Yealink | 亿联网络

Configuration Encryption Tool User Guide

Mar. 2020

About This Guide

Configuration files contain sensitive information such as user accounts, login passwords, registration or information. To protect sensitive information from being tampered, you must encrypt configuration files. Yealink provides tools for encrypting configuration files on the Windows platform and Linux platform respectively.

You can use the encryption tool to encrypt the following three types of configuration files:

- MAC-Oriented CFG file: <MAC>.cfg
- Common CFG file: y0000000000xx.cfg
- Other custom CFG file: sip.cfg, account.cfg and so on.

You can ask the distributor or the Yealink Field Application Engineer for the encryption tool, or you can download them online: http://support.yealink.com/documentFront/forwardToDocumentFrontDisplayPage.

This guide provides detailed information on how to encrypt configuration files using Yealink-supplied encryption tools, and how to deploy Yealink IP phones using these encrypted configuration files.

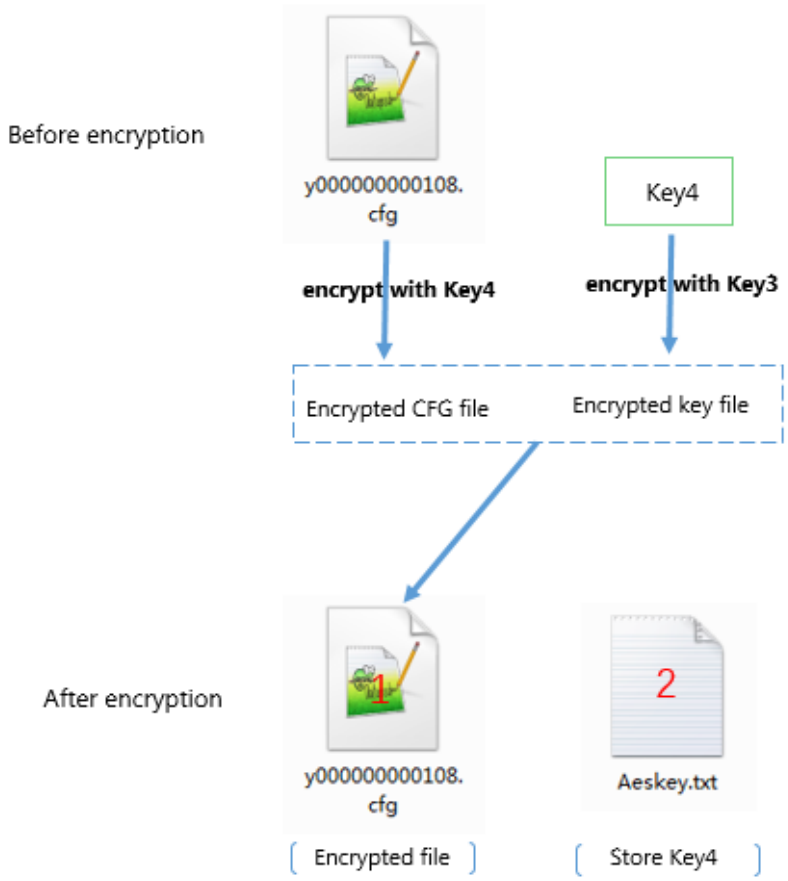
Encryption Mode Introduction

The encryption tool supports two encryption modes: RSA Mode and Compatibility Mode.

RSA Mode

The encryption tool encrypts plaintext configuration files (one by one or in batch) using 16-character/32-character symmetric keys (the same or different keys for configuration files). This tool also encrypts the plaintext 16-character/32-character symmetric keys using a fixed RSA public key. Then two encrypted files (encrypted configuration and key file) are generated into one configuration file.

This tool generates another new file named Aeskey.txt storing the plaintext 16-character/32-character symmetric keys for each configuration file.

