Reducing Client-Side Cost in Secure Inner Product Evaluations

Changhee Hahn
dept. Electrical and Information Engineering
Seoul National University of Science and Technology
South Korea

Dongyoung Koo
dept. Convergence Security
Hansung University
South Korea

Junbeom Hur
dept. Computer Science and Engineering
Korea University
South Korea

***Abstract*—In the era of ubiquitous data collection and analysis, preserving privacy while utilizing data, such as inner product evaluations, poses a significant challenge. One such method is inner product functional encryption (IPFE), which enables the computation of inner products on encrypted vectors without exposing the vectors themselves. However, the computational intensity of IPFE decryption poses challenges, particularly for resource-limited client devices and high-dimensional vectors. To facilitate partial decryption, prior IPFE schemes require the client to derive a partial decryption key from an assigned secret key, incurring computation costs linear to the vector dimension. In this paper, we provide a framework for the challenges of IPFE, enabling efficient decryption outsourcing, maintaining data privacy, and facilitating key generation outsourcing in IPFE schemes. In the proposed framework, outsourced decryption using a partial decryption key generates a masked plaintext that only the client can unmask. Furthermore, the process of deriving the partial decryption key can be fully outsourced, thereby ensuring that the client's computational burden remains constant.**

***Index Terms*—Security, inner-product, outsourced computation**

I. Introduction

In the ever-expanding landscape of modern computing, the analysis of vast and complex datasets is crucial in numerous applications and industries. Central to this analytical endeavor is the concept of inner product evaluations, a fundamental mathematical operation with profound implications for data analysis [1], machine learning [2], and beyond. Inner product evaluations, also known as dot products, provide a means to quantify the similarity or dissimilarity between vectors, offering valuable insights into the underlying structure and relationships within data.

However, alongside the immense utility of inner product evaluations, privacy concerns are present. Traditional approaches often involve the centralization of data in a single location, making it susceptible to unauthorized access, breaches, and misuse. Moreover, the aggregation and analysis of disparate datasets from multiple sources raise concerns about data ownership, consent, and the potential for unintended disclosures or re-identification of individuals.

In light of these privacy challenges, there is a growing imperative to develop techniques and frameworks for preserving the privacy of inner product evaluations. One such technique is inner product functional encryption (IPFE) [3]–[6], which enables users to compute the inner product of two encrypted vectors without revealing the vectors themselves. On the other hand, outsourced decryption of ciphertexts has become a prevalent practice in modern computing, where resource-limited clients delegate extensive computations to cloud providers that offer computing resources over the Internet [7], [8].

While prior studies devise a decryption-outsourceable IPFE scheme, e.g., [9], there arises a strong motivation to outsource not only the decryption process but also the computation of partial decryption keys. This stems from the potential need that clients may possess multiple vectors, leading to the need for multiple decryption keys to be evaluated with another vectors in ciphertexts. This, in turn, necessitates the generation of a substantial number of partial decryption keys. In light of these considerations, we present our design goal: On top of decryption-outsourceable function-hiding IPFE, can the partial decryption key generation process be outsourced, allowing clients to offload the majority of computation burdens?

This design goal aims to address the challenge of computational complexity associated with generating partial decryption keys for high-dimensional vectors in IPFE. By allowing clients to delegate the key generation process to external entities, this goal seeks to alleviate the client-side computational burden. Accomplishing this objective would significantly improve the practicality and scalability of IPFE, facilitating its broader adoption in real-world scenarios such as privacy-preserving large-scale data analysis and secure computation outsourcing, among others.

In this paper, we propose an efficient partial decryption key algorithm for performing decryption of IPFE ciphertexts. In contrast to prior approach [10], which requires $\mathcal{O}(n)$ exponentiation operations over pairing groups to derive a partial decryption key, our method eliminates this computational cost, where n represents the dimension of the plaintext vectors involved in IPFE.

II. Background

A. Pairing Vector Spaces

Suppose an additive cyclic group $(\mathbb{G} = \langle G \rangle, +)$ of prime order q. A vector space $\mathbb{G}^n$ of dimension n over $\mathbb{Z}_q$ is defined

as

$$\left\{\mathbf{X} = \mathbf{x} \cdot G = (X_1 = x_1 \cdot G, \ldots, X_n = x_n \cdot G) | \mathbf{x} \in \mathbb{Z}_q^n \right\}.$$

Given two vectors $\mathbf{x}, \mathbf{y}$ of the same dimension and $a \in \mathbb{Z}_q$, the following operations are defined.

