Function Secret Sharing Using Fourier Basis

Takuya Ohsawa[1], Naruhiro Kurokawa[2], and Takeshi Koshiba[3](✉)

[1] Graduate School of Science and Engineering, Saitama University, Saitama, Japan
[2] Information System Services Department, Bank of Japan, Tokyo, Japan
[3] Faculty of Education and Integrated Arts and Sciences, Waseda University, Tokyo, Japan
tkoshiba@waseda.jp

Abstract. Function secret sharing (FSS) scheme, formally introduced by Boyle et al. at EUROCRYPT 2015, is a mechanism that calculates a function $f(x)$ for $x \in \{0,1\}^n$ which is shared among p parties, by using distributed functions $f_i : \{0,1\}^n \to \mathbb{G}$ $(1 \le i \le p)$, where $\mathbb{G}$ is an Abelian group, while the function $f : \{0,1\}^n \to \mathbb{G}$ is kept secret to the parties. We observe that any function f can be described as a linear combination of the basis functions by regarding the function space as a vector space of dimension 2^n and give a new framework for FSS schemes based on this observation. Based on the new framework, we introduce a new FSS scheme using the Fourier basis. This method provides efficient computation for a different class of functions (e.g., hard-core predicates of one-way functions), which may be inefficient to compute if we use the standard basis such as point functions. Our FSS scheme based on the Fourier basis is quite simple due to the fact that the Fourier basis is closed under the multiplication, while the previous constructions have to incorporate some complex mechanisms to overcome the difficulty.

1 Introduction

The notion of secret sharing (SS) schemes was introduced by Blakley [3] and Shamir [18]. In SS schemes, share information S_i $(0 \le i \le p)$, generated from the secret information S, are distributed to p parties. In (n,p)-threshold SS scheme, the secret information S can be recovered from n shares, but no information on S is leaked from at most $n-1$ shares. A simple (n,n)-threshold SS scheme can be realized as follows: for a secret information $S \in \mathbb{G}$, share information S_i $(1 \le i \le p)$ are selected so as to satisfy $S = \sum_{i=1}^{p} S_i$. (We will use this simple SS scheme as an ingredient of our schemes.)

As an extension of SS schemes, an idea that functions would be secretly distributed to several parties has been arisen. Gilboa et al. [11] consider distributed point function (DPF) to distribute point functions $f_{a,b} : \{0,1\}^n \to \mathbb{G}$, where $f_{a,b}(x) = b$ if $x = a$ for some $a \in \{0,1\}^n$ and $f_{a,b}(x) = 0$ otherwise. In a basic DPF scheme, the function f is partitioned into two keys f_0, f_1 and each

This article does not include the official view of Bank of Japan.

L. Barolli et al. (eds.), *Advances in Network-based Information Systems*,
Lecture Notes on Data Engineering and Communications Technologies 7,
DOI 10.1007/978-3-319-65521-5_78

key is distributed to the respective party of the two parties. Each party calculates the share $y_i = f_i(x)$ for common input x by using the key f_i. They can recover the solution of the point function $f_{a,b}(x)$ by summing up two shares of the two parties. Boyle et al. [6] study the efficiency in the key size and introduce a DPF scheme in the multi-party setting. Moreover, they generalize the target functions (i.e., point functions) to some class of functions, and propose an FSS scheme for some function class $\mathscr{F}$ in which functions $f : \{0,1\}^n \to \mathbb{G}$ can be calculated efficiently. In the FSS scheme we partition a function $f \in \mathscr{F}$ into p distributed functions $(f_1, \dots, f_p)$. Like SS schemes mentioned above, an equation $f(x) = \sum_{i=1}^{p} f_i(x)$ is satisfied with respect to any x, and the information about the secret function f (except the domain and the range) does not leak out from at most $p-1$ distributed functions. Moreover, distributed functions f_i can be described as short keys k_i and it is required to be efficiently evaluated. Boyle et al. constructs FSS schemes for interval functions and for partial matching functions by the essential use of pseudo-random generators (PRG) like the Goldreich-Goldwasser-Micali [12] construction of pseudo random functions. They also show that FSS schemes for functions which can be calculated in polynomial time are implemented by using some obfuscation techniques and one-way functions. Furthermore, Boyle et al. [7] recently construct a two-party FSS scheme for *branching programs* under the DDH assumption, which implies low-communication two-party secure computation protocols. A p-party FSS scheme for *circuits* was proposed in [10] under the LWE assumption, by using multi-key fully-homomorphic encryption.

