ON THE CAPACITY OF SECURE DISTRIBUTED FAST FOURIER TRANSFORM

Wei-Ting Chang and Ravi Tandon

Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ, USA
E-mail: {*wchang, tandonr*}@email.arizona.edu

ABSTRACT

Discrete Fourier transform (DFT) is a fundamental building block for various signal processing applications, and speeding up large-scale DFT/FFT using distributed processing is often desirable. In this paper, we consider the problem of distributed DFT computation from untrustworthy servers. In particular, we study the scenario where the goal is to compute the DFT of an s-length signal from N distributed and untrustworthy servers, where any ℓ out of N servers can collude. The signal must be kept information-theoretically secure from any ℓ out of N servers. The goal of this paper is to characterize the minimum communication overhead while maintaining security in an information-theoretic sense. The capacity of the secure distributed FFT is defined as the maximum possible ratio of the desired information (signal length s) and the total information received from all N distributed servers. We present lower and upper bounds on capacity of secure distributed FFT, and show that these bounds match exactly when $N-\ell$ is a power of 2, and the bounds match asymptotically when N becomes large.

Index Terms— Information-theoretic Security, Distributed FFT.

1. INTRODUCTION

Distributed processing for large-scale computations can provide significant speed-ups. However, this is often accompanied with a communication overhead. Also, if the distributed architecture is untrustworthy, sensitive information about the data could be prone to leakage. This type of secure distributed computation problem falls into the category of secure multi-party computation (SMPC). Several previous works on SMPC include [1–4], however, communication overhead was not the main focus of these works. Hence, it is important to devise distributed process algorithms with minimal communication overhead while guaranteeing information-theoretical security.

As one of the fundamental operations in signal and image processing, discrete Fourier transform (DFT) has been studied intensively. The focus of this work is on secure distributed DFT. Since DFT can also be viewed as matrix-vector multiplication, many algorithms for distributed matrix-vector/matrix

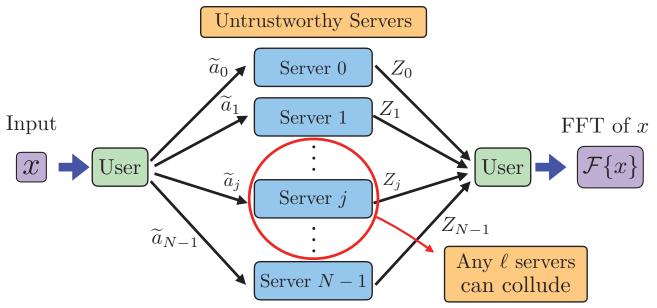

Fig. 1. Secure distributed FFT using untrustworthy servers. The goal is to compute the Fourier transform while keeping servers from learning about x and simultaneously minimizing the communication overhead.

multiplication can be directly applied to compute DFT distributedly. However, such generic strategies do not fully exploit the unique structure of a DFT matrix unlike the existing FFT algorithms for a single machine, such as Cooley-Tukey algorithm [5]. Hence, some works have focused specifically on how to distribute the computation of FFT [6–8]. These works focused on the implementations of the parallel/distributed FFT algorithms. However, these works do not aim to characterize the fundamental limits of the communication cost.

Recently, researchers in information/coding theory communities have made significant progress on understanding the communication cost and the additional speed-ups gained by using codes for distributed matrix multiplication problems. Some of the codes that are efficient for these problems include polynomial codes and MatDot codes [9–11]. These codes introduce redundancy in computations and provide speed-ups by mitigating the stragglers in the distributed system. However, for the similar reason mentioned above, the computation strategies designed along with these codes were not tailored to the computations of FFT. Two recent works, [12, 13], each presented a computing strategy that is designed to distribute the computation of FFT and provides speed-ups by using codes.

However, the aforementioned works do not consider security while designing the algorithms/computing strategies. A few works that do take security into account include [14, 15] where both works used secret sharing scheme along with different encoding strategies to make sure that the computations are secure, and [16] proposed a novel coded computing strat-

This work was supported by NSF Grants CAREER 1651492, and CNS 1715947.

egy that simultaneously provides resiliency against stragglers, security against malicious servers and colluding servers.

