Scientific and Practical Cyber Security Journal (SPCSJ) 3(3): 40 - 44 ISSN 2587-4667 Scientific Cyber Security Association (SCSA)

Improvement of Implementation of Merkle Crypto System

Улучшение Реализации Крипто Системы Меркле

Артуро Аракельян[1], Олег Полихенко[2]
Грузинский университет, Тбилиси, Грузия[1]
Национальный авиационный университет, Киев, Украина[2]

ABSTRACT:

This article describes hash-based digital signature systems. These systems are safe against quantum computer attacks. These systems have performance problems. We implemented the Merkle digital signature algorithm using recursion. A performance analysis was conducted. To improve the efficiency in the implementation of this algorithm, we replaced the recursion with loops. An analysis of the resulting implementation was carried out. Changing the implementation gave us very good results.

Резюме.

В данной статье описаны системы электронной подписи, основанные на хешировании. Данные системы защищены от атак квантового компьютера. Данные системы имеют проблемы эффективности. Мы реализовали алгоритм электронной подписи Меркле, с помощью рекурсии. Проведен был анализ эффективности. Для улучшения эффективности в реализации данного алгоритма рекурсию мы заменили циклами. Был проведен анализ полученной реализации. Изменение реализации дало довольно хорошие результаты.

KEYWORDS: Merkle, crypto system, improvement, cryptography

Идет активная работа над разработкой и усовершенствованием квантовых компьютеров. Криптосистемы, которые используются в практике уязвимы к атакам квантовых компьютеров. Безопасность этих систем основана на проблеме факторизации больших чисел и вычислении дискретных логарифмов, а квантовый компьютер может легко решить эту проблему[1,2].

Ведется активная работа над созданием криптосистем, которые защищены от атак квантового компьютера. Такими являются системы электронной подписи, основанные на хешировании. Безопасность данных крипто систем основывается на стойкости к коллизиям хеш функций.

Схема одноразовой подписи Лэмпорта

Была предложена схема одноразовой подписи Лэмпорта (Lamport–Diffie one-time signature scheme), данная схема является электронной подписью, основанной на хешировании. В этой схеме генерация ключа и генерация подписи являются эффективнымы, но размер подписи является довольно большим.

Схема одноразовой подписи Винтерница

Для уменьшения размера подписи была предложена схема одноразовой подписи Винтерница (Winternitz one-time signature scheme). В данной схеме одной строчкой ключа подписываются одновременно несколько битов хешированного сообщения, этим уменьшается длина подписи.

Схемы одноразовой подписи неудобны в использовании, потому что для подписи каждого сообщения нужно использовать уникальную пару ключей.

Крипто система Меркле

Была предложена крипто система Меркле для решения проблемы одноразовой пары ключей. В Merkle используется бинарное дерево, для замещения большого количества ключей верификации одним открытым ключом, корнем бинарного дерева. Данная криптосистема использует схему одноразовой подписи Лэмпорта или Винтерница и криптографическую хеш функцию

Нами был реализован алгоритм данной системы, в данном алгоритме использована рекурсия.

Рекурсия:

1. Importing necessary libs
2. Define class
3. Defining “ alt_hashes(hashes) “ method
4. Set list “ arr ”
5. If hashes == “ ”, raise Exception
6. Foreach loop
   6.1. sorting hashes and appending into arr
7. Length_of_block == length of arr
8. While loop, if length is odd, copy last element in list
   8.1. append it into arr list
9. Set list “ another_arr ”
10. Foreach loop
    10.1. For loop with range from 0 to length of “ arr “ and iteration by 2
        10.1.1. Define variable with “ sha512() “ value
        10.1.2. Hash elements that are in “ arr “ list
        10.1.3. Apennd them into new “ another_arr “ list
        10.1.4. Return this list in hex
11. Set list “ hash_arr ”
12. Foreach loop
    12.1. Generate Hex and put it into “ hash_arr ” list
13. Create message put it in “ st “ variable
14. Convert “ st “ value in binary
15. First_secret_key = hash_arr[0]
16. Second_secret_key = hash_arr[1]

