SCIENCE CHINA
Information Sciences

• HIGHLIGHT •

April 2018, Vol. 61 048102:1–048102:3
https://doi.org/10.1007/s11432-017-9224-9

Simulation-based security of function-hiding inner product encryption

Qingsong ZHAO[1,2,4], Qingkai ZENG[1,2*], Ximeng LIU[3] & Huanliang XU[4]

[1]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023*, China;*
[2]*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023*, China;*
[3]*School of Information Systems, Singapore Management University, Singapore* 178902*, Singapore;*
[4]*College of Information Science and Technology, Nanjing Agricultural University, Nanjing* 210095*, China*

Received 19 April 2017/Revised 20 Jnue 2017/Accepted 16 August 2017/Published online 19 January 2018

Citation Zhao Q S, Zeng Q K, Liu X M, et al. Simulation-based security of function-hiding inner product encryption. Sci China Inf Sci, 2018, 61(4): 048102, https://doi.org/10.1007/s11432-017-9224-9

Functional encryption (FE) [1,2] is a modern type of encryption scheme that extends several previous notions and allows tremendous flexibility in controlling and computing on encrypted data. FE enables an authority to derive constrained decryption keys that are used by a user to obtain specific functions of encrypted messages. Informally, the authority generates a secret key sk_f for a function f from a master secret key. Then, the user can only learn $f(x)$ from a ciphertext $\mathrm{Enc}(x)$ with sk_f and reveal nothing else about x.

Inner product encryption (IPE) is an important example for functional encryption in which secret keys and ciphertexts are all related to vectors: given the ciphertext of a vector $\boldsymbol{x}$, the secret key for a vector $\boldsymbol{y}$ allows computing $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ [3–8]. An inner product encryption scheme is function-hiding if, in the secret keys and ciphertexts, there is no knowledge leakage about $\boldsymbol{x}$ and $\boldsymbol{y}$ other than the value $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. The existing function-hiding inner product encryption schemes satisfied the indistinguishability-based security notion, or could be proved secure only under the simulation-based security notion, which is much stronger than the indistinguishability-based security notion, in the generic group model.

Our contribution. In this article we construct a simulation-based inner product encryption (SIPE) scheme in the standard model for the first time. The scheme uses asymmetric bilinear pairing groups of prime order under the symmetric external Diffie-Hellman (SXDH) assumption. Our scheme is function-hiding in the private-key setting that handles an unbounded number of ciphertext queries and adaptive key queries. To obtain correctness of our scheme, it requires that inner products are in a polynomial-size range, which is consistent with [3–5]. As mentioned in [3–5], this is a reasonable requirement for statistical applications because the computation, for instance the average over a polynomial-size database, will naturally be contained within a polynomial range. Our SIPE scheme with a polynomial-size range can tolerate an unbounded number of ciphertext queries. This means that the adversary is restricted to learn something implied by a polynomial-size range. We compare our scheme with related work in Table 1.

Preliminaries. We now introduce the definitions that are used in the SIPE scheme.

Definition 1 (Inner product encryption)**.** An (private-key) IPE scheme is composed of the four PPT algorithms defined below.

Setup$(1^\lambda, n) \rightarrow$ (MSK, PP). The setup algorithm uses the security parameter λ and the vector length n to output a master secret key MSK and public parameters PP.

Encrypt(MSK, PP, $\boldsymbol{x}$) $\rightarrow$ CT. The encryption

* Corresponding author (email: zqk@nju.edu.cn)

Table 1 Comparison with related work. Definitions of security belong to the class xx-yy-zzz where xx ∈ {One, Many} denotes one or multiple challenge ciphertexts; yy ∈ {SEL, AD} denotes ciphertext queries are selectively or adaptively chosen; zzz ∈ {IND, SIM} denotes indistinguishability or simulation-based security

	Setting	Security	Assumption	Model
ABDP15 [3]	Public-key	Many-SEL-IND	DDH	Standard
ALS16 [6]	Public-key	Many-AD-IND	DDH, LWE, Paillier	Standard
BJK15 [4]	Private-key	Many-AD-IND	SXDH	Standard
DDM16 [5]	Private-key	Many-AD-IND	SXDH	Standard
TAO16 [7]	Private-key	Many-AD-IND	DLIN	Standard
SKAHAD16 [8]	Private-key	Many-AD-SIM		Generic group
This work	Private-key	Many-AD-SIM	SXDH	Standard

algorithm uses the master secret key MSK, the public parameters PP, and a vector $\boldsymbol{x} \in \mathbb{Z}_q^n \setminus \{\boldsymbol{0}\}$ to output a ciphertext CT.