Main Contributions: We consider a model where a user wants to compute the DFT of $x \in \mathbb{F}^s$ securely using N untrustworthy servers when any ℓ of them may collude. We leverage the idea of secret sharing [15] and the idea of distributed FFT in [12] for devising a novel distributed FFT scheme which is (a) information-theoretically secure; (b) has a small communication overhead and (c) explicitly utilizes the butterfly structure of FFT. We show that our secure distributed FFT scheme achieves a rate of $2^{\lfloor \log_2(N-\ell) \rfloor}/N$. We also show an upper bound on the rate as $(N-\ell)/N$, which matches with the achievable scheme when $N-\ell$ is a power of 2. Furthermore, the upper and lower bounds match asymptotically when N goes to infinity for a fixed ℓ.

2. SYSTEM MODEL AND PROBLEM FORMULATION

We consider the problem where a user wishes to compute the discrete Fourier transform $\mathbf{X} = \mathcal{F}\{x\} = [X_0 \; X_1 \dots X_{s-1}]^T \in \mathbb{F}^s$ securely for a given input $x = [x_0 \; x_1 \dots x_{s-1}]^T \in \mathbb{F}^s$ using N untrustworthy servers, for some integer s and a sufficiently large field $\mathbb{F}$. Each server is connected to the user through a private link and is assumed to be honest but curious (see Fig. 1). At the same time, any ℓ out of N servers may collude and the user does not know which ℓ servers collude.

To keep the computation secure, the user distributes securely encoded versions of x to N servers. The encoding function is defined as $\boldsymbol{f} = (f_0, f_1, \dots, f_{N-1})$, where f_j is the encoding function for server j. We denote the encoded version of x for server j as $\widetilde{a}_j$, where $\widetilde{a}_j = f_j(x)$ for $j = 0, 1, \dots, N-1$. We denote the subset of colluding servers using the index set $\mathcal{L} \subset [0 : N-1], |\mathcal{L}| = \ell$, and the corresponding encoded vectors the colluding servers received are defined as $\widetilde{a}_{\mathcal{L}} \triangleq (\widetilde{a}_{i_0}, \widetilde{a}_{i_1}, \dots, \widetilde{a}_{i_{\ell-1}})$. For a scheme to be considered secure, $\widetilde{a}_{\mathcal{L}}, \forall \mathcal{L} \subset [0 : N-1], |\mathcal{L}| = \ell$ must not reveal any information about the original x. Hence, a scheme must satisfy the following security constraint,

$$I(x; \widetilde{a}_{\mathcal{L}}) = 0, \forall \mathcal{L} \subset [0 : N-1], |\mathcal{L}| = \ell. \tag{1}$$

The answer from server i is denoted as Z_i for $i = 0, 1, \dots, N-1$. In order to obtain the final result, the user decodes X using all the answers, i.e., $X = d(Z_0, Z_1, \dots, Z_{N-1})$, where $d(.)$ denotes the decoding function. The decodability constraint can be written as $H(X|Z_0, Z_1, \dots, Z_{N-1}) = 0$.

For a scheme which satisfies decodability and security constraints, the rate R is defined as the ratio of the number of desired bits per total downloaded bits, as follows,

$$R = \frac{H(X)}{\sum_{i=0}^{N-1} H(Z_i)}. \tag{2}$$

The capacity $C_{\text{FFT}}(N, \ell)$ of secure distributed FFT is then defined as the supremum of R over all achievable schemes.

3. SECURE DISTRIBUTED FOURIER TRANSFORM

Theorem 1. *For the (N, ℓ) secure Fourier transform problem, in which the input vector x is kept hidden from any ℓ colluding servers while computing the Fourier transform $\mathbf{X} = \mathcal{F}\{x\}$, the capacity is upper and lower bounded as follows,*

$$\frac{2^{\lfloor \log_2(N-\ell) \rfloor}}{N} \leq C_{\text{FFT}}(N, \ell) \leq \frac{N-\ell}{N}. \tag{3}$$

Remark 1. *It is worth noting that the upper bound and lower bound match asymptotically as N goes to infinity for a fixed ℓ, and the bounds also match when $N-\ell$ is a power of 2.*