17. Generate “ one-time signature ”
    17.1. If st == 0
        17.1.1. Choose “ First_secret_key “ bit
    17.2. Else
        17.2.1. Choose “Second _secret_key “ bit
18. First_pub_key = hash(hash_arr[0])
19. Second_pub_key = hash (hash_arr[1])
20. Encryption
    20.1. Concatenate “ one-time signature ” with message’s hash
21. Verification of “ one-time signature ”
    21.1. If bit of “ one-time signature ” == 0
        21.1.1. Compare with “ First_secret_key “ bit
    21.2. Else
        21.2.1. Compare with “Second _secret_key “ bit
22. Verification of “ signature ”
    22.1. Concatenate siblings with each other
    22.2. If this equals to public key
        22.2.1. Sign is correct
    22.3. Else
        22.3.1. Sign is not correct

В данном примере показана реализация алгоритма „Меркле“ с помощью рекурсии. Время подсчета “public key” для 8 элементов составляет 0.0159 секунд, время шифрования - 0.01684, время подтверждения - 0.0288883 .

Мы изменили рекурсию на циклы для улучшения эффективности.

Реализация с помощью циклов:

1. Importing necessary libs
2. Define class
3. Defining “ loop_hashes(hashes) “ method
4. Set list “ arr ”
5. If hashes == “ ”, raise Exception
6. Foreach loop
    6.1. sorting hashes and appending into arr
7. Length_of_block == length of arr

8. While loop, if length is odd, copy last element in list
    8.1. append it into arr list
9. Set list “ another_arr ”
10. Set i = 0
11. While loop, Length_of_block > 1
    11.1. Set to hash_f sha512()
    11.2. Concatenate arr[i] and arr[i+1]
    11.3. append it into “another_arr” list
    11.4. append arr[i+1] to “auth_list” list
    11.5. i = i + 2
    11.6. if i equal to “Length_of_block”
        11.6.1. set to “Length_of_block” “Length_of_block / 2”
        11.6.2. i = 0
        11.6.3. set “another_arr” to “arr”
        11.6.4. empty “another_arr”
    11.7. return “arr”
12. Set list “ hash_arr ”
13. Foreach loop
    13.1. Generate Hex and put it into “ hash_arr ” list
14. Create message put it in “ st “ variable
15. Convert “ st “ value in binary
16. First_secret_key = hash_arr[0]
17. Second_secret_key = hash_arr[1]
18. Generate “ one-time signature ”
    18.1. If st == 0
        18.1.1. Choose “ First_secret_key “ bit
    18.2. Else
        18.2.1. Choose “Second _secret_key “ bit
19. First_pub_key = hash(hash_arr[0])
20. Second_pub_key = hash (hash_arr[1])
21. Encryption
    21.1. Concatenate “ one-time signature ” with message’s hash
22. Verification of “ one-time signature ”
    22.1. If bit of “ one-time signature ” == 0

22.1.1. Compare with “ First_secret_key “ bit

22.2. Else

22.2.1. Compare with “Second _secret_key “ bit

23. Verification of “ signature ”

23.1. Concatenate siblings with each other

23.2. If this equals to public key

23.2.1. Sign is correct

23.3. Else

23.3.1. Sign is not correct

В данном примере показана реализация алгоритма „Меркле“ с помощью цикла. Время подсчета “public key” для 8 элементов составляет 0.0061761 секунд, время шифрования - 0.0080878, время подтверждения - 0.0181923. Как мы видим изменения реализации дало довольно хорошие результаты.

БИБЛИОГРАФИЯ:

1. Guang Hao Low, Artur Scherer, and Dominic W. Berry , Black-Box Quantum State Preparation without Arithmetic Yuval R. Sanders, Phys. Rev. Lett. 122, 020502 – Published 16 January 2019
2. Liu J. et al. (2019) Formal Verification of Quantum Algorithms Using Quantum Hoare Logic. In: Dillig I., Tasiran S. (eds) Computer Aided Verification. CAV 2019. Lecture Notes in Computer Science, vol 11562. Springer, Cham