SIP TLS

SIP Encryption Primer

FreeSWITCH supports both encrypted signaling known as SIPS which can be SSL or TLS with signed certificates, as well as encrypted audio/media known as SRTP. Typical convention is to have the unencrypted SIP control channel on UDP port 5060 (although the standards also allow for using TCP port 5060 as well), and an SSL encrypted or TLS encrypted SIP control channel known as SIPS on TCP port 5061. The illustrations below depict SIP as being on port 5060, and SIPS as being on port 5061 and port (X). FreeSWITCH allows you to configure this port within the SIP profile.

The RTP data utilizes UDP, but the port that RTP uses is dynamic in that it's negotiated within the SIP control channel. FreeSWITCH can be configured to use a specific port range for the RTP streams. In section 1 of this page we will go over the anatomy of a SIP connection and possible configurations so that you can choose which configuration would best suit your scenario and configure your FreeSWITCH system accordingly.

Anatomy of a SIP Connection

First, let's review the anatomy of a SIP connection in order to dispel any confusion that may arise out of the assumption that encryption implies that the audio (or RTP) streams are encrypted.

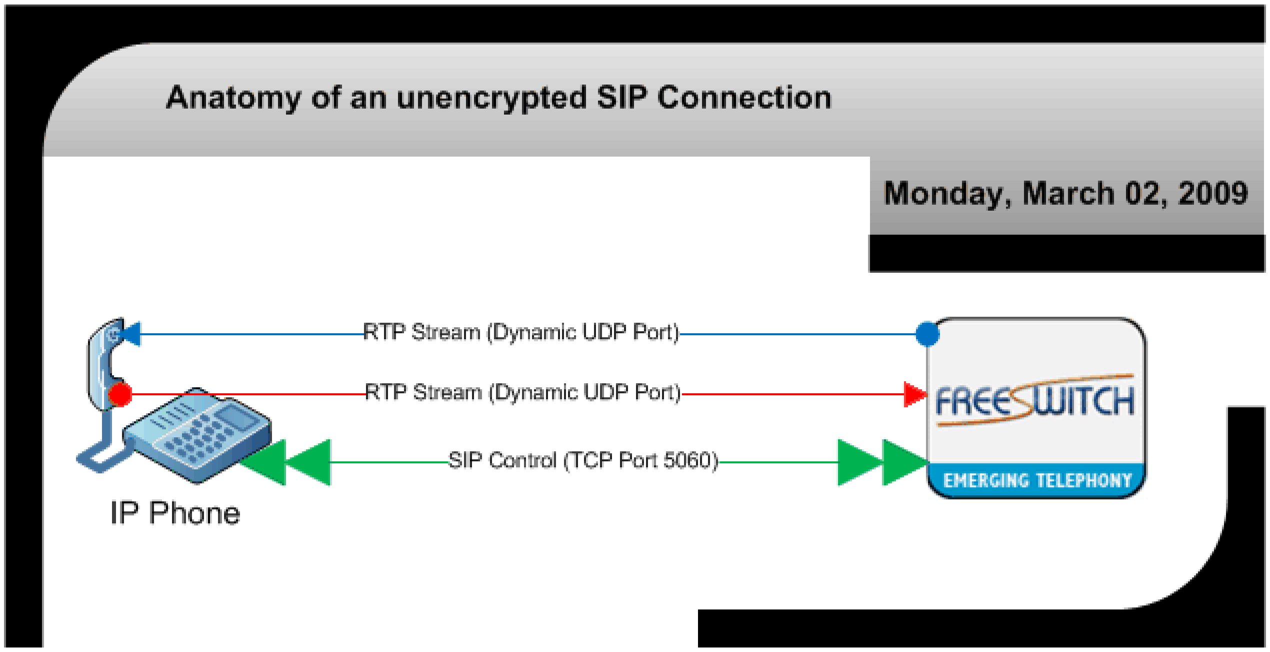
Anatomy of an unencrypted SIP Connection
Monday, March 02, 2009
RTP Stream (Dynamic UDP Port)
RTP Stream (Dynamic UDP Port)
SIP Control (TCP Port 5060)
IP Phone
FREESWITCH
EMERGING TELEPHONY

Figure 1:

In figure 1 you'll note that there are indeed three separate connections, one TCP for control, and the other two are UDP for passing the audio streams to and from the switch. The control channel facilitates a number of capabilities, but with regard to encryption the ones that we're primarily concerned about is DTMF digits dialed during the call, such as a bank PIN, or some other sort of access code which is personal and, most importantly, private information. With the scenario depicted in figure 1, anyone on the network may sniff and decode this valuable data.