$$\mathbf{x} \cdot G + \mathbf{y} \cdot G = (\mathbf{x} + \mathbf{y}) \cdot G, \quad a \cdot (\mathbf{x} \cdot G) = (a \cdot \mathbf{x}) \cdot G.$$

Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, G_1, G_2, q)$ be a description of pairing groups, where e is a bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_t$, G_1 and G_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Set $g_t = e(G_1, G_2)$. We define the following operation with regard to the pairing vector space as

$$\begin{aligned}(\mathbf{x} \cdot G_1) \times (\mathbf{y} \cdot G_2) &= \prod_i e(x_i \cdot G_1, y_i \cdot G_2) \\ &= \prod_i g_t^{x_i \cdot y_i} = g_t^{\mathbf{x} \cdot \mathbf{y}^\top} = g_t^{\langle \mathbf{x}, \mathbf{y} \rangle}.\end{aligned}$$

B. (Function-Hiding) Inner-Product Functional Encryption

An IPFE scheme is function-hiding if the functional decryption key, as well as the ciphertext, reveals no additional information about the underlying vectors beyond their inner product. A function-hiding IPFE scheme is a tuple of algorithms $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Encrypt}, \Pi.\mathsf{Decrypt})$ defined in a message space $\mathbb{Z}_q^n$ as follows.

- $\Pi.\mathsf{Setup}(1^\lambda)$. Given a security parameter 1^λ, the setup algorithm outputs the public parameter pp and the master secret key msk, where msk includes pp.
- $\Pi.\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$. Given msk and a vector $\mathbf{x} = (x_1, ..., x_n)$, the key generation algorithm outputs the functional decryption key $\mathsf{sk}_\mathbf{x} = (\mathsf{sk}_1, ..., \mathsf{sk}_n)$.
- $\Pi.\mathsf{Encrypt}(\mathsf{msk}, \mathbf{y})$. Given msk and a vector $\mathbf{y} = (y_1, ..., y_n)$, the encryption algorithm outputs $\mathsf{ct}_\mathbf{y} = (\mathsf{ct}_1, ..., \mathsf{ct}_n)$ as a ciphertext.
- $\Pi.\mathsf{Decrypt}(\mathsf{pp}, \mathsf{sk}_\mathbf{x}, \mathsf{ct}_\mathbf{y})$. Given pp, $\mathsf{sk}_\mathbf{x}$, and $\mathsf{ct}_\mathbf{y}$, the decryption algorithm outputs $\langle \mathbf{x}, \mathbf{y} \rangle = \sum x_i y_i \in \mathbb{Z}_q$ or a special symbol $\perp$ indicating a decryption failure.

C. System Model

The IPFE system aims to compute the inner product of two vectors within its framework. This system consists of a data provider (DP), a data consumer (DC), and a decryption proxy (DeP). The DP holds a master secret key msk and derives a ciphertext $\mathsf{ct}_\mathbf{y}$, which represents the encrypted form of the vector $\mathbf{y}$. This ciphertext is subsequently stored in the DeP. It is important to note that in a typical IPFE system without decryption outsourcing, the DeP may function as an on-line storage unit, holding the ciphertext and sending it to the DC when needed.

On the other hand, the DC possesses a vector $\mathbf{x}$. To compute the inner product between his vector $\mathbf{x}$ and the vector $\mathbf{y}$ held by the DeP, the DC requests DP to issue a functional decryption key $\mathsf{sk}_\mathbf{x}$. Once the functional decryption key is obtained, the DC may either request the DeP to derive the desired inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ by sending $\mathsf{sk}_\mathbf{x}$ to the DeP, or retrieve the ciphertext $\mathsf{ct}_\mathbf{y}$ from the DeP and then perform the decryption operation by himself. In the first case, $\langle \mathbf{x}, \mathbf{y} \rangle$ is trivially revealed in the DeP. It is noteworthy that the DP and DC may be the same entity. In this case, a single entity holding msk uploads his dataset $\{\mathbf{y}_1, \mathbf{y}_2, \cdots\}$ to DeP in encrypted form $\{\mathsf{ct}_{\mathbf{y}_1}, \mathsf{ct}_{\mathbf{y}_2}, \cdots\}$, later issuing a query for the inner product on $\mathbf{x}$ by sending $\mathsf{sk}_\mathbf{x}$.

