Ingeniería Investigación y Tecnología
volumen XX (número 2), abril-junio 2019 1-10
ISSN 2594-0732 FI-UNAM artículo arbitrado
Información del artículo: Recibido: 20 de junio de 2017, reevaluado: 21 y 23 de junio de 2017 y 24 de agosto de 2018, aceptado: 7 de enero de 2019

http://dx.doi.org/10.22201/fi.25940732e.2019.20n2.018

Public hash signature for mobile network devices
Firma electrónica por medio de funciones hash para dispositivos móviles

Lizama-Pérez Luis Adrián
Universidad Politécnica de Pachuca
Departamento de Posgrado
E-mail: luislizama@upp.edu.mx
http://orcid.org/0000-0001-5109-2927

Montiel-Arrieta Leonardo Javier
Universidad Politécnica de Pachuca
Departamento de Posgrado
E-mail: leo8677@hotmail.com
https://orcid.org/0000-0002-9259-535X

Hernández-Mendoza Flor Seleyda
Universidad Politécnica de Pachuca
Departamento de Posgrado
E-mail: flors.hm@gmail.com
https://orcid.org/0000-0002-3110-5599

Lizama-Servín Luis Adrián
Centro de Ingeniería y Desarrollo Industrial, Conacyt
Departamento de Posgrado
E-mail: luis.lizama@alumni.fh-aachen.de
https://orcid.org/0000-0002-4035-1361

Simancas-Acevedo Eric
Universidad Politécnica de Pachuca
Departamento de Posgrado
E-mail: ericsimancas@upp.edu.mx
https://orcid.org/0000-0001-7823-709X

Abstract

In this work we have developed a digital signature protocol using hash functions that once implemented on mobile devices have demonstrated to be secure and efficient. It has been incorporated a model for a Certification Authority to exchange public keys between users. This work constitutes an experimental research, which bears a certain resemblance to theoretical research, but is not intended to propose a new theory, but to establish the behavior of a system to know its characteristics, in order to improve its knowledge and/or its performance. The hash signature system was tested on mobile communication devices. The experimental results show that the hash signature improves the efficiency to generate the cryptographic keys and the signing and verification processes when compared to ECC. Likewise, when generating 2048 keys, the hash signature is faster than RSA. In addition, the larger RSA keys consume a significative time, while the hash does not require to increase the size of the keys. Although we have not included here a formal analysis about the protocol, we highlight some points that improve the security of the proposed protocol. Finally, this work constitutes a new approach to public key cryptography based on hash functions that could be used to make digital signatures in electronic commerce. This method is suitable for mobile network devices due to the high speed and low hardware requirements of the hash functions. The method described here, which is compatible with hash functions, belongs to the field of post-quantum cryptography. The security of the method is based on the security of the hash cryptography, which is widely known and discussed.
Keywords: Hash function, hash chain, Merkle tree, digital signature, mobile devices.

Resumen

En este trabajo se ha desarrollado un protocolo de firma digital utilizando funciones hash que demuestra que es seguro, efciente y adecuado para operar en dispositivos móviles. Además, se desarrolló el modelo de una Autoridad Certificadora para el intercambio seguro y efciente de llaves públicas. El método que se utilizó fue la investigación experimental, que tiene cierta semejanza con la investigación teórica, pero no tiene por objeto plantear una teoría nueva, sino establecer el comportamiento de un sistema para conocer sus características, a fin de mejorar su conocimiento y su rendimiento. El sistema de firma hash fue implementado en dispositivos móviles. Los resultados experimentales demuestran que la firma hash mejora la eficiencia para generar las claves y los procesos de firma y verificación cuando se compara a ECC. Asimismo, al generar 2048 claves la firma hash es más rápida que RSA. Además, las claves RSA más grandes consumen mucho tiempo, mientras que el sistema hash no requiere aumentar el tamaño de las claves. Aunque no se ha incluido un análisis formal sobre la seguridad del protocolo, se destacan algunos puntos que mejoran la seguridad del protocolo propuesto. Finalmente, este trabajo constituye un nuevo enfoque para la criptografía de llave pública basada en funciones hash que podrían aprovecharse para realizar firmas digitales en comercio electrónico. Este método es adecuado para dispositivos de red móvil debido a la velocidad y los requisitos de hardware de las funciones hash. El método descrito aquí, el cual es compatible con las funciones hash, pertenece al campo de la criptografía post-cuántica. La seguridad del método se basa en la seguridad de la criptografía hash, la cual es ampliamente conocida y discutida.
Descriptores: Función hash, cadena hash, Árbol de Merkle, firma digital, dispositivos móviles.

http://dx.doi.org/10.22201/fi.25940732e.2019.20n2.018

Introduction

Today, cryptographic hash functions constitute a main component of a variety of authentication protocols to achieve data integrity and non-repudiation services. Hash cryptography has been implemented throughout a variety of algorithms which include the Message Digest Algorithm (MD5 (Rivest, 1992)), the Secure Hash Algorithm SHA-1(Eastlake 3rd and Jones, 2001), SHA-2 (P. U. B. FIPS, 2002) and SHA-3 (DRAFT, 2014). One of the major advantages of hash cryptography is that it is not based on modular mathematics, so it does not require large prime computations and it is suitable for small mobile computing devices. Hash cryptography is resistant to quantum-crypto-analysis (Perlner and Cooper, 2009), which is not the case of modular arithmetic that is based on the Integer Factoring or the Discrete Logarithm Problem (Nielsen and Chuang, 2010; Dattani and Bryans, 2014; Dridi and Alghassi, 2016) due to Shor's factoring algorithm for quantum computation (Shor, 1994). Moreover, special purpose devices run hash chains in the order of 300 peta hashes per second (Zohar, 2015). Hash based signatures run at least 104 faster than modular-arithmetic based methods (Muñoz *et al*., 2004).

