International Journal of Grid and Distributed Computing
Vol. 10, No. 12 (2017), pp.65-80
http://dx.doi.org/10.14257/ijgdc.2017.10.12.07

A Study on Secure and Efficient KSI System based on Multi Path Hash Chain with Universal Hashing Function

Gyeong-Jin Ra[1] and Im-Yeong Lee[2]

[1,2] Soonchunhyang University, Republic of Korea
[[1]rababi, [2]imylee]@sch.ac.kr]
**Corresponding author : Im-Yeong Lee*

Abstract

Recent advances in quantum computer technology and Shor algorithms have increased the importance of new data integrity verification for the security of existing public key cryptography. Keyless signature can perform high-speed data integrity verification using only hash operation and immunity to quantum computer using One Time Password(OTP). Keyless Signature Infrastructure(KSI) is an environment for Keyless Signature. It provides user certificate creation through distributed network server and block time chain global time stamp token for message integrity. However, when a large number of users are added to the KSI and the OTP is managed, the number of end nodes increases, which increases the KSI overhead. In this paper, we propose a KSI system that reduces overhead by efficiently merging a large number of nodes by applying multi path hash chain. Finally, we proposed universal hashing function so that improved secure KSI system.

Keywords*: Block Chain, Keyless Signature Infrastructure, Multi Path Chain, Global Time Stamp*

1. Introduction

Recent developments in ICT(Information and Communication Technologies) and the fourth industrial revolution of many major industrialized nations are expanding into a super-connected society, where various types and amounts of data are being collected and processed. Therefore, the importance of cyber security has been increased so that data is secured from legitimate user authentication and tampering with the attacker. However, existing digital signature schemes based on public keys have disadvantages such as high computational complexity, problems with public key certificates, and periodic key updates. There are also security threats of existing public key cryptosystems such as technological advances in quantum computers and Shor algorithms[1]. Keyless digital signatures have been proposed to manage large amounts of data securely and efficiently. Keyless digital signature has features of high speed operation by hash operation by digital signature method using hash function and hash tree without using existing public key[2]. Also, there is no need for separate key management, and the period of validity of the key is not fixed, so that it is not required to update periodically[3](Figure 1).

Received (September 11, 2017), Review Result (December 2, 2017), Accepted (December 7, 2017)

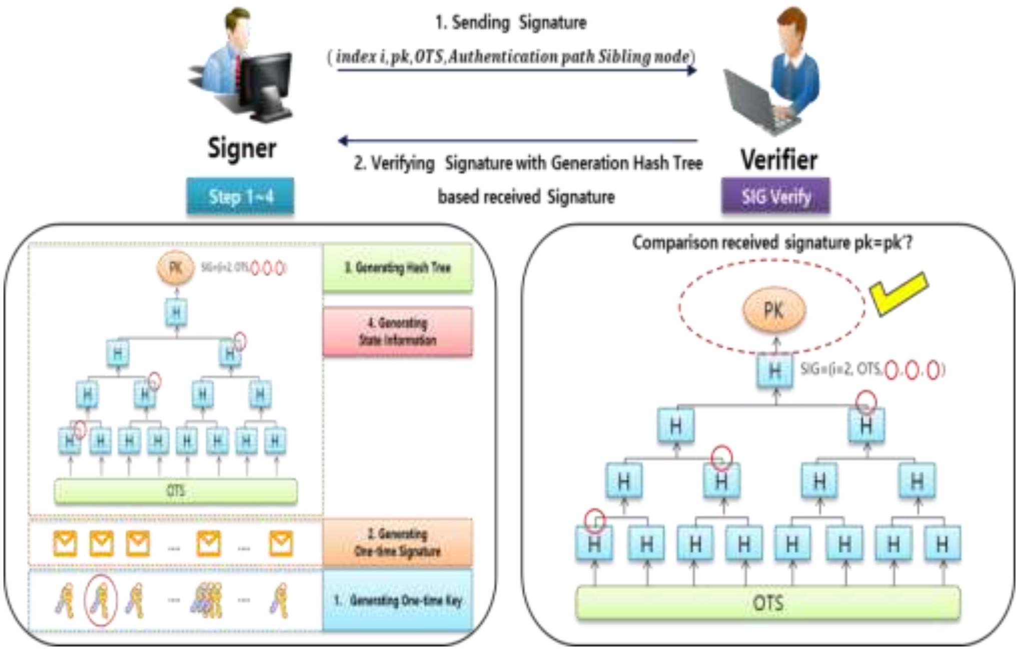

1. Sending Signature
(index i, pk, OTS, Authentication path Sibling node)
Signer
Verifier
2. Verifying Signature with Generation Hash Tree based received Signature
Step 1~4
SIG Verify
PK
3. Generating Hash Tree
4. Generating State Information
2. Generating One-time Signature
1. Generating One-time Key
OTS
H
Comparison received signature pk=pk'?
SIG=(i=2, OTS,○,○,○)

Figure 1. General Keyless Signature Scheme Operation

To do this, a keyless digital signature generates a one-off key and wraps the disposable signature into a hash tree to generate a signature containing the root value. KSI generates a signature through a certificate issued after user authentication with keyless digital signature as OTP and generates global timestamp by block chaining with distributed network collaboration [4]. In a keyless digital signature, the signer is safe for retransmission attacks by using a one-time key, but since a sequence of keys is pre-created and used sequentially, a malicious attacker may preempt a key or take a back-estate attack. The hash function is a defense against omnidirectional attack and constitutes a key of KSI with hash binary tree. In this paper, we propose a keyless digital signature scheme that uses a multi path hash chain based KSI system to improve the overhead and traffic of a large number of data. We propose a mechanism that minimizes the increase of overhead for nodes [5].

2. Related Works

The KSI is a type of block chain, which is an infrastructure for keyless digital signatures that authenticate users based on a one-time key, compute a message as a whole hash tree, and commit the global timestamp to the block chain[6]. Therefore, unlike the existing PKI signing method, KSI is different from the method of private key signing after hashing the message. User authentication and message integrity verification are separated. The KSI maintains the collaborative and public records of the distributed network through the block chain. Thus providing robust and fast operation for forgery and modulation. KSI is classified according to user's one-time key generation method and keyless digital signature method. It is classified into LDOTS(Lamport-Diffie One-Time Signature), WOTS(Winternitz One-Time Signature), and hash chain according to the one-time key generation method, and then Merkle Tree MSS (Mekle Signature Sheme) and extended XMSS (Extemed Merkle Siganture Scheme)[7,8,9].

