Secure Data Exchange method (*SDEx*)

Artur Hłobaż, Krzysztof Podlaski, Piotr Milczarski
Faculty of Physics and Applied Informatics, University of Lodz
Pomorska 149/153, 90-236 Lodz, Poland
{artur.hlobaz,krzysztof.podlaski,piotr.milczarski}@uni.lodz.pl

May 11, 2017

1 Introduction

This document is a short description of encryption algotithm Secure Data Exchange method (*SDEx*) developed by Artur Hłobaż, Krzysztof Podlaski and Piotr Milczarski [1–5]. The reference implementation of this algorithm can be found in GitHub repository [6]. The reference implementation is created in c++ language.

2 Data encryption method used in *SDEx*

The algorithm of encryption used in *SDEx* method [1–3] is based on hash functions. In a standard application, hash functions are used to check integrity of data. These functions take as an input any length string of bits and return the result of a fixed size called a hash or a message digest. The length of hash usually is in the range of 128 to 512 bits, while the whole message can contain thousands or even millions of bits.

All hash functions are constructed iterative and divide the input data into a sequence of fixed size blocks $M_1, \dots ,M_t$ (Fig.1). Message blocks are sequentially processed using a hash function to the intermediate state of constant size. The process starts with the predetermined value h_0, while successive states $h_1, \dots, h_t$ are defined as $h_t = \mathrm{hash}(h_{t-1}; M_t)$. The h_t - result of the last iteration of the process - is the searched message digest.

The hash function used in *SDEx* encryption method performs a role of dynamic pseudorandom string of bits generator. The security level of hash functions has been estimated at half of the generated hash length because of the possibility of collision. To avoid this type of attack, we can modify the previously described method of calculating the message digest to hinder the reversal of the various stages of calculation. One of the possibilities is to use Davies-Meyer schema (Fig. 2) which allows hash function collision resistance

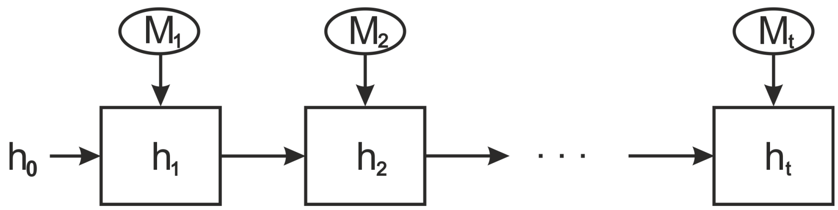

Figure 1: Message digest counting for any length string of bits.

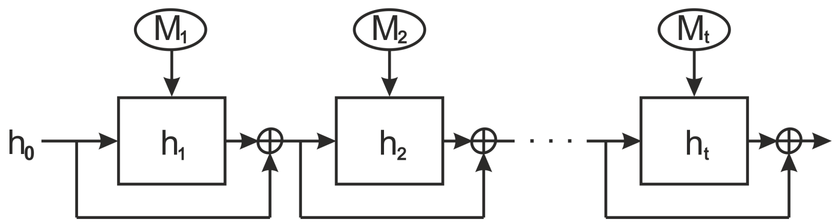

Figure 2: Davies-Meyer schema.

2.1 Description of the enhanced *SDEx* method

The common symbols used on diagrams 3-4 and equations:

- M_1, M_2, $\ldots$, M_i - plaintext blocks,
- C_1, C_2, $\ldots$ C_i - ciphertext blocks,
- $H_0 = H_{IV+h_0}$.
- IV - initialization vector (session key),
- h_1, h_2, $\ldots$ h_k - particular iterations of hash computation,
- H_{IV} - hash from the initialization vector,
- H_U - hash from user password,
- $\oplus$ - XOR operation.

The new enhanced schemes for encryption and decryption methods are shown in Figures 3 and 4. The equations that describe encryption operations take the following form:

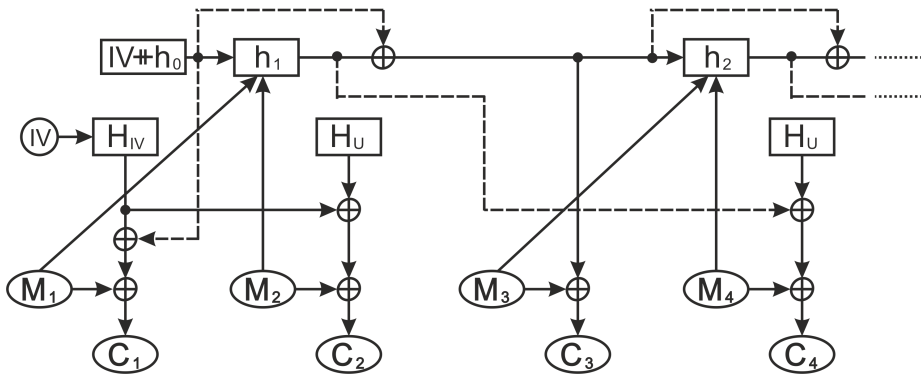

Figure 3: Strengthened encryption method.