Anatomy of a SIPS Connection

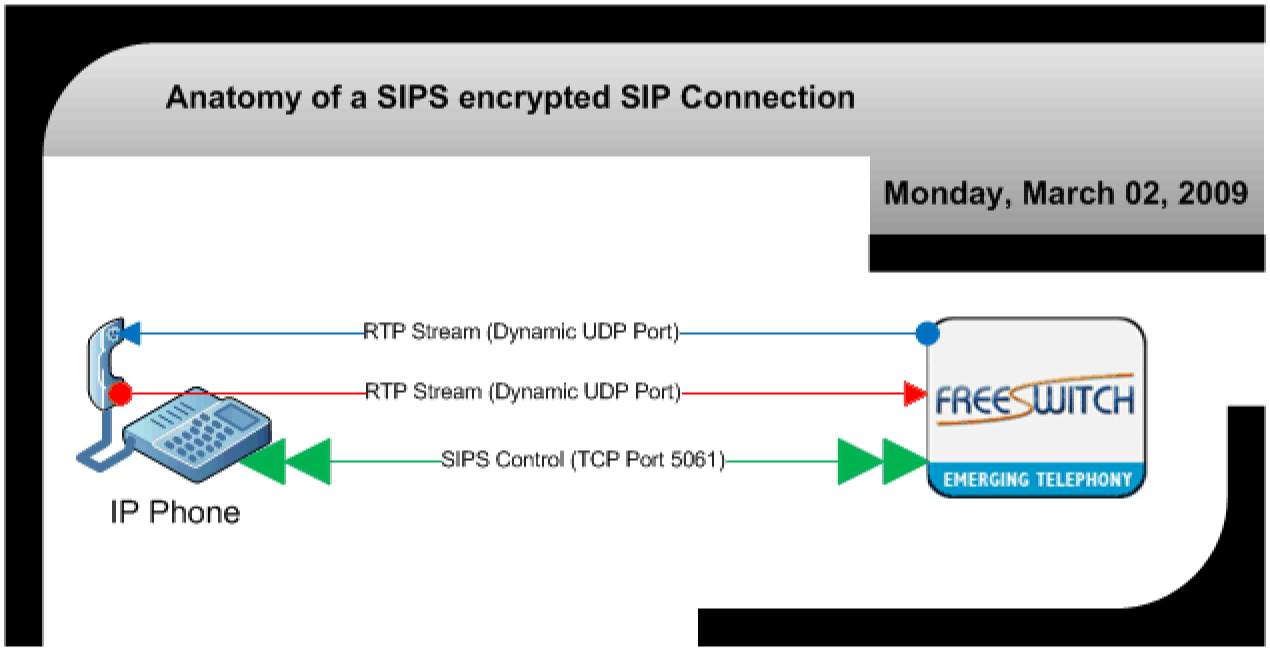
Anatomy of a SIPS encrypted SIP Connection
Monday, March 02, 2009
RTP Stream (Dynamic UDP Port)
RTP Stream (Dynamic UDP Port)
SIPS Control (TCP Port 5061)
IP Phone
FREESWITCH
EMERGING TELEPHONY

Figure 2:

The answer to this problem is depicted in figure 2 and to encrypt the SIP channel with an SSL or TLS wrapper such that anyone sniffing packets on the wire cannot decrypt the data in the control channel. You'll note that in figure 2 we depict this as on port 5061, but with FreeSWITCH you can in fact configure it to be on any particular port you desire. This scenario here still leaves the packets for the audio data in the clear. Someone sniffing the packets will be able to reconstruct your audible conversation and "listen in" without much difficulty.

