IEICE TRANS. FUNDAMENTALS, VOL.E103–A, NO.10 OCTOBER 2020

INVITED PAPER *Special Section on Discrete Mathematics and Its Applications*

Recent Advances in Practical Secure Multi-Party Computation*

Satsuya OHATA[†a)], ***Nonmember***

SUMMARY Secure multi-party computation (MPC) allows a set of parties to compute a function jointly while keeping their inputs private. MPC has been actively studied, and there are many research results both in the theoretical and practical research fields. In this paper, we introduce the basic matters on MPC and show recent practical advances. We first explain the settings, security notions, and cryptographic building blocks of MPC. Then, we show and discuss current situations on higher-level secure protocols, privacy-preserving data analysis, and frameworks/compilers for implementing MPC applications with low-cost.

key words: *secure multi-party computation, privacy-preserving data analysis*

1. Introduction

Secure multi-party computation (MPC) [1], [2] allows a set of parties to compute a function f jointly while keeping their inputs private. More precisely, the $N(\geq 2)$ parties, each holding private input x_i for $i \in [1, N]$, are able to compute the output $f(x_1, \cdots, x_N)$ without revealing their private inputs x_i. There is much progress in the research on MPC, and its performance is dramatically improved. Moreover, many applications on MPC have been proposed.

In this paper, we explain the recent situations in practical MPC and its applications. More concretely, we denote the following topics.

- We introduce the settings and security notions we usually consider in the MPC research.
- We show some building blocks (homomorphic encryption, garbled circuit, secret sharing, and trusted hardware) for efficient MPC.
- We show recent advances in secure higher-level protocols (e.g., equality check, less-than comparison), applications (machine learning, string analysis, the nearest neighbor search), and implementation-related efforts on MPC.

In Sect. 2, we introduce the settings and security notions on MPC. Then, we show some building blocks (homomorphic encryption, garbled circuit, secret sharing, and trusted hardware) for MPC. As a concrete example, we take up secret sharing-based two-party computation and explain how to securely compute arithmetic and boolean gates. We also discuss pros and cons between building blocks. In Sect. 3, we explain the recent advances in practical MPC. More concretely, we show how to construct higher-level secure protocols, recent situations on privacy-preserving data analysis including secure machine learning, and the efforts for implementing MPC with low-cost. Section 4 is the conclusion of this paper.

2. An Overview of Secure Multi-Party Computation: Settings, Security Notions, and Building Blocks

In this section, we show the technical overview of secure multi-party computation.

2.1 Settings

There are many different settings that the previous works have been considering. We here explain these settings.

(1) Client-Server Settings

We consider the N clients that have their secrets. We call these clients as input parties. We consider the situation that these N clients execute MPC to communicate with each other. We call the parties that execute MPC as computing parties. In the above case, input parties and computing parties are the same. However, not necessarily, every input party has rich computing resources. Moreover, we cannot execute MPC using the secrets of arbitrary N clients if the MPC protocol is optimized for fixed $M(\neq N)$. In these situations, we usually consider the client-server (or outsourcing) setting; that is, input parties are not the same as computing parties. In this setting, N clients send their (processed) secrets to the M servers (=computing parties), and then M servers execute MPC. After that, servers return their execution results to the clients.

(2) The Number of Parties

M varies in building blocks. In secret sharing-based MPC, for example, we can set arbitral $M \geq 2$. There are many efficient and optimized MPC schemes for fixed M. Note that the collusion resistance depends on the building blocks. For example, (see Sect. 2.3.1 for more details) when we consider three-party computation ($M = 3$) using 2-out-of-3 secret sharing schemes, the confidentiality is broken if two computing parties collude. Besides, the property of computing

Manuscript received October 22, 2019.
Manuscript revised February 28, 2020.

[†]The author is with Digital Garage, Inc., Tokyo, 150-0042 Japan.

*This work was done when the author was working at National Institute of Advanced Industrial Science and Technology (AIST) and supported by JST CREST JPMJCR 19F6.

a) E-mail: satsuya-ohata@dglab.com

DOI: 10.1587/transfun.2019DMI0001

parties also affects on security and efficiency of MPC. The setting that all computing parties do not deviate from the protocol is called "semi-honest" settings (for more details, see Sect. 2.2). As stronger security notions, we usually consider "malicious" settings. In this security notions, we consider the computing parties that deviate from the protocol. In this setting, the rate of malicious parties affects the efficiency of MPC. In recent situations, there are many practical schemes in semi-honest or honest-majority settings.

(3) Communication Environments

MPC requires communications between computing parties. Therefore, the performance of MPC relies not only on the machine resources but on communication environments. In many previous research results, the performance of MPC has been evaluated over local area networks (LAN). In MPC over LAN environments, the performance bottleneck is case by case. In practice, however, we have many situations that cannot assume LAN environments. In secret sharing-based MPC with the client-server setting (in Sect. 2.3.1), for example, MPC over LAN environments means all computing parties work under the same administrator. It is difficult to say the secrets are appropriately distributed between non-colluding servers in this situation. Based on such a situation, recently, we can see research results on MPC over wide area networks (WAN). In this setting, the performance bottleneck is not computation but communication in most cases because of its poor bandwidth and large latency. Which is more important depends on the circuits we compute and the data size we treat.

