Encrypted mobile to landline phone calls with Asterisk 1.8, Snom 300 and PrivateGSM Enterprise

Introduction

GSM has been cracked and and now mobile phone calls can be intercepted with cheap hardware and Open Source tools.
VoIP can be intercepted with simple windows point and click VoIP interception tool.
VoIP encryption protocols exists to protect from eavesdropping:

- SRTP does end-to-site voice encryption
- SDES does key exchange for SRTP
- SIP/TLS encrypt signaling channel over which SDES keys are exchanged

Till now Asterisk, the most used Open Source telephony engine, did not support it but Asterisk 1.8 finally supports SRTP voice encryption with SDES key exchange!

This howto explains the building of an integrated mobile to landline phone calling platform with:

- Asterisk 1.8 RC2
- PrivateGSM Enterprise for mobile secure calling (iPhone, Blackberry and Nokia S60)
- Snom 300 for landline secure calling

The procedures described below will guide you in the following steps:

1. Install Asterisk
2. Configure Asterisk
3. Configure Snom 300
4. Configure PrivateGSM Enterprise
5. Make secure calls

Please note that the security provided is end-to-site, that means that no one except the PBX system administrator can eavesdrop in on the call.

If you need end-to-end security (typical government need) you need to use ZRTP that does end-to-end encryption.

Target audience

This howto requires an understanding of the following areas of technology:

- Unix / Linux system administration knowledge
- VoIP networking
- IP networking
- Asterisk installation and configuration experience

Please note that this howto is not for newbies and require this specific expertise and skill set.

Infrastructure planning

You may need to setup different kinds of secure infrastructure depending on the secure calling needs and configuration could require specific configuration to work properly in certain setups:

- Mobile to Landline
- Landline to Landline
- Mobile to Mobile

You can use several IP transport networks:

- LAN/Broadband (Landline secure phones)
- WiFi/Broadband (Mobile secure phones)
- Mobile/Narrowband (mobile secure phones)

Below are shown several infrastructure setups that can be built:

- **LAN environment**

Snom 300 ---LAN---> Asterisk ---LAN--> Snom 300

- **LAN environment with WIFI**

Snom 300 ---LAN-----> Asterisk ---WIFI----> PrivateGSM Enterprise

- **LAN environment and UMTS/GPRS**

PrivateGSM Enterprise (Private GSM)amr codec --UMTS/GPRS--> Asterisk ---LAN--> Snom 300


- **UMTS/GPRS environment**

PrivateGSM Enterprise (Private GSM) amr codec ---UMTS/GPRS---> Asterisk --UMTS/GPRS-> PrivateGSM


Enterprise (Private GSM)amr codec

Technologies

The technologies used to build this setup are the following:
Voice encryption

- SRTP (AES128) with 32bit hash

Signaling encryption

- SIP/TLS with x509v3 certificates

Key Exchange

- SDES (AES128) over SIP/TLS

Codec for LAN

- GSM 13kbit

Codec for Mobile

- AMR 4.75kbit / 12.2kbit

They get implemented by the following set of software tools that interoperate with each other.

Please consider that AMR codec **must** be used because it's 4.75kbit and G.729 8kbit codec does not work for mobile network connections in a reliable way.

For example G.729 or G.723.1

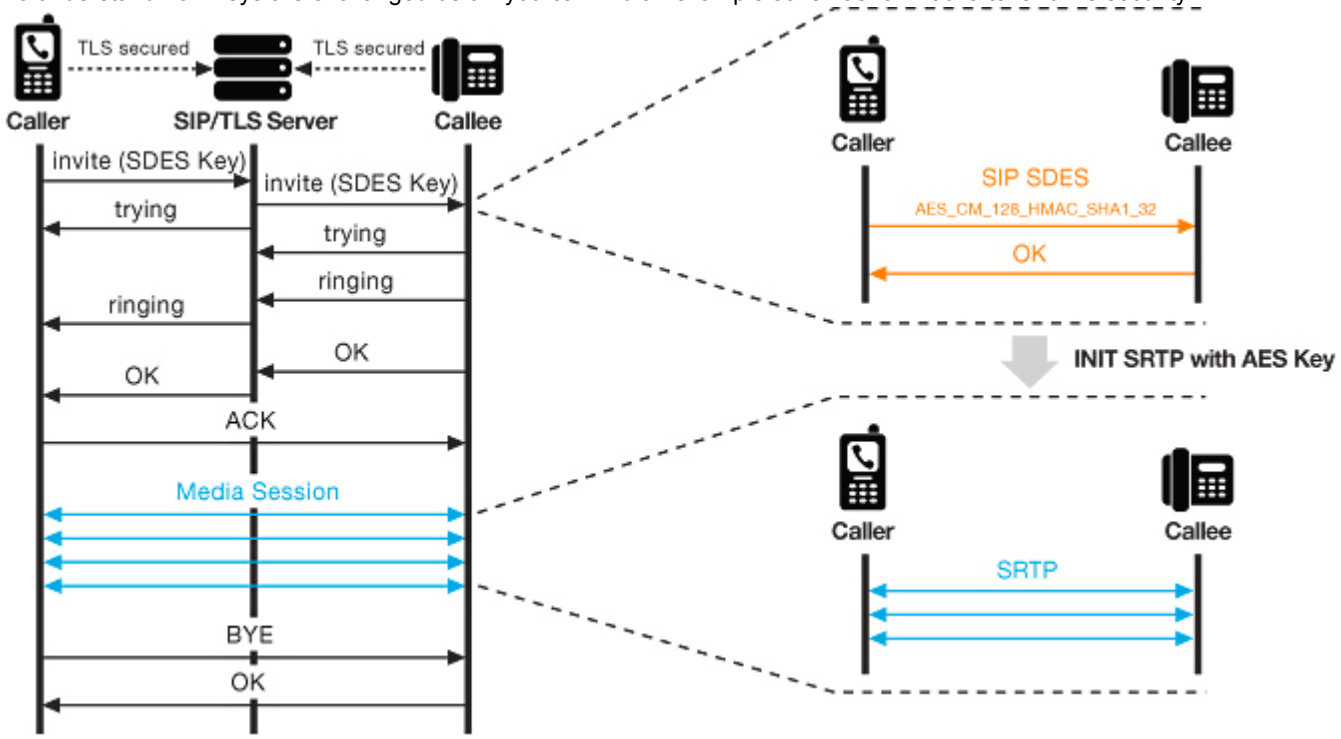
- does not perform over GPRS
- does not work in a degraded network conditions

AMR codec has been designed specifically for mobile networking, it's bit resistant and ultra-narrowband, so currently it's the only reliable solution to handle GPRS and high packet loss.