As an application of FSS schemes, we can consider private information retrieval (PIR) [8,9,15]. For example, we can know how many times some keyword w appears in a distributed database D without revealing the keyword w. A client first distributes the point function $f_{w,1}$ into k distributed functions $(f_1, \dots, f_k)$. Each distributed database calculates the share value $\sum_{w_j \in D} f_i(w_j)$ and send it back to the client. After that, the client can obtain $f_{w,1}(w)$ from all shares. If an FSS scheme can distribute arbitrary functions, it enables us to compute the functional value without revealing any information on the algorithm to evaluate the function to servers.

In this paper, we consider an FSS scheme for *hard-core predicates* of one-way functions. A predicate $b(x)$ is hard-core for a one-way function $f(x)$ if $b(x)$ is computationally unpredictable given $f(x)$. It is well known that there exists a hard-core predicate for any one-way function [13]. From candidates of one-way functions such as the RSA function and other number-theoretic functions, the existence of hard-core bits are shown [2,4,5,14]. Akavia, Goldwasser and Safra [1] consider a new framework for proving hard-core properties in terms of Fourier analysis. Any predicates can be represented as a linear combination of Fourier basis functions. Akavia et al. show that if the number of non-zero coefficients in the Fourier representation of hard-core predicates is polynomially bounded then the coefficients are efficiently approximable. This fact leads to the hard-core properties. Actually, several hard-core predicates can be discussed in the framework of Akavia et al. [1,16]. Hard-core predicates are useful not only in theory but in practice. By using hard-core predicate, pseudorandom generators and many cryptographic protocols such as bit commitment and oblivious transfer can be obtained. Besides hard-core predicates, it is known that low-degree polynomials are Fourier-concentrated [17].

1.1 Our Contribution

We observe that any function $f : \{0,1\}^n \to \{0,1\}$ can be described as a linear combination of the basis functions by regarding the function space as a vector space of dimension 2^n and give a natural framework for FSS schemes based on this observation. We see that an FSS scheme for point functions can be discussed in this framework. Any point function corresponds with a 2^n-dimensional vector whose elements are all zero but one axis. Such a vector of the special form has a succinct representation because its Kolmogorov complexity is quite small. However, point functions are not so easy to partition into several distributed functions while keeping the succinct representation. We can regard the scheme by Gilboa et al. for point functions as such a realization.

Instead of the standard basis (point functions), we consider to use the Fourier basis. Then we construct a new FSS scheme using the Fourier basis. This method provides efficient computation for some classes of functions (e.g., hard-core predicates of one-way functions) which may be inefficient to compute if we use the standard basis (as point functions). As mentioned, many hard-core predicates have succinct Fourier representation. Namely, the number of non-zero Fourier coefficients for hard-core predicates is polynomially bounded. We will see that our new FSS scheme is available for hard-core predicates.

Our FSS scheme based on the Fourier basis is simpler than the previous schemes. This is because of the fact that the Fourier basis is closed under the multiplication.

Finally, we would like to stress that our contribution is rather providing a new framework that enables us simple constructions of FSS schemes. We believe that our framework gives a new insight for FSS schemes.

2 Preliminaries

2.1 Definitions

We review a formal definition of FSS schemes, given in [6]. In an FSS scheme, we partition a function f into keys k_i (the succinct descriptions of f_i) which the corresponding parties P_i receive. Each party P_i calculates the share $y_i = f_i(x)$. The functional value $f(x)$ is obtained from all shares $\mathbf{y} = (y_1, y_2, \ldots, y_p)$ of the parties by using a decode function *Dec*. Each key k_i does not leak any information on function f except the domain and the range of f. We first define the decoding process from shares.