Since Leslie Lamport introduced hash chains (Lamport, 1979; Lamport, 1981c; Hu, *et al*., 2005), some technologies implement this algorithm as the S/Key One Time Password (OTP) (Haller, 1995) but few attempts have been made to get authentication protocols based on hash chains (Perrig et al., 2005; Anderson *et al*., 1998). Other protocols have been developed under hash functionality as Time-based One-Time Password (RFC (M'Raihi *et al*., 2011)) or HMAC-based One-Time Password (RFC (M'Raihi *et al*.,2005)).

An attempt to achieve a public key cryptosystem by means of hash cryptography is the One Time Signature (OTS) method developed by Leslie Lamport (1979). However, in this protocol the message signature process can be executed only once. OTS can be optimized through some methods (Merkle, 1982; 1987; 1989; Buchmann *et al*., 2009). Using a Merkle tree, it is possible to increase the number of messages that can be signed, however one pair of public/private OTS key must be generated by each leaf and they must be stored in the user end device for future message signing.

On the other side, modular-arithmetic signature schemes include the Digital Signature Algorithm proposed by the US National Institute of Standards and Technology (NIST) (N. FIPS, 1998; Fips, 2000; P. U. B. FIPS, 2009), the RSA algorithm (Rivest *et al*., 1978) and the Probabilistic Signature Scheme which is part of the standard (P. U. B. FIPS, 2009) and the Elliptic Curve Digital Signature Algorithm (ECDSA) (Triwinarko, 2006) which is also included in (P.U.B. FIPS, 2009).

Such algorithms are based on modular arithmetic computations, which requires in some cases large prime numbers and in other cases modular multiplication and/or exponentiation. Moreover, the generation and preparation of the pair of public/private key is computationally costly. In contrast, hash technology is advantageous for potential use in mobile and low-performance hardware devices.

Since there is no quantum algorithm to efficiently analyze hash functions (Perlner and Cooper, 2009), it is assumed that cryptography based on hash functions, as Merkle trees, are post-quantum (Merkle, 1982; Buchmann, 2016). However, for exhaustive search, Grover's quantum algorithm reduce the search space approximately to . This leads to a recommended hash length of 256 for (Buchmann *et al*., 2016).

According to (Buchmann *et al*., 2016) "there are no hash-based public-key encryption schemes." However, multivariate signature schemes exploit hash functions to get its advantages. As a result, multivariate cryptography produce signatures speedily and the verification process is time reduced. Moreover the signatures are shorter than RSA. Unfortunately, the key sizes of signatures are still relatively large (Buchmann *et al*., 2016).

In this work we will introduce a signature scheme based just on hash functions where the key sizes varies from 256 to 512 bits. The method is the first known for public key infrastructure based just on hash functions and it can be used to achieve Digital Signature services. Moreover, it allows the Certification Authority functionality.

We should remark that hash cryptography is resistant to quantum-crypto-analysis (Perlner and Cooper, 2009), which is not the case of modular arithmetic that is based on the Integer Factoring or the Discrete Logarithm Problem (Nielsen and Chuang, 2010; Dattani and Bryans, 2014; Dridi and Alghassi, 2016) due to Shor's factoring algorithm for quantum computation (Shor, 1994).

The hash signature algorithm

In this section, we will detail our proposal called the hash chained signature protocol. However, let us introduce some previous concepts of hash functions and hash chains.

Hash function

A hash function takes as input a message which is a string of bits of arbitrary length, and it produces as an output a string of bits of fixed length determined by the hash function. The output is called hash code. The hash function is expressed as $h = f(X)$ where X corresponds to the input or message (Nigel, 2016).

Hash functions are characterized by two main properties:

1. From the computational point of view, it is infeasible to get the message X if known h.
2. It is infeasible to find two different messages with the same hash value.

On the other hand, a hash chain is a sequence of values derived consecutively from a hash function from an initial value (Figure 1). Due to the properties of the hash function, it is relatively easy to calculate successively chained values (Dwight, 2011).

The hash chained signature protocol

In our protocol, Alice wants to sign a message and send it to Bob. Alice and Bob must execute the next steps:

1. Alice and Bob generate their pair of public-private keys.
2. They exchange their public keys through the public channel.
3. Alice signs a message using her private key.
4. Bob verifies the signature using Alice's public key.

To generate the keys the following cryptographic functions are necessary:

1. A True (or Pseudo) Random Generator.
2. A cryptographic hash function.

To produce the keys Alice (or Bob) generates a small random number (e.g. 128, 160, 256, 512 bits length). We call this number the sieve X_A (or X_B). Now, using the hash function Alice (or Bob) computes the hash chain $f^{N_A}(X_A))$ where f denotes the hash function of the system. The exponent N_A represents the times the function is applied over the sieve X_A. The public key is $f^{N_A}(X_A))$ whose size corresponds to the hash length. On the other side, Alice's secret values (X_A, N_A) yield her private keys $f^{NA-i}(X_A)$ where $1 \leq i < N_A$. Thus, $f^{NA-1}(X_A), f^{NA-1}(X_A), \ldots, f^1(X_A)$ are the Alice's private keys. We represent the keys in Table 1.

