East Asian Math. J.
Vol. 39 (2023), No. 3, pp. 291–298
http://dx.doi.org/10.7858/eamj.2023.021

ANALYSIS OF THE SECURITY OF GENERIC HOMOMORPHIC AUTHENTICATED ENCRYPTION

Jinsu Kim*

Abstract. Recently, a new type of encryption called Homomorphic Authenticated Encryption (HAE) has been proposed. This combines the functionality of homomorphic encryption with authentication. Several concrete HAE schemes have been developed and security results for homomorphic authenticated encryption, designed by combining a homomorphic message authentication scheme with a homomorphic secret-key encryption, have been partially reported.

In this paper, we analyze the security of a design method that combines homomorphic message authentication and homomorphic encryption, with a focus on the encryption after authentication (EAA) type. The results of our analysis show that while non-forgeability and indistinguishability are maintained, strong non-forgeability is not.

1. Introduction

Homomorphic encryption is a promising approach to enhance information security in cloud computing environments ([3], [6], [7]). It enables operations to be performed on plaintext while it remains encrypted, allowing for the processing of information without the need for decryption. However, the reliability of the results of these operations is not guaranteed. This is because the subject of the operation may use an arbitrary ciphertext other than the intended one, or a third-party may generate a random ciphertext and manipulate the operation results on the cloud server. To address this issue, homomorphic authenticated encryption, which provides both encryption and authentication, has been recently proposed in studies ([8], [5], [11]).

The first homomorphic authenticated encryption scheme was proposed by Joo et al. [8]. Their scheme was designed to provide both indistinguishability and strong non-forgeability against chosen plaintext attackers, using the approximate greatest common divisor problem. However, the size of the ciphertext was

Received December 09, 2022; Accepted January 26, 2023.
2010 *Mathematics Subject Classification.* 11A11.
Key words and phrases. Homomorphic authenticated encryption, HAE, Security, Non-forgeability, Indistinguishability.
This research was supported by Korea Naval Academy Institute for Ocean Research(2022).
*Corresponding Author.

(pISSN 1226-6973, eISSN 2287-2833)

increased to support the multiplication operation in their scheme. To address this issue, Struck et al. [11] proposed a more efficient public-key type homomorphic authenticated encryption scheme, which only supports the addition operation. This scheme provides indistinguishability and non-forgeability against a chosen ciphertext attacker. Another efficient symmetric-key type homomorphic authenticated encryption, also supporting only the addition operation, was later proposed [5]. However, its security level is relatively low compared to other homomorphic authenticated encryption schemes. All of these schemes prevent plaintext exposure by encrypting the data with a randomized label used for authentication, similar to the generic encryption-after-authentication (EAA) construction [2]. However, this design approach has yet to be analyzed.

In this work, we perform a security analysis of the EAA type homomorphic authenticated encryption. Our analysis shows that if a homomorphic authenticated encryption scheme is constructed using a non-forgeable homomorphic message authentication code, it also exhibits non-forgeability. However, this does not hold for the stronger property of strong forgery. On the other hand, we confirm that the indistinguishability property is satisfied when a homomorphic authenticated encryption is designed using a homomorphic secret-key encryption that itself has indistinguishability.

The remainder of the paper is structured as follows. In Section 2, we provide definitions and relevant terms for our security analysis. In Section 3, we present the results of our analysis of the encryption-after-authentication (EAA) type homomorphic authenticated encryption. Finally, in Section 4, we present our conclusions and discuss potential future work.

2. Definitions and Related Terms

In order to analyze the security of the combined homomorphic authentication encryption (EAA), we define terms related to the homomorphic authentication encryption and the homomorphic message authentication scheme. Some notations and definitions refer to references ([8], [9], [10], [1], [4]).

2.1. Homomorphic Message Authentication

The homomorphic message authentication scheme (HMAC) consists of the following four probabilistic polynomial time algorithms.

- $(ek, sk) \leftarrow KeyGen(1^{\lambda})$: an operation key ek and a secret key sk are outputted for the security parameter λ.
- $\mu \leftarrow Aut(\tau, m, sk)$: given a label $\tau \in L$ and a secret key sk, a tag $\mu \in T$ is outputted for the message m.
- $\mu \leftarrow Eval(ek, f, \mu_1, \cdots, \mu_l)$: given an evaluation key ek, an function $f : M^l \rightarrow M$ and l number of tags, $\mu_1, \cdots, \mu_l \in T$, this algorithm outputs a tag $\mu \in T$.

