can help by avoiding spurious FLAG signaling in the intermediate time, as they might offer to do when HDLC is off.

Codec translators may not be able to switch to byte-stealing mode, or perhaps they are unwilling to engage in bit-stealing mode when SubliMe is detected. They may inject or modify commands to signal this while the channel is under null cryptography.

BREAK and FLAG signaling remains visible in encrypted HDLC. As a result, the stealing mode can be switched under any cryptographic mode. Bit-stealing mode selects inner or outer cryptography, while byte-stealing mode can only use outer cryptography.

3.4. UI :- Unacknowledged Information

UI commands send-and-forget an Information field. The interpretation of the Information field is determined by the service configuration for the target Address. It is possible to send UI commands before or after a connection, in that case using null encryption.

One possible use of UI commands is to pass inner codec bytes. The cryptographic mode of the Address may protect the UI command.

3.5. I :- Information

I commands send user data in an Information field, possibly as part of a larger message to a service. When the Information field length is at least 256 then it is used to build up a user message up to the peer's MRU. Otherwise, it is the final part of a user message.

The (combined) message is interpreted by the service as specified for the service UUID from the last XID message sent to the target Address.

I commandds are sent with an incremented window index number (modulo 8). The peer should confirm the reception, either piggybacked on its I-frames or explicitly in an S-frames. The peer may also use S-frames REJ, SREJ and RR to request resends (two forms) or to acknowledge reception, and it may use RNR and RR to pause and resume the flow of I frames. These facilities imply that the I command can only be used inside a connection.

One possible use of I commands is to pass protocol data. The cryptographic mode of the Address protects the I command.

3.6. UP :- Polling for Progress Feedback

As long as traffic is bidirectional, there are many opportunities for feedback from the receiver to the sender, to acknowledge progress of the window pointer up to N(R). This allows the sending window pointer N(S) to move to this later position, thus freeing up sending buffers.

Positive acknowledgement is given with RR, rejection of anything beyond a window pointer is given with REJ. For individual missing frames, selective rejection with SREJ may be sent, though that would usually be done as soon as decoding of a hDLC frame fails.

3.7. TEST :- Detection of Codec Mangling

The standard behaviour of the TEST command in HDLC is to send back a TEST with the same Information field. This makes it a suitable candidate to see if the channel is mangled (and needs escapes on certain codes). When mangling occurs, the Check will not match the Information field and the response is not received. This means that a few experiments can be sent, and the responses indicate bytes that do not require escaping.

To test the codec, the TEST command is sent to Address 0x00. This should be done in byte-streaming mode to obtain meaningful results. Note that the TEST command may also be defined on other addresses, which may relay it in a service-specific manner.

3.8. Window Management and Acknowledgement Timing

The window size in each direction is 8 entries, since the N(S) and N(R) counters are 3 bits each, per direction. The percentage 12.5% * ((N(S) - N(R)) mod 8) shows how much of the window arrived but was not acknowledged yet. Some care for the window timing in terms of this percentage of the window is useful to keep data flowing:

* At any time, when a full information frame is available, it should be sent as an I-frame. HDLC framing then helps to simplify the processing of user data messages, and this is broken by combining those into one HDLC frame, except for continuous flows.

* Between 25% and 50% of the window received, the timing is suggesting to send back an I-frame for continuous flows. This results in a reply rate that is a querter up to half of the sending rate, suggesting that the interaction may slow down if not hastened by other factors.

* At 50% of the window received, the receiver may want to offload the sender, and send feedback with RR. This would only be used when no other interaction has taken care of it yet.

* With 75% of the window sent, the sender may get a little agitated, and request feedback with UP. This can help to receive active feedback before the window is full.

At any time that a frame cannot be recognised, the recipient may want to send SREJ. When the HDLC frame is encrypted, it will have to guess that the Address matches the last frame that was properly received, and the N(R) one higher than that frame. Since HDLC frames do not change order, this can still be sufficient information in situations where an Address change occurred, and redirect the resend to the address following it. (That is not standard in HDLC, but

SubliMe happens to cover multiple addresses in one endpoint, even with shared encryption.)

