CRYPTONOTE PROTOCOL VERSION 1

January 31, 2021

Nigel Bess

Contents

1 Introduction

CryptoNote is an open-source encryption tool that allows users to securely read and write sensitive data.

Private information on storage devices (Hard drives, SSD's, flash drives, etc.) is vulnerable to anyone who gains physical access to those devices. The solution is to encrypt this information. There are many ways to encrypt sensitive data, but most of them have blatant vulnerabilities, which CryptoNote addresses.

1.1 Problems With Existing Solutions

1.1.1 File Encryption tools

One simple approach to encrypting data is to save it in an unencrypted format, such as a .txt file, and encrypt that file using a tool like 7Zip. This approach encrypts the data as a new file, but even if the original file is deleted, a data recovery software can find the original file and read it.

1.1.2 Encrypted Storage Devices

One can purchase an encrypted storage device, but this doesn't guarantee security either. Encrypted storage devices aren't inherently different from other storage devices, but use software to read and write in an encrypted format. Some programs dont't interface well with encryption software and therefore can't write directly to encrypted storage devices. For example a notepad application might not be able to save notes on an encrypted drive.
Encrypted drives also cost money, and don't provide value beyond that of an existing storage device and good encryption software.

1.1.3 Closed-Source Encryption Software

Many programs exist to solve the problems mentioned above, but most of them have one thing in common: they are not open-source. A user of one of those programs has no way of knowing what the program is actually doing with their data. The user must trust that the creator of the program is not a malicious actor and that the program actually works

as intended. Encryption exists to eliminate the need for trust, so closed-source programs invalidate the security provided by encryption.

2 How CryptoNote Works

CryptoNote accepts user input (sensitive data) and stores it in memory. When a user saves that data, is is encrypted in memory and saved to the storage device in an encrypted form. Sensitive data is never written to the storage device, only to memory. CryptoNote also erases data from memory whenever possible, but this does not provide security from malicious programs already installed on a computer. CryptoNote's purpose is to allow users to save encrypted data without opening vulnerabilities to data recovery software.

3 Encryption Algorithm

CryptoNote uses AES-256-CBC to encrypt and decrypt data. The 32 byte AES-256 key is derived from a user-provided password using PBKDF2 with a pseudorandom function of HMACSHA1.

1. **User Input**:
   The user provides a message they wish to encrypt, and a password to encrypt and decrypt the message.

2. **Key Generation**:
   A 32 byte AES-256 key is derived from the password using PBKDF2.

$$Key = PBKDF2(Prf, Password, Salt, Iterations, Length) \tag{1}$$

   - **Key:** The 32 byte Key generated.
   - **Prf:** The pseudorandom function used for key derivation. CryptoNote uses HMACSHA1.
   - **Password:** The user-provided password.
   - **Salt:** A randomly generated array of 32 bytes.

- **Iterations:** A user-provided unsigned 32-bit integer defining the number of PBKDF2 iterations for key generation. Default value is 1028.
- **Length:** The length in bytes of the key generated. This will always be 32.

3. **Message Preparation:**
   The validity check is prepended to the user's message. The validity check is the ASCII representation of:

   ```
   "<CryptoNoteVailidityCheck/>"
   ```

   An ASCII message of `"Hello World"` becomes, `"<CryptoNoteVailidityCheck/>Hello World"`. The message with prepended validity check is called the validated message.

4. **Encryption:**
   The cipher is generated using AES-256-CBC

$$Cipher = AES(Vm, Key, Iv) \tag{2}$$

   - **Cipher:** The encrypted cypher of the validated message.
   - **Vm:** The validated message.
   - **Key:** The key that was generated in step 2.
   - **Iv:** The AES-256 Initialization Vector. This is a randomly generated array of 16 bytes.

5. **Decryption:**
   In order to decrypt a cryptonote cipher, the following must be known:

   - **Cipher**
   - **Password**
   - **Salt**
   - **Iterations**
   - **Iv**

(a) **Key Derivation**:
The key is derived as outlined in 2 using the password, salt, and iterations.

(b) **Cipher Decryption:**
The validated message is derived by reversing the AES-256-CBC function using the key and the initialization vector.

(c) **Verification:**
The success of the decryption is verified by confirming that the validated message begins with the validity check, `"<CryptoNoteVailidityCheck/>"`. The remaining bytes are the original message.

4 File Format

A .cryptonote file contains the items in the following table. The order they appear in the table is the order they appear in the file (Protocol version is at bytes 0-1, Salt is at bytes 2-33, etc), Integers are in little-endian format. Signed integers use two's complement format.

Size in bytes	**Data type**	**Contents**
2	Unsigned Int	Protocol Version
32	Byte Array	Salt
4	Signed Int	Iterations
16	Byte Array	Initialization Vector
All remaining bytes	Byte Array	Cipher