- $h \rightarrow Ver((f, \tau_1, \cdots, \tau_l), m, \mu, sk)$: given a secret key sk, a labeled program $(f, \tau_1, \cdots, \tau_l)$, a message $m \in M$ and a tag $\mu \in T$, this algorithm outputs the verification result as a bit $h \in \{0, 1\}$.

The security of the homomorphic message authentication scheme is determined whether or not an attacker can arbitrarily create a forgery without secret information. The attack model is divided into two types of attacks, a chosen plaintext attack and a chosen authentication attack, depending on the level of the attacker's ability. In the case of a chosen plaintext attack, an attacker can obtain authentication by querying an arbitrary plaintext, whereas in the case of a chosen authentication attack, a verification query for authentication is also possible in addition to an authentication query for plaintext [1]. In both cases, the label used to generate the result of the attacker's authentication query is not recycled and must be used only once.

Forgery, which is an attacker's goal for HMAC, is defined in two forms as follows.

- $1 = Ver((f, \tau_1, \cdots, \tau_l), m', \mu', sk)$, and $f(m_i)_{i \in I}$ is not constant.
- $1 = Ver((f, \tau_1, \cdots, \tau_l), m', \mu', sk)$, and $f(m_i)_{i \in I}$ is constant. But, $f(m_i)_{i \in I} \neq m'$

where I is the set of indices used for the label, $f(m_i)_{i \in I}$ is a function whose domain is $M^{l-|I|}$ and its range is M.

Another type of forgery, strong forgery, is defined in two forms as follows. By definition, strong forgery includes all forgery.

- $1 = Ver((f, \tau_1, \cdots, \tau_l), m', \mu', sk)$, and $f(m_i)_{i \in I}$ or $Eval(f, \mu_1, \cdots, \mu_l, ek)$ is not constant.
- $1 = Ver((f, \tau_1, \cdots, \tau_l), m', \mu', sk)$, and $f(m_i)_{i \in I}$ or $Eval(f, \mu_1, \cdots, \mu_l, ek)$ is constant. But, $f(m_i)_{i \in I} \neq m'$ or $Eval(f, \mu_1, \cdots, \mu_l, ek) \neq \mu'$

Based on the definition of forgery, the non-forgeability for the chosen message attacker is defined as the following game, $UF - CMA_A^{\lambda}$.

- Initialization : The challenger is issued an evaluation key ek and a secret key sk for security parameter λ, and ek is passed to the attacker.
- Queries : To the attacker's authentication query for τ, m, the challenger calculates $\mu = Aut(\tau, m, sk)$ and responds to the attacker.
- Finalization : The attacker submits a forgery attempt $((f, \tau_1, \cdots, \tau_l), m', \mu')$ to the challenger. The challenger checks whether it is a forgery, outputs 1 if it is a forgery, and 0 if it is not, and ends the game.

The advantage of attacker in the above game is defined as follows.

$$Adv_{HMAC,A}^{UF-CMA}(\lambda) := Pr[UF - CMA_A^{\lambda} = 1]$$

If the value is negligible for a probabilistic polynomial-time attacker, the homomorphic message authentication scheme is said to have non-forgeability against a chosen plaintext attacker.

Similarly, based on the definition of strong non-forgeability, the strong non-forgeability for the attacker is defined as the above game. (In the finalization phase, it is checked whether it is a strong forgery or not.) The advantage of attacker in the game is defined as follows.

$$Adv^{SUF-CMA}_{HMAC,A}(\lambda) := Pr[SUF - CMA^{\lambda}_{A} = 1]$$

The non-forgeability for the chosen tag attacker is defined as the following game, $UF - CTA^{\lambda}_{A}$.

- Initialization : The challenger is issued an evaluation key ek and a secret key sk for security parameter λ, and ek is passed to the attacker.
- Queries : To the attacker's authentication query for τ, m, the challenger calculates $\mu = Aut(\tau, m, sk)$ and responds to the attacker. On the other hand, for each verification query $((f, \tau_1, \cdots, \tau_l), m, \mu)$ of A, give $b \leftarrow Ver((f, \tau_1, \cdots, \tau_l), m', \mu', sk)$ to A as an answer for the query.
- Finalization : The attacker submits a forgery attempt $((f, \tau_1, \cdots, \tau_l), m', \mu')$ to the challenger. The challenger checks whether it is a forgery, outputs 1 if it is a forgery, and 0 if it is not, and ends the game.