$$
\begin{aligned}
C_1 &= M_1 \oplus H_{IV} \oplus H_{IV \mathbin{+\!\!+} h_0}, && && (1)\\
C_2 &= M_2 \oplus H_U \oplus H_{IV}, && && (2)\\
C_{2k+1} &= M_{2k+1} \oplus h_k \oplus h_{k-1} && (\text{where } k \geq 1), && (3)\\
C_{2k+2} &= M_{2k} \oplus H_U \oplus h_k && (\text{where } k \geq 1), && (4)\\
h_1 &= \mathrm{hash}(H_{IV \mathbin{+\!\!+} h_0}; M_1 \mathbin{+\!\!+} M_2), && && (5)\\
h_2 &= \mathrm{hash}((h_{k-1} \oplus H_{IV \mathbin{+\!\!+} h_0}); M_1 \mathbin{+\!\!+} M_2), && && (6)\\
h_k &= \mathrm{hash}((h_{k-1} \oplus h_{k-2}); M_{2k-1} \mathbin{+\!\!+} M_{2k}) && (\text{where } k \geq 3). && (7)
\end{aligned}
$$

Hence, the new decryption algorithm is defined as follows:

$$
\begin{aligned}
M_1 &= C_1 \oplus H_{IV} \oplus H_{IV \mathbin{+\!\!+} h_0}, && && (8)\\
M_2 &= C_2 \oplus (H_U \oplus H_{IV}), && && (9)\\
M_{2k+1} &= C_{2k+1} \oplus h_k \oplus h_{k-1} && (\text{where } k \geq 1), && (10)\\
M_{2k+2} &= C_{2k} \oplus H_U \oplus h_k && (\text{where } k \geq 1). && (11)
\end{aligned}
$$

The element $H_{IV \mathbin{+\!\!+} h_0}$ in equations (1, 8) is just a hash from the string created as a concatenation of IV and h_0. The enhanced encryption/decryption methods are presented of Figures 3 - 4.

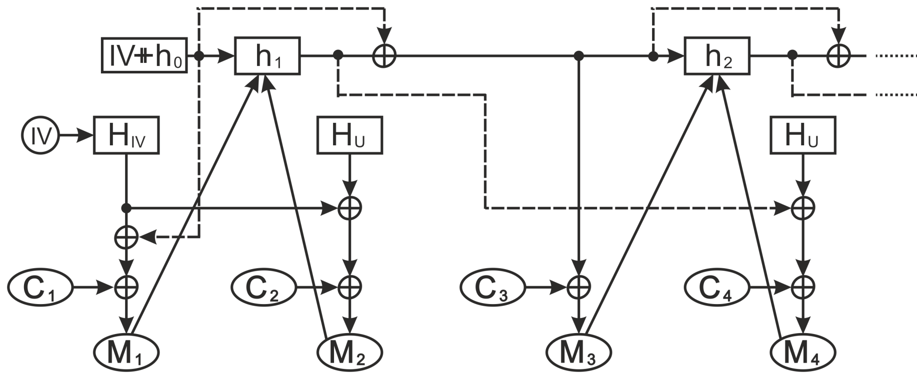

Figure 4: Strengthened decryption method.

3 Test samples

In the repository [6] one can find also an example file `a.txt` and its encrypted version with use of IV:*Ala_ma_kota* and password:*Ala_nie_ma_kota* (see method main_encrypt_test() in SDEx.cpp file). This two files can be used for test of any implementations.

4 License

The reference implementation of SDEx method is published on GitHub and can be used on the base of Modified BSD License below. The implementation of sha256 algorithm is based on zeedwood.com implementation [7].

References

[1] Podlaski, K., Hłobaż, A., Milczarski, P.: Secure data exchange based on social networks public key distribution. In: Internet of Things. IoT Infrastructures - Second International Summit, IoT 360° 2015, Rome, Italy, October 27-29, 2015, Revised Selected Papers, Part I. (2015) 52–63

[2] Milczarski, P., Podlaski, K., Hłobaż, A.: Applications of secure data exchange method using social media to distribute public keys. In: Computer Networks - 22nd International Conference, CN 2015, Brunów, Poland, June 16-19, 2015. Proceedings. (2015) 389–399

[3] Hłobaż, A., Podlaski, K., Milczarski, P.: Applications of qr codes in secure mobile data exchange. In Kwiecień, A., Gaj, P., Stera, P., eds.: Computer Networks: 21st International Conference, CN 2014, Brunów, Poland, June 23-27, 2014. Proceedings, Springer International Publishing (2014) 277–286

[4] Hłobaż, A.: Security of measurement data transmission - modifications of the message encryption method along with concurrent hash counting. FSNT NOT (1) (2008) 39–42

[5] Hłobaż, A.: Security of measurement data transmission - message encryption method with concurrent hash counting. SEP (1) (2007)

[6] Github repository: SDEx reference implementation. https://github.com/kpodlaski/SDEx_ref_impl (2017)

[7] zedwood.com: C++ sha256 function. http://www.zedwood.com/article/cpp-sha256-function (2012)