https://doi.org/10.7840/kics.2023.48.8.920

통신환경을 이용한 CKKS 동형암호 암호문 초기화 기법

채 승 재•, 이 준 우*, 이 용 우**, 노 종 선***

Ciphertext Refresh Using Communication Cost on CKKS Fully Homomorphic Encryption

Seungjae Chae•, Joon-Woo Lee*, Yong-Woo Lee**, Jong-Seon No***

요 약

CKKS(Cheon-Kim-Kim-Song) 완전 동형 암호(Fully-Homomorphic Encryption, FHE)는 현재 연구자들의 가장 큰 관심을 받고 있다. 완전동형암호를 활용할 수 있는 많은 분야가 있지만, 그중에 연구가 많이 진행되고 있는 정보보호 머신러닝에서는 현재 완전 동형 암호 외에도 다자간 계산(Multi-Party Computation, MPC)를 이용하거나 혹은 둘을 혼합한 hybrid한 방식 등 다양한 방식으로 연구가 진행되고 있다. 본 논문에서는 기존 CKKS 완전 동형 암호를 사용하는 경우 가장 많은 연산시간이 소요되는 bootstrapping과정을 통신환경을 이용할 수 있을 때, 랜덤한 값을 생성하여 통신을 통해 암호문 초기화 과정을 설계하고, 이 과정으로 부트스트래핑을 대체할 수 있음을 제시한다.

Key Words : Communication, fully-homomorphic encryption, CKKS, bootstrapping, cryptography

ABSTRACT

Cheon-Kim-Kim-Song (CKKS) fully-homomorphic encryption (FHE) is currently receiving the most attention from researchers in this research area. There are many fields in which fully homomorphic encryption can be utilized. In privacy-preserving machine learning, where a lot of research is being conducted, not only using FHE, but also Multi-Party Computation (MPC) is used or a hybrid method of the two algorithms is used. In this paper, the bootstrapping process, which takes the most computation time when using CKKS FHE is replaced using ciphertext refresh when using communication cost.

Ⅰ. 서 론

최근 기계학습에 대한 연구가 비약적으로 발전하면서 데이터를 암호화된 상태로 인공지능에 적용하는 기술이 매우 빠르게 연구가 진행되고 있다. 클라이언트는 자신의 데이터를 유출하지 않은 채로 안전하게 데이터 분석에 대한 서비스 제공을 받고 싶어하고, 서버 역시 모든 사용자의 데이터를 암호화된 상태로 기계학습을 진행하여야 한다. 이러한 상황에서 사용되는 완전 동형암호(Fully Homomorphic Encryption, FHE)를 이용하

※ 본 연구는 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행되었습니다. (No.2021-0-00400, 저사양 디바이스 대상 고효율 PQC 안전성 및 성능 검증 기술 개발)

• First Author : Department of Electrical and Computer Engineering, INMC, Seoul National University, chae950104@ccl.snu.ac.kr, 학생회원

* Department of Computer Science and Engineering, Chung-Ang University, jwlee2815@cau.ac.kr, 정회원

** Department of Information and Communication Engineering, Inha University, yongwoo@inha.ac.kr, 정회원

*** Department of Electrical and Computer Engineering, INMC, Seoul National University, jsno@snu.ac.kr, 종신회원

논문번호 : 202304-080-A-RU, Received April 18, 2023; Revised May 19, 2023; Accepted May 24, 2023

는 방식과 다자간 계산 (Multi-Party Computation, MPC)를 이용하는 방식 그리고 마지막으로 둘의 단점을 보완하기 위하여 각 특징을 섞어서 진행하는 hybrid한 방식도 연구가 진행되고 있다.

두 방식은 각각의 장단점을 갖고 있는데, 먼저 완전 동형암호는 서버가 모든 유저의 암호화된 데이터로 연산을 진행하는 점에서 서버의 연산량이 과중하다는 단점을 갖고 있지만, 다자간 계산은 유저간의 통신이 포함된다는 점과 연산이 필요할 때마다 온라인 상태를 유지해야한다는 단점을 갖고 있다[12]. 완전동형암호를 사용할 때 가장 큰 문제점은 주기적으로 부트스트래핑을 수행하여야 하고 그의 수행 시간이 매우 길다는 것인데 그 문제를 해결하기 위해 본 논문에서 통신 환경을 이용하여 서버와 클라이언트 간의 통신을 통해 암호문을 추가 초기화 시키고 이 과정이 부트스트래핑을 대체할 수 있음을 실험적으로 확인하였다.

