UDC 004 056 55

O. KACHKO, Ph.D., D. TELEVNYI

THE KUPYNA HASH FUNCTION CRYPTANALYSIS WITH MERKLE TRESS SIGNATURE SCHEMES

Introduction

In the modern world, digital signatures (DSAs) have become a crucial element in any cryptographic system. Their usage is not limited only by enterprise or banking systems. The application happens to be huger than thought. The «MUST HAVE» feature of every modern CRM or ERP system. The most widely used systems are based on the asymmetric pair cryptography.

With the development of quantum computing a new problem appears for existing signatures. Some are based on the asymmetric transformation, mostly in GF or EC. Thus, quantum algorithms can solve Discrete logarithm tasks or factorization in seemingly short time and memory which makes existing schemes vulnerable. To gain enough strength either a key size must be increased, or a signature run timing, that would result to insufficiency of the signatures.

Since late 70’s other schemes were developed. One of them is hash-based signatures. But the machine capabilities did not allow to use them rather than RSA or DSAs.

Modern hardware lacks such problems, as well as new algorithms were developed. The signature schemes can be divided into OTS (Lamport, Winternitz, etc.) and FTS (Merkle trees, etc.).

Since the large scaling of systems, the later are more preferred. The main goal of this paper is to analyze the security of Merkle Tree Signature Schemes and the national standard application to it.

The Kupyna hash function

Ukraine had used the GOST 34.311-95 [1] has function before it was replaced in 2015 by DSTU 7564:2014 [2]. According to authors, the new hash function is based on common, well-known and reliable constructs. [3].

The construction concept features the Even-Mansour scheme with Davies-Mayer compress function and inner permutation block from Kalyna (DSTU 7664:2014) [2].

The hash function supports several modes, defined as Kupyna-n and the set of

$$n \in \{8s \mid s = 1,2,..64\}. \tag{1}$$

The recommended modes are Kupyna-256, 384, 512.

The hash function pads the input message into $m_1, \ldots, m_n$ parts of l bits (512, 1024). The computation consists of the compression function, which iteratively updates the previous block-hash, and the reduction function to form the output.

$$\begin{gathered} h_0 = IV \\ h_i = f\left(h_{i-1}, m_i\right), i = 1,..t \\ h = \Omega\left(h_t\right) \end{gathered} \tag{2}$$

The compression function is the part which can be attacked. The standard [dstu] defines it as

$$f\left(h_{i-1}, m_i\right) = T^{xor}\left(h_{i-1} xor m_i\right) xor T^{+}\left(m_i\right) xor h_{i-1} \tag{3}$$

The reduction function takes the n most significant bits from the sum by modulo 2 of the permuted last block hashes with non-processed one.

$$\Omega\left(h_t\right) = trunc_n\left(T^{xor}\left(h_t\right) xor h_t\right) \tag{4}$$

The latest step may be attacked by the collision search of a pair:

$$h = \Omega\left(h'_t\right)$$
$$h = \Omega\left(h_t\right) \tag{5}$$

Where both n-most significant bits of the value are the same despite the hash values being different.

It the current section we focus on the permutation steps security analysis. The construction of both T-permutations is similar with Grostl hash.

In the «Analysis of the Kupyna-256 Hash Function» [4] paper authors performed cryptanalysis on the Kupyna-256 permutation function.

They describe collision attacks on the round-reduced hash up to 5 rounds and collisions to the compression function up to 7 rounds.

The compression function attack included semi-free-start collisions and based on the rebound attack on Grøstl using SuperBox matching. [5,6]

Considering T^+ permutation has round constant adding with modulo 2^{64}, the paper provides modified attack rather than on Grostl.

The attack presumes finding the pairs of input value for T^+ and T^{xor}.

The results of rebound attacks are listed in table 1.

Table 1
Overview of collision attack on the compression function

rounds	Time complexity	Memory complexity
6	2^{70}	2^{70}
7	2^{125}	2^{70}

The collision attacks were performed on the Kupyna-256.

The attack on the reduced hash is a straight-forward rebound attack on the reduced Grostl-256. The idea of attack is representation of the hash function with permutations without the left-multiplication with 8X8 MDS matrix over GF 2^8 (so-called MixBytes step) and defining the MixBytes^{-1} inverse transformation. The hash function now has the following look (6).

$$\hat{h}_0 = MB^{-1}\left(IV\right)$$
$$\hat{h}_i = \hat{T}^{xor}\left(MB\left(\hat{h}_{i-1}\right) xor m_i\right) xor\, \hat{T}^{+}\left(m_i\right) xor\, \hat{h}_{i-1},$$
$$i = 1,..,t \tag{6}$$
$$h = \Omega\left(MB\left(\hat{h}_t\right)\right)$$

The results for collision attacks on the reduced Kupina-256 listed in table 2.