III. Proposed Framework

We delve into the details of our proposed framework. In most pairing-based IPFE schemes, particularly function-hiding IPFE [3]–[6], denoted as Π, the functional decryption key $\mathsf{sk}_\mathbf{x} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and the ciphertext $\mathsf{ct}_\mathbf{y} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ are computed pairwise using pairings, where $\mathsf{sk}_\mathbf{x} \in \mathbb{G}_a^n$, $\mathsf{ct}_\mathbf{y} \in \mathbb{G}_b^n$, with $a \neq b$, and $a, b \in \{1, 2\}$.

In the case where $a = 2$ and $b = 1$, the pairwise computation takes the following form:

$$\begin{aligned} T_1 &= \mathsf{ct}_1 \times \mathsf{sk}_1 & & & T_n &= \mathsf{ct}_n \times \mathsf{sk}_n \\ &= g_t^{x_1 \cdot y_1 \cdot r_1 \cdot R}, & \cdots, & & &= g_t^{x_n \cdot y_n \cdot r_n \cdot R}, \end{aligned}$$

where $r_1, \cdots, r_n, R \in \mathbb{Z}_q$ represent random values derived during the pairings of $\mathsf{sk}_\mathbf{x}$ and $\mathsf{ct}_\mathbf{y}$. Significantly, the n random values $r_1, \cdots, r_n$ are nullified during the computation of $\prod_i T_i$. Consequently, the DC can obtain $g_t^{\sum x_i y_i \cdot R}$.

Additionally, g_t^R is computed separately. As a result, the DC can deduce $\sum x_i y_i \in S$, where S is a polynomial-sized subset of the message space. This is achieved by identifying $\delta \in S$ such that $\prod_i T_i \stackrel{?}{=} g_t^{R \cdot \delta}$.

To enhance the security and efficiency of the framework, we adopt the transformation approach [10]. Herein, the DC selects an additional secret key $\mathsf{z} \in \mathbb{Z}_q$ and derives the transformation key $\mathsf{tk}_\mathbf{x} = \mathsf{sk}_\mathbf{x}^\mathsf{z} = (\mathsf{sk}_1^\mathsf{z}, \ldots, \mathsf{sk}_n^\mathsf{z})$.

Subsequently, the DC transmits $\mathsf{tk}_\mathbf{x}$ to the DeP, which performs the following computation and transmits $\mathsf{ct}_\mathbf{y}^\mathsf{out}$ back to the DC as $\mathsf{ct}_\mathbf{y}^\mathsf{out} = \prod_i \left(\mathsf{ct}_{y_i} \times \mathsf{sk}_{x_i}^\mathsf{z}\right) = g_t^{(\sum x_i y_i) \cdot R \cdot \mathsf{z}}$. Finally, the DC cancels out z from $\mathsf{ct}_\mathbf{y}^\mathsf{out}$, enabling the retrieval of $\langle \mathbf{x}, \mathbf{y} \rangle$.

From a security standpoint, any attempt by the DeP to reveal $\mathbf{y}$ is reduced to breaking the underlying Π, while the DeP remains oblivious to $\mathbf{x}$ due to the function-hiding property of Π. Stated differently, from the perspective of the DeP, $\mathsf{tk}_\mathbf{x}$ is indistinguishable from $\mathsf{sk}_{\mathsf{z} \cdot \mathbf{x}}$. Additionally, the DeP cannot learn $\langle \mathbf{x}, \mathbf{y} \rangle$ as it lacks the knowledge of z.

Regarding efficiency, the $\mathcal{O}(n)$ pairing operations initially assigned to the DC are outsourced to the DeP. Furthermore, the DeP transmits $\mathsf{ct}_\mathbf{y}^\mathsf{out}$ as the result of the transformation, incurring a constant $\mathcal{O}(1)$ outbound bandwidth cost.

However, it is important to note that the DC bears the additional computation costs, specifically $\mathcal{O}(n)$ exponentiation operations over $\mathbb{G}_a$, when deriving the transformation key $\mathsf{tk}_\mathbf{x}$ from $\mathsf{sk}_\mathbf{x}$. This additional computational load can be undesirable in various practical scenarios, potentially offsetting the computational benefits obtained through outsourced decryption. For example, envision a situation where the DC is provided with a set of vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ and intends to evaluate pairwise inner products with an (encrypted) vector $\mathbf{y}$. While outsourcing decryption allows the DC to avoid m

pairing operations, it still necessitates performing m exponentiation operations to derive distinct transformation keys for each vector. Hence, the computational advantage achieved through outsourced decryption is counterbalanced by the overhead of generating multiple transformation keys, rendering it less practical.