2.1. KSI (Keyless Signature Infrastructure

The KSI provides an infrastructure for keyless digital signatures. The user transmits the hash value of the message and the private key pair to the KSI after the authentication request through the one-time key [3]. The KSI verifies the user's public key via a message and private key pair and generates a message tree. Afterwards, it links to the global time and commits to the block chain after linking and returns the global timestamp to provide the user with the role of digital signature [6]. Finally, the verifier requests the KSI to verify the global timestamp of the OTS corresponding to the message, thereby verifying the integrity of the user authentication and data. In this case, when applying a hash algorithm known to be safe, the attacker must find the collision value of the entire node value of the tree in order to maintain the integrity while attacking the data, which is very difficult. Also, KSI servers distributed by committing by block chain manage the timestamp value in common, so they are resistant to counterfeiting and modification. In particular, we expect the safety of digital signatures based on the fact that attacks are more difficult because keys and signatures are one time (Figure 2).

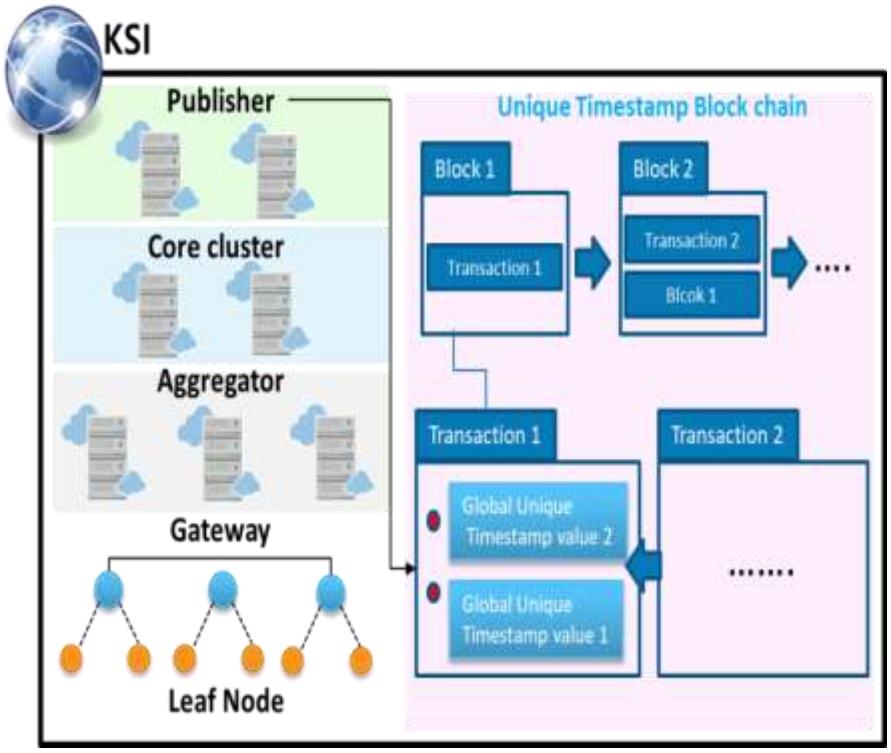


Figure 2. KSI's Global Timestamp System

2.2. Lampot-Diffie One Time Signature

The LDOTS was proposed in 1979 by Lamport and Diffie [1]. Based on this, a number of OTS techniques with increased safety and efficiency have been proposed. A key pair generation step for signing, a signature step, and a verification step, and outputs a 256-bit result using the hash function.

1) Key generation phase

<u>Step 1</u>. The signer generates a private key and a public key for signing. Private keys generate 256 pairs, 512 total secret key pairs using pseudo random.

$$\textit{Private Key} = (pseudo)random\ ((s_0, 0, s_1, 1), \ldots, (s_{255}, 0, s_{255}, 1))$$
$$for\ s_{i,j} \in 0, \ldots, 2^{256} - 1$$
$$\textit{Public Key} = ((h(s_0, 0), h(s_0, 1)), \ldots, h(s_{255}, 0), h(s_{255}, 1))$$

<u>Step 2.</u> The signer generates 512 public keys using one-way private key.

2) Signature Generation Phase

The signer hashes the message M to be signed. According to the bit order of the hash *M*, if 0, select the key of the first column of the private key, and if it is 1, select the key of the second column.

$$M = (m_0, \dots, m_{256})$$
$$\sigma(M) = (s_0, m_0, \dots, s_0, m_{256})$$

3) Signature Verification Phase

The verifier hashes the message *M* to be verified. According to the bit order of the hash *M*, if it is 0, the key of the first column of the public key is selected. If it is 1, the key of the second column is selected.

2.3. Winterniz One Time Signature

WOTS generates the key pair through random value by *SEED* and BitMask, F function in the way suggested by Robar Winternitz in 1994. Winternitz Parameter value is selected and used [7].

1) Key generation phase

The signer generates a private key and a public key for signing. The private key generates a bit string of n bit length using pseudo random and generates a public key using the f function and w in the private key. At this time, the value of *t* according to the Winternitz parameter is calculated by the following equation.

$$\text{Private Key } X = (x_{t-1}, \dots, x_1, x_0) \in_R 0{,}1^{(n,t)}$$
$$\text{Public Key } Y = (y_{t-1}, \dots, y_1, y_0) \in_R 0{,}1^{(n,t)},\ y_i = f^{2w-1}(x_i), 0 \le i \le t-1$$
$$\mathrm{t}_1 = \left\lceil \frac{n}{w} \right\rceil, \mathrm{t}_2 = \left\lceil \lfloor \log_2 t_1 \rfloor + 1 + \frac{w}{w} \right\rceil, t = t_1 + t_2$$

2) Signature generation phase

The signer divides the message to be signed into w numbers of length *d*. Filled with 0 by using pre-padding for w multiple. Obtains the split d by bit string. The checksum *C* is obtained by the following formula

$$d{=}b_{t-1} \,||\, \dots b_{t-t_1}$$
$$C{=}\textstyle\sum_{i=t-t_1}^{t-1}(2^w - b_i)$$

At this time, $\lfloor log_2 t_1 2^w \rfloor + 1 = \lfloor log_2 t_1 \rfloor + w + 1$ is smaller than according to $C \le t_1 2^w$. The signer prepends the obtained *C* to the *w*-length as shown in the previous message, and then obtain the separated *c* to generate the signature.

$$\mathrm{c} = b_{t_{2-1}} || \dots || b_0$$
$$\sigma = (f^{b_{t-1}}(x_{t-1}), \dots., f^{b_1}(x_1), f^{b_0}(x_0))$$

3) Signature verification phase

The validator verifies the validity of the signature by checking whether the signature is the same as the public key y_i through the following formula verification process.

$$\left(f^{2w-1-b_t-1}(\sigma_{n-1}), \dots, f^{2w-1-b_0}(\sigma_0)\right) = (y_{n-1}, \dots, y_0)$$
$$f^{2w-1-b_i}(\sigma_i) = f^{2w-1}(x_i) = y_i$$

2.4. MSS (Merkle Signature Scheme)

MSS is a method proposed by Merkle in 1979. The leaf node is *WOTS*, and the upper node is calculated through the hash after connecting the adjacent nodes from the bottom. This is done repeatedly to make the final Root node and make it a public value that is part of the signature value [1]. The initial hash-based digital signature scheme has the drawbacks of long signature keys and signature length.