2.2 Security Notions

In the security evaluation of cryptographic research, we define the security goal and prove that the proposed schemes satisfy them via security reduction. The security against semi-honest (or, honest but curious) adversaries [3] is the basic and minimum security notion in MPC research. In this security notion, the adversaries try to obtain information from their views but do not deviate from the protocol. More concretely, security against the semi-honest adversary is defined as follows:

Definition: Security against Semi-Honest Adversary Let λ be the number of parties. Let $f : (\{0,1\}^*)^\lambda \to (\{0,1\}^*)^\lambda$ be a probabilistic λ-ary functionality and $f_i(\vec{x})$ denotes the i-th element of $f(\vec{x})$ for $\vec{x} = (x_1, x_2, \cdots, x_\lambda) \in (\{0,1\}^*)^\lambda$ and $i \in \{1, 2, \cdots, \lambda\}$; $f(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \cdots, f_\lambda(\vec{x}))$. Let Π be a λ-party protocol to compute the functionality f. The view of party P_i for $i \in \{1, 2, \cdots, \lambda\}$ during an execution of Π on input $\vec{x} = (x_1, x_2, \cdots, x_\lambda) \in (\{0,1\}^*)^\lambda$ where $|x_1| = |x_2| = \cdots = |x_\lambda|$, denoted by $\textsc{View}_i^\Pi(\vec{x})$, consists of $(x_i, r_i, m_{i,1}, \ldots, m_{i,t})$, where x_i represents P_i's input, r_i represents its internal random coins, and $m_{i,j}$ represents the j-th message that P_i has received. The output of all parties after an execution of Π on input $\vec{x}$ is denoted as $\textsc{Output}^\Pi(\vec{x})$. Then for each party P_i, we say that Π *privately computes* f in the presence of semi-honest corrupted party P_i if there exists a probabilistic polynomial-time algorithm $\mathcal{S}$ such that

$$\{(\mathcal{S}(i, x_i, f_i(\vec{x})), f(\vec{x}))\} \equiv \{(\textsc{View}_i^\Pi(\vec{x}), \textsc{Output}^\Pi(\vec{x}))\}$$

where the symbol $\equiv$ means that the two probability distributions are statistically indistinguishable.

As described in [3], the composition theorem for the semi-honest model holds; that is, any protocol is privately computed as long as its subroutines are privately computed. We also usually consider the stronger security model; security against malicious adversaries. In this notion, the adversaries may deviate from the protocol, and the composition theorem does not hold; that is, the higher-level protocols based on maliciously secure building blocks do not always satisfy malicious security.

2.3 Building Blocks

There are some building blocks for MPC. Here, we mainly explain the MPC based on secret sharing as an example and show a short overview for others.

2.3.1 Secret Sharing

By using secret sharing (SS) (e.g., [4]), we can compute arithmetic or boolean gates over shares (e.g., [2], [5]–[8]). By combining the above gates, we can compute arbitrary circuits over shares. In this setting, both inputs and outputs are shares. Here, as an example, we explain how to securely compute arithmetic or boolean gates in two-party settings via Beaver triple-based secure two-party computation with preprocessing [2], [9].

(1) Arithmetic Gates

We explain how to compute arithmetic $\mathsf{ADD}/\mathsf{MULT}$ gates on $(2,2)$-additive SS. We use the standard the $(2,2)$-additive SS scheme, defined by

- $\mathsf{Share}(x)$: randomly choose $r \in \mathbb{Z}_{2^n}$ and let $[\![x]\!]_1^{\mathsf{A}} = r$ and $[\![x]\!]_2^{\mathsf{A}} = x - r$.
- $\mathsf{Reconst}([\![x]\!]_1^{\mathsf{A}}, [\![x]\!]_2^{\mathsf{A}})$: output $[\![x]\!]_1^{\mathsf{A}} + [\![x]\!]_2^{\mathsf{A}}$.

We can compute fundamental operations: $\mathsf{ADD}(x, y) := x + y$ and $\mathsf{MULT}(x, y) := xy$ as follows:

- $[\![z]\!] \leftarrow \mathsf{ADD}([\![x]\!], [\![y]\!])$ can be done locally by just adding each party's share on x and on y.
- $[\![w]\!] \leftarrow \mathsf{MULT}([\![x]\!], [\![y]\!])$ can be done in various approaches. Here we explain the method based on Beaver triples (BT) [9]. Such a triple consists of $[\![bt]\!]_1 = ([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ and $[\![bt]\!]_2 = ([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ such that $([\![a]\!]_1 + [\![a]\!]_2)([\![b]\!]_1 + [\![b]\!]_2) = ([\![c]\!]_1 + [\![c]\!]_2)$. Hereafter, a, b, and c denote $[\![a]\!]_1 + [\![a]\!]_2$, $[\![b]\!]_1 + [\![b]\!]_2$, and $[\![c]\!]_1 + [\![c]\!]_2$, respectively. We can compute these BT (that are used as auxiliary inputs of the secure multiplication protocol) in offline phase; that is, we can generate BT in advance since it is independent of shares. In this protocol, each

i-th party P_i ($i \in \{1, 2\}$) can compute the multiplication share $[\![z]\!]_i = [\![xy]\!]_i$ as follows:

1. P_i first compute $([\![x]\!]_i - [\![a]\!]_i)$ and $([\![y]\!]_i - [\![b]\!]_i)$.
2. P_1/P_2 sends them to P_2/P_1, respectively.
3. P_i reconstruct $x' = x - a$ and $y' = y - b$.
4. P_1 computes $[\![z]\!]_1 = x'y' + x'[\![b]\!]_1 + y'[\![a]\!]_1 + [\![c]\!]_1$ and P_2 computes $[\![z]\!]_2 = x'[\![b]\!]_2 + y'[\![a]\!]_2 + [\![c]\!]_2$.

$[\![z]\!]_1$ and $[\![z]\!]_2$ calculated as the above procedures are valid shares of xy; that is, $\mathsf{Reconst}([\![z]\!]_1, [\![z]\!]_2) = xy$.