To address this issue, we allow the DC to submit a modified vector $\mathbf{x}^\star = (\mathsf{z} \cdot G_b, \mathbf{x}) \in \mathbb{G}_b \times \mathbb{Z}_q^n$ instead of receiving from the DP the functional decryption key $\mathsf{sk}_\mathbf{x}$ and then deriving $\mathsf{sk}_\mathbf{x}^\mathsf{z}$. The DP then executes the $\Pi.\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ algorithm, where $\mathsf{z} \cdot G_b$ is used in part instead of G_b to derive $\mathsf{sk}_\mathbf{x}^\mathsf{z}$. It should be noted that implementing this modified approach may require slight adjustments in the implementation of the $\Pi.\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ algorithm. In terms of efficiency, this approach significantly reduces the computation cost from $\mathcal{O}(n)$ (i.e., deriving $\mathsf{sk}_\mathbf{x}^\mathsf{z}$ from $\mathsf{sk}_\mathbf{x}$) to $\mathcal{O}(1)$ on the DC side. Importantly, this approach does not introduce additional computation costs on the DP side. Indeed, the only modifications made on the DP side involve using $\mathsf{z} \cdot G_b$ instead of G_b when G_b is used.

Interestingly, the storage cost on the DC side can also be reduced. In other words, the DC only needs to store z locally, eliminating the necessity to store every $\mathsf{sk}_\mathbf{x}$. This is feasible because, with $\mathsf{tk}_\mathbf{x}$ outsourced to the DeP, the DC can simply request the DeP to perform partial decryption when needed. This property can be particularly advantageous when the DC possesses multiple transformation keys.

IV. Related Work

Functional encryption is an area of cryptography that allows fine-grained access control and selective disclosure of encrypted data [11]. In recent years, several variants of functional encryption have been proposed, each with its own set of properties and applications. One such variant is inner product functional encryption (IPFE), which specifically focuses on the evaluation of inner products on encrypted data [12]–[15].

However, decryption can be computationally intensive if the length of vectors is long. To delegate the decryption to a resource-powerful server, the functional decryption should be handed. In this regard, researchers endeavor to design a function-hiding IPFE [3], [4], [6], which enables the evaluation of the inner products without revealing the specific function being computed. To this end, a functional decryption key is sent to the server which stores ciphertexts. The server then derives the inner product results between the functional decryption key and ciphertexts. It is noteworthy that such a decryption delegation model inherently reveals the inner product results, which may be sensitive.

V. Conclusion

This paper addressed the challenges associated with inner product functional encryption (IPFE). The computational overhead for decrypting IPFE ciphertexts poses difficulties for resource-constrained client devices and high-dimensional vectors. We proposed an efficient and generic framework for both decryption and key generation outsourcing in IPFE.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00244079).

References

[1] J. Balasch, S. Faust, and B. Gierlichs, "Inner product masking revisited," in *Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34*. Springer, 2015, pp. 486–510.

[2] W. Chen, Z. Miao, and Q. Qiu, "Inner product-based neural network similarity," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[3] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part I*. Springer, 2016, pp. 470–491.

[4] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," in *Public-Key Cryptography–PKC 2016: 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*. Springer, 2016, pp. 164–195.

[5] J. Tomida, M. Abe, and T. Okamoto, "Efficient functional encryption for inner-product values with full-hiding security," in *Information Security: 19th International Conference, ISC 2016, Honolulu, HI, USA, September 3-6, 2016. Proceedings 19*. Springer, 2016, pp. 408–425.

[6] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11*. Springer, 2018, pp. 544–562.

[7] I. Menache, O. Shamir, and N. Jain, "On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud," in *11th International Conference on Autonomic Computing ({ICAC} 14)*, 2014, pp. 177–187.

[8] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 71–84, 2015.

[9] H. Yang, Y. Su, J. Qin, and H. Wang, "Privacy-preserving outsourced inner product computation on encrypted database," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1320–1337, 2020.

[10] M. Green, S. Hohenberger, B. Waters *et al.*, "Outsourcing the decryption of abe ciphertexts." in *USENIX security symposium*, vol. 2011, no. 3, 2011.

[11] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*. Springer, 2011, pp. 253–273.

[12] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan, "Functional encryption for inner product predicates from learning with errors," in *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*. Springer, 2011, pp. 21–40.

[13] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," *Cryptology ePrint Archive*, 2015.

[14] S. C. Ramanna, "More efficient constructions for inner-product encryption," in *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings 14*. Springer, 2016, pp. 231–248.

[15] S. Agrawal, B. Libert, and D. Stehlé, "Fully secure functional encryption for inner products, from standard assumptions," in *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*. Springer, 2016, pp. 333–362.