Any attempt to use other codecs such as G.729 or G.723 will be revealed, after some time of **real-scenario-usage**, to not be suitable for mobile networking.

Security model

The overall security model of Voice Encryption by using SRTP with SDES key exchange over SIP protected with TLS communication security is exactly the same as **HTTPS**, so it's **end-to-site security that protects against third party attempt to eavesdrop in on phone calls**.
To understand how keys are exchanged below you can find an example schemas for mobile-to-landline security:



For any end-to-end security needs (because your security model requires you to not trust the PBX administrator) the right security technology is ZRTP.

Asterisk 1.8

Asterisk is the most known and used telephony engine and PBX .
It's designed to work with most VoIP technologies and it requires for our environment specific tuning and configuration but given its flexibility everything works fine.
This howto has been written with Asterisk 1.8 RC2 in mind, but it works fine with Asterisk 1.8 final release.

Hardware supported: PC/Embedded
OS Supported: Linux & various unix
Codec supported: GSM (for this environment we need only this)
Manufacturer: Digium - http://www.Asterisk.org

PrivateGSM Enterprise

PrivateGSM Enterprise is the most performant mobile secure VoIP client for Nokia S60, iPhone and Blackberry phones that support VoIP Security protocols (SRTP and SIP/TLS).
It has been specifically created to work in a difficult networking environment such as mobile networks and **has advanced TLS digital certificate verification checking that no other mobile voip client has.**
It uses mostly Open Source components for low level VoIP encryption and multimedia.
A different product version, PrivateGSM Professional, does end-to-end encryption with ZRTP.

Hardware supported: Nokia / iPhone / Blackberry
OS Supported: SymbianOS /iOS 4 / RIMOS 5
Codec supported: AMR 4.75 (Nokia, iPhone), AMR 12.2 (Blackberry)
Manufacturer: PrivateWave - http://www.privatewave.com

Snom 300

Snom technology AG produces VoIP telephones for business communications which are based on the open standard SIP and are one of the few supporting VoIP Security protocols (SRTP and SIP/TLS).
They have been specifically created to work in LAN networking environment and **have advanced TLS digital certificate verification checking that no other desktop voip phone has**.
Snom use Linux as underlying OS and Snom release GPL source code as a part of the phone firmware.

Hardware supported: DeskPhone / WiFi Phone / Conference Phone
OS Supported: Linux (Linux based ROM)
Codec supported: GSM 13kbit (LAN)
Manufacturer: Snom - http://www.Snom.com

Installing Asterisk

Asterisk installation Required tools :

1. libSRTP
2. Asterisk 1.8 RC2
3. Linux Distribution

Asterisk 1.8 comes with SRTP and TLS, SRTP use SDES (defined in RFC4568) key exchange.

We start from a fresh install with Fedora Core 13 or CentOS 5.X

Update the system & reboot:

1. yum -y update

Install wget to pull down Asterisk:

2. yum -y install wget

Download and compile Asterisk

Download Asterisk , DAHDI, libpri, libSRTP. First got to /usr/src/ directory

3. cd /usr/src/
4. wget http://downloads.Asterisk.org/pub/telephony/Asterisk/Asterisk-1.8-current.tar.gz
5. wget http://downloads.Asterisk.org/pub/telephony/dahdi-linux-complete/releases/dahdi-linux-complete-2.4.0+2.4.0.tar.gz
6. wget http://downloads.Asterisk.org/pub/telephony/libpri/releases/libpri-1.4.11.4.tar.gz
7. wget http://sourceforge.net/projects/srtp/files/srtp/1.4.4/srtp-1.4.4.tgz/download

Settle Asterisk and libSRTP dep's

8. yum -y install kernel-devel gcc make gcc-c++ libxml2-devel pkgconfig zlib-devel openssl-devel ncurses-devel autoconf automake libtool zip unzip

Extract Asterisk & DAHDi & libpri & libsrtp

9. tar -xzvf Asterisk-1.8-current.tar.gz
10. tar -xzvf dahdi-linux-complete-2.4.0+2.4.0.tar.gz
11. tar -xzvf libpri-1.4.11.4.tar.gz
12. tar -xzvf srtp-1.4.4.tgz

Compile srtp, dahdi , libpri and Asterisk (the compilations order is important)

13. cd /usr/src/srtp
14. ./configure --prefix=/usr
15. make
16. make runtest
17. make install

If Is used Fedora 13 X86_64 use the following insted :

18. make clean
19. CFLAGS="-Wall -O4 -fexpensive-optimizations -funroll-loops -fPIC" ./configure --prefix=/usr
20. make
21. make runtest
22. make install

Time to compile dahdi

23. cd ../dahdi-linux-complete-2.4.0+2.4.0 (now we are in /usr/src/dahdi-linux-complete-2.4.0+2.4.0)
24. make all
25. make install
26. make config

Time to compile libpri

27. cd../libpri-1.4.11.4 (now we are in /usr/src/libpri-1.4.11.4)
28. make
29. make install

Time to copmpile Asterisk

30. cd ../Asterisk-1.8.0/
31. ./configure
32. make menuconfig

Running this command will be show the following menu :

```
**********************************************
    Asterisk Module and Build Option Selection
**********************************************

               Press 'h' for help.

               Add-ons (See README-addons.txt)
               Applications
               Bridging Modules
               Call Detail Recording
               Channel Event Logging
               Channel Drivers
               Codec Translators
               Format Interpreters
               Dialplan Functions
               PBX Modules
          --->  Resource Modules
               Test Modules
               Compiler Flags
               Voicemail Build Options
               Utilities
               AGI Samples
               Module Embedding
               Core Sound Packages
               Music On Hold File Packages
               Extras Sound Packages
```