Remark 2. *In our previous work [15], we obtained an upper bound on the capacity of the secure distributed matrix multiplication problem as $(N-\ell)/N$. Therefore, this bound is also applicable for the secure distributed FFT problem. We also proposed a scheme in [15] to securely compute matrix multiplication AB, where $A \in \mathbb{F}^{r_0 \times r_1}$ and $B \in \mathbb{F}^{r_1 \times r_2}$. It is worth noting that the scheme of [15] works only when r_0 is a multiple of $N-\ell$. However, in the distributed FFT setting, the matrix A is replaced by a vector $x^T \in \mathbb{F}^{1 \times s}$, which does not satisfy the divisibility condition. Therefore, we need to devise a new scheme for secure distributed FFT.*

The proof of the lower bound is provided in Section 3.1. Before presenting the proof, we present an illustrating example and show the intuition behind the proposed scheme. A standard FFT breaks the computation of the original DFT of an s-length signal into $\log s$ stages. In our scheme, the user breaks the signal into $m = 2^{\lfloor \log_2(N-\ell) \rfloor}$ parts and securely outsources the first $\log \frac{s}{m}$ stages of FFT to N untrustworthy servers. After receiving the responses from servers, the user then decodes the intermediate values, and computes the rest of $\log m$ stages.

Example 1. $(N = 5, \ell = 2)$ *Consider a secure DFT problem with 5 servers and any 2 servers may collude. The input vector is $x = [x_0 \; x_1 \; x_2 \; x_3]^T$ of length $s = 4$. The goal of the user is to compute the DFT of x. Using standard DFT algorithm, the following needs to be computed,*

$$X = \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -\sqrt{-1} & -1 & \sqrt{-1} \\ 1 & -1 & 1 & -1 \\ 1 & \sqrt{-1} & -1 & -\sqrt{-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{4}$$

Individual elements of X can be seen as linear combinations of the elements of x, i.e.,

$$X_i = \sum_{n=0}^{3} x_n (-\sqrt{-1})^{in} \tag{5}$$

$$= \left[(-\sqrt{-1})^{0i} x_0 + (-\sqrt{-1})^{2i} x_2\right] + (-\sqrt{-1})^{i} \left[(-\sqrt{-1})^{0i} x_1 + (-\sqrt{-1})^{2i} x_3\right]. \tag{6}$$

It can be seen that by splitting the summation into two parts according to the indices of the elements in x, the computations inside the brackets are essentially two two-point DFT with different inputs. This property has been exploited by the

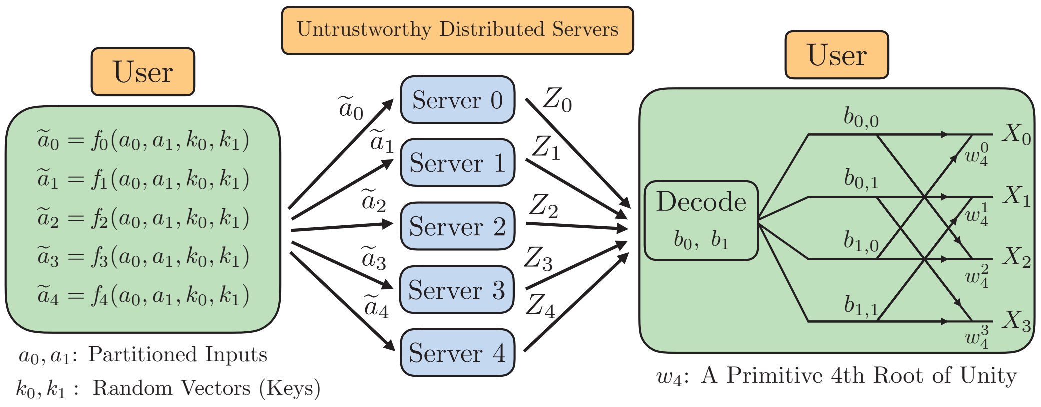
Untrustworthy Distributed Servers
User
$\widetilde{a}_0 = f_0(a_0, a_1, k_0, k_1)$
$\widetilde{a}_1 = f_1(a_0, a_1, k_0, k_1)$
$\widetilde{a}_2 = f_2(a_0, a_1, k_0, k_1)$
$\widetilde{a}_3 = f_3(a_0, a_1, k_0, k_1)$
$\widetilde{a}_4 = f_4(a_0, a_1, k_0, k_1)$
a_0, a_1: Partitioned Inputs
k_0, k_1 : Random Vectors (Keys)
Server 0
Server 1
Server 2
Server 3
Server 4
Z_0 Z_1 Z_2 Z_3 Z_4
User
Decode
$b_0,\ b_1$
$b_{0,0}$ $b_{0,1}$ $b_{1,0}$ $b_{1,1}$
w_4^0 w_4^1 w_4^2 w_4^3
X_0 X_1 X_2 X_3
w_4: A Primitive 4th Root of Unity