Table 1. Classification of Keyless Signature

	Strength	Weakness	Example
- Simple key generation - Suitable for disposable use	- Simple key generation - Suitable for disposable use	- Inadequate management of multiple data -Long signature length - Key management storage inefficiency	LD-OTS WOTS
Hash Tree-based Signature	- Efficient management of public keys - One-Time signature efficient verification	-State Full -Key search algorithm required	MSS XMSS

2.5. XMSS (Extended Signature Scheme)

XMSS was proposed by Johannes Buchmann in 2013[9]. Before concatenating differently from MSS, we XOR with Bit Mask and weave hash tree. XMSS has twice the intensity of hash collision than conventional MSS. In other words, it has the same strength as the output value of half, and has faster calculation and short signature length than MSS. When using SHA2-256 at 128 bit security strength, it shows performance of 2,001,479 cycles and is currently in the (Internet Engineering Task Force)IETF final candidates[2,11].

3. Security Requirements

The security requirements for KSI - based authentication management and communication for the secure smart home environment proposed in this study are as follows.

- **Signer certification**: Anyone can verify the signer of the signature.
- **No forgery**: Only legitimate signers should be able to generate signatures.
- **Reusable**: The signature of the data should not be used as a signature of other data.
- **Can not change**: The contents of the signed data must not be changed.
- **Non-repudiation**: The signer must not be able to deny the fact that he or she has signed it.
- **Data confidentiality**: Only authorized users should be able to access information assets.
- **Data integrity**: Only authorized users should be able to change information assets.
- System Availability: Must be accessible by authorized users at the appropriate time.
- **Mutual authentication**: Mutual authentication should be ensured through the smart home device key, and the session key must be agreed.

Finally, the attacks that can occur in this research environment are as follows.

- **Sybil Attack**: A malicious attacker can take out a user's private key and create several corresponding public keys to create a valid block. An malicious attacker creates multiple public keys in a single private key and creates many nodes by pretending to be a legitimate user. This occurs in decentralized systems without a central controller.

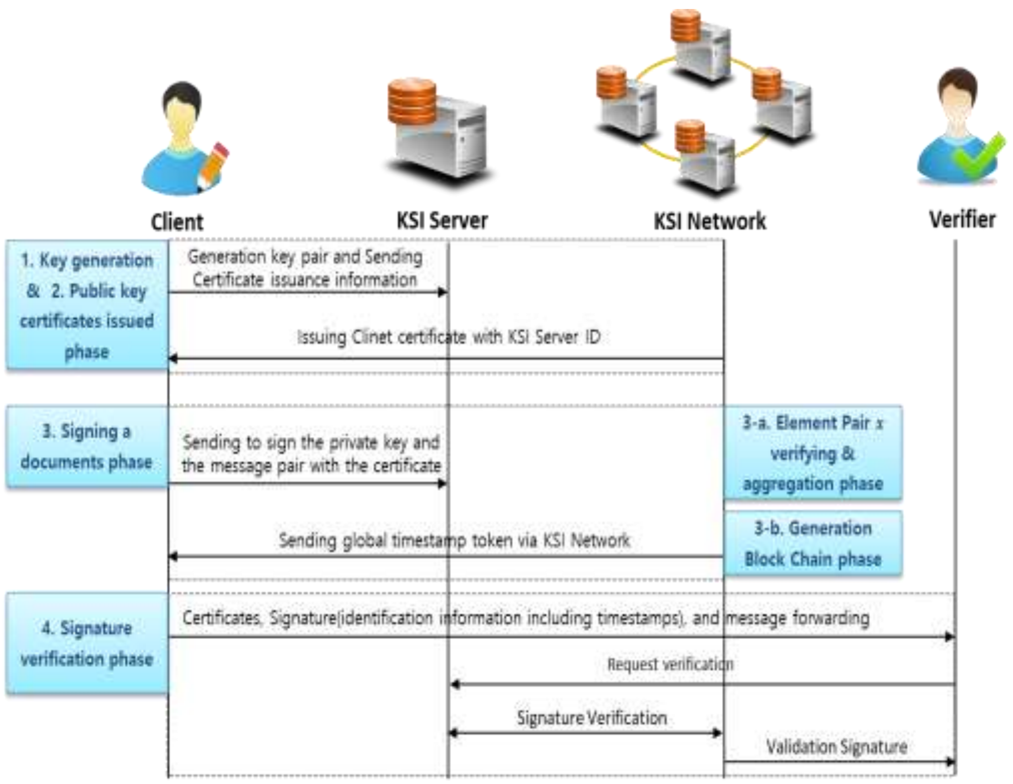

Client
KSI Server
KSI Network
Verifier
1. Key generation & 2. Public key certificates issued phase
Generation key pair and Sending Certificate issuance information
Issuing Clinet certificate with KSI Server ID
3. Signing a documents phase
Sending to sign the private key and the message pair with the certificate
3-a. Element Pair x verifying & aggregation phase
Sending global timestamp token via KSI Network
3-b. Generation Block Chain phase
4. Signature verification phase
Certificates, Signature(identification information including timestamps), and message forwarding
Request verification
Signature Verification
Validation Signature

Figure 3. All Scenario of the Proposed Scheme

4. Proposed Scheme

The proposed scheme consists of user and verifier KSI distributed network server and has total 1 ~ 4 phase (Figure 3). In phase 1, the client creates a public key-private key pair using a hash chain and a hash tree for authentication. In phase 2, the client receives certificate information from the KSI by issuing the signing certificate along with his public key pair. In phase 3, the KSI server creates a global hash tree from the user's message and private key pair, commits it to the block chain, and returns a global timestamp token. In this step detailed has 2 phase 3-a,b. first, In phase 3-a, the KSI aggregator aggregates the pair x data generated by the client. This step we applied multi path hash chain so that KSI aggregator aggregates signature low overhead. In phase 3-b the KSI publisher publishes timestamp token and commits block chain via inner consensus rule. In phase 4, the verifier requests the KSI server to verify the signature information received from the user. In the proposed scheme, when constructing a hash tree using a message received from a user and a private key pair as a leaf node, the proposed scheme attempts to reduce the overhead by verifying authentication and integrity of nodes using a multi-hash chain and merging data.