Ⅱ. 배경 이론

2.1 완전 동형 암호

동형 암호는 암호화된 데이터를 복호화하는 과정을 거치지 않고 검색, 통계처리 등의 연산처리가 가능하게 하는 암호화 기술이다. 최근 많이 사용되는 클라우드 컴퓨팅 환경에 적용하여 정보유출을 방지할 수 있고, 사용자가 메시지를 암호화하여 서버로 전송하면 서버는 암호화된 상태에서 계산을 수행하고, 사용자에게 다시 전송한 뒤 마지막으로 사용자는 받은 암호문을 복호화하는 과정을 거친다. 이 과정에서 어떠한 정보의 유출 없이 덧셈과 곱셈에 대해 암호문 연산이 수행 가능하면 두 연산을 통해 필요한 대부분의 연산이 수행가능하게 되는데, 이를 완전 동형 암호라고 부른다.

다만 연산이 진행될수록 에러 값이 증폭된다는 점에서 문제를 갖고 있는데, 2009년에 Gentry가 부트스트래핑 개념을 제안함으로써[3,7] 완전 동형 암호가 계속하여 연산을 진행해도 문제가 없을 보였다. 다음의 2.2에서는 현재 가장 많이 알려지고 연구가 진행되는 완전 동형 암호 알고리즘인 CKKS에 대하여 소개한다.

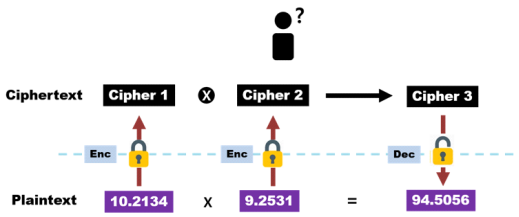

그림 1. 완전 동형 암호
Fig. 1. Fully Homomorphic Encryption

2.2 CKKS 알고리즘

CKKS(Cheon-Kim-Kim-Song)알고리즘은 2017년에 발표된 완전 동형 암호 알고리즘[1,6]으로 lattice-based hard problem에 기반하여 안전하고, 실수 연산이 동형적으로 가능한 알고리즘이다. $R_q = Z_q[X]/(X^N+1)$이라 하고 ring-LWE sample $(b,a)\in R_Q^2$는 공개 키이고, $b+a \bullet s = e$를 만족한다. 이때 암호문은 아래와 같이 암호화 된다.

$$ct = (b',a') = v \bullet (b,a)+(m+e_o,e_1)\in R_Q^2 \quad (1)$$

v,e_1,e_2는 Gaussian random polynomial이다. CKKS 알고리즘의 동형적 연산은 아래와 같이 진행된다.

덧셈은 아래의 (2)와 같이 진행된다.

$$Enc(x_0,x_1,\cdots,x_n)\oplus Enc(y_o,y_1,\cdots,y_n) = Enc(x_o+y_o,x_1+y_1,\cdots,x_n+y_n) \quad (2)$$

그리고 곱셈은 (3)과 같이 진행된다.

$$Enc(x_0,x_1,\cdots,x_n) \bullet Enc(y_o,y_1,\cdots,y_n) = Enc(x_o \bullet y_o,x_1 \bullet y_1,\cdots,x_n \bullet y_n) \quad (3)$$

Rescaling 과정은 두 개의 real number 데이터를 곱셈한 후에 해줘야하는 과정으로 아래의 (4)와 같이 진행된다.

$$(\triangle \bullet m_1) \bullet (\triangle \bullet m_2) = \triangle^2 \bullet m_1 \bullet m_2 \Rightarrow \triangle \bullet m_1 \bullet m_2 \quad (4)$$

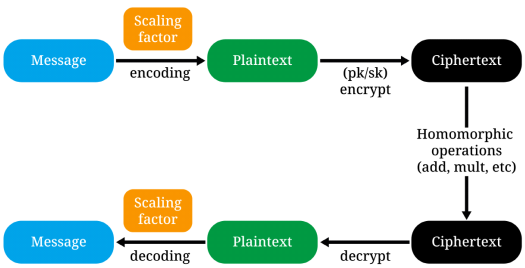

그림 2. CKKS 완전 동형 암호 알고리즘 암호화 복호화 과정
Fig. 2. Encryption and decryption process of CKKS fully homomorphic encryption