Table 1. The keys of the Hash Chained Signature protocol, here $1 \leq i(j) < N_A(N_B)$

User	Public Key	Private Key
Alice	$f^{NA}(X_A)$	$f^{NA-i}(X_A)$
Bob	$f^{NB}(X_\beta)$	$f^{NB-i}(X_B)$

To sign a message, it will be used the Hash Message Authentication Code (HMAC) function (Krawczyk *et al.*, 1997). Suppose Alice wants to sign a message and send it to Bob, then Alice and Bob perform the following steps:

1. Using a public channel, Alice and Bob exchange their public keys $f^{N_A}(X_A)$ and $f^{N_B}(X_B)$.
2. Applying the HMAC function, Alice signs the message m using her private key f^{N_A-1}.
3. Bob sends his private key f^{N_B-1} to Alice. Then Alice verifies Bob's authenticator because she computes) $f(f^{N_B-1})$ which must return Bob's public key f^{N_B}.
4. Only in the case that authentication is verified successfully Alice sends to Bob the message m and her private key f^{N_A-1}.
5. Finally, Bob performs the HMAC computation of the message denoted as $< <m> f^{N_A-1}$ with f^{N_A-1} to check the message integrity. In the favorable case, Bob can be sure that Alice wrote the message.

Table 2 depicts the protocol where the <> symbol represents the HMAC function. Some significant features of this hash based signature protocol are:

1. Up to our knowledge, this is the first protocol that uses hash chains for signing message with a private key. The private key consists of a chain of private keys of short duration. The seed used to compute the chain, as well as the private keys remain secret until the key is used for a transaction. In such a case, the private key is no more secret and becomes public. This process continues until the chain of keys is renewed.

X	$f^1(X)$	$f^2(X)$	...	$f^{N-j}(X)$	$f^N(X)$

Figure 1. Simple representation of a hash chain computation

http://dx.doi.org/10.22201/fi.25940732e.2019.20n2.018

2. Unlike conventional public key schemes, here the public and private keys change at each transaction. That is, the private key is substituted each time. This is needed in order to authenticate the user, whose private key is validated by applying the hash function to the key, the result must be the public key.
3. The protocol is interactive and gives mutual authentication between Alice and Bob. So, it is useful for one-to-one communication because the destination authenticates the signed message.
4. It is dynamic because every time a private key is revealed it becomes part of the public key. So, verification of the public key can be achieved using the current public key or any key in the public hash chain.

Table 2. The basic Hash Chained Signature protocol

$A \rightarrow B$:	$<m>^{f N_A-1}$
$A \leftarrow B$:	f^{N_B-1}
$A \rightarrow B$:	m, f^{N_A-1}

Security analysis

However, Alice and Bob do not have a way to verify the authentication keys, f^{NA-1} and f^{NB-1}. The basic Hash Chained Signature protocol is vulnerable to a Man In The Middle (MITM) attack because the attacker implements a replay attack using the keys that she has obtained previously. So, the attacker can impersonate Alice's identity (Table 3 where *E* denotes the eavesdropper). A countermeasure to this drawback can be conceived adding time stamps to the protocol but some flaws can still be present.

Table 3. MITM attack

$E \rightarrow B$:	$<x>f^{N_E-1}$
$E \leftarrow B$:	f^{N_B-1}
$E \rightarrow B$:	x, f^{N_E-1}
$A \rightarrow E$:	$<m>f^{N_A-1}$
$A \leftarrow E$:	f^{N_B-1}
$A \rightarrow E$:	m, f^{N_A-1}
$E \rightarrow B$:	$<m'>f^{N_A-1}$

Despite the MITM attack described before, the basic Hash Chained Signature protocol is still usable for dedicated point-to-point links where as stated before, each new key can be verified from the previous one, e.g. $f^{NB-i+1} = f(f^{NB-i})$ Once a private key is revealed it cannot be reused to sign another message. Since Alice and Bob sign messages such revealing a private key, over time private keys consume and it will be necessary a method to renew the keys. We discuss this issue later. Let us write the protocol symbolically, for this assume that i and j are the indexes of the keys from Alice and Bob, respectively, they must exchange the index data, see Table 4. The index i (j) increases by one each time a signature is performed. To verify Bob's authenticator Alice computes the hash of the Bob's authenticator f^{NB-j} which returns Bob's public key, that is $f^{j}(f^{NB-j}) = f^{NB}$

Table 4. Over time the private keys consume. A private key that is revealed becomes part of the public key

$A \rightarrow B$:	$<m>_{f N_A-i}$
$A \leftarrow B$:	f^{N_B-j}
$A \rightarrow B$:	m, f^{N_A-i}, i

Certification authority

To surpass such MITM attack we will introduce the Certification Authority (CA) to the protocol. But before discussing the CA, we must mention two protocols that share some similarities with the hash signature system:

1. The Delayed Authentication Message (DEMA) Protocol presented in (Groza, 2006) is in fact equivalent to the Hash Chained Signature protocol discussed here. However, DEMA protocol is aimed for point to point synchronized connections. In contrast, we preserve the basic structure of the protocol but we achieve signature services through the Certification Authority.
2. In (Buldas *et al*., 2014b) is described a signature scheme, in which the client computes a Merkle tree constructed with a hash chain. From the Merkle tree the user generates his public key. In addition, the client emits a time stamp to the signature server to determine the interval the certificate will be valid. The server generates a hash tree from the time stamp to be used with the client. In this scheme, the authors identify a problem: many keys are not used because they are only valid for a certain period of time. In (Buldas *et al*., 2014a) they propose a new scheme where the main idea is to use the signature message as a public key revocation. The message includes the period that the key will be valid. Another difference is that the client must store all the created signatures in case of later disputes.

The Certification Authority (CA) will be responsible to maintain the Hash Chained Signature system, registering new users and exchanging signatures. It must be remarked that the CA does not store the secret keys of users, as typically achieved with a Key Distribution Center (KDC). By contrary, each private key is kept by its owner and never leaves the user side. The CA just act forwarding the message signatures between users. For this purpose, the CA prepares a public key to communicate with each user, for example in the case of Alice the CA generates the public key $f^{NAC,A}$. As a result, the CA and Alice maintain a control of the index key so, the index x increases unitary, $x = 1,2, \ldots$ each time they communicate (Table 5).

Table 5. Alice (or Bob) and the CA maintain an ordered public key exchange

$A \rightarrow CA$:	$<m>_f^{N_A - x}$
$A \leftarrow CA$:	$f^{N_{AC,A}-x}$
$A \rightarrow CA$:	m, f^{N_A-x}

In this way, the CA acts as a gateway between Alice and Bob to exchange their signatures each time they communicate. To guarantee non-repudiation Alice sends the hash of the message she wants to sign. The CA performs message accounting by storing message signatures (she does not store messages themselves). The complete protocol has been specified in Table 6. We have included an authentication key of the user, f^{NA-1} to avoid fake requests to the CA. So, the CA verifies authenticity before giving any response.

The CA does not have access to private keys of the users, however she must be in the middle each time a pair of users want to communicate securely. Therefore, the computational cost of the CA process would be a concern for this protocol. However, it is common that e-transactions in Internet based protocols are verified by a Third Trusted Party (3TP).

The next step is to develop a method to efficiently manage the keys of the users. We discuss this question in the following section.

Table 6. Signature exchange where $s_0 = f(m)$, $s_1 = < s_0 >_f N_A^{-2}$ and $s_2 = < s_1 >_f N_{CA,B}^{-1}>$

$A \rightarrow CA$:	A, B, f^{N_A-1}, s_1
$AC \rightarrow B$:	A, s_2
$AC \leftarrow B$:	f^{N_B-1}
$A \leftarrow CA$:	$f^{N_{CA,A}-1}$
$A \rightarrow CA$:	f^{N_A-2}, s_0
$AC \rightarrow B$:	$f^{N_A-2}, f^{N_{CA,B}-1}, s_1$
$A \rightarrow B$:	m

Merkle tree chain

The CA must prepare a public key to communicate with each user. To accomplish this task the CA implements a Merkle Tree (Merkle, 1982) whose leafs are deserved to allocate the pairs of public keys between the CA and each user. The Merkle Tree will be publicly announced and the CA identity will be verified through the computation of the root of the tree.

To validate the identity of the CA each user will receive the required Hash nodes to compute the Hash root. However, a pair of leaves of the Merkle Tree will not be used neither revealed, instead of that they will be kept secret so that the last leaf will be deserved to bind the current tree with the next tree of the tree chain. Therefore, a pair of leaves of each new tree that is concatenated to the tree chain is kept secret, Figure 1 shows this concept.

In the example of Figure 2, the $H_{1.48}$ leaf in the first tree is the same leaf identified as $H_{2.41}$ in the second tree but $H_{1.47}$ and $H_{1.48}$ will remain secret until the tree of root $H_{2.11}$ is publicly announced by the AC. The next tree is computed and publicly announced by the CA once the first tree reaches half of its total capacity. In the same way, $H_{2.47}$ and $H_{2.48}$ will be kept secret until the new tree is advertised. In this way, the CA can register new users to the system and each user can validate the CA identity. The CA identity is verified through the Hash roots $H_{i,11}$, where $i = 1.2, \ldots$

As time goes, multiple trees are created and the CA stops creating new trees but she proceeds to assign the

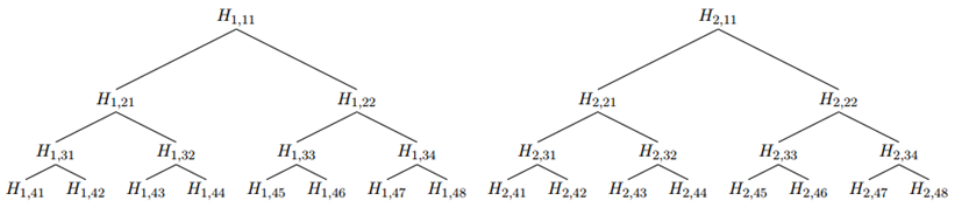

Figure 2. A Merkle tree chain

http://dx.doi.org/10.22201/fi.25940732e.2019.20n2.018

unused leaves. For this task, the CA enumerates again the unused leaves and the algorithm is executed again. A higher order level can be appended to the tree merging the roots $H_{1,11}$ and $H_{2,11}$ into a new root, such that the Merkle tree algorithm keeps running efficiently.