Go to Resource Modules and check if SRTP is properly installed :

```
                    [*] res_crypto
                    XXX res_curl
                    [*] res_fax
                    XXX res_fax_spandsp
                    XXX res_http_post
                    XXX res_jabber
                    [*] res_limit
                    [*] res_monitor
                    [*] res_musiconhold
                    [*] res_mutestream
                    XXX res_odbc
                    [*] res_phoneprov
                    [ ] res_pktccops
                    [*] res_realtime
                    [*] res_rtp_asterisk
                    [*] res_rtp_multicast
                    [*] res_security_log
                    [*] res_smdi
                    XXX res_snmp
                    [*] res_speech
                    [*] res_srtp
                    [*] res_stun_monitor
                    [*] res_timing_dahdi
                    XXX res_timing_kqueue
                    [*] res_timing_pthread
                    XXX res_timing_timerfd


               Secure RTP (SRTP)
               Depends on: srtp(E)
```

You should see res_srtp module present into the configuration menu **[*] res_srtp .**
Now that we have checked that srtp is present and successfully compiled and seen from Asterisk we can continu onto the Asterisk compilation step.

33. make
34. make install
35. make config
36. make samples (this command will generate sample configuration files for Asterisk, under /etc/Asterisk/ folder)

We have successfully compiled srtp ,dahdi, libpri ,Asterisk (in this order).
Now is time to start dahdi and Asterisk :

37. service dahdi start
38. service asterisk start

Asterisk is successfully running, now we should configure TLS and SRTP in order to make **encrypted voip calls** between user agents.

Add AMR codec support

Before start to configure Asterisk, we would like to add one more tool, we reallly need to use ARM-NB codec (Adaptive-Multirate-Codec Ultra Narrow Band) in conjunction with PrivateGSM Enterprise on a mobile phone to get the best voice quality using mobile internet connections.
Please consider that G.729 codec does not work for mobile network connections in a reliable way, for example it does not work over GPRS or in degraded network conditions (that on a mobile, means often).

Be sure to read **notes on licensing** at the end of this howto in order to properly license the AMR codec.

Stop Asterisk

39. service asterisk stop

We go back to /usr/src/ folder,download the Asterisk-amr patch.

40. cd /usr/src/

Download Asterisk-amr patch

41. wget https://sourceforge.net/projects/Asterisk-amr/files/1.8.0-rc2_Asterisk_amr_patch.diff/download

Apply the patch for AMR codec

42. cd Asterisk-1.8.0/ && patch -p2 < ../1.8.0-rc2_Asterisk_amr_patch.diff

Go to Asterisk-1.8.0/codecs/amr/ folder

43. cd Asterisk-1.8.0/codecs/amr/

Download AMR codec from 3GPP site

44. wget http://www.3gpp.org/ftp/Specs/archive/26_series/26.104/26104-700.zip
45. unzip -j 26104-700.zip
46. unzip -j 26104-700_ANSI_C_source_code.zip

go back to Asterisk folder /usr/src/Asterisk-1.8.0/

47. cd ../.. /..

Now we have Asterisk 1.8 SRTP/TLS AMR-NB compiled .

Patch SRTP for 32bit

SRTP have a cryptographic hash to check the integrity of the encrypted packets.
It support two hash size:

- 32bit
- 80bit

In order to properly fine tune SRTP for mobile networks and to have compatibility with PrivateGSM Enterprise we must use SRTP with hash at 32bit (HMAC_SHA1_32).

Asterisk 1.8 by default does not announce in SDP both 32bit and 80bit, but only the 80bit version even if both are supported.
This very small 1 line patch make Asterisk by default work with SRTP hash at 32bit .

Download the patch for HMAC_SHA1_32 RTP crypto offer

48. wget http://sourceforge.net/projects/Asterisk-amr/files/1.8.0-rc2_crypto_offer.diff/download

Apply the patch

49. cd Asterisk-1.8.0/ && patch -p2 < ../1.8.0-rc2_crypto_offer.diff

Go to Asterisk-1.8.0/ folder

50. cd ..

Recompile Asterisk ,

51. make ; make install

Check SRTP support

Be sure now to check that SRTP support is properly installed:

52. module load res_srtp.so

This command will show the following output if res_srtp is present and already loaded

```
*CLI> module load res_srtp.so
Unable to load module res_srtp.so
Command 'module load res_srtp.so ' failed.
WARNING[30610]: loader.c:829 load_resource: Module 'res_srtp.so' already exists.
```

Check AMR support

Be sure now to check that AMR codec support is properly installed:

53. rasterisk

54. core show codecs

```
*CLI> core show  codecs
        1 (1 <<  0)          (0x1)  audio        g723   (G.723.1)
        2 (1 <<  1)          (0x2)  audio        gsm    (GSM)
        4 (1 <<  2)          (0x4)  audio        ulaw   (G.711 u-law)
        8 (1 <<  3)          (0x8)  audio        alaw   (G.711 A-law)
        16 (1 <<  4)                (0x10)  audio   g726aal2   (G.726 AAL2)
        32 (1 <<  5)                (0x20)  audio      adpcm   (ADPCM)
        64 (1 <<  6)                (0x40)  audio      slin    (16 bit Signed Linear PCM)
        128 (1 <<  7)               (0x80)  audio      lpc10   (LPC10)
        256 (1 <<  8)               (0x100)  audio     g729    (G.729A)
        512 (1 <<  9)               (0x200)  audio     speex   (SpeeX)
        1024 (1 << 10)              (0x400)  audio     ilbc    (iLBC)
        2048 (1 << 11)              (0x800)  audio     g726    (G.726 RFC3551)
        4096 (1 << 12)              (0x1000)  audio    g722    (G722)
        8192 (1 << 13)              (0x2000)  audio           amr   (AMR NB)
        16384 (1 << 14)             (0x4000)  audio    siren14   (ITU G.722.1 Annex C, (Siren14, licensed from Polycom))
        32768 (1 << 15)             (0x8000)  audio    slin16   (16 bit Signed Linear PCM (16kHz))
        65536 (1 << 16)             (0x10000)  image           jpeg   (JPEG image)
        131072 (1 << 17)            (0x20000)  image           png   (PNG image)
```

Configuring Asterisk

In this section you will find details on how to properly configure Asterisk for Security and Mobile networking setup.
Be sure to follow all steps.