4.1. System Parameter

The proposed scheme system parameter as shown follows Table 2.

Table 2. Proposed Scheme System Parameter

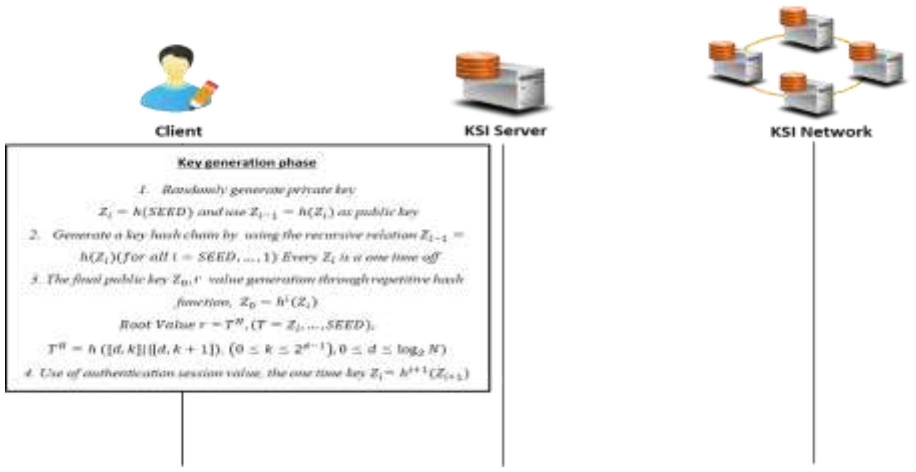
ID_c	The identity of the client	n	The hash tree height created with a disposable signature $n \geq 0$
Z_i	The private key of the hash chain generated by using a random *SEED*	i	The private key index in keychain
r, Z_0	The public key pair generated by using a hash chain and hash tree	C_i	The minimum authentication path node used to calculate the root value of the hash tree
h	The one-way hash function operation	S_t	The KSI creates a hash tree time stamp and commits block chain token
m	The message to sign	t_0	The time when the certificate become valid (*i.e.* Z_i is intended to sign documents at time t_0+1, Z_2 is for signing at t_0+2 *etc*)
ID_s	The identity of the signature server that is authorized to serve the client	k_l	The random key according to the hash chain length used for key generation
l	The hash chain length value used to generate the key.	x_l, y_l	The secondary path according to k_l
r	The root value calculated by using the recursive hash a concatenation and hash of adjacent two nodes		

4.2. Key Generation Phase

In this phase as shown follows Figure 4, the client generates a private key and a public key pair using a hash chain and a hash tree in order to issue a certificate and authenticate to the KSI.

Step 1. Client generates a private key Z_i by using a random $SEED$ and use Z_{i-1} as public key by using a hash value which private key Z_i

$$private\ key, Z_i = h(SEED)$$
$$public\ key, Z_{i-1} = h(Z_i)$$



Key generation phase

1. Randomly generate private key $Z_i = h(SEED)$ and use $Z_{i-1} = h(Z_i)$ as public key
2. Generate a key hash chain by using the recursive relation $Z_{i-1} = h(Z_i)(for\ all\ i = SEED, \ldots, 1)$ Every Z_i is a one time off
3. The final public key Z_0, r value generation through repetitive hash function, $Z_0 = h^i(Z_i)$
Root Value $r = T^n, (T = Z_i, \ldots, SEED)$,
$T^n = h([d,k] \| [d, k+1]), (0 \le k \le 2^{d-1}), 0 \le d \le \log_2 N)$
4. Use of authentication session value, the one time key $Z_i = h^{i+1}(Z_{i+1})$

Figure 4. Key Generation Phase

<u>Step 2.</u> Client generates a key hash chain by using the recursive relation $Z_{i-1} = h(Z_i)$(for all i=$SEED$,…,1). Every Z_i is a one-time password for a particular time.

<u>Step 3.</u> Client computes the public key pair Z_0, root hash r of a hash tree, created with a hash function H. Let $r = T^H$, $(T = Z_i, \ldots., SEED)$.

$$Z_0 = h^i(Z_i)$$
$$root\ value\ r = T^H = h([d,k]||[d,k+1]),\ (0 \leq \mathrm{k} \leq 2^{\mathrm{d}-1}, 0 \leq \mathrm{d} \leq \log_2 N)$$
$$(h = Z_i, \ldots, SEED),$$

<u>Step 4.</u> Client uses the one-time key sequentially in the opposite direction that created $Z_i = h^{i+1}(Z_{i+1})$.

4.3. Public Key Certificates Phase

In this phase as shown follows Figure 5, the client creates a key pair and then requests the KSI to generate a certificate by using the client information. The KSI generates a certificate to the client as a global hash tree.

<u>Step 1.</u> Client sends client's authentication information 3-tuple <ID_c, Z_0, r> to KSI server.

<u>Step 2.</u> KSI server creates a public key certificate for client by using hash tree. The hash tree for user certificate generation is operated as a single distributed network server and distinguished by the tag corresponding to the user.

<u>Step 3.</u> KSI server returns <$ID_c, Z_0, r, t_{0,}\ ID_s$> to the client with the server ID_s and the certificate valid time t_0.