We abuse notations and write the ADD and MULT protocols simply as $[\![x]\!] + [\![y]\!]$ and $[\![x]\!] \cdot [\![y]\!]$, respectively. Note that similarly to the ADD protocol, we can also locally compute multiplication by constant c, denoted by $c \cdot [\![x]\!]$.

(2) Boolean Gates

In boolean XOR/AND gates, we use the standard $(2, 2)$-SS scheme, defined by

- $\mathsf{Share}(x)$: randomly choose $r \in \mathbb{Z}_2$ and let $[\![x]\!]^{\mathsf{B}}_1 = r$ and $[\![x]\!]^{\mathsf{B}}_2 = x \oplus r$.
- $\mathsf{Reconst}([\![x]\!]^{\mathsf{B}}_1, [\![x]\!]^{\mathsf{B}}_2)$: output $[\![x]\!]^{\mathsf{B}}_1 \oplus [\![x]\!]^{\mathsf{B}}_2$.

By converting $+$ and $-$ to $\oplus$ in arithmetic ADD and MULT protocol, we can obtain XOR and AND protocol, respectively. We can construct NOT and OR protocols from the properties of these gates as follows:

- $\mathsf{NOT}([\![x]\!]^{\mathsf{B}}_1, [\![x]\!]^{\mathsf{B}}_2)$: P_1 and P_2 output $\neg[\![x]\!]^{\mathsf{B}}_1$ and $[\![x]\!]^{\mathsf{B}}_2$, respectively.
- $\mathsf{OR}([\![x]\!], [\![y]\!])$: From a basic property of OR gate, we can construct OR by combining NOT and AND; that is, $\mathsf{OR}([\![x]\!], [\![y]\!]) = \neg\mathsf{AND}(\neg[\![x]\!], \neg[\![y]\!])$.

When we construct higher-level secure protocols, we use these two types of shares and their conversion protocols (in Sect. 3.1).

(3) Generation of Beaver Triples

We can compute BT in an offline phase since it is independent of the inputs (x, y). We usually generate BT using HE or Oblivious transfer (OT) between two computing parties in advance [10]. To improve the time we need for this generation, the model that the third party generates BT without using HE/OT and distributes them to the computing parties has been proposed [11], [12].

2.3.2 Homomorphic Encryption

Homomorphic encryption (HE) is an encryption scheme that can execute some operations (addition, multiplication, or both) over ciphertexts. We have some types of HE; additive HE (e.g., [13]) that can compute addition, multiplicative HE (e.g., [14]) that can compute multiplication, somewhat (or leveled) HE (e.g., [15], [16]) that can compute addition and the restricted number of multiplication, and fully HE (FHE) (e.g., [17]) that can compute addition and multiplication. There are also many papers on toolkit (e.g., [18]) and applications (e.g., [19]–[23]) based on HE. In MPC with client-server settings based on FHE, there is no communication between computing parties, and this is a definite advantage compared with MPC based on other building blocks. However, it tends to require high computation costs. Moreover, we need to use the same key for encrypting plaintexts, and this is a worrisome problem in multi-client settings. If we adopt multi-key FHE (e.g., [24]), we do not need to take care of this problem. However, in multi-key FHE, we need to execute MPC for decrypting ciphertexts. Besides, practical performance is not enough and far from practical use.

2.3.3 Garbled Circuit

In garbled circuits (GC) (e.g., [1], [25]), we can construct the garbled circuit and securely evaluate it as follows:

1. The party called "garbler" generates a garbled circuit; that is, garbler prepares some ciphertexts for one AND gate and assign keys for all input patterns. By repeating this procedure, the garbler generates the garbled circuit $C(x, \cdot)$. Here, x is the input of the garbler, and $C(x, \cdot)$ does not leak the information of x. Then, the garbler sends the garbled circuit to the other party called "evaluator".
2. The evaluator tries to decrypt all the ciphertexts using the key corresponding to the input of the evaluator y. The evaluator can decrypt only one ciphertext and obtain a new key for the next input. By repeating this procedure, evaluator eventually obtains the result of function evaluation $C(x, y)$ without knowing x.

When we compute a large circuit, the size of data transfer becomes large since we need to send more than one ciphertext (= 128 bit, for example) per one AND gate [26], [27]. On the other hand, we can compute an arbitrary circuit with constant small communication rounds. This means GC is suitable for the computation of non-linear functions and the computation over large-latency networks [28], [29]. We have many application papers using GC (e.g., [30], [31]).

2.3.4 Trusted Hardware

We can execute MPC using trusted execution environments (TEE). In Intel SGX [32], [33], a well-known realization of TEE, we set the key in CPU as a root of trust and construct isolated spaces (enclaves) on the RAM using the key. Even the operating system cannot refer to the data in the enclaves. Therefore, we can execute MPC via TEE [34], [35]; that is, we compute functions in enclaves. Although there are some limitations (e.g., the size of enclaves), TEE-based MPC works faster than all the other approaches. On the other hand, there are many reports on side-channel attacks for TEE (e.g., [36], [37]). Moreover, to believe the security of TEE (and TEE-based MPC), we need to trust the correctness of its design and manufacturing processes. TEE is a relatively new technology, and there are some discussions on its security including formalization (e.g., [38]).

2.4 Discussion

In Sect. 2.3, we show some building blocks for MPC. If we focus on execution speed. TEE-based MPC is faster than others. CYBERNETICA (a technology company in Estonia) releases a new MPC system based on TEE called "Sharemind HI" (previously, it has been proposed "Sharemind MPC", a SS-based MPC system). As we previously denoted, however, whether the TEE-based MPC is secure or not is related to the trust. Moreover, it is not easy to verify the security of TEE by third parties since it is a hardware-based system.