KeyGen(MSK, PP, $\boldsymbol{y}$) $\to$ SK. The key generation algorithm uses the master secret key MSK, the public parameters PP, and a vector $\boldsymbol{y} \in \mathbb{Z}_q^n \setminus \{\boldsymbol{0}\}$ to output a secret key SK.

Decrypt(PP, CT, SK) $\to m \in \mathbb{Z}_q$ or $\perp$. The decryption algorithm uses the public parameters PP, the ciphertext CT encrypting some vector $\boldsymbol{x}$, and a secret key SK corresponding to some vector $\boldsymbol{y}$ to output either a value $m \in \mathbb{Z}_q$ or the dedicated symbol $\perp$.

Correctness. An IPE scheme defined over a message space $\mathbb{Z}_q^n$ is correct if for all non-zero vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n \setminus \{\boldsymbol{0}\}$, we have

$$\Pr\left[\begin{array}{l}(\mathrm{MSK},\mathrm{PP}) \xleftarrow{\$} \mathrm{Setup}(1^\lambda, n);\\ \mathrm{CT} \xleftarrow{\$} \mathrm{Encrypt}(\mathrm{MSK},\mathrm{PP},\boldsymbol{x});\\ \mathrm{SK} \xleftarrow{\$} \mathrm{KeyGen}(\mathrm{MSK},\mathrm{PP},\boldsymbol{y});\\ \mathrm{Decrypt}(\mathrm{PP},\mathrm{CT},\mathrm{SK}) = \langle \boldsymbol{x},\boldsymbol{y}\rangle\end{array}\right] = 1 - \mathrm{negl}(\lambda).$$

Definition 2 (Asymmetric bilinear pairing groups)**.** $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is a pairing group with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ of order q, where q is a prime. g_1 is the generator of $\mathbb{G}_1$, and g_2 is the generator of $\mathbb{G}_2$. The tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is used to define an asymmetric bilinear group with the following properties.

(1) (Bilinearity) $e(g_1^s, g_2^t) = e(g_1, g_2)^{st}$ for all $s, t \in \mathbb{Z}_q$.

(2) (Non-degeneracy) $e(g_1, g_2)$ has order q in $\mathbb{G}_T$.

We use a description $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ to denote the output of the algorithm $\mathcal{G}_{\mathrm{abpg}}(1^\lambda)$.

Definition 3 (SXDH)**.** $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a tuple produced by $\mathcal{G}_{\mathrm{abpg}}(1^\lambda)$. Consider the following problem: given the distributions $\mathcal{G}_\sigma^{\mathrm{SXDH}}(1^\lambda) = ((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T,\ e), g_1^\mu, g_1^\nu, Y_\sigma)$ for $\sigma \in \{0,1\}$ such that $\mu, \nu \xleftarrow{\$} \mathbb{Z}_q, Y_0 = g_1^{\mu\nu}$, and $Y_1 = g_1^{\mu\nu+r}$, where $r \xleftarrow{\$} \mathbb{Z}_q \setminus 0$, output $\mathcal{G}_0^{\mathrm{SXDH}}$ if σ is 0 and output $\mathcal{G}_1^{\mathrm{SXDH}}$ otherwise. We refer to the problem as the SXDH problem. For a PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ is defined as:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{SXDH}}(\lambda) = |\Pr[\mathcal{A}(1^\lambda, \phi) \to 1 | \phi \xleftarrow{\$} \mathcal{G}_0^{\mathrm{SXDH}}(1^\lambda)] - \Pr[\mathcal{A}(1^\lambda, \phi) \to 1 | \phi \xleftarrow{\$} \mathcal{G}_1^{\mathrm{SXDH}}(1^\lambda)]|.$$

If for all PPT algorithms $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{SXDH}}(\lambda)$ is negligible in λ, we say $\mathcal{G}_\sigma^{\mathrm{SXDH}}(1^\lambda)$ satisfies the SXDH assumption. If $\mathbb{G}_1, \mathbb{G}_2$ reverse the roles, that applies also to the analogous distributions $\mathcal{G}_\sigma^{\mathrm{SXDH}}(1^\lambda) = ((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_2^{\mu'}, g_2^{\nu'}, Y'_\sigma)$ for $\sigma \in \{0,1\}$ such that $\mu', \nu' \xleftarrow{\$} \mathbb{Z}_q, Y'_0 = g_2^{\mu'\nu'}$, and $Y'_1 = g_2^{\mu'\nu'+r'}$, where $r' \xleftarrow{\$} \mathbb{Z}_q \setminus 0$.