Renewing the certificate

To update her public key Alice computes a new pair of public/private keys and she signs the new public key with an older private key. Then, the CA advertises the key update. Since the new key is signed by Alice and it is published by the CA the renewing process does not compromises the security of the protocol. The steps of the process are represented in Table 7.

Table 7. Renewing a public key. In this example Alice's current private key is f^{N_A-i} while the corresponding CA's private key is $f^{N_{AC,A}-j}$

$A \rightarrow CA$:	$f^{N_A-i}, <f^{N'_A}(X_A)>_f^{N_A-i-1}$
$A \leftarrow CA$:	$f^{N_{CA,A}-j}, <f^{N'_{CA,A}}(X'_{CA,A})>_f^{N_{CA,A}-j-1}$
$A \rightarrow CA$:	$f^{N'_A}(X'_A), f^{N_A-i-1}$
$A \leftarrow CA$:	$f^{N'_{CA,A}}(X'_{CA,A}) f^{N_{CA,A}-j-1}$

Multiple CA's

Multiple CA's can contribute to decrease the computational effort of a single CA. To allow interoperability between multiple CA's each CA must register with the other CA, this process is represented in Table 8. Suppose for example, that $H_{1,44}$ in Figure 1 has been registered with CA_2. Thus, to interconnect the two CA domains it will be necessary that CAs register mutually.

Table 8 shows the protocol when users are registered to different CA's. This scheme can be generalized to multiple CA's. However, we leave for an ulterior work the implementation of multiple CA's and the analysis of interoperability with multiple domains.

Experimental results

In this section, we describe the results obtained with the Hash Signature system applying different sizes of the message. The CA was developed on NetBeans 8.1 in a computer desktop equipped with Pentium Dual-Core, 4 GB RAM. The client application was built over Android Studio 2.3. Tests were performed on two different mobile phones. They are detailed in Table 9.

We generated 1024-leaf trees in 5 milliseconds. In the first test we used the same string as it was used in (Alese *et al*., 2012): "ECDLP is believed to be harder than both the Integer Factorization and Discrete logarithm Problems". After 10 tests, we take the average of the elapsed interval. To make comparisons we take the encryption process as the signature generation (Alese *et al*., 2012). The size of the key, in the mobile device and the AC, is the corresponding to SHA-2 (256 bits).

The key generation interval was computed taking the time to get the random seed x and the time to compute the public key f^{N_x}. Such processes are performed during the user registration with the AC. The

Table 8. Alice wants to send a signed message to Bob but Alice is registered with CA_1 while Bob is with CA_2. Here, $s_0 = f(m)$, $s_1 = <s_0> f^{NA-2}$, $s_2 = <s_1> f^{N_{CA_1}, CA_2 - 2}$ and $s_3 = <s_2> f^{N_{CA_2}, B-2}$

$,A \rightarrow CA_1$:	A, B, f^{N_A-1}, s_1
$CA_1 \rightarrow CA_2$:	$A, B, f^{N_{CA_1}, CA_2 - 1}, s_2$
$AC_2 \rightarrow B$:	$A, f^{N_{CA_2}, B-1}, s_3$
$B \rightarrow AC_2$:	f^{N_B-1}
$CA_2 \rightarrow CA_1$:	$f^{N_{CA_2}, CA_1 - 1}$
$CA_1 \rightarrow A$:	$f^{N_{CA_1}, A-1}$
$A \rightarrow CA_1$:	f^{N_A-2, s_0}
$CA_1 \rightarrow CA_2$:	$f^{N_A-2}, f^{N_{CA_1}, CA_2-2}, s_1$
$CA_2 \rightarrow B$:	$f^{N_A-2}, f^{N_{CA_1}, CA_2-2}, f^{N_{CA_2}, B-2} s_2$
$A \rightarrow B$:	m

Table 9. Mobile devices used to perform tests

Device	Model	CPU	Android version	Internal GB memory
Samsung	GT I8190	1 GHz dual core ARM Cortex-A9	4.1.2 Jelly Bean Fou	8
Samsung	SM G920I	2.1 GHz, 1.5 GHz Octa-Core	6.0.1 Marshmallow	32

signature interval includes the computation of f^{N_x-1} for user authentication, f^{N_x-2} to sign the message, s_0, HMAC and the authentication of the AC. To verify the signature we compute f^{N_x-1}, s_0, s_1, s_2 and the comparison process performed by the AC. Results from the CA with respect to (Alese *et al*., 2012) are compared in Table 10. In adittion, Table 11 compare the performance of the hash signature system with (Mahto *et al*., 2016) after computing a message input of 8 and 256-bit.

If we compare the performance of the hash signature system on mobile devices with a 256-bit message we get the results shown in Table 12. Results are computed taking the average of twenty rounds. Although some parameters do not match (Tayoub *et al*., 2013) still are useful to compare the performance on mobile devices.