Definition 1 (Output Decoder). An output decoder *Dec* from shares of p parties is a tuple $(S_1, \ldots, S_p, R, Dec)$, where S_i $(1 \le i \le p)$ is a share space of the

i-th party, R is the output space, and $Dec : S_1 \times \ldots \times S_p \to R$ is a decoding function.

Next, we define FSS schemes. We denote, by $[p]$, the set of integers $\{1, 2, \ldots, p\}$, which corresponds to p parties. We assume that $T \subsetneq [p]$ be a set of corrupted parties.

Definition 2. For any $p \in \mathbb{N}$, $T \subsetneq [p]$, a (p-party, T-secure) FSS scheme with respect to a function class $\mathscr{F}$ is a pair of PPT algorithms $(Gen, Eval)$ satisfying the following.

- The key generation algorithm $Gen(1^\lambda, f)$, on input the security parameter 1^λ and a function $f : D \to R$ in $\mathscr{F}$, outputs p keys $(k_1, \ldots, k_p)$.
- The evaluation algorithm $Eval(i, k_i, x)$, on input a party index i, a key k_i, and an element $x \in D$, outputs a value $y_i \in S_i$, corresponding to the i-th party's share of $f(x)$.

Moreover, these algorithms must satisfy the following properties:

- *Correctness*: For all $f \in \mathscr{F}$ and $x \in D$,

$$\Pr[Dec(Eval(1, k_1, x), \ldots, Eval(p, k_p, x)) = f(x) \mid (k_1, \ldots, k_p) \leftarrow Gen(1^\lambda, f)] = 1.$$

- *Security* : Consider the following indistinguishability challenge experiment for corrupted parties T:
  1. The adversary $\mathscr{A}$ outputs $(f_0, f_1) \leftarrow \mathscr{A}(1^\lambda)$, where $f_0, f_1 \in \mathscr{F}$.
  2. The challenger chooses $b \leftarrow \{0, 1\}$ and $(k_1, \ldots, k_p) \leftarrow Gen(1^\lambda, f_b)$.
  3. $\mathscr{A}$ outputs a guess $b' \leftarrow \mathscr{A}((k_i)_{i \in T})$, given the keys for corrupted parties T.

The advantage of the adversary $\mathscr{A}$ is defined as $Adv(1^\lambda, \mathscr{A}) := \Pr[b = b'] - 1/2$. We say the scheme $(Gen, Eval)$ is T-secure if there exists a negligible function ν such that for all non-uniform PPT adversaries $\mathscr{A}$ which corrupts parties in T, it holds that $Adv(1^\lambda, \mathscr{A}) \leq \nu(\lambda)$.

FSS scheme for the point functions
We give the definition of point functions below.

Definition 3. For $a, b \in \{0, 1\}^n$, the point function $P_{a,b} : \{0, 1\}^n \to \{0, 1\}^m$ is defined by $P_{a,b}(a) = b$ and $P_{a,b}(a') = 0^m$ for all $a' \neq a$.

Gilboa et al. [11] propose a 2-party FSS scheme for point functions, and Boyle et al. [6] improve the efficiency of key size for the 2-party FSS. Let $(Gen^\bullet, Eval^\bullet)$ be the FSS scheme in [6] for point functions.

$Gen^\bullet(1^\lambda, a, b)$ is an algorithm which distributes the point function $P_{a,b}$ to two keys k_0, k_1 for two parties. In their scheme, each key is a binary tree of depth n where pseudo-random numbers (generated by a PRG) are assigned to all nodes. Binary trees k_0, k_1 are made exactly the same structure with the exception of the path from the root to the target point $a = a_1, \ldots, a_n$. With respect to the path from the root to the target point $a = a_1, \ldots, a_n$, the corresponding paths in both binary trees are completely independent.

Eval$^\bullet(\beta, k_\beta, x)$ is an algorithm which calculates the shares of $P_{a,b}$ of k_β for $\beta \in \{0,1\}$. An input x determines the path in the binary tree k_β. The algorithm repeats a recursive process from the root to obtain random numbers $S^\beta_{x_i}$ for all $i \in [n]$ by using the PRG G and finally outputs the share $G(S^\beta_{x_n}) \cdot w$, where $w = (G(S^0_{a_n}) + G(S^1_{a_n}))^{-1} \cdot b$. So, if $x = a$ then the sum of the shares of the two parties is equal to $G(S^0_{a_n}) \cdot w + G(S^1_{a_n}) \cdot w = (G(S^0_{a_n}) + G(S^1_{a_n})) \cdot w = b$, which coincides with the output value b of the point function. This means that the condition $S_0 \neq S_1$ is required and if the event $S^0_{a_n} = S^1_{a_n}$ occurs in the execution of *Gen*$^\bullet$ then another execution of *Gen*$^\bullet$ is required.