***Note:** *IP phones may advertise they support encryption, but you may want to check the specifications to validate they support both SSL and TLS signed certificates.*

Anatomy of a SIPS and SRTP Connection

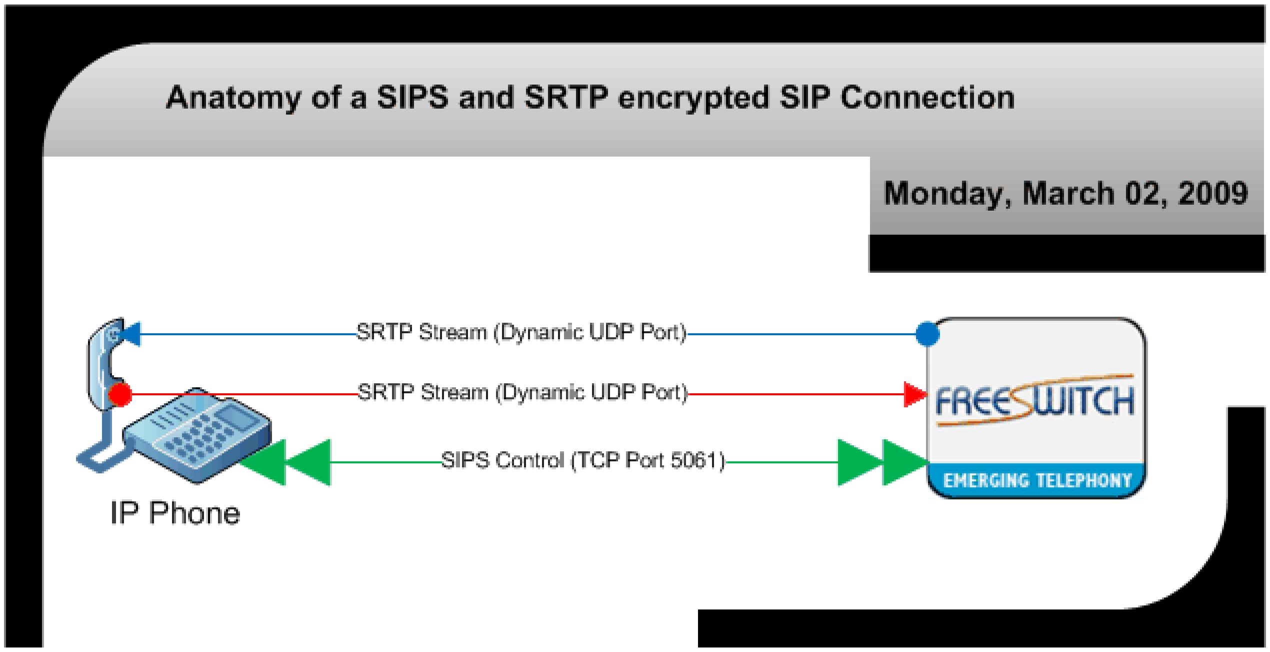
Anatomy of a SIPS and SRTP encrypted SIP Connection
Monday, March 02, 2009
SRTP Stream (Dynamic UDP Port)
SRTP Stream (Dynamic UDP Port)
SIPS Control (TCP Port 5061)
IP Phone
FREESWITCH
EMERGING TELEPHONY

Figure 3:

In figure 3 you'll note that not only is the control channel encrypted, but the RTP streams are also encrypted and sent as SRTP. In this way, and only in this way, is your entire conversation reasonably secure from eavesdroppers. This is also important since the DTMF digits can also be sent in the RTP data as well (inband). Having both SIPS and SRTP encryption is your best option when it comes to making sure your SIP calls are secure.

***Note:** *IP phones may advertise they support encryption, you will need to check the phones specifications to determine if it supports SRTP. More will be said on this below along with interoperability issues with certain phones at the bottom of this page.*

FreeSWITCH Encryption Capabilities

This page will cover only TLS, SSL and SRTP encryption and does not cover ZRTP as of yet.

TLS, SSL and SRTP Encryption

FreeSWITCH supports SIPS via SSL and/or TLS, along with SRTP encryption capabilities. It is unclear as to whether or not FreeSWITCH will encrypt video streams since this has not been tested in the kjv lab, however I suspect that it will. If you conduct testing encrypting SRTP video, by all means add your results to the Wiki.