4. Service Definitions

Services should briefly state a purpose "Goal". They may use a DNS "Name" as a source for UUID3, but the formal identifier can be any "UUID". The protocol version "Pver" is required for clarity in service versioning. If Service Parameters are used in XID, they should be formalised in a "Parm" description. Inasfar as I and UI frames are permitted, their Information fields combine to user messages or to a continuous flow, whose grammar must be specified, preferrably as (a reference to) a formal grammar, in "Istx" and "Ustx" respectively. Inserted bytes for the cryptographic mode are not included in the latter.

These aspects may be named as "Goal", "Name", "UUID", "Parm", "Ustx" and "Istx". Any further semantics may be described in additional "Info" field.

```
Goal: Key agreement with TLS
Name: ietf-tls.sublime.0cpm.org
UUID: 1c469ae6-fb6b-3516-9689-d953e0513880
Pver: 0.0
Istx: One TLS record from the Handshake phase.
Info: When ClientHello messages meet, the one with the lower Random fiel
      will be the client.  When the handshake succeeds, export a key wit
      RFC 5705 using label "EXPORTER-SubliMe-cryptography"; it is used u
      another key agreement procedure succeeds.

Goal: Key agreement with ARPA2 KIP
Name: arpa2-kipdoc-keyexch.sublime.0cpm.org
UUID: fb3abc9a-bd42-3db0-9458-c1955ea6df5d
Pver: 0.0
Istx: One KIP Document intended to share a key after remote peer authent
Info: This allows key exchange in a KIP Document in general.  This can
      be used, among others, for key exchange.  The key is extracted
      after the receiving end authenticates to their KIP service.

Goal: Inner codec transmission
Name: <one specific codec>
UUID: <codec UUID>
Pver: 0.0
Ustx: The raw bytes for the codec
Istx: One RTCP frame
```

Goal: Sharing contact information
Name: ietf-vcard.sublime.0cpm.org
UUID: b038dd65-9d02-373d-92a2-d99bd6f625bb
Pver: 0.0
Istx: Textual, "vcard" grammar in Section 3.3. of RFC 6350.

Goal: Calendaring actions
Name: ietf-itip.sublime.0cpm.org
UUID: 793fad76-3725-3c23-af8b-eb0dbce103d6
Pver: 0.0
Istx: Textual, formatted as in Section 3 of RFC 5545.

Goal: Facsimile transmission
Name: itu-t38.sublime.0cpm.org
UUID: b0725c13-01d7-358d-b099-19fd72233c53
Pver: 0.0
Istx: One HDLC frame as defined in ITU T.38

Goal: Realtime text communication
Name: itu-t140.sublime.0cpm.org
UUID: 6d7bf8a4-6054-318c-b3ab-0ebc41d54e3d
Pver: 0.0
Istx: Any number of whole characters as defined in ITU T.140

Goal: Telephone-compliant Short Messaging
Name: org-smpp.sublime.0cpm.org
UUID: 818212af-821e-36ac-a61b-7369b6a44c15
Pver: 0.0
Istx: Continuous flow following the SMPP protocol

Goal: Telephone-compliant Multimedia Messaging
Name: etsi-mms-mm4.sublime.0cpm.org
UUID: 269a5933-c9cf-3807-abf1-3af4804c2769
Pver: 0.0
Istx: An email header and body, where every dot on the start of a
prefixed with another dot, and the email is followed by a dot on a
of its own (like the input to the SMTP DATA command).