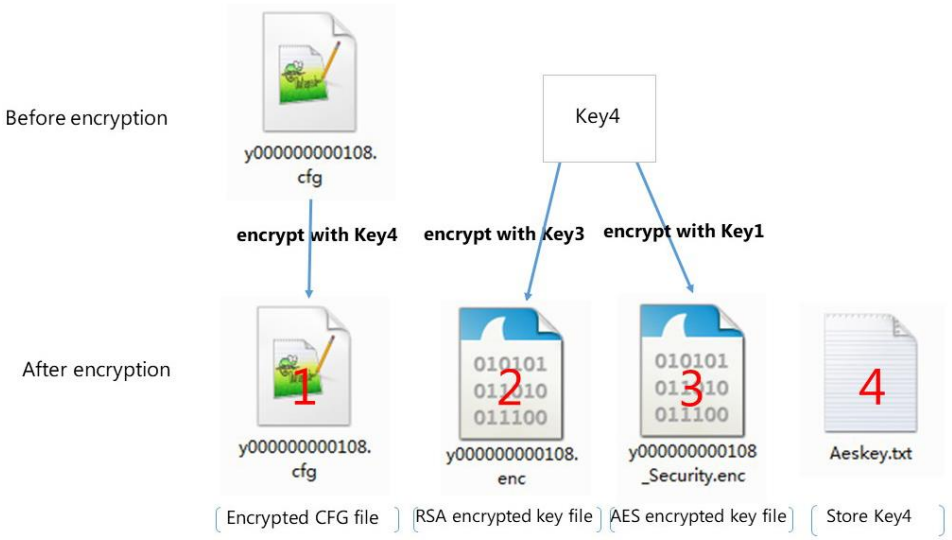


*Key3: RSA public key (built in tool by default)
*Key4: Manually set or automatically generated AES key

Compatibility Mode

The encryption tool encrypts plaintext configuration files (one by one or in batch) using 16-character symmetric keys (the same or different keys for configuration files) and generates encrypted configuration files with the same file name as before.

This tool also encrypts the plaintext 16-character symmetric keys using a fixed key, which is the same as the one built in the IP phone and generates new files named as <xx_Security>.enc (xx indicates the name of the configuration file, for example, y000000000108_Security.enc for y000000000108.cfg file). This tool generates another new file named Aeskey.txt storing the plaintext 16-character symmetric keys for each configuration file.



*Key1: Phone built-in AES key
*Key3: RSA public key (built in tool by default)
*Key4: Manually set or automatically generated AES key

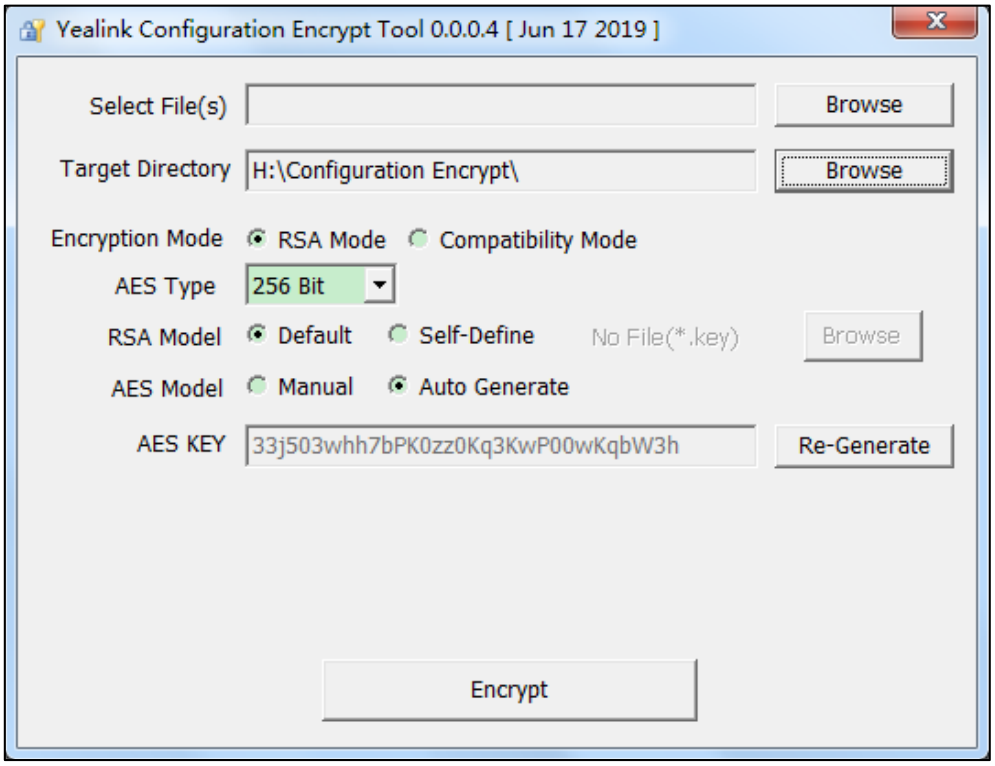
Encrypting Configuration Files on Windows Platform