ZRTP Encryption

ZRTP is an opportunistic encryption capability that can be compiled into FreeSWITCH. See the ZRTP page for the how-to and additional information pertaining to ZRTP.

Hybrid Encryption

FreeSWITCH can in fact encrypt data for SIP traffic that passes through it. In this manner you can provide encryption for IP phones that don't natively support encryption. Since there is a plethora of SIP hardware that doesn't support encryption, this configuration may prove useful for any number of practical situations.

Figure 4:

In figure 4 you'll notice that the SIP control channel from the phone to the switch is not encrypted whereas the control channel from FreeSWITCH to the other SIP PBX is encrypted. It's important to understand that in this situation SIP control and RTP (audio) data is both unencrypted between the IP phone and FreeSWITCH. Hence, if your SIP IP phone is somewhere behind a NAT router on a cable modem, and FreeSWITCH is not behind this router, both the phone's LAN and the Internet connection to FreeSWITCH are entirely in the open, meaning that someone could sniff those packets and obtain valuable information.

Furthermore, the RTP (audio) data between FreeSWITCH and the remote endpoint illustrated as "Other SIP PBX" are also unencrypted and can be intercepted, however the control channel for this connection is encrypted and therefore reasonably secure. If FreeSWITCH were on the same LAN as the phone the possibility of interception is less, but the audio data is still capable of being intercepted.

Figure 5:

In figure 5 you'll notice that both SIP and RTP data from the IP phone is unencrypted, while the SIP and RTP data to the other SIP PBX is both SIPS and SRTP encrypted. This option will encrypt both the control channel and the audio data for the connections between FreeSWITCH and other endpoints that support encryption.

It still needs to be understood that the connection from the phone to FreeSWITCH is entirely capable of being intercepted by a packet sniffer.

Choosing Between Encryption Options

There are a variety of encryption options for FreeSWITCH as described above. TLSV1 encrypts everything over the TCP connection; this has the downside that jitter or delays due to TCP can occur. UDP is generally preferred for RTP and using TLSV1 has some additional traffic overhead. ZRTP has the advantage that it allows for end-to-end encryption without pre-exchanging keys or certificates but is a bit more complex to set up, especially for some clients. Finally there is SSLV23; this encrypts the SIP control channel over an encrypted TCP connection using SSL certificates, but by default doesn't provide anything for the RTP data. The login information and call metadata is transmitted over the control channel so if this is what you care about protecting this is enough and and adds no overhead to the RTP data. If you want to encrypt the voice data itself (so the voice data cannot be understood) it can be combined with SRTP. SRTP enables encryption of the RTP data with minor overhead to each of the RTP UDP packets. This has the benefit that the call data are encrypted but still is over UDP so there should be virtually no difference with SRTP on as with it off. The encryption key used for SRTP is exchanged over the control channel (which SSLV23 encrypts) so gives you something of the best of both worlds. Generally SSLV23 + SRTP is the most firewall friendly/has the fewest changes for existing installations. SSLv23 + SRTP is also fairly easy to configure on the FreeSWITCH server and most likely the most supported encryption method for most clients and SIP phones so generally should be where someone should start for encrypting call data.

About IP Phones and Encryption

Some vendors will advertise that their product supports SIP encryption. The question becomes how well it supports SIP encryption. So let's review this convoluted marketing conundrum.

- The phone may support SSL and/or TLS encrypted SIP, which is to say "SIPS".
- The phone may go so far as to support TLS with signed certificates.
- Above and beyond those caveats, the phone may go so far as to support SRTP encrypted audio data.
- Last but not least, make sure the IP phone supports **standard** SIPS and/or SRTP encryption. Essentially it's not too far fetched for a manufacturer to come up with a hair-brained encryption method which only works with their products.

WARNING: Check the specifications and do your research before buying any SIP IP phone that advertises itself as capable of performing encryption. You may also want to check to see if there are any issues with the version of SIP firmware your phone is running and its interoperability with FreeSWITCH.

