Electronic Preprint for Journal of Information Processing Vol.24 No.5

Regular Paper

Software Library for Ciphertext/Key-Policy Functional Encryption with Simple Usability

Keisuke Hasegawa[1,a)] Naoki Kanayama[1] Takashi Nishide[1] Eiji Okamoto[1]

Received: December 3, 2015, Accepted: June 2, 2016

Abstract: In traditional public key encryption schemes, data encrypted by a public key pk can be decrypted only by a secret key sk corresponding to pk, and the relation between pk and sk is static. Therefore, the schemes are unsuitable for control of access to a single data by several users. Meanwhile, functional encryption (FE) is an encryption scheme that provides more sophisticated and flexible relations between pk and sk. Thus, FE enables only one pk to encrypt the data with any conditions for decryption, so it is considered a very useful tool for the access control of data on the cloud server. However, implementing the current FE scheme is a non-trivial task because the deep knowledge of the scheme is required. This is an obstacle factor to deploy the FE scheme in the real-world security systems. In this paper, we propose an implementation of the FE (Ciphertext-Policy FE and Key-Policy FE, which are useful classes of FE) library usable even for people who do not have the deep knowledge of these schemes.

Keywords: functional encryption, attribute-based encryption, implementation

1. Introduction

In recent years, cloud computing has become common for enterprise systems. For example, suppose that a company uses a cloud server and does access control on the server. When we want only a specific person (e.g., a person who belongs to the personnel department or an administrative manager) to access data, we usually have to let the server control access so that only the person can access the data. However, if we use a cloud server and store data on it without considering security of the data, it may be insecure. If the server controls access permissions of the data on it, an attacker may get all the data on the server by breaking into the server. In traditional public key encryption schemes, the relation between public key pk and secret key sk is static, so the schemes cannot encrypt data properly in the aforementioned scenario. Therefore, if more than one user with different access permissions respectively access the cloud server, we need to use several public keys to encrypt with different policies for decryption. Therefore, we may need to have a lot of public keys, and this situation is sometimes inconvenient.

Meanwhile, functional encryption (FE) [15], [16] is an encryption scheme that provides more sophisticated and flexible relations between pk and sk. Using an FE scheme, we can encrypt the data with an access policy like ("the personnel department" OR ("the general affairs department" AND "manager")), and as a result, even if an attacker succeeds in attacking the server, he or she cannot get any information of the encrypted data on the server. An FE scheme enables only one pk to encrypt the data with any policy for decryption. Therefore, we can easily manage access permissions on the server by using FE. In addition, because the server does not have control on access permissions, it is possible to store data on the server securely, even if the server cannot be trusted. Hence, an FE scheme is considered a very useful tool for access control of data, but the implementation of an FE scheme is a non-trivial task because the special knowledge of the scheme is required. This is considered an obstacle factor to deploy the FE scheme in the real-world security systems.

1.1 Our Result

This paper[*1] proposes an implementation of the CP-FE and KP-FE (useful classes of FE scheme based on Refs. [15], [16]) library usable even for people who do not have the deep knowledge of the FE scheme based on pairing-based cryptography[*2]. Using our system, we can easily manage data securely on the cloud server even if the server is untrusted. Furthermore, a user can encrypt data by specifying an access policy (a logical formula expressing a condition for decryption, which consists of attributes and logic gates), and only the user with proper attributes can decrypt the data.

1.2 Related Works

Sahai and Waters [17] introduced attribute-based encryption (ABE) for access control on encrypted data, and some functional encryption schemes have been introduced after that. Thus, there is some trend toward the realization of ABE such as Refs. [3], [8], and also the standardization of pairing-based cryptography [10]. Okamoto and Takashima [15], [16] introduced functional encryption in 2010, which can support non-monotone access structures

[1] University of Tsukuba, Tsukuba, Ibaraki 305–8577, Japan
[a)] hasegawa.keisuke.wm@alumni.tsukuba.ac.jp

[*1] A preliminary conference version of this paper was presented at [9]. This paper is the extended version of Ref. [9].

[*2] The field of pairing-based cryptography is about cryptosystems using pairing, and has been growing rapidly since Refs. [5], [18] were proposed. The definition of paring is in Appendix A.1.

and is adaptively secure in the standard model. Therefore, by virtue of its rich properties, the Okamoto-Takashima functional encryption can be a building block in other cryptographic protocols. So, it can be used in much more cases in the real life compared with other schemes and the implementation and standardization of the Okamoto-Takashima scheme are expected. However, the existing functional encryption libraries Refs. [1], [2], [7], [12], [21] are not based on Refs. [15], [16], and the construction of Refs. [15], [16] is so complex an algorithm that there is no library implementing the scheme of Refs. [15], [16] until now as far as we know. Implementing FE schemes usually requires pairing operations on elliptic curves such as Refs. [7], [14], [20] and our implementation uses Ref. [20].

2. Functional Encryption

In this section, we explain functional encryption (FE). In traditional public key encryption schemes, the ciphertext encrypted by a public key pk is decrypted only by a single secret key sk corresponding to the pk, and the relation is unchanged. Meanwhile, FE provides more sophisticated and flexible relations between the keys where a secret key, sk_Ψ, is associated with a parameter, Ψ, and message m is encrypted to a ciphertext $ct_\Upsilon := \mathrm{Enc}(m, \mathrm{pk}, \Upsilon)$ using system public key pk along with another parameter Υ. Ciphertext ct_Υ can be decrypted by secret sk_Ψ if and only if a relation $R(\Psi, \Upsilon) = True$ holds. Then, Ψ is a parameter relevant to secret key (decryption key) sk.

In our library, two types of FE schemes based on Refs. [15], [16] are available [*3]. One is called Ciphertext-Policy Functional Encryption (CP-FE) scheme, and the other is Key-Policy Functional Encryption scheme. Although these two schemes seem to be similar, these schemes are supposed to be used in different situations respectively.

