PS-HOMO: A Protection Scheme for iOS SMS

Guomiao Zhou¹, Qianqiong Wu², Ming Duan¹* and Qi Xi¹

¹State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, Henan,450000, China
²Institute of Electrical Engineering and Automation, Luoyang Institute of Science and Technology, Luoyang, Henan,471000, China
*Corresponding author’s e-mail: mdscience@sina.com

Abstract. iOS Short Message Service (SMS) transmits messages in plaintext, which would lead to potential security threats of data. It is meaningful to design a safe and effective SMS. In this paper, all iOS Security Guides are referred to and all the cryptographic algorithms used in iOS data protection are analyzed. Because there is no homomorphic encryption, a library of homomorphic encryption algorithms for iOS based on RSA and Paillier is developed. Using this library, a protection scheme PS-HOMO for protecting iOS SMS data is proposed and implemented. Under the premise of ensuring the data confidentiality, PS-HOMO also has the function of ciphertext operation due to the multiplicative homomorphism of RSA and additive homomorphism of Paillier. A comprehensive performance test was performed on PS-HOMO. The results show that the performance is in line with our expectations and will not have a major impact on the original system. The security of PS-HOMO is theoretically analyzed from man-in-the-middle attack, replay attack, and traffic analysis attack. Two application scenarios of the PS-HOMO, remote anonymous voting and privacy telemedicine service are envisaged. We hope PS-HOMO would play a significant role in privacy protection.

1. Introduction

As the iPhone becomes part of our lives, we use it in many ways. One of ways is communicating with others by its SMS which we rely heavily on. But according to our previous research [1], we found that the iOS SMS message is plaintext transmission without any confidential protection. Thus, if an iPhone jailbroken, the communication content will be exposed, and the privacy information will be leaked.

The privacy information on iOS SMS is under risks. Thus, it is necessary to add protection for the private data. In this paper, we referred to all the iOS Security Guides, and combed all the cryptographic algorithms used in iOS data protection. Because there is no homomorphic encryption, a library of homomorphic encryption algorithms for iOS based on RSA and Paillier is developed. Using this library, a protection scheme PS-HOMO for protecting iOS SMS data is proposed and implemented. By injecting a self-development tweak, hooking the key functions of sending and receiving messages, the functionality of iOS Message application is modified. Some tests are also designed and did, to illustrate fully its practicability. And the results show that the performance is in line with our expectations and will not have a major impact on the original system. The security of PS-HOMO is analyzed theoretically from man-in-the-middle attack, replay attack, traffic analysis attack, and the results show that PS-HOMO can be trusted. Two practical application scenarios of PS-HOMO, remote anonymous vote, private telemedicine service, are also envisaged.
2. Background

iOS is an operating system for iPhone, iPad, and other Apple devices. It is well-known by its high security. In this section, some background information covered in this paper is briefly introduced. On iOS tweak development, the major tools are as follows.

Cycrypt [2-3]. Cycrypt is a JavaScript interpreter which also understands Objective-C syntax. It allows developers to explore and modify running applications through an interactive console.

Theos [4-5]. Theos is an open-source jailbreaking development tool, created by Dustin Howett. It can develop iOS software without Xcode which is official Integrated Development Environment (IDE). It is the most iOS tweak developers' choices.

Cydia Substrate [6-7]. Cydia Substrate, which formerly called Mobile Substrate, contains three major components: Mobile Hooker, Mobile Loader and Safe Mode. It is a framework that allows third-party developers to provide run-time patches to system functions, created by Jay Freeman (Saurik).

3. Cryptographic Algorithms in iOS File System

For the data protection, iOS built-in many cryptographic algorithms, as table 1 which refers to all the iOS Security Guides shows.