Other approaches (MPC based on HE/GC/SS) have pros and cons on their performances. Although FHE-based MPC does not need communications between computing parties, its performance is far from practice in many cases since the costs for data transfer and computation is extremely high. In other MPC protocols, whole performances depend on the communication costs in many cases. GC-based MPC is suitable for large-latency environments since it can execute via small constant-round communications. However, the size of data transfer is large. SS-based MPC has opposite properties to GC-based MPC. Recently, to overcome the disadvantage of SS-based MPC (that is, it requires many communication rounds than other approaches), the mutual transformations (or mixed protocols) between SS-based MPC and GC-based MPC (e.g., [10], [39]) have been proposed.

3. Recent Advances of Secure Multi-Party Computation

In this section, we show recent advances in higher-level secure protocols, privacy-preserving data analysis, and frameworks/compilers for implementing MPC applications with low-cost. We mainly show the research results on SS-based MPC, and some of them are based on GC or HE. We do not mention the details in each section, parallelizing the computations and communications are the major premises to improve the performance in SS-based MPC.

3.1 Higher-Level Secure Protocols

By combining the protocols computing arithmetic/boolean gates in Sect. 2.3.1, we can construct higher-level secure protocols. Here we explain the basic strategies for constructing these protocols efficiently.

(1) Equality Check

An equality check protocol $\mathsf{Equality}(\llbracket x \rrbracket^{\mathsf{A}}, \llbracket y \rrbracket^{\mathsf{A}})$ (e.g., [10], [40], [41]) outputs $\llbracket z \rrbracket^{\mathsf{B}}$, where $z = 1$ iff $x = y$. A basic strategy for constructing Equality is as follows: we first compute $t = x - y$ and then check all bits of t are 0 or not. If all the bits of t are 0, it means $t = x - y = 0$. We need $O(\log n)$ communication rounds for the above procedure when we use OR with a tree structure. Although the computation and memory costs increase, we can obtain more round-efficient Equality based on multi-fan-in gates [42][†].

(2) Less-Than Comparison

A less-than comparison protocol ($\mathsf{Comparison}(\llbracket x \rrbracket^{\mathsf{A}}, \llbracket y \rrbracket^{\mathsf{A}})$) outputs $\llbracket z \rrbracket^{\mathsf{B}}$, where $z = 1$ iff the condition $x < y$ holds (e.g., [10], [40], [41]). A basic strategy for constructing Comparison is to check the sign of $x - y$. More precisely, we execute the following procedures:

1. We compute the MSB of $x, y, x - y$ over shares and set them as $\llbracket x' \rrbracket^{\mathsf{B}}, \llbracket y' \rrbracket^{\mathsf{B}}, \llbracket d' \rrbracket^{\mathsf{B}}$, respectively.
2. We set $\llbracket z' \rrbracket^{\mathsf{B}} = \llbracket x' \rrbracket^{\mathsf{B}} \oplus \llbracket y' \rrbracket^{\mathsf{B}}$.
3. We compute $\llbracket s \rrbracket^{\mathsf{B}} = \mathsf{AND}(\llbracket z' \rrbracket^{\mathsf{B}}, \llbracket y' \rrbracket^{\mathsf{B}})$ and $\llbracket t \rrbracket^{\mathsf{B}} = \mathsf{AND}(\llbracket z' \rrbracket^{\mathsf{B}}, \llbracket d' \rrbracket^{\mathsf{B}})$.
4. We compute $\llbracket cp \rrbracket^{\mathsf{B}} = \llbracket s \rrbracket^{\mathsf{B}} \oplus \llbracket t \rrbracket^{\mathsf{B}}$.

Here, $cp = 1$ if $x < y$ holds. We can execute MSB extraction using an overflow detection algorithm Overflow. We need to $O(\log n)$ communication rounds since we calculate the overflow from the lower bits. If we adopt not the shares over $\mathbb{Z}_{2^n}$ but $\mathbb{Z}_p$ (p: prime), we can construct constant-round Comparison via the different strategy [43], [44].

(3) Division

A division protocol $\mathsf{Division}(\llbracket N \rrbracket^{\mathsf{A}}, \llbracket D \rrbracket^{\mathsf{A}})$ outputs $\llbracket z \rrbracket^{\mathsf{A}}$, where $z = \lfloor N/D \rfloor$ (e.g., [40], [45]). We usually follow the strategy by Goldschmidt; that is, we set

$$\frac{N}{D} = \frac{NY_0Y_1\cdots}{DY_0Y_1\cdots}$$

and make the numerator $(DY_0Y_1\cdots)$ closer to 1 by choosing appropriate Y_i $(i = 0, 1, \cdots)$ to approximate the value of $\frac{N}{D}$ by the value of the denominator $(NY_0Y_1\cdots)$. We need very high computation costs and communication rounds for the above procedures. Moreover, we need to expand the field size from 2^n to $2^{n'}$ (with $n < n'$) in this protocol since we have to appropriately treat decimal numbers. When we use $n = 64$, for example, we have to set $n' = 206$.

(4) Share Conversion

The outputs of some protocols (e.g., Equality, Comparison) are boolean shares. When we would like to use them as the next inputs, we usually have to convert them into arithmetic shares. When we execute table lookup over shares, for example, the basic strategy is (1) converting index into one-hot vector using Equality and (2) computing a dot product. However, we need to execute a boolean to arithmetic conversion protocol after (1) since the outputs of Equality are boolean shares, and we cannot use them as inputs for an arithmetic dot product protocol. Share conversion protocols (boolean-to-arithmetic, arithmetic-to-boolean) have been actively studied [10], [40], [42], [46]–[49]. A basic strategy for boolean-to-arithmetic conversion protocols is considering boolean shares as arithmetic ones and adding

[†]This is the same for almost all the following protocols.

correction terms. We also have some taylor-made share conversion protocols [39], [42] for specific applications (e.g., ReLU function in neural networks).