Encrypting Configuration Files via Graphical Tool

To encrypt configuration files:

1. Double click “Config_Encrypt_Tool.exe” to start the application tool.

The screenshot of the main page is shown below:



When you start the application tool, a file folder named “Encrypted” is created automatically in the directory where the application tool is located.

2. Click **Browse** to locate configuration file(s) (for example, y000000000108.cfg) in your local system in the **Select File(s)** field.

   To select multiple configuration files, you can select the first file and then press and hold the Ctrl key to select the next files.

3. (Optional.) Click **Browse** to locate a target directory from your local system in the **Target Directory** field.

   The tool uses the file folder “Encrypted” as the target directory by default.

4. Mark the desired radio box in the **Encryption Mode** field.

   - **RSA Mode**: It is applicable to the environment where all phones are running firmware version V85 or later. AES keys can be 16 characters or 32 characters.
   - **Compatibility Mode**: It is applicable to the environment where not all phones are running firmware version V85 or later. AES keys must be 16 characters.

5. If you select **RSA Mode**, select the desired AES type from the **AES Type** drop-down menu.

6. (Optional.) Mark the desired radio box in the **RSA Model** field.

   - **Default**: The tool uses the built-in RSA public key (pub.key) to encrypt files.

     The key is located in the directory where the encryption tool is stored.

   - **Self-Define**: Click **Browse** to locate a target RSA public key file from your local system. The tool uses the self-define RSA public key to encrypt files.