Table 1. Application And Evolution Of iOS Cryptographic Algorithms.

| iOS Version | Hardware Key | RNG® | Data Protection Class | Password | Keychain | Per-File Key | IV® |
|-------------|--------------|------|-----------------------|----------|----------|-------------|-----|
| iOS 4 X | AES-128 Yarrow | — | — | AES-128 Hash | AES-128 Hash | LFSR |
| iOS 5 (2010.06) | AES-128 Yarrow | ECDH® | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |
| iOS 5 (2011.06) | AES-256 Yarrow | ECDH | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |
| iOS 6 (2012.06) | AES-256 Yarrow | ECDH | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |
| iOS 7 (2013.06) | AES-256 CTR_DRBG | ECDH | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |
| iOS 8 (2014.06) | AES-256 CTR_DRBG | ECDH | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |
| iOS 9 (2015.09) | AES-256 CTR_DRBG | ECDH | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |
| iOS 10 (2016.06) | AES-256 CTR_DRBG | ECDH | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |
| iOS 11 (2017.09) | AES-256 CTR_DRBG | ECDH | SHA-256 | AES-128 (GCM) SHA-1 | AES-128 (GCM) SHA-1 | LFSR |

According to table 1, all the cryptographic algorithms iOS uses are standard algorithms, and can resist to cryptanalysis very well.

Extending hardware key length from 128 bits to 256 bits and embedding the AES-256 in the crypto engine correspondingly, provides higher security for data encryption. It is crypto engine that mitigates the slow operation speed and consuming more power while extending to 256 bits.

Before iOS 6, Yarrow, which includes SHA-1, are used in random number generator. In 2005, professor Wang Xiaoyun proposed a new collision search against SHA-1 [8], which reduces the complexity of the full SHA-1 collision search to 269 and destroyed the security of SHA-1 theoretically. In 2007, researchers of Google public their results [9-10], the first SHA-1 collision in practice. Owing to the unsafe SHA-1, Counter mode Deterministic Random Byte Generator (CTR_DRBG) are introduced.
In the newest iOS 12, AES-256-GCM are used in encryption of keychain items. Galois/Counter Mode (GCM) is mixed of Galois Message Authentication Code Mode (GMAC) and Counter Mode (CTR), providing encryption and integrity checking of messages.

In the early iOS, AES-CBC was used in encryption for the files. After iOS 9, AES-XTS was introduced. In iOS 12, AES-XTS are used in most devices. In XTS, tweak keys, which are mutually independent but AES keys are same, are added. Compared to the CBC, XTS has two characteristics: One is that the same block gets different ciphertexts, and the other is that different blocks can be encrypted and decrypted independently. The results of SHA-1 are mainly used as initialization vector, so it is still in use.

A larger number of iterations has been set for the PBKDF2 (at least 10,000 times in iOS 4 [11]) by Apple, making a password attempt takes about 80 milliseconds. So, traversing all alphanumeric combinations of length 6 takes more than five and a half years. The larger number of iterations also means that the stronger the password, the stronger the exported password key.

4. Homomorphic Encryption Algorithms Library for iOS File System

Homomorphism is a concept in modern algebra originally. The definition of group homomorphism is as follows.

Let \((G, +)\) and \((H, \ast)\) be groups. The map \(\varphi : G \rightarrow H\) is a homomorphism if
\[
\varphi(x + y) = \varphi(x) \ast \varphi(y) \quad \forall x, y \in G
\]

In 1978, Ronald L. Rivest et al. first proposed the idea of homomorphic encryption which appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption [12], and four possible encryption functions (RSA is one of them) are introduced in their paper. The encryption function has homomorphic if
\[
\text{Dec}(\text{Enc}(m_1) \otimes \text{Enc}(m_2)) = m_1 \oplus m_2 \quad \forall m_1, m_2 \in M
\]

\(M\) is a plaintext domain. \(\text{Enc}, \text{Dec}\) are encryption and decryption functions, respectively. \(\oplus, \otimes\) are operations on plaintext and ciphertext domains, respectively.

After this idea proposed, many homomorphic encryption schemes were designed by cryptologists, such as schemes in papers [13-17] have additive homomorphism and scheme in paper [18] has multiplicative homomorphism. Also, these schemes are applied in practice, like E-Voting [19-20], private information retrieval (PIR) [21-22], cloud computing [23-27] and so on. In 2005, Dan Boneh et al. proposed the first scheme [28] which has both additive and multiplicative homomorphism, and it supports any number of additions but only one multiplication.