Goal: Realtime interaction with XMPP
Name: ietf-xmpp.sublime.0cpm.org
UUID: 8affc68e-7819-3c18-864e-dcb888a26cbb
Pver: 0.0
Istx: One XMPP stanza as defined in RFC 6120

```
Goal: Remote database access
Name: ietf-ldap.sublime.0cpm.org
UUID: a22ff2c0-b7f8-3f0c-897b-455fb14e8211
Pver: 0.0
Istx: One LDAPMessage as defined in RFC4511
Info: Note that LDAP may be used bidirectionally

Goal: Document sharing by name
Name: community-zmodem.sublime.0cpm.org
UUID: c5375679-bc7a-3506-bcd3-f57fad341593
Pver: 0.0
Istx: Continuous flow adhering to Z-Modem specifications

Goal: Remote text terminal
Name: arpa2-tty.sublime.0cpm.org
UUID: 55f0fbe1-c230-3430-944e-d24b68b05c18
Pver: 0.0
Istx: Continuous flow of UTF-8 bytes with mulTTY extensions

Goal: Remote desktop access with VNC/RFB
Name: ietf-rfb.sublime.0cpm.org
UUID: 081a98f4-883f-3042-adbc-661c8e14ecf1
Pver: 0.0
Istx: One RFB protocol message as defined in Section 7 of RFC 6143

Goal: Remote device control over Modbus
Name: org-modbus.sublime.0cpm.org
UUID: ecb81231-643a-39af-97bf-80f9b0f664e4
Pver: 0.0
Istx: One frame of Modbus TCP

Goal: KIP Document exchange
Name: arpa2-kipdoc.sublime.0cpm.org
UUID: 7cc50f02-4d3b-36f2-8991-b964b0a31c49
Pver: 0.0
Istx: Streaming content forming a KIP Document.
```

5. Cryptographic Framework

The cryptographic framework adds authenticated encryption to HDLC. This is used for end-to-end security, involving assurance of the remote peer and integrity of its data.

5.1. Null Cryptography

Null cryptography is a mock cryptographic mode. it does not encrypt, and uses CRC-16/6sub8 to form the Check field. It can detect transport errors, but provides neither originator integrity nor privacy.

Null cryptography is always used outside of SABM/DISC connections, but also inside connections when no keyed cryptographic mode was available at the time that SABM is sent. Finally, codec translators also use it to send RNR and RR to Address 0x00 to control a peer's flow without knowing their key material.

Null cryptography uses the polynomial CRC-16/6sub8 which offers HD(3) up to 2048 Information bytes and HD(5) up to 10 Information bytes. Detected bit error rates may go up to 0.018% for 2048 Information bytes, 0.145% up to 256 and 18.75% when it is empty. The polynomial for this checksum is x^16 + x^8 + x^4 + x^3 + x^1 + x^0.

Null cryptography is identified with code 0x00. In SABM, its Information field is empty and DISC uses a Check with the same 32 bits as for other U-frames. UI, XID and UP have no initial bytes inserted in their Information fields.

5.2. Counter Management Framework

Cryptographic modes may rely on unique counter values to be secure. These values can be counted from 0 up for the exchange of SABM, I and DISC, which may then be resent only in the exact same form. Their orderly acknowledgement in HDLC allows an implicit counter discipline for such frames.

UI-, XID- and UP-frames also use encryption when sent inside a connection, but require different handling because the recipient may be confused about counter values when they do not arrive. For those frames, counting starts at the end, and lowered just enough to allow encryption and signing of a frame to be sent. The resulting counter value has a fixed size and will be inserted in the beginning of the Information fields of these frames. Care must be taken that the down-counter for UI and XID does not cross the up-counter for I. The recipient may not allow counter values to go up, and it may lower the boundary for that check once a lower counter value arrives with a proper, non-overlapping signature.

Most commands used in a connection may trigger a response. When allocating counter values, there are three areas to handle, in order:

Responses: The bytes needed for Check field authentication for each response separately. See response chaining below, also for the order of appearance of these encryption bytes.

Signature: The bytes needed for Check field authentication for the actual message.

Information: The bytes needed to encrypt the Information field of the message.

5.3. Streaming Galois Counter Mode (SGCM)

This section defines a variation on Galois Counter Mode (GCM) for streaming bytes and names it Streaming GCM, or SGCM. This is used by SubliMe, both in null cryptography and in the AES128-SGCM cryptographic mode.

Both GCM and SGCM use a block cipher to produce an XOR pad for encryption. It is essential to use every byte in the XOR pad at most once for any given key. This is established with the combined use of a 96-bit IV a 32-bit counter value that counts up from 0 and down from 4294967296 or 4294967295 and that must not overlap.