Limitations of the Freeswitch TLS/SSLv23 Implementation (FS-3877)

Unfortunately the default lib_sofia stack and FreeSWITCH configuration has one default (possibly unexpected) behavior which can cause hard-to-diagnose encryption problems. When a client connects to a server it creates an encrypted TCP channel used to communicate with the server. Anytime it places a call, registers, etc., it does so using this TCP session. Unfortunately, when FreeSWITCH attempts to communicate with the client to tell it about an incoming call, NOTIFYs, etc., it establishes a second TCP channel from the server to the client. While this is encrypted, it can cause some problems for some firewall configurations and for many of the TLS/SSL security options in FreeSWITCH. If FreeSWITCH cannot establish this channel the client will be able to make calls but it will not be able to receive calls (or anything else that is originated from the server to the client). In addition, many TLS features may act unexpectedly, so direction is important to consider. For example, tls-verify-policy set to "out" would cause FreeSWITCH to try to verify the client certificate when it connects back to the client, so even though the client is connecting to the server initially and that would skip validation for calls from the server->client, FreeSWITCH would verify it. In addition, this can open security holes, for example setting the policy to "in" for verification of incoming clients would NOT verify the client certificate when the server connects back to the client, which would allow a MITM attack to occur there. Things like subject validation can also be tricky, especially if FreeSWITCH is acting as a client. For example, subject validation checks the hostname to which FreeSWITCH connects against the certificate common name, while a nice security check cannot possibly be done when the server connects back to FreeSWITCH (as at that point FreeSWITCH actually tries to check the IP address against the common name). In addition, purpose validation (making sure when FreeSWITCH is connecting to a server its a certificate that has been issued for a server and not a client, or vice versa) currently is not supported. This allows MITM attacks against a FreeSWITCH server using a client certificate (or vice versa), but due to the connection going both ways, purpose validation is currently not supported (as an incoming connection may be from a server certificate while connecting back). So while FreeSWITCH's gentls_cert does properly assign the purpose it is not currently used by FreeSWITCH itself. One solution to avoid this MITM hole is to issue client certs from a different CA for now.

Configuration

FreeSWITCH supports the encryption of SIP signaling traffic via SSL and/or TLS. The convention is to run the SIPS on port 5061. More complex configurations are possible, however they will not be covered in this documentation.

You will need the following in order to compile FreeSWITCH with TLS encryption support:

- libssl-dev (the OpenSSL development) package installed.

If you do not have the libssl-dev package installed, the problem will be the sip_profile that contains the encryption configuration directives specified below in Step 2 will fail to start.

```
If you've compiled FreeSWITCH without this package installed, there will be no support for encryption and you
will need to recompile it after you install the libssl-dev package.
```

For Debian do aptitude install libssl-dev (on Lenny install libcurl4-openssl-dev) and then compile.

Step 1 - Generate the CA (Root) Certificate

To use TLS/SSL you need at least two certificates: the root (CA) certificate and a certificate for every server. There is a script at */{prefix}/freeswitch/bin /gentls_cert* or within the source tarball *{tarball}/scripts/gentls_cert* that helps generate these files. Assuming that the DNS name of your FreeSWITCH PBX is pbx.freeswitch.org, With

```
./gentls_cert setup -cn pbx.freeswitch.org -alt DNS:pbx.freeswitch.org -org freeswitch.org
```

This will create CA certificate and key along with in conf/ssl/CA directory and certificate in the conf/ssl folder.

[**Note:** The name given for -cn and -alt should be the same as the DNS name of your freeswitch installation and used as the registrar name on the phone (at least on Polycoms).] You can change the "DAYS=2190" line in the gentls_cert file to make the certificate valid for longer time. However making it too long has some wrap around problem, it appears.

Step 2 - Generate the Server Certificate

The command:

```
./gentls_cert create_server -cn pbx.freeswitch.org -alt DNS:pbx.freeswitch.org -org freeswitch.org
```

creates the server certificate at */{prefix}/freeswitch/conf/ssl/agent.pem*. This file contains the certificate and the private key. It should contain the domain name in the common and alternate name. If you need to generate certificates for other servers use the -out flag for gentls_cert to specify the output certificate/key file name and copy this to the remote server.