A cryptosystem that supports arbitrary computation on ciphertexts is known as fully homomorphic encryption (FHE) [29]. In 2009, the first fully homomorphic encryption (FHE) using ideal lattices is proposed by Craig Gentry in his PhD thesis [30]. After Gentry’s contribution, many cryptologists made progress on FHE. In 2010, Marten van Dijk et al. proposed a FHE over the integers [31] based on Gentry’s technology. In 2010, Nigel P. Smart and Frederik Vercauteren simplified the implement of Gentry09 scheme over PKC [32]. In 2010, Craig Gentry and Shai Halevi implemented more simplification for Gentry09 scheme [33]. But due to the computational complexity, all the FHE schemes mentioned above are not practical.

Other previous researches on iOS with cryptography mainly focused on the following aspects. First, literatures [34-35] have proposed the analysis tools for misuse of cryptography in iOS applications. Second, literatures [36-37] have implemented the cryptographic algorithm simply. Third, literature [38] has implemented the cryptographic algorithm and applied it to the application developed by the author.

In 1978, Ron Rivest et al. proposed the first public key cryptosystem called RSA [39] and it widely used for secure data transmission. RSA has multiplicative homomorphism. In 1999, Paillier cryptosystem is introduced by Pascal Paillier in his paper [20]. Paillier has additive homomorphism. From the chapter 3, there are not any homomorphic encryption algorithms in iOS. Thus, in this paper, a homomorphic encryption algorithms library is developed for iOS. Due to the computational complexity and mobile computing power, it is difficult to use FHE on mobile platform. So, the
combination of RSA and Paillier are built-in homomorphic encryption algorithms library, supporting multiplicative and additive homomorphism, respectively.

4.1 RSA Multiplicative Homomorphic Encryption

4.1.1 Key Generation. In theory, RSA keys generation is as follows. Choose two large prime numbers \( p \) and \( q \) in secret randomly. Compute
\[
|n| = pq.
\]
\[
\phi(n) = (p - 1)(q - 1), \phi \text{ is Euler's totient function.}
\]
Select a positive integer \( e(1 < e < \phi(n)) \) randomly, and make
\[
\gcd(e, \phi(n)) = 1.
\]
Determine the positive integer \( d \), which meets
\[
e d \equiv 1 \pmod{\phi(n)}.
\]
\((n, e)\) is released as the public key.
\((n, d)\) is kept as the private key.

4.1.2 Encryption and Decryption Operation.
Encryption. For plaintext \( m \in \mathbb{Z}_n \), the corresponding ciphertext is
\[
c \equiv m^e \pmod{n}.
\]
Decryption. For ciphertext \( c \in \mathbb{Z}_n \), the corresponding plaintext is
\[
m \equiv c^d \pmod{n}.
\]

4.1.3 Verification of Multiplicative Homomorphism. Encrypt different plaintext \( m_1 \) and \( m_2 \).
\[
E(m_1) \equiv m_1^e \pmod{n}, E(m_2) \equiv m_2^e \pmod{n}, E \text{ is RSA encryption function.}
\]
Multiply ciphertexts,
\[
E(m_1) \cdot E(m_2) \equiv m_1^e \cdot m_2^e \equiv (m_1 \cdot m_2)^e \equiv E(m_1 \cdot m_2) \pmod{n}.
\]
Thus, RSA cryptosystem is multiplicative homomorphism.