Digital certificates

In order to protect SIP signaling with TLS, exactly the same way HTTPS works, we must properly handle digital certificates.
By default insecure VoIP uses unencrypted UDP for the signaling that's unsecure and can be eavesdropped.

Because with SRTP with SDES key exchange, the keys are exchanged over the SIP/TLS authenticated and encrypted channel, the understanding of configuration and handling of digital certificate **it's extremely sensitive.**

When setting up digital certificates you have 3 choice that you can follow:

- Buy a digital certificate from a certification authority (CA)
- **Create your own CA, create your own digital certificate and sign it with your own CA**
- Create a self-signed digital certificate

We **strongly discourage** the use of self-signed digital certificates because they are not properly managed by some mobile devices and do not provide a reliable way to manage trusting of certificates in a multi-server environments.

This howto explain how to quickly create your own CA, create your own digital certificate and then sign it with th previously created CA and it's based on the very good howto available on http://octaldream.com/~scottm/talks/ssl/opensslca.html and http://www.mobilefish.com/developer/openssl/openssl_quickguide_create_ca.html .

For more complex PKI and X509v3 certification authority matters please refer to your software or PKI provider documentation.

Generate a Private CA

We'll first create our Private Certificate Authority to handle our certificates so we can later import one CA Root on all phones.

In most unix system with OpenSSL installed there is available a SSL directory where there are useful tools to make our job in creating digital certificate stuff.

cd /usr/lib/ssl/misc
./CA.sh -newca

When prompted for filename, hit return:
CA certificate filename (or enter to create)
Type a password to protect CA private Key:
Enter PEM pass phrase: 1234
Verifying - Enter PEM pass phrase: 1234
Now enter CA root details intelligently:
Country Name (2 letter code) [AU]: IT
State or Province Name (full name) [Some-State]: Italy
Locality Name (eg, city) []: Milan
Organization Name (eg, company) [Internet Widgits Pty Ltd]: PrivateWave Italia SpA
Organizational Unit Name (eg, section) []: Security
Common Name (eg, YOUR name) []: PrivateWave
Email Address []: supportNO-SPAM@privatewave.com
Now type again the passphrase and check that resulting details are the one you inserted:
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/./cakey.pem: 1234
Check that the request matches the signature
Signature ok
Certificate Details:
Serial Number: 0 (0x0)
Validity
Not Before: Oct 19 14:08:24 2010 GMT
Not After : Oct 18 14:08:24 2013 GMT
Subject:
countryName = IT
stateOrProvinceName = Italy
organizationName = PrivateWave Italia SpA
organizationalUnitName = Security
commonName = PrivateWave
emailAddress = support@privatewave.com
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Comment:
OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
4C:DC:7A:46:F6:9D:01:03:B0:68:1B:6B:9E:F6:1B:42:82:9B:32:33

X509v3 Authority Key Identifier:
keyid:4C:DC:7A:46:F6:9D:01:03:B0:68:1B:6B:9E:F6:1B:42:82:9B:32:33

Certificate is to be certified until Oct 18 14:08:24 2013 GMT (1095 days)

Now the CA has been correctly created with our organization name and password protected with **1234** password (choose a safer one!).
The CA root public key is in file **/usr/lib/ssl/misc/demoCA/cacert.pem**

Create Digital Certificate

Now we are going to create the digital certificate for the SIP/TLS server, so it's **highly relevant that you will type the right HOSTNAME that you will later configure into the phones as SIP/TLS server**.

./CA.sh -newreq
Follow the same procedure as you have done for the Private Certification Authority creation but be sure to configure your own **HOSTNAME** as a common name:
Common Name (eg, YOUR name) []: **securesip.yourcompany.com**

The Digital Certificate private key is in file **/usr/lib/ssl/misc/newkey.pem**
The Digital Certificate Signing Request is in file **/usr/lib/ssl/misc/newreq.pem**

Now the TLS Server private key is encrypted and, in order to avoid requiring at any Asterisk restart, to be in console to type the password we need to decrypt the **newkey.pem** :
openssl rsa < newkey.pem > newkey-decrypted.pem
mv newkey-decrypted.pem newkey.pem
Now the private key is stored on the server in a decrypted format.

Sign Digital Certificate

Now it's time to **Sign** with the previously created CA the Digital Certificate request for the server with hostname/common name **securesip.yourcompany.com** .

./CA.sh -sign
Type the password that protect the CA private key and acknowledge the signing request.
The CA Signed Digital Certificate is in file **/usr/lib/ssl/misc/newcert.pem**

To summarize the file that we will need to configure Asterisk SIP/TLS and later the Secure Phones (Snom and PrivateGSM):

TLS Server Private Key: **/usr/lib/ssl/misc/newkey.pem**
TLS Server Digital Certificate: **/usr/lib/ssl/misc/newcert.pem**
CA ROOT public key: **/usr/lib/ssl/misc/demoCA/cacert.pem**

Export digital certificate

Please understand that digital certificates exist in a lot of different formats, like PEM, DER, P7B (pkcs7), PFX (pkcs12) .
The CA ROOT public key and the server certificate that we created before will be required to be imported on PrivateGSM and Snom Phones for proper TLS server certificate authentication.

In this guide we create all PEM format digital certificate that are suitable for use by Asterisk.
However Snom and PrivateGSM enabled smartphones require the CA ROOT public key and the Server certificate to be in **DER** format.

Here you can find a guide to convert digital certificate formats .

To convert the CA ROOT certificate and the Server we created we need to issue the following command:
openssl x509 -outform der -in **/usr/lib/ssl/misc/demoCA/cacert.pem** -out **/usr/lib/ssl/misc/demoCA/cacert.der**
openssl x509 -outform der -in **/usr/lib/ssl/misc/newcert.pem** -out **/usr/lib/ssl/misc/newcert.der**

The ROOT CA certificate that we will need to import to the phone is **/usr/lib/ssl/misc/demoCA/cacert.der** .
The server certificate that we will need to import to the phone is **/usr/lib/ssl/misc/newcert.der .**

SIP/TLS Configuration

Enabling TLS will open up the port 5061/TCP which will add the TCP reliability control to the connection (and the crypto TLS brings).