2.1 Ciphertext-Policy Functional Encryption (CP-FE)

In the CP-FE scheme which is implemented in this library, the parameter Υ expresses an access structure described by attributes and threshold gates, and the parameter Ψ expresses a set of attributes of the secret key holder. An attribute that expresses information of user is expressed by a category (e.g., "Gender", "Position") and value that is relevant to the category (e.g., "male", "manager") in the scheme. For example, "Gender = male" is used as one attribute, Υ is a predicate like ("Gender = male" AND "Position = manager"), and Ψ is a set like ("Gender = male", "Position = manager"). We note that AND gates can be constructed as n-out-of-n threshold gates and OR gates as 1-out-of-n threshold gates, and the predicate of t-out-of-n threshold is expressed as $(p_1, p_2, ..., p_n, t)$ (p represents an attribute, and $n \geq t$) in this paper. For example, ("Gender = male" AND "Position = manager") is expressed as ("Gender = male", "Position = manager", 2).

In the scheme, the relation $R(\Psi, \Upsilon) = True$ holds and the encrypted data can be decrypted if and only if the set of attributes Ψ satisfies the access structure Υ (**Fig. 1**).

Thus, only one public parameter pk can specify the access policy and receiver that can decrypt, so it is considered useful for more complex access control.

The CP-FE scheme is defined as follows.

Setup$(1^\lambda, \vec{n} := (d; n_1, ..., n_d))$ -

On input 1^λ(λ :security parameter), the max number of categories for attributes d, format $\vec{n} := (d; n_1, ..., n_d)$ of attributes (n_i:dimension of vector of value $\vec{x_i}$. For all i, $n_i = 2$ in the case of our library.), output public parameter pk and master secret key sk.

KeyGen$(pk, sk, \Psi := \{(t, \vec{x_t} := (x_{t,1}, ..., x_{t,n_t}) \in \mathbb{F}_q^{n_t} \setminus \{\vec{0}\}) \mid 1 \leq t \leq d\})$ -

On input pk, sk and a set of attributes $\Psi := \{(t, \vec{x_t} := (x_{t,1}, ..., x_{t,n_t}) \in \mathbb{F}_q^{n_t} \setminus \{\vec{0}\}) \mid 1 \leq t \leq d\}$, output decryption key sk_Ψ, which corresponds to Ψ.

Enc$(pk, \Upsilon := (M, \rho), m)$ -

On input pk, an access structure Υ which is used for decryption step, and plaintext m, output a ciphertext ct_Υ. The formal definition about an access structure is given in Appendix A.2, and how to convert an access policy (which is given as a logical formula by the user of our FE library) to an access structure is given in Appendix A.2.1.

Dec$(pk, sk_\Psi, ct_\Upsilon)$ -

On input pk, sk_Ψ and ct_Υ, output m if relation $R(\Psi, \Upsilon) = True$ holds.

2.2 Key-Policy Functional Encryption (KP-FE)

In KP-FE scheme, on the other hand, the parameter Υ expresses a set of attributes, and the parameter Ψ expresses an access structure for a secret key holder. Namely, Υ used in KP-FE scheme is equal to Ψ used in CP-FE scheme, and Ψ used in KP-FE scheme also is equal to Υ used in CP-FE scheme. For example, Υ is a set like ("Gender = male", "Position = manager"), and Ψ is a predicate like ("Gender = male" AND "Position = manager").

In the scheme, the encrypted data can be decrypted if and only if the set of attributes Υ satisfies the access structure Ψ.

The situation in which the KP-FE scheme is used is different from that of the CP-FE scheme. For example, KP-FE scheme is supposed to be used for a management of streaming service (**Fig. 2**).

The KP-FE scheme is defined as follows.

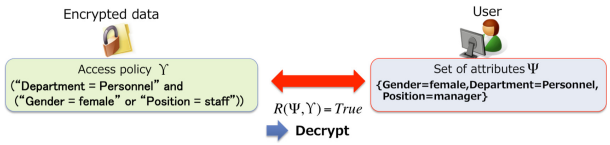

Fig. 1 Example of CP-FE.

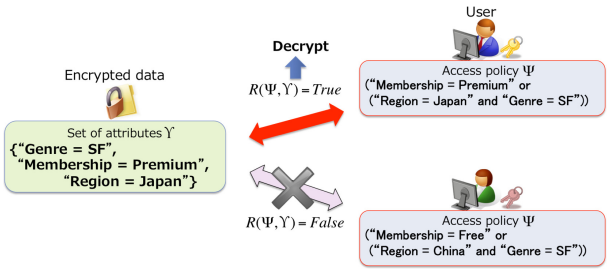

Fig. 2 Example of KP-FE.

[*3] These schemes are based on Appendix G.1 and G.2 of Refs. [15], [16].

Setup$(1^\lambda, \vec{n} := (d; n_1, ..., n_d))$ -

On input 1^λ(λ :security parameter), the max number of categories for attributes d, format $\vec{n} := (d; n_1, ..., n_d)$ of attributes (n_i:dimension of vector of value $\vec{x_i}$. For all i, $n_i = 2$ in the case of our library.), output public parameter pk and master secret key sk.

KeyGen$(pk, sk, \Psi := (M, \rho))$ -

On input pk, sk and an access structure $\Psi := (M, \rho)$, output decryption key sk_Ψ, which corresponds to Ψ.

Enc$(pk, \Upsilon := \{(t, \vec{x_t} := (x_{t,1}, ..., x_{t,n_t}) \in \mathbb{F}_q^{n_t} \backslash \{\vec{0}\}) \,|\, 1 \le t \le d\}, m)$ -

On input pk, a set of attribute Υ which is used for decryption step, and plaintext m, output a ciphertext ct_Υ.