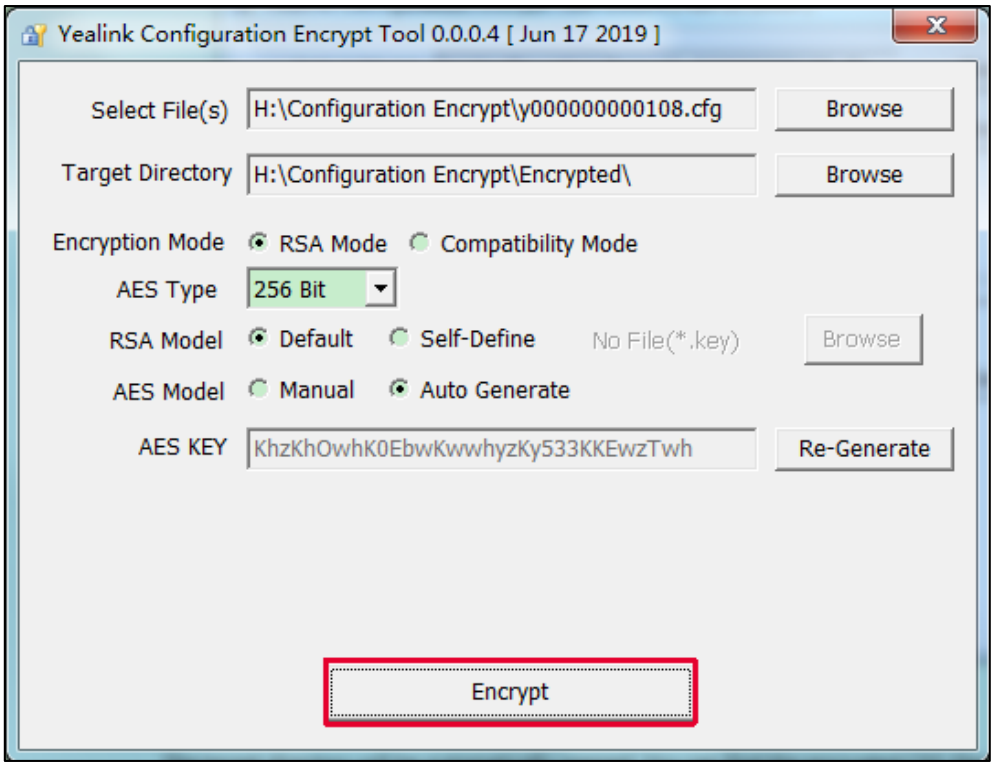
     When you import the new self-define public key, the tool will automatically overwrite the old

self-define public key. **Note that if you use the self-define public key to encrypt files, you need to import the matching self-define RSA private key on the phone.**

7. (Optional.) Mark the desired radio box in the **AES Model** field.
   - **Manual**: Enter an AES key in the **AES KEY** field or click **Re-Generate** to generate an AES key in the **AES KEY** field. The configuration file(s) will be encrypted using the AES key in the **AES KEY** field.
   - **Auto Generate**: The configuration file(s) will be encrypted using a random AES key. The AES keys of different configuration files are different.

Note The supported characters contain: 0 ~ 9, A ~ Z, a ~ z.

8. Click **Encrypt** to encrypt the configuration file(s).



9. Click **OK**.

   The target directory will be automatically opened. You can find the encrypted file(s) and an Aeskey.txt file storing plaintext AES key(s).

Encrypting Configuration Files via DOS Command Line

Command Line Instructions

Command line format:

"YealinkEncrypt CMD.exe" -f file1.cfg [file2.cfg ...] [-p DESTPATH(Default as 'Encrypted')] [-k AESKEY(Default as random)] [-rsa pubfile] [–cmpt] [-aes128]

Command	Description
-f file.cfg	Used to specify the configuration file to be encrypted. If it is not entered, encryption cannot be performed.
-k AESKEY	Used to specify the encryption key. If it is not entered, the random key will be used for encryption.
-p destdir	Used to specify the storage path of the generated files. If it is not specified, the default is the **Encrypted** directory of the current directory.
-m	Identifies that the encryption is a batch encryption. Each file is encrypted using the randomly specified AESKEY.
-cmpt	Generates a secret key using compatibility mode (three files xxx.cfg, xxx.enc, and xxx_Security.enc are generated). If it is not specified, the command line will generate encrypted files in a completely new format.
-rsa pubfile	Indicates the public key file used by the command line. If it is not set, encryption cannot be performed.
-aes128	Indicates that the command line will encrypt the CFG file with the key of aes128.
-aes256	Indicates that the command line will encrypt the CFG file with the key of aes256.

Note The default length of the command line AES key is 256 bits. If you specify -aes128 / 256 and the entered -k key length does not match the prefix, the command line will report an error.