The advantage of attacker in the above game is defined as follows.

$$Adv^{UF-CTA}_{HMAC,A}(\lambda) := Pr[UF - CTA^{\lambda}_{A} = 1]$$

If the value is negligible for a probabilistic polynomial-time attacker, the homomorphic message authentication scheme is said to have non-forgeability against a chosen plaintext attacker.

Similarly, based on the definition of strong non-forgeability, the strong non-forgeability for the attacker is defined as the above game. (In the finalization phase, it is checked whether it is a strong forgery or not.) The advantage of attacker in the game is defined as follows.

$$Adv^{SUF-CTA}_{HMAC,A}(\lambda) := Pr[SUF - CTA^{\lambda}_{A} = 1]$$

2.2. Homomorphic Authenticated Encryption

The homomorphic authenticated encryption scheme (HAE) consists of the following four probabilistic polynomial-time algorithms.

- $(ek, sk) \leftarrow KeyGen(1^{\lambda})$: an operation key ek and a secret key sk are outputted for the security parameter λ.
- $c \leftarrow Enc(\tau, m, sk)$: given a label $\tau \in L$ and a secret key sk, a cipher-text $c \in C$ is outputted for the message m.
- $c' \leftarrow Eval(f, c_1, \cdots, c_l, ek)$: given an evaluation key ek, an function $f : C^l \rightarrow C$ and l number of cipher-texts, $c_1, \cdots, c_l \in C$, this algorithm outputs a cipher-texts $c' \in C$.
- m or $\perp \rightarrow Dec((f, \tau_1, \cdots, \tau_l), \bar{c}, sk)$: given a secret key sk, a labeled program $(f, \tau_1, \cdots, \tau_l)$, a cipher-text $\bar{c} \in C$, this algorithm outputs a message m or $\perp$.

The attacker's goals for homomorphic authenticated encryption are to decrypt the cipher-text and forge it. Forgery, and its strong variant are defined in two forms as follows in a manner similar to the homomorphic message authentication scheme.

Forgery

- $Dec((f, \tau_1, \cdots, \tau_l), c', sk) \neq \perp$, and $f(m_i)_{i \in I}$ is not constant.
- $Dec((f, \tau_1, \cdots, \tau_l), c', sk) \neq \perp$, and $f(m_i)_{i \in I}$ is constant. But, $Dec((f, \tau_1, \cdots, \tau_l), c', sk) \neq f(m_i)_{i \in I}$

Strong forgery

- $Dec((f, \tau_1, \cdots, \tau_l), c', sk) \neq \perp$, and $f(m_i)_{i \in I}$ or $Eval(f, c_1, \cdots, c_l, ek)$ is not constant.
- $Dec((f, \tau_1, \cdots, \tau_l), c', sk) \neq \perp$, and $f(m_i)_{i \in I}$ and $Eval(f, c_1, \cdots, c_l, ek)$ is constant. $Dec((f, \tau_1, \cdots, \tau_l), c', sk) \neq f(m_i)_{i \in I}$ or $Eval(f, c_1, \cdots, c_l, ek) \neq c'$

where I is the set of indices used for the label, $f(m_i)_{i \in I}$ is a function whose domain is $M^{l-|I|}$ and its range is M.

Based on these definitions, the (strong) non-forgeability for the chosen plaintext or cipher-text attacker is defined similar to the homomorphic message authentication scheme. IND-CPA and IND-CCA security of HAE is defined in a manner similar to the traditional secret-key encryption scheme.

3. Security Analysis

One common way to design a homomorphic authenticated encryption is to combine a homomorphic encryption with a homomorphic message authentication scheme. Specifically, one method is to generate authentication for plaintext and then encrypt it. The homomorphic authenticated encryption designed in this way is called HAE_{EAA}, it is composed of the following key generation ($KeyGen$), encryption (Enc), operation ($Eval$), and decryption (Dec) algorithms.