Dec$(pk, sk_\Psi, ct_\Upsilon)$ -

On input pk, sk_Ψ and ct_Υ, output m if relation $R(\Psi, \Upsilon) = True$ holds.

3. Implementation

3.1 Application

In this section, we explain how to use our library. Our implementation uses the TEPLA library [20] for pairing operations, and is an asymmetric version of dual pairing vector spaces constructed using asymmetric bilinear pairing groups. We consider the application like the example below.

3.1.1 Preparation (Setup)

This library needs to distribute an attribute list describing a set of attributes, public parameter (pk) used in Encryption step, and decryption key (sk_Ψ) in which information of attributes of a user Ψ is embedded. An administrator that manages access to files on the system executes **Setup** and then generates pk and sk. In **Setup** step, the administrator inputs the max number of categories for attributes d as argument (**Fig. 3**). The administrator has only to execute Setup once, even when using both FE schemes. The generated pk and sk mainly include the information for encrypting a file and generating a decryption key respectively. The file containing pk currently needs to be placed in a directory where the operation **Enc** is executed, and the file including sk needs to be placed in a directory where the operation **KeyGen** is executed.

Before generating a decryption key (sk_Ψ, which is used to decrypt an encrypted file), the administrator needs to create a file describing an attribute list corresponding to Ψ. The attributes in the list are defined as tuples of category and value. The current supported file format is csv. In the example below (**Fig. 4**), AttributeList.csv defines three categories "Gender", "Department", and "Position". These categories have several values respectively. Once a list file is created, any category can be added later at any time unless the number of categories described in the list exceeds d. The value of a category can also be added to the list file later unless the number of values exceeds the limit determined by the public parameter. When the list file is updated, the administrator needs to re-distribute the file to users (but the administrator does not need to re-issue pk).

Fig. 3 setup command.

3.1.2 Key Generation

To generate sk_Ψ, the administrator inputs pk, sk, attribute list, and the information of attributes of a user Ψ. In the CP-FE scheme the parameter Ψ is a set of attributes, and in the KP-FE scheme, the parameter Ψ is an access policy (**Fig. 5**). The administrator executes **KeyGen** (**Fig. 6**), and the generated sk_Ψ is distributed to each user securely (**Fig. 7**). These figures are for the case of CP-FE scheme.

3.1.3 Encryption

When a user encrypts a file, the user inputs pk, attribute list, and the information of attributes. In the CP-FE scheme, the in-

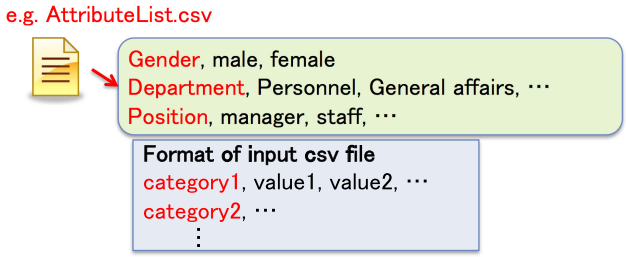

Fig. 4 attributelist.csv.

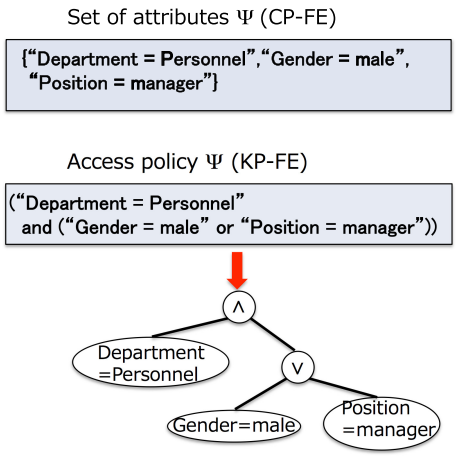

Fig. 5 Example of Ψ.

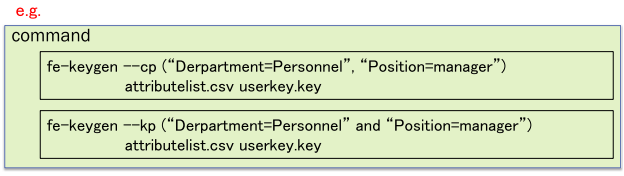

Fig. 6 keygen command.

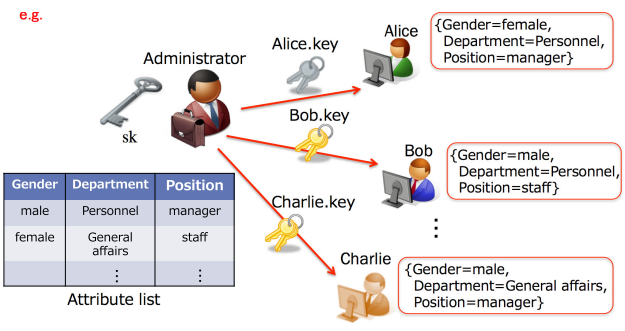

Fig. 7 keygen.

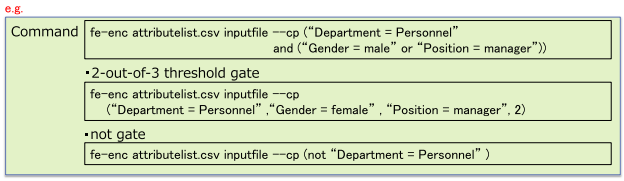

Fig. 8 enc command.