Here we would like to stress that the above mentioned scheme by Boyle et al. is quite complex. The other existing schemes are also complex. This complexity comes from the hardness of compatibility between succinct representation of point functions and secure partitioning of the point functions.

2.2 Basis Functions

The function space of functions $f : \{0,1\}^n \to \mathbb{C}$ can be regarded as a vector space of dimension 2^n. Therefore, the basis vectors for the function space exist and we call the basis function $h_i(x)$. Any function f in the function space is described as a linear combination of the basis functions $f(x) = \sum_{j\in\{0,1\}^n} \beta_j h_j(x)$, where β_j's are coefficients in $\mathbb{C}$. The point functions mentioned above can be considered as basis functions, so we can describe any functions by a linear combination of the point functions.

The Fourier basis

Let $f : \{0,1\}^n \to \mathbb{C}$. The Fourier transform of the function f is defined as

$$\hat{f}(a) = \sum_{x\in\{0,1\}^n} f(x) e^{-\pi i\langle a,x\rangle}, \tag{1}$$

where $\langle a, x\rangle$ is the inner product $\sum_{j=1}^n a_j x_j$. Then, $f(x)$ can be described as a linear combination of the basis functions $\chi_a(x) = e^{\pi i\langle a,x\rangle}$, that is,

$$f(x) = \sum_{a\in\{0,1\}^n} \hat{f}(a)\chi_a(x).$$

In the above, $\hat{f}(a)$ is called Fourier coefficient of $\chi_a(x)$. As a consequence of Euler's formula $e^{\pi i} = -1$ we have

$$\chi_a(x) = (-1)^{\langle a,x\rangle}.$$

It is easy to see that the Fourier basis is orthonormal since

$$\frac{1}{2^n} \sum_{x\in\{0,1\}^n} \chi_a(x)\chi_b(x) = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In this paper, we consider only Boolean functions and assume that the range of the boolean function is $\{\pm 1\}$ instead of $\{0, 1\}$ without loss of generality. That is, we regard boolean functions as mappings from $\{0, 1\}^n$ to $\{\pm 1\}$. Also, we have

$$\chi_{a+b}(x) = \chi_a(x)\chi_b(x).$$

This multiplicative property plays an important role in this paper.

Let $\mathscr{B}_P = \{P_{a,1} \mid a \in \{0, 1\}^n\}$ and $\mathscr{B}_F = \{\chi_a \mid a \in \{0, 1\}^n\}$ be the sets of the basis functions with respect to point functions and the Fourier basis respectively. Finally note that the above form of Fourier transform is often called Walsh-Hadamard transform.

3 Our Proposal

FSS schemes for some function classes such as point functions, interval function, and partial pattern matching functions are proposed by Boyle et al. As mentioned, any function can be described as a linear combination of basis functions. If the function is described as a linear combination of a super-polynomial number of basis functions, then the computational cost for evaluating the function might be inefficient. We say that a function has a *succinct* description (with respect to the basis $\mathscr{B}$) if the function f is described as $f(x) = \sum_{h \in \mathscr{B}'} \beta_h h(x)$ for some $\mathscr{B}' \subset \mathscr{B}$ such that $|\mathscr{B}'|$ is polynomially bounded in n. If we can find a good basis set $\mathscr{B}$, some functions may have a succinct description with respect to $\mathscr{B}$. We consider to take the Fourier basis as such a good basis candidate.

We will provide an FSS scheme for some function class whose elements are functions with succinct description with respect to the Fourier basis $\mathscr{B}_F$. Since the Fourier basis has nice properties, our FSS scheme can be more simply realized than the previous FSS schemes.