(5) Other Protocols

There are many other efficient secure protocols. These are useful for privacy-preserving database operations and data analysis (in Sect. 3.2). More concretely, we have the protocols like table lookup [42], [50], the maximum/minimum value extraction [42], sorting [51]–[56], database join [57], [58], and floating point number computation [59]–[61]

3.2 Privacy-Preserving Data Analysis

There are many research results on privacy-preserving data analysis based on MPC.

(1) Machine Learning

There are many research results on privacy-preserving data mining (start from [62], [63]) and secure machine learning (e.g., [12], [39], [41], [64], [65]). Research targets have varied, and recently there are many results on privacy-preserving deep neural networks. The accuracy is lower than the training using plaintexts in most cases because of the following reasons:

- We usually approximate the non-linear functions (e.g., a sigmoid function, a softmax function) by polynomials because of the performance. If we use lower-degree polynomials (for making computation more efficient), the approximation gets rough and loses the accuracy of the machine learning model.
- Most of the previous research results adopt the small parameters (e.g., the number of layers in multilayer perceptron, the number of filters in convolutional neural networks) to complete the calculation within an acceptable time.

Recently, we have the results on fast neural network inferences using the specificity of learning models [66]. In binarized neural networks (neural networks that all parameters are restricted to the binary), for example, a heavy Max-pooling function can be replaced by a lightweight OR function since all the parameters in this model are binary. However, we know immoderate quantization of parameters leads to significant accuracy loss. To overcome this problem, the researcher on machine learning investigated the moderate quantization for the training and inference in neural networks [67]. Moreover, this scheme executes training/inference using not floating-point numbers but integers. This property is suitable for MPC, and the researcher on MPC immediately adopted this strategy and showed a more efficient and accurate construction of privacy-preserving neural networks [68]. Although the original motivation of quantization (in machine learning fields) was efficient hardware implementations, it also profited to the MPC research since a smaller range of plaintext leads to the more efficient execution of MPC.

(2) String Analysis

There are many papers on privacy-preserving edit distance computation. In this setting, each two-party has a string (e.g., genome string) and computes the edit distance between these two strings. There are many research results on approximate edit distance computation (e.g., [30], [69]) since we need costly dynamic programming for exact edit distance computation. We also have results on the extended edit distance computation like weighted edit distance and Needleman-Wunsch distance [70]. There also exist some results on string analysis other than edit distance computation. For example, we have some results on privacy-preserving full-text search [21], [71], [72]. All previous works on privacy-preserving full-text search are based on HE, and its performance is not enough for large-scale applications (e.g., human genome analysis) so far. We can also construct privacy-preserving text classification protocols [73], [74] and apply them to the tweet analysis and hate-speech detection. In this field, there is an example that we can feel the importance of considering specific algorithms of MPC. In the exact edit distance computation, we need to compute $\mathsf{min}(a+1, b+e, c+1)$. Here, a, b, and c are arbitrary integers and $e \in \{0, 1\}$. Although it requires high costs for computing the above function since $a+1$, $b+e$, and $c+1$ can be large integers. However, we can reduce the costs by considering $b + \mathsf{min}(a-b+1, e, c-b+1)$ since the conditions $a-b+1 \in \{0,1,2\}$ and $c-b+1 \in \{0,1,2\}$ hold from the property of edit distance computation. By using this algorithm, we can significantly improve the efficiency of privacy-preserving edit distance computation [23], [70]. Although such a trial is also meaningful in the standard computation, it is more critical in MPC.

(3) The Nearest Neighbor Search

A (k-)nearest neighbor search protocol is a basis for many applications (e.g., biometric matching, similar data search), and there are many results [10], [30], [75]–[79] in the field of MPC. Most of the protocols are based on HE or GC, and recently we can see some results based on SS. Similar to the case of edit distance computation, we can also the approximate nearest neighbor search [30], [79] for speeding up the execution.

3.3 Low-Cost Implementation and Social Experiments

In general, implementing fast MPC correctly and securely is difficult in practice. Therefore, only a few specialists can implement MPC now, and this can be a barrier to wide usage in the real world. To break this situation, there are some results on MPC compilers [48], [80], [81]. MPC compilers take a (standard) program code (= function) written by major programming languages (e.g., C, Python) or domain specific language as input, and output a program code for MPC of the same function. Some optimizations (e.g., making the circuit shallow as possible, convert loop to vector) are automatically applied. Such results are very important

for the social implementation of MPC.

A proof of concept is also important to proceed with the social implementation of MPC. There is a paper on the experiment by Boston University [82]. This experiment computed the disparity in salaries between companies while keeping the salary secret.

4. Conclusion

In this paper, we first explained the settings, security notions, and building blocks for MPC. Then, we showed current situations on applications and implementation of MPC. As recent advances in this field,

- we explained the case example that integration with other fields is progressing. An appropriate quantization has been studied in machine learning fields, and the researcher of privacy-preserving machine learning adopted this trend for making their model more efficient and accurate.
- we explained the importance of specific algorithms and appropriate approximation for MPC. In edit distance computation, by carefully devising the protocol, we can replace the costly minimum value extraction protocol by alternative lightweight calculation while keeping their outputs.
- we explained the efforts for improving usability (e.g., MPC compilers). Many organizations continue to develop and enhance their frameworks/compilers. We consider these are vital activities since requiring artisans of MPC in developing privacy-preserving applications can be a bottleneck of social implementations.

We consider the performance of MPC applications will continue to be improved by accelerating the above research trends. We would like to expect that MPC contributes a safer and more convenient society in the future.

