Journal of the Korea Industrial Information Systems Research Vol. 20 No.2, Apr. 2015 :11-18
http://dx.doi.org./10.9723/jksiis.2015.20.2.011 ISSN:1229-3741

Secure and Scalable Key Aggregation Scheme for Cloud Storage

YoHan Park[1] and YoungHo Park[2]*

Abstract As the communication technology and mobile devices develop, the need for the efficient and secure remote storage is required. And recently, many companies support cloud storages to meet the requirements of the customers. Especially in the business field where various companies collaborate, data sharing is an essential functionality to enhance their work performance. However, existing researches have not fully satisfied the requirement either efficiency and security. This paper suggests efficient and secure data sharing scheme for cloud storage by using secret sharing scheme. Proposed scheme can be applied to business collaborations and team projects.

Key Words : Cloud Storage, Data Sharing, Key Aggregation

1. Introduction

Cloud computing is a promising technology for the next generation of IT enterprises. The advantages of this technology, such as convenient remote data access and usable on-demand application services, are very profitable commercial items to both IT companies and their customers. As the lucrative technologies are focused on mobile devices, the importance of cloud computing technology also is growing up rapidly. Especially cloud storage (CS) has many advantages in finance. Virtual storage in the cloud is cheaper and more efficient than the hard drive connected to the personal computer. So that, many communication companies are operating the CS, for example, Apple runs iCloud and Google operates Google Drive.

However many unsolved security problems in CS restrict the extension of cloud technology to various IT areas. The basic security problem is originated by the remote place to store user's data. Outsourced storages are vulnerable to attackers and even to service providers. Therefore, the data stored in the CS should be essentially encrypted before storing [1-2]. But as the data are encrypted, the CS is limited in scalability and efficiency. Moreover computation load is increased as well. Thus a service provider needs to consider both security and efficiency when designing CS.

Recently, data sharing is considered as one of the important function in CS[3-4]. For example, when members of a team which is organized in collaboration with different companies work together, they need to share their works. But they cannot meet frequently for restrictions. In this case, data sharing method using CS can be an excellent solution. Thus data sharing should be equipped in CS.

But because of limited computing and

* Corresponding Author
Manuscript received November 3, 2014 / Revised December 30, 2014 / Accepted February 25, 2015
1) Kyungpook National University, Department of Electronics Engineering, Lecturer, 1st Author
2) Kyungpook National University, School of Electronics Engineering, Professor, Corresponding Author (parkyh@knu.ac.kr)

energy-constrained devices, symmetric encryption techniques are applied to CS generally[5-6]. It means data-owners, who upload their data to CS, have to use different keys whenever they upload or verify data. It makes users share data inefficiently. Because data-owner have to give each different key to the other party when he or she requests data. C-K Chu et. al [4] used pairing cryptosystem to share data. Even though, their scheme provides data sharing, computation load is high because they used paring operation.

This paper suggests efficient and secure data sharing method in CS using the secret sharing scheme. We do not use paring operations considering mobile devices limited computing ability. The scheme helps to share less key than sharing every key when others want several data concurrently.

The rest of the paper is organized as follows. In Section 2, we survey the related works. Next we present secure and scalable key aggregation scheme in CS in Section 3, followed by a analysis in Section 4. The paper is finally concluded in Section 5.

2. Preliminaries

In this section, we introduce the framework for key-aggregate encryption which we focus on. Then we present a scenario of the application using key-aggregate encryption, the cryptographic system and notations used as building blocks.

2.1 Framework

A key-aggregate encryption scheme first introduced by C.-K. Chu et al. [4]. In this paper, we follow their process but slightly revise the contents in the framework.

- **Setup**$(1^{\lambda}, n)$: Data owner executes the Setup function to get a $(n-1)$ degree polynomial $f(x)$. On input security level parameter 1^{λ}, it outputs public system parameter $\widetilde{P}=(P_1,...,P_t)$ and hash function H_1 (where $P_t=(x_t, f(x_t))$ and $t \ll n$).
- **KeyGen**: Data owner executes the KeyGen function to get a secret key $P_k=(r, f(r))$ (where $k=f(r)$).
- **Encrypt**(k, m): Data owner executes the Encrypt function to encrypt data m. It outputs a ciphertext C.
- **Extract**: Data owner executes the Extract function to devise an aggregated key. It outputs a class i and extra points P_{ext} and a secret point $P_K=(x_K, f_2(x_K))$.
- **Decrypt**(i, P_K, C): Data receiver executes the Decrypt function to compute original data from encrypted data. On input a class i, a secret point P_K, and encrypted data C, it outputs the decrypted data m.