To guarantee the non-repudiation service, the CA maintains a log of the transactions between users. In Table 13 we show an example of an accounting record after an electronic transaction has been performed between Alice (A) and Bob (B) through the AC.

The Android App that was developed for the hash signature system occupies 4.57 Mbytes. Figure 5 shows its interface. In general terms, we can describe the behavior of the system when Alice wants to sign a message m for Bob: Alice sends the signature code to the CA and she also sends a notification to Bob. He gets the validation code from the CA. Then, Alice shares him the message m (or file) through the Internet, e.g. a social network. Finally, Bob verifies the validation code along with the message *m*.

Discussion

Although we don't have included a formal analysis about the security of the protocol, some points that enhance the security of the proposed protocol are: it is supported on the properties of hash functions that have been broadly discussed in the field. Despite this, we believe that a MITM could be tried by the attacker when users register with the AC. However, A MITM attack is limited by the following factors:

- The MITM could attack two randomly users of the Merkle tree. Otherwise, the attacker must know the exact moment in which the users A and B will be registered before the AC, in order to attack the nodes.
- Higher security certificates can be managed during the registration of users with the AC, which makes the attack more difficult.
- To block the original message, the malicious entity would have to manipulate the application software through the message is sent, along the message service of the mobile phone or any application as social network. In this case, the attacker must create false messages and notifications.

Table 10. Results of the SHA-256 hash signature are compared with respect to Alese *et al*. (2012), values are represented in milliseconds

Size of Key (bits)	Key Generation (ms)	Signature Generation (ms)	Signature Verification (ms)
RSA 1024	1312.7±190.8	166.9±46.3	15.7±0.4
RSA 2048	6804.6±2540.6	290.2±29.8	122.4±9.1
RSA 3072	32180.0±18947.7	310.5±75.5	293.2±71.8
RSA 7680	322843.0±233809.0	352.1±154.1	2932.8±44.7
RSA 15360	N/A	N/A	N/A
ECC P-160	198.6±12.5	17.9±4.9	15.7±0.1
ECC P-224	208.3±13.4	95.9±6.8	18.7±5.5
ECC P256	243.5±22.2	35.1±6.1	21.1±6.8
ECC P384	294.0±26.5	74.9±7.1	47.7±3.2
ECC P-521	447.8±90.9	138.2±4.9	109.9±0.3
SHA-256 hash signature	240.25	39.21	12.5

- The attacker would have to build a fake website of the CA, which would be detectable because the page is expected to be of public knowledge.

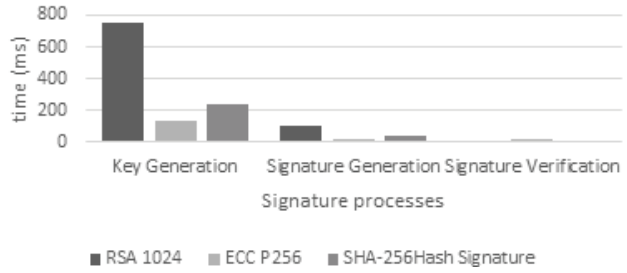

Figure 3. The hash signature system SHA-256 is compared to Alese *et al*. (2012). These results show that the signature with SHA-256 is similar to ECC P256 (key and signature generation) but faster in signature verification. With respect to RSA our proposal is faster most of the times

Table 11. The processes of signature generation (SG) and signature verification (SV) are compared between the hash signature system (AC with SHA-256) and Mahto *et al*., 2016 taking an 8 / 256-bit message input (results are computed in milliseconds)

	8 bits	256 bits		
Size of Key	SG (ms)	SV (ms)	SG (ms)	SV (ms)
RSA 1024	30.7	754.3	550	19310
RSA 2048	29.9	2707.5	580	102030
RSA 3072	30.5	6940.9	560	209600
ECC 160	488.5	1326.7	7920	22880
ECC 224	2203	1586.3	39700	26330
ECC 256	3876.3	1769	58430	27400
AC with SHA-256	32.7	18.5	39.87	17.8

Table 12. The hash signature performance is compared against ECC (ECDSA) and RSA (results in milliseconds)

ECC (ECDSA) (Tayoub, Walid; Somia, Lakehali; Chikouche, 2013)				
Device	Size key Operation (bits)	Generation key (ms)	Signature (ms)	Verification (ms)
GT-S6102	160	767	561	1285
	224	615	699	1217
	521	595	562	141
GT-I9100	160	303	412	681
	224	443	360	624
	521	595	562	141
RSA (Tayoub, Walid; Somia, Lakehali; Chikouche, 2013)				
GT-S6102	1024	1088.66	14.33	1
	2048	3896	77.66	1
	15360	>1h	\	\
GT-I9100	1024	1150	28	0.33
	2048	2341.66	95	1
	15360	>1h	\	\
Hash Signature system with SHA-256				
GT-I8190	256	1678.47	566.75	468.91
SM-G920I	256	158.55	128.65	94.22

Table 13. The CA maintains registers of the transactions to provide accounting services. Here is an example: f^{N_A-i-1} where i = counter of Sender. Also, if f^{N_B-j-1} counter of Receiver, $s_0 = f(m)$ and $s_1 = <s_0> f^{N_A-i-1}$