Definition 4 (Dual pairing vector spaces (DPVS))**.** $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a tuple produced by $\mathcal{G}_{\mathrm{abpg}}(1^\lambda)$. $\mathbb{V} = \mathbb{G}_1^n$ and $\mathbb{V}^* = \mathbb{G}_2^n$ over $\mathbb{Z}_q^n$ are n-dimensional vector spaces. $\mathbb{A} = \{g_1^{\boldsymbol{e}_i}\}_{i=1,\ldots,n}$ of $\mathbb{V}$ and $\mathbb{A}^* = \{g_2^{\boldsymbol{e}_i^*}\}_{i=1,\ldots,n}$ of $\mathbb{V}^*$ are canonical bases, where $\boldsymbol{e}_i = (0^{i-1}, 1, 0^{n-i})$ and $\boldsymbol{e}_i^* = (0^{i-1}, 1, 0^{n-i})$. $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E)$ is used to define dual pairing vector space, where a pairing $E : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, i.e., $E(g_1^{\boldsymbol{x}}, g_2^{\boldsymbol{y}}) = \prod_{i=1}^n e(g_1^{\boldsymbol{x}_i}, g_2^{\boldsymbol{y}_i}) = e(g_1, g_2)^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} \in \mathbb{G}_T$, and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$ with the following properties.

(1) (Bilinearity) $E(sg_1^{\boldsymbol{x}}, tg_2^{\boldsymbol{y}}) = E(g_1^{s\boldsymbol{x}}, g_2^{t\boldsymbol{y}}) = E(g_1^{\boldsymbol{x}}, g_2^{\boldsymbol{y}})^{st}$ for $s, t \in \mathbb{Z}_q$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_q^n$.

(2) (Non-degeneracy) $E(g_1^{\boldsymbol{x}}, g_2^{\boldsymbol{y}})$ has order q in $\mathbb{G}_T$ for all $\boldsymbol{y} \in \mathbb{Z}_q^n$, then $\boldsymbol{x} = \boldsymbol{0}$.

We use a description $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E)$ to denote the output of the algorithm $\mathcal{G}_{\mathrm{dpvs}}(1^\lambda, n, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$, where $n \in \mathbb{N}$.

We then present random dual orthonormal basis generator $\mathcal{G}_{\mathrm{ob}}(1^\lambda, \mathbb{Z}_q^n)$.

$\mathcal{G}_{\mathrm{ob}}(1^\lambda, \mathbb{Z}_q^n)$: $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E) \xleftarrow{\$} \mathcal{G}_{\mathrm{dpvs}}(1^\lambda, n, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$,

$\boldsymbol{V} = (v_{i,j}) \xleftarrow{\$} \mathrm{GL}(n, \mathbb{Z}_q)$, $(\omega_{i,j}) = (\boldsymbol{V}^{\mathrm{T}})^{-1}$,

$\boldsymbol{b}_i = \sum_{j=1}^n v_{i,j} g_1^{\boldsymbol{e}_j}$, $\mathbb{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$,

$\boldsymbol{b}_i^* = \sum_{j=1}^n \omega_{i,j} g_2^{\boldsymbol{e}_j^*}$, $\mathbb{B}^* = \{\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*\}$,

return $(\mathbb{B}, \mathbb{B}^*)$.

SIPE scheme.

Setup$(1^\lambda, n) \to$ (MSK, PP). The setup algorithm takes as input λ and a integer n, which is the length of vectors. Firstly the algorithm runs $\mathcal{G}_{\mathrm{abpg}}(1^\lambda)$ to generate an asymmetric bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. It then samples dual pairing vector spaces $(q, \mathbb{V}_1, \mathbb{V}_1^*, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_1^*, E_1) \xleftarrow{\$} \mathcal{G}_{\mathrm{dpvs}}(1^\lambda, 2n+2, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$ and $(q, \mathbb{V}_2, \mathbb{V}_2^*, \mathbb{G}_T, \mathbb{A}_2, \mathbb{A}_2^*, E_2) \xleftarrow{\$} \mathcal{G}_{\mathrm{dpvs}}(1^\lambda, 4, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$. Next, it samples dual orthonormal bases $(\mathbb{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2n+2}\}, \mathbb{B}^* = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2n+2}^*\}) \xleftarrow{\$} \mathcal{G}_{\mathrm{ob}}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $(\mathbb{D} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_4\}, \mathbb{D}^* = \{\boldsymbol{d}_1^*, \ldots, \boldsymbol{d}_4^*\}) \xleftarrow{\$} \mathcal{G}_{\mathrm{ob}}(1^\lambda, \mathbb{Z}_q^4)$. It defines $\widehat{\mathbb{B}} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2n}, \boldsymbol{b}_{2n+2}\}$, $\widehat{\mathbb{B}}^* = \{\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*, \boldsymbol{b}_{2n+1}^*\}$, $\widehat{\mathbb{D}} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_4\}$, $\widehat{\mathbb{D}}^* = \{\boldsymbol{d}_1^*, \boldsymbol{d}_3^*\}$. The master secret key is MSK$= (\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*)$, and the public parameters are PP$= (q, \mathbb{V}_1, \mathbb{V}_1^*, \mathbb{V}_2, \mathbb{V}_2^*, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_1^*, \mathbb{A}_2, \mathbb{A}_2^*, E_1, E_2)$.