GCM derives a subkey from the encryption key by applying the block cipher to an all-zero block. This key is used to drive a multiplication in a Galois Field on as many bits as the block cipher. Every data block is multiplied in this way and, if another block follows, the output is XOR-ed with the next block before that too is multiplied. The last block is always shuffled because there is such a multiplication after it ends. SGCM uses a variation on GCM multiplication, also for subkey derivation.

SGCM differs by not multiplying complete blocks, but it uses XOR on one byte at a time, followed by a multiplication in a Galois Field with 8 bits in the block cipher; the multiplicand is one byte from the subkey, in a cyclic manner, plus 256 to avoid multiplication with zero. At this point, the signature may be forked for extension, or it may be finished here and now. In the latter case, an extra full cycles through all the bytes of the subkey is made, continuing like for normal bytes; this ensures that the last data byte is also shuffled by all the bytes matching the subkey.

As a result treating a byte at a time, there is no need for padding, and the disturbance varies unpredictably between the various positions. This means that no explicit insertion of the number of bits in the message is required.

GCM can start with additional data before it takes encrypted data into account. The additional data is multiplied as plaintext bytes and the remainder as encrypted bytes. The separation point is incorporated into the authenticated multiplication. Since the logic of HDLC and SubliMe also indicates where plaintext and encrypted bytes toggle, this strict separation and singular ordering is not useful; the explicit inclusion of messages sizes if forgeone in SGCM and considered a responsibility of the secured stream, which is met for the HDLC approach defined herein.

Both GCM and SGCM derive their final authentication tag by XOR-ing the repeated multiplication result with the first block from the

counted cipher to produce the final authentication tag. For authenticated encryption in HDLC, the most significant 32 bits of this value are shared in the Check field. The cryptographic assurance is around 2^(32-n) for a message of size 2^n, so around 2^-24 for 256 byte messages and around 2^-21 for 2048 bytes.

As long as the authententication tag does not reuse XOR pad bytes, it is possible to continue the multiplication, even in a manner that forks between alternatives. The multiplication (with blocks or bytes) serves to separate data elements, but the security derives from the unique XOR pad. It is required to shuffle bits before applying this XOR pad, so forks in SGCM start after a one-byte multiplication and before further perturbation.

5.4. AES128-SGCM Cryptography

AES128-SGCM uses AES128 as a block cipher in SGCM mode. The cryptographic mode is idenitified with code 0x01. In SABM, the Information field carries an initialisation vector of 96 bits and DISC carries a Check field with the full 128 bit signature. UI, XID and UP frames within an encrypted connection start with a 32-bit down-counter value.

UI, XID and UP commands start from the previously lowest counter and subtract the rounded-up number of blocks to cover the encryption of the Information and Check fields. The initial counter is set to 4294967296 or 4294967295.

SABM commands start a 32-bit upward counter at 0 and each consecutive signature I command adds to this counter the rounded-up number of blocks needed for Information and Check fields. The DISC command continues after the last I command.

All allocations of counter blocks are required to ensure that the down-counter for UI, XID and UP never drops below the up-counter for SABM, I and DISC.

5.5. Inner and Outer Cryptography

When codec translation is required on the communication path, then not all security properties can be resolved. Note however that switching between A-law and μ-law can be handled without this damage, by taking mangling into account.

Outer cryptography involves encryption and signing of the entire codec. The signature is passed in HDLC frames, both in bit-stealing mode and byte-stealing mode. Signatures are 32 bits long, so they do not have cryptographic assurance on their own, but chaining of I-frames works like a thread that increases assurance up to 128 bit in

the last frame. The final DISC includes a full signature as defined by the cryptographic mode.

Inner cryptography involves encryption and signing for HDLC frames. This is always possible, even with codec translation in place. It does not protect the codec into which bit-stealing mode injects, however, and that should be signaled to the user. Using UI frames, it is possible to send inner sound as part of HDLC, so it is secure.