As mentioned in Sect. 1, Akavia et al. [1] show that many hard-core predicates have succinct description with respect to the Fourier basis $\mathscr{B}_F$. Thus, our proposal can be used as an FSS scheme for hard-core predicates.

3.1 General FSS Scheme for Succinct Functions

As mentioned, any function can be described as a linear combination of basis functions. Before mentioning an FSS scheme with respect to the Fourier basis, we give a general construction of a naive FSS scheme for some succinct function f with respect to some basis $\mathscr{B}$ from an FSS scheme for each basis function in $\mathscr{B}$. That is, $f(x) = \sum_{h_i \in \mathscr{B}'} h_i(x)$ for some $\mathscr{B}' \subset \mathscr{B}$ such that $|\mathscr{B}'|$ is polynomially bounded in n. Let $\mathscr{F}_{\mathscr{B},\ell}$ be a class of functions f which can be represented as a linear combination of ℓ basis functions (with respect to $\mathscr{B}$) at most, where ℓ is a polynomial in n. Suppose that a p-party FSS scheme $(Gen^h_{Multi}, Eval^h_{Multi}, Dec^h_{Multi})$ for a basis function h satisfies the following.

- $Gen^h_{Multi}(1^\lambda, h)$: On input the security parameter 1^λ and a basis function h, the key generation algorithm outputs p keys $(k_1, \ldots, k_p)$.

Algorithm 1. $Gen(1^\lambda, f)$

```
for $i = 1$ to $\ell$ do
    $(k_1^i, k_2^i, \ldots, k_p^i) \leftarrow Gen_{Multi}^h(1^\lambda, h_i)$
end for
for $j = 1$ to $p$ do
    Set $\mathbf{k_j} \leftarrow (k_j^1, k_j^2, \ldots, k_j^\ell)$
end for
Return $(\mathbf{k_1}, \mathbf{k_2}, \ldots, \mathbf{k_p})$
```

Algorithm 2. $Eval(i, \mathbf{k_i}, x)$

```
for $j = 1$ to $\ell$ do
    $y_j^i \leftarrow Eval_{Multi}^h(i, k_j^i, x)$
end for
Return $\mathbf{y_i} = (y_1^i, y_2^i, \ldots, y_\ell^i)$
```

Algorithm 3. $Dec(\mathbf{y_1}, \ldots, \mathbf{y_p})$

```
for $i = 1$ to $\ell$ do
    $g_i \leftarrow Dec_{Multi}^h(y_i^1, \ldots, y_i^p)$
end for
Return $g = \sum_{i=1}^{\ell} g_i$
```

- $Eval_{Multi}^h(i, k_i, x)$: On input a party index i, a key k_i, and an input string $x \in \{0,1\}^n$, the evaluation algorithm outputs a value $y_i \in S_i$, corresponding to i-th party's share of $h(x)$.
- $Dec_{Multi}^h(y_1, \ldots, y_p)$: On input shares $y_1, \ldots, y_p$ of all the parties, the decryption algorithm outputs a solution $h(x)$ of the basis function h for x.

By using the above FSS scheme $(Gen_{Multi}^h, Eval_{Multi}^h, Dec_{Multi}^h)$ for basis functions, we construct an FSS scheme $(Gen, Eval, Dec)$ for succinct functions $f \in \mathscr{F}_{\mathscr{B},\ell}$: Algorithm 1 for *Gen*, Algorithm 2 for *Eval* and Algorithm 3 for *Dec*. In this construction, we partition each basis function h_i into p keys $(k_1^i, k_2^i, \ldots, k_p^i)$ and set a key of f for the j-th party P_j as $\mathbf{k_j} = (k_j^1, k_j^2, \ldots, k_j^i)$. Each party P_j which gets a key vector $\mathbf{k_j}$ executes *Eval* on every component k_j^i of $\mathbf{k_j}$ and obtains a vector $\mathbf{y_j} = (y_j^1, y_j^2, \ldots, y_j^n)$, which is a share of the input x. We collect all the share vectors $(\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_p})$ and execute *Dec* on all the i-th elements over the share vectors for every i. Then, by summing up all resulting values from *Dec*, we can obtain the solution $f(x)$ for the input x of the function f. Since all the basis functions h_i of the target function f are hidden, f is also kept hidden.