4.2 Paillier Additive Homomorphic Encryption

4.2.1 Key Generation. In theory, Paillier keys generation is as follows. Choose two large prime numbers \( p \) and \( q \) randomly. Compute
\[
|n| = pq,
\]
\[
\lambda = \text{lcm}(p - 1, q - 1) = \frac{(p - 1)(q - 1)}{\gcd(p - 1, q - 1)}, \text{lcm is least common multiple.}
\]
Select random integer \( g \) where \( g \in \mathbb{Z}_n^* \).
Compute
\[
|\mu| = \left( L\left(g^x \pmod{n^2}\right) \right)^{-1} \pmod{n}, L(x) = (x - 1)/n.
\]
\((n, g)\) is released as the public key.
\((\lambda, \mu)\) is kept as the private key.

4.2.2 Encryption and Decryption Operation.
Encryption. For plaintext \( m \in \mathbb{Z}_n \), select random number \( r \in \mathbb{Z}_n^* \), the corresponding ciphertext is
\[
c \equiv g^m \cdot r^n \pmod{n^2}.
\]
Decryption. For ciphertext \( c \in \mathbb{Z}_n^* \), the corresponding plaintext is
\[
m \equiv L\left(c^x \pmod{n^2}\right) \cdot \mu \pmod{n}.
\]

4.2.3 Verification of Additive Homomorphism. Encrypt different plaintext \( m_1 \) and \( m_2 \).
\[
E(m_1, r_1) \equiv g^{m_1} \cdot r_1^n \pmod{n^2}, E(m_2, r_2) \equiv g^{m_2} \cdot r_2^n \pmod{n^2},
\]
$E$ is Paillier encryption function.

Add ciphertexts.

$$E(m_1, r_1) \cdot E(m_2, r_2) \equiv g^{m_1 \cdot r_1^n} g^{m_2} \cdot r_2^n \equiv g^{m_1 + m_2} \cdot (r_1 r_2)^n \equiv E(m_1 + m_2)(\text{mod} n^2).$$

Thus, Paillier cryptosystem is additive homomorphism.

5. PS-HOMO: A Protection Scheme for iOSSMS Message

PS-HOMO includes users and a cloud server. Cloud server which is a third-party trusted certificate server is responsible for store and distribute certificates. It maintains a certificate list whose items are users’ public key. In fact, these certificates are sent by users when users start up. When Alice wants to send a SMS message to Bob, the following steps are automatic. At a start, PS-HOMO checks the local certificate list whether stores the Bob’s certificate or not. If there is, Alice sends encrypted message directly to Bob using the Bob’s public key in the certificate, otherwise, Alice sends a request to the cloud server for Bob’s certificate. When the cloud server receives the request, it sends the Bob’s certificate to Alice with its signature. Alice receives Bob’s certificate and verifies the signature using public key of cloud server (root certificate). After verification, Alice sends encrypted message to Bob, and Bob decrypts the message by his private key. The overview of PS-HOMO as Figure 1 shows.

Figure 1. The Overview of PS-HOMO.

A tweak is injected into the imagent process to implement the modification of functionality. Hook function is the main method, through the reverse engineering, by hooking functions “sendMessage:” on the sending side and “_process Received Dictionary: storage Context:” on the receiving side. GMP (GNU Multiple Precision), which is a free library for arbitrary precision arithmetic, is introduced to support the calculation of big numbers [40-41]. Two types of SMS messages, certificate message and homomorphic encryption message, and other related technologies are introduced as follows.

5.1 Certificate Message

When Alice needs Bob’s public key, she sends Bob’s phone number to cloud server for Bob’s certificate. The cloud server searches Bob’s certificate and signs the information header using its private key. Then, it sends the certificate and signature to Alice. Alice verifies the signature using
public key (root certificate) of cloud server and extracts Bob’s public key. Alice’s processing of getting certificate as Figure 2 shows.

Figure 2. Alice’s Processing of Requesting Certificate.

5.2 Homomorphic Encryption Message
After Alice obtained Bob’s public key, she can send encrypted message to Bob. First of all, Alice input text on the interface of iOS Message application; Then, encrypting timestamp and the text using Bob’s public keys; What’s more, adding an information header at the start of ciphertext; At last, encoding the message using Base64 for the efficient transmission. On the Bob’s side, the opposite process is executed to obtain the Alice’s input. As Figure 3 shows.