Table 2
Overview of collision attack on the reduced hash function

rounds	Time complexity	Memory complexity
4	2^{67}	2^{59}
5	2^{120}	2^{59}

Merkle Signature Scheme

One of the most notable cons of OTS (one-time signatures) is the key management. The cryptsystem must guarantee the identity of the used key and its consistency. Few public keys are to be used and their length should be rather short. To make such schemes feasible, an efficient key management system must be used to reduce the number of keys and their size.

Ralph Merkle introduced in his research paper a new signature scheme for signing many messages with one key. [7] The following scheme is based on the tree structure where all steps of signature process can be observed as a tree traverse. In fig. 1 a such tree is listed.

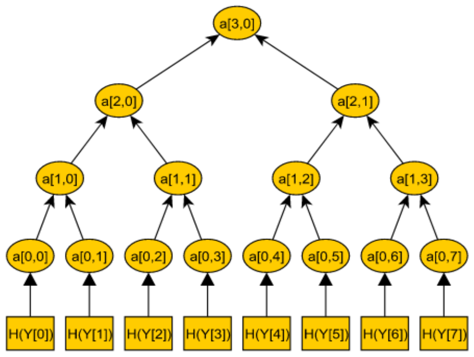

Figure 1. Merkle tree scheme

The big advantage of Merkle Signature Scheme is the fact that many messages are signed with a short number of keys. But the cost of such efficiency is inadequate. To generate a public key *pub*, 2^n OTS must be generated. If tree contains 2^{n+1} -1 nodes, the same number of hash function applying must be performed to get a public key. It's obvious that the size of the tree is limited with available memory and runtime complexity.

The signature generation requires *auth* nodes to be computed. To reduce of such inter-state nodes, some cache techniques must be applied. This increases the storage requirements.

Nevertheless, the verification time is quite fast comparing to the other steps.

The last researches were headed to build an algorithm to reduce the storage size and compute in efficient time. This strategy was called Merkel Tree traversal.

The signature of the Merkle Signature Scheme consists of the ots *sig'* and *n* nodes $auth_0, \ldots, auth_{n-1}$. If a 256-bit hash function is used, the signature size would be $|sig|=|sig'|+n*256$ bits.

To efficiently compute a node in tree a tree-hash algorithm is required. The main idea of it is calculating the needed sub-tree from left to right only saving the needed nodes.

The classic traversal computes at O(*2(H-1)*) complexity and O(H*(H+1)) memory. But in [8] a logarithm time and space algorithm was introduced by M.Szydlo. The main idea is reducing the active tree-hash instances. The presented algorithm stores *3log(N)* and computed in *2log(N)*, where N – number of available signatures.

The further development brought the idea of storing such trees and computing them in hypertrees – trees with sub-tree nodes of the same height. This approach is a fractal presentation of traversal.

A good value for h, in which the space requirements are minimal, would be $h = logH = loglogN$. Using this parameter would result in a time and space bound of $sig_{time} = 2logN/loglogN$ and $sig_{space} = 5/2log2N/loglogN$.

In [9] improvements were proposed: using PRNG with seeds for private keys generation. This idea supposes storing only seeds, relatively smaller to the private keys themselves.

The other idea is using many Merkle trees rather than a big one. The [9] lists the results of such approach to sign a message on Pentium dual core 1.8GHz. The following result is shown in table 3 below.

Table 3

Timing and memory results

signatures	Mem_{upd}	mem_{out}	mem_{sign}	t_{kgen}	t_{sign}	t_{verify}
2^{40}	3160 bytes	1640, bytes	1860 bytes	723 m	26.1 ms	19.6 ms
2^{40}	3200 bytes	1680, bytes	2340 bytes	390 m	10.7 ms	10.6 ms
2^{80}	7320 bytes	4320 bytes	3620 bytes	1063 m	26.0 ms	18.1 ms
2^{80}	7500 bytes	4500 bytes	4240 bytes	592 m	10.1 ms	10.1 ms

Merkle signature scheme cryptanalysis

The given section describes the security of the Merkle Signature Scheme. If an attacker has message *m* and *sig* and wants to counterfeit a signature of the *m`*, he has two cases.