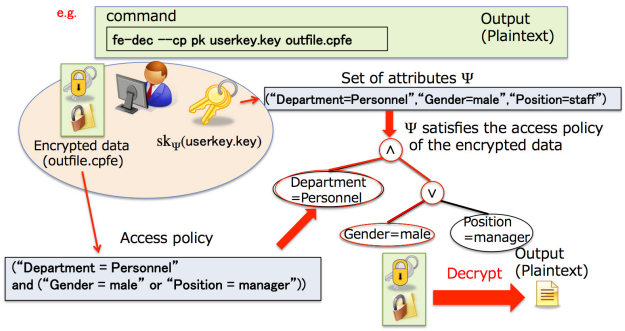

Fig. 9 decrypt.

formation of attributes for a ciphertext is an access policy. The access policy used in our library is a logical formula using categories and values described in the attribute list. In the access policy, we can use not only AND or OR gates but also t-out-of-n threshold gates and NOT gates to specify a more sophisticated and flexible access policy (**Fig. 8**). In the access policy used in the KP-FE scheme (the parameter Ψ of decryption key sk_Ψ), we can also use these gates. The user executes **Enc** and generates a ciphertext file that has the access policy specified by the user.

In the KP-FE scheme, the information of attributes for a ciphertext is a set of attributes (equal to the parameter Υ of ciphertext ct_Υ in CP-FE scheme).

The generated ciphertext file is divided into two parts. One is the encrypted data and the other is the data describing the access policy (CP-FE scheme) or the set of attributes (KP-FE scheme). For efficient encryption, the plaintext file is encrypted by AES [*4], and the random temporary key of AES is encrypted by the CP-FE scheme or the KP-FE scheme. Only if the access policy for the ciphertext or decryption key is satisfied, the random temporary key of AES is obtained and the ciphertext can be decrypted.

3.1.4 Decryption

When a user decrypts an encrypted file, the user inputs pk and sk_Ψ. In the CP-FE scheme, the user executes **Dec** and can decrypt the encrypted file only if sk_Ψ including the user's set of attributes Ψ satisfies the access policy for the encrypted file (**Fig. 9**). In the KP-FE scheme, in a similar way, the user executes **Dec** and can decrypt the encrypted file only if the set of attributes Ψ for the file satisfies the access policy for the decryption key sk_Ψ. Our library verifies whether the user can decrypt the encrypted file or not. If and only if Ψ satisfies Υ, the key of AES is obtained and the ciphertext file encrypted by AES can be decrypted.

[*4] Advanced Encryption Standard (AES) [6] is a symmetric key encryption scheme adopted by National Institute of Standards and Technology (NIST), and is used widely as a standard symmetric key encryption scheme.

Table 1 Environment of Performance Measurement.

CPU	Intel Core i7 CPU @2.9 GHz
RAM	8 GB 1600 MHz DDR3
OS	Mac OS X 10.9.5
Compiler	gcc version 4.8.2
Language	C
External library	GMP 6.1.0
	OpenSSL 1.0.1g
	TEPLA 1.0.0

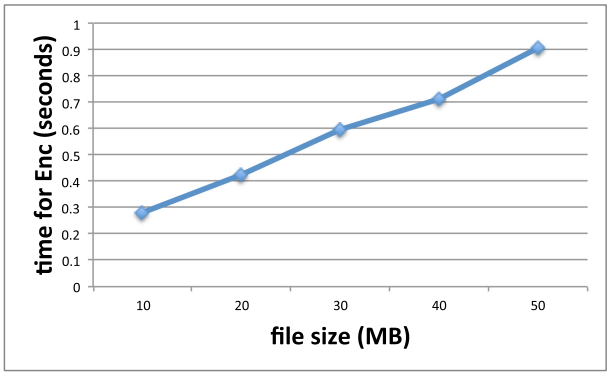

Fig. 10 time for Enc regarding the file size.

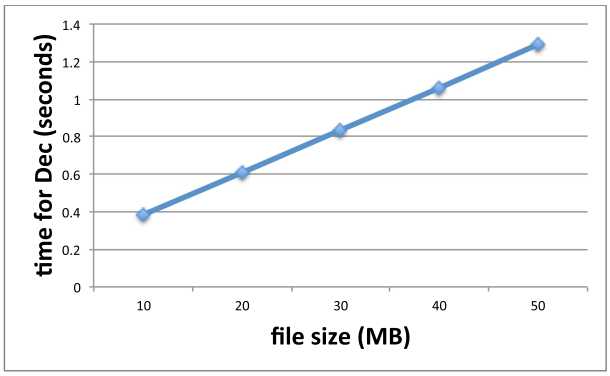

Fig. 11 time for Dec regarding the file size.

3.2 Performance Measurements

We now provide the result on the performance of our library. The measurements were taken on the environment shown in **Table 1**. Our implementation uses OpenSSL for AES and a hash function (SHA-1 [*5]), and the random number generation in our implementation uses the function of GMP (GNU Multiple Precision Arithmetic Library) [*6]. TEPLA is used for pairing operations, and the calculation on pairing is using Optimal Ate Pairing over Barreto-Naehrig (BN) curves at the 128-bit security level [4]. Although the performance measurements were taken on the Mac OS environment, our library supports also Windows OS and Linux OS. The library still has room for improvement of algorithms, so the measurements here are only a rough estimate. **Figures 10** and **11** display measurements of time for **Enc** and **Dec** regarding the file size respectively, and the access policy used in **Enc** consists of one attribute. The plaintext data used in this measurements is a text file, and the access policy consists of 1AND-gate. **Figures 12** and **13** display measurements of time for **Enc** and **Dec** regarding the number of attributes of an access policy respectively, and the file size of plaintext is 10 MB. As expected,

[*5] Secure Hash Algorithm 1 (SHA-1) is a hash function adopted by NIST.
[*6] https://gmplib.org/

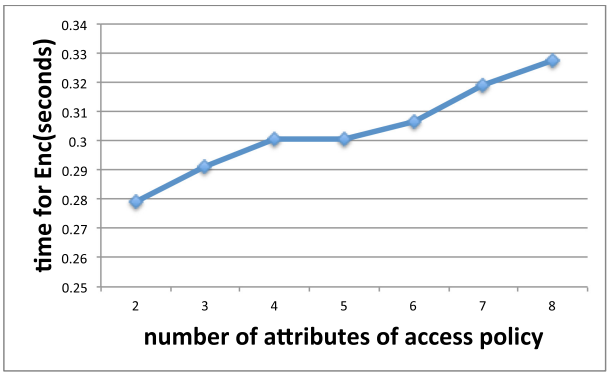

Fig. 12 time for Enc regarding the number of attributes of access policy.