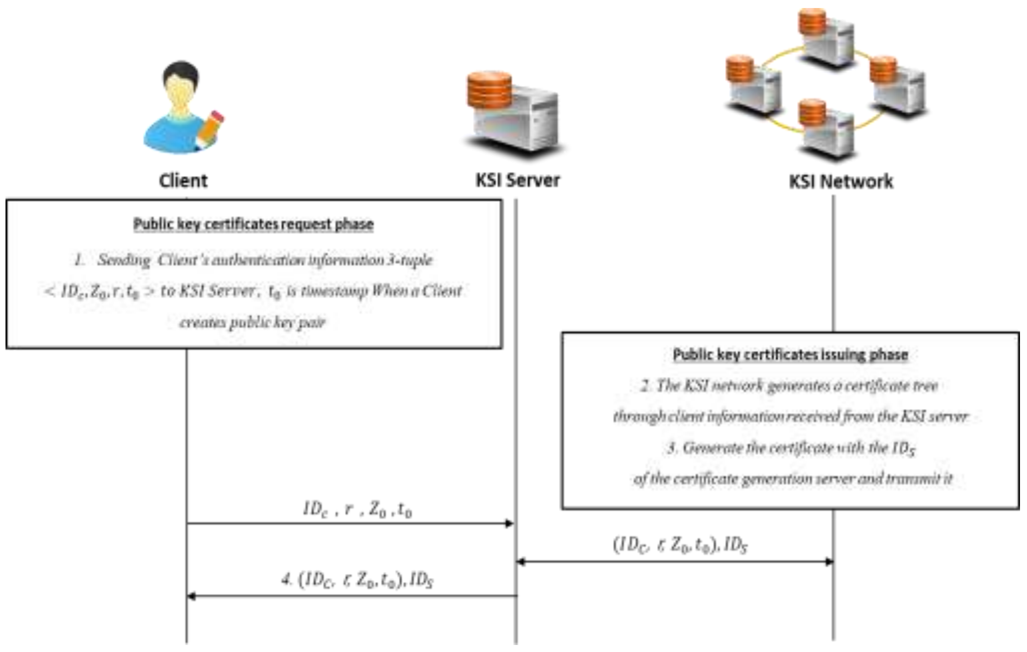

Client
KSI Server
KSI Network
Public key certificates request phase
1. Sending Client's authentication information 3-tuple
$< ID_c, Z_0, r, t_0 >$ to KSI Server, t_0 is timestamp When a Client
creates public key pair
Public key certificates issuing phase
2. The KSI network generates a certificate tree
through client information received from the KSI server
3. Generate the certificate with the ID_S
of the certificate generation server and transmit it
ID_c , r , Z_0 , t_0
$(ID_C, r, Z_0, t_0), ID_S$
4. $(ID_C, r, Z_0, t_0), ID_S$

Figure 5. Public Key Certificates Phase

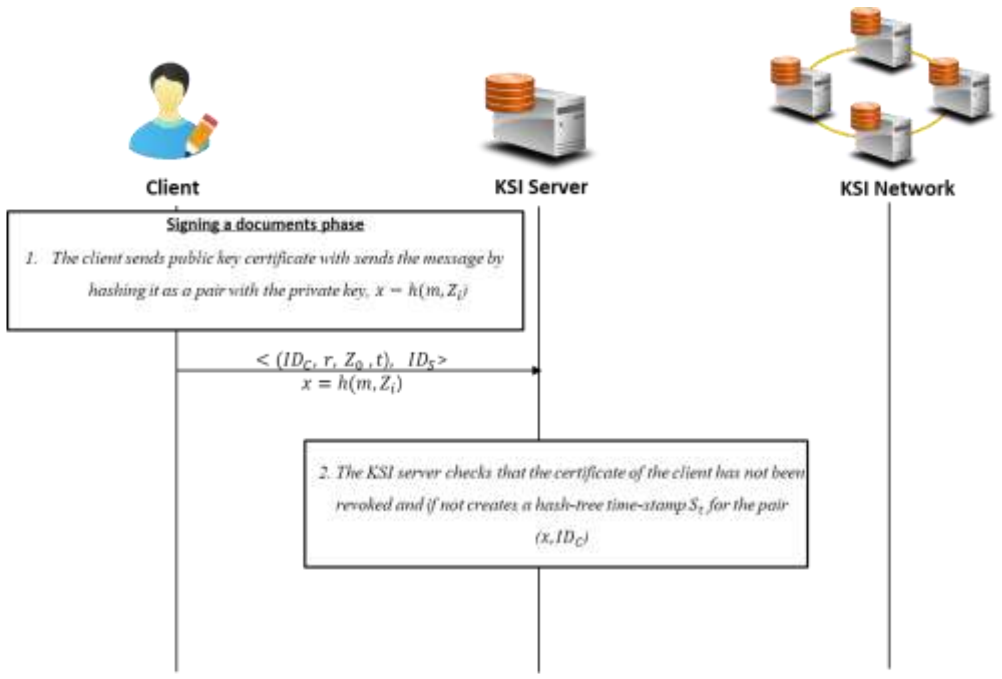

Client
KSI Server
KSI Network
Signing a documents phase
1. The client sends public key certificate with sends the message by hashing it as a pair with the private key, $x = h(m, Z_i)$
$< (ID_C, r, Z_0, t), \ ID_S >$
$x = h(m, Z_i)$
2. The KSI server checks that the certificate of the client has not been revoked and if not creates a hash-tree time-stamp S_t for the pair (x, ID_C)

Figure 6. Signing a Document Phase

Step 4. When a client requests a signature, authenticates using client's a certificate <$ID_c, Z_0, r, t_{0,}$ ID_s> received from the KSI server.

4.4. Signing a Document Phase

In this phase as shown follows Figure 6, the client requests the KSI to sign the hash of the private key and message pair generated by the hash chain. The KSI generates global timestamps with global hash trees and commits them to block chains, making forgery and tampering impossible.

Step 1. Client to sign a message m (or a hash of a message) at time $t > t_0$ (where $t = t_0 + i$), computes $x = h(m, z_i)$ and sends x together with its identity $\mathrm{ID_c}$ as a request to the signature server.

Step 2. The KSI server checks that the certificate of the client has not been revoked and if not creates a hash-tree time-stamp $\mathrm{S_t}$ for the pair (x, ID_C)

a). Signing a Document Element pair *x* verifying and aggregate Phase

In this phase as shown follows Figure 7, KSI aggregator collects and merges pair x, which is one of the signature information of many clients, and verifies integrity and authentication of nodes using multi-path hash chain and merges them efficiently.

Step 1. KSI aggregator decrements the value of *l* delivered from the client by one.

Step 2. KSI aggregator divides the case where the value of n transmitted from the client is an even number or an odd number, and calculates a main path k_l as follows. The KSI Aggregator generates the hash value of the main path k_l through the random *SEED*. Then, the Secondary paths y_l and x_l are generated and calculated by the following equation.