To configure **sip general configuration** edit sip.conf file as follow :
tlsenable=yes
tlsbindaddr=you-ip-address
tlsprivatekey=/usr/lib/ssl/misc/newkey.pem
tlscertfile=/usr/lib/ssl/misc/newcert.pem
tlscafile=/usr/lib/ssl/misc/demoCA/cacert.pem
tlsclientmethod=tlsv1

To enable a sip account to use sip tls add a sip buddies with following parameters :
transport=tls
port=5061

a full examples is in next part called srtp configuration for sip account **101**

SRTP Configuration

To enble SRTP for a sip account and TLS
Edit file sip.conf

1. nano /etc/Asterisk/sip.conf
   add a SIP account with srtp enabled

[101]
....
encryption=yes
transport=tls
port=5061
See below for further detail on example SIP account configuration for Snom and PrivateGSM Enterprise .

Dialplan to make secure calls

Edit /etc/Asterisk/extensions.conf

1. nano /etc/Asterisk/extensions.conf

add the following context to the end of the file, 600 is user for echo test and show some infomation about TLS and SRTP inside Asterisk CLI, 10X permit secure calls between user agent.

```
[internal]
exten => 10X,1,Set(_SIPSRTP_CRYPTO=enable)
exten => 10X,2,Dial(SIP/${EXTEN})
exten => 600,1,NoOp( start)
exten => 600,n,NOOp( SECURE SIGNALING   ${CHANNEL(secure_signaling)} )
exten => 600,n,NOOp( SECURE media       ${CHANNEL(secure_media)} )
exten => 600,n,Answer()
exten => 600,n,Playback(demo-echotest)
exten => 600,n,Echo()
```

Mobile networking tuning

Because mobile networks have very specific (and dirty) characteristics there are certain telecommunication related areas that require specific fine tuning and must be known.

Stay narrowband

Mobile networking require to reduce as much as possible the bandwidth used.
That means that AMR must be configured in order to use only AMR 4.75kbit (the lower bitrate) when Asterisk send data to the mobile clients.

Edit file /etc/Asterisk/codecs.conf

1. nano /etc/Asterisk/codecs.conf

Add the following at the end of the file

2. [amr]
3. octet-aligned=1
4. ;codec_amr: Must be one of MR475, MR515, MR59, MR67, MR74, MR795, MR102, MR122, MRDTX
5. **mode = MR475**
6. dtx=0
7. cng = 1
8. vad = 0

Restart Asterisk or reload the codec amr .
Please note that VAD is not properly implemented in Asterisk AMR patch set and so it **must not be used**.

Stay relaxed with timers

Mobile networks can have very high latency such as 1400ms round-trip-time and may have high packet loss creating bad TCP socket network congestion, so it's required to relax the several timer of Asterisk in order to work properly with mobile networks.
To say relaxed with timers in Asterisk we can do the following.

SIP Timers

SIP timers related to the amount of time that an INVITE can take to complete a session need to be adjusted as a call setup could even require, in very bad network condition, 30 seconds.

Asterisk handle the configuration in a dynamic way with measured round trip time when there is qualify=yes, but instead use the static parameters when there qualify=no is setup.

To adjust it:

edit sip.conf in [general] sections :

```
;t1min=100                      ; Minimum roundtrip time for messages to monitored hosts
                                ; Defaults to 100 ms
Set timert1 to 3000ms
timert1=3000                    ; Default T1 timer
                                ; Defaults to 500 ms or the measured round-trip
                                ; time to a peer (qualify=yes).
;timerb=32000                   ; Call setup timer. If a provisional response is not received
                                ; in this amount of time, the call will autocongest
                                ; Defaults to 64*timert1
```

So this means that **only Snom** need to have **qualify=yes** in sip account configuration for measured round-trip time between

Asterisk and the device and do not care about default configuration that is only related to PrivateGSM Enterprise mobile client.
PrivateGSM Enterprise need to have **qualify=no** as shown below in example SIP account, otherwise the battery life of the mobile phone will be seriously impacted because of frequent ping-pong of SIP OPTIONS packets to check RTT.

Disable SIP Session Timers

SIP Session Timers are used during a call to check that the other party it's alive.
Due to specific behaviour of mobile networks, such as packet loss and consequent TCP network congestion, the SIP session timer may cause a mobile call to be hangup automatically by the server even if it still up.

On Asterisk those must be disabled as follow:

edit sip.conf in [general] section:
session-timers=refuse

Reduce RTP timeout

By default Asterisk use an rtp timeout of 60s, so that if it does not see RTP ongoing during a call it just bring down the call.
The timeout it's quite high and PrivateGSM already have it setup to 15 seconds by default.

The parameter must be adjusted to 15 seconds as follow:

edit sip.conf in [general] section:
rtptimeout=15

Audio buffer tuning

Mobile networks can have very high variable latency and that means that the default configuration of jitter buffer does not work fine and can cause very strong audio chopping.

With this extra tuning the voice quality should be much more better while making calls between PrivateGSM Enterprise and Snom phones.

Edit sip.conf and put the following value in [general] section:
jbenable = yes
jbforce = no
jbmaxsize = 200
jbresyncthreshold = 1200
jbimpl = fixed

In theory it's also possible to use adaptive jitterbuffer but this is not optimized for burst based networking that characterize mobile networking. So a static jitter buffer must be used.
If anyone is able to make the adaptive jitter buffer works fine also in bad network conditions with variable latency that during the call move from 400ms up to 1400ms, let us to know.

Use a bit of data for mobile always-on

On mobile devices you need to save as much bandwidth as possible in order to strongly save battery life for always on management.
That means using a set of specific tuning reported below.

Disable qualify

Asterisk, when **qualify=yes** is present, does a periodic check of round-trip-time between the PBX and the SIP client.
On PrivateGSM **qualify must be disabled** in order to avoid battery drain due to the periodic ping-pong in the always on connection.

Disable TCP keepalive

Most modern operating system support TCP keepalive at Layer3 of ISO/OSI pile.