Fig. 2. A four-point DFT with $N = 5$, $\ell = 2$ and $s = 4$. The user sends the encoded inputs, which are kept private by the keys, to servers. After servers compute and return the answers, the user decodes b_0 and b_1, and computes the final result using b_0 and b_1.

Cooley-Tukey fast Fourier transform algorithm [5] for the single machine setting and by authors of [12] for the insecure distributed setting. We exploit this property in a similar fashion. In this example, the computation is broken into $\log s = 2$ stages. Our scheme consists of three phases. During Phase 1, the user divides x into $m = 2^{\lfloor \log_2(N-\ell) \rfloor} = 2$ parts as follows,

$$a_0 = \begin{bmatrix} x_0 \\ x_2 \end{bmatrix}, \quad a_1 = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix}. \tag{7}$$

The user then encodes $\widetilde{a}_0$ and $\widetilde{a}_1$ for server i as follows,

$$\widetilde{a}_i = a_0 + (i+1)a_1 + (i+1)^2 k_0 + (i+1)^3 k_1, \tag{8}$$

where k_0 and k_1 are random vectors of same length as a_0 and a_1 whose entries are drawn i.i.d. from the uniform distribution for all $i = 0, 1, \ldots, 4$. By adding these two random vectors, the two colluding servers will not be able to obtain a clean version of any a_i. Servers return the results of the first $\log \frac{s}{m} = 1$ stage after computing $\mathcal{F}\{\widetilde{a}_i\} = W_2 \widetilde{a}_i$, i.e.,

$$\begin{aligned}
Z_0 &= W_2 a_0 + W_2 a_1 + W_2 k_0 + W_2 k_1 \\
Z_1 &= W_2 a_0 + 2W_2 a_1 + 2^2 W_2 k_0 + 2^3 W_2 k_1 \\
Z_2 &= W_2 a_0 + 3W_2 a_1 + 3^2 W_2 k_0 + 3^3 W_2 k_1 \\
Z_3 &= W_2 a_0 + 4W_2 a_1 + 4^2 W_2 k_0 + 4^3 W_2 k_1 \\
Z_4 &= W_2 a_0 + 5W_2 a_1 + 5^2 W_2 k_0 + 5^3 W_2 k_1,
\end{aligned} \tag{9}$$

where W_2 is the 2-point DFT matrix. The answers from servers can be seen as the evaluations of a degree 4 polynomial at 5 distinct points.

In Phase 2, the user takes the 5 linearly independent equations and decodes $W_2 a_0$ and $W_2 a_1$,

$$b_i = \begin{bmatrix} b_{i,0} \\ b_{i,1} \end{bmatrix} \triangleq W_2 a_i = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} a_{i,0} \\ a_{i,1} \end{bmatrix}. \tag{10}$$

Here, we have defined $b_i \triangleq W_2 a_i$ as the 2-point DFT of $a_i, i = 0, 1$.

In Phase 3, the user follows the combining rules of the FFT algorithm and computes the last $\log m = 1$ stage, which can be visualized with a butterfly diagram shown in Fig. 2, i.e.,

$$\begin{aligned}
X_0 &= b_{0,0} + b_{1,0} \\
X_1 &= b_{0,1} - \sqrt{-1} b_{1,1} \\
X_2 &= b_{0,0} - b_{1,0} \\
X_3 &= b_{0,1} + \sqrt{-1} b_{1,1}
\end{aligned} \tag{11}$$

3.1. Achievable Scheme and Proof of Theorem 1

We now present the general scheme and describe the details of each phase and their computational complexity.