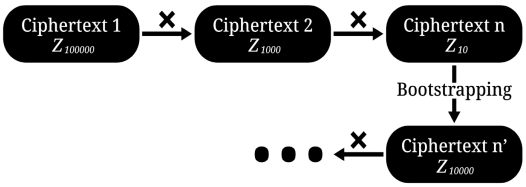

그림 3. CKKS 부트스트래핑 과정
Fig. 3. Bootstrapping process of CKKS scheme

CKKS 알고리즘에서 암호문 레벨이란 bootstrapping을 하지 않고 가능한 곱셈의 수이고 depth는 연속으로 가능한 최대의 곱셈 횟수 이고 암호문 모듈러스는 곱셈 depth의 capacity이다. 연산을 여러번 진행함에 따라 암호문 모듈러스가 작아지고, 위에서 언급한 부트스트래핑을 통해 다시 연산을 진행할 수 있도록 해줘야한다. 또한 large depth가 필요한 환경에서는 부트스트래핑이 많이 시행되어야 하는 문제가 발생한다.

2.3 정보보호 머신러닝

완전 동형 암호가 사용되는 분야들 중에 암호화된 데이터를 다루는 정보보호 머신러닝의 중요성이 강조되고 있다. 과거 동형암호를 사용하는 정보보호 머신러닝 연구들은 동형암호 연산에 맞게 수정된 HE-friendly한 네트워크를 사용했다. 이러한 경우에는 얕은 네트워크만 가능하고(약3 ~ 11레이어), ReLU와 같은 activation function은 low-degree polynomial만 사용가능했다. 2022년에 [2]이 완전 동형 암호를 사용하여 pre-trained data를 inference하였고, 본 논문에서도[2]의 실험 결과에서 가장 문제가 되는 bootstrapping cost를 줄이는 데에 초점을 두고 연구를 진행하였다. 아래의 표 1에서 정보보호 머신러닝에 사용되는 다양한 경우들의 특징과 장단점을 확인할 수 있다.

표 1. 정보보호 머신러닝의 다양한 케이스 비교
Table 1. Comparison between several cases of PPML

	HE-friendly	FHE	HE+MPC
Accuracy	Low	High	High
Communication Cost	Low	Low	High
Model Exposure	No	No	Partially Yes
Re-training	Need	No	No
Server Computation	Low	High	Low

Ⅲ. 통신환경을 이용한 암호문 초기화 과정

완전 동형 암호에서 임의의 연산을 수행하기 위해서는 bootstrapping 과정이 반드시 접목되어야 한다. 특히 ResNet과 같은 deep neural network에 완전 동형 암호를 적용한 상태로 Inference를 수행하기 위해서는 수많은 bootstrapping 연산이 필요하게 된다. 한 번의 bootstrapping은 현재 구현된 라이브러리상에서 시스템에 따라 다르나 대략 300여초의 시간이 소요가 된다. 이로 인해 ResNet-20에서 CIFAR-10이라는 비교적 단순한 이미지 하나를 분류하기 위해서는 3시간 이상의 시간이 소요가 된다. 따라서 만일 bootstrapping을 다른 방법으로 대체할 수가 있다면 완전 동형 암호를 사용한 정보보호 머신러닝의 성능을 크게 향상시키는 것이 가능하다.

위와 같은 관점에서 본 논문에서는 완전 동형 암호의 bootstrapping을 서버에서 수행하는 것이 아닌 클라이언트에서 통신을 이용하여 암호문 refresh과정을 암호문의 복호화 및 재 암호화 과정을 통하여 수행하는 것이 가능하다. 본 논문에서는 암호문의 재 암호화 과정[11]을 단지 다른 키를 이용한 재 암호화가 아닌 통신환경을 이용하여 bootstrapping 연산을 줄이는 아이디어를 제시한다. 이 과정에서 서버는 클라이언트에게 자신의 인공신경망 모델에 대한 정보를 제공하면 안되고, 클라이언트 역시 서버에게 자신의 데이터를 공개하지 않은 상태로 진행되어야 한다. 아래의 그림 1과 같이 서버가 랜덤한 값을 생성해준 뒤, 암호화를 하여 초기화 시켜야 할 대상인 레벨이 낮은 암호문에 더해준다. 그 후에 클라이언트에 전송하고, 그것을 받은 클라이언트는 복호화를 해주고 다시 재 암호화를 해준다. 재 암호화된 레벨이 높은 암호화를 서버에 전송해주고, 서버는 본인이 생성했던 랜덤 값을 암호문에서 빼줘서 refresh과정이 진행된다.