Figure 3. Process of Sending Homomorphic Encryption Message.

5.3 Message Information Header
Information header which is used in two message types is designed to specify the information of a message. As Table 2 shows.

Table 2. Information Header of SMS Message.

| 8 bytes                        |                  |
|-------------------------------|-----------------|
| Header Start Tag (4)          | Message Type (4) |
| Encryption Mode (4)           | Text Length (4)  |
| Phone Number (16)             |                 |
| Timestamp (14)                |                 |
| Key ID (8)                    |                 |
| Signature Length (4)          |                 |
| Header End Tag (4)            |                 |
The meaning of every field is as table 3 shows.

Table 3. Meaning of Every Field.

| Index | Field                | Length | Value       | Meaning                                      |
|-------|----------------------|--------|-------------|----------------------------------------------|
| 1     | Header Start Tag     | 4 bytes| ‘#HS&’      | Information header starts                    |
| 2     | Message Type         | 4 types| ‘EncM’      | Message of ciphertext                        |
|       |                      |        | ‘KeyC’      | Message of certificate                       |
|       |                      |        | ‘EncR’      | Ciphertext by RSA                            |
| 3     | Encryption Mode      | 4 bytes| ‘EncP’      | Ciphertext by Paillier                       |
|       |                      |        | ‘ER&P’      | Ciphertext by RSA & Paillier                 |
| 4     | Text Length          | 4 bytes| Length      | Bytes of message                            |
| 5     | Phone Number         | 16 bytes| Number     | Phone number of certificate                  |
| 6     | Timestamp            | 14 bytes| Timestamp  | Creation time of message                     |
| 7     | Reserved             | 2 bytes| ‘f’ for all | Reserved                                     |
| 8     | Signature Length     | 4 bytes| Signature   | Length of cloud server signature             |
| 9     | Key ID               | 8 bytes| ID          | ID of key pairs                              |
| 10    | Header End Tag       | 4 bytes| ‘#HE&’      | Information header ends                      |

5.4 SMS Message Encoding
For higher information transmission efficiency and reliability, Base64 is used to encode the message to be sent. According to RFC 2045 [42], 64 printable characters are chosen to represent binary data.

5.5 Performance Test
In order to verify the performance of modified iOS Message application, some tests are designed. Two devices, one as user and one as cloud server, have the same parameters, and the specific information of the devices are as table 4 shows.

Table 4. The Specific Information of Devices for The Tests.

| Item                  | Information                                      |
|-----------------------|--------------------------------------------------|
| Model                 | iPhone 6                                         |
| Operating System      | iOS 10.2                                        |
| Darwin Kernel Version | 16.3.0                                          |
| XNU Version           | 3789.32.1                                       |
| CPU                   | A8 chip with 64-bit architecture & M8 motion coprocessor |
| Core Numbers          | Dual Core                                       |
| CPU Clock Speed       | 1.4 GHz                                         |
| GPU                   | Imagination PowerVR SGX6450                      |
| RAM                   | 1 GB                                             |
| ROM                   | 16 GB                                            |
| Modem Firmware        | 5.32.00                                         |
| Network               | CHINA MOBILE                                    |

Usage of CPU and Memory. Instruments [43], which is a performance-analysis and testing tool, is powerful and flexible. As part of Xcode tool set, it is used for measuring the real-time usage of CPU and memory. The results of four cases, non encryption, RSA only, Paillier only, both RSA and Paillier, are listed in Figure 4 and table 5.
According to the measured data, the statistics are in reasonable intervals. Thus, the new added operations do not affect the program execution significantly.

5.5.1 Time Cost of Key Generation. Capturing the moments precisely before and after key generation operation, the time difference is the time cost of key generation. The results of three measurements for each case are shown as table 6. The target lengths of RSA and Paillier key pairs are 1024 bits and 512 bits, respectively.