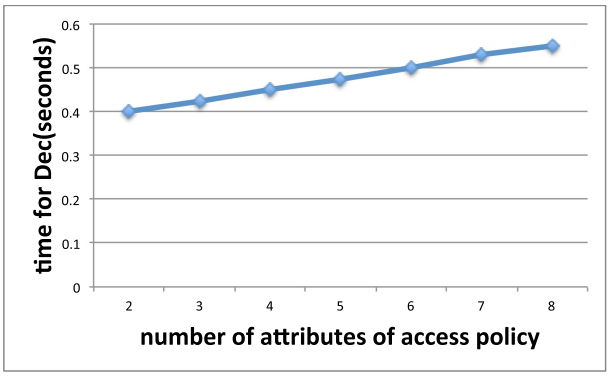

Fig. 13 time for Dec regarding the number of attributes of access policy.

the time for **Enc** and **Dec** is linear in the number of attributes associated with the access policy. Furthermore, as the result on the performance (Fig. 13) shows, the time for **Dec** is actually linear in the minimum number of attributes that are required to satisfy the access policy. For example, in comparison between the runtimes for **Dec** of two access policies ("A" AND "B" AND "C") and ("A" OR "B" OR "C"), the former access policy needs more time compared with the latter access policy because the number of pairing operations needed for decryption with the former access policy is larger than that of the latter access policy. However, the file itself is encrypted by AES, so the time for **Enc** or **Dec** will be mainly affected by AES as the file size becomes large.

4. Future Work

There are some functions or another FE scheme that we are going to implement to make the library more convenient to use.

4.1 Numerical Attributes and its Expression

When a category has numerical values like age, an access policy including the category may need not only "=" or "≠" but also "<" or ">". Though using only "=" or "≠" can also express "<" or ">", it can be inconvenient. For example, when a category "A" has "1" to "30" as numerical values, access policy ("A ≤ 15") needs to be expressed as ("A = 1" or "A = 2" or ... or "A = 15"), and as the range for the values becomes larger, it is more awkward to express the access policy only by using equality (without inequality).

To avoid this inconvenience, we allow users to use inequalities "<" and ">" in the descriptions of access policies as syntactic sugar. Here we assume that the numerical attributes are represented in one of two representations, i.e., non-binary representation (the aforementioned example) or binary representation (**Fig. 14**)[*7]. As the aforementioned example, the former representation is sometimes inconvenient when the size of description of access policy is large. The latter representation, on the other hand, can make the description of the access policy decrease to about the bit size of the value as mentioned in Ref. [3]. Therefore, to improve the usability, we are going to implement this functionality (syntactic sugar) such that our library converts the inequalities "<" and ">" in the description of an access policy into the appropriate description automatically (Fig. 14).

[*7] The administrator can choose one of these two according to application scenarios.

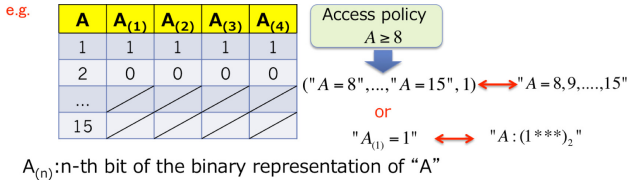

Fig. 14 Representation of numerical attribute.

4.2 Inner-Product Encryption

The ABE schemes like CP-FE and KP-FE possess the security property called payload hiding, which guarantees that a ciphertext associated with attribute I hides all information about the underlying message unless one holds a secret key giving the explicit ability to decrypt. However, these schemes may not be able to guarantee that a ciphertext hides all information about the associated attribute I. The inner-product encryption (IPE) scheme, on the other hand, guarantees that property, which is called attribute hiding. Attribute hiding security is the stronger notion of security than that of payload hiding security [11]. Therefore, by using the IPE scheme, a user can encrypt a data with all information about the associated attribute of the ciphertext secret.

The IPE scheme takes as an input $\Psi := \vec{v} \in \mathbb{F}_q^n$ and $\Upsilon := \vec{x} \in \mathbb{F}_q^n$, and, the encrypted data can be decrypted if and only if the inner product $\vec{x} \cdot \vec{v} = 0$. We are going to add the IPE functionality based on our implementation of CP/KE-FE.

5. Conclusion

In this paper, we proposed the implementation of the library of ciphertext-policy functional encryption and key-policy functional encryption based on the scheme of Refs. [15], [16]. As a future work, we will release the library as open source software for building security systems. Furthermore, we are planning to implement the additional functionality and the IPE scheme which are mentioned in Section 4.

Acknowledgments This work was supported in part by JSPS KAKENHI Grant Number 26330151, JSPS A3 Foresight Program, and JSPS and DST under the Japan - India Science Cooperative Program.