2.2 Scenario

In this paper, we consider data sharing model in CS. We refer and use the Fig. 1 used at [4]. Two users who are working different companies are team members of same project, thus they want to share their works using cloud system.

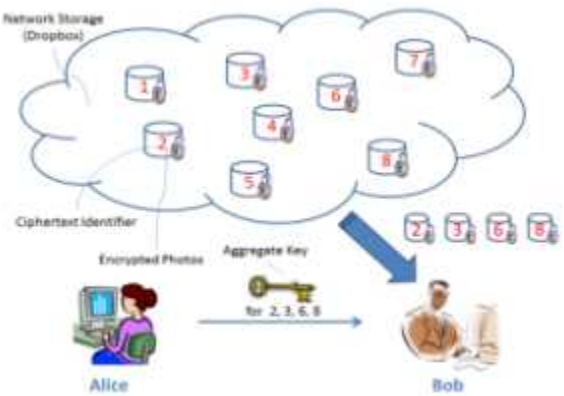

Fig. 1 Data Sharing Scenario using Key Aggregation Scheme[4].

We assume that Alice tries to send 4 data numbered by 2,3,6,8 among 8 data which are stored in CS to Bob. First, Alice encrypts 8 data using 8 different keys. and then uploads those to CS. Next, she sends 4 different keys to Bob using secure channel such as e-mail (the size of key is short enough to send using e-mail because the system uses symmetric cryptosystem). After receiving 4 different keys, Bob downloads 4 data from the CS, then decrypts using 4 different keys received using e-mail. Now Bob can check messages 2,3,6,8.

If the cloud system supports key aggregation scheme, Alice only sends one aggregated key which can decrypt 4 encrypted data rather than sending 4 keys. Likewise, our scheme provides efficient and secure key aggregate method, so that users are no need to send every keys what others want to decrypt.

2.3 Secret Sharing Scheme

Secret sharing schemes were independently introduced by the Blakley and the Shamir [7] in 1979. They introduced a way to split a secret K into n shares. And only t or more than t shares among n can reconstruct a secret K. It is called (t,n)-secret sharing, denoted as (t,n)-SS.

Shamir's (t,n)-SS. Shamir's (t,n)-SS is based on polynomial interpolation. The scheme consists of two algorithms:

① **Secret Sharing Generation** : A trusted party T distributes shares of a secret K to n users as follow:

- T chooses a prime $p > \max(K,n)$, and defines $a_0 = K$.
- T picks a polynomial $f(x)$ of degree $(t-1)$ randomly: $f(x) = a_0 + a_1 x + \cdots a_{t-1}x^{t-1}$, in which the secret $K = a_0 = f(0)$ and all coefficients $(a_0,\ldots,a_{t-1})$ are in a finite field $F_p = GF(p)$ with p elements.
- T computes $K_i = f(s_i) \pmod p$ for i=1,...,n. and securely transfer the shares K_i to each user.

② **Secret Reconstruction** : Any group of size t or more than t can reconstruct the polynomial $f(x)$ as

$$f(x) = \Sigma_{i\in A}\lambda_i(x)K_i \pmod q,$$

, where $A = \{1,\ldots,t\} \subseteq \{1,\ldots,n\}$, $\lambda_i(x) = \Pi_{j\in A\setminus i}\frac{s_j - x}{s_j - s_i}$ is called a Lagrange coefficient. The secret is recovered by $f(0) = K$.

For more information on this scheme, readers can refer to the original paper [7].

3. Key Aggregation Scheme for Cloud Storage

This section presents a key aggregation scheme for CS. We describe the basic construction (for one message) in Section 3.1 Next, we extend it to general form (up to N messages) in Section 3.2 and show a example of our scheme in Section 3.3.

Table 1 lists some important notations whose concrete meanings will be further explained.

3.1 Basic Construction

We assume Alice shares one message to Bob. Alice can encrypt at most $(n-t-1)$ messages. If she encrypt more than $(n-t-1)$ messages, adversaries who got $(n-t)$ key points can recover the master polynomial *f(x)*.