To set up new CA and create new certificate under Windows go here.

In order for the new certificate to take effect (the only way for FreeSWITCH to use it), FreeSWITCH must be restarted.

Note: The name given for -cn and -alt should be the same as the DNS name of your freeswitch installation and used as the registrar name on the phone (at least on Polycoms). This is required for Eyebeam (and probably Pangolin too).

Review your certificate

You can review your certificate details with the command:

```
openssl x509 -noout -inform pem -text -in /usr/local/freeswitch/conf/ssl/agent.pem
```

Step 3 - Sofia Profile Configuration

Freeswitch requires only one file for acting as a TLS server and thats the agent.pem file. This contains the certificate and key it will use for listening. **Note it is extremely important that your agent.pem (and optionally cacert.pem) have read permissions for the user freeswitch will run as.** That means if you use -u freeswitch you want to "chmod 640 agent.pem cacert.pem" and "chown root.freeswitch agent.pem cacert.pem". Incorrect permissions will not allow the TLS listener to spin up properly. On your vars.xml:

ATTENTION: TLS is disabled by default; set internal_ssl_enable and/or external_ssl_enable to "true" to enable.

```
<!--
    SIP and TLS settings.
-->
<X-PRE-PROCESS cmd="set" data="sip_tls_version=sslv23"/>

<!-- Internal SIP Profile -->
<X-PRE-PROCESS cmd="set" data="internal_auth_calls=true"/>
<X-PRE-PROCESS cmd="set" data="internal_sip_port=5060"/>
<X-PRE-PROCESS cmd="set" data="internal_tls_port=5061"/>
<X-PRE-PROCESS cmd="set" data="internal_ssl_enable=false"/>
<X-PRE-PROCESS cmd="set" data="internal_ssl_dir=$${base_dir}/conf/ssl"/>

<!-- External SIP Profile -->
<X-PRE-PROCESS cmd="set" data="external_auth_calls=false"/>
<X-PRE-PROCESS cmd="set" data="external_sip_port=5080"/>
<X-PRE-PROCESS cmd="set" data="external_tls_port=5081"/>
<X-PRE-PROCESS cmd="set" data="external_ssl_enable=false"/>
<X-PRE-PROCESS cmd="set" data="external_ssl_dir=$${base_dir}/conf/ssl"/>
```

Step 4 Client Configuration

Clients should all have at least the CA root certificate installed onto them in order to ensure security. Without enabling chain verification (that the server certificate was issued by the approved CA) a MITM attack is possible against a client. The CA certificate is the conf/ssl/cafile.pem it contains only a certificate and clients use it to ensure the server certificate is issued by the CA. In addition you may want (or have to depending on the device) install a client certificate issued by the same CA. You can generate a client certificate using the command:

```
./gentls_cert create_client -cn Client1 -out Client1
```

This will create Client1.pem (note the common name here doesn't really matter, rarely would a server ever be enabled to check the subject of a client certificate so you can set this to what you want). This certificate/key pair should be installed on the client (recommended a different for each client for security) but does NOT need to remain on the server, even if you want the server to verify the client certificate. If the client is another freeswitch server that Client1.pem you would want to rename to agent.pem. Note IF the other freeswitch server is going to also use this certificate as a server and not just a client make sure to follow the steps above using the create_server command to create a server certificate.

Commercial Certificates

Most commercial Web server certificates should work for FreeSWITCH. For example, Thawte SSL123 certificates (their cheapest option) are known to work in at least one installation. If you have doubts about whether a certificate will work, you should only purchase the certificate from a company offering 30-day refund policy.

Multiple Profile TLS

If you have multiple Sofia SIP profiles, you may find yourself wanting to enable TLS support for each of the profiles. However, each may be represented to third parties using a different DNS record. In this case, simply create a new directory under /{prefix}/freeswitch/conf/ssl/ for each DNS record . Then place an agent.pem and cafile.pem into each of the directories. Point each profile to the individual directory which contains its specific agent.pem file.