이 과정에서 부트스트래핑에 소모되는 level 값을 고려해야한다. 기존의 연구들에서는 부트스트래핑에 10개 이상의 레벨을 소모하게 되지만[8-10] 본 논문에서 제안한 방식에는 레벨 소모가 없게 된다. 그러한 이유로 부트스트래핑을 사용할 때에 비해서 더 많은 연산을 수행하는 것이 가능하다. 또한 낮은 레벨의 암호문에 대해서 연산을 진행하는 것이 높은 레벨의 암호문에서 진행하는 것보다 더 빠른 속도로 연산이 가능하기 때문에 더 정보보호 머신러닝을 최적화 시킬 수 있다.

기존 방식의 레벨 소모와는 다르게 두 가지 형태로 레벨을 다룰 수 있다. 하나는 한 번의 리프레시 과정에 하나의 레이어를 사용하는 방식으로 레벨이 18 사용되

는 방식이고 다른 방식은 한 번의 리프레시 과정에 두 개의 레이어를 사용하고 레벨이 35를 사용한다. 기존의 방식에선 ReLU에 레벨 15, convolution에 레벨2, 그리고 bootstrapping에 레벨이 14 소모되는데, 이를 풀어쓰면 bootstrapping modraise과정에 31 그 후에 slottocoeff, modular reduction, 그리고 coefftoslot 과정 후에 레벨 17이 된다. 이 과정을 새로 제안한 두 가지 방식과 함께 풀어쓰면 아래의 표 2와 같이 된다. 처음 방식은 통신 횟수가 2배이지만 통신량은 절반이기 때문에 두 방식의 통신비용은 거의 동일하게 된다. 첫 번째 방식은 대역폭이 제한된 상황에서 이득을 볼 수 있고, 두 번째 방식은 통신을 할 때마다 해당 클라이언트가 온라인 상태여야 하는 단점이 존재하고 낮은 레벨의 암호문을 다루는 첫 번째 방식 위주로 실험을 진행하였다. 첫 번째 방식의 레벨을 맞춰주기 위해 서버가 리프레시된 암호문을 받은 뒤에 레벨을 낮춰주는 과정을 포함시켰고, 레벨 30을 사용할 때에 1,551초 정도 걸리는 과정을 722초까지 줄일 수 있었고, 더 최적화를 시켜 최종적으로 526초까지 낮출 수 있었다. 기존의 방식과는 다르게 통신량이 추가되는데 이 값은 총 리프레시 하는 횟수*(레벨이 낮은 암호문의 크기 + 레벨이 높은 암호문의 크기)가 된다.

표 2. 기존의 레벨 소모 방식과 제안된 레벨 소모 방식
Table 2. Level consumption method of bootstrapping and ciphertext refresh

	Level Consumption(ReLU = 15, Conv = 2, Boot = 14)
Original	31 - 0 - 17 - 0 - 17 - 0 ...
Proposed 1	18 - 1 - 18 - 1 - 18 - 1 ...
Proposed 2	35 - 1 - 35 - 1 - 35 - 1 ...