HAE_{EAA} :

- $(ek, sk) \leftarrow KeyGen(1^\lambda)$: an operation key $ek = (HSE.ek, HMAC.ek)$ and a secret key $sk = (HSE.sk, HMAC.sk)$ are outputted via $HSE.Key-Gen(1^\lambda), HMAC.KeyGen(1^\lambda)$ for the security parameter λ.
- $c \leftarrow Enc(\tau, m, sk)$: given a label $\tau \in L$ and a secret key sk, a cipher-text $c = HSE.Enc((m, \mu), HSE.sk)$ is outputted for the message m after the authentication $\mu = HMAC.Aut(\tau, m, HMAC.sk)$.
- $c' \leftarrow Eval(f, c_1, \cdots, c_l, ek)$: given an evaluation key ek, an function $f : C^l \rightarrow C$ and l number of cipher-texts, $c_1, \cdots, c_l \in C$, this algorithm outputs a cipher-texts $c' \in C$.
- m or $\perp \rightarrow Dec((f, \tau_1, \cdots, \tau_l), \bar{c}, sk)$: given a secret key $sk = (HSE.sk, HMAC.sk)$, a labeled program $(f, \tau_1, \cdots, \tau_l)$, a cipher-text $\bar{c} \in C$, this algorithm decrypt (m, μ)¸$HSE.Dec(c, HSE.sk)$, then verify $h \leftarrow$

$HMAC.Ver((f, \tau_1, \cdots, \tau_l), m, \mu, HMAC.sk)$. If $h = 1$, then return m. Otherwise, return $\perp$.

The following theorem shows the homomorphic authenticated enryption, HAE_{EAA}, preserves IND-CPA security.

Theorem 3.1. *If the homomorphic secret-key encryption (HSE) is IND-CPA, then the homomorphic authenticated encryption* ($HAE_{EAA)}$ *is also IND-CPA.*

Proof. Let A be a PPT adversary for the game IND-CPA of HAE_{EAA}. We can construct another adversary A' for the game IND-CPA of HSE from A as follows.

- Initialization :For a given key $HSE.ek$, A' generate a pair of keys $(HMAC.ek, HMAC.sk) \leftarrow HMAC.KeyGen(1^\lambda)$. Then $ek = (HSE.ek, HMAC.ek)$ is given to A.
- Queries : For each encryption query τ, m of A, A' get $\mu \leftarrow HMAC.Aut(\tau, m, HMA.sk)$, and an answer c from the encryption oracle $HSE.Enc$. Return c to A' as an answer for the query.
- Challenge : For the challenge (τ^*, m_0^*, m_1^*) of A, A' get tags, $\mu_b^* \leftarrow HMAC.Aut(\tau^*, m_b^*, HMAC.sk)$ for each $b \in \{0, 1\}$, and the random challenge ciphertext c^* from the encryption oracle, $HSE.Enc$. A' gives c^* to A.
- Queries : Proceed in the same way as in the previous queries.
- Finalization : For the output bit $h \in \{0, 1\}$ of A, A' submits her result $h' = h$.

Therefore, from the above simulation, $Adv_{HSE,A'}^{IND-CPA}(\lambda)$ is non-negligible if $Adv_{HAE,A}^{IND-CPA}(\lambda)$ is non-negligible.

□

The following theorem shows the homomorphic authenticated enryption, HAE_{EAA}, preserves also non-forgeability security.

Theorem 3.2. *If* $HMAC$ *is* $UF-CMA$*, then* HAE_{EAA} *is* $UF-CPA$*. .*

Proof. Let A be a PPT adversary for the game UF-CPA of HAE_{EAA}. We can construct another adversary A' for the game UF-CMA of $HMAC$ from A as follows.

- Initialization :For a given key $HMAC.ek$, A' generate a pair of keys $(HSE.ek, HSE.sk) \leftarrow HSE.KeyGen(1^\lambda)$. Then $ek = (HSE.ek, HMAC.ek)$ is given to A.
- Queries : For each encryption query (τ, m) of A, A' get μ from the authentication oracle, and an answer $c = HSE.Enc((m, \mu), HSE.sk)$ from the encryption oracle $HSE.Enc$. A' gives c to A as an answer for the query.
- Finalization : For the forgery attempt $((f, \tau, \cdots, \tau_l), c')$ of A, A' output $((f, \tau, \cdots, \tau_l), m', \mu')$, where $(m', \mu') \leftarrow HSE.Dec(c', HSE.sk)$.

Therefore, from the above simulation, $Adv^{UF-CMA}_{HMAC,A'}(\lambda)$ is non-negligible if $Adv^{UF-CPA}_{HAE,A}(\lambda)$ is non-negligible.

□

Similarly, we can show its non-forgeability against chosen cipher-text attacker.