The choice between inner and outer cryptography is made while bootstrapping SubliMe with XID to Address 0x00. This is independently done for both directions. The flag to insist on outer cryptography always causes outer crypto, even if this breaks a codec translator, because anything else would be supportive of a downgrade attack. When both sides agree to byte-stealing mode, that switch is made first. Codec translators should not pass the XID to Address 0x00 if it insists on outer crypto, but either it does not ask for byte-stealing mode or the codec translator cannot offer that. Codec translators should can safely pass the inner cryptography flag, provided that they take the HDLC frames out from the incoming codec and inject it into the outgoing codec.

5.6. Uplink and Downlink Cryptography

In his theory of special relativity, Einstein explains that there can be no notion of two things happening at the same time in different locations. The SABM connections cause precisely this kind of problem, making it generally impossible to order the frames sent from either end. The two directions of frames therefore send independent frame sequences.

Accommodating this, the crypto used is independently set for each of the directions. Bootstrapping with XID to Address 0x00 has established what cryptographic modes may be used, and connections to an key agreement Address guides the choice of key exchange, but keys may be setup separately in each direction. It is possible to use one key exchange phase to derive two keys separately, but it is also possible to start another key exchange, for instance to extend single-sided authentication into mutual authentication.

Having independent crypto in each direction helps to switch it on or off more elegantly. This coincides with the (theoretic) option to use different codecs in each direction. It also matches the idea that codecs independently switch between bit-stealing mode and byte-stealing mode after agreeing on it with SABM and DISC commands sent to Address 0x00.

5.7. Framing, Escaping and Bit-Stuffing under Encryption

There is no length field in HDLC because it surrounds frames with a FLAG and escaping or bit-stuffing similar patterns inside it. The frame boundaries remain in plaintext, and the same is true for bit-stuffing in bit-stealing mode, and for escaping in byte-stealing mode.

This means that before encryption and after decryption, the HDLC frame has no internal escaping for SubliMe to take care of.soso These are transport modifications only, and indeed dependent on whether the transport is based on bit-stealing mode or byte-stealing mode.

HDLC also defines a BREAK, which is evaded by the same practice of bit-stuffing or escaping. This also sites outside of the encryption, and it is used to switch the codec to the desired mode, if it was recently changed by commands exchanged inside the encryption layer.

5.8. Encryption Framework

The cryptographic mode organises most of the nuts and bolts of encryption. This specification only clarifies the areas that are encrypted, and how traffic in transit may connect to encryption.

Encryptionn can be implemented as inner or outer cryptography. While XID bootstrapping to Address 0x00, the options are set to choose the variant that will be used. They are not both employed, but the choice is made separately for the uplink and downlink.

5.8.1. Inner Encryption Framework

Inner encryption applies to the HDLC frame format only. It does not protect the outer codec. It produces the same results in bit-stealing mode and byte-stealing mode.

The Address and Command fields are not encrypted. When the Address is dynamically allocated, its service negotation may however be concealed by running XID in a connection with a cryptographic mode.

5.8.2. Outer Encryption Framework

In byte-stealing mode, outer encryption coincides with inner encryption. In bit-stealing mode, outer encryption involves encryption of the codec as well as the bits stolen to pass HDLC frames.

Outer encryption makes codec translation impossible, so it may be disabled by an intermediate. To protect against that, the XID to

Address 0x00 may insist on outer encryption. If this creates an impasse, then byte-stealing mode is a way out.

Outer encryption is a steam cipher applied to the codec bytes between the transmission channel and the detection of FLAG and BREAK signals, as well as HDLC frame bits. Outer encryption will be updated to the desired setting after a BREAK is sent out via the codec in the old format. After the BREAK, the switch is made and scanning for a FLAG continues in the new format. It is recommended to send an extra BREAK in the new format to facilitate synchronisation.

5.9. Signature Framework

The cryptographic mode organises most of the nuts and bolts of signing. This specification only clarifies the areas that are signed, and how traffic in transit may connect to signatures.

All HDLC frames need a Check, and this constitutes an inner signature, which is always present. When outer crypto is used, there will additionally be UI frames sent to Adress 0x00 with the signature for the outer codec.