Table 1 Notations

p	large prime
$f(x)$	master polynomial of degree $n-1$ for a sender
$f_1(x)$	slave polynomial of degree n_1-1 for a single key
$f_2(x)$	slave polynomial of degree n_2-1 for an aggregated key
$\widetilde{P}$	t points passing through $f(x)$
P_t	the point (x_t, y_t)
i_a	the subset of $\widetilde{P}$, called the class
K_1	digital signature of message m generated by entity A
$Enc_{K_1}(x)/Dec_{K_1}(x)$	Encryption/Decryption of x
$H_1(x)$	hash function, mapping $Z_p^* \rightarrow \{0,1\}^t$, where t is length of key

- **Setup**
  - Alice generates a $(n-1)$ degree polynomial $f(x)$ as a master polynomial: $f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$.
  - Alice chooses and publishes random t points which pass through the polynomial $f(x)$ as public parameters. Let $\widetilde{P} = P_1, P_2, \ldots, P_t$, where $P_t = (x_t, y_t)$ and $(t \ll n)$.
  - Alice chooses hash function H_1 and publishes it.
- **KeyGen**
  - Alice chooses a random $r_1 \in Z_p^*$ and computes a k_1 : $f(r_1) = k_1$ and $K_1 = H_1(k_1)$.
  - Let $P_{k_1} = (r_1, k_1)$ be a key point.
- **Encrypt**
  - Alice chooses a message m_1 and encrypts using $K_1 : c_1 = Enc_{K_1}(m_1)$
  - Alice stores a ciphertext $C_1 = (r_1, c_1)$ to the CS.
- **Extract**
  - Alice chooses random $(n_1 - 1)$ points among $\widetilde{P}$, where $(2 < n_1 < t+1)$. A set of $(n_1 - 1)$ points is indexed by the class i_1. Then Alice publishes the class i_1 as the corresponding class of the message m_1.
  - Alice generates the $(n_1 - 1)$ degree polynomial $f_1(x) = b_0 + b_1 + \cdots + b_{n-1} x^{n-1}$. $f_1(x)$ is composed of $(n_1 - 1)$ points indexed by the class i_1 and the key point P_{k_1}.
  - Alice computes $Auth = H_1(b_0)$.
  - Alice chooses one secret point $P_K = (x_K, y_K)$ passing through $f_1(x)$ randomly, and sends $(Auth, i_1, P_K)$ to Bob securely.
- **Decrypt**
  - Bob reconstructs the polynomial $f_1'(x)$ using $(n_1 - 1)$ points which are indexed by the i_1 and the secret point P_K received from Alice.
  - Bob checks the validity of the polynomial $f_1'(x)$: $H_1(f_1'(0)) = Auth$. If this is incorrect, stop processing. If this is correct, then Bob computes $k_1' = f_1'(r_1)$ and $K_1' = H_1(k_1')$
  - Bob decrypts a message $m_1 = Dec_{K_1'}(c_1)$ (Alice can compute k_1 easily without reconstructing $f_1(x)$ by using her master polynomial $k_1 = f(r_1)$.

3.2 Extension to N messages

We assume Alice encrypts and stores M messages. And then Alice sends N out of M messages to Bob.