Table 6. Time Cost of Key Generation

| Index | Examples      | First (ms) | Second (ms) | Third (ms) |
|-------|---------------|------------|-------------|------------|
| 1     | RSA (Fig. 4.b)| 159.40     | 76.64       | 88.59      |
| 2     | Paillier      | 177.86     | 135.89      | 206.47     |

According to the measured data, time cost of all three cases are within the acceptable limits.

5.5.2 Time Cost of Encryption and Decryption. Capturing the moments precisely before and after encryption and decryption operations, the time differences are the time cost of encryption and decryption. The results of three measurements for each case are shown as table 7. The plaintext input is ‘Life was like a box of chocolates, you never know what you're gonna get.’
Table 7. Time Cost of Encryption.

| Index | Examples | First (ms) | Second (ms) | Third (ms) |
|-------|----------|------------|-------------|------------|
| 1     | RSA      | 1.0280     | 1.0670      | 1.1024     |
| 2     | Paillier | 41.9169    | 40.1079     | 50.0589    |

Time Cost Of Decryption.

| Index | Examples | First (ms) | Second (ms) | Third (ms) |
|-------|----------|------------|-------------|------------|
| 1     | RSA      | 6.0459     | 5.8261      | 6.3700     |
| 2     | Paillier | 45.6119    | 47.7629     | 47.4399    |

According to the measured data, time cost of all three cases are within the acceptable limits.

5.5.3 Security and Pressure Test

Another tweak has been developed, to intercept the processed messages and log them. According to the log information, all messages are encrypted, and contents are confidential.

The group sending and extra-long message sending are also tried, to verify the tweak for iOS Message application still working under extreme conditions.

6. Security Analysis of PS-HOMO

PS-HOMO is a secure scheme and can against many mainstream attacks.

6.1 Man-In-The-Middle (MITM) Attack

A MITM attack is an attack where the attacker secretly relays and possibly alters the communications between two parties who believe they are directly communicating with each other [44]. In PS-HOMO, the existence of third-party trusted certificate server can resist the MITM attack. Certificate server is root trust, so it is trusted by all users. Alice trusts Bob’s certificate from cloud server, for she can verify signature of cloud server and confirm that no any other attacker tampered with it in the middle. Alice obtains correct Bob’s public keys.

6.2 Replay Attack

A replay attack (a.k.a. playback attack) is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed [45]. In PS-HOMO, timestamp is used to prevent from replay attack. For preventing timestamps from being tampered with, in certificate message, timestamp is set in information header which cloud server signs. Alice verifies the signature and checks the timestamp. In homomorphic encryption message, timestamp is set in information header and ciphertext. Bob compares the two timestamps after decrypting the ciphertext.

6.3 Traffic Analysis Attack

Due to the encryption operation, the ciphertext is fully confused and spread. If an attacker intercepts the message, he can’t obtain communication contents.

7. Applications of PS-HOMO

PS-HOMO is secure communication and has homomorphic operation capability. Thus, it can be applied in some scenarios, if specific feature plugins are developed and inserted into PS-HOMO. Some application scenarios are listed.

7.1 Remote Anonymous Voting

When you are on a business trip, your boss called you to vote for something, but you don’t want to let him know your vote. It is anonymous and remote for you to using PS-HOMO. Deciding your vote and encrypting by homomorphic encryption. Then, message which carries your vote arrives at your boss phone. After ciphertext calculation and decryption, remote anonymous voting was operated.
7.2 Privacy Telemedicine Service
With the help of PS-HOMO, it is convenient to implement privacy telemedicine service. Remote computing service provider receives the encrypted medical record data from client. Then, the data are processed by service provider without decryption because of homomorphic encryption technology. The processing results in encryption state are sent back to client. Client decrypts and reads them.