Signatures work on the Address, Command and Information fields, not the Check field. Before signing, the Information field is encrypted and the signed fields are escaped by replacing bytes 0x7d..0x7f with an escape 0x7d and the original byte XOR 0x40.

5.9.1. Signature Chaining

HDLC frames can be chained, with an unescaped 0x7e FLAG between their escaped content-for-signing. The Check fields are not incorporated into signatures and there shall be no initial or trailing FLAG byte, just separating FLAG bytes. Signatures chaining can be efficient if it uses a forking point in the cryptographic mode.

Command	Data chaining	Response chaining
XID	-	-
TEST	-	-
SABM	-	DM
DISC	SABM, I*	DM
I	SABM, I*	RR, RNR, REJ, SREJ
UP	-	RR, REJ, SREJ
UI	-	-

Certain HDLC commands may trigger a response, which should be part of the security framework, both for reasons of privacy (encryption

of the Address and connection progress) and for authentication (actually being entitled to respond).

All these responses at the HDLC level are chained to the signature for the command with an intermediate FLAG byte. There may also be a need to allocate counter values for this purpose.

Besides response chaining, there is also a signature chain per connection direction, again based on FLAG separator bytes. This chain starts with the SABM command, includes all sent I frames and the final DISC command. Unconfirmed I frames may lead to confusion; the UP command and resends should be done before signing for the whole connection with DISC.

Connection chaining does not incorporate the responses in the chain; response chaining works like forks from the connection chain. Resends also are concealed from the connection chain, which is possible as long as the encrypted HDLC frame is literally sent in the same manner. The result is a connection chain that indicates the setup, data exchange and teardown of an entire connection, as seen from one side.

Connections use the same chaining mechansim from the cryptographic mode as used for responses. There will be no need to allocate counter values for connection chaining.

5.9.2. Inner Signature Framework

Inner signatures add the Check value to HDLC frames. This is always done. During outer cryptography, the unencrypted HDLC frames are signed before they are encrypted. During inner cryptography, the HDLC frame is encrypted and may be signed in encrypted or plaintext form, as defined by the cryptographic mode. For null cryptography, this is makes no difference because the encrypted content equals the plaintext.

Signing starts together with encryption, unless there is a reason for chaining, namely connection chaining or response chaining. Note that response chains may branch off from a connection chain, as described above.

5.9.3. Outer Signature Framework

Outer signatures are sent if and only if outer cryptography is used, as a result of XID bootstrapping via Address 0x00. It coincides with outer encryption. Null cryptography does not count as "cryptography is used", and no outer cryptography is applied. This protects from permissible phenomenons during the setup process, such as modifications by codec translators and synchronisation problems at the start of communication.

Outer signatures start with the first byte that is also subjected to outer encryption, so right after detection of a BREAK marker.

Signatures are sent over 1 second worth of outer codec, where the number of samples per second from the official documentation is used, rounded down if necessary. For G.711 and G.722 this would mean that every sequence of 8000 samples is signed.

The signature is made over those samples in isolation, so the line can recover from temporary bursts, showing only a glitch in the line assurances. Such bursts may be signaled with flashing lights and/or audible tones, and it is not helpful if such distractions continue as a result of resilient line problems.

Some codecs are subjected to mangling, which bit-stealing mode corrects for. If this is the case, then the mangled bits are set to 0 for the purpose of computing the signature.

The full signature from the cryptographic mode is sent in the Information field of a UI frame targeted at Address 0x00. Note that HDLC will add its own checksum as well.

6. Security Considerations

SubliMe is an opportunistic protocol and must be actively negotiated over an existing protocol. As a result, it is sensitive to denial-of-service attacks. This may interfere, among others, with the privacy and integrity of call data. It is helpful to communicate these desirable properties explicitly to the user.

Service connections may be open before key agreement succeeds. Such services would not be protected. If software is not explicit about such distinctions, then wrongful security assumptions may be made. Where security is a requisite, the advised approach is to suppress XID negotiation until a suitable security mode for the respective service has been achieved.

Sound snippets are signed independently, without connecting information. This helps the sound to recover after problems, but it also subjects the audio stream to replay of sound, and order changes.