- **Setup**
  Same as above.
- **KeyGen**
  - Alice chooses random $r_a \in Z_p^*$ and computes k_a : $f(r_a) = k_a$ and $K_a = H_1(k_a)$, where $(1 < a < M)$.
  - Let $P_{k_a} = (r_a, k_a)$ be key points.
- **Encrypt**
  - Alice chooses messages m_a and encrypts using K_a : $c_a = Enc_{K_a}(m_a)$.
  - Alice stores ciphertexts $C_a = (r_a, c_a)$ to the CS.
- **Extract**
  - Alice chooses random $(n_2 - N)$ points among $\widetilde{P}$, where $(N+1 \le n_2 \le N+t)$. A set of $(n_2 - N)$ point is indexed by the class i_a. Then Alice publishes the class i_a as the corresponding class of messages m_a.
  - Alice generates the $(n_2 - 1)$ degree polynomial $f_2(x) = c_0 + c_1 + \cdots + c_{n_2-1}x^{n_2-1}$. $f_2(x)$ is composed of $(n_2 - N)$ points indexed by the i_a and key points P_{k_a}.
  - Alice chooses arbitrary extra N points, $< P_{ext_1}, \ldots, P_{ext_N} >$, passing through $f_2(x)$. Then, Alice sets one point among extra N points as the secret point $P_K = (x_K, y_K)$. Then Alice publishes other $(N-1)$ points by indexing i_{ext}, i.e. $P_K \notin i_{ext}$.
  - Alice computes $Auth = H_1(c_0)$.
  - Alice sends $(Auth, i_a, i_{ext}, P_K)$ to Bob securely.
- **Decrypt**
  - Bob reconstructs the polynomial $f_2'(x)$ using $(n_2 - N)$ points indexed by i_a, $N-1$ points indexed by i_{ext}, and the secret point P_K
  - Bob checks the validity of the polynomial $f_2'(x)$: $H_1(f_2'(0)) = Auth$. If it is incorrect, stop processing. If this is correct, then Bob computes $k_a' = f_2'(r_a)$ and $K_a' = H_1(k_a')$.
  - Bob decrypts messages $m_a = Dec_{k_a'}(c_a)$ (Alice can compute k_a easily without reconstructing $f_2(x)$ by using her master polynomial $k_a = f(r_a)$)

3.3 An Example of the Proposed Scheme

We assume Alice wants to share 3 data to Bob among 6 data in the CS. Let $m = 6$ be total messages, $N = 3$ be sharing messages, $t = 5$ be the number of public parameters, $n = 10$ be the degree of the master polynomial, and $n_2 = 6$ be the degree of the slave polynomial. Among 6 messages, Alice wants to send m_1, m_2, m_3.

- **Setup**
  - Alice generates 9 degree master polynomial $f(x)$: $f(x) = a_0 + a_1 x + \cdots a_9 x^9$
  - Alice publishes 5 public parameters passing through the polynomial $f(x)$. Let $\widetilde{P} = \{P_1, \ldots, P_5\}$, where $P_1 = (x_1, y_1)$
- **KeyGen**
  - Alice chooses 5 random numbers, $r_1, \ldots, r_5$ and computes $k_1, \ldots, k_5$, where $k_1 = f(r_1)$.
  - Alice generates 6 keys, $P_{k_1}, \ldots, P_{k_6}$ to encrypt messages. Let $P_{k_1} = (r_1, k_1)$ be a point of key.
- **Encrypt**
  - Alice encrypts 6 messages, $m_1, \ldots, m_6$, using 6 keys generated the previous step. Let $c_1 = Enc_{k_1}(m_1), \ldots, c_6 = Enc_{k_6}(m_6)$ be ciphertexts $C_1, \ldots, C_6$, where $C_1 = (r_1, c_1)$.
  - Alice stores 6 ciphertexts to the CS.
- **Extract**
  - Alice chooses random 3 points among $\widetilde{P}$. To simplify, we selects P_1, P_2, P_3 among $\widetilde{P}$ and indexes i_1 and publishes those as the corresponding class of m_1.