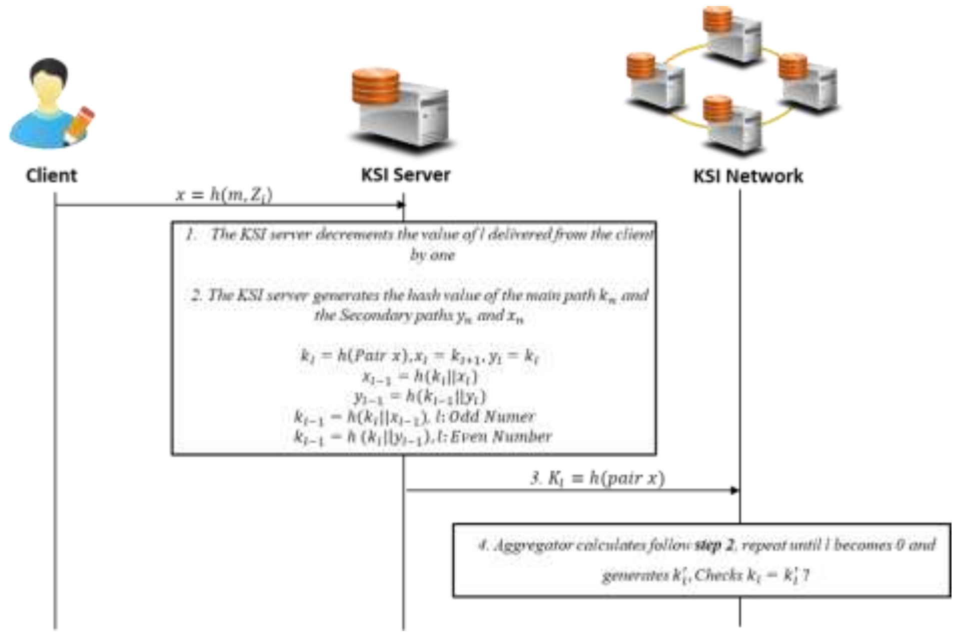

Client
KSI Server
KSI Network
$x = h(m, Z_l)$
1. The KSI server decrements the value of l delivered from the client by one
2. The KSI server generates the hash value of the main path k_n and the Secondary paths y_n and x_n
$k_l = h(Pair\ x), x_l = k_{l+1}, y_l = k_l$
$x_{l-1} = h(k_l||x_l)$
$y_{l-1} = h(k_{l-1}||y_l)$
$k_{l-1} = h(k_l||x_{l-1})$, l: Odd Numer
$k_{l-1} = h(k_l||y_{l-1})$, l: Even Number
3. $K_l = h(pair\ x)$
4. Aggregator calculates follow **step 2**, repeat until l becomes 0 and generates k_l', Checks $k_l = k_l'$?

Figure 7. Signing a Document Element pair *x* verifying and Aggregate Phase

$$Initial\ setting, k_l = h(SEED), x_l = k_{l+1},\ y_l = k_l$$
$$x_{l-1} = h(k_l||x_l)$$
$$y_{l-1} = h(k_{l-1}||y_l)$$
$$k_{l-1} = h(k_l||x_{l-1}),\ l:\ \text{Odd Number}$$
$$k_{l-1} = h(k_l\ ||y_{l-1}),\ l:\ \text{Even Number}$$

Step 3. KSI aggregator computes the secondary path x_l to the hash function in $h(k_{l+1})$ to check the authentication and integrity of the previous node of the client. $x_l = h(k_{l+1})$

Step 4. KSI aggregator computes the secondary path y_l as a value applied to the hash function as the secondary path x_l to compute the main path k_l.

b) Generation Block Chain

In this phase as shown follows Figure 8, KSI network publisher generates block chain via inner consensus rule.

Step 1. KSI network's publisher commits the global timestamp S_t to block chaining through an inner agreement algorithm. The final publisher of the KSI links the root value of the hash tree to the global coordinated time.

Step 2. KSI network's publisher commits a block chain with a consensus through a certain timestamp transaction and broadcasts it.

Step 3. KSI server returns S_t token to the client.

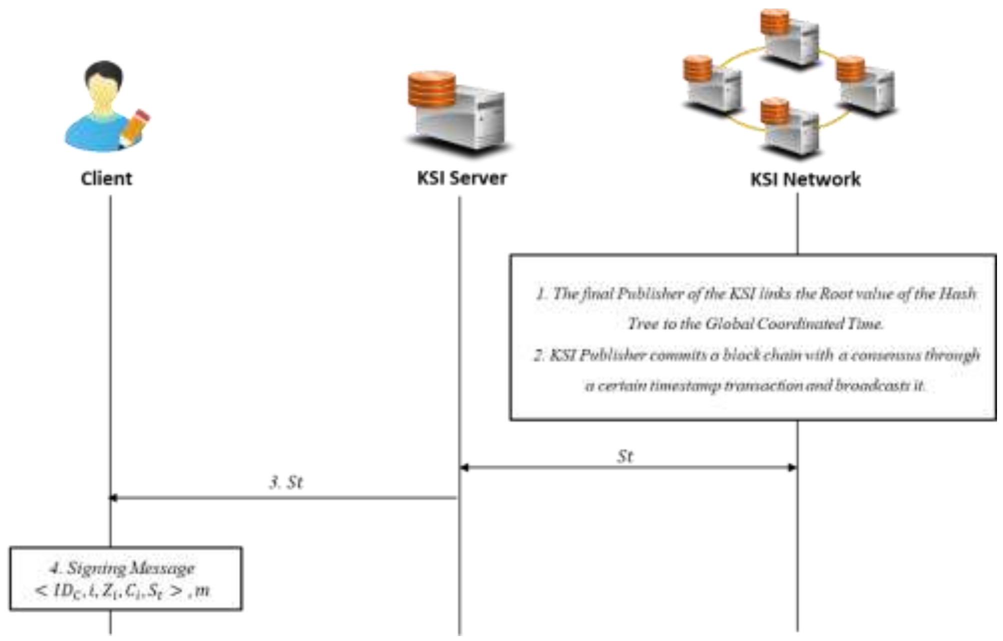

Client
KSI Server
KSI Network
1. The final Publisher of the KSI links the Root value of the Hash Tree to the Global Coordinated Time.
2. KSI Publisher commits a block chain with a consensus through a certain timestamp transaction and broadcasts it.
St
3. St
4. Signing Message
$< ID_C, i, Z_i, C_i, S_t >$, m

Figure 8. Generation Block Chain

<u>Step 4.</u> Client signs m $< ID_C, i, Z_i, C_i\ S_t >$, where C_i is the hash chain which proves that Z_i is the i-th element of the key chain.

4.5. Signature Verification Phase

In this phase as shown follows Figure 9, any verifier who wants to verify the client's signature can verify the signature by passing the signature information and certificate to KSI. The KSI is a distributed network server that can be quickly and easily verified because it shares a single ledger with a global timestamp as a block chain.