Encrypting Configuration Files

Procedure

1. Place the encryption tool “YealinkEncrypt CMD.exe” and configuration files in the same directory.
2. Open a Command prompt.
3. Locate the directory where the encryption tool is stored.
4. Execute one of the following commands according to your requirements:

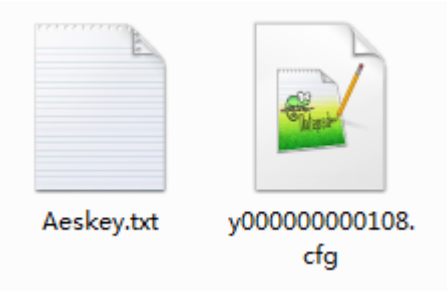
Example 1: RSA Encryption Mode

"YealinkEncrypt CMD.exe" -f y000000000108.cfg -k 0123456789abcdef -rsa pub.key -aes128

Write file to Encrypted\\Aeskey.txt!

[pub.key][0123456789abcdef][y000000000108.cfg][Encrypted\].

Result: You can find the encrypted y000000000108.cfg file and an Aeskey.txt file storing the plaintext AES key 0123456789abcdef in the specified directory. The encrypted y000000000108.cfg file contains the encrypted key ciphertext.



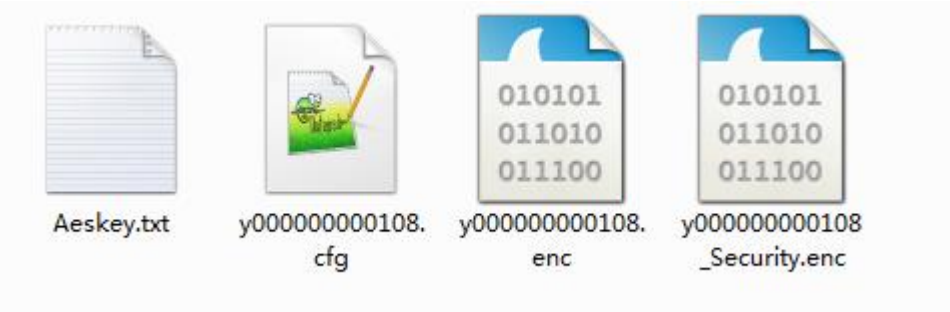
Example 2: Compatibility Encryption Mode

"YealinkEncrypt CMD.exe" -f y000000000108.cfg -k 0123456789abcdef -rsa pub.key -cmpt

Write file to Encrypted\\Aeskey.txt!

[pub.key][0123456789abcdef][y000000000108.cfg][Encrypted\].

Result: You can find the encrypted y000000000108.cfg file, y000000000108.enc, y000000000108_Security.enc, and an Aeskey.txt file storing the plaintext AES key 0123456789abcdef in the specified directory. The encrypted y000000000108.enc and y000000000108_Security.enc files contain the encrypted key ciphertext.



Example 3: Encrypt multiple configuration files in the current directory.

"YealinkEncrypt CMD.exe" -f y000000000108.cfg y000000000058.cfg -rsa pub.key –m

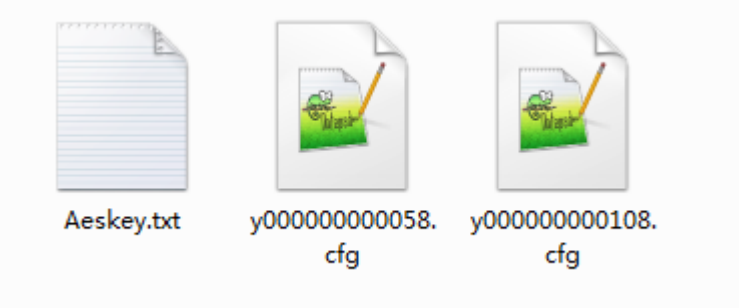
Write file to Encrypted\\Aeskey.txt!

[pub.key][3hhK0l5Kzl5u30j3wEhuwu3whzhWzpwz][y000000000108.cfg][Encrypted\].

Write file to Encrypted\\Aeskey.txt!

[pub.key][3hhK0l5Kzl5u30j3wEhuwu3whzhWzpwz][y000000000058.cfg][Encrypted\].

