

Helioka - Telecommunication Protocol over the Internet

Paul Feuvraux

July 11, 2017

1 Introduction

This paper aims to explain Helioka, a simple protocol based on the RSA and AES algorithms only.

2 Motivations

For fun.

3 Terms

- **Key Agreement Key (KAK)** 4096-bit asymmetric RSA keys (PKA and PKB) generated at the client-side.
- **Private Key (PKA)**: Used to decrypt the TEK .
- **Public Key (PKB)**: Used to encrypt the TEK .
- **Temporary Encryption Key (TEK)**: 256-bit symmetric AES key generated at the client-side. Used to encrypt the communications.
- **Content Encryption Key (CEK)**: 256-bit symmetric AES key generated at the client-side. Used to encrypt KAK .
- **Key Encryption Key (KEK)**: 256-bit symmetric AES key derived from the user's Passphrase P .
- **Passphrase (P)**: User's defined alphanumeric UTF-8 passphrase during registration on the client side.
- **Key Derivation function**: Every derivation function is performed with PBKDF2. We denote this process as $KDF(x, s, i)$, where x is the passphrase, s the Salt, and i is the strengthen by factor and always equals 7000.
- **Symmetric Encryption function**: Every symmetric encryption is performed with AES under the Galois/Counter mode on 128-bit block cipher. We denote this process as $EncSym(k, x, i, a)$, where k is a symmetric AES key, x the content to be encrypted, i is the initialization vector (IV) and a is the authentication tag (AT).

- **Symmetric Decryption function:** Every symmetric decryption is performed with AES under the Galois/Counter mode on 128-bit block cipher. We denote this process as $DecSym(k, x, i, a)$, where k is a symmetric AES key, x the encrypted content to be decrypted, i is the IV and a is the AT.
- **Asymmetric Encryption function:** Every asymmetric encryption function is performed with RSA. We denote this process as $EncAsym(k, x)$, where k is a Public Key PKB and x is the content to be encrypted.
- **Asymmetric Decryption function:** Every asymmetric decryption is performed with RSA. We denote this process as $DecAsym(k, x)$, where k is a Private Key PKA and x is the encrypted content to be decrypted.

4 Protocol

4.1 Registration

While the user is typing his Passphrase P , the necessary keys (KEK and CEK) are generated.

4.1.1 Symmetric key encryption

A random Salt is generated on 128 bits. Once the Passphrase P defined by the user, the client proceeds to a key derivation such as $KEK = KDF(P, Salt, 7000)$. Once the KEK obtained through this key derivation step, a random IV and AT are both generated on 128 bits. Then, the client encrypts the CEK under the KEK such as $EncSym(KEK, CEK, IV, AT)$ and sends it to the server as a packet such as $packet = (CEK || IV || AT || Salt)$, where CEK is the encrypted CEK .

4.1.2 Asymmetric key encryption

The same CEK is used to encrypt PKA . The client sends PKB in plain text to the server while a random IV and AT are both generated on 128 bits. Once these parameters generated, the client encrypts PKA under CEK such as $EncSym(CEK, PKA, IV, AT)$, where CEK is the plain text CEK . Once encrypted, the client sends it as a packet $packet$ such as $packet = (PKA || IV || AT)$, where PKA is the encrypted PKA .

4.2 Connection

The user types his Passphrase P .

4.2.1 CEK decryption

The client gets the CEK and extract the Salt from it. Once the Salt extracted, the client proceeds to a Key Derivation such as $KEK = KDF(P, Salt, 7000)$. Once the KEK derived, the clients extract the IV and the AT from the CEK and decrypts the CEK such as $DecSym(KEK, CEK, IV, AT)$, where CEK is the encrypted CEK .

4.2.2 *PKA* decryption

The client gets the *KAK* and extract the *PKA* from it, which is a packet composed of the encrypted *PKA* and its cryptographic parameters. The client decrypts *PKA* under *CEK* such as $DecSym(CEK, PKA, IV, AT)$.

4.3 *TEK* exchange

One of the two participants (users) of the communication generates a *TEK* and encrypts it under the other user's *PKB* such as $EncAsym(PKB, TEK)$. The other user decrypts the *TEK* such as $DecAsym(PKB, TEK)$, where *TEK* is encrypted.

4.4 Communication Encryption

Every "song" is turned into small packets. Every packet is encrypted under *TEK*. For every session is generated a random IV and AT on 128 bits. The clients encrypt the packets such as $EncSym(TEK, packet, IV, AT)$ and send it to each other.

4.5 User's Passphrase change

The user defines his new Passphrase *P*. A random Salt is generated on 128 bits, the client proceeds to a Key Derivation such as $KEK = KDF(P, Salt, 7000)$. Once the new *KEK* obtained from the Key Derivation function, a random IV and AT are both generated on 128 bits. Then, the clients encrypts the *CEK* under the new *KEK* derived from the new *P* such as $EncSym(KEK, CEK, IV, AT)$. Once the *CEK* encrypted, the client send the encrypted *CEK* to the server as a packet *packet* such as $packet = (CEK || IV || AT || Salt)$, where *CEK* is the encrypted *CEK*.