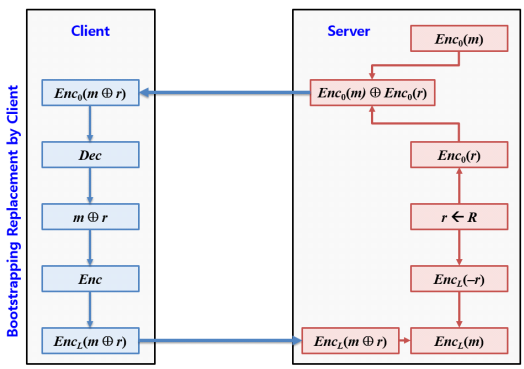

그림 4. 통신환경을 이용한 암호문 초기화 과정
Fig. 4. Ciphertext refresh using communication

Ⅳ. 실 험

이번 제안된 방식은 CIFAR-10 dataset[4]을 이용하여 ResNet에 적용하였다. 테스트엔 총 1,000개의 이미지를 사용하였다. AMD Ryzen threadripper PRO 3995WX(512GB RAM 포함)에서 SEAL 라이브러리[5]를 사용하였고, Ubuntu 20.04 시스템에서 진행하였다. Pre-trained된 파라미터는 ResNet-20/32/44에 사용되었고, polynomial degree $N = 2^{16}$, 그리고 full slots $n = 2^{15}$을 사용하였다. 레벨 18부터 시작하여 레벨이 1이 되면 암호문 리프레시를 진행하였다. 아래의 표3은 하나의 이미지에 걸리는 inference시간을 요약한 것이고, 이전 부트스트래핑을 사용하는 연구에 비해 약 4 ~ 5배 정도의 시간단축을 보임을 확인하였다. Resnet 모델에 따라 레이어 개수가 다르고, 그로 인해 부트스트래핑 대체 횟수가 다른데, 교체 횟수가 증가 함에 따라 줄일 수 있는 시간도 더 커지는 것을 확인 할 수 있었다.

또한 1,000개의 이미지에 대하여 정확도를 측정해봤을 때 기존 연구에 비해 accuracy 역시 큰 변화 없이 잘 수행됨을 확인할 수 있었다.

표 3. 하나의 이미지에 대한 inference 시간
Table 3. Classification runtime for one image using ResNet model

Model	Bootstrapping	Ciphertext Refresh
Resnet-20	2,271s	526s
Resnet-32	3,730s	850s
Resnet-44	5,224s	1,167s

표 4. 1,000개의 이미지 분류에 대한 정확도
Table 4. Classification accuracy of CIFAR-10 images using ResNet model

Model	Bootstrapping	Ciphertext Refresh
Resnet-20	91.31%	91.3%
Resnet-32	92.4%	92.7%
Resnet-44	92.65%	92.9%

Ⅴ. 결론 및 개선 사항

본 논문에서는 제안한 통신환경을 이용한 암호문 초기화 기법을 통해 기존 완전 동형 암호의 부트스트래핑을 대체하였고, 실험 시간을 줄일 수 있었다. 랜덤한 값을 더해서 사용자에게 보내준 후 유저가 복호화 및

재 암호화를 통해 암호문의 레벨을 올려주어 부트스트래핑 효과를 대체할 수 있었고, 기존 연구들에 비해 시간 및 정확도 측면에서도 좋은 성능을 유지하면서 pre-trained 네트워크에서도 사용가능한 것을 확인하였다. 하지만 추가적으로 개선해야할 점들이 많은데 첫 번째로 기존의 연구에서는 부트스트래핑 때문에 파라미터 값이 $N=2^{16}, n=2^{15}$으로 고정적으로 진행하였지만, 이를 더 줄인 상황에서도 사용 가능하다. 이 파라미터 값은 위 실험에서 통신 비용에 직접적으로 영향을 미치는 값으로 전체 통신량을 절반 정도 줄일 수 있는 장점을 기대 할 수 있다. 또한 정보보호 머신러닝 연구들에서 GPU가속을 이용하여 더 좋은 실험 결과들을 얻은 것들이 있지만 본 연구에서는 CPU만으로 실험을 진행하여 추후 연구에서 더 좋은 실험 결과를 기대할 수 있을 것이다. GPU 가속을 사용한다면, 현재 CPU에서의 실험 결과보다 통신의 연산 계산량 비중이 더 낮아지게 되는 점이 있지만 본 논문 아이디어의 효과는 유효하다. 또한, 현재 GPU를 기반으로 하는 신뢰도가 높은 라이브러리가 없다는 점에서 CPU만 사용하는 경우보다 구현상 고려해야 할 것이 더욱 많아 중요한 future work로 수행하려 한다.