Troubleshooting

Issues with TLSV1

If you've opted for this configuration in your sip_profile:

```
<param name="tls-version" value="tlsv1"/>
```

There are a number of gotchas and snafus possible with TLS. TLS runs on TCP, rather than UDP, so make sure any firewalls are configured to work with TCP. Generate the proper certificate, and make sure it's loaded into the phone or ATA. Also, ensure that the time is configured properly on your endpoint as you will get cryptic "bad certificate" error messages if the time is too far off, and it will fail to handshake properly. (For example, certificates have "not valid before" and "not valid after" attributes.) If your phone won't register, or fails to work properly, you might consider checking the Interoperability list, and /or your phones documentation and firmware revision. If it still fails to work properly, you might consider sslv23 as a fallback.

Issues with SSLV23

If you've opted for this configuration option in your sip_profile:

```
<param name="tls-version" value="sslv23"/>
```

This will give you the ability to encrypt the data. Security is similar to that of with web servers, certificate trusted roots can be checked, client certificate authentication is supported, subjects and cert dates can be validated (some of these do depend on client support for the feature). This may in fact be what you want, and considering that several phones do not support TLS, but do support SSL, you may want to configure accordingly.

If your SIP profile fails to start up

If you go into the FreeSWITCH CLI and type:

```
sofia status
```

You might find that your sip_profile for which you configured the TLS/SSL settings is missing. Should that be the case, you're probably missing the libssl-dev package or the agent.pem file is not in place or has incorrect permissions. Ensure the user FreeSWITCH runs as has read permissions to the certificates and keys. Turning on sofia tport debug logging to high will help print out errors on why it is not starting up properly. Refer to your distribution in order to install the OpenSSL development package. This is necessary for the code which performs the encryption within FreeSWITCH to be compiled into the binary. Enabling a higher level of debug in sofia (sofia loglevel all 9, or sofia loglevel tport 9 generally give enough) and in the console can help debug issues and handshake problems.

TLS Negotiation Error 'No matching cipher'

If you receive a 'no matching cipher' error message, first check to ensure you have concatenated your certificate key into your agent.pem file. If you still receive the error, regenerate both the key and the certificate and make a new agent.pem with the new key and certificate, then try again.

Further Debugging Steps

If you are still having issues getting TLS running the following are suggested:

-) set sofia loglevel to 9 (<param name="log-level" value="9"/>) in autoload_configs/sofia.conf.xml
-) Set freeswitch console loglevel to debug (F8)
-) Reload mod_sofia and review the the logs
-) If everything looks fine on the server side, try using freeswitch or a freeswitch based Softphones like FSClient or FSComm. These will allow you to do the same on the client side (including attach fs_cli to them to get log data) to try and see any errors the client may be having
-) Last resort put all the logs on pastebin and ask for help.

Using Freeswitch as a SSL/TLS Client

Freeswitch can communicate with another server and act as a sip client (AKA use another gateway). To do this you should have the cafile.pem on the client in the conf/ssl folder with the CA certificate. You will also need an agent.pem file (this can be a randomly generated public/private key pair or a valid client certificate) in the same folder used as the client certificate.

Advanced TLS Options

Freeswitch supports some advanced TLS parameters in the profile settings see Sofia Configuration Files for details.

Quick Setup of SSLv23 with SRTP

Quick Install on the Internal Profile if your server internal hostname is pbx.freeswitch.org:

```
./gentls_cert setup -cn pbx.freeswitch.org -alt DNS:pbx.freeswitch.org -org freeswitch.org
./gentls_cert create_server -cn pbx.freeswitch.org -alt DNS:pbx.freeswitch.org -org freeswitch.org
```

If your clients support/validate the server certificate copy the ca certificate conf/ssl/cafile.pem to your clients.

and in vars.xml

```
<X-PRE-PROCESS cmd="set" data="sip_tls_version=sslv23"/>
<X-PRE-PROCESS cmd="set" data="internal_ssl_enable=true"/>
```

Have your clients connect to freeswitch on port 5061 instead of 5060 and enable TLS (and if possible SRTP) on the client. If your client is freeswitch setting transport=tls under tls-bind-params in addition to the enable options above from the server. Finally on your freeswitch server for calls bound for the client you will want to enable them to go over SRTP by setting rtp_secure_media=true for the call (See Step 5 below for more details on how to do this more programatically for clients that only support this).