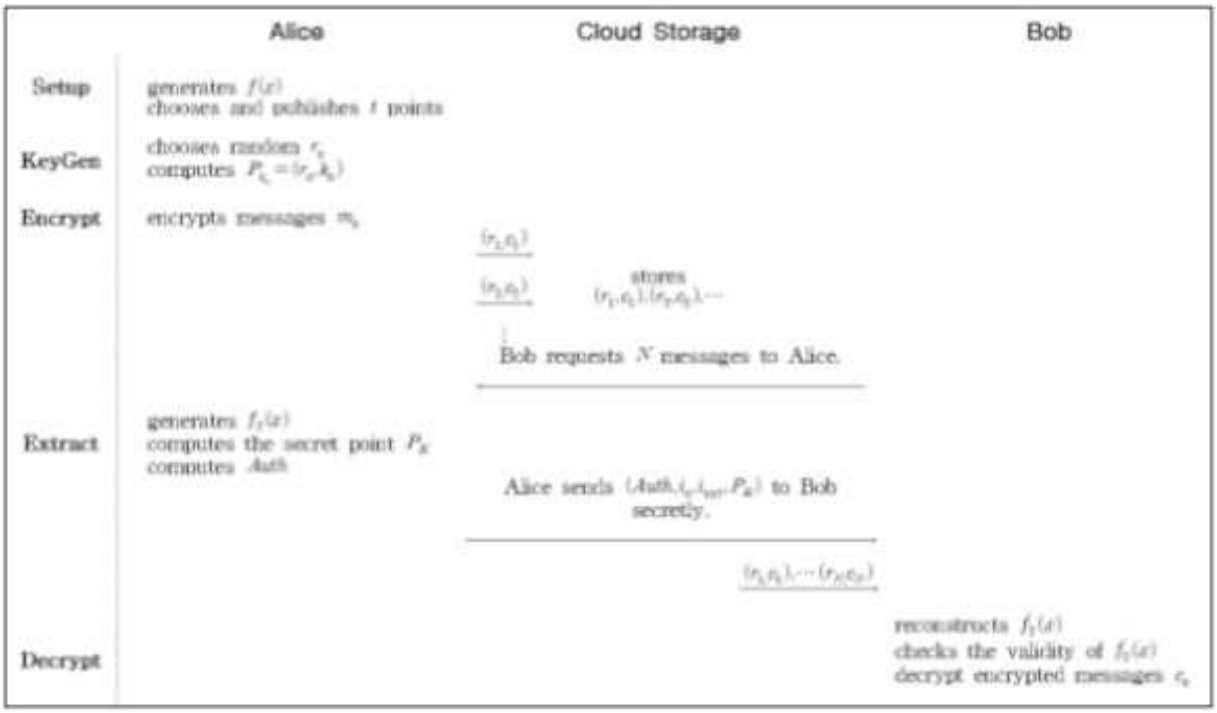

Fig. 2 The Proposed Key-Aggregation Scheme.

- Alice generates 5 degree polynomial $f_2(x)$ using the class i_1 (3 points) and $P_{k_1}, P_{k_2}, P_{k_3}$.
- Alice chooses 3 other extra points passing through $f_2(x)$. Among 3 extra points, set 1 point as the secret key P_K. Then Alice publishes other 2 points, P_{ext_1}, P_{ext_2}.
- Alice sends $(Auth, P_K)$to Bob secretly using e-mail.

- **Decrypt**
  - Bob reconstructs $f_2'(x)$ using the class i_1 and 2 extra points P_{ext_1}, P_{ext_2}, and secret key P_K.
  - Bob checks the validity of the reconstructed polynomial $f_2'(x)$ by comparing $H_1(f_2'(0)) = Auth$.
  - Bob computes $K_1 = H_1(f_2'(r_1))$, $K_2 = H_1(f_2'(r_2))$, $K_3 = H_1(f_2'(r_3))$.
  - Bob gets ciphertexts C_1, C_2, C_3 for the CS and decrypts the messages using K_1, K_2, K_3.

Fig. 2 shows the overview of our proposed scheme.

4. Analysis

We analyze the security in Section 4.1 and performance in Section 4.2.

4.1 Security

We use the secret sharing scheme to share data efficiently and securely. Our scheme is theoretically secure if adversaries have no more than $(n-1)$ points or (n_2-1). Because they cannot reconstruct the master and slave polynomials with the information that they have.

- **Cloud Service Provider (CSP)**

CSP is also a promising attacker[3]. To provide the privacy of user's data, data should be encrypted

before storing at CS. To attack and decrypt the data, CSP has to know each secret key or aggregated key or have enough information to reconstruct the master and slave polynomials. However, each key P_{k_m} encrypting each data is not disclosed during the communication. And though CSP can have n_2-2 key points which are passing through $f_2(x)$, it cannot get a secret key P_K which is passing through a secure channel to another user. So that CSP cannot know any information of the data.

- User

Users who get a secret key P_K have more information than the CSP. However this secret key is only valid on the requested data. And the points which were used to construct $f_2(x)$ is independent to other slave polynomials which are going to be made for other aggregated keys. So that using this secret key P_K, the users cannot decrypt other encrypted data stored in CS.

4.2 Performance

We compare our scheme with [4]. C.-K. Chu et al. designed the scheme based on tree structure. They classified data depend on certain criteria. They called it as a class. Same class means the data are branches of a mother point. If a receiver requests data in a same class, the performance is nice, unless computation load is high. Therefore, the performance of [4] is dependent on the class However, our scheme does not consider class. Thus the performance is independent on the class and constant.

Table 2 shows computation load of each scheme. For simplicity, we only consider the decryption part, because service providers support encryption and secret distribution, users are no need to consider these computation loads.