Phase 1 (Secure Encoding at the User): The user first splits the input vector x into m disjoint vectors $a_i, \forall\ i = 0, 1, \ldots, m-1$, where $m = 2^{\lfloor \log_2(N-\ell) \rfloor}$. The elements of a_i's are given as,

$$a_{i,t} = x_{i+tm}, \quad t = 0, 1, \ldots, \frac{s}{m} - 1. \tag{12}$$

In order to provide security to a_i's against any ℓ colluding servers, the user generates ℓ random vectors k_n's whose elements are drawn i.i.d. from the uniform distribution and encodes a_i's as follows,

$$\widetilde{a}_j = \sum_{i=0}^{m-1} a_i y_j^i + \sum_{n=0}^{\ell-1} k_n y_j^{n+m}, \tag{13}$$

where y_j is a distinct non-zero element in $\mathbb{F}$ assigned to server j. Each server j computes an $\frac{s}{m}$-point FFT using their received $\widetilde{a}_j$ and returns the following polynomial,

$$h(y) = \sum_{i=0}^{m-1} W_{\frac{s}{m}} a_i y^i + \sum_{n=0}^{\ell-1} W_{\frac{s}{m}} k_n y^{n+m}, \tag{14}$$

at $y = y_j, j = 0, 1, \ldots, N-1$, where we denote $h(y_j)$ as Z_j. Servers are assumed to be working at the same speed and in parallel. The complexity of this phase is, thus, the complexity of a single machine computing an $\frac{s}{m}$-point FFT. It is well-known that the complexity of computing an $\frac{s}{m}$-point FFT is $O(\frac{s}{m} \log \frac{s}{m})$.

We next show that the proposed scheme is information-theoretically secure and that the security constraint (1) is satisfied. We start with the following sequence of inequalities:

$$I(x; \widetilde{a}_{\mathcal{L}}) = I(x; \widetilde{a}_{i_0}, \ldots, \widetilde{a}_{i_{\ell-1}})$$

$$
\begin{aligned}
&= H(\widetilde{a}_{i_0}, \ldots, \widetilde{a}_{i_{\ell-1}}) - H(\widetilde{a}_{i_0}, \ldots, \widetilde{a}_{i_{\ell-1}}|x) \\
&\overset{(a)}{=} H(\widetilde{a}_{i_0}, \ldots, \widetilde{a}_{i_{\ell-1}}) - H(k_0, \ldots, k_{\ell-1}) \\
&\overset{(b)}{=} H(\widetilde{a}_{i_0}, \ldots, \widetilde{a}_{i_{\ell-1}}) - \ell \frac{s}{m} \log |\mathbb{F}| \\
&\overset{(c)}{\leq} H(\widetilde{a}_{i_0}) + \cdots + H(\widetilde{a}_{i_{\ell-1}}) - \ell \frac{s}{m} \log |\mathbb{F}| \\
&\overset{(d)}{=} \ell \frac{s}{m} \log |\mathbb{F}| - \ell \frac{s}{m} \log |\mathbb{F}| = 0, \qquad (15)
\end{aligned}
$$

where (a) follows from the fact that all encoded vectors $\widetilde{a}_{\mathcal{L}}$ are functions of x and random vectors $k_i, \forall\, i = 0, 1, \ldots, \ell-1$, (b) is due to the fact that an uniformly distributed random variable in the field $\mathbb{F}$ has entropy $\log |\mathbb{F}|$ and there are $\frac{s}{m}$ elements in each one of the ℓ random vectors, (c) follows from upper bounding the joint entropy by using the sum of individual entropies, and (d) follows from argument similar to (b). Since mutual information is non-negative, we conclude that the scheme is information-theoretically secure.

Phase 2 (Distributed Computation at Servers): Each answer Z_j can be viewed as a polynomial evaluated at point $y = y_j$ for all j whose degree is at most $m + \ell$. Since $m + \ell$ is always less or equal to N, the user can always solve for the coefficients of the polynomial by polynomial interpolation using the N evaluations to decode the Fourier transforms of a_i's. We denote the Fourier transforms of a_i as b_i, whose elements are,

$$b_{i,t} = \sum_{n=0}^{\frac{s}{m}-1} a_{i,n} w_s^{tmn}, \forall \quad t = 0, 1, \ldots, \frac{s}{m} - 1, \qquad (16)$$

where w_s denotes the primitive sth root of unity. The complexity of interpolating a polynomial with degree at most $m + \ell$ is known to be at most $O((m + \ell) \log^2(m + \ell) \log\log(m + \ell))$. After multiplying the length of a_i, the complexity becomes $O(\frac{s}{m}(m + \ell) \log^2(m + \ell) \log\log(m + \ell))$.