8. Conclusion
According to our previous research, the iOS SMS message is plaintext transmission without any confidential protection. This puts private information at risks. Thus, we referred to all the Apple security white books and sorted all cryptographic algorithms out. There is no any homomorphic encryption algorithm in iOS. So, a library of homomorphic encryption algorithms is developed for iOS SMS data based on RSA and Paillier. Then, using this library, a protection scheme PS-HOMO for iOS SMS is proposed and implemented. After injecting a self-development tweak, the functionality of iOS SMS is modified. Some performance tests are designed and did. The results show it has effect and in line with our expectations. The security of PS-HOMO is also theoretically analyzed from man-in-the-middle attack, replay attack, traffic analysis attack, and the results show that PS-HOMO can be trusted. Two practical application scenarios of PS-HOMO, remote anonymous vote, private telemedicine service, are also envisaged. So, PS-HOMO is practical and we wish it would play a significant role in iOS SMS privacy information protection.

References
[1] Zhou, G.M., Duan, M., Xi, Q., Wu, H. (2018) ChanDet: Detection Model for Potential Channel of iOS Applications. In: ISPECE 2018, Xi’an. pp. 1106-1114.
[2] J. Freeman. (2014) Cycript. http://www.cycript.org/.
[3] iPhoneDevWiki. (2016) Cycript. http://iphonedevwiki.net/index.php/Cycript.
[4] GitHub. (2018) Theos. https://github.com/theos/theos/wiki/Installation.
[5] iPhoneDevWiki. (2019) Theos. http://iphonedevwiki.net/index.php/Theos.
[6] J. Freeman. (2012) Cydia Substrate. http://www.cydiasubstrate.com/.
[7] iPhoneDevWiki. (2018) Cydia Substrate. http://iphonedevwiki.net/index.php/Cydia_Substrate.
[8] Wang, X., Yin, Y.L., Yu, H. (2005) Finding Collisions in the Full SHA-1. In Advances in Cryptology – CRYPTO 2005, Santa Barbara, pp. 17-36.
[9] Google. (2007) Shattered. https://shattered.it/.
[10] Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y., (2017) The first collision for full SHA-1. In Advances in Cryptology – CRYPTO 2017, Santa Barbara, pp. 570-596.
[11] Katalov, V. (2010) Smartphone Forensics: Cracking BlackBerry Backup Passwords. https://blog.elcomsoft.com/2010/09/smartphone-forensics-cracking-blackberry-backup-passwords/.
[12] Rivest, R.L., Adleman, L., Dertouzos, M.L. (1978) On data banks and privacy homomorphisms. Foundations of secure computation, vol. 4(11): 169-180.
[13] Damgard, I., Jurik, M. (2001) A Generalisation, a Simplification and Some Applications of Paillier’s Probabilistic Public-Key System. In: The 4th International Workshop on Practice and Theory in Public Key Cryptosystems, Cheju Island, pp. 119-136.
[14] Goldwasser, S., Kharchenko, D. (2005) Proof of Plaintext Knowledge for the Ajtai-Dwork Cryptosystem. In: Theory of Cryptography Conference, Cambridge, pp. 529-555.
[15] Goldwasser, S., Micali, S. (1982) Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information. In the fourteenth annual ACM symposium on Theory of computing, San Francisco, pp. 365-377.
[16] Kawachi, A., Tanaka, K., Xagawa, K. (2007) Multi-bit Cryptosystems Based on Lattice Problems. In: International Workshop on Public Key Cryptography, Beijing, pp. 315-329.
[17] Naccache, D., Stern, J. (1998) A New Public-Key Cryptosystem Based on Higher Residues. In: 5th ACM Conference on Computer and Communications Security, San Francisco, pp. 59–66.
[18] Elgamal, T. (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, vol. 31, no. 4: 469-472.

[19] Cohen, J.D., Fischer, M.J. (1985) A robust and verifiable cryptographically secure election scheme. http://www.cs.yale.edu/homes/fischer/pubs/tr416.pdf.

[20] Paillier, P. (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology — EUROCRYPT ’99, Prague, pp. 223-238.

[21] Melchor, C.A., Gaborit, P. (2007) A lattice-based computationally-efficient private information retrieval protocol. In: WEWoRC 2007, Bochum, pp. 50-54.

[22] Lipmaa, H. (2005) An oblivious transfer protocol with log-squared communication. In: 8th International Conference on Information Security, Singapore, pp. 314-328.