Result: You can find the encrypted y000000000108.cfg file, y000000000058.cfg, and an Aeskey.txt file storing the plaintext AES key 3hhK0l5Kzl5u30j3wEhuwu3whzhWzpwz in the specified directory. The encrypted y000000000108.cfg file and y000000000058.cfg contain the encrypted key ciphertext.



Note The supported characters contain: 0 ~ 9, A ~ Z, a ~ z.

Encrypting Configuration Files on Linux Platform

Command Line Instructions

Command line format:

"YealinkEncrypt CMD.exe" -f file1.cfg [file2.cfg ...] [-p DESTPATH(Default as 'Encrypted')] [-k AESKEY(Default as random)] [-rsa pubfile] [-aes128]

Command	Description
-f file.cfg	Used to specify the configuration file to be encrypted. If it is not entered, encryption cannot be performed.
-k AESKEY	Used to specify the encryption key. If it is not entered, the random key will be used for encryption.
-p destdir	Used to specify the storage path of the generated files. If it is not specified, the default is the **Encrypted** directory of the current directory.
-m	Identifies that the encryption is a batch encryption. Each file is encrypted using the randomly specified AESKEY.
-cmpt	Generates a secret key using compatibility mode (three files xxx.cfg, xxx.enc, and xxx_Security.enc are generated). If it is not specified, the command line will generate encrypted files in a completely new format.
-rsa pubfile	Indicates the public key file used by the command line. If it is not set, encryption cannot be performed.
-aes128	Indicates that the command line will encrypt the CFG file with the key of aes128.
-aes256	Indicates that the command line will encrypt the CFG file with the key of aes256.

Note The default length of the command line AES key is 256 bits. If you specify -aes128 / 256 and the entered -k key length does not match the prefix, the command line will report an error.

Encrypting Configuration Files

Procedure

1. Place the encryption tool “yealinkencrypt” and configuration files in the same directory.
2. Open a terminal window.
3. Execute the **cd** command to locate the directory where the encryption tool is stored.

   For example, execute **cd /tmp** to locate the **tmp** directory.

5. Execute one of the following commands according to your requirements:

   Example 1: RSA Encryption Mode

   [root@localhost tmp]#./yealinkencrypt -f y000000000108.cfg -k 0123456789abcdef -rsa pub.key -aes128

   Write file to Encrypted\\Aeskey.txt!

   [pub.key][0123456789abcdef][y000000000108.cfg][Encrypted\].

   Result: You can find the encrypted y000000000108.cfg file and an Aeskey.txt file storing the plaintext AES key 0123456789abcdef in the specified directory. The encrypted y000000000108.cfg file contains the encrypted key ciphertext.

   Example 2: Compatibility Encryption Mode

   [root@localhost tmp]#./yealinkencrypt -f y000000000108.cfg -k 0123456789abcdef -rsa pub.key -cmpt

   Write file to Encrypted\\Aeskey.txt!

   [pub.key][0123456789abcdef][y000000000108.cfg][Encrypted\].

   Result: You can find the encrypted y000000000108.cfg file, y000000000108.enc, y000000000108_Security.enc, and an Aeskey.txt file storing the plaintext AES key 0123456789abcdef in the specified directory. The encrypted y000000000108.enc and y000000000108_Security.enc files contain the encrypted key ciphertext.

   Example 3: Encrypt multiple configuration files in the current directory.

   [root@localhost tmp]#./yealinkencrypt -f y000000000108.cfg y000000000058.cfg -rsa pub.key –m

   Write file to Encrypted\\Aeskey.txt!

   [pub.key][3hhK0l5Kzl5u30j3wEhuwu3whzhWzpwz][y000000000108.cfg][Encrypted\].

   Write file to Encrypted\\Aeskey.txt!

   [pub.key][3hhK0l5Kzl5u30j3wEhuwu3whzhWzpwz][y000000000058.cfg][Encrypted\].