Restart your FreeSWITCH server (or reload mod_sofia) and have your client connect and you should be able to make and receive secure calls! To improve security check the full TLS options and enable certificate checking (also to stop MITM).

Step 5 - Securing the RTP Channels (optional)

Calls that originate from the phone have rtp_secure_media set if TLS is set up. Check the global extension. There is a section commented that out will require SRTP on the outbound leg if the inbound leg is encrypted. Enabling this will be problematic with most ITSPs since they do not support TLS.

For calls that originate from (or are routed through) FreeSWITCH and are terminated on the user/ endpoint (e.g., calls to a phone), the following change will enable SRTP if the endpoint registered with TLS. Note that it is a valid configuration to register with TLS but not require SRTP. This disables that valid configuration option for user/ endpoints. It would also require further refinement to support ZRTP on user/ endpoints that connect with TLS. In that case, a better approach would be to set something on the user's directory entry that specifies which RTP encryption to support. (In other words, there is a reason this is not the default setting.)

Edit conf/directory/default.xml and change the **dial-string** param to:

```
 <param name="dial-string" value="{rtp_secure_media=${regex(${sofia_contact(${dialed_user}@${dialed_domain})}
|transport=tls)},
  presence_id=${dialed_user}@${dialed_domain}}${sofia_contact(${dialed_user}@${dialed_domain})}" />
```

Why it's a good idea

In the SIP Encryption Primer above we discussed why encrypting the RTP data may be a good idea. This is largely done in the dialplan and has its own page dedicated to its functionality.

SRTP by itself without TLS is not secure since the keys are exchanged between the two endpoints in the clear over SIP, which is insecure without TLS or SSL.

See Secure RTP page for how to deploy SRTP.

Note that the `sip_secure_media` parameter is no longer implemented and you should be setting `rtp_secure_media`

For completely secure connection (signaling + media) use TLS + SRTP. TLS without SRTP secures SIP. SRTP without TLS does not really secure RTP!

Step 6 - DNS NAPTR & SRV Records (optional)

Configuring DNS NAPTR and SRV for TLS

This isn't required as TLS will function without it, but it is recommended. If you're going to set up TLS you might take the time to set up the correct DNS SRV and NAPTR records.

Here is an example of the NAPTR records you will need (make subtitutions for "domain.com" as necessary):

```
domain.com.               IN      NAPTR 10 0       "s"     "SIPS+D2T"  "" _sips._tcp.domain.
com.
domain.com.               IN      NAPTR 20 0       "s"     "SIP+D2T"   "" _sip._tcp.domain.
com.
domain.com.               IN      NAPTR 30 0       "s"     "SIP+D2U"   "" _sip._udp.domain.com.
```

If you configure with --enable-sctp and have sctp loaded on Linux you can add this record:

```
domain.com.               IN      NAPTR 10 0       "s"     "SIPS+D2S"  "" _sip._sctp.domain.com.
```

Then you will want to set up these SRV records in addition to the above NAPTR:

```
_sips._tcp.domain.com.       IN      SRV 10 0 5061   sip.domain.
com.
_sip._udp.domain.com.        IN      SRV 10 0 5060   sip.domain.
com.
_sip._tcp.domain.com.        IN      SRV 10 0 5060   sip.domain.com.
```

If you configure with --enable-sctp and have SCTP loaded on Linux you can add this record:

```
_sip._sctp.domain.com.       IN      SRV 10 0 5060   sip.domain.com.
```

Example: Snom phones will fully honor both the NAPTR and SRV as will Sofia-SIP on outbound calls originating from FreeSWITCH.

SIP TLS Device Interoperability

Aastra TLS Setup

See Aastra TLS Setup

Polycom TLS Setup

See Polycom TLS Setup

Linksys TLS Setup

See Linksys TLS Setup

Snom TLS Setup

See Snom TLS Setup

EyeBeam/Bria Setup

See Bria TLS Setup and/or Eyebeam TLS Setup

Nokia / Symbian S60 Setup

See Nokia and/or Symbian TLS Setup