Phase 3 (Decoding at the User): The user then computes the final result by using the elements of all b_i according to FFT algorithm as follows,

$$
\begin{aligned}
X_i &= \sum_{t=0}^{m-1} \left(\sum_{n=0}^{\frac{s}{m}-1} a_{t,n} w_s^{imn} \right) w_s^{it} \qquad (17) \\
&= \sum_{t=0}^{m-1} b_{t,\mathrm{mod}(i,\frac{s}{m})} w_s^{it}, \qquad (18)
\end{aligned}
$$

for all $i = 0, 1, \ldots, s-1$. To remove the modulo in (18), we substitute i with $p + q\frac{s}{m}$. After rearranging, we have,

$$X_{p+q\frac{s}{m}} = \sum_{t=0}^{m-1} \left(b_{t,p} w_s^{pt} \right) w_s^{qt\frac{s}{m}}, \qquad (19)$$

for $p = 0, 1, \ldots, \frac{s}{m} - 1$ and $q = 0, 1, \ldots, m-1$. By expanding (19) for all $p + q\frac{s}{m}$, it can be seen that equation (19) is, in fact, the Fourier transform of $\frac{s}{m}$ vectors of length m, and the tth element of the pth vector equals to $b_{t,p} w_s^{pt}$.

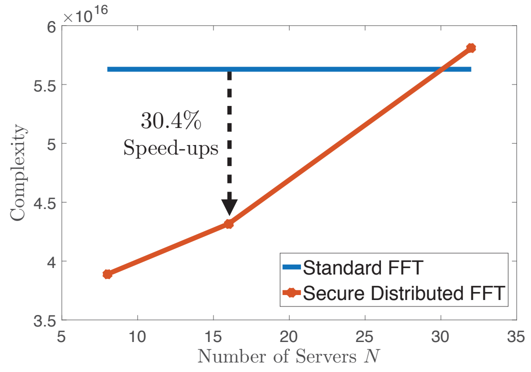

Fig. 3. A comparison between the secure distributed FFT and standard FFT when $s = 2^{50}$, $\ell = 3$ and $N = 8, 16$ and 32.

The complexity of the Fourier transform of a length m vector is known to be upper bounded by $O(m \log m \log\log m)$ by using Bluestein's FFT algorithm [17] and fast polynomial multiplication [18]. By multiplying the number of vectors, the complexity is $O(s \log m \log\log m)$. The overall complexity is, hence, the sum of complexities of three phases.

We next compare the secure distributed FFT to standard FFT (see Fig. 3). We let $s = 2^{50}$ and $\ell = 3$. The curve shows the complexity of the secure distributed FFT at $N = 8, 16$ and 32, and the horizontal line represents the complexity of standard FFT computed at a single machine. Clearly, when $N = 8$ and 16, the secure distributed FFT provides significant speed-ups. However, as N increases, the number of partitions m also increases. Hence, the decoding of Phase 2 becomes the bottleneck, and the overall complexity increases. When $N = 32$ or larger, it is no longer efficient to compute FFT distributedly using the proposed scheme as the decoding complexity of Phase 2 becomes too large, and the overall complexity exceeds the complexity of standard FFT.

The communication rate of the scheme can be computed by counting the number of useful terms received at the user, i.e., number of b_i's, divides by the total number of downloaded terms, i.e., number of Z_j's. Since there are m b_i's and N Z_j's, the communication rate is $R_{\text{FFT}}(N, \ell) = m/N = 2^{\lfloor \log_2(N-\ell) \rfloor}/N$. Using the achievable rate $R_{\text{FFT}}(N, \ell)$ as a lower bound on the capacity, i.e., $R_{\text{FFT}}(N, \ell) \leq C_{\text{FFT}}(N, \ell)$, along with Remark 2 and [15], i.e., $C_{\text{FFT}}(N, \ell) \leq \frac{N-\ell}{N}$, we obtain the expression of Theorem 1. This completes the proof.

4. CONCLUSION

In this paper, we studied the problem of secure distributed fast Fourier transform of a long input vector x, where x must be kept private from the servers while any ℓ of them may collude during the computations. Our proposed scheme is a combination of the secret sharing scheme and the Cooley-Tukey FFT algorithm. The capacity is lower bounded by the achievable rate of $2^{\lfloor \log_2(N-\ell) \rfloor}/N$ and is upper bounded by $(N - \ell)/N$. We show that the upper and lower bounds match asymptotically as N becomes large. In addition, these bounds match exactly when $N - \ell$ is a power of 2.