References

[1] A.C. Yao, "How to generate and exchange secrets (extended abstract)," 27th Annual Symposium on Foundations of Computer Science, pp.162–167, Toronto, Canada, Oct. 1986.
[2] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," Proc. 19th Annual ACM Symposium on Theory of Computing, pp.218–229, New York, New York, USA, 1987.
[3] O. Goldreich, The Foundations of Cryptography - Volume 2, Basic Applications, Cambridge University Press, 2004.
[4] A. Shamir, "How to share a secret," Commun. ACM, vol.22, no.11, pp.612–613, 1979.
[5] I. Damgård and J.B. Nielsen, "Scalable and unconditionally secure multiparty computation," Proc. Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, pp.572–590, Santa Barbara, CA, USA, Aug. 2007.
[6] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," Proc, 2016 ACM SIGSAC Conference on Computer and Communications Security, pp.805–817, Vienna, Austria, Oct. 2016.
[7] T. Araki, A. Barak, J. Furukawa, T. Lichter, Y. Lindell, A. Nof, K. Ohara, A. Watzman, and O. Weinstein, "Optimized honest-majority MPC for malicious adversaries - breaking the 1 billion-gate per second barrier," 2017 IEEE Symposium on Security and Privacy, SP 2017, pp.843–862, San Jose, CA, USA, May 2017.
[8] K. Chida, D. Genkin, K. Hamada, D. Ikarashi, R. Kikuchi, Y. Lindell, and A. Nof, "Fast large-scale honest-majority MPC for malicious adversaries," Proc. Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Part III, pp.34–64, Santa Barbara, CA, USA, Aug. 2018.
[9] D. Beaver, "Efficient multiparty protocols using circuit randomization," Proc. Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, pp.420–432, Santa Barbara, California, USA, Aug. 1991.
[10] D. Demmler, T. Schneider, and M. Zohner, "ABY - A framework for efficient mixed-protocol secure two-party computation," 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, Feb. 2015.
[11] P. Mohassel, O. Orobets, and B. Riva, "Efficient server-aided 2pc for mobile phones," PoPETs, vol.2016, no.2, pp.82–99, 2016.
[12] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," 2017 IEEE Symposium on Security and Privacy, SP 2017, pp.19–38, San Jose, CA, USA, May 2017.
[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," Proc. Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, pp.223–238, May 1999.
[14] R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol.21, no.2, pp.120–126, 1978.
[15] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," Innovations in Theoretical Computer Science 2012, pp.309–325, Cambridge, MA, USA, Jan. 2012.
[16] N. Attrapadung, G. Hanaoka, S. Mitsunari, Y. Sakai, K. Shimizu, and T. Teruya, "Efficient two-level homomorphic encryption in prime-order bilinear groups and A fast implementation in webassembly," Proc. 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, pp.685–697, Incheon, Republic of Korea, June 2018.
[17] C. Gentry, "Fully homomorphic encryption using ideal lattices," Proc. 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp.169–178, Bethesda, MD, USA, May–June 2009.
[18] X. Liu, R.H. Deng, K.R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," IEEE Trans. Inf. Forensics Security, vol.11, no.11, pp.2401–2414, 2016.
[19] R. Bost, R.A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, Feb. 2015.
[20] R. Gilad-Bachrach, N. Dowlin, K. Laine, K.E. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," Proc. 33nd International Conference on Machine Learning, ICML 2016, pp.201–210, New York City, NY, USA, June 2016.
[21] K. Shimizu, K. Nuida, and G. Rätsch, "Efficient privacy-preserving string search and an application in genomics," Bioinformatics, vol.32, no.11, pp.1652–1661, 2016.
[22] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," 27th USENIX Security Symposium, USENIX Security 2018, pp.1651–1669, Baltimore, MD, USA, Aug. 2018.
[23] K. Nuida, S. Ohata, S. Mitsunari, and N. Attrapadung, "Arbitrary univariate function evaluation and re-encryption protocols over lifted-elgamal type ciphertexts," IACR Cryptology ePrint Archive, vol.2019, p.1233, 2019.