   Result: You can find the encrypted y000000000108.cfg file, y000000000058.cfg, and an Aeskey.txt file storing the plaintext AES key 3hhK0l5Kzl5u30j3wEhuwu3whzhWzpwz in the specified directory. The encrypted y000000000108.cfg file and y000000000058.cfg contain the encrypted key ciphertext.

Note The supported characters contain: 0 ~ 9, A ~ Z, a ~ z.

Configuring Yealink IP Phones

Before deploying IP phones using the encrypted configuration files, you need to configure the following parameters for the IP phones using the configuration files first.

1. Add/Edit the following parameters in the configuration file (e.g., sip.cfg).

Parameters	Permitted Values	Default
static.auto_provision.aes_key_in_file	**0 or 1**	**0**
Description: It configures how the phone decrypts files. **0**-The phone will decrypt the encrypted configuration files using plaintext AES keys configured on the phone. **1**-The phone automatically determines which encryption mode is performed by the tool (RSA mode or compatible mode). Refer to Phone Decryption for more information. **Note: This guide mainly introduces the usage after the parameter is set to 1.**		
static.auto_provision.rsa_pri_key.enable	**0 or 1**	**0**
Description: It enables or disables the self-define RSA private key. **0**-Disabled, the phone decrypts the encrypted configuration files using phone built-in RSA keys. **1**-Enabled, the phone decrypts the encrypted configuration files using a self-define RSA private key. **Note**: If you select to use a self-define RSA public key to encrypt files in the tool, you need to set this parameter to 1. **Web User Interface:** Settings > Auto Provision > Self-Define RSA Pri Key		
static.auto_provision.rsa_pri_key.url	**URL within 511 characters**	**Blank**
Description: It configures the URL to import the self-define RSA private key file. **Note**: The key file must be in *.key format. It works only if "static.auto_provision.rsa_pri_key.enable" is set to 1 (Enabled). **Web User Interface:** Settings > Auto Provision > Import RSA Pri Key		
static.auto_provision.rsa_pri_key.delete	**http://localhost/all**	**Blank**
Description: It deletes the self-define RSA private key file.		

2. Reference the configuration file in the boot file (e.g., y000000000000.boot).

   Example:

   include:config “http://10.2.1.158/sip.cfg”

   include:config “http://10.2.1.158/account.cfg”

3. Upload the boot file and configuration file to the root directory of the provisioning server.
4. Trigger IP phones to perform an auto provisioning for a configuration update.

For more information on auto provisioning, refer to the latest Auto Provisioning Guide on Yealink Technical Support.

Deploying Phones Using Encrypted Configuration Files and the RSA key

Scenario: Encrypt configuration files and ensure no plaintext configurations and keys are transmitted across the network

Scenario Conditions:

- The administrator wants to encrypt configuration files to protect sensitive information in configuration files from being tampered.
- SIP-T46U IP phone MAC: 001565918998.
- The following configurations have been set:

  static.auto_provision.aes_key_in_file = 1

  static.auto_provision.rsa_pri_key.enable = 1

  static.auto_provision.rsa_pri_key.url = https://192.168.1.100/pri.key
- Files encrypted by the encryption tool:

  y000000000108.cfg (encrypted) and sip.cfg (encrypted)

Scenario Operations:

1. Upload 001565918998.boot, y000000000108.cfg (encrypted), and sip.cfg (encrypted) files to the root directory of the provisioning server.
2. Edit the boot file (001565918998.boot file) as follows:

```
#!version:1.0.0.1
## The header above must appear as-is in the first line

include:config <sip.cfg>
include:config "y000000000108.cfg"

overwrite_mode = 1

specific_model.excluded_mode = 0
```

3. Reboot the phone to trigger an auto provisioning.

For more information on auto provisioning, refer to the latest Auto Provisioning Guide on Yealink Technical Support.