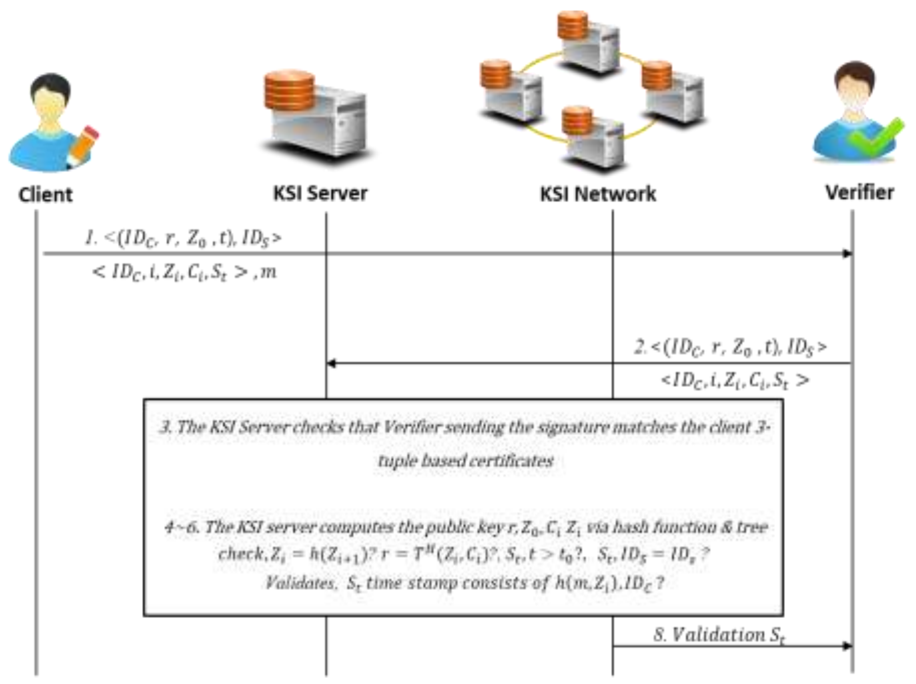

Client
KSI Server
KSI Network
Verifier
1. $<(ID_C, r, Z_0, t), ID_S>$
$< ID_C, i, Z_i, C_i, S_t >$, m
2. $<(ID_C, r, Z_0, t), ID_S>$
$<ID_C, i, Z_i, C_i, S_t >$
3. The KSI Server checks that Verifier sending the signature matches the client 3-tuple based certificates
4~6. The KSI server computes the public key $r, Z_0, C_i\ Z_i$ via hash function & tree check, $Z_i = h(Z_{i+1})$? $r = T^H(Z_i, C_i)$?, $S_t, t > t_0$?, $S_t, ID_S = ID_s$?
Validates, S_t time stamp consists of $h(m, Z_i), ID_C$?
8. Validation S_t

Figure 9. Signature Verification Phase

Step 1. Client sends the signature $< ID_c, i, Z_i, C_i\ S_t >$ on the message m to the KSI server with the certificate $<ID_c, Z_0, r, t_{0,}\ ID_s>$.

Step 2. Verifier request the KSI server to verify the signature $< ID_c, i, Z_i, C_i\ S_t >$ on the message m with the certificate $<ID_c, Z_0, r, t_{0,}\ ID_s>$.

Step 3.KSI server checks that the signature matches the client ID_c of the clinet certificate.

Step 4. KSI server computes the public key r, which is the root value of the hash tree, through the private key Z_i of the client and the hash chain C_i, or that hash function iterated on private key Z_i i times leads to Z_0.

Step 5. KSI server validates S_t time stamp on *(h(m,Z_i), ID_C)*

Step 6. KSI checks time t extracted from S_t satisfies t=t_0+i, *i.e.* correct key was used.

Step 7. KSI server checks identities in S_t and the certificate, *i.e.* server was authorized by the client to the signature.

Step 8. KSI server coincides signature and returns it to the verifier.

5. Analysis

In this section, the existing schemes and the proposed scheme is classified by efficiency and safety. In particular, safety is described in accordance with Section 3 of the security requirements as shown in Table 3.

Table 3. Analysis of Proposed Schemes

	MSS	XMSS	KSI	Proposed Scheme
Authentication/ Integrity	*Offer*	*Offer*	*Offer*	*Offer*
Sybil Attack	*Unsafe*	*Unsafe*	*Safe*	*Safe*
Key Schedule Scheme	$WOTS$	$WOTS^+$	*Hash Tree Hash Chain*	
Hash Tree	-	-	*Merkle Tree*	*XOR Tree based Universal Hash function*
Key number	$(P_k, S_k) * \frac{m}{d}$		$(S_k) * \frac{m}{d}$	
Key Search complexity	$O(n)$	$O(log\, n)$		
Cluster Node Aggregation	*Not offer*	*Not offer*	*Merkle* Tree	*Multi path hash chain*
Number of Changed Key bit in hash values	$\approx Hash\ Key\ Number$			$\approx 64(bit)$

S_k : Signature Key, P_k : Public Key, m : Message Length d : Message Block
H : Tree Depth, n : Layer Number

For the proposed scheme analysis, we use two items of change of average number of key bits for hash chain. Compared with the conventional scheme, the average key length number of the hash chain is compared.

5.1 Analysis of Security

Integrity: The proposed scheme prevents duplicate transactions of the message timestamp token registration through the block chain and provides strong integrity using the structure of the hash.

Authentication: The proposed scheme guarantees that the user is a legitimate user by performing user authentication and public key authentication based on the public key through the key generated by the user based on the hash chain.

Sybil Attack: The proposed scheme generates and verifying the public key is different each time depending on the sibling node C_i. Therefore, by creating multiple public keys in one private key, arbitrary attacker masquerades as a legitimate user, thereby preventing Sybil attacks that generate a large number of nodes.

5.2. Analysis of Efficiency

For the experiment evaluation, we use the average number of key bits for the hash chain. We use the average number of key bits used in the experiment for evaluation in item [10]. The following paper is based on experiments on the rate of change in the hash key in the Caotic map in the manner proposed in 2005 Xiao *et al.* Table 4 shows the key change rates for the input keys[10]. We apply this to the number of proposal scheme nodes will result in the following. The overhead of the storage space according to the merge and change rate after the use of the proposal scheme is shown follow Figure 10. In Figure 10, the overhead after merging clusters with our suggested scheme is as follows compared to before application.