[24] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," Proc. 44th Symposium on Theory of Computing Conference, STOC 2012, pp.1219–1234, New York, NY, USA, May 2012.
[25] B. Applebaum, Y. Ishai, and E. Kushilevitz, "How to garble arithmetic circuits," SIAM J. Comput., vol.43, no.2, pp.905–929, 2014.
[26] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole - Reducing data transfer in garbled circuits using half gates," Proc. Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II, pp.220–250, Sofia, Bulgaria, April 2015.
[27] C. Kempka, R. Kikuchi, and K. Suzuki, "How to circumvent the two-ciphertext lower bound for linear garbling schemes," Proc. Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Part II, pp.967–997, Hanoi, Vietnam, Dec. 2016.
[28] A. Ben-Efraim, Y. Lindell, and E. Omri, "Optimizing semi-honest secure multiparty computation for the internet," Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security, pp.578–590, Vienna, Austria, Oct. 2016.
[29] M. Byali, A. Joseph, A. Patra, and D. Ravi, "Fast secure computation for small population over the internet," Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pp.677–694, Toronto, ON, Canada, Oct. 2018.
[30] G. Asharov, S. Halevi, Y. Lindell, and T. Rabin, "Privacy-preserving search of similar patients in genomic data," PoPETs, vol.2018, no.4, pp.104–124, 2018.
[31] B.D. Rouhani, M.S. Riazi, and F. Koushanfar, "DeepSecure: Scalable provably-secure deep learning," Proc. 55th Annual Design Automation Conference, DAC 2018, pp.2:1–2:6, San Francisco, CA, USA, June 2018.
[32] V. Costan, I.A. Lebedev, and S. Devadas, "Secure processors part I: Background, taxonomy for secure enclaves and intel SGX architecture," Foundations and Trends in Electronic Design Automation, vol.11, no.1-2, pp.1–248, 2017.
[33] V. Costan, I.A. Lebedev, and S. Devadas, "Secure processors part II: Intel SGX security analysis and MIT sanctum architecture," Foundations and Trends in Electronic Design Automation, vol.11, no.3, pp.249–361, 2017.
[34] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A. Sadeghi, G. Scerri, and B. Warinschi, "Secure multiparty computation from SGX," Financial Cryptography and Data Security - 21st International Conference, FC 2017, pp.477–497, Sliema, Malta, April 2017.
[35] B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "IRON: Functional encryption using intel SGX," Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, pp.765–782, Dallas, TX, USA, Oct.–Nov. 2017.
[36] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi, "Software grand exposure: SGX cache attacks are practical," 11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, Aug. 2017.
[37] S. Lee, M. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside SGX enclaves with branch shadowing," 26th USENIX Security Symposium, USENIX Security 2017, pp.557–574, Vancouver, BC, Canada, Aug. 2017.
[38] P. Subramanyan, R. Sinha, I.A. Lebedev, S. Devadas, and S.A. Seshia, "A formal foundation for secure remote execution of enclaves," Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, pp.2435–2450, Dallas, TX, USA, Oct.–Nov. 2017.
[39] P. Mohassel and P. Rindal, "ABY^3: A mixed protocol framework for machine learning," Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pp.35–52, Toronto, ON, Canada, Oct. 2018.
[40] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson, "High-performance secure multi-party computation for data mining applications," Int. J. Inf. Secur., vol.11, no.6, pp.403–418, 2012.
[41] M.S. Riazi, C. Weinert, O. Tkachenko, E.M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," Proc. 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, pp.707–721, Incheon, Republic of Korea, June 2018.
[42] S. Ohata and K. Nuida, "Communication-efficient (client-aided) secure two-party protocols and its application," International Conference on Financial Cryptography and Data Security, pp 369–385, July 2020.
[43] H. Morita, N. Attrapadung, T. Teruya, S. Ohata, K. Nuida, and G. Hanaoka, "Constant-round client-aided secure comparison protocol and its applications," Proc. Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Part II, pp.395–415, Barcelona, Spain, Sept. 2018.
[44] H. Morita, N. Attrapadung, T. Teruya, S. Ohata, K. Nuida, and G. Hanaoka, "Constant-round client-aided two-server secure comparison protocol and its applications," IEICE Trans. Fundamentals, vol.E103-A, no.1, pp.21–32, Jan. 2020.
[45] H. Morita, N. Attrapadung, S. Ohata, K. Nuida, S. Yamada, K. Shimizu, G. Hanaoka, and K. Asai, "Secure division protocol and applications to privacy-preserving chi-squared tests," International Symposium on Information Theory and Its Applications, ISITA 2018, pp.530–534, Singapore, Oct. 2018.
[46] R. Kikuchi, K. Chida, D. Ikarashi, W. Ogata, K. Hamada, and K. Takahashi, "Secret sharing with share-conversion: Achieving small share-size and extendibility to multiparty computation," IEICE Trans. Fundamentals, vol.E98-A, no.1, pp.213–222, Jan. 2015.
[47] R. Kikuchi, D. Ikarashi, K. Hamada, and K. Chida, "Adaptively and unconditionally secure conversion protocols between ramp and linear secret sharing," IEICE Trans. Fundamentals, vol.E98-A, no.1, pp.223–231, Jan. 2015.
[48] T. Araki, A. Barak, J. Furukawa, M. Keller, Y. Lindell, K. Ohara, and H. Tsuchida, "Generalizing the SPDZ compiler for other protocols," Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pp.880–895, Toronto, ON, Canada, Oct. 2018.
[49] R. Kikuchi, N. Attrapadung, K. Hamada, D. Ikarashi, A. Ishida, T. Matsuda, Y. Sakai, and J.C.N. Schuldt, "Field extension in secret-shared form and its applications to efficient secure computation," Proc. Information Security and Privacy - 24th Australasian Conference, ACISP 2019, pp.343–361, Christchurch, New Zealand, July 2019.
[50] G. Dessouky, F. Koushanfar, A. Sadeghi, T. Schneider, S. Zeitouni, and M. Zohner, "Pushing the communication barrier in secure computation using lookup tables," 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, Feb.–March, 2017.
[51] B. Zhang, "Generic constant-round oblivious sorting algorithm for MPC," Proc. Provable Security - 5th International Conference, ProvSec 2011, pp.240–256, Xi'an, China, Oct. 2011.
[52] K.V. Jónsson, G. Kreitz, and M. Uddin, "Secure multi-party sorting and applications," IACR Cryptology ePrint Archive, vol.2011, p.122, 2011.
[53] K. Hamada, R. Kikuchi, D. Ikarashi, K. Chida, and K. Takahashi, "Practically efficient multi-party sorting protocols from comparison sort algorithms," Information Security and Cryptology - ICISC 2012 - 15th International Conference, pp.202–216, Seoul, Korea, Nov. 2012.
[54] D. Bogdanov, S. Laur, and R. Talviste, "A practical analysis of oblivious sorting algorithms for secure multi-party computation," Proc. Secure IT Systems - 19th Nordic Conference, NordSec 2014, pp.59–74, Tromsø, Norway, Oct. 2014.
[55] K. Hamada, D. Ikarashi, K. Chida, and K. Takahashi, "Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation," IACR Cryptology ePrint Archive, vol.2014,