During auto provisioning, the phone will try to download 001565918998.boot file firstly and then download the configuration files referenced in the boot file in sequence from the provisioning server.

In this scenario, the phone requests to download the sip.cfg file first. Because the downloaded configuration file is encrypted, the phone decrypts the file into the plaintext key (e.g., key4) using the self-define RSA private key (pri.key). The phone then decrypts the configuration file sip.cfg using the key4. After decryption, the phone resolves configuration files and updates configuration settings onto the phone system.

The way the phones process the y000000000108.cfg file is the same as that of the sip.cfg file.

Note You can also set "static.auto_provision.rsa_pri_key.enable = 0" to use the built-in RSA private key to decrypt the files.

Deploying Phones Using Encrypted Configuration Files and the AES key

Scenario: Encrypt configuration files and ensure no plaintext configurations and keys are transmitted across the network

Scenario Conditions:

- The administrator wants to encrypt configuration files to protect sensitive information in configuration files from being tampered.
- SIP-T46U IP phone MAC: 001565918998.
- The following configuration has been set:

  static.auto_provision.aes_key_in_file = 1
- Files encrypted by the encryption tool:

  y000000000108_Security.enc, sip_Security.enc, y000000000108.cfg (encrypted) and sip.cfg (encrypted)

Scenario Operations:

1. Upload 001565918998.boot, y000000000108_Security.enc, sip_Security.enc, y000000000108.cfg (encrypted) and sip.cfg (encrypted) files to the root directory of the provisioning server.
2. Edit the boot file (001565918998.boot file) as follows:

```
#!version:1.0.0.1
## The header above must appear as-is in the first line

include:config <sip.cfg>
include:config "y000000000108.cfg"

overwrite_mode = 1

specific_model.excluded_mode = 0
```

3. Reboot the phone to trigger an auto provisioning.

   For more information on auto provisioning, refer to the latest Auto Provisioning Guide on Yealink Technical Support.

During auto provisioning, the phone will try to download 001565918998.boot file firstly and then download the configuration files referenced in the boot file in sequence from the provisioning server.

In this scenario, the phone requests to download the sip.cfg file first. Because the downloaded configuration file is encrypted, the phone requests the sip.enc file first but finds none. Then the phone requests to download sip_Security.enc file and decrypts it into the plaintext key (e.g., key2) using the built-in AES key (e.g., key1). The phone then decrypts the configuration file sip.cfg using the key2. After decryption, the phone resolves configuration files and updates configuration settings onto the phone system.

The way the phones process the y000000000108.cfg file is the same as that of the sip.cfg file.

Phone Decryption

When “static.auto_provision.aes_key_in_file =1”, the phone will determine whether the file is an encrypted file generated in RSA mode.

If the encrypted file is generated in RSA mode, the phone directly decompresses the encrypted configuration file and obtains two encrypted files (configuration file and key file).

① Using built-in/self-define RSA private key decode Key4

② Using Key4 decode config file

- If “static.auto_provision.rsa_pri_key.enable = 0”, the phone decrypts the key file into the plaintext key (e.g., key4) using the built-in RSA private key. The phone then decrypts the configuration file using the key4.
- If “static.auto_provision.rsa_pri_key.enable = 1”, the phone decrypts the key file into the plaintext key (e.g., key4) using the self-define RSA private key. The phone then decrypts the configuration file using the key4.

If the encrypted file is not generated in RSA mode, the phone will request to download the <xx>.enc file first and decrypt this file using RSA private key. When there is no <xx>.enc file, the phone will request to download the <xx_Security>.enc file and decrypt this file into the plaintext key (e.g., key4) using the phone built-in AES key (e.g., key1). The IP phone then decrypts the encrypted configuration file using the corresponding key (e.g., key4).

① Using built-in AES key decode Key4

② Using Key4 decode config file

After decryption, the phone resolves configuration files and updates configuration settings onto the phone system.