Therefore, the proposed scheme uses the user authentication key according to the irreversibility characteristics of the one-way hash function. The value before hashing is used as the private key and then used as the public key of the certificate. As a result, KSI-based systems reduce the length of the entire keychain by half compared with other methods. In addition, the hash tree is used to verify the public key, so the key search time is reduced to log time. Finally, the major rate of change for a KSI network cluster node is [10], based on the experimental results, even if the number of nodes increases, there is no more overhead. Therefore, this proposed method works well and performs verification and merging from one-off signing and key usage.

$$K' = \frac{1}{N}\sum_{i=1}^{N} K_i$$

K_i: Input Node Key Length
K' : Number of hash chain conversion average key length
N : Number of experiment iterations

Table 4. Number of Changed Key Bit in Hash Values According To Experiments

Evaluation Item	Input Key length				
	256	512	1024	2048	4096
K'	64.396	64.724	64.066	64.097	64.019

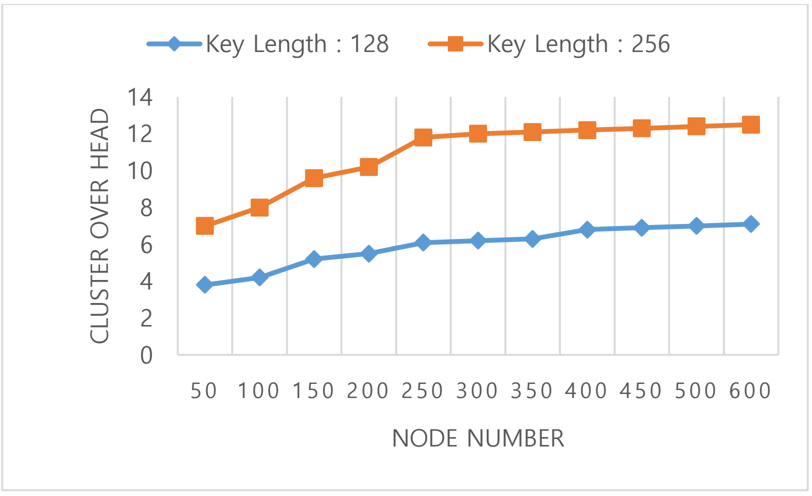


Figure 10. Graph of Comparison to Key Length Overhead with Proposed Scheme and Other Schemes

6. Conclusion

In this section is conclusion. Recent developments in ICT and the fourth industrial revolution in many major industrial countries are expanding into a super-connected society where data of various types and volumes are collected and processed. This increases the importance of cybersecurity, protecting data from legitimate user authentication and intrusion by intruders. However, existing digital signature schemes based on public keys have drawbacks such as high computational complexity, public key certificate issues, and periodic key updates. There are also security threats to existing public key cryptographic systems, such as the technological advances of quantum computers and show algorithms. Keyless digital signatures have been proposed to manage large amounts of data safely and efficiently. Keyless digital signature is characterized by high speed operation by hash operation by digital signature method using hash function and hash tree without using existing public key. There is no need for separate key management, and the validity of the key is not fixed, so you do not have to update it periodically.

To do this, a keyless digital signature generates a one-off key and a signature containing the root value by wrapping a one-off signature on the hash tree. KSI generates Keyless Digital Signature to generate signatures via certificates issued as OTP after user authentication, and global timestamps through block chaining through distributed network collaboration. In a keyless digital signature, the signer is secure against retransmission attacks using a one-time key, but a series of keys are pre-built and subsequently used so that a malicious attacker can preempt a key or receive a real estate transaction. The hash function is a defense against omnidirectional attacks and is the core of the KSI with a hash binary tree. In this paper, we propose a keyless digital signature scheme that improves overhead and traffic by using a KSI system based on a multipath hash chain.

So, we apply the hash chain and multipath hash chain to satisfy the security requirement of digital signature according to the proposed scheme analysis and improve the storage space and efficiency in the existing .Then, we prove that we merge the data while defending the existing attack and reducing the key change rate according to the proposed method analysis. And then, we applied an efficient and secure KSI system to minimize node overhead increase and analyzed it. We will draw conclusions by applying them to actual environment in the future and plan to develop them.

Acknowledgements

This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (2017-0-00156, The Development of a Secure Framework and Evaluation Method for Blockchain).

References

[1] R. C. Merkle, "A digital signature based on a conventional encryption function", Conference on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, **(1987)**.
[2] E. Dahmen, K. Okeya, T. Takagi and C. Vuillaume, "Digital Signatures Out of Second-Preimage Resistant Hash Functions", PQCrypto5299, **(2008)**, pp. 109-123.
[3] A. Buldas, A. Kroonmaa and R. Laanoja, "Keyless signatures' infrastructure: How to build global distributed hash-trees", In Nordic Conference on Secure IT Systems, **(2013)**, pp. 313-320.
[4] N. Emmadi and H. Narumanchi, "Reinforcing Immutability of Permissioned Blockchains with Keyless Signatures' Infrastructure", Proceedings of the 18th International Conference on Distributed Computing and Networking ACM, **(2017)**, pp.46.
[5] G.-C. Park, "A Multi-hash Chain Scheme for Ensure Data Integirty Nodes in Wireless Sensor Network", Journal of the Korea Institute of Information and Communication Engineering, vol. 14, no. 10, **(2010)**, pp. 358-2364.
[6] C. Jämthagen, and M. Hell, "Blockchain-based publishing layer for the Keyless Signing Infrastructure", In 13th IEEE International Conference on Advanced and Trusted Computing, IEEE--Institute of Electrical and Electronics Engineers Inc, 2016.
[7] J. A. Buchmann, E. Dahmen, S. Ereth, A. Hülsing and M. Rückert, "On the Security of the Winternitz One-Time Signature Scheme", Africacrypt, vol. 11, **(2011)**, pp.363-378.
[8] J. Buchmann, E. Dahmen, E. Klintsevich, K. Okeya and C. Vuillaume, "Merkle signatures with virtually unlimited signature capacity", In Applied Cryptography and Network Security, **(2007)**, pp. 31-45.
[9] J. Buchmann, E. Dahmen and A. Hülsing, "XMSS-a practical forward secure signature scheme based on minimal security assumptions", Post-Quantum Cryptography, **(2011)**, pp. 117-129
[10] D. Xiao, X. Liao and S. Deng. "One-way Hash function construction based on the chaotic map with changeable-parameter", Chaos, Solitons and Fractals, vol. 24, no. 1, **(2005)**, pp. 65-71
[11] M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical", In Advances in Cryptology-CRYPTO'97: 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 1997. Proceedings, **(1997)**, pp. 470.

Authors

Gyeong-Jin Ra, she received her B.S in Department of Computer Software Engineering, Soonchunhyang University (2015) and M.S in Department of Computer Science and Engineering, Soonchunhyang University (2015-Present).

Her Research Interests includes Cryptographic Application Protocols, Digital Signatures, Block Chain and Computer Security

Email: rababi@sch.ac.kr

Im-Yeong Lee, he received his BS in Department of Electronic Engineering, Hongik University (1981), MS in Communication Engineering, Osaka University (1986) and Ph.D in Communication Engineering, Osaka University (1989).

He is a Senior Researcher in Korea Electronics and Telecommunications Research Institute (1989~1994). He is also a Professor in Department of Computer Software Engineering, Soonchunhyang University since 1994 until present. His Research Interests includes Cryptography Theory, Information Theory, Computer Security.

Email: imylee@sch.ac.kr