References

[1] Advanced Crypto Software Collection, available from ⟨http://hms.isi.jhu.edu/acsc/cpabe⟩.
[2] Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M.,

Green, M. and Rubin, A.D.: Charm: A framework for rapidly prototyping cryptosystems, *Journal of Cryptographic Engineering*, Vol.3, No.2, pp.111–128 (2013).
[3] Bethencourt, J., Sahai, A. and Waters, B.: Ciphertext-policy attribute-based encryption, *IEEE Symposium on Security and Privacy, 2007. SP'07*, pp.321–334, IEEE (2007).
[4] Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F. and Teruya, T.: High-speed software implementation of the optimal ate pairing over Barreto–Naehrig curves, *Pairing-Based Cryptography-Pairing 2010*, pp.21–39, Springer (2010).
[5] Boneh, D. and Franklin, M.: Identity-based encryption from the Weil pairing, *Advances in Cryptology–CRYPTO 2001*, pp.213–229, Springer (2001).
[6] Daemen, J. and Rijmen, V.: *The design of Rijndael: AES-the advanced encryption standard*, Springer Science & Business Media (2013).
[7] De Caro, A. and Iovino, V.: jPBC: Java pairing based cryptography, *2011 IEEE Symposium on Computers and Communications (ISCC)*, pp.850–855, IEEE (2011).
[8] Goyal, V., Pandey, O., Sahai, A. and Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data, *Proc. 13th ACM Conference on Computer and Communications Security*, pp.89–98, ACM (2006).
[9] Hasegawa, K., Kanayama, N., Nishide, T. and Okamoto, E.: Software Implementation of Ciphertext-Policy Functional Encryption with Simple Usability, *5th International Conference on IT Convergence and Security, ICITCS 2015*, Kuala Lumpur, Malaysia, pp.1–4 (online), DOI: 10.1109/ICITCS.2015.7293020 (2015).
[10] IEEE P1363.3:Identity-Based Public-key Cryptography Using Pairings (2008).
[11] Katz, J., Sahai, A. and Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products, *Advances in Cryptology–EUROCRYPT 2008*, pp.146–162, Springer (2008).
[12] libfenc: The Functional Encryption Library, available from ⟨http://code.google.com/p/libfenc⟩.
[13] Liu, Z. and Cao, Z.: On Efficiently Transferring the Linear Secret-Sharing Scheme Matrix in Ciphertext-Policy Attribute-Based Encryption, *IACR Cryptology ePrint Archive*, Vol.2010, p.374 (2010).
[14] Lynn, B.: The Pairing-Based Cryptography (PBC) library, available from ⟨http://crypto.stanford.edu/pbc/⟩.
[15] Okamoto, T. and Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption, *IACR Cryptology ePrint Archive*, Vol.2010, p.563 (2010) (online), available from ⟨http://eprint.iacr.org/2010/563⟩.
[16] Okamoto, T. and Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption, *Advances in Cryptology–CRYPTO 2010*, pp.191–208, Springer (2010).
[17] Sahai, A. and Waters, B.: Fuzzy identity-based encryption, *Advances in Cryptology–EUROCRYPT 2005*, pp.457–473, Springer (2005).
[18] Sakai, R., Ohgishi, K. and Kasahara, M.: Cryptosystems based on Pairing, *Proc. Symposium on Cryptography and Information Security (SCIS2000)* (2000).
[19] Shamir, A.: How to share a secret, *Comm. ACM*, Vol.22, No.11, pp.612–613 (1979).
[20] University of Tsukuba: TEPLA - LCIS, available from ⟨http://www.cipher.risk.tsukuba.ac.jp/tepla/⟩.
[21] Zavattoni, E., Perez, L.J.D., Mitsunari, S., Sanchez-Ramirez, A.H., Teruya, T. and Rodriguez-Henriquez, F.: Software implementation of an Attribute-Based Encryption scheme, *IEEE Trans. Comput.*, Vol.64, No.5, pp.1429–1441 (2015).

Appendix

A.1 Pairing

Pairing is a 2-input 1-output function which is defined over elliptic curves. Pairing has the following properties and functional encryption schemes can be realized efficiently by using these properties.

Definition A.1.1 (Pairing) *Let $\mathbb{G}$ and $\mathbb{G}_T$ be a cyclic additive group and multiplicative group of prime order q. Also let g be a generator of $\mathbb{G}$. Then, a mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is called pairing when the following properties are satisfied.*

(1) *Bilinearity: For all $a, b \in \mathbb{Z}_q$, $e(g^a, g^b) = e(g^b, g^a) = e(g, g)^{ab}$.*
(2) *Nondegeneracy: $e(g, g) \neq 1$.*
(3) *Computablity: Computable in polynomial-time.*

There exist several pairing software libraries to realize pairing operations efficiently (Refs. [7], [14], [20]), and there also exist several software libraries (Refs. [1], [2], [7], [12], [21]) using these pairing libraries.

A.2 Access Policy and Access Structure

In our CP-FE library, an access policy, i.e., a condition for decryption is given by a user as a logical formula. To deal with the access policy in an encryption scheme and realize access control, it is usually converted into an access structure, and as a result, we can combine an encryption scheme with a secret-sharing scheme [*8].

An access structure used in a secret-sharing scheme can be represented in several ways. In the scheme of Refs. [15], [16], an access structure is represented via a matrix called span program. Therefore, by converting an access policy into an access structure via a span program, we can realize access control such that only a user that has a proper set of attributes can decrypt a ciphertext.

A span program $\hat{M} := (M, \rho)$, an access structure represented by a span program $S := (M, \rho)$ (which is defined by adapting a span program to the scheme of Refs. [15], [16]), and a secret-sharing scheme based on a span program are defined as follows [15], [16].