The first presumes that attacker finds a valid sig'_a with a public key Y_i` and $H(Y_i`) = H(Y_i) = A_0$. The attacker can achieve this by finding a valid OTS of the message *m`* with equal public keys, i.e $Y_i` = Y_i$. When found, this would mean that OTS is broken. Therefore, breaking OTS means the Merkle Signature Scheme also breaks. If this cannot be achieved, then the attack of a second preimage can be performed, i.e. finding sig'_a $H(Y_i`) = H(Y_i), Y_i` \neq Y_{i.}$

Thus, the Merkle Signature Scheme is secure if the hash function in OTS is second preimage resistant. As mentioned in section 1, the Kupyna hash has such characteristics. The possible attack may occur only in case of a weak public key.

The other option is generation a valid sig'_a with a public key Y_i` and $A`_i = H(Y_i`) \neq H(Y_i) = A_{0.}$ In this case an auth path must be changed $A'n=An=pub$ with $A'i=H(a'i-1||auth'i-1)$ $i=1, \ldots, n$ to make a valid signature. If the attacker finds a single $auth'_i$ so that $H(A'_i||auth'_i) = A_{i+1}=A_{i+1,}$ then a valid auth. path is found.

Hence to counterfeit an attack $auth'_i$ must be found, so that $H(A'_i||auth'_i) = H(A_i||auth_i)$. If the hash function in OTS is not second image resistant, then the attack is possible.

So, the Merkle Tree is secure when OTS is secure and the hash function is second image resistant.

The cryptographic hash function is secure when is second preimage and collision resistant. But the Merkle Tree Scheme does not require it to be collision resistant. Thus, we can reduce the bit level of security of second preimage resistant hash function in this scheme. Thus, more lightweight hash functions can be used.

Conslusions

The modern national standard defines the modes of Kupyna-n hash function. Kupyna-256, 384, 512 modes have second preimage attack resistance and collisions resistance. Due to having the structure like Grøstl, the round attack to reduced hash can be performed. The listed above modes are strong after the 7 rounds.

The Merkle trees were known since 80s, but improvements were published past few years.

Thus, making Merkle Signature Schemes an alternative to conventional existing schemes. The message can be signed with reasonable time and one public key can be used for 2^{80} signatures.

These schemes may utilize hash functions lighter than the existing standards with lesser bit-security level.

Since Kupyna-n has varying security levels, the lesser level function can be used in OTS in Merkle Signature Schemes due to the property of Merkle trees to be collision resistant.

References

1. Metrology and Certification of the Commonwealth of Independence States. GOST 34.311-95. Information technology. Cryptographic Data Security. Hash function. Metrology and Certification of the Commonwealth of Independence States. Minsk, 1995. (In Rus).

2. Roman Oliynykov, Ivan Gorbenko, Oleksandr Kazymyrov, Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov, Ruslan Mordvinov, Dmytro Kaidalov. A new encryption standard of Ukraine: The Kalyna block cipher. Cryptology ePrint Archive. Report 2015/650, 2015. http://eprint.iacr.org/2015/650.pdf

3. Roman Oliynykov, Ivan Gorbenko, Oleksandr Kazymyrov, Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Artem Boiko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov. A New Standard of Ukraine: The Kupyna Hash Function. Cryptology ePrint Archive. Report 2015/885, 2015. https://eprint.iacr.org/2015/885.pdf

4. Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Analysis of the Kupyna-256 Hash Function, Graz University of Technology, Austria, Cryptology ePrint Archive. Report 2015/956, 2015. https://eprint.iacr.org/2015/956.pdf

5. Mendel F., Rechberger C., Schl affer M., Thomsen S.S.: Rebound attacks on the reduced Grøstl hash function. In: Pieprzyk, J. (ed.) Topics in Cryptology – CT-RSA 2010. LNCS. – Vol. 5985. – P. 350–365. Springer (2010)

6. Jean J., Naya-Plasencia M., Peyrin T. Improved rebound attack on the finalist Grøstl. In: Canteaut, A. (ed.) Fast Software Encryption – FSE 2012. LNCS. – Vol. 7549. – P. 110–126. Springer (2012)

7. Ralph Merkle. Secrecy, authentication and public key systems // A certified digital signature.Ph.D. dissertation, Dept. of Electrical Engineering, Stanford University, 1979.

8. Michael Szydlo. Merkle tree traversal in log space and time. Eurocrypt, 2004.

9. Klintsevich K., Okeya, Vuillaume C., Buchmann J., Dahmen E. Merkle signatures with virtually unlimited signature capacity. 5th International Conference on Applied Cryptography and Network Security. – ACNS07, 2007.

V.N. Karazin Kharkiv National University *Received 05.09.2018*