When I frame transmission fails for some reason, the security mode may not be able to close the DISC message with a desirable signature.

7. IANA Considerations

There is no registration task for IANA. Services are identified with UUIDs and a documentation format is suggested, but their publication

is the responsibility of service authors who aim for general acceptance.

Appendix A. Data in Codecs

Codecs usually pack analog or complex data in a lossy manner in accurately transported byte sequences. By playing with the noise level, the bit-stealing mode can be added. And when requested, the entire byte flow of a codec might be claimed for HDLC frame transport. The manner in which this is done is specific to a codec.

This appendix is normative.

A.1. Data in the CLEARMODE Codec

Where available, the RFC4040 RTP type for audio/clearmode may be used to negotiate a completely undisturbed 64 kb/s channel. The link to PSTN is described in ITU Q.1912.5 so telecom providers may indeed facilitate it.

```
CLEARMODE byte
--------------
 .xxxxxxxx        Legend:
                  x = Clearmode bit dedicated to data
                  . = Noise level separator
```

Clearmode channels are not subjected to mangling, and so the provisions that skip the lowest bit will not be used. In fact, since there are no defined sound semantics, the bandwidth can be completely used for HDLC transport.

Although the channel starts in bit-stealing mode and considers that a different setting from byte-stealing mode, there is no practical difference. The full 8 bits per byte are available, without mangling, so its implementation of bit-stealing does not work with bit stuffing but with the same escaping mechanism (for FLAG, BREAK and escape bytes within HDLC frames) as for byte-stealing mode.

Clearmode offers no outer codec, but is open to inner codecs.

Bytes are XOR-ed with 0x55 for transport. This helps to set lots of transitions in zero content, and since this is an interface to a digital telephony backbone that is useful.

A.2. Data in the G.722 Codec

In byte-stealing mode, the entire G.722 codec would be considered a transport layer for bytes, just like CLEARMODE. It would escape bytes for FLAG, BREAK and escape inside HDLC frames.

In bit-stealing mode, the least-valued 2 bits of each sample are used for data. This is explicitly permitted by the G.722 specification, without indications of a particular purpose. These bits represent finer details that may be replaced by arbitrary data. They are used as HDLC bits, where the higher-valued bit comes before the lower-valued bit.

```
G.722 byte
----------
 zzzzzz.xx    Legend:
              z = G.722 bit dedicated to audio
              x = G.722 bit dedicated to data
              . = Noise level separator
```

It is not sufficient to combine 4 consecutive blocks of 2 bits to form a byte; firstly because synchronisation of the byte start would be difficult, and secondly because bit stuffing would end up being awkward. Instead of taking this approach, bit-stealing mode for G. 722 will consider the least-valued 2 bits in every sample just like for the G.711 form with 2 data bits. This may also improve code sharing.

Mangling does not occur in G.722 because it runs over a CLEARMODE channel, as ITU Q.1912.5 suggests. This means that no provisioning is needed for the lower words. Indeed, mangling is a G.711 phenomenon.

Bytes are XOR-ed with 0x55 for transport. This helps to set lots of transitions in zero content, and since this is an interface to a digital telephony backbone that is useful.

A.3. Data in the G.711 Codecs

Even though ISDN is going or gone for subscriber lines, it still forms a vital part of the telephony backbone and, because its codecs are rigidly enforced, the more flexible VoIP systems have all adapted to include A-law and μ-law, the two forms defined in G. 711.

The G.711 codec used in SubliMe is A-law only. There are predefined translations between A-law and μ-law and back, and no matter how often this is used the mangling of codec bytes that it causes is known and constant. The reason to choose A-law only is that it retains detail in the lower bits during such translations, which is where we can transmit most of our data. Mangling of samples occurs in the higher values, but this is less damaging to the transmission of data in the codec.