[23] Tebaa, M., Hajii, S.E., Ghazi, A.E. (2012) Homomorphic encryption applied to the cloud computing security. In: the World Congress on Engineering 2012, London, pp. 4-6.

[24] Zhao, F., Li, C., Liu, C.F. (2014) A cloud computing security solution based on fully homomorphic encryption. In: 16th International Conference on Advanced Communication Technology, Pyeongchang, pp. 485-488.

[25] Ibtiial, M., Driss, E.O., Hassan, N. (2017) Homomorphic encryption as a service for outsourced images in mobile cloud computing environment. International Journal of Cloud Applications and Computing (IJCAC), vol. 7, no. 2: 27-40.

[26] Tebaa, M., Hajii, S.E. (2013) Secure cloud computing through homomorphic encryption. International Journal of Advancements in Computing Technology (IJACT), vol. 5, no. 16.

[27] Atayero, A.A., Feyisetan, O. (2011) Security issues in cloud computing: The potentials of homomorphic encryption. Journal of Emerging Trends in Computing and Information Sciences, vol. 2, no. 10: 546-552.

[28] Boneh, D., Goh, E.J., Nissim, K. (2005) Evaluating 2-DNF formulas on ciphertexts. In: Theory of Cryptography Conference, Cambridge, pp. 325-341.

[29] WikiPedia. (2016) Homomorphic Encryption. https://en.wikipedia.org/wiki/Homomorphic_encryption.

[30] Gentry, C. (2009) Fully homomorphic encryption using ideal lattices. In: STOC 2009, Bethesda, pp. 169-178.

[31] Dijk, M.V., Gentry, C., Halevi, S., Vaikuntanathan, V. (2010) Fully homomorphic encryption over the integers. In: Advances in Cryptology – EUROCRYPT 2010, French Riviera, pp. 24-43.

[32] Smart, N.P., Vercauteren, F. (2010) Fully homomorphic encryption with relatively small key and ciphertext sizes. In: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, pp. 420-443.

[33] Gentry, C., Halevi, S. (2011) Implementing gentry's fully-homomorphic encryption scheme. In: Advances in Cryptology – EUROCRYPT 2011, Tallinn, pp. 129-148.

[34] Feichtner, J., Missmann, D., Spreitzer, R. (2018) Automated Binary Analysis on iOS: A Case Study on Cryptographic Misuse in iOS Applications. In: 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks, Stockholm, pp. 236-247.

[35] Li, Y., Zhang, Y., Li, J., Gu, D. (2014) icerptracer: Dynamic analysis on misuse of cryptography functions in iOS applications. In: 8th International Conference on Network and System Security, Xi’an, pp. 349-362.

[36] Shen, Y., Wang, H. (2017) Enhancing data security of iOS client by encryption algorithm. In: IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, pp. 366-370.

[37] Deng, G., Long, Y. (2018) SM2 Mobile Cryptosystem Based on iOS Terminal. COMPUTER ENGINEERING & SOFTWARE, vol. 39, no. 2: 28-31.

[38] Qin, Y. (2016) Implementation and Application of Elliptic Curve Cryptography on iOS Platform. M.S. thesis, Dept. science & technology, Huazhong Univ., Wuhan, China, 2016.
[39] Rivest, R.L., Shamir, A., Adleman, L. (1978) A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, vol. 21, no. 2: 120-126.

[40] GMP. (2016) What is GMP. https://gmplib.org/#WHAT.

[41] CSDN. (2019) Generate framework for GMP and introduce the new framework into theos. https://blog.csdn.net/wudishuishuishui/article/details/89647331.

[42] Request ForComments. (1996) Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.

[43] Apple Inc. (2019) Instruments Help. Cupertino.

[44] Wikipedia. (2017) Man in the middle attack. https://en.wikipedia.org/wiki/Man-in-the-middle_attack.

[45] Wikipedia. (2016) Replay attack. https://en.wikipedia.org/wiki/Replay_attack.