(1) Span program
A span program $\hat{M} := (M, \rho)$ is a labeled matrix where M is an $\ell \times r$ matrix, and ρ is a labeling of the rows of M by literals from $\{p_1, p_2, ..., p_n, \neg p_1, ..., \neg p_n\}$, i.e., $\rho : \{1, ..., \ell\} \to \{p_1, ..., p_n, \neg p_1, ..., \neg p_n\}$. For every input sequence $\delta \in \{0, 1\}^n$, we define the submatrix M_δ of M consisting of those rows labels set to 1 by the input δ, i.e., either rows labeled by some p_i such that $\delta_i = 1$ or rows labeled by some $\neg p_i$ such that $\delta_i = 0$ (i.e., $\gamma : \{1, ..., \ell\} \to \{0, 1\}$ is defined by $\gamma(j) = 1$ if $[\rho(j) = p_i] \wedge [\delta_i = 1]$ or $[\rho(j) = \neg p_i] \wedge [\delta_i = 0]$, and $\gamma(j) = 0$ otherwise. $M_\delta := (M_j)_{\gamma(j)=1}$, where M_j is the j-th row of M.). The span program $\hat{M}$ accepts δ if and only if $\epsilon := \vec{1} \in \mathbf{span}\langle (M_j)_{\gamma(j)=1} \rangle$, i.e., some linear combination of the rows of M_δ gives ϵ, where ϵ is called target vector.

(2) Access structure (span program used in the scheme of Refs. [15], [16])
An access structure for the scheme of Refs. [15], [16] is a span program $\mathbf{S} := (M, \rho)$ along with variables $p := (t, \vec{v})$, where $t \in \{1, ..., d\}$ and $\vec{v} \in \mathbb{F}_q^{n_t}$, i.e., $\mathbf{S} := (M, \rho)$ such that $\rho : \{1, ..., \ell\} \to \{(t_1, \vec{v_1}), (t_2, \vec{v_2}), ..., \neg(t_1, \vec{v_1}), \neg(t_2, \vec{v_2})...\}$ (the variables p mean attributes in this scheme). Let Γ be a set of attributes, i.e., $\Gamma := \{(t, \vec{x_t}) \,|\, \vec{x_t} \in \mathbb{F}_q^{n_t} \backslash \{\vec{0}\}, 1 \leq t \leq d\}$. When Γ is given to access structure $\mathbf{S}$, map $\gamma(i) : \{1, ..., \ell\} \to \{0, 1\}$ is defined as follows. For $i = 1, ..., \ell$, set $\gamma(i) = 1$ if $[\rho(i) = (t, \vec{v_i})] \wedge [(t, \vec{x_t}) \in \Gamma] \wedge [\vec{v_i} \cdot \vec{x_t} = 0]$ or $[\rho(i) = \neg(t, \vec{v_i})] \wedge [(t, \vec{x_t}) \in \Gamma] \wedge [\vec{v_i} \cdot \vec{x_t} \neq 0]$, and set $\gamma(i) = 0$ other-

[*8] A secret-sharing scheme [19] is a method for distributing a secret to n members of a group. The secret can be reconstructed only when any shares of $t(\leq n)$ or more members are combined together, and the shares of fewer than t members cannot reconstruct the secret.

wise. The access structure $\mathbf{S} := (M, \rho)$ accepts Γ if and only if $\epsilon \in \mathbf{span}\langle (M_i)_{\gamma(i)=1} \rangle$.

(3) Secret-sharing scheme based on span program

Let a column vector $\vec{f}^{\mathrm{T}} := (f_1, ..., f_r)^{\mathrm{T}} \xleftarrow{\cup} \mathbb{F}_q^r$. Then, $s_0 := \vec{1} \cdot \vec{f}^{\mathrm{T}} = \sum_{k=1}^{r} f_k$ is the secret to be shared, and $\vec{s}^{\mathrm{T}} := (s_1, ..., s_\ell)^{\mathrm{T}} := M \cdot \vec{f}^{\mathrm{T}}$ is the vector of ℓ shares of the secret s_0 and the share s_i belongs to $\rho(i)$.

If a span program $\hat{M} := (M, \rho)$ accepts δ, or an access structure $\mathbf{S} := (M, \rho)$ accepts Γ, i.e., $\epsilon = \vec{1} \in \mathbf{span}\langle (M_i)_{(\gamma(i)=1)} \rangle$ with $\gamma : \{1, ..., \ell\} \to \{0, 1\}$, then there exist constants $\{\alpha_i \in \mathbb{F}_q\}$ such that $I \subseteq \{i \in \{1, ..., \ell\} \,|\, \gamma(i) = 1\}$ and $\sum_{i \in I} \alpha_i s_i = s_0$. Furthermore, these constants $\{\alpha_i\}$ can be computed in time polynomial in the size of matrix M.

For example, when an access policy Υ is specified as ((A, B, 2), (C, D, 1), 2), which means ((A and B) and (C or D)), in the CP-FE scheme, a span program matrix M of the access structure $\mathbf{S}$ corresponding to the access policy is generated as

$$M = \begin{pmatrix} 1 & 2 & 2 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & 3 & 1 & 1 \end{pmatrix}.$$

Then, $\rho(1)$ maps the first row of M to attribute "A", and similarly $\rho(i)$ $(2 \le i \le 4)$ maps the i-th row of M to each attribute "B", "C", "D" respectively. When a user has a set of attributes Ψ = {A, B, D}, $M_{i(\gamma(i)=1)}$ is

$$M_{i(\gamma(i)=1)} = \begin{pmatrix} 1 & 2 & 2 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 3 & 1 & 1 \end{pmatrix}$$

and, the shares that the user possesses (s_1, s_2, s_4) are

$$s_1 = f_1 + 2f_2 + 2f_3 + f_4$$
$$s_2 = f_1 + 2f_2 + 3f_3 + f_4$$
$$s_4 = f_1 + 3f_2 + f_3 + f_4$$

then, $\epsilon = \vec{1} \in \mathbf{span}\langle (M_i)_{(\gamma(i)=1)} \rangle$. Therefore, there exist $\{\alpha_i \in \mathbb{F}_q\}$ such that $I \subseteq \{i \in \{1, ..., \ell\} \,|\, \gamma(i) = 1\}$ $(\alpha_1 = 4, \alpha_2 = -2, \alpha_4 = -1)$, and the secret s_0 can be computed because

$$\alpha_1 \cdot s_1 + \alpha_2 \cdot s_2 + \alpha_4 \cdot s_4$$
$$= f_1 + f_2 + f_3 + f_4$$
$$= s_0$$

and the user can decrypt the encrypted data.