Username of Sender	Alice
Authenticity key of the issuer (f^{N_A-1})	d9e3ac586c94e9f4fcbc3ebdc6bde5e9e059ffc569e78909d574a90d23981cc9
HMAC of Message Hash ($s1$)	b5500c57b24aff2e8f0a223a8db300760f7bbc7cb382ccb8c5c4dc5547a8c0a3
Hash of Message ($s0$)	b221d9dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79
Key for HMAC Sing (f^{N_A-2})	0ff4d4bd8d0190b1579d57435abc4ba3c19f6b79b7e16017d586603d18aeae5b
Counter of Sender (i)	3
Username of Receiver	Bob
Key of Receiver (f^{N_B-j})	bacab276d9268c4d5c2d48ca8ce0bd3d8ad3b93be4b7370abe6f6b93f3ca7efd
Counter of Receiver (j)	2

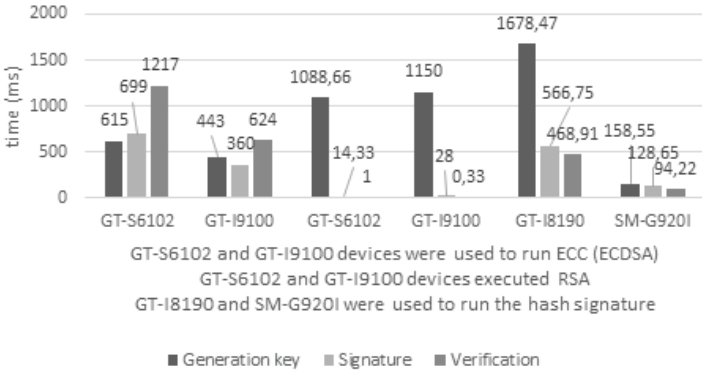

Figure 4. The hash signature system SHA-256 is compared to Tayoub *et al*. (2013). These results show that the hash signature system 256-bit, running in the SM-G9201 device is faster than ECC (ECDSA) 224-bit and RSA 1024-bit when comparing the time required for key generation, signature and verification

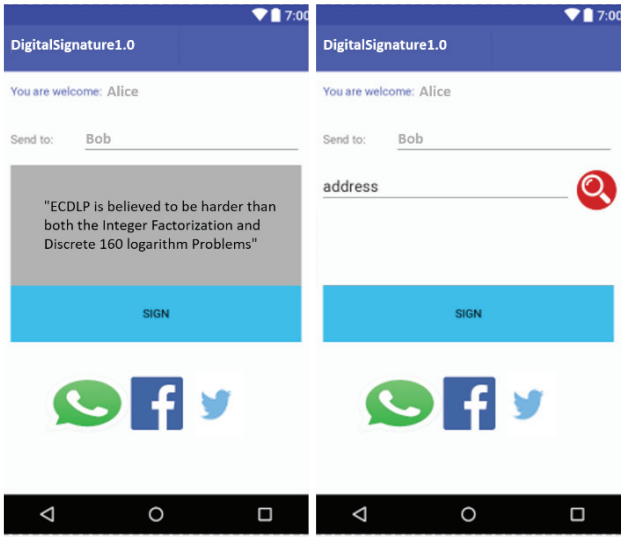

Figure 5. The interface of the hash signature system

Conclusions

It has been discussed a method to achieve a Digital Signature protocol based on hash functions. When compared to ECC, experimental results on mobile devices demonstrate that the hash signature improve the efficiency to generate the keys and the signing and verification processes. At the same time, the hash signature is faster than RSA when generating 2048 keys. In addition, larger RSA keys consume much time while the hash system does not require to increase the size of the keys.

The hash signature system defines the Certification Authority to provide certification of users to guarantee the non-repudiation service. In such a case, the generation, sign and verification processes is faster than RSA and ECC.

The method discussed here which is supported by hash functions, belongs to post-quantum cryptography, and seems to be suitable for mobile network devices often limited in hardware capacities. Security of the method is highlighted because it is founded on the well-known security of hash cryptography.

References

Alese, B.K., Philemon, E.D., Falaki, S.O. (2012). Comparative analysis of public-key encryption schemes. *International Journal of Engineering and Technology*, 2(9), Citeseer, 1552-68.

Anderson, R., Bergadano F., Crispo B., Lee J.H., Manifavas Ch., Needham R. (1998). A new family of authentication protocols. *ACM SIGOPS Operating Systems Review*, 32(4). ACM, 9-20. https://doi.org/10.1145/302350.302353

Buchmann A., Johannes B., Butin D., Florian G., Petzoldt A. (2016). *Post-Quantum cryptography : state of the art. The New Codebreakers*. Berlin Heidelberg: Springer, 88-108. https://doi.org/10.1007/978-3-662-49301-4_6

Buchmann, J., Dahmen, E., Szydlo, M. (2009). Hash-Based digital signature schemes. In: Post-Quantum Cryptography, Springer, 35-93. https://doi.org/10.1007/978-3-540-88702-7_3

Buldas, A., Laanoja, R., Truu, A. (2014a). *Efficient implementation of keyless signatures with hash sequence authentication.* IACR Cryptology ePrint Archive, 689.

Buldas, A., Laanoja, R., Truu, A. (2014b). *Efficient quantum-immune keyless signatures with identity*. IACR Cryptology ePrint Archive. Citeseer. 321.