The correctness of $(Gen, Eval, Dec)$ just comes from the correctness of each FSS $(Gen_{Multi}^h, Eval_{Multi}^h, Dec_{Multi}^h)$ for the basis function h. But some care must be done. From the assumption, $f \in \mathscr{F}_{\mathscr{B},\ell}$ has ℓ terms at most. If we represent f as a linear combination of exactly ℓ terms, some coefficients for basis functions must be zero. Since the 0-function which maps any element $x \in D$ to 0 can be partitioned into several functions as the basis functions can be, we can apply $(Gen_{Multi}^h, Eval_{Multi}^h, Dec_{Multi}^h)$ as well.

The security of $(Gen, Eval, Dec)$ is also straightforward. Since basis functions are linearly independent orthogonal vectors, any information on some basis function h_i is independent to the information of the other basis function h_j where $i \neq j$. Thus, a simple reduction to $(Gen_{Multi}^h, Eval_{Multi}^h, Dec_{Multi}^h)$ guarantees the security of $(Gen, Eval, Dec)$.

Note that if we do not care about the leakage of the number of terms with non-zero coefficients for f, we can omit the partitioning of zero-functions, which increases the efficiency of the scheme.

3.2 FSS Scheme for Succinct Linear Combinations of Point Functions

Here, we give a slight generalization of Boyle et al's FSS scheme for point functions. In this generalization, we give an FSS scheme for functions which can be described by linear combinations of poly(n) point functions $P_{a,b}$, that is, $f(x) = P_{a_1,b_1} + P_{a_2,b_2} + \cdots + P_{a_{\mathrm{poly(n)}},b_{\mathrm{poly(n)}}}$. The generalization can be realized by using Boyle et al's scheme as $(Gen^h_{Multi}, Eval^h_{Multi}, Dec^h_{Multi})$ discussed in the previous subsection.

3.3 FSS Scheme for the Fourier Basis

In this subsection, we give FSS schemes for the Fourier basis. As mentioned, any boolean function is described as a linear combination of the Fourier basis functions $\chi_a(x) = (-1)^{\langle a,x\rangle}$. Since we do not know how to evaluate any function efficiently, we limit ourselves to succinct functions with respect to the Fourier basis $\mathscr{B}_F$. Note that succinct functions with respect to $\mathscr{B}_F$ do not coincide with succinct functions with respect to point functions $\mathscr{B}_P$. Simple periodic functions are typical examples of succinct functions with respect to $\mathscr{B}_F$, which might not be succinct functions with respect to $\mathscr{B}_P$. As mentioned, some hard-core predicates of one-way functions are succinct functions with respect to $\mathscr{B}_F$. Thus, our results extend the applicability of FSS schemes to a wider class of functions. Our construction is much simpler than Boyle et al's FSS for point functions which essentially uses a pseudorandom generator. Moreover, the key length of our scheme is $O(\lambda)$ for the security parameter λ.

3.3.1 2-Party FSS Scheme for the Fourier Basis

First, we consider a 2-party FSS scheme for the Fourier basis. Our scheme consists of three algorithms Gen^2_F (Algorithm 4), $Eval^2_F$ (Algorithm 5), and Dec^2_F (Algorithm 6). Gen^2_F is an algorithm that divides a into two random strings a_0 and a_1 so as to satisfy that $a = a_0 + a_1$. Note that this division of a is a (2, 2)-threshold SS scheme. Moreover, this implies that a Fourier basis $\chi_a(x) = (-1)^{\langle a,x\rangle}$ can be decomposed into the product of two Fourier basis functions

$$\chi_a(x) = (-1)^{\langle a,x\rangle} = (-1)^{\langle a_0,x\rangle} \cdot (-1)^{\langle a_1,x\rangle} = \chi_{a_0}(x) \cdot \chi_{a_1}(x). \tag{3}$$

By regarding $(-1)^{\langle a_0,x\rangle}$ and $(-1)^{\langle a_1,x\rangle}$ as keys k_0, k_1 respectively, we distribute them to both parties P_0, P_1. In $Eval^F_2$, each party obtains the share by feeding x to the function distributed as the key. Dec^F_2 is invoked in order to obtain the