The TCP keepalive of Linux must be disabled because the TCP keepalive to keep the TLS connection is done at SIP protocol level by PrivateGSM Enterprise that every 3 minutes send a \r\n to the SIP/TLS server (56byte of data).

On Linux you can disable TCP keepalive with the following command:

- **echo 0 > /proc/sys/net/ipv4/tcp_keepalive_probes**

Choose right data plan

In order to use mobile VoIP networking efficiently you must consider
Think that on mobile networks with:

- AMR 4.75kbit you consume about 100Kbyte/s per minute
- AMR 12.2kbit you consume about 200Kbyte/s per minute

Additionally PrivateGSM require an always-on connection that consume 1-2Mb/months to keep connection to PBX always on.

Best data plans are:

- Flat
- Traffic based

You must not use a time based mobile data plan.

Example SIP account

In this section you will find example extension (VoIP account) with the specific configuration parameters required to configure a PrivateGSM Enterprise and a Snom phone.
The extension has to be put in sip.conf or in realtime sql configuration.

PrivateGSM SIP account

Below is reported an example for SIP account for a PrivateGSM Enterprise phone.
Here we add only the required field that must be used for Private GMS Enterprise edition.
disallow=all
allow=amr:100
autoframing=yes
encryption=yes
transport=tls
port=5061
qualify=no

Snom SIP account

Below is reported an example for SIP account for a Snom phone:
disallow=all
allow=gsm
autoframing=yes
encryption=yes
transport=tls
port=5061
qualify=yes

Configuring Snom 300

In this section you will be guided trough the relevant configurations of a Snom 300 phone to be connected securely, with all proper TLS checks, to Asterisk 1.8 with GSM codec using SIP/TLS and SRTP 32bit with SDES key exchange.

Firmware update

Snom phones support advanced SRTP and TLS related configuration only from Firmware 8.
This configuration set has been done by using a Snom 300 with Firmware 8.4.18
Below a procedure to update the phone

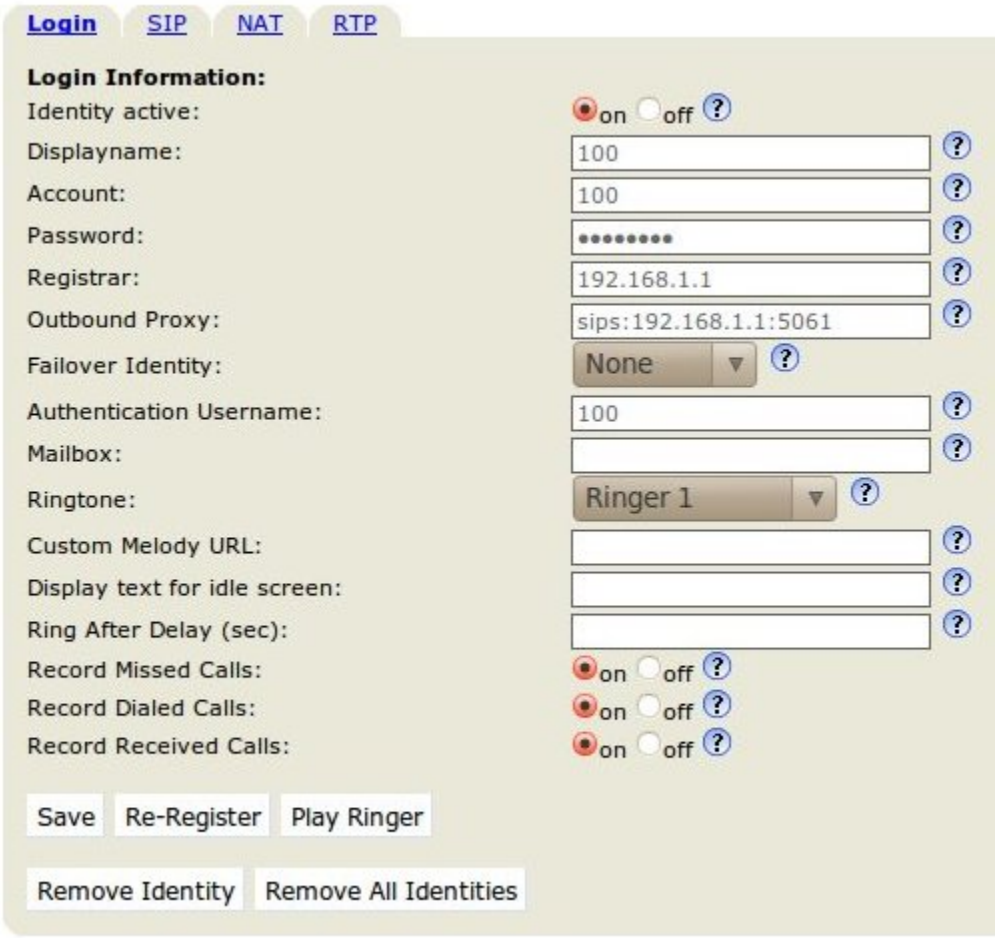
a. Open the Web User Interface of the Snom and navigate to the *Software Update* page.
b. Copy and paste this URL:
http://provisioning.Snom.com/download/fw/Snom300-8.4.18-SIP-f.bin
into the *Firmware* field and press

load
c. The phone reboots and may ask you to perform the update, click 'Yes'.