Dattani, N.S. and Bryans, N. (2014). Quantum factorization of 56153 with only 4 qubits. arXiv Preprint arXiv:1411.6758.

DRAFT, FIPS PUB. (2014). 202. SHA-3 Standard: Permutation-Based hash and extendable-output functions. Information Technology Laboratory, National Institute of Standards and Technology. Recovered on May 2014 at http://csrc.Nist. Gov/publications/drafts/fips-202/fips_202_draft.Pdf. https://doi.org/10.6028/NIST.FIPS.202

Dridi, R. and Alghassi, H. (2016). Prime factorization using quantum annealing and computational algebraic geometry. *arXiv Preprint*, arXiv:1604.05796. https://doi.org/10.1038/srep43048

Dwight, H. (2011). Hash chain. In: Encyclopedia of cryptography and security, Springer. 542-543. https://doi.org/10.1007/978-1-4419-5906-5

Eastlake, 3rd, D. and Jones, P. (2001). US secure hash algorithm 1 (SHA1). https://doi.org/10.17487/RFC3174

FIPS, NIST. (1998). 186-1. Digital Signature Standard.

FIPS, PUB. (2000). 186-2. Digital Signature Standard (DSS). National Institute of Standards and Technology (NIST).

FIPS, PUB. (2002). 180-2. Federal information processing standards publication. SECURE HASH STANDARD, National Institute of Standards and Technology.

FIPS, PUB. (2009). 186-3. Digital Signature Standard (DSS).

Groza, B. (2006). Using one-way chains to provide message authentication without shared secrets. In: Security, privacy and trust in pervasive and ubiquitous computing, 2006. SecPerU, Second International Workshop, 82-87. http://doi.ieeecomputersociety.org/10.1109/SECPERU.2006.21

Haller, N. (1995). The S/KEY One-Time password system.

Hu, Y.C., Jakobsson, M., Perrig, A. (2005). Efficient constructions for one-way hash chains. In: International Conference on Applied Cryptography and Network Security, 423-41. https://doi.org/10.1007/11496137_29

Krawczyk, H., Canetti, R., Bellare, M. (1997). HMAC: Keyed-Hashing for message Authentication. https://doi.org/10.17487/RFC2104

Lamport, L. (1981) Password authentication with insecure communication. *Communications of the ACM*, 24(11), 770-772. https://doi.org/ 10.1145/358790.358797

Lamport, L. (1979). Constructing digital signatures from a one-way function.

M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., Ranen, O. (2005). Hotp: An hmac-based one-time password algorithm. https://doi.org/ 10.17487/RFC4226

M'Raihi, D., Machani, S., Pei, M., Rydell, J. (2011). Totp: time-based one-time password algorithm. https://doi.org/ 10.17487/RFC6238

Mahto, D., Ali-Khan D., Kumar-Yadav D. (2016). Security analysis of elliptic curve cryptography and RSA. In: Proceedings of the World Congress on Engineering, Vol. 1.

Merkle, R.C. (1982). Method of providing digital signatures. Google Patents.

Merkle, R.C. (1987). A digital signature based on a conventional encryption function. In: Advances in Cryptology, CRYPTO 87, 369-78. https://doi.org/10.1007/3-540-48184-2_32

Merkle, R.C. (1989). A certified digital signature. In: Conference on the Theory and Application of Cryptology, 218-38.

Muñoz, J.L., Forne, J., Esparza, O., Soriano, M. (2004). Certificate revocation system implementation based on the merkle hash

tree. *International Journal of Information Security*, 2(2),. 110-24. https://doi.org/ 10.1007/s10207-003-0026-4

Nielsen, M.A. and Chuang, I.L. (2010). *Quantum computation and quantum information*. Cambridge university press. https://doi.org/10.1063/1.1428442

Nigel, P.S. (2016). *Hash functions, message authentication codes and key derivation functions*, Springer International Publishing, 271-294. https://doi.org/10.1007/978-3-319-21936-3_14

Perlner, R.A. and Cooper, D.A. (2009). Quantum resistant public key cryptography: A survey. In: Proceedings of the 8th Symposium on Identity and Trust on the Internet, 85-93. https://doi.org/10.1145/1527017.1527028

Perrig, A., Canetti, R., Doug, J.T., Song, D. (2005). The TESLA broadcast authentication protocol. RSA CryptoBytes 5. RSA.

Rivest, R. (1992). The MD5 Message-Digest Algorithm. https://doi.org/10.17487/RFC1321

Rivest, R.L., Shamir, A., Adleman, L. (1978). A Method for obtaining digital signatures and Public-Key cryptosystems. *Communications of the ACM*, 21(2), ACM, 120-26. https://doi.org/10.1145/359340.359342

Shor, P.W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In: Foundations of Computer Science, 1994 Proceedings, 35th Annual Symposium, 124-34. https://doi.org/10.1109/SFCS.1994.365700

Tayoub, W., Lakehali, S., Noureddine, C. (2013). Implementation of Public-Key cryptographic systems on embedded devices (case : Computation speed).

Triwinarko, A. (2006). Elliptic curve digital signature algorithm (ECDSA). Program Studi Teknik Informatika ITB, Bandung.

Zohar, A. (2015). Bitcoin: Under the hood. *Communications of the ACM*, 58(9). ACM, 104-13. https://doi.org/10.1145/2701411