Algorithm 4. $Gen_2^F(1^\lambda, a = a_1 \cdots a_n)$

```
Choose a bit string $a_0 \in \{0,1\}^n$ uniformly
$a_1 = a \oplus a_0$
$k_0 = a_0$
$k_1 = a_1$
Return $(k_0, k_1)$
```

Algorithm 5. $Eval_2^F(i, k_i, x = x_1 \cdots x_n)$

```
Parse $k_i$ as $k_i = (a_{i,1} \| a_{i,2} \| \ldots \| a_{i,n})$
$w_i = \bigoplus_{j=1}^{n} x_j a_{i,j}$.
Return $w_i$.
```

Algorithm 6. $Dec_2^F(w_1, w_2)$

```
Let $ans = w_0 \oplus w_1$.
Return $w = (-1)^{ans}$.
```

Fourier basis function $\chi_a(x)$ from the shares. Distributed keys k_0, k_1 do not leak any information about the Fourier basis function $\chi_a(x)$ because a_0 is uniformly random. Unlike the scheme by Boyle et al., *Gen* does not err, since there is no restriction on choosing random numbers.

Correctness:
Due to the construction, the correctness follows from Eq. (3).

Security:
We show the above scheme is (2-party, 1-secure) FSS scheme. We assume that an adversary $\mathscr{A}$ chooses (f_0, f_1) where $f_0 = \chi_a$ and $f_1 = \chi_b$. Then the challenger chooses a random bit c to select f_c and invokes $Gen_2^F(1^\lambda, a)$ if $c = 0$ and $Gen_2^F(1^\lambda, b)$ if $c = 1$. If $c = 0$ then a is divided into two random strings a_0 and a_1 such that $a = a_0 \oplus a_1$. If $c = 1$ then b is divided into two random strings b_0 and b_1 such that $b = b_0 \oplus b_1$. In any case, the adversary $\mathscr{A}$ accesses one of four Fourier basis functions χ_{a_0}, χ_{a_1}, χ_{b_0} and χ_{b_1}. Even if the adversary $\mathscr{A}$ knows a_i (resp., b_i), $\mathscr{A}$ cannot guess a_{1-i} (resp., b_{1-i}) with probability more than 1/2. Thus, $\mathscr{A}$ cannot guess the values a and b because they are randomly masked. It implies that only $\mathscr{A}$ can do for guessing the random bit c selected by the challenger is just a random guess. So, $Adv(1^\lambda, \mathscr{A}) = 0$.

3.3.2 Multi-party FSS for the Fourier Basis

It is easy to extend our 2-party FSS for the Fourier basis to a multi-party scheme. (Note that it is difficult to extend 2-party FSS schemes for the point functions to the multi-party case. Actually, Gilboa et al's 2-party FSS scheme [11] for point functions and Boyle et al's multi-party FSS scheme [6] are quite different in techniques.)

Since $\chi_a(x)$ can be decomposed into the product $\chi_a(x) = \prod_{i=1}^{p} (-1)^{a_i \cdot x}$, we send a_i to the i-th party P_i. We similarly define the three algorithms Gen_{Multi}^F (Algorithm 7), $Eval_{Multi}^F$ (Algorithm 8), and Dec_{Multi}^F (Algorithm 9) as multi-party FSS scheme for the Fourier basis.

Algorithm 7. $Gen^F_{Multi}(1^\lambda, a)$

```
for $i = 1$ to $p - 1$ do
    Choose $a_i \in \{0,1\}^n$ uniformly
    $k_i = a_i$
end for
$a_p = a \oplus a_1 \oplus a_2 \oplus \cdots \oplus a_{p-1}$,
$k_p = a_p$
Return $(k_1, k_2, \ldots, k_p)$
```

Algorithm 8. $Eval^F_{Multi}(i, k_i, x)$

```
Parse $k_i$ as $k_i = (a_{i,1} \| a_{i,2} \| \cdots \| a_{i,n})$
$w_i = \bigoplus_{j=1}^n x_j a_{i,j}$
Return $w_i$
```

Algorithm 9. $Dec^F_{Multi}(w_1, w_2, \ldots, w_p)$

```
Let $ans = w_1 \oplus w_2 \oplus \cdots \oplus w_p$.
Return $(-1)^{ans}$
```

As in the two-party case, the correctness of the above multi-party FSS scheme can be shown. Also the security follows from the perfect security of (p,p)-threshold SS scheme. Actually, the above scheme is (p-party, $(p-1)$-secure) FSS scheme.