A-law output	µ-law output	A-law output	Mangled
25, 26	32	25	26
27, 28	33	27	28
29, 30	34	29	30
31, 32	35	31	32
45, 46	48	46	45
47, 48	49	48	47
63, 64	64	64	63
79, 80	79	79	80

TODO:CAPTION: Some A-law codec bytes are mangled by the transition to µ-law and back. Sign was removed in the byte values. The bytes are the wire values, no transport XOR mask 0x55 will be applied.

In byte-stealing mode, the codec carries HDLC frame bytes directly as codec data, but it should be mindful that some of the A-law bytes may translate to one μ-law byte and back to one A-law byte; one of the original A-law values is then mangled. These mangles values should be sent with escaping if it is unknown whether this may occur on the communications channel. This may be tested for explicitly, by sending a TEST command to Address 0x00 and checking the response.

A-law byte ^ 0x55	audio sample
s111mmmm	s1mmmm00.00000
s110mmmm	s01mmmm0.00000
s101mmmm	s001mmmm.00000
s100mmmx	s0001mmm.x0000
s011mmxx	s00001mm.xx000
s010mxxx	s000001m.xxx00
s001xxxx	s0000001.xxxx0
s000xxxx	s0000000.xxxx0

Legend:
s = Sign bit
e = Exponent bit
m = Mantisse bit
above the noise level
x = Mantisse bit
under the noise level
. = Noise level separator

TODO:CAPTION: A-law codec bytes, after XOR with transport mask 01010101, map to sample values which may be cut off at a consistent noise level to make room for data bits. Represenation of the sign can vary.

In bit-stealing mode, the exponent determines how far the mantisse is shifted. The insertion of a fixed point in the actual samples shows where SubliMe makes a cut-off between audio content above and under the noise level. The bits under the noise level are stolen to carry the HDLC bit flow, in the direction from most to least significant bit.

Mangling	A-law change	A-law byte ^ 0x55	
26 -> 25	0x4c -> 0x4d	s100.110q	
28 -> 27	0x4e -> 0x4f	s100.111q	
30 -> 29	0x48 -> 0x49	s100.100q	Legend:
32 -> 31	0x4a -> 0a4b	s100.101q	s = Sign Bit
45 -> 46	0x79 -> 0x78	s1111.00q	. = Noise level separator
47 -> 48	0x7b -> 0x7a	s1111.01q	q = Possibly mangled bit
63 -> 64	0x6b -> 0x6a	s11010.1q	
80 -> 79	0x1a -> 0x1b	s001101q.	

TODO:CAPTION: A-law bytes that may be mangled on the wire
cause one bit in the A-law codec byte without the transport
XOR mask 01010101 to have an uncertain least significant bit.

Just before stealing the least significant bit for data, it may become clear that it will be part of a mangled pair of codec values. In this case, this bit cannot carry data. It should instead be set so the total byte becomes a mangled value. Upon reception, the occurrence of this same value is a sign that no mangling has taken place.

As an efficiency measure, when no HDLC frames are being transmitted, the bit-stealing mode may switch off by sending a BREAK after the last FLAG, and then set the lowest data bit to 0 where data could be. In case of a mangled pair of codec values, the one-but-lowest data bit would be set to 0 instead. Since no more than 4 data bits are carried in any codec byte, this lower 0 bit enables an efficient test that the byte can be skipped. The "off" mode therefore becomes a chase for a lowest bit set to 1 and then continues to match for BREAK and FLAG marks. This lower bit is not detectable to the ear, but the more signifcant bits can be heard as noise, and when they are not altered the sound quality improves when no bits are stolen for the transmission of HDLC frames.

The A-law codec in G.711 already ensures that all bytes are XOR-ed with 0x55 for transport. This helps to set lots of transitions in zero content, and since this is an interface to a digital telephony backbone that is useful.

Appendix B. Acknowledgements

This work was supported by NLnet.nl as one element of the Subliminal Messaging project, along with KIP-secured SIP connection management, and Wireshark VPN connectivity configuration over SIP and/or telephone links.

I also owe gratitude to my father, Harry van Rein, who brought me as a kid an endless supply of telephony waste from his repair job to tinker with.

Author's Address

Rick van Rein
OpenFortress.nl
Haarlebrink 5
Enschede

Email: rick@openfortress.nl