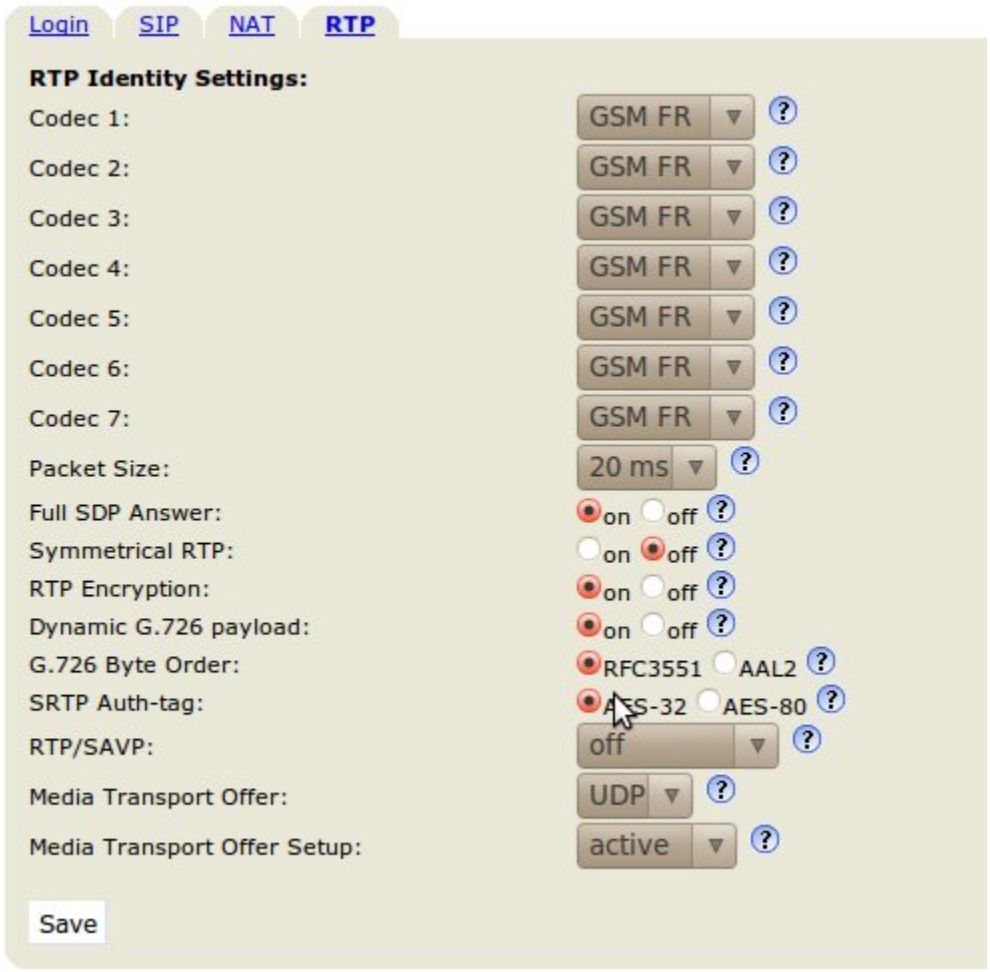
Note: Do not disconnect the power at anytime!
After that, the phone is upgraded to version 8.4.18.

Configuration

a. Open the Web User Interface of the Snom and navigate to the *Setup/Identity1* page, *login* tab.
b. Set *Account*, *Authentication Username* and *password* with the correct data that you have.
c. Set *Registrar* with the IP Address of the Asterisk Server.
d. Set *Outbound Proxy* in this form : **'*sips*:*ip_of_the_srv*:*5061*'** (ip_of_the_srv is the Asterisk Server)

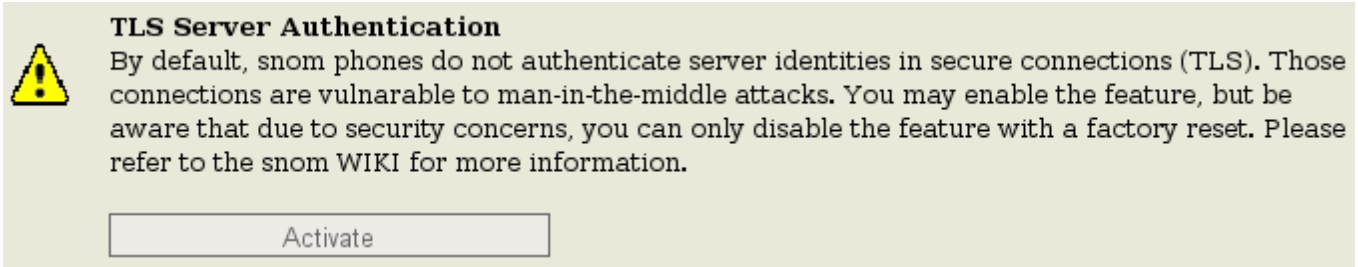


e. Goto in the *Setup/Identity1* page, *SIP* tab
f. Set *Support Broken registrar* to ON (click on this check two times for activate).
g. Goto in the *Setup/Identity1* page, *RTP* tab
h. Set *RTP Encryption* to ON.
i. Set *SRTP Auth-tag* to *AES-32.*
j. Set *Packet Size* to 20ms
k. Set *Media Transport Offer* to UDP



Security Configuration

Snom phones do not verify server identities by default.
From firmware version 8.2.30 you can explicitly state to verify server certificates though. You can activate the feature on the *certificates page* of the web interface:



The phone will reject all secure connections of peers offering an unknown certificate that could not be verified by one of build-in CA's of the Snom phone. Please refer to the *Certificate Authorities* tab to see which authorities are supported by the phone. Due to security concerns, you can only disable the feauture by resetting the phone to the factory defaults.

A certificate is trusted if its signature is signed by a certificate authority. Snom has pre-installed CA's which are listed on the *Certificate Authorities* tab of the *Certificates* page.

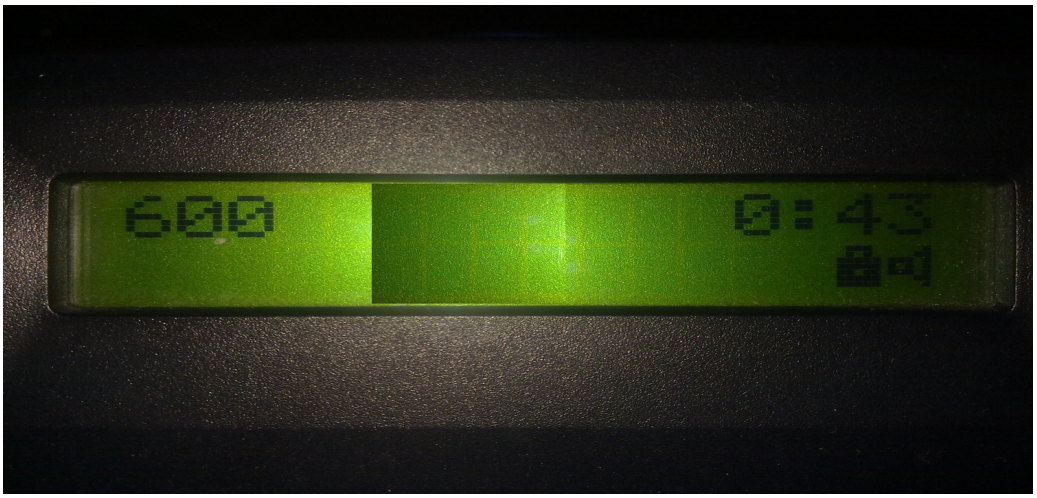
All rejected certificates are listed in the *Unknown Certificates* tab. If you persist on trusting the identification you can add it as an exception. Another way to trusting the identification is to import your TLS certificate (X.509 DER format) through the *Unknown Certificates* page.
After that, the imported certificate is listed in the *Server Certificates* tab and the registration to the proxy server works.