A.2.1 Converting Access Policy to Matrix

The algorithm to convert an access policy to a span program matrix needs to be designed carefully because the size of the matrix generated from an access policy including t-out-of-n threshold gates may become large if it is done in a naive way. Therefore, there is a need to perform the conversion efficiently. The conversion algorithm used in the library is composed by combining the following two basic operations [13].

(1) Constructing Access Structure from General t-out-of-n Threshold Gate

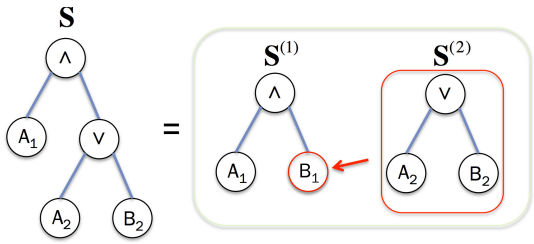

Fig. A·1 Inserting access structure.

An access structure of t-out-of-n threshold gate can be satisfied by only possessing arbitrary more than t shares from n shares. This access structure M is defined as follows.

$$M = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & \cdots & 2^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & n & n^2 & \cdots & n^{t-1} \end{pmatrix}$$

(2) Inserting Access Structure

When converting an access structure $\mathbf{S}$ like **Fig. A·1**, it is constructed by inserting $\mathbf{S}^{(2)}$ to the row of B_1 in $\mathbf{S}^{(1)}$. The matrix $\mathbf{S}^{(1)}, \mathbf{S}^{(2)}$ is constructed by Section A.2.1 (1).

The case where $m_2 \times d_2$ matrix $M^{(2)}$ is inserted into $m_1 \times d_1$ matrix $M^{(1)}$ is considered. $M^{(1)}$, $M^{(2)}$ can be expressed as

$$M^{(1)} = \begin{pmatrix} \tilde{M}^{(11)} \\ v \\ \tilde{M}^{(12)} \end{pmatrix}, M^{(2)} = \begin{pmatrix} u^{(2)} & \tilde{M}^{(2)} \end{pmatrix}$$

respectively, such that v is the row into which $M^{(2)}$ is inserted, and $u^{(2)} = (u_1, u_2, ..., u_{m_2})^{\mathrm{T}}$ (u_i is the first column of $M_i^{(2)}$). The resultant matrix M is expressed as

$$M = \begin{pmatrix} \tilde{M}^{(11)} & 0 \\ v \otimes u^{(2)} & \tilde{M}^{(2)} \\ \tilde{M}^{(12)} & 0 \end{pmatrix}$$

where,

$$v \otimes u^{(2)} = \begin{pmatrix} u_1 v \\ u_2 v \\ \vdots \\ u_{m_2} v \end{pmatrix}$$

In Refs. [15], [16], the target vector ϵ of the matrix is $(1, 1, 1, ..., 1) = \vec{1}$, though the target vector of the matrix generated by this conversion algorithm is $(1, 0, 0, ..., 0)$. Therefore, there is a need to add the first column to the other columns of the generated matrix in the last step to make the target vector $\vec{1}$. For example, access policy $L = ((A, B, 1), (C, D, E, 2), 2)$ is converted to the matrix M as follows by using this algorithm.

$$M = \begin{pmatrix} 1 \end{pmatrix}, \quad L = \begin{pmatrix} (A, B, 1), (C, D, E, 2), 2 \end{pmatrix}$$
$$M = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \quad L = \begin{pmatrix} (A, B, 1) \\ (C, D, E, 2) \end{pmatrix}$$
$$M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}, \quad L = \begin{pmatrix} A \\ B \\ (C, D, E, 2) \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}, \quad L = \begin{pmatrix} A \\ B \\ C \\ D \\ E \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 3 & 2 \\ 1 & 3 & 3 \\ 1 & 3 & 4 \end{pmatrix}$$

Keisuke Hasegawa received his B.S. and M.S. degrees from University of Tsukuba in 2014 and 2016, respectively. He is currently working at SECOM Intelligent Systems Laboratory.

Naoki Kanayama received his B.E., B.S., M.S. and D.S. degrees from Waseda University, Tokyo, Japan, in 1904, 1996, 1998 and 2003, respectively. In 2003–2006, he was a postdoctoral fellow of the Japan Society for the Promotion of Science. In 2006–2013, he was a research fellow at University of Tsukuba. He is an assistant professor at University of Tsukuba. Dr. Kanayama is a member of the Japan Society for Industrial and Applied Mathematics and of the Information Processing Society of Japan.

Takashi Nishide received his B.S. degree from the University of Tokyo in 1997, M.S. degree from the University of Southern California in 2003, and Dr.E. degree from the University of Electro-Communications in 2008. From 1997 to 2009, he had worked at Hitachi Software Engineering Co., Ltd. developing security products. From 2009 to 2013, he had been an assistant professor at Kyushu University and from 2013 he is an associate professor at University of Tsukuba. His research is in the areas of cryptography and information security.

Eiji Okamoto received his B.S., M.S. and Ph.D. degrees in electronics engineering from Tokyo Institute of Technology in 1973, 1975 and 1978, respectively. He worked and studied communication theory and cryptography for NEC central research laboratories since 1978. In 1991 he became a professor at Japan Advanced Institute of Science and Technology, then at Toho University. Now he is a professor at Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests are cryptography and information security. He is members of IEEE and ACM.