p.121, 2014.
[56] K. Chida, K. Hamada, D. Ikarashi, R. Kikuchi, N. Kiribuchi, and B. Pinkas, "An efficient secure three-party sorting protocol with an honest majority," IACR Cryptology ePrint Archive, vol.2019, p.695, 2019.
[57] S. Laur, R. Talviste, and J. Willemson, "From oblivious AES to efficient and secure database join in the multiparty setting," Proc. Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, pp.84–101, Banff, AB, Canada, June 2013.
[58] D. Bogdanov, L. Kamm, S. Laur, and V. Sokk, "Rmind: A tool for cryptographically secure statistical analysis," IEEE Trans. Dependable Sec. Comput., vol.15, no.3, pp.481–495, 2018.
[59] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele, "Secure computation on floating point numbers," 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, Feb. 2013.
[60] L. Kamm and J. Willemson, "Secure floating point arithmetic and private satellite collision analysis," Int. J. Inf. Secur., vol.14, no.6, pp.531–548, 2015.
[61] M. Aliasgari, M. Blanton, and F. Bayatbabolghani, "Secure computation of hidden Markov models and secure floating-point arithmetic in the malicious model," Int. J. Inf. Secur., vol.16, no.6, pp.577–601, 2017.
[62] R. Agrawal and R. Srikant, "Privacy-preserving data mining," Proc. 2000 ACM SIGMOD International Conference on Management of Data, pp.439–450, Dallas, Texas, USA, May 2000.
[63] Y. Lindell and B. Pinkas, "Privacy preserving data mining," Proc. Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, pp.36–54, Santa Barbara, California, USA, Aug. 2000.
[64] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, pp.619–631, Dallas, TX, USA, Oct.–Nov. 2017.
[65] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-party secure computation for neural network training," PoPETs, vol.2019, no.3, pp.26–49, 2019.
[66] M.S. Riazi, M. Samragh, H. Chen, K. Laine, K.E. Lauter, and F. Koushanfar, "XONN: XNOR-based oblivious deep neural network inference," 28th USENIX Security Symposium, USENIX Security 2019, pp.1501–1518, Santa Clara, CA, USA, Aug. 2019.
[67] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," 6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings, Vancouver, BC, Canada, April–May 2018.
[68] N. Agrawal, A.S. Shamsabadi, M.J. Kusner, and A. Gascón, "QUOTIENT: Two-party secure neural network training and prediction," Proc. 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, pp.1231–1247, London, UK, Nov. 2019.
[69] T. Schneider and O. Tkachenko, "EPISODE: Efficient privacy-preserving similar sequence queries on outsourced genomic databases," Proc. 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, pp.315–327, Auckland, New Zealand, July 2019.
[70] R. Zhu and Y. Huang, "Efficient privacy-preserving edit distance and beyond," IACR Cryptology ePrint Archive, vol.2017, p.683, 2017.
[71] Y. Yamamoto and M. Oguchi, "A decentralized system of genome secret search implemented with fully homomorphic encryption," 2017 IEEE International Conference on Smart Computing, SMARTCOMP 2017, pp.1–6, Hong Kong, China, May 2017.
[72] H. Sudo, K. Nuida, and K. Shimizu, "An efficient private evaluation of a decision graph," Information Security and Cryptology - ICISC 2018 - 21st International Conference, pp.143–160, Seoul, South Korea, Nov. 2018.
[73] G. Costantino, A.L. Marra, F. Martinelli, A. Saracino, and M. Sheikhalishahi, "Privacy-preserving text mining as a service," 2017 IEEE Symposium on Computers and Communications, ISCC 2017, pp.890–897, Heraklion, Greece, July 2017.
[74] D. Reich, A. Todoki, R. Dowsley, M.D. Cock, and A.C.A. Nascimento, "Privacy-preserving classification of personal text messages with secure multi-party computation," Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, pp.3752–3764, Vancouver, BC, Canada, Dec. 2019.
[75] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," Proc. Privacy Enhancing Technologies, 9th International Symposium, PETS 2009, pp.235–253, Seattle, WA, USA, Aug. 2009.
[76] A. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," Information, Security and Cryptology - ICISC 2009, 12th International Conference, pp.229–244, Seoul, Korea, Dec. 2009.
[77] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," Proc. Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, Feb. 2011.
[78] M. Barni, T. Bianchi, D. Catalano, M.D. Raimondo, R.D. Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva, "Privacy-preserving fingercode authentication," Multimedia and Security Workshop, MM&Sec 2010, pp.231–240, Roma, Italy, Sept. 2010.
[79] H. Chen, I. Chillotti, Y. Dong, O. Poburinnaya, I.P. Razenshteyn, and M.S. Riazi, "SANNS: Scaling up secure approximate k-nearest neighbors search," 29th Usenix Security Symposium, 2020.
[80] N. Büscher, D. Demmler, S. Katzenbeisser, D. Kretzmer, and T. Schneider, "HyCC: Compilation of hybrid protocols for practical secure computation," Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pp.847–861, Toronto, ON, Canada, Oct. 2018.
[81] Y. Li and W. Xu, "PrivPy: General and scalable privacy-preserving data mining," Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, pp.1299–1307, Anchorage, AK, USA, Aug. 2019.
[82] L. Qin, A. Lapets, F. Jansen, P. Flockhart, K.D. Albab, I. Globus-Harris, S. Roberts, and M. Varia, "From usability to secure computing and back again," Fifteenth Symposium on Usable Privacy and Security, SOUPS 2019, Santa Clara, CA, USA, Aug. 2019.

Satsuya Ohata received B.Eng. degree at Chiba University in 2011 and Ph.D. (Information Science and Technology) degree at The University of Tokyo in 2016. He is currently a postdoctoral researcher at the National Institute of Advanced Industrial Science and Technology (AIST). His research interests are practical cryptography and theoretical cybersecurity.