As mentioned, our multi-party FSS scheme is a straightforward generalization of the two-party case. This heavily owes the multiplicative property of Fourier basis functions, which can relate to a simple (p,p) threshold SS scheme.

3.4 FSS Scheme for Succinct Functions w.r.t. the Fourier Basis

As mentioned in Subsect. 2.2, any function is described by a linear combination of the Fourier basis functions. Let f be a succinct Boolean function with respect to $\mathscr{B}_F$, that is $f(x) = \sum_{a \in A} \beta_a \chi_a(x)$ for some $A \subset \{0,1\}^n$ such that $|A|$ is a polynomially bounded in n. (Note that if f is a Boolean function then all coefficients β_a are either 0 or 1.) By using $(Gen^F_2, Eval^F_2, Dec^F_2)$ or $(Gen^F_{Multi}, Eval^F_{Multi}, Dec^F_{Multi})$, we can construct a general FSS scheme $(Gen, Eval, Dec)$ for succinct Boolean functions with respect to $\mathscr{B}_F$.

4 Conclusion

By observing that any function can be described as a linear combination of the basis functions, we have provided simple FSS schemes for several classes of functions. Especially for succinct functions with respect to the Fourier basis, our scheme is much simpler than the previous FSS schemes. Since the class of succinct functions with respect to the Fourier basis is different from point functions and their variants, our results broaden functions for which FSS schemes are available.

Furthermore, to distribute a function, we have used a simple (n,n) threshold SS scheme. That is, to divide a secret a, we randomly choose $a_1, \ldots, a_n$ such that $a = a_1 \oplus \cdots \oplus a_n$. An interesting open question is as follows: Can we use instead Shamir's (n,k) threshold scheme to construct (n-party, $(k-1)$-secure) FSS?

Acknowledgements. The third author is supported in part by JSPS Grant-in-Aids for Scientific Research (A) JP16H01705 and for Scientific Research (B) JP17H01695.

References

1. Akavia, A., Goldwasser, S., Safra, S.: Proving hard-core predicates using list decoding. In: Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS 2003), pp. 146–157 (2003)
2. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.P.: RSA and Rabin functions: Certain parts are as hard as the whole. SIAM J. Comput. **17**(2), 194–209 (1988)
3. Blakley, G.R.: Safeguarding cryptographic keys. In: American Federation of Information Processing Societies: National Computer Conference, pp. 313–317 (1979)
4. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM J. Comput. **15**(2), 364–383 (1986)
5. Blum, M., Micali, S.: How to generate cryptographically strong sequence of pseudo-random bits. SIAM J. Comput. **13**(4), 850–864 (1984)
6. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 337–367 (2015)
7. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 509–539 (2016)
8. Chor, B., Gilboa, N.: Computationally private information retrieval. In: Proceedings of the 29th Annual Symposium on Theory of Computing (STOC 1997), pp. 304–313 (1997)
9. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
10. Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 93–122 (2016)
11. Gilboa, N., Ishai, Y.: Distributed point functions and their applications. In: EUROCRYPT 2014. LNCS, vol. 8441, pp. 640–658 (2014)
12. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
13. Goldreich, O., Levin, L.: A hard-core predicate for all one-way functions. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC 1989), pp. 25–32 (1989)
14. Håstad, J., Näslund, M.: The security of all RSA and discrete log bits. J. ACM **51**(2), 187–230 (2004)
15. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS 1997), pp. 364–373 (1997)
16. Morillo, P., Ráfols, C.: The security of all bits using list decoding. In: PKC 2009. LNCS, vol. 5443, pp. 15–33 (2009)
17. O'Donnell, R.: Analysis of Boolean Functions. Cambridge University Press, Cambridge (2014)
18. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)