Encrypt(MSK, PP, $\boldsymbol{x}$) $\to$ CT. The encryption algorithm takes as input MSK, PP, and a vector $\boldsymbol{x} \in \mathbb{Z}_q^n \setminus \{\boldsymbol{0}\}$. It chooses four uniformly random elements $\alpha, \alpha', \varphi, \varphi' \xleftarrow{\$} \mathbb{Z}_q$ and outputs CT $= (c_1 = g_1^{\alpha \sum_{i=1}^n x_i \boldsymbol{b}_i + \alpha' \sum_{i=1}^n x_i \boldsymbol{b}_{n+i} + \varphi \boldsymbol{b}_{2n+2}}, c_2 = g_1^{\alpha \boldsymbol{d}_1 + \alpha' \boldsymbol{d}_2 + \varphi' \boldsymbol{d}_4})$.

KeyGen(MSK, PP, $\boldsymbol{y}$) $\to$ SK. The secret key generation algorithm takes as input MSK, PP, and a vector $\boldsymbol{y} \in \mathbb{Z}_q^n \setminus \{\boldsymbol{0}\}$. It chooses three uniformly random elements $\beta, \eta, \eta' \xleftarrow{\$} \mathbb{Z}_q$ and outputs SK $= (k_1^* = g_2^{\beta \sum_{i=1}^n y_i \boldsymbol{b}_i^* + \eta \boldsymbol{b}_{2n+1}^*}, k_2^* = g_2^{\beta \boldsymbol{d}_1^* + \eta' \boldsymbol{d}_3^*})$.

Decrypt(PP, CT, SK) $\to m \in \mathbb{Z}_q$ or $\perp$. The decryption algorithm takes in PP, the ciphertext (c_1, c_2), and the secret key (k_1^*, k_2^*). It outputs $D_1 = E(c_1, k_1^*), D_2 = E(c_2, k_2^*)$.

It remains to be seen whether there exits $m \in \mathbb{Z}_q$ such that $(D_2)^m = D_1$ as elements of $\mathbb{G}_T$. If the equation holds, the algorithm outputs m. Otherwise, it outputs $\perp$. In order to ensure that the decryption algorithm takes polynomial time, the size of the plaintext space is subject to a static polynomial-size range.

Theorem 1. Under the SXDH assumption our SIPE scheme is one-AD-SIM-secure.

The proof of Theorem 1 is given in Appendix C.

Remark 1. We also can prove the SIPE sheme is many-AD-SIM-secure if the adversary gets hold of an unbounded number of ciphertexts and secret keys. The solution is to sample different dual orthonormal bases for each message.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61772266, 61572248, 61431008), and Humanities and Social Science Fund of Central University Basic Scientific Research Business Expenses in Nanjing Agricultural University (Grant No. 6J0123).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 O'Neill A. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. Http://eprint.iacr.org/2010/556
2 Boneh D, Sahai A, Waters B. Functional encryption: definitions and challenges. In: Proceedings of the 8th Conference on Theory of Cryptography, Providence, 2011. 253–273
3 Abdalla M, Bourse F, de Caro A, et al. Simple functional encryption schemes for inner products. In: Proceedings of the 18th International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, 2015. 733–751
4 Bishop A, Jain A, Kowalczyk L. Function-hiding inner product encryption. In: Proceedings of the 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, 2015. 470–491
5 Datta P, Dutta R, Mukhopadhyay S. Functional encryption for inner product with full function privacy. In: Proceedings of the 19th International Conference on the Theory and Practice of Public-Key Cryptography, Taipei, 2016. 164–195
6 Agrawal S, Libert B, Stehlé D. Fully secure functional encryption for inner products, from standard assumptions. In: Proceedings of the 36th Annual International Cryptology Conference, Santa Barbara, 2016. 333–362
7 Tomida J, Abe M, Okamoto T. Efficient functional encryption for inner-product values with full-hiding security. In: Proceedings of the 19th Information Security Conference, Honolulu, 2016. 408–425
8 Kim S, Lewi K, Mandal A, et al. Function-hiding inner product encryption is practical. Cryptology ePrint Archive, Report 2016/440, 2016. Http://eprint.iacr.org/2016/440