5. REFERENCES

[1] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science*, Nov. 1982, pp. 160–164.

[2] David Chaum, Ivan B. Damgård, and Jeroen van de Graaf, "Multiparty Computations Ensuring Privacy of Each Party's Input and Correctness of the Result," in *Advances in Cryptology*, Berlin, Heidelberg, 1988, pp. 87–119, Springer Berlin Heidelberg.

[3] Divya G. Nair, V. P. Binu, and G. Santhosh Kumar, "An Improved E-voting scheme using Secret Sharing based Secure Multi-party Computation," *CoRR*, vol. abs/1502.07469, 2015.

[4] Patrick Ah-Fat and Michael Huth, "Secure Multi-party Computation: Information Flow of Outputs and Game Theory," in *Principles of Security and Trust*, Berlin, Heidelberg, 2017, pp. 71–92, Springer Berlin Heidelberg.

[5] James W Cooley and John W Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[6] Matteo Frigo and Steven G Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[7] Michael Pippig, "PFFT: An extension of FFTW to massively parallel architectures," *SIAM Journal on Scientific Computing*, vol. 35, no. 3, pp. C213–C236, 2013.

[8] Amir Gholami, Judith Hill, Dhairya Malhotra, and George Biros, "AccFFT: A library for distributed-memory FFT on CPU and GPU architectures," *CoRR*, vol. abs/1506.07933, 2015. [Online]. Available:http://arxiv.org/abs/1506.07933.

[9] Qian Yu, Mohammad Ali Maddah-Ali, and Amir Salman Avestimehr, "Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication," *CoRR*, vol. abs/1705.10464, 2017. [Online]. Available:http://arxiv.org/abs/1705.10464.

[10] Qian Yu, Mohammad Ali Maddah-Ali, and Amir Salman Avestimehr, "Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding," *CoRR*, vol. abs/1801.07487, 2018. [Online]. Available:http://arxiv.org/abs/1801.07487.

[11] Sanghamitra Dutta, Mohammad Fahim, Farzin Haddadpour, Haewon Jeong, Viveck R. Cadambe, and Pulkit Grover, "On the Optimal Recovery Threshold of Coded Matrix Multiplication," *CoRR*, vol. abs/1801.10292, 2018. [Online]. Available:http://arxiv.org/abs/1801.10292.

[12] Qian Yu, Mohammad Ali Maddah-Ali, and Amir Salman Avestimehr, "Coded Fourier Transform," *CoRR*, vol. abs/1710.06471, 2017. [Online]. Available:http://arxiv.org/abs/1710.06471.

[13] Haewon Jeong, Tze Meng Low, and Pulkit Grover, "Coded FFT and Its Communication Overhead," *CoRR*, vol. abs/1805.09891, 2018. [Online]. Available:http://arxiv.org/abs/1805.09891.

[14] Rawad Bitar, Parimal Parag, and Salim El Rouayheb, "Minimizing Latency for Secure Distributed Computing," *CoRR*, vol. abs/1703.01504, 2017. [Online]. Available:http://arxiv.org/abs/1703.01504.

[15] Wei-Ting Chang and Ravi Tandon, "On the Capacity of Secure Distributed Matrix Multiplication," *CoRR*, vol. abs/1806.00469, 2018. [Online]. Available:http://arxiv.org/abs/1806.00469.

[16] Qian Yu, Netanel Raviv, Jinhyun So, and Amir Salman Avestimehr, "Lagrange Coded Computing: Optimal Design for Resiliency, Security and Privacy," *CoRR*, vol. abs/1806.00939, 2018. [Online]. Available:http://arxiv.org/abs/1806.00939.

[17] L. Bluestein, "A linear filtering approach to the computation of discrete Fourier transform," *IEEE Transactions on Audio and Electroacoustics*, vol. 18, no. 4, pp. 451–455, December 1970.

[18] David G. Cantor and Erich Kaltofen, "On Fast Multiplication of Polynomials Over Arbitrary Algebras," *Acta Informatica*, vol. 28, pp. 693–701, 1991.