Corollary 3.1. *If $HMAC$ is $UF-CTA$, then HAE_{EAA} is $UF-CCA$.*

Proof. The only difference is the query phase in the previous game. Therefore, we only describe the query phase.

- Queries : For each encryption query (τ, m) of A, A' get μ from the authentication oracle, and an answer $c = HSE.Enc((m,\mu), HSE.sk)$ from the encryption oracle $HSE.Enc$. A' gives c to A as an answer for the query. For each decryption query $((f,\tau_1,\cdots,\tau_l),c)$, A' has (m,μ), and an verification result h from $HSE.Dec(c, HSE.sk)$ and $HMAC.Ver((f,\tau_1,\cdots,\tau_l),m,\mu)$. If $h=1$, A' gives m to A. Otherwise, A' gives $\perp$.

Therefore, from the above simulation, $Adv^{UF-CTA}_{HMAC,A'}(\lambda)$ is non-negligible if $Adv^{UF-CCA}_{HAE,A}(\lambda)$ is non-negligible.

□

Unlike in previous cases, the homomorphic authenticated encryption does not preserve the strong non-forgeability.

Theorem 3.3. *The HAE_{EAA} does not preserve the strong non-forgeability security of an $HMAC$.*

Proof. Due to its homomorphic property, we can easily produce another cipher-text c' that has the same decryption result of A. This means that c' is a strong forgery in HAE_{EAA} by definition of the strong forgery.

□

4. Conclusion

In this work, we analyzed the security level of homomorphic authenticated encryption synthesized by combining a homomorphic message authentication code and a homomorphic secret-key encryption. We focused on the encryption-after-authentication (EAA) type of generic design methods for homomorphic authenticated encryption. Through the definition of relevant terms and the application of simulation and reduction processes, we showed that the non-forgeability and indistinguishability properties of the homomorphic message authentication scheme are preserved in the homomorphic authenticated encryption. However, strong non-forgeability is not guaranteed due to the homomorphic property. As a future research direction, it will also be important to study the security of authentication-after-encryption designed homomorphic authenticated encryption schemes.

Acknowledgment

This work is supported by internal funds in Naval Academy, Republic of Korea.

References

[1] M. Bellare, O. Goldreich, A. Mityagin, *The power of verification queries in message authentication and authenticated encryption*, Crypt. ePrint Arch., Rep. 2004/309, (2004).
[2] M. Bellare and C. Namprempre, *Authenticated encryption: Relations among notions and analysis of the generic composition paradigm*, J. Crypt., (2008), 469-491.
[3] Z. Brakerski, V. Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) LWE*, Found. Comp. Sci. - FOCS 2011, (2011), 97-106.
[4] D. Catalano, D Fiore, *Practical homomorphic MACs for arithmetic circuits*, Adv. Crypt. - EUROCRYPT 2013, 7881, (2013), 336-352.
[5] J. Cheon, K. Han, S. Hong, H. Kim, J. Kim, Y. Song, *Toward a secure drone system: Flying with real-time homomorphic authenticated encryption*, IEEE acc., 6, (2018), 24325-24339.
[6] J. Coron, T. Lepoint, M. Tibouchi, *Scale-invariant fully homomorphic encryption over the integers*, Pub. Crypt. – PKC 2014, 8383, (2014), 311-328.
[7] C. Gentry, *Fully homomorphic encryption using ideal lattices*, Proc. of the 41st ann. ACM symp. on The. comp. - STOC 2009, (2009), 169-178.
[8] C. Joo, A. Yun, *Homomorphic authenticated encryption secure against chosen-ciphertext attack*, Int. Conf. The. and App. Crypt. and Inf. Sec., (2014), 173-192.
[9] C. Joo, A. Yun, *A strongly unforgeable homomorphic mac over integers*, J. Kor. Ins. Inf. Sec. and Crypt., (2014), 461-475.
[10] J. Kim, *Analysis of Homomorphic Authenticated Encryption*, Conv. sec. j., 21(1), (2021), 33-44.
[11] P. Struck, L. Schabhüser, D. Demirel, J. Buchmann, *Linearly homomorphic authenticated encryption with provable correctness and public verifiability*, Int. Conf. Cod. Crypt. and Inf. Sec., (2017), 142-160.

Jinsn Kim
(Associate Professor) Division of Science Republic of Korea Naval Academy PO box number 88-4-1, 1 Jungwon-ro, Jinhae-gu, Changwon-si, Gyungnam, 51704 Republic of Korea
Email address: nemokjs1@gmail.com