Please refer to the log and assure your certificate is in DER format and is either signed by one of phone's authorities or server certificates.

Make secure call

To call with Snom it's a matter of pulling the secure configured line and dial the extension required.
It's highly relevant to show that the line is secure by looking at the Snom phone display, to be sure that there is a lock on it:

Screenshot with a Snom 300 Secure calls to echo test :



Configuring PrivateGSM Enterprise

PrivateGSM Enterprise support only SIP/TLS and SRTP with SDES key exchange without support for standard unsecured VoIP.

Check compatibility

You need to check the compatibility of your mobile devices with PrivateGSM Enterprise as most iPhone, Nokia and Blackberry phones are supported but not all of them.
If a device is not supported it means that it has serious firmware bugs and/or some specific platform limitation.

- iPhone
- Nokia
- Blackberry

Configuration

In order to configure PrivateGSM Enterprise , here below you will find the steps to follow.

1. Launch PrivateGSM
2. Click on ‘Settings’ -> ‘Advanced Settings’ -> ‘SIP Account settings’

- **SIP Security**: TLS
  SIP connection is done ove
- **TLS Security**

Policy: Secure (default)

If an invalid certificate is prompted by the server and the client does not recognize it the connection will be dropped and the user will be just informed with a "Security Warning: Invalid Sever certificate, check certificate settings" .

- . **Security Protocol** : SRTP
  Enable SRTP security usage for end-to-site encryption (AES128 encryption with SDES key exchange over SIP/TLS channel)
- **SIP Server** – SIP server hostname or IP address
  NOTE: If using SIP/TLS it MUST match server digital certificate common name (CN).
- **SIP Server port** – SIP server port to be used for connections
  NOTE: Some mobile operators filter out SIP 5061/TCP and 5060/UDP standard port.
- **SIP username** – SIP server username with numeric input (only numbers)
- **SIP password** – SIP server username's password
- **Use proxy** – Enable/Disable use of proxy server
- **SIP Proxy Security**: TLS
- **SIP Proxy Server** - SIP proxy server hostname or IP address
  NOTE: If using SIP/TLS, the proxy server hostname MUST match server digital certificate common name (CN).
- **SIP Proxy server port** – SIP Proxy server port to be used for connections

Security Configuration

PrivateGSM Enterprise has by default strict TLS digital certificate checking, that means that untrusted digital certificates are refused for security reason.
In order to let it connect to the SIP/TLS server, depending on your digital certificate setup:

- Use recognized digital certificates (buy a digital certificate from an already trusted CA)
- Install your digital certificate CA root on the mobile phone

The procedure to load the digital certificates on Nokia, iPhone and Blackberry are differents, in this howto the procedure is specific and releant to Nokia S60 phones.

The certificate to be imported has to be in **DER** format.

Import certificate on Nokia S60

The import of digital certificate into Nokia S60 phones can be slightly different depending on the Nokia version being them:

- Nokia S60 MR
- Nokia S60 FP1
- Nokia S60 FP2
- Nokia S60 5th edition

The procedure should go as follow:

1. Send the digital certificate file via bluetooth in DER format and with .der extension
2. Import the digital certificate file

Now check that in the Certificate Settings of the phone it has been correctly imported, by looking at Tools -> Settings -> General -> Security -> Certificate Management -> Trusted site certificates ->

Please note that, only to make a test, you can setup the TLS security policy to **Insecure** in PrivateGSM -> Settings -> Advanced Settings -> SIP Settings -> TLS Security Policy:

By using this setting PrivateGSM **WILL NOT CHECK THE SECURITY** and this configuration is not secure nor supported by the PrivateGSM's manufacturer.
Still if you need to make a quick test without importing digital certificate (or using already recognized digital certificates) you can try using TLS Security Policy as Insecure.

Please stay with **TLS Security Policy: Secure** .

Import certificate on Blackberry RIMOS

To be done, in the next release of this howto.

Import certificate on iPhone iOS

To be done, in the next release of this howto.

Make secure call

To make secure calls with PrivateGSM it's not required to dial from within the software application.
The user can just use secure prefix +801 to be put in front of the number he want to call trough secure VoI.
In the example below the user is calling the number **101** by dialing +801**101**

After a call will be established you will see a connection status on mobile phones as follow:

Notes on AMR licensing

If you do only landline secure telephony networks you can use royalty free GSM 12kbit codec.
If you need to do mobile secure telephony networks with optimal performance you need to use AMR 4.75kbit narrowband codec
AMR codec, like most mobile related technologies, require licensing because it's patent-protected and the company licensing the codec trough the AMR Patent Pool is VoiceAge.
In order to stay legal, some options is:

- you can acquire a license from VoiceAge
- use a platform suite like Enterprise VoIP Security Suite (EVSS) with SIP/TLS+SRTP+AMR
- use a SBC like AcmePacket that support SIP/TLS+SRTP+AMR
- Use Cisco PBX that support AMR trough a separate license

Authors

The authors of this how-to are reported below :

Andrea Cristofanini
Fabio Pietrosanti
Gianluca Varisco
Luca Nava

Changes

Further guide on how to integrate PrivateGSM Enterprise and Snom phones with other PBX platforms will come (such as FreeSWITCH and several VoIP Firewalls).

For any change requests / update / mistake / improvements relevant to this howto you can reach the authors to the address support@privatewave.com