# Synology Notification Service White Paper

Based on DSM 6.0.3

Synology®

# Table of Contents


Overview 3

Terms 3

Diffie–Hellman Key Exchange 4

Encryption and Decryption Flow 4

- Server encryption
- Client decryption

Revocation Mechanism 5

- Browsers
- Mobile devices
- Server

# Overview

This white paper explains the encryption mechanism when Synology services are transmitting push notifications to users' mobile devices via third-party services, such as Synology Notification Service (SNS), Apple Push Notification Service (APNs), and Google Cloud Messaging (GCM). Encryption is enforced during data delivery to ensure the confidentiality of messages.

This encryption mechanism adopts the following techniques:

- Diffie–Hellman key exchange (D–H) and symmetric key algorithm are applied to this mechanism.
- Different keys are utilized by different users, devices, and packages.
- Plain messages are not allowed when this mechanism is activated.

Details regarding the encryption–decryption procedure is provided in the following sections.

# Terms

Explanations of the relevant terms mentioned in this document are offered as follows:

- **APNs** is the acronym for Apple Push Notification Service, a notification service hosted by Apple to push notifications to iOS devices.
- **Clients** refer to mobile applications (Android and iOS), browser extensions (Chrome and Safari), and web browser push notifications (Chrome and Firefox).
- **GCM** is the acronym for Google Cloud Messaging, a notification service hosted by Google to push notifications to Android devices.
- **Server** refers to Synology Application Service (SAS) installed on DSM.
- **SNS** is the acronym for Synology Notification Service, a public cloud service hosted by Synology to push messages to APNS and GCM.

# Diffie–Hellman Key Exchange

Key exchange is the first step of encryption and the procedure is initiated from clients to the server.[1] Clients use **crypto_kx_* functions** to initiate key exchange to the server according to the following steps:

1. Clients generate **key pair**, **pk**, and **sk** using **crypto_kx_keypair()**.
2. Clients send the Exchange API request to the server.
3. Adopting the function **crypto_kx_client_session_keys**, clients utilize **public_key** and their own **key pairs** to generate **sharedRx** and **sharedTx**.
4. Clients use **base64 + JSON.parse** to decrypt **encrypted_data** and to obtain **nonce** and **ciphertext**.
5. Clients bring **sharedRx**, **nonce**, and **ciphertext** into the function **crypto_secretbox_open_easy** to obtain **secret_key**.

# Encryption and Decryption Flow

The push notification procedure can be divided into two parts, namely, server procedure and client procedure. The encryption–decryption procedure adopts the **crypto_box_easy_afternm** function and standardizes the results according to JSON encoding and Base64 encoding formats, enabling both the client and server sides to process encryption and decryption. Details regarding the procedure are provided in the following:

## Server encryption

**encrypted_data** is added to the original **event_contents** of the JSON object parameter to ensure the downward compatibility. The encryption procedure is shown below:

1. SNS will proceed the notification delivery in the original way if the encryption function is disabled.
2. **raw_data** is set as the assigned or default message, with the default message showing “You have encrypted messages”.
3. Only **raw_data** will be transmitted to SNS if the clients do not have secret keys.
4. Secret keys and notifications are brought in and **crypto_secretbox_easy** is executed to encrypt messages; subsequently, **nonce** and **ciphertext** are obtained. All the relevant data will be placed in **encrypted_data**.
5. Encrypted data will be transmitted to SNS after it is encoded by base64.

1. Prerequisite: the server’s public key must be known by all the clients.

### Client decryption

Client devices should be capable of processing push notifications before the message content is displayed. The decryption procedure is shown below:

1. The client device receives a push notification and retrieves **raw_data** as the notification message.
2. The client device will leave the decryption flow if the encryption function is disabled or if the device has no **secret_key**.
3. The client device uses base64 to decode **encrypted_data**; if failed, the device will leave the decryption flow.
4. The client device examines whether the function value is **crypto_secretbox_easy**; if failed, the device will leave the decryption flow.
5. The client device brings **nonce**, **ciphertext**, and **secret_key** into **crypto_secretbox_open_easy** to obtain a notification message.

## Revocation Mechanism

Disabling push notifications will cause the secret keys to be revoked from different client devices.

### Browsers

When a user unpairs with the server, the associated browsers will clear the secret key and send the unpair request to the server.

### Mobile devices

A client will send the unpair request to the server and clear the secret key when logging out.

### Server

The server will clear the correspondent secret key when the unpair request is received.