References

[1] J. H. Cheon, et al., "Homomorphic encryption for arithmetic of approximate numbers," *Int. Conf. Theory and Appl. Cryptology and Inf. Secur.*, Springer, Cham, 2017. (https://doi.org/10.1007/978-3-319-70694-8_15)

[2] E. Lee, et al., "Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions," *Int. Conf. Mach. Learn. PMLR*, pp. 12403-12422, 2022.

[3] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. Forty-first Annu. ACM Symp. Theory Comput.*, pp. 169-178, 2009. (https://doi.org/10.1145/1536414.1536440)

[4] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Apr. 2009.

[5] *https://github.com/Microsoft/SEAL*, Nov. 2020. Microsoft Research, Red-mond, WA.

[6] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "A full RNS variant of approximate homomorphic encryption," in *Sel. Areas in Cryptography–SAC 2018: 25th Int. Conf.*, Calgary, AB, Canada, Aug. 2018, Revised Selected Papers 25 (pp. 347-368), Springer International Publishing, 2019. (https://doi.org/10.1007/978-3-030-10970-7_16)

[7] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," *Advances in Cryptology–CRYPTO 2013: 33rd Annu. Cryptology Conf.*, Santa Barbara, CA, USA, Aug. Part I. Springer Berlin Heidelberg, 2013. (https://doi.org/10.1007/978-3-642-40041-4_5)

[8] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No, "Precise approx-imation of convolutional neural networks for homomorphically encrypted data," *arXiv preprint arXiv:2105.10879*, 2021. (https://doi.org/10.48550/arXiv.2105.10879)

[9] J.-W. Lee, et al., "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30039-30054, 2022. (https://doi.org/10.1109/access.2022.3159694)

[10] J.-W. Lee, et al., "High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function," *Advances in Cryptology–EUROCRYPT 2021: 40th Annu. Int. Conf. Theory and Appl. Cryptographic Techniques*, Zagreb, Croatia, Oct. 2021, Proceedings, Part I 40. Springer International Publishing, 2021. (https://doi.org/10.1007/978-3-030-77870-5_22)

[11] A. Al Badawi, et al., "OpenFHE: Open-source fully homomorphic encryption library," in *Proc. 10th Wkshp. Encrypted Computing & Applied Homomorphic Cryptography*, 2022. (https://doi.org/10.1145/3560827.3563379)

[12] S.Chae, J.-S. No, "Usage of fully homomorphic encryption and multi-party computation in privacy-preserving machine

learning," KICS Fall Conference 2021, pp. 581-582, Yeosu, Korea, Nov. 2021.

채 승 재 (Seungjae Chae)

2017년 2월 : 성균관대학교 전자공학과 졸업
2017년 3월~현재 : 서울대학교 전기정보공학부 석박사 통합과정
<관심분야> 전자공학, 암호학

[ORCID:0000-0001-6743-0927]

이 준 우 (Joon-Woo Lee)

2016년 8월 : 서울대학교 전기정보공학부 졸업
2016년 9월~2022년8월 : 서울대학교 전기정보공학부 공학 박사
2022년9월~현재 : 중앙대학교 소프트웨어공학부 조교수
<관심분야> 전자공학, 암호학
[ORCID:0000-0002-4125-6331]

이 용 우 (Yong-Woo Lee)

2015년 2월 : 광주과학기술원 전기전산트랙 공학사
2015년 3월~2017년 9월 : 서울대학교 전기정보공학부 공학석사
2017년 10월~2021년2월 : 서울대학교 전기정보공학부 공학박사
2021년 3월~2023년 2월 : 삼성전자 SAIT Staff Researcher
2023년 3월~현재 : 인하대학교 정보통신공학과 조교수
<관심분야> 암호학
[ORCID:0000-0001-9424-6498]

노 종 선 (Jong-Seon No)

1981년 2월 : 서울대학교 전자공학과 공학사
1984년 2월 : 서울대학교 대학원 전자공학과 공학석사
1988년 5월 : University of Southern California 전기공학과 공학박사
1998년 2월~1990년 7월 : Hughes Network Systems, Senior MTS
1990년 9월~1999년 7월 : 건국대학교 전자공학과 부교수
1999년 8월~현재 : 서울대학교 전기컴퓨터공학부 교수
<관심분야> 시퀀스, 협력통신, 시공간부호, 네트워크코딩, LDPC부호, OFDM, 이동통신, 암호학
[ORCID:0000-0002-3946-0958]