Table 2 Comparison of Computation Load

	[4]	Proposed Scheme
structure	tree-based	independent
Decryption load (Simplied Form)	$2T_P+3T_M$	$2T_M+T_E$

T_P : pairing operation, T_M : scalar multiplication, T_E : symmetric encryption

[4] used pairing operations to encrypt and decrypt data. The pairing operation generally consume heavy computation loads rather than scalar multiplications. Cao et al. [8] showed a pairing consumes about double computation load with those of an exponentiation and three times computation load with those of scalar multiplication in G_1. Our scheme used only scalar multiplications and encryption scheme. We can use any secure and simple symmetric encryption method based on application areas. According to this comparison, our scheme is lighter than [4] and flexible in terms of structures.

5. Conclusion

Protecting users's privacy and data are important requirements to use CS in business fields. And recently, efficient and secure data sharing method also is required in cooperated work for enhancing work performance.

We suggested a secure and efficient data sharing scheme in cloud storage. To share the date efficiently, we suggested key aggregation scheme. Data-owner is no need to send every corresponding symmetric keys although receivers request several data. Instead, data-owner generates one aggregated key which is related to requested data. Then receivers can decrypt several encrypted data stored in CS using this aggregated key. The aggregated key is sent through a secure channel

from the data-owner to the receiver who requests data, so that CSP and other users cannot get enough information to decrypt data.

Compared to previous works, we do not arrange data in a hierarchy to make a aggregated key, but we select and generate a slave polynomial. Thus our scheme is flexible than hierarchical key aggregation scheme and the performance is independent on tree structures. Thus proposed scheme is suitable for business fields or group projects.

References

[1] S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu, "SPICE-Simple Privacy-Preserving Identity-Management for Cloud Environment," ACNS 2012, LNCS 7341, pp. 526-543, 2012.

[2] U. Somani, K. Lakhani, and M. Mundra, "Digital Signature with RSA Encryption Algorithm to Enhance the Data Security of Cloud in Cloud Computing," 2010 1st International Conference on Parallel, Distributed and Grid Computing, pp. 211-216, 2012.

[3] G. Zhaio, C. Rong, J. Li, F. Zhang, and Y. Tang, "Trusted Data Sharing over Untrusted Cloud Storage Providers," IEEE International Conference on Cloud Computing Technology and Science, pp. 97-103, 2010

[4] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage," IEEE Transactions on Parallel and Distributed Systems, Vol 25, No.2, pp. 468-477, 2014.

[5] S. Kamara and K. Lauater, "Cryptographic Cloud Storage," Financial Cryptography and Data Security 2010, LNCS 6054, pp. 136-149, 2010.

[6] L. Arockiam1 and S. Monikandan, "Data Security and Privacy in Cloud Storage using Hybrid Symmetric Encryption Algorithm," International Journal of Advanced Research in Computer and Communication Engineering, Vol 2, No.8, pp. 3064-3070, 2013.

[7] A. Shamir, "How to share a secret," Communications of the ACM, Vol 22, No.11, pp. 612 - 613, 1979.

[8] X. Cao, W. Kou, and X. Du, "A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges," Information Sciences, Vol. 180, No.15, pp. 2895 - 2903, 2010.

박 요 한 (YoHan Park)

- 정회원
- 2006년 2월: 경북대학교 전자전기컴퓨터 학부 학사
- 2008년 2월: 경북대학교 전자공학과 석사
- 2008년 3월~2013년 2월: 경북대학교 전자전기컴퓨터 학부 박사
- 2013년~2014년: National University of Singapore 박사후연구원
- 2014년~현재: 경북대학교 산업전자공학과 시간강사
- 관심분야: 정보보호, 무선통신보안, 네트워크보안

박 영 호 (YoungHo Park)

- 종신회원
- 1989년 2월: 경북대학교 전자공학과 학사
- 1991년 2월: 경북대학교 전자공학과 석사
- 1995년 8월: 경북대학교 전자공학과 박사
- 1996년~2008년: 상주대학교 전자전기공학부 교수
- 2003년~2004년: Oregon State Univ. 방문교수
- 2008년~2014년: 경북대학교 산업전자공학과 교수
- 2014년~현재: 경북대학교 전자공학부 교수
- 관심분야: 정보보호, 네